HTML

INTRODUCTION

HTML is the standard markup language for creating Web pages.

What is HTML?

- HTML stands for Hyper Text Markup Language
- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

A Simple HTML Document

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- The <!DOCTYPE html> declaration defines that this document is an HTML5 document
- The <html> element is the root element of an HTML page
- The <head> element contains meta information about the HTML page
- The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.

- The <h1> element defines a large heading
- The <p> element defines a paragraph

What is an HTML Element?

An HTML element is defined by a start tag, some content, and an end tag:

```
<tagname>Content goes here...</tagname>
```

The HTML **element** is everything from the start tag to the end tag:

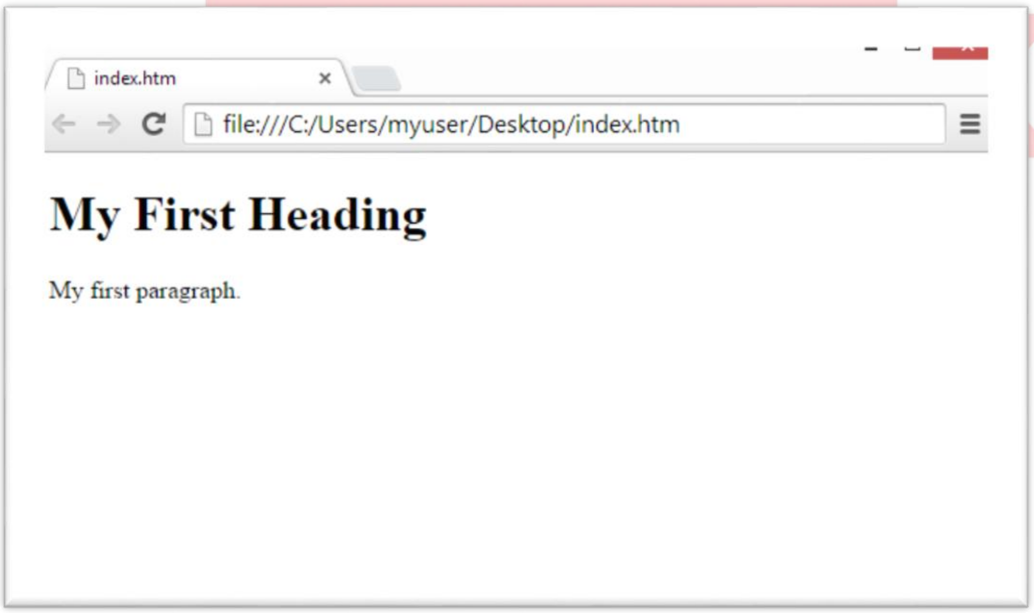
```
<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
<br>	*none*	*none*

Note: Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

Web Browsers

- ✓ The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to read HTML documents and display them correctly.
- ✓ A browser does not display the HTML tags, but uses them to determine how to display the document:



My First Heading

My first paragraph.

HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
 <head>
          <title>Page title</title>
 </head>
 <body>
          <h1>This is a heading</h1>
          <p>This is a paragraph. </p>
          <p>This is another paragraph. </p>
 </body>
</html>
```

Note: The content inside the <body> section (the white area above) will be displayed in a browser. The content inside the <title> element will be shown in the browser's title bar or in the page's tab.

HTML History

Since the early days of the World Wide Web, there have been many versions of HTML:

Year	**Version**
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	Recommendation: HTML 3.2
1999	Recommendation: HTML 4.01
2000	Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	Recommendation: HTML5
2016	Candidate Recommendation: HTML 5.1
2017	Recommendation: HTML5.1 2nd Edition
2017	Recommendation: HTML5.2

EDITORS

A simple text editor is all you need to learn HTML.

Learn HTML Using Notepad or Text Edit

- ✓ Web pages can be created and modified by using professional HTML editors.
- ✓ However, for learning HTML we recommend a simple text editor like Notepad (PC) or Text Edit (Mac).
- ✓ We believe in that using a simple text editor is a good way to learn HTML.
- ✓ Follow the steps below to create your first web page with Notepad or TextEdit.

Step 1: Open Notepad (PC)

Windows 8 or later:

Open the **Start Screen** (the window symbol at the bottom left on your screen).
Type **Notepad**.

Windows 7 or earlier:

Open **Start** > **Programs > Accessories > Notepad**

Step 1: Open Text Edit (Mac)

Open **Finder > Applications > Text dit**

Also change some preferences to get the application to save files correctly. In **Preferences > Format >** choose **"Plain Text"**

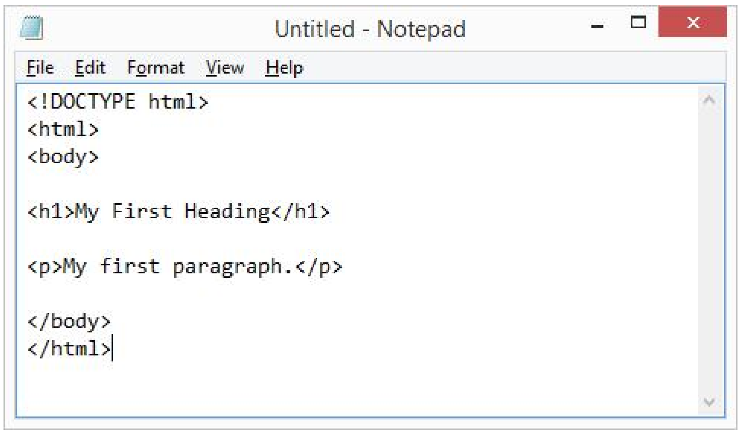
Then under "Open and Save", check the box that says "Display HTML files as HTML code instead of formatted text".

Then open a new document to place the code.

Step 2: Write Some HTML

Write or copy the following HTML code into Notepad:

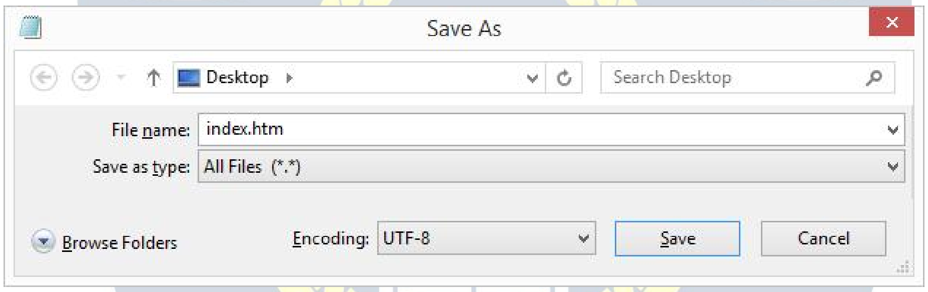
```
<! DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```



```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

Step 3: Save the HTML Page

- ✓ Save the file on your computer. Select **File > Save as** in the Notepad menu.
- ✓ Name the file **"index.htm"** and set the encoding to **UTF-8** (which is the preferred encoding for HTML files).

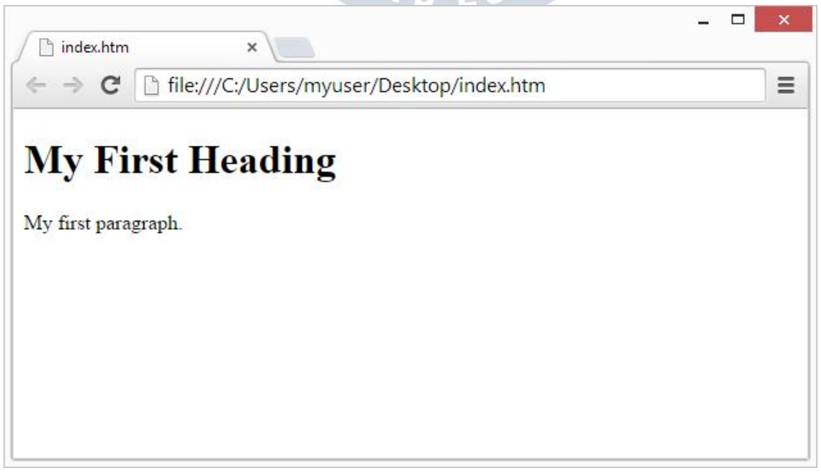


File name: index.htm
Save as type: All Files (*.*)
Encoding: UTF-8

Tip: You can use either .htm or .html as file extension. There is no difference, it is up to you.

Step 4: View the HTML Page in Your Browser

- ✓ Open the saved HTML file in your favorite browser (double click on the file, or right-click - and choose "Open with").
- ✓ The result will look much like this:



My First Heading

My first paragraph.

- ✓ You can edit the HTML code and view the result in your browser.
- ✓ It is the perfect tool when you want to **test** code fast. It also has color coding and the ability to save and share code with others:

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

BASIC EXAMPLES

- ✓ In this chapter we will show some basic HTML examples.
- ✓ Don't worry if we use tags you have not learned about yet.

HTML Documents

- ✓ All HTML documents must start with a document type declaration: <!DOCTYPE html>.
- ✓ The HTML document itself begins with <html> and ends with </html>.
- ✓ The visible part of the HTML document is between <body> and </body>.

Example:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

The <!DOCTYPE> Declaration

- ✓ The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.
- ✓ It must only appear once, at the top of the page (before any HTML tags).
- ✓ The <!DOCTYPE> declaration is not case sensitive.
- ✓ The <!DOCTYPE> declaration for HTML5 is:

```
<!DOCTYPE html>
```

HTML Headings

- ✓ HTML headings are defined with the <h1> to <h6> tags.
- ✓ <h1> defines the most important heading. <h6> defines the least important heading:

Example:

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
```

HTML Paragraphs

HTML paragraphs are defined with the <p> tag:

Example:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML Links

HTML links are defined with the <a> tag:

Example:

```
<a href="https://www.nexstep.com">This is a link</a>
```

- ✓ The link's destination is specified in the href attribute.
- ✓ Attributes are used to provide additional information about HTML elements.
- ✓ You will learn more about attributes in a later chapter.

HTML Images

- ✓ HTML images are defined with the <img> tag.
- ✓ The source file (src), alternative text (alt), width, and height are provided as attributes:

Example:

```
<img src="ditrp.jpg" alt="ditrp.com" width="104" height="142">
```

How to View HTML Source?

Have you ever seen a Web page and wondered "Hey! How did they do that?"

View HTML Source Code:

Right-click in an HTML page and select "View Page Source" (in Chrome) or "View Source" (in Edge), or similar in other browsers. This will open a window containing the HTML source code of the page.

Inspect an HTML Element:

Right-click on an element (or a blank area), and choose "Inspect" or "Inspect Element" to see what elements are made up of (you will see both the HTML and the CSS). You can also edit the HTML or CSS on-the-fly in the Elements or Styles panel that opens.

ELEMENTS

An HTML element is defined by a start tag, some content, and an end tag.

HTML Elements

The HTML **element** is everything from the start tag to the end tag:

```
<tagname>Content goes here...</tagname>
```

Examples of some HTML elements:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

Start tag	Element content	End tag
<h1>	My First Heading	</h1>
<p>	My first paragraph.	</p>
<br>	*none*	*none*

Note: Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

Nested HTML Elements

- ✓ HTML elements can be nested (this means that elements can contain other elements).
- ✓ All HTML documents consist of nested HTML elements.
- ✓ The following example contains four HTML elements (<html>, <body>, <h1> and <p>):

Example:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Example Explained

- ✓ The <html> element is the root element and it defines the whole HTML document.
- ✓ It has a start tag <html> and an end tag </html>.
- ✓ Then, inside the <html> element there is a <body> element:

```
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
```

- ✓ The <body> element defines the document's body.
- ✓ It has a start tag <body> and an end tag </body>.
- ✓ Then, inside the <body> element there are two other elements: <h1> and <p>:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

- ✓ The <h1> element defines a heading.
- ✓ It has a start tag <h1> and an end tag </h1>:

```
<h1>My First Heading</h1>
```

The <p> element defines a paragraph.
It has a start tag <p> and an end tag </p>:

```
<p>My first paragraph.</p>
```

Never Skip the End Tag

Some HTML elements will display correctly, even if you forget the end tag:
Example:

```
<html>
<body>

<p>This is a paragraph
<p>This is a paragraph

</body>
</html>
```

However, never rely on this! Unexpected results and errors may occur if you forget the end tag!

Empty HTML Elements

- ✓ HTML elements with no content are called empty elements.
- ✓ The <br> tag defines a line break, and is an empty element without a closing tag:

Example:

```
<p>This is a <br> paragraph with a line break.</p>
```

HTML is Not Case Sensitive

- ✓ HTML tags are not case sensitive: <P> means the same as <p>.
- ✓ The HTML standard does not require lowercase tags, but we **recommends** lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

HTML Tag Reference

Tag reference contains additional information about these tags and their attributes.

Tag	**Description**
<html>	Defines the root of an HTML document
<body>	Defines the document's body
<h1> to <h6>	Defines HTML headings

For a complete list of all available HTML tags, visit our HTML Tag Reference.

ATTRIBUTES

HTML attributes provide additional information about HTML elements.

HTML Attributes

- All HTML elements can have **attributes**

- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**
- Attributes usually come in name/value pairs like: **name="value"**

The href Attribute

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to:

Example:
```
<a href="https://www.w3schools.com">Visit W3Schools</a>
```

The src Attribute

The <img> tag is used to embed an image in an HTML page. The src attribute specifies the path to the image to be displayed:

Example:
```
<img src="img_girl.jpg">
```

There are two ways to specify the URL in the src attribute:

1. Absolute URL - Links to an external image that is hosted on another website. Example: src="https://www.w3schools.com/images/img_girl.jpg".

Notes: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

2. Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. Example: src="img_girl.jpg". If the URL begins with a slash, it will be relative to the domain. Example: src="/images/img_girl.jpg".

Tip: It is almost always best to use relative URLs. They will not break if you change domain.

The width and height Attributes

The <img> tag should also contain the width and height attributes, which specifies the width and height of the image (in pixels):

Example:
```
<img src="img_girl.jpg" width="500" height="600">
```

The alt Attribute

The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to slow connection, or an error in the src attribute, or if the user uses a screen reader.

Example:

```
<img src="img_girl.jpg" alt="Girl with a jacket">
```

Example:

See what happens if we try to display an image that does not exist:

```
<img src="img_typo.jpg" alt="Girl with a jacket">
```

The style Attribute

The style attribute is used to add styles to an element, such as color, font, size, and more.

Example:

```
<p style="color:red;">This is a red paragraph.</p>
```

The lang Attribute

- ✓ You should always include the lang attribute inside the <html> tag, to declare the language of the Web page. This is meant to assist search engines and browsers.
- ✓ The following example specifies English as the language:

```
<!DOCTYPE html>
<html lang="en">
<body>
...
</body>
</html>
```

- ✓ Country codes can also be added to the language code in the lang attribute. So, the first two characters define the language of the HTML page, and the last two characters define the country.
- ✓ The following example specifies English as the language and United States as the country:

```
<!DOCTYPE html>
<html lang="en-US">
<body>
...
</body>
</html>
```

The title Attribute

- ✓ The title attribute defines some extra information about an element.
- ✓ The value of the title attribute will be displayed as a tooltip when you mouse over the element:

Example:

```
<p title="I'm a tooltip">This is a paragraph.</p>
```

We Suggest: Always Use Lowercase Attributes

- ✓ The HTML standard does not require lowercase attribute names.
- ✓ The title attribute (and all other attributes) can be written with uppercase or lowercase like **title** or **TITLE**.
- ✓ However, **recommends** lowercase attributes in HTML, and **demands** lowercase attributes for stricter document types like XHTML.
- ✓ We always use lowercase attribute names.

We Suggest: Always Quote Attribute Values

- ✓ The HTML standard does not require quotes around attribute values.
- ✓ However, **recommends** quotes in HTML, and **demands** quotes for stricter document types like XHTML.

Good:

```
<a href="https://DITRP.com/html/">Visit our HTML tutorial</a>
```

Bad:

```
<a href=https://www.DITRP.com/html/>Visit our HTML tutorial</a>
```

Sometimes you have to use quotes. This example will not display the title attribute correctly, because it contains a space:

Example:

```
<p title=About DITRP>
```

We always use quotes around attribute values.

Single or Double Quotes?

- ✓ Double quotes around attribute values are the most common in HTML, but single quotes can also be used.
- ✓ In some situations, when the attribute value itself contains double quotes, it is necessary to use single quotes:

```
<p title='John "ShotGun" Nelson'>
```

Or vice versa:

```
<p title="John 'ShotGun' Nelson">
```

HEADINGS

HTML headings are titles or subtitles that you want to display on a webpage.
Example:

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

HTML Headings

- ✓ HTML headings are defined with the <h1> to <h6> tags.
- ✓ <h1> defines the most important heading. <h6> defines the least important heading.

Example:

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Note: Browsers automatically add some white space (a margin) before and after a heading.

Headings Are Important

- ✓ Search engines use the headings to index the structure and content of your web pages.
- ✓ Users often skim a page by its headings. It is important to use headings to show the document structure.
- ✓ <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

Note: Use HTML headings for headings only. Don't use headings to make text **BIG** or **bold**.

Bigger Headings

Each HTML heading has a default size. However, you can specify the size for any heading with the style attribute, using the CSS font-size property:
Example:

```
<h1 style="font-size:60px;">Heading 1</h1>
```

PARAGRAPHS

A paragraph always starts on a new line, and is usually a block of text.

HTML Paragraphs

- ✓ The HTML <p> element defines a paragraph.
- ✓ A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

Example:

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML Display

- ✓ You cannot be sure how HTML will be displayed.
- ✓ Large or small screens, and resized windows will create different results.
- ✓ With HTML, you cannot change the display by adding extra spaces or extra lines in your HTML code.
- ✓ The browser will automatically remove any extra spaces and lines when the page is displayed:

Example:

```
<p>
This paragraph
contains a lot of lines
in the source code,
but the browser
ignores it.
</p>

<p>
This paragraph
contains         a lot of spaces
in the source         code,
but the        browser
ignores it.
</p>
```

HTML Horizontal Rules

- ✓ The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.
- ✓ The <hr> element is used to separate content (or define a change) in an HTML page:

Example:

```
<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
```

The <hr> tag is an empty tag, which means that it has no end tag.

HTML Line Breaks

- ✓ The HTML <br> element defines a line break.
- ✓ Use <br> if you want a line break (a new line) without starting a new paragraph:

Example:

```
<p>This is<br>a paragraph<br>with line breaks.</p>
```

The <br> tag is an empty tag, which means that it has no end tag.

The Poem Problem

This poem will display on a single line:

Example:

```
<p>
  My Bonnie lies over the ocean.

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.

  Oh, bring back my Bonnie to me.
</p>
```

Solution - The HTML <pre> Element

- ✓ The HTML <pre> element defines preformatted text.
- ✓ The text inside a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks:

Example:
```
<pre>
  My Bonnie lies over the ocean.

  My Bonnie lies over the sea.

  My Bonnie lies over the ocean.

  Oh, bring back my Bonnie to me.
</pre>
```

STYLES

The HTML style attribute is used to add styles to an element, such as color, font, size, and more.

Example:
I am Red
I am Blue
I am Big

The HTML Style Attribute

- ✓ Setting the style of an HTML element, can be done with the style attribute.
- ✓ The HTML style attribute has the following syntax:

```
<tagname style="property:value;">
```

The ***property*** is a CSS property. The ***value*** is a CSS value.

Background Color

The CSS background-color property defines the background color for an HTML element.

Example:
Set the background color for a page to powderblue:

```
<body style="background-color:powderblue;">

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
```

Example:
Set background color for two different elements:

```
<body>

<h1 style="background-color:powderblue;">This is a heading</h1>
<p style="background-color:tomato;">This is a paragraph.</p>

</body>
```

Text Color

The CSS color property defines the text color for an HTML element:

Example:

```
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
```

Fonts

The CSS font-family property defines the font to be used for an HTML element:

Example:

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
```

Text Size

The CSS font-size property defines the text size for an HTML element:

Example:

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

Text Alignment

The CSS text-align property defines the horizontal text alignment for an HTML element:

Example:

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

TEXT FORMATTING

HTML contains several elements for defining text with a special meaning.

Example:

This text is bold

This text is italic

This is $_{\text{subscript}}$ and $^{\text{superscript}}$

HTML Formatting Elements

Formatting elements were designed to display special types of text:

- <b> - Bold text
- <strong> - Important text
- <i> - Italic text
- <em> - Emphasized text
- <mark> - Marked text
- <small> - Smaller text
- <del> - Deleted text
- <ins> - Inserted text
- <sub> - Subscript text
- <sup> - Superscript text

HTML <b> and <strong> Elements

The HTML <b> element defines bold text, without any extra importance.

Example:

```
<b>This text is bold</b>
```

The HTML <strong> element defines text with strong importance. The content inside is typically displayed in bold.

Example:

```
<strong>This text is important!</strong>
```

HTML <i> and <em> Elements

✓ The HTML <i> element defines a part of text in an alternate voice or mood. The content inside is typically displayed in italic.

✓ **Tip:** The <i> tag is often used to indicate a technical term, a phrase from another language, a thought, a ship name, etc.

Example:

```
<i>This text is italic</i>
```

- ✓ The HTML <em> element defines emphasized text. The content inside is typically displayed in italic.
- ✓ **Tip:** A screen reader will pronounce the words in <em> with an emphasis, using verbal stress.

Example:

```
<em>This text is emphasized</em>
```

HTML <small> Element

The HTML <small> element defines smaller text:

Example:

```
<small>This is some smaller text.</small>
```

HTML <mark> Element

The HTML <mark> element defines text that should be marked or highlighted:

Example:

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

HTML <del> Element

The HTML <del> element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

Example:

```
<p>My favorite color is <del>blue</del> red.</p>
```

HTML <ins> Element

The HTML <ins> element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

Example:

```
<p>My favorite color is <del>blue</del> <ins>red</ins>.</p>
```

HTML <sub> Element

The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O:

Example:

```
<p>This is <sub>subscripted</sub> text.</p>
```

HTML <sup> Element

The HTML <sup> element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font. Superscript text can be used for footnotes, like WWW[1]:

Example:

```
<p>This is <sup>superscripted</sup> text.</p>
```

QUOTATION AND CITATION ELEMENTS

In this chapter we will go through the <blockquote>,<q>, <abbr>, <address>, <cite>, and <bdo> HTML elements.

Example:

Here is a quote from WWF's website:

For nearly 60 years, WWF has been protecting the future of nature. The world's leading conservation organization, WWF works in 100 countries and is supported by more than one million members in the United States and close to five million globally.

HTML <blockquote> for Quotations

- ✓ The HTML <blockquote> element defines a section that is quoted from another source.
- ✓ Browsers usually indent <blockquote> elements.

Example:

```
<p>Here is a quote from WWF's website:</p>
<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 50 years, WWF has been protecting the future of nature.
The world's leading conservation organization,
WWF works in 100 countries and is supported by
1.2 million members in the United States and
close to 5 million globally.
</blockquote>
```

HTML <q> for Short Quotations

- ✓ The HTML <q> tag defines a short quotation.
- ✓ Browsers normally insert quotation marks around the quotation.

Example:

```
<p>WWF's goal is to: <q>Build a future where people live in harmony with
nature.</q></p>
```

HTML <abbr> for Abbreviations

- ✓ The HTML <abbr> tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".
- ✓ Marking abbreviations can give useful information to browsers, translation systems and search-engines.
- ✓ **Tip:** Use the global title attribute to show the description for the abbreviation/acronym when you mouse over the element.

Example:

```
<p>The <abbr title="World Health Organization">WHO</abbr> was founded in
1948.</p>
```

HTML <address> for Contact Information

- ✓ The HTML <address> tag defines the contact information for the author/owner of a document or an article.
- ✓ The contact information can be an email address, URL, physical address, phone number, social media handle, etc.
- ✓ The text in the <address> element usually renders in *italic,* and browsers will always add a line break before and after the <address> element.

Example:

```
<address>
Written by John Doe.<br>
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```

HTML <cite> for Work Title

- ✓ The HTML <cite> tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).
- ✓ **Note:** A person's name is not the title of a work.
- ✓ The text in the <cite> element usually renders in *italic*.

Example:

```
<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>
```

HTML <bdo> for Bi-Directional Override

- ✓ BDO stands for Bi-Directional Override.
- ✓ The HTML <bdo> tag is used to override the current text direction:

Example:

```
<bdo dir="rtl">This text will be written from right to left</bdo>
```

COMMENTS

HTML comments are not displayed in the browser, but they can help document your HTML source code.

HTML Comment Tags

You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.

Note: Comments are not displayed by the browser, but they can help document your HTML source code.

With comments you can place notifications and reminders in your HTML code:

Example:

```
<!-- This is a comment -->

<p>This is a paragraph.</p>

<!-- Remember to add more information here -->
```

Comments are also great for debugging HTML, because you can comment out HTML lines of code, one at a time, to search for errors:

Example:

```
<!-- Do not display this image at the moment
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
```

COLORS

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

Color Names

In HTML, a color can be specified by using a color name:

Tomato
Orange
DodgerBlue
MediumSeaGreen
Gray
SlateBlue
Violet
LightGray

HTML supports 140 standard color names.

Background Color

You can set the background color for HTML elements:

Hello World

Lorem *ipsum* dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

Text Color

You can set the color of text:

Hello World

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Example:

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example:

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid Dodger Blue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

In HTML, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

The following three <div> elements have their background color set with RGB, HEX, and HSL values:

rgb(255, 99, 71)

#ff6347

hsl(9, 100%, 64%)

The following two <div> elements have their background color set with RGBA and HSLA values, which adds an Alpha channel to the color (here we have 50% transparency):

rgba(255, 99, 71, 0.5)

hsla(9, 100%, 64%, 0.5)

Example:

```
<h1 style="background-color:rgb(255, 99, 71);">...</h1>
<h1 style="background-color:#ff6347;">...</h1>
<h1 style="background-color:hsl(9, 100%, 64%);">...</h1>

<h1 style="background-color:rgba(255, 99, 71, 0.5);">...</h1>
<h1 style="background-color:hsla(9, 100%, 64%, 0.5);">...</h1>
```

RGB and RGBA Colors

- ✓ An RGB color value represents RED, GREEN, and BLUE light sources.
- ✓ An RGBA color value is an extension of RGB with an Alpha channel (opacity).

RGB Color Values

In HTML, a color can be specified as an RGB value, using this formula:

rgb **(red, green, blue*)***

Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.

This means that there are 256 x 256 x 256 = 16777216 possible colors!

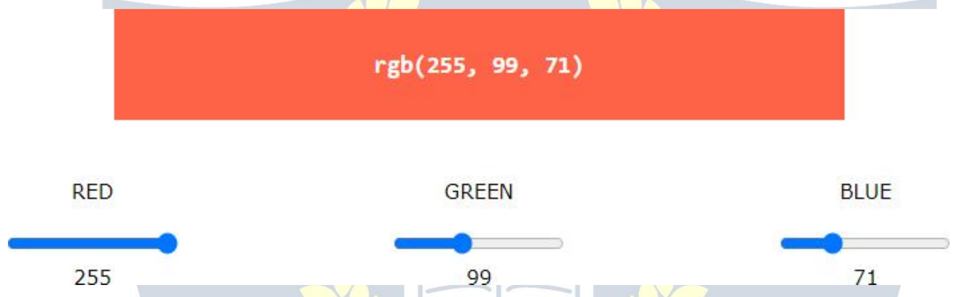
For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255), and the other two (green and blue) are set to 0.

Another example, rgb(0, 255, 0) is displayed as green, because green is set to its highest value (255), and the other two (red and blue) are set to 0.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

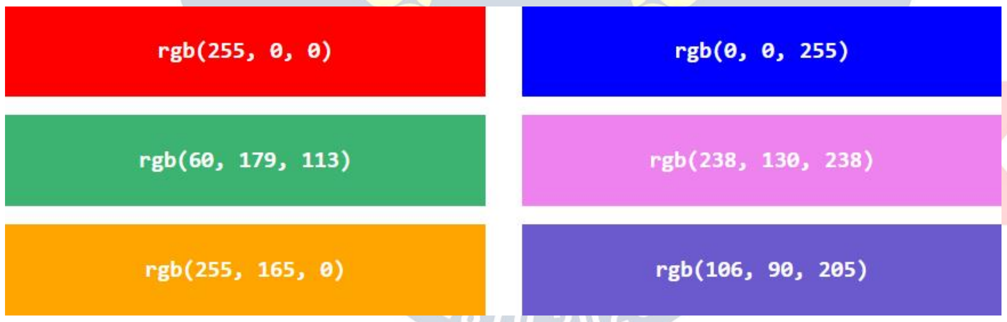
To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:



```
rgb(255, 99, 71)
```

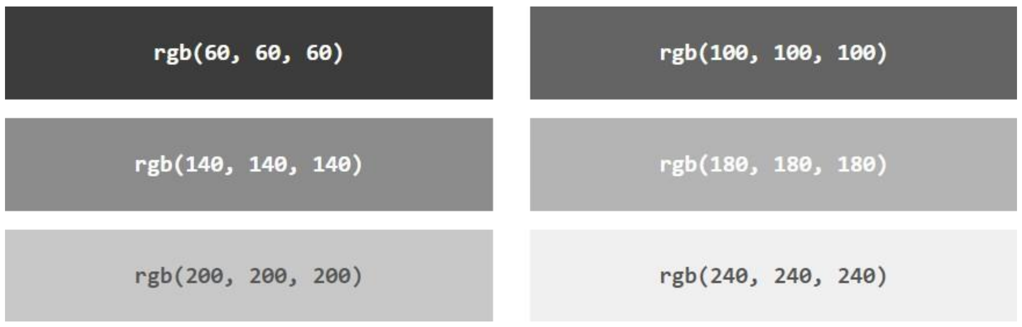
Example:



```
rgb(255, 0, 0)
rgb(0, 0, 255)
rgb(60, 179, 113)
rgb(238, 130, 238)
rgb(255, 165, 0)
rgb(106, 90, 205)
```

Shades of Gray

Shades of gray are often defined using equal values for all three parameters:
Example:



```
rgb(60, 60, 60)
rgb(100, 100, 100)
rgb(140, 140, 140)
rgb(180, 180, 180)
rgb(200, 200, 200)
rgb(240, 240, 240)
```

RGBA Color Values

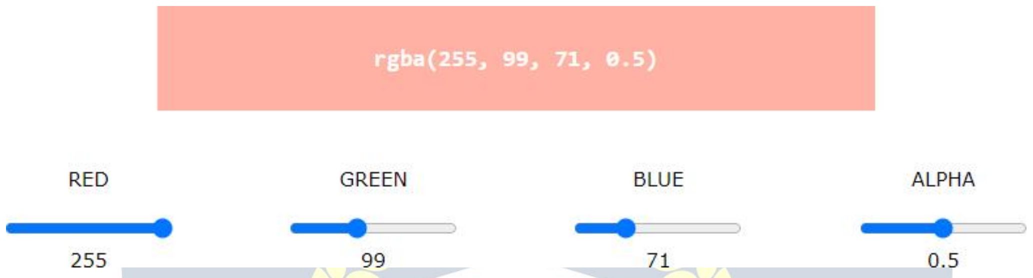
RGBA color values are an extension of RGB color values with an Alpha channel - which specifies the opacity for a color.

An RGBA color value is specified with:

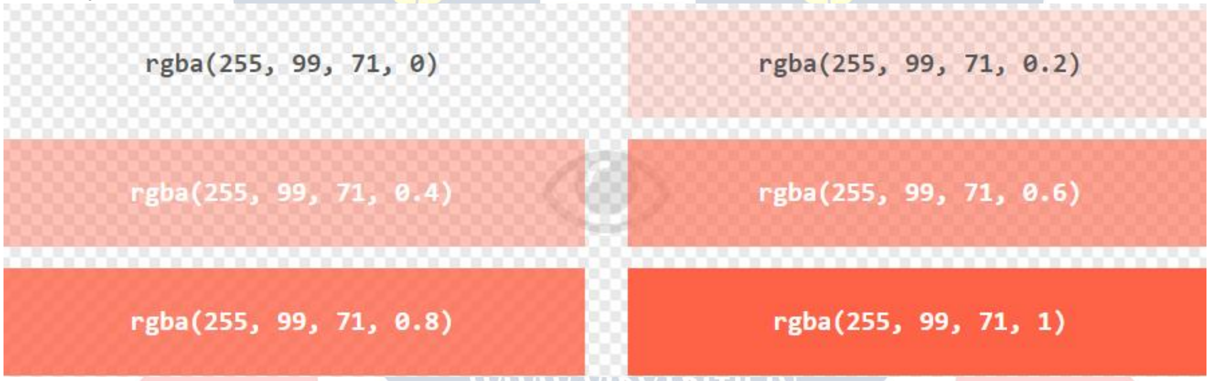
rgba(red, green, blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Experiment by mixing the RGBA values below:



Example:



HEX Colors

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

HEX Color Values

In HTML, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

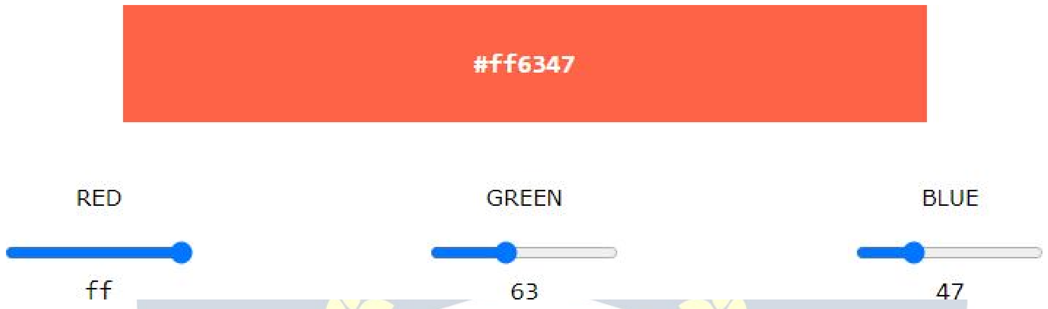
For example, #ff0000 is displayed as red, because red is set to its highest value (ff), and the other two (green and blue) are set to 00.

Another example, #00ff00 is displayed as green, because green is set to its highest value (ff), and the other two (red and blue) are set to 00.

To display black, set all color parameters to 00, like this: #000000.

To display white, set all color parameters to ff, like this: #ffffff.

Experiment by mixing the HEX values below:

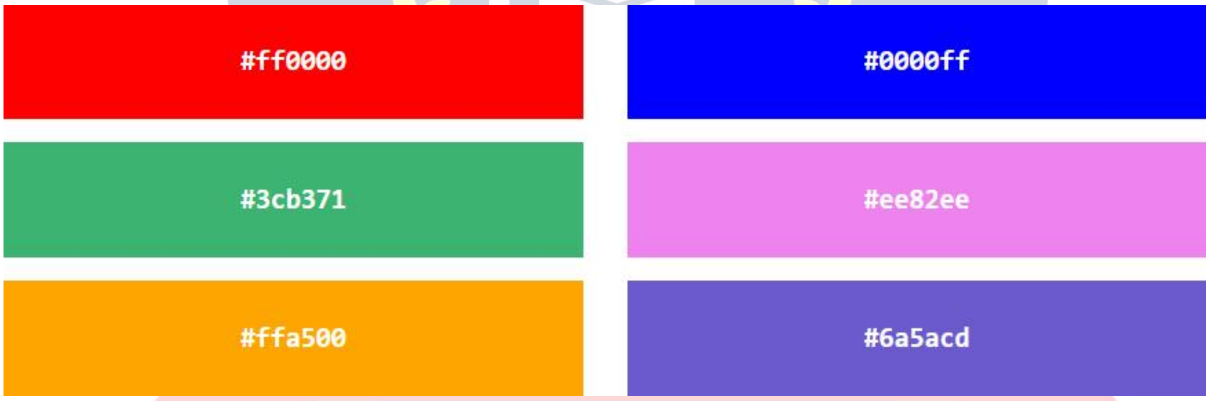

#ff6347

RED ff

GREEN 63

BLUE 47

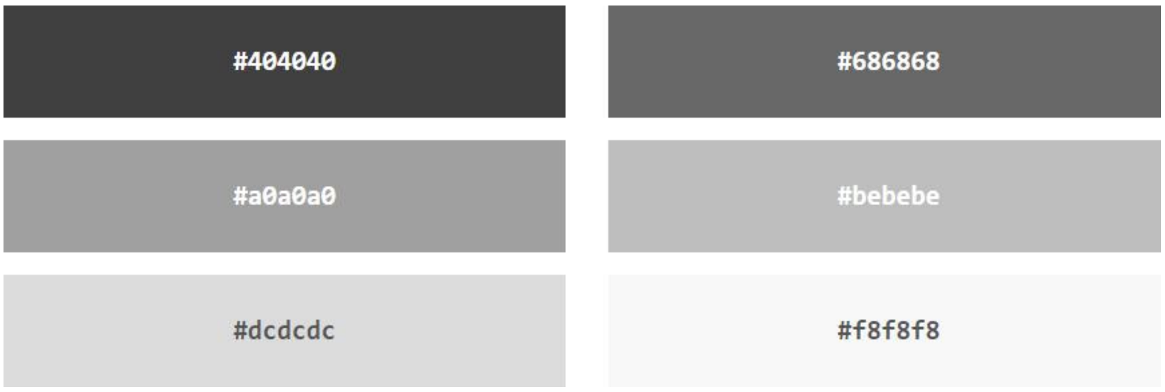
Example:



Shades of Gray

Shades of gray are often defined using equal values for all three parameters:

Example:



HSL and HSLA Colors

HSL stands for hue, saturation, and lightness.
HSLA color values are an extension of HSL with an Alpha channel (opacity).

HSL Color Values

In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:

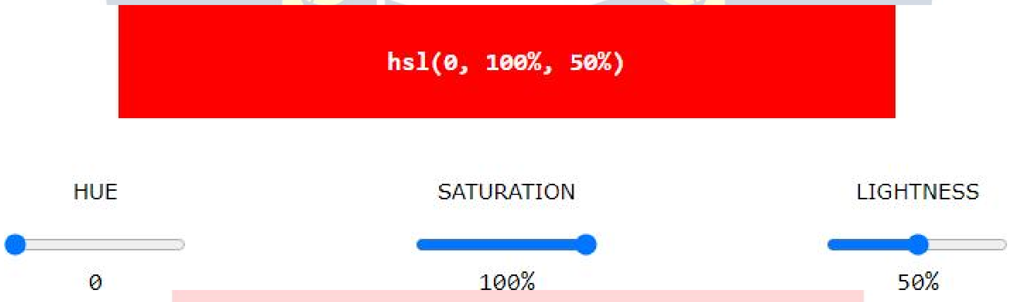
hsl (*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage value, 0% is black, and 100% is white.

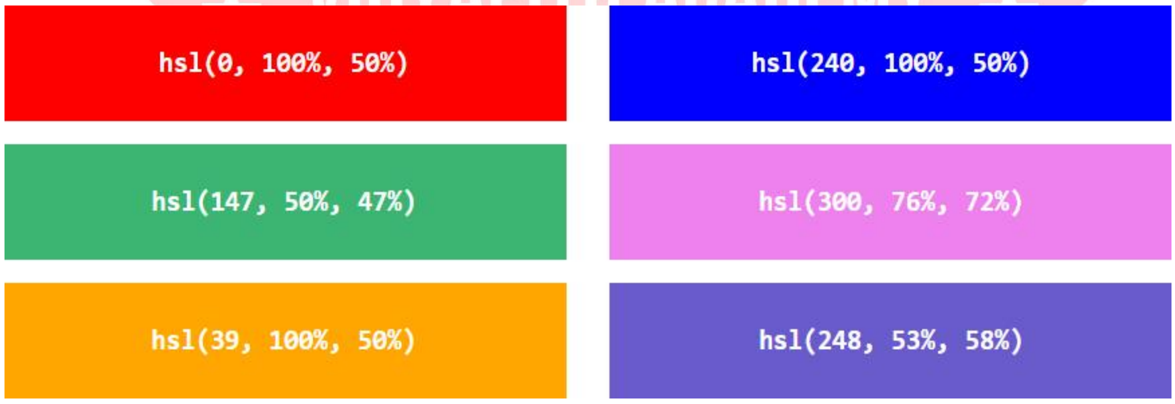
Experiment by mixing the HSL values below:



```
hsl(0, 100%, 50%)
```

HUE	SATURATION	LIGHTNESS
0	100%	50%

Example:



```
hsl(0, 100%, 50%)      hsl(240, 100%, 50%)
hsl(147, 50%, 47%)     hsl(300, 76%, 72%)
hsl(39, 100%, 50%)     hsl(248, 53%, 58%)
```

Saturation

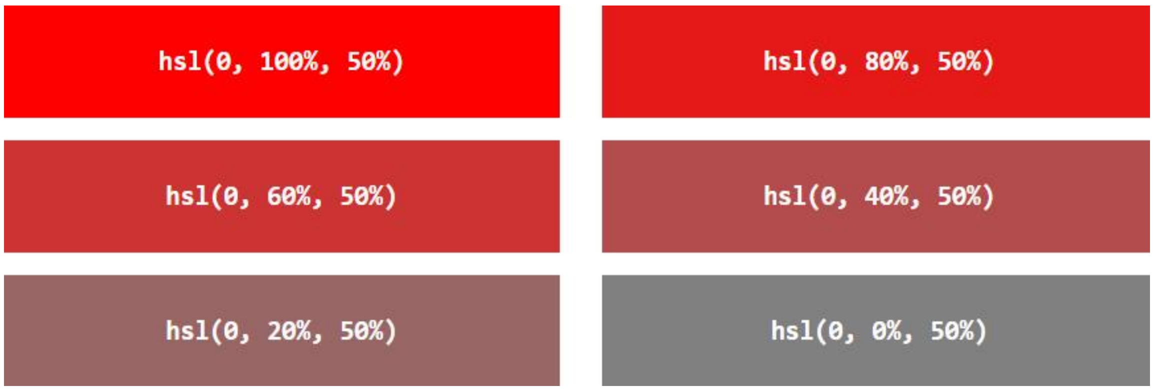
Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

Example:

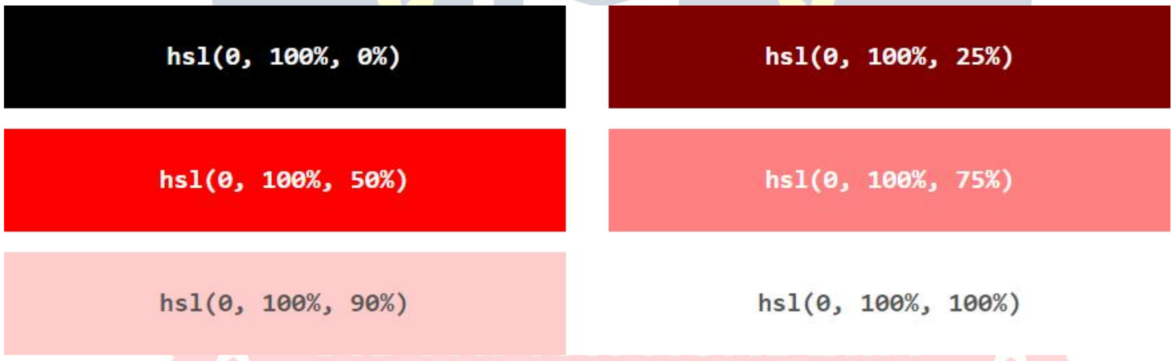


hsl(0, 100%, 50%)	hsl(0, 80%, 50%)
hsl(0, 60%, 50%)	hsl(0, 40%, 50%)
hsl(0, 20%, 50%)	hsl(0, 0%, 50%)

Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

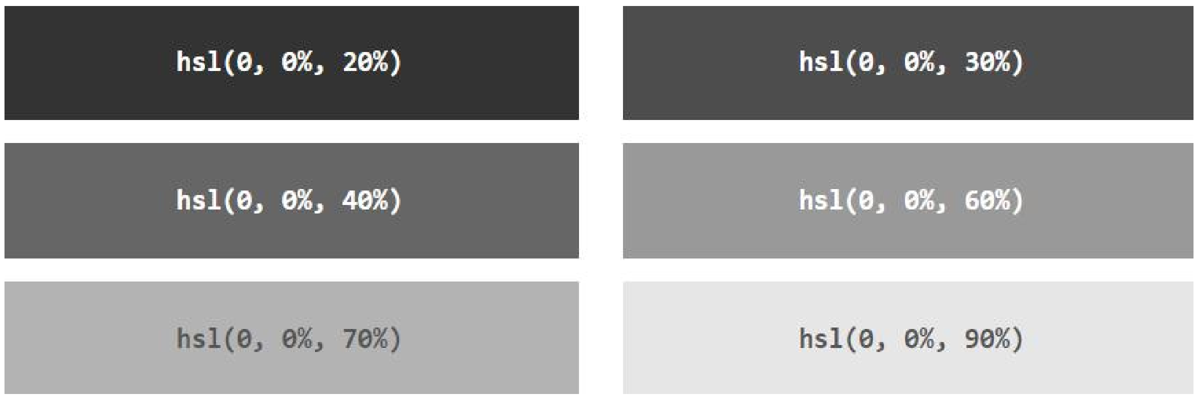
Example:



hsl(0, 100%, 0%)	hsl(0, 100%, 25%)
hsl(0, 100%, 50%)	hsl(0, 100%, 75%)
hsl(0, 100%, 90%)	hsl(0, 100%, 100%)

Shades of Gray

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades: Example:



hsl(0, 0%, 20%)	hsl(0, 0%, 30%)
hsl(0, 0%, 40%)	hsl(0, 0%, 60%)
hsl(0, 0%, 70%)	hsl(0, 0%, 90%)

HSLA Color Values

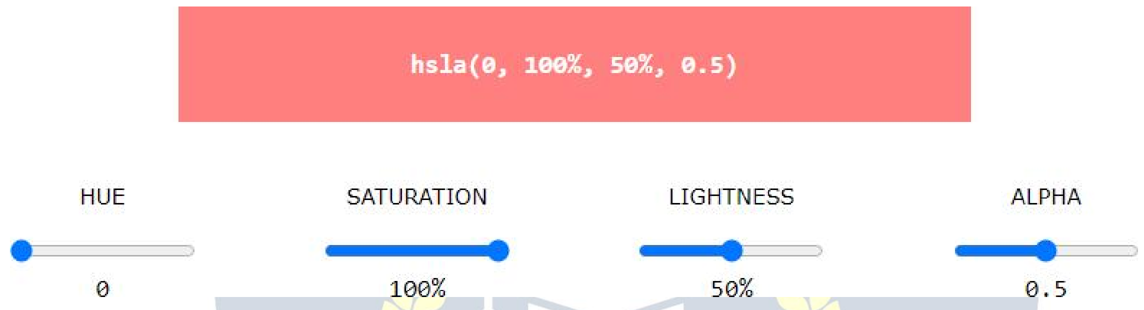
HSLA color values are an extension of HSL color values with an Alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

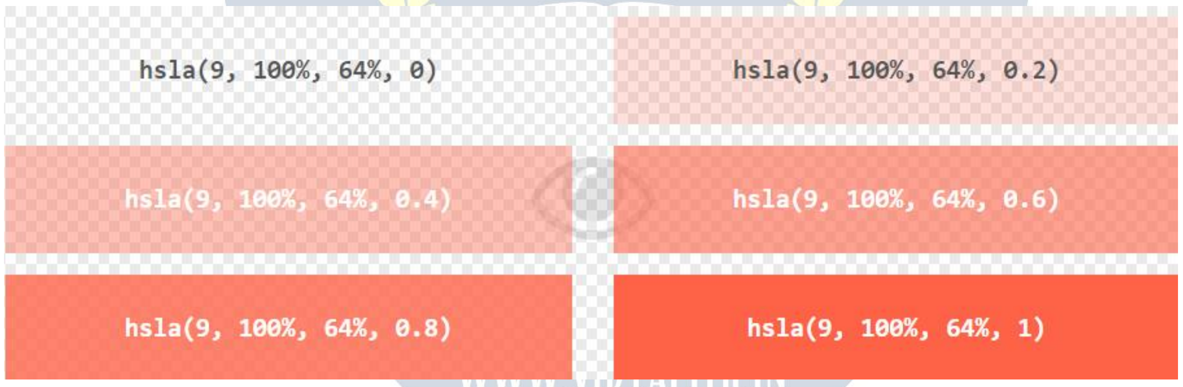
hsla (*hue, saturation*, *lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Experiment by mixing the HSLA values below:



Example:



LINKS

Links are found in nearly all web pages. Links allow users to click their way from page to page.

HTML Links - Hyperlinks

- ✓ HTML links are hyperlinks.
- ✓ You can click on a link and jump to another document.
- ✓ When you move the mouse over a link, the mouse arrow will turn into a little hand.

Note: A link does not have to be text. A link can be an image or any other HTML element!

HTML Links - Syntax

The HTML <a> tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

- ✓ The most important attribute of the <a> element is the href attribute, which indicates the link's destination.
- ✓ The *link text* is the part that will be visible to the reader.
- ✓ Clicking on the link text, will send the reader to the specified URL address.

Example:

This example shows how to create a link to W3Schools.com:

```
<a href="https://www.w3schools.com/">Visit W3Schools.com!</a>
```

By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

Tip: Links can of course be styled with CSS, to get another look!

HTML Links - The target Attribute

- ✓ By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.
- ✓ The target attribute specifies where to open the linked document.
- ✓ The target attribute can have one of the following values:
  - _self - Default. Opens the document in the same window/tab as it was clicked
  - _blank - Opens the document in a new window or tab
  - _parent - Opens the document in the parent frame
  - _top - Opens the document in the full body of the window

Example:

Use target="_blank" to open the linked document in a new browser window or tab:

```
<a href="https://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

Absolute URLs vs. Relative URLs

- ✓ Both examples above are using an **absolute URL** (a full web address) in the href attribute.
- ✓ A local link (a link to a page within the same website) is specified with a **relative URL** (without the "https://www" part):

Example:

```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Links - Use an Image as a Link

To use an image as a link, just put the <img> tag inside the <a> tag:

Example:

```
<a href="default.asp">
<img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">
</a>
```

Link to an Email Address

Use mailto: inside the href attribute to create a link that opens the user's email program (to let them send a new email):

Example:

```
<a href="mailto:someone@example.com">Send email</a>
```

Button as a Link

To use an HTML button as a link, you have to add some JavaScript code.

JavaScript allows you to specify what happens at certain events, such as a click of a button:

Example:

```
<button onclick="document.location='default.asp'">HTML Tutorial</button>
```

Link Titles

The title attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

Example:

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML
section">Visit our HTML Tutorial</a>
```

More on Absolute URLs and Relative URLs

Example:
Use a full URL to link to a web page:

```
<a href="https://www.w3schools.com/html/default.asp">HTML tutorial</a>
```

Example:
Link to a page located in the html folder on the current web site:

```
<a href="/html/default.asp">HTML tutorial</a>
```

Example:
Link to a page located in the same folder as the current page:

```
<a href="default.asp">HTML tutorial</a>
```

HTML Links - Different Colors

An HTML link is displayed in a different color depending on whether it has been visited, is unvisited, or is active.

HTML Link Colors

By default, a link will appear like this (in all browsers):

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

You can change the link state colors, by using CSS:

Example:
Here, an unvisited link will be green with no underline. A visited link will be pink with no underline. An active link will be yellow and underlined. In addition, when mousing over a link (a:hover) it will become red and underlined:

```
<style>
a:link {
  color: green;
  background-color: transparent;
  text-decoration: none;
}
a:visited
  { color:
  pink;
  background-color: transparent;
```

```
 text-decoration: none;
}
a:hover
 { color:
 red;
 background-color: transparent;
 text-decoration: underline;
}
a:active
 { color:
 yellow;
 background-color: transparent;
 text-decoration: underline;
}
</style>
```

Link Buttons

A link can also be styled as a button, by using CSS:

THIS IS A LINK

Example:

```
<style>
a:link, a:visited {
 background-color: #f44336;
 color: white;
 padding: 15px 25px;
 text-align: center;
 text-decoration: none;
 display: inline-block;
}
a:hover, a:active
 { background-color:
 red;
}
</style>
```

HTML Link Tags

Tag	**Description**
<a>	Defines a hyperlink

Links - Create Bookmarks

HTML links can be used to create bookmarks, so that readers can jump to specific parts of a web page.

Create a Bookmark in HTML

- ✓ Bookmarks can be useful if a web page is very long.
- ✓ To create a bookmark - first create the bookmark, then add a link to it.
- ✓ When the link is clicked, the page will scroll down or up to the location with the bookmark.

Example:

First, use the id attribute to create a bookmark:

```
<h2 id="C4">Chapter 4</h2>
```

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

Example:

```
<a href="#C4">Jump to Chapter 4</a>
```

You can also add a link to a bookmark on another page:

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

IMAGES

Images can improve the design and the appearance of a web page.

Example:

```
<img src="pic_trulli.jpg" alt="Italian Trulli">
```

Example:

```
<img src="img_girl.jpg" alt="Girl in a jacket">
```

Example:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

HTML Images Syntax

- ✓ The HTML <img> tag is used to embed an image in a web page.
- ✓ Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.
- ✓ The <img> tag is empty, it contains attributes only, and does not have a closing tag.
- ✓ The <img> tag has two required attributes:
  - src - Specifies the path to the image
  - alt - Specifies an alternate text for the image

Syntax

```
<img src="url" alt="alternatetext">
```

The src Attribute

- ✓ The required src attribute specifies the path (URL) to the image.
- ✓ **Note:** When a web page loads; it is the browser, at that moment, that gets the image from a web server and inserts it into the page. Therefore, make sure that the image actually stays in the same spot in relation to the web page, otherwise your visitors will get a broken link icon. The broken link icon and the alt text are shown if the browser cannot find the image.

Example:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

The alt Attribute

- ✓ The required alt attribute provides an alternate text for an image, if the user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).
- ✓ The value of the alt attribute should describe the image:

Example:

```
<img src="img_chania.jpg" alt="Flowers in Chania">
```

If a browser cannot find an image, it will display the value of the alt attribute:
Example:

```
<img src="wrongname.gif" alt="Flowers in Chania">
```

Tip: A screen reader is a software program that reads the HTML code, and allows the user to "listen" to the content. Screen readers are useful for people who are visually impaired or learning disabled.

Image Size - Width and Height

You can use the style attribute to specify the width and height of an image.
Example:

```
<img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">
```

Alternatively, you can use the width and height attributes:
Example:

```
<img src="img_girl.jpg" alt="Girl in a jacket" width="500" height="600">
```

The width and height attributes always define the width and height of the image in pixels.
Note: Always specify the width and height of an image. If width and height are not specified, the web page might flicker while the image loads.

Width and Height, or Style?

- ✓ The width, height, and style attributes are all valid in HTML.
- ✓ However, we suggest using the style attribute. It prevents styles sheets from changing the size of images:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 100%;
}
</style>
</head>
<body>

<img src="html5.gif" alt="HTML5 Icon" width="128" height="128">

<img src="html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">
```

```
</body>
</html>
```

Images in Another Folder

If you have your images in a sub-folder, you must include the folder name in the src attribute:

Example:

```
<img src="/images/html5.gif" alt="HTML5Icon" style="width:128px;height:128px;">
```

Images on Another Server/Website

- ✓ Some web sites point to an image on another server.
- ✓ To point to an image on another server, you must specify an absolute (full) URL in the src attribute:

Example:

```
<img src="https://www.w3schools.com/images/w3schools_green.jpg" alt="W
3Schools.com">
```

Notes on external images: External images might be under copyright. If you do not get permission to use it, you may be in violation of copyright laws. In addition, you cannot control external images; it can suddenly be removed or changed.

Animated Images

HTML allows animated GIFs:

Example:

```
<img src="programming.gif" alt="Computer Man" style="width:48px;height:48px;">
```

Image as a Link

To use an image as a link, put the <img> tag inside the <a> tag:

Example:

```
<a href="default.asp">
  <img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">
</a>
```

Image Floating

Use the CSS float property to let the image float to the right or to the left of a text:

Example:

<p><img src="smiley.gif" alt="Smileyface" style="float:right;width:42px;height:42px;">
The image will float to the right of the text.</p>

<p><img src="smiley.gif" alt="Smiley face" style="float:left;width:42px;height:42px;">
The image will float to the left of the text.</p>

Common Image Formats

Here are the most common image file types, which are supported in all browsers(Chrome, Edge, Firefox, Safari, Opera):

Abbreviation	**File Format**	**File Extension**
APNG	Animated Portable Network Graphics	.apng
GIF	Graphics Interchange Format	.gif
ICO	Microsoft Icon	.ico, .cur
JPEG	Joint Photographic Expert Group image	jpg, .jpeg, .jfif, .pjpeg, .pjp
PNG	Portable Network Graphics	.png
SVG	Scalable Vector Graphics	.svg

Image Maps

With HTML image maps, you can create clickable areas on an image.

Image Maps

The HTML <map> tag defines an image map. An image map is an image with clickable areas. The areas are defined with one or more <area> tags.
Try to click on the computer, phone, or the cup of coffee in the image below:

Example:
Here is the HTML source code for the image map above:
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">

<map name="workmap">

```
  <area shape="rect" coords="34,44,270,350" alt="Computer" href="computer.
htm">
  <area shape="rect" coords="290,172,333,250" alt="Phone" href="phone.htm
">
  <area shape="circle" coords="337,300,44" alt="Coffee" href="coffee.htm">
</map>
```

How Does it Work?

- ✓ The idea behind an image map is that you should be able to perform different actions depending on where in the image you click.
- ✓ To create an image map you need an image, and some HTML code that describes the clickable areas.

The Image

The image is inserted using the <img> tag. The only difference from other images is that you must add a usemap attribute:

```
<img src="workplace.jpg" alt="Workplace" usemap="#workmap">
```

The usemap value starts with a hash tag # followed by the name of the image map, and is used to create a relationship between the image and the image map.
Tip: You can use any image as an image map!

Create Image Map

Then, add a <map> element.
The <map> element is used to create an image map, and is linked to the image by using the required name attribute:

```
<map name="workmap">
```

The name attribute must have the same value as the <img>'s usemap attribute

The Areas

Then, add the clickable areas.
A clickable area is defined using an <area> element.
Shape
You must define the shape of the clickable area, and you can choose one of these values:

- rect - defines a rectangular region
- circle - defines a circular region
- poly - defines a polygonal region

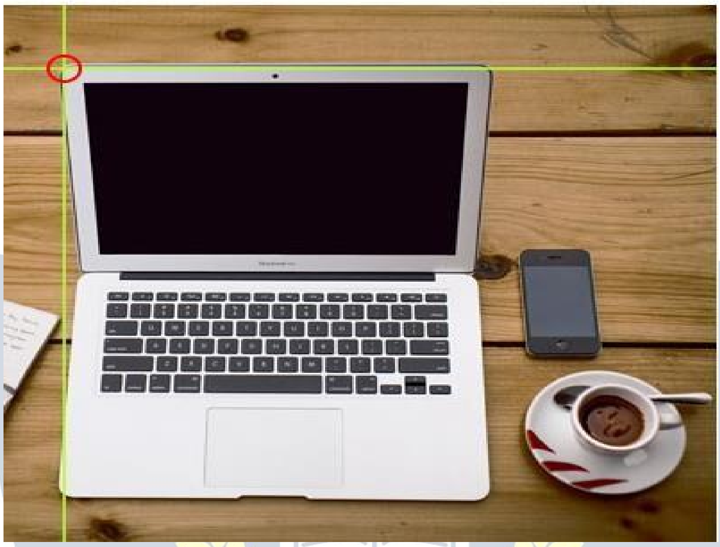
- default - defines the entire region

You must also define some coordinates to be able to place the clickable area onto the image.

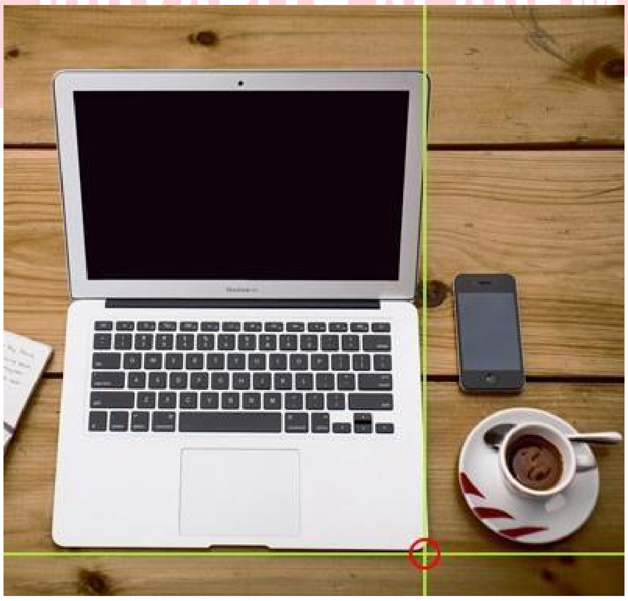
Shape="rect"

The coordinates for shape="rect" come in pairs, one for the x-axis and one for the y-axis.

So, the coordinates 34,44 is located 34 pixels from the left margin and 44 pixels from the top:



The coordinates 270,350 is located 270 pixels from the left margin and 350 pixels from the top:

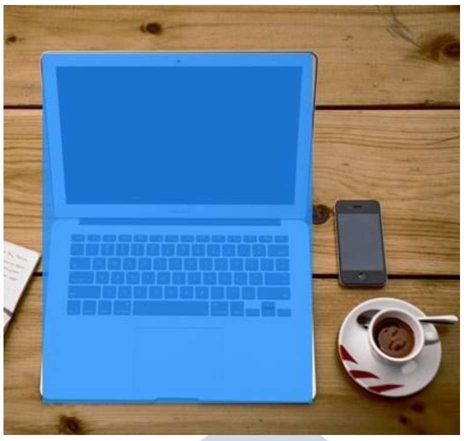


Now we have enough data to create a clickable rectangular area:

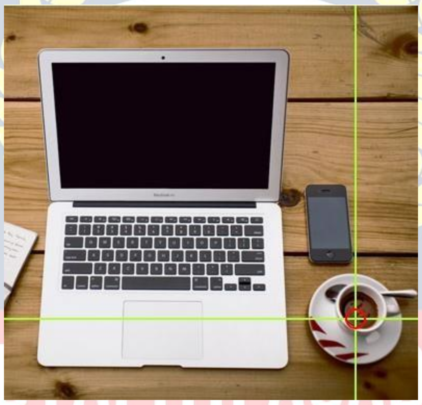
Example

```
<area shape="rect" coords="34, 44, 270, 350" href="computer.htm">
```

This is the area that becomes clickable and will send the user to the page "computer.htm":



Shape="circle"
To add a circle area, first locate the coordinates of the center of the circle:
337,300



Then specify the radius of the circle:

44 pixels

Now you have enough data to create a clickable circular area:
Example:

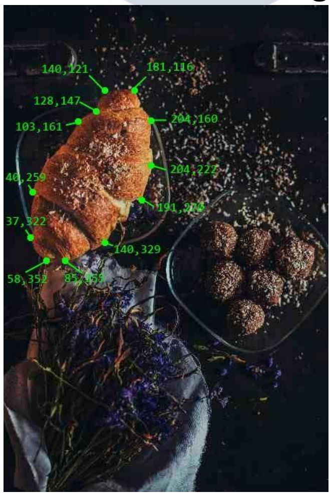
```
<area shape="circle" coords="337, 300, 44" href="coffee.htm">
```

This is the area that becomes clickable and will send the user to the page "coffee.htm":

<u>Shape="poly"</u>

The shape="poly" contains several coordinate points, which creates a shape formed with straight lines (a polygon).
This can be used to create any shape.
Like maybe a croissant shape!
How can we make the croissant in the image below become a clickable link?

We have to find the x and y coordinates for all edges of the croissant:



The coordinates come in pairs, one for the x-axis and one for the y-axis:
Example:

```
<area shape="poly"
coords="140,121,181,116,204,160,204,222,191,270,140,329,85,355,58,352,37,
322,40,259,103,161,128,147"
href="croissant.htm">
```

This is the area that becomes clickable and will send the user to the page "croissant.htm":

Image Map and JavaScript

A clickable area can also trigger a JavaScript function.
Add a click event to the <area> element to execute a JavaScript function:
Example:
Here, we use the onclick attribute to execute a JavaScript function when the area is clicked:

```
<map name="workmap">
  <area shape="circle" coords="337,300,44" href="coffee.htm" onclick="myFun
ction()">
</map>
<script>
function myFunction() {
  alert("You clicked the coffee cup!");
}
</script>
```

Background Images

A background image can be specified for almost any HTML element.

Background Image on a HTML element

To add a background image on an HTML element, use the HTML style attribute and the CSS background-image property:

Example:
Add a background image on a HTML element:

```
<div style="background-image: url('img_girl.jpg');">
```

You can also specify the background image in the <style> element, in the <head> section:

Example:
Specify the background image in the <style> element:

```
<style>
div {
  background-image: url('img_girl.jpg');
}
</style>
```

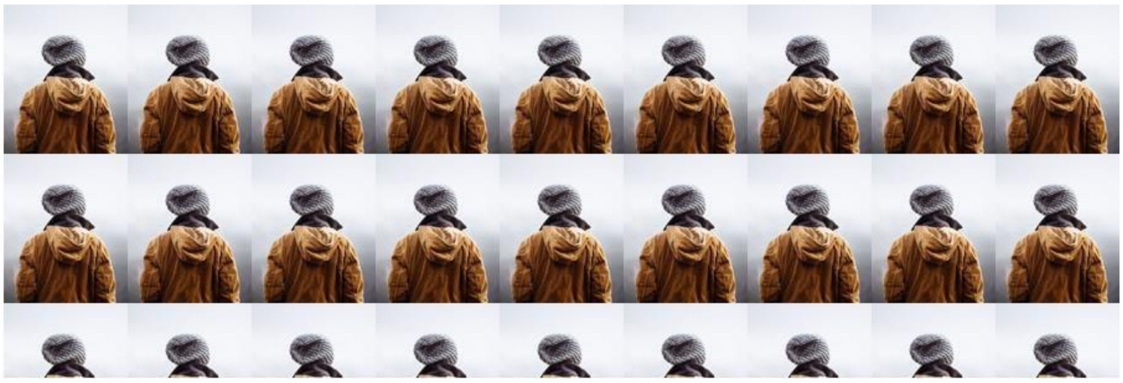
Background Image on a Page

If you want the entire page to have a background image, you must specify the background image on the <body> element:
Example:
Add a background image for the entire page:

```
<style>
body {
  background-image: url('img_girl.jpg');
}
</style>
```

Background Repeat

If the background image is smaller than the element, the image will repeat itself, horizontally and vertically, until it reaches the end of the element:


Example:

```
<style>
body {
  background-image: url('example_img_girl.jpg');
}
</style>
```

To avoid the background image from repeating itself, set the background-repeat property to no-repeat.
Example:

```
<style>
body {
  background-image: url('example_img_girl.jpg');
  background-repeat: no-repeat;
}
</style>
```

Background Cover

- ✓ If you want the background image to cover the entire element, you can set the background-size property to cover.
- ✓ Also, to make sure the entire element is always covered, set the background-attachment property to fixed:
- ✓ This way, the background image will cover the entire element, with no stretching (the image will keep its original proportions):

Example:

```
<style>
body {
  background-image: url('img_girl.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
background-size: cover;
}
</style>
```

Background Stretch

If you want the background image to stretch to fit the entire element, you can set the background-size property to 100% 100%:

Try resizing the browser window, and you will see that the image will stretch, but always cover the entire element.

Example:

```
<style>
body {
  background-image: url('img_girl.jpg');
  background-repeat: no-repeat;
  background-attachment: fixed;
  background-size: 100% 100%;
}
</style>
```

HTML <picture> Element

The HTML <picture> element allows you to display different pictures for different devices or screen sizes.

The HTML <picture> Element

- ✓ The HTML <picture> element gives web developers more flexibility in specifying image resources.
- ✓ The <picture> element contains one or more <source> elements, each referring to different images through the srcset attribute. This way the browser can choose the image that best fits the current view and/or device.
- ✓ Each <source> element has a media attribute that defines when the image is the most suitable.

Example:
Show different images for different screen sizes:

```
<picture>
  <source media="(min-width: 650px)" srcset="img_food.jpg">
  <source media="(min-width: 465px)" srcset="img_car.jpg">
  <img src="img_girl.jpg">
</picture>
```

Note: Always specify an <img> element as the last child element of the <picture> element. The <img> element is used by browsers that do not support the <picture> element, or if none of the <source> tags match.

When to use the Picture Element

There are two main purposes for the <picture> element:
1. Bandwidth
If you have a small screen or device, it is not necessary to load a large image file. The browser will use the first <source> element with matching attribute values, and ignore any of the following elements.
2. Format Support
Some browsers or devices may not support all image formats. By using the <picture> element, you can add images of all formats, and the browser will use the first format it recognizes, and ignore any of the following elements.
Example
The browser will use the first image format it recognizes:

```
<picture>
  <source srcset="img_avatar.png">
  <source srcset="img_girl.jpg">
  <img src="img_beatles.gif" alt="Beatles" style="width:auto;">
</picture>
```

Note: The browser will use the first <source> element with matching attribute values, and ignore any following <source> elements.

HTML Image Tags

Tag	Description
<img>	Defines an image
<map>	Defines an image map
<area>	Defines a clickable area inside an image map
<picture>	Defines a container for multiple image resources

TABLES

HTML tables allow web developers to arrange data into rows and columns.

Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Define an HTML Table

- ✓ The <table> tag defines an HTML table.
- ✓ Each table row is defined with a <tr> tag. Each table header is defined with a <th> tag. Each table data/cell is defined with a <td> tag.
- ✓ By default, the text in <th> elements are bold and centered.
- ✓ By default, the text in <td> elements are regular and left-aligned.

Example

A simple HTML table:

```
<table style="width:100%">
 <tr>
   <th>Firstname</th>
   <th>Lastname</th>
   <th>Age</th>
 </tr>
 <tr>
   <td>Jill</td>
   <td>Smith</td>
   <td>50</td>
 </tr>
 <tr>
```

```
   <td>Eve</td>
  <td>Jackson</td>
   <td>94</td>
 </tr>
</table>
```
Note: The <td> elements are the data containers of the table.
They can contain all sorts of HTML elements; text, images, lists, other tables, etc.

HTML Table - Add a Border
To add a border to a table, use the CSS border property:
Example:
```
table, th, td {
  border: 1px solid black;
}
```
Remember to define borders for both the table and the table cells.

HTML Table - Collapsed Borders
To let the borders collapse into one border, add the CSS border-collapse property:
Example:
```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

HTML Table - Add Cell Padding
- ✓ Cell padding specifies the space between the cell content and its borders.
- ✓ If you do not specify a padding, the table cells will be displayed without padding.
- ✓ To set the padding, use the CSS padding property:

Example:
```
th, td  { padding:
  15px;
}
```

HTML Table - Left-align Headings
- ✓ By default, table headings are bold and centered.
- ✓ To left-align the table headings, use the CSS text-align property:

Example:

```
th {
  text-align: left;
}
```

HTML Table - Add Border Spacing

- ✓ Border spacing specifies the space between the cells.
- ✓ To set the border spacing for a table, use the CSS border-spacing property:

Example:

```
table {
  border-spacing: 5px;
}
```

Note: If the table has collapsed borders, border-spacing has no effect.

HTML Table - Cell that Spans Many Columns

To make a cell span more than one column, use the colspan attribute:

Example:

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <th colspan="2">Telephone</th>
  </tr>
  <tr>
    <td>Bill Gates</td>
    <td>55577854</td>
    <td>55577855</td>
  </tr>
</table>
```

HTML Table - Cell that Spans Many Rows

To make a cell span more than one row, use the rowspan attribute:

Example:

```
<table style="width:100%">
  <tr>
    <th>Name:</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>55577854</td>
  </tr>
  <tr>
    <td>55577855</td>
  </tr>
</table>
```

HTML Table - Add a Caption

To add a caption to a table, use the <caption> tag:

Example:

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

Note: The <caption> tag must be inserted immediately after the <table> tag.

A Special Style for One Table

To define a special style for one particular table, add an id attribute to the table:

Example:

```
<table id="t01">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
```

```
   <td>94</td>
  </tr>
</table>
```

Now you can define a special style for this table:

```
#t01 {
  width: 100%;
  background-color: #f1f1c1;
}
```

And add more styles:

```
#t01 tr:nth-child(even)
{ background-color: #eee;
}
#t01 tr:nth-child(odd)
  { background-color:
  #fff;
}
#t01 th
  { color:
  white;
  background-color: black;
}
```

LISTS

HTML lists allow web developers to group a set of related items in lists.
Example:
An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

Unordered HTML List

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.
The list items will be marked with bullets (small black circles) by default:

Example:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Ordered HTML List

- ✓ An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.
- ✓ The list items will be marked with numbers by default:

Example:

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML Description Lists

- ✓ HTML also supports description lists.
- ✓ A description list is a list of terms, with a description of each term.
- ✓ The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

HTML List Tags

Tag	**Description**
<ul>	Defines an unordered list
<ol>	Defines an ordered list
<li>	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term in a description list
<dd>	Describes the term in a description list

HTML Unordered Lists

The HTML <ul> tag defines an unordered (bulleted) list.

Unordered HTML List

- ✓ An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.
- ✓ The list items will be marked with bullets (small black circles) by default:

Example:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Unordered HTML List - Choose List Item Marker

The CSS list-style-type property is used to define the style of the list item marker. It can have one of the following values:

Value	**Description**
disc	Sets the list item marker to a bullet (default)
circle	Sets the list item marker to a circle
square	Sets the list item marker to a square
none	The list items will not be marked

Example - Disc

```
<ul style="list-style-type:disc;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Example - Circle

```
<ul style="list-style-type:circle;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Example - Square

```
<ul style="list-style-type:square;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Example - None

```
<ul style="list-style-type:none;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Nested HTML Lists

Lists can be nested (list inside list):

Example

```
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
      <li>Black tea</li>
      <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
```

Note: A list item (<li>) can contain a new list, and other HTML elements, like images and links, etc.

Horizontal List with CSS

- ✓ HTML lists can be styled in many different ways with CSS.
- ✓ One popular way is to style a list horizontally, to create a navigation menu:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  overflow: hidden;
  background-color: #333333;
}

li {
  float: left;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 16px;
  text-decoration: none;
}

li a:hover {
  background-color: #111111;
}
</style>
</head>
<body>

<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>

</body>
</html>
```

HTML List Tags

Tag	Description
<ul>	Defines an unordered list
<ol>	Defines an ordered list

<li>	Defines a list item
<dl>	Defines a description list
<dt>	Defines a term in a description list
<dd>	Describes the term in a description list

HTML Ordered Lists

The HTML <ol> tag defines an ordered list. An ordered list can be numerical or alphabetical.

Ordered HTML List

- ✓ An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.
- ✓ The list items will be marked with numbers by default:

Example:

```
<ol>
 <li>Coffee</li>
 <li>Tea</li>
 <li>Milk</li>
</ol>
```

Ordered HTML List - The Type Attribute

The type attribute of the <ol> tag, defines the type of the list item marker:

Type	**Description**
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

Numbers:

```
<ol type="1">
 <li>Coffee</li>
 <li>Tea</li>
 <li>Milk</li>
</ol>
```

Uppercase Letters:

```
<ol type="A">
 <li>Coffee</li>
 <li>Tea</li>
```

```
  <li>Milk</li>
</ol>
```

Lowercase Letters:

```
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Uppercase Roman Numbers:

```
<ol type="I">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Lowercase Roman Numbers:

```
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Control List Counting

By default, an ordered list will start counting from 1. If you want to start counting from a specified number, you can use the start attribute:

Example:

```
<ol start="50">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Nested HTML Lists

Lists can be nested (list inside list):

Example:

```
<ol>
  <li>Coffee</li>
  <li>Tea
    <ol>
    <li>Black tea</li>
    <li>Green tea</li>
  </ol>
 </li>
 <li>Milk</li>
</ol>
```

HTML Other Lists

HTML also supports description lists.

HTML Description Lists

- ✓ A description list is a list of terms, with a description of each term.
- ✓ The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example:

```
<dl>
 <dt>Coffee</dt>
 <dd>- black hot drink</dd>
 <dt>Milk</dt>
 <dd>- white cold drink</dd>
</dl>
```

BLOCK AND INLINE ELEMENTS

- ✓ Every HTML element has a default display value, depending on what type of element it is.
- ✓ There are two display values: block and inline.

Block-level Elements

- ✓ A block-level element always starts on a new line.
- ✓ A block-level element always takes up the full width available (stretches out to the left and right as far as it can).
- ✓ A block level element has a top and a bottom margin, whereas an inline element does not.

The <div> element is a block-level element.

Example:

```
<div>Hello World</div>
```

Here are the block-level elements in HTML:

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>
<div>	<dl>	<dt>	<fieldset>	<figcaption>	
<figure>	<footer>	<form>	<h1>-<h6>	<header>	<hr>
<li>	<main>	<nav>	<noscript>	<ol>	<p>
<pre>	<section>	<table>	<tfoot>	<ul>	<video

Inline Elements

- ✓ An inline element does not start on a new line.
- ✓ An inline element only takes up as much width as necessary.

This is a <span> element inside a paragraph.

Example:

```
<span>Hello World</span>
```

Here are the inline elements in HTML:

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>
<br>	<button>	<cite>	<code>	<dfn>	<em>
<i>	<img>	<input>	<kbd>	<label>	<map>
<object>	<output>	<q>	<samp>	<script>	<select>
<small>	<span>	<strong>	<sub>	<sup>	
<textarea>	<time>	<tt>	<var>		

Note: An inline element cannot contain a block-level element!

The <div> Element

- ✓ The <div> element is often used as a container for other HTML elements.
- ✓ The <div> element has no required attributes, but style, class and id are common.
- ✓ When used together with CSS, the <div> element can be used to style blocks of content:

Example:

```
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous city in the
United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
</div>
```

The <span> Element

- ✓ The <span> element is an inline container used to mark up a part of a text, or a part of a document.

- ✓ The <span> element has no required attributes, but style, class and id are common.
- ✓ When used together with CSS, the <span> element can be used to style parts of the text:

Example:

```
<p>My mother has <span style="color:blue;font-
weight:bold">blue</span> eyes and my father
has <span style="color:darkolivegreen;font-weight:bold">dark
green</span> eyes.</p>
```

CLASS ATTRIBUTE

- ✓ The HTML class attribute is used to specify a class for an HTML element.
- ✓ Multiple HTML elements can share the same class.

Using The class Attribute

- ✓ The class attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.
- ✓ In the following example we have three <div> elements with a class attribute with the value of "city". All of the three <div> elements will be styled equally according to the .city style definition in the head section:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
  background-color: tomato;
  color: white;
  border: 2px solid black;
  margin: 20px;
  padding: 20px;
}
</style>
</head>
<body>

<div class="city">
  <h2>London</h2>
  <p>London is the capital of England.</p>
```

```
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>

<div class="city">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>

</body>
</html>
```

In the following example we have two <span> elements with a class attribute with the value of "note". Both <span> elements will be styled equally according to the .note style definition in the head section:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> Heading</h1>
<p>This is some <span class="note">important</span> text.</p>

</body>
</html>
```

Tip: The class attribute can be used on **any** HTML element.

Note: The class name is case sensitive!

The Syntax For Class

To create a class; write a period (.) character, followed by a class name. Then, define the CSS properties within curly braces {}:

Example:
Create a class named "city":

```
<!DOCTYPE html>
<html>
<head>
<style>
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>
</head>
<body>

<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>

</body>
</html>
```

Multiple Classes

- ✓ HTML elements can belong to more than one class.
- ✓ To define multiple classes, separate the class names with a space, e.g. <div class="city main">. The element will be styled according to all the classes specified.
- ✓ In the following example, the first <h2> element belongs to both the city class and also to the main class, and will get the CSS styles from both of the classes:

Example:

```
<h2 class="city main">London</h2>
<h2 class="city">Paris</h2>
<h2 class="city">Tokyo</h2>
```

Different Elements Can Share Same Class

- ✓ Different HTML elements can point to the same class name.
- ✓ In the following example, both <h2> and <p> points to the "city" class and will share the same style:

Example:

```
<h2 class="city">Paris</h2>
<p class="city">Paris is the capital of France</p>
```

Use of The class Attribute in JavaScript

- ✓ The class name can also be used by JavaScript to perform certain tasks for specific elements.
- ✓ JavaScript can access elements with a specific class name with the getElementsByClassName() method:

Example:

Click on a button to hide all elements with the class name "city":

```
<script>
function myFunction() {
  var x = document.getElementsByClassName("city");
  for (var i = 0; i < x.length; i++) {
    x[i].style.display = "none";
  }
}
</script>
```

ID ATTRIBUTE

- ✓ The HTML id attribute is used to specify a unique id for an HTML element.
- ✓ You cannot have more than one element with the same id in an HTML document.

Using The id Attribute

- ✓ The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.
- ✓ The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

- ✓ The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.
- ✓ In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

Note: The id name is case sensitive!
Note: The id name must contain at least one character, cannot start with a number, and must not contain whitespaces (spaces, tabs, etc.).

Difference Between Class and ID

A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:

Example:

```
<style>
/* Style the element with the id "myHeader" */
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}

/* Style all elements with the class name "city" */
.city {
  background-color: tomato;
  color: white;
  padding: 10px;
}
</style>

<!-- An element with a unique id -->
<h1 id="myHeader">My Cities</h1>

<!-- Multiple elements with same class -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>
```

HTML Bookmarks with ID and Links

- ✓ HTML bookmarks are used to allow readers to jump to specific parts of a webpage.
- ✓ Bookmarks can be useful if your page is very long.
- ✓ To use a bookmark, you must first create it, and then add a link to it.
- ✓ Then, when the link is clicked, the page will scroll to the location with the bookmark.

Example:
First, create a bookmark with the id attribute:

```
<h2 id="C4">Chapter 4</h2>
```

Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:

Example:

```
<a href="#C4">Jump to Chapter 4</a>
```

Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:

```
<a href="html_demo.html#C4">Jump to Chapter 4</a>
```

Using The id Attribute in JavaScript

- ✓ The id attribute can also be used by JavaScript to perform some tasks for that specific element.
- ✓ JavaScript can access an element with a specific id with the getElementById() method:

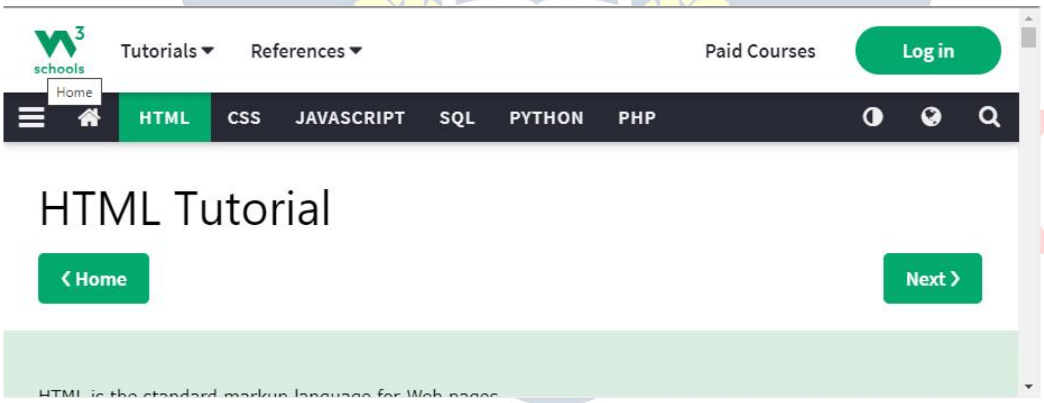
Example:

Use the id attribute to manipulate text with JavaScript:

```
<script>
function displayResult()
 { document.getElementById("myHeader").innerHTML = "Have a nice day!";
}
</script>
```

IFRAMES

An HTML iframe is used to display a web page within a web page.



HTML is the standard markup language for Web pages.

HTML Iframe Syntax

- ✓ The HTML <iframe> tag specifies an inline frame.
- ✓ An inline frame is used to embed another document within the current HTML document.

Syntax:

```
<iframe src="url" title="description"></iframe>
```

Tip: It is a good practice to always include a title attribute for the <iframe>. This is used by screen readers to read out what the content of the iframe is.

Iframe - Set Height and Width

- ✓ Use the height and width attributes to specify the size of the iframe.
- ✓ The height and width are specified in pixels by default:

Example:

```
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe
Example"></iframe>
```

Or you can add the style attribute and use the CSS height and width properties:

Example:

```
<iframe src="demo_iframe.htm" style="height:200px;width:300px;" title="Ifra
me Example"></iframe>
```

Iframe - Remove the Border

- ✓ By default, an iframe has a border around it.
- ✓ To remove the border, add the style attribute and use the CSS border property:

Example:

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe
Example"></iframe>
```

With CSS, you can also change the size, style and color of the iframe's border:

Example:

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe
Example"></iframe>
```

Iframe - Target for a Link

- ✓ An iframe can be used as the target frame for a link.
- ✓ The target attribute of the link must refer to the name attribute of the iframe:

Example:

```
<iframe src="demo_iframe.htm" name="iframe_a" title="Iframe
Example"></iframe>

<p><a href="https://www.w3schools.com" target="iframe_a">W3Schools.com
</a></p>
```

JAVASCRIPT

JavaScript makes HTML pages more dynamic and interactive.

Example:

My First JavaScript

Click me to display Date and Time

The HTML <script> Tag

- ✓ The HTML <script> tag is used to define a client-side script (JavaScript).
- ✓ The <script> element either contains script statements, or it points to an external script file through the src attribute.
- ✓ Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- ✓ To select an HTML element, JavaScript most often uses the document.getElementById() method.
- ✓ This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":

Example:

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

A Taste of JavaScript

Here are some examples of what JavaScript can do:

Example:
JavaScript can change content:

```
document.getElementById("demo").innerHTML = "Hello JavaScript!";
```

Example:
JavaScript can change styles:

```
document.getElementById("demo").style.fontSize = "25px";
document.getElementById("demo").style.color = "red";
document.getElementById("demo").style.backgroundColor = "yellow";
```

Example:
JavaScript can change attributes:

```
document.getElementById("image").src = "picture.gif";
```

The HTML <noscript> Tag

The HTML <noscript> tag defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support scripts:

Example:

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
<noscript>Sorry, your browser does not support JavaScript!</noscript>
```

FILE PATHS

A file path describes the location of a file in a web site's folder structure.

File Path Examples:

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

HTML File Paths

A file path describes the location of a file in a web site's folder structure.

File paths are used when linking to external files, like:

- Web pages
- Images
- Style sheets
- JavaScripts

Absolute File Paths

An absolute file path is the full URL to a file:

Example:

```
<img src="https://www.ditrp.com/images/picture.jpg" alt="Mountain">
```

The <img> tag is explained in the chapter: HTML Images.

Relative File Paths

✓ A relative file path points to a file relative to the current page.

- ✓ In the following example, the file path points to a file in the images folder located at the root of the current web:

Example:
`<img src="/images/picture.jpg" alt="Mountain">`

In following example, the file path points to a file in the images folder located in the current folder:

Example:
`<img src="images/picture.jpg" alt="Mountain">`

In the following example, the file path points to a file in the images folder located in the folder one level up from the current folder:

Example:
`<img src="../images/picture.jpg" alt="Mountain">`

HEAD

The HTML <head> element is a container for the following elements: <title>, <style>, <meta>, <link>, <script>, and <base>.

The HTML <head> Element

- ✓ The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag.
- ✓ HTML metadata is data about the HTML document. Metadata is not displayed.
- ✓ Metadata typically define the document title, character set, styles, scripts, and other meta information.

The HTML <title> Element

- ✓ The <title> element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.
- ✓ The <title> element is required in HTML documents!
- ✓ The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine-results

So, try to make the title as accurate and meaningful as possible!
A simple HTML document:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title>A Meaningful Page Title</title>
</head>
<body>

The content of the document......

</body>
</html>
```

The HTML <style> Element

The <style> element is used to define style information for a single HTML page:

Example:

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

The HTML <link> Element

- ✓ The <link> element defines the relationship between the current document and an external resource.
- ✓ The <link> tag is most often used to link to external style sheets:

Example:

```
<link rel="stylesheet" href="mystyle.css">
```

The HTML <meta> Element

- ✓ The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.
- ✓ The metadata will not be displayed on the page, but are used by browsers (how to display content or reload page), by search engines (keywords), and other web services.

Examples:

Define the character set used:

<meta charset="UTF-8">

Define keywords for search engines:
<meta name="keywords" content="HTML, CSS, JavaScript">

Define a description of your web page:
<meta name="description" content="Free Web tutorials">

Define the author of a page:
<meta name="author" content="John Doe">

Refresh document every 30 seconds:
<meta http-equiv="refresh" content="30">

Setting the viewport to make your website look good on all devices:
<meta name="viewport" content="width=device-width, initial-scale=1.0">
Example of <meta> tags:

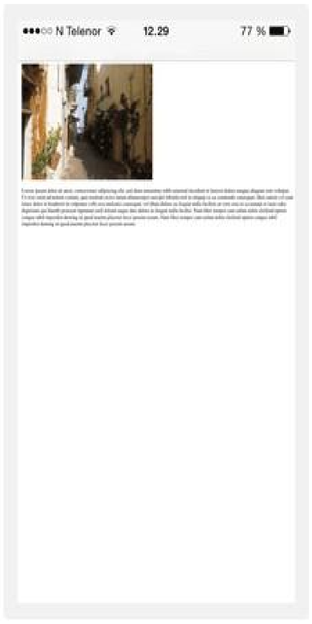
Example:
```
<meta charset="UTF-8">
<meta name="description" content="Free Web tutorials">
<meta name="keywords" content="HTML, CSS, JavaScript">
<meta name="author" content="John Doe">
```

Setting The Viewport

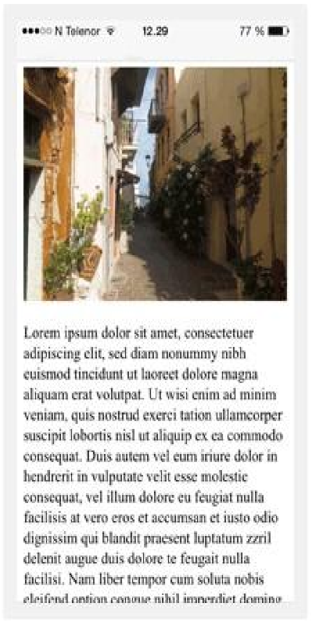
- ✓ The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.
- ✓ You should include the following <meta> element in all your web pages:

<meta name="viewport" content="width=device-width, initial-scale=1.0">

- ✓ This gives the browser instructions on how to control the page's dimensions and scaling.
- ✓ The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- ✓ The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.
- ✓ Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:



Without the viewport meta tag



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

With the viewport meta tag

The HTML <script> Element

The <script> element is used to define client-side JavaScripts.
The following JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":

Example:

```
<script>
function myFunction()
 { document.getElementById("demo").innerHTML = "Hello
 JavaScript!";
}
</script>
```

The HTML <base> Element

- ✓ The <base> element specifies the base URL and/or target for all relative URLs in a page.
- ✓ The <base> tag must have either an href or a target attribute present, or both.
- ✓ There can only be one single <base> element in a document!

Example:

Specify a default URL and a default target for all links on a page:

```
<head>
<base href="https://www.w3schools.com/" target="_blank">
```

```
</head>

<body>
<img src="images/stickman.gif" width="24" height="39" alt="Stickman">
<a href="tags/tag_base.asp">HTML base Tag</a>
</body>
```

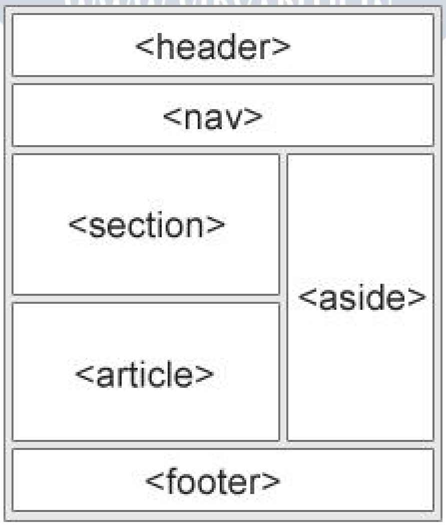
LAYOUT ELEMENTS AND TECHNIQUES

Websites often display content in multiple columns (like a magazine or a newspaper).
Example:

Cities	
• London • Paris • Tokyo	**London** London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants. Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.
Footer	

HTML Layout Elements

HTML has several semantic elements that define the different parts of a web page:



- <header> - Defines a header for a document or a section
- <nav> - Defines a set of navigation links

- <section> - Defines a section in a document
- <article> - Defines an independent, self-contained content
- <aside> - Defines content aside from the content (like a sidebar)
- <footer> - Defines a footer for a document or a section
- <details> - Defines additional details that the user can open and close on demand
- <summary> - Defines a heading for the <details> element

You can read more about semantic elements in our HTML Semantics chapter.

HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:

- CSS framework
- CSS float property
- CSS flexbox
- CSS grid

CSS Frameworks

If you want to create your layout fast, you can use a CSS framework, like W3.CSS or Bootstrap.

CSS Float Layout

It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. **Disadvantages:** Floating elements are tied to the document flow, which may harm the flexibility. Learn more about float in our CSS Float and Clear chapter.

Example:

Cities	
• London • Paris • Tokyo	**London** London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants. Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.
Footer	

CSS Flexbox Layout

✓ Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

✓ Learn more about flexbox in our CSS Flexbox chapter.

Example:

Cities	
• London • Paris • Tokyo	**London** London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants. Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.
Footer	

CSS Grid Layout

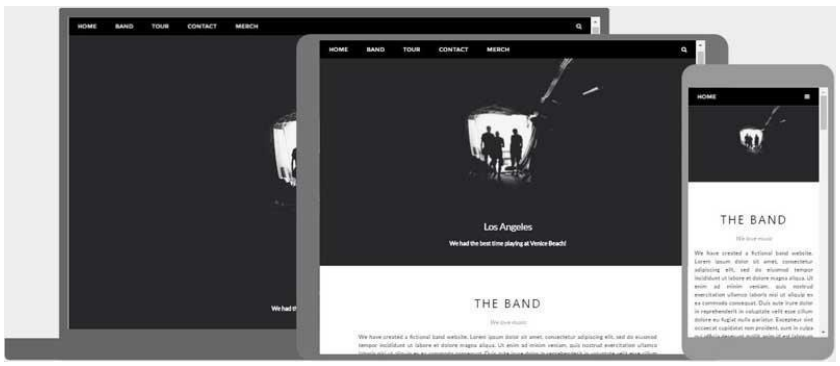
✓ The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

✓ Learn more about CSS grids in our CSS Grid View chapter.

RESPONSIVE WEB DESIGN

Responsive web design is about creating web pages that look good on all devices!

A responsive web design will automatically adjust for different screen sizes and viewports.



What is Responsive Web Design?

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

Setting The Viewport

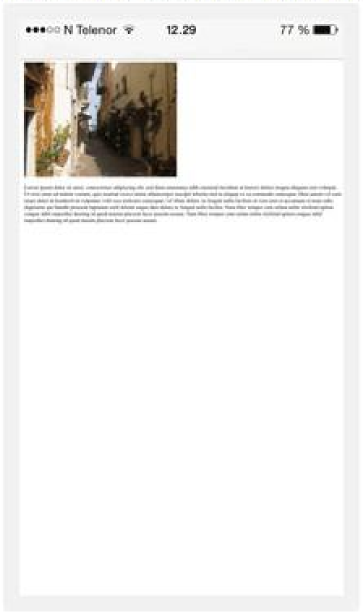
To create a responsive website, add the following <meta> tag to all your web pages:

Example:

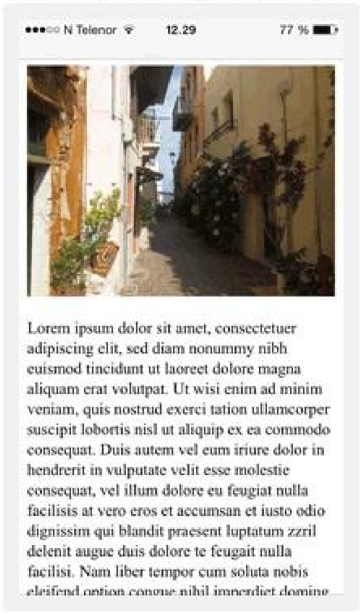
```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- ✓ This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.
- ✓ Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag:



With the viewport meta tag:



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

Tip: If you are browsing this page on a phone or a tablet, you can click on the two links above to see the difference.

Responsive Images

Responsive images are images that scale nicely to fit any browser size.

Using the width Property

If the CSS width property is set to 100%, the image will be responsive and scale up and down:

Example:
```
<img src="img_girl.jpg" style="width:100%;">
```
Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

Using the max-width Property

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

Example:
```
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
```

Show Different Images Depending on Browser Width

The HTML <picture> element allows you to define different images for different browser window sizes.

Resize the browser window to see how the image below change depending on the width:

Example:

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width". That way the text size will follow the size of the browser window:

Hello World

Resize the browser window to see how the text size scales.

Example:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

Media Queries

- ✓ In addition to resize text and images, it is also common to use media queries in responsive web pages.
- ✓ With media queries you can define completely different styles for different browser sizes.
- ✓ Example: resize the browser window to see that the three div elements below will display horizontally on large screens and stacked vertically on small screens:

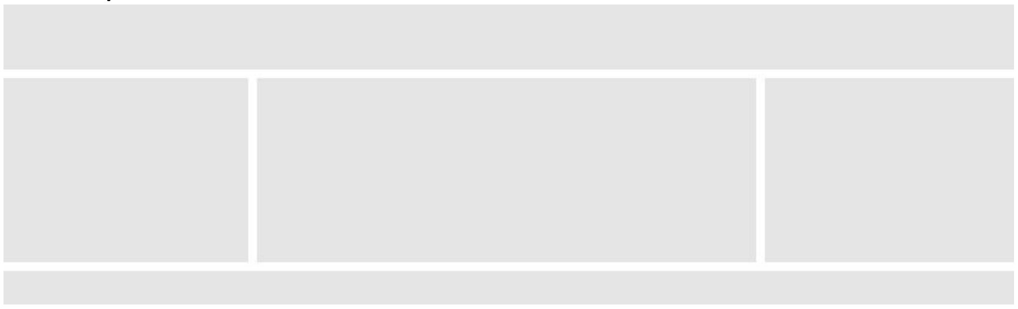
Left Menu	Main Content	Right Content

Example:

```
<style>
.left, .right
 {float: left;
 width: 20%; /* The width is 20%, by default */
}
.main
 { float:
 left;
 width: 60%; /* The width is 60%, by default */
}

/* Use a media query to add a breakpoint at 800px: */
@media screen and (max-width: 800px) {
 .left, .main, .right {
   width: 100%; /* The width is 100%, when the viewport is 800px or smaller */
 }
}
</style>
```

Responsive Web Page - Full Example

A responsive web page should look good on large desktop screens and on small mobile phones.

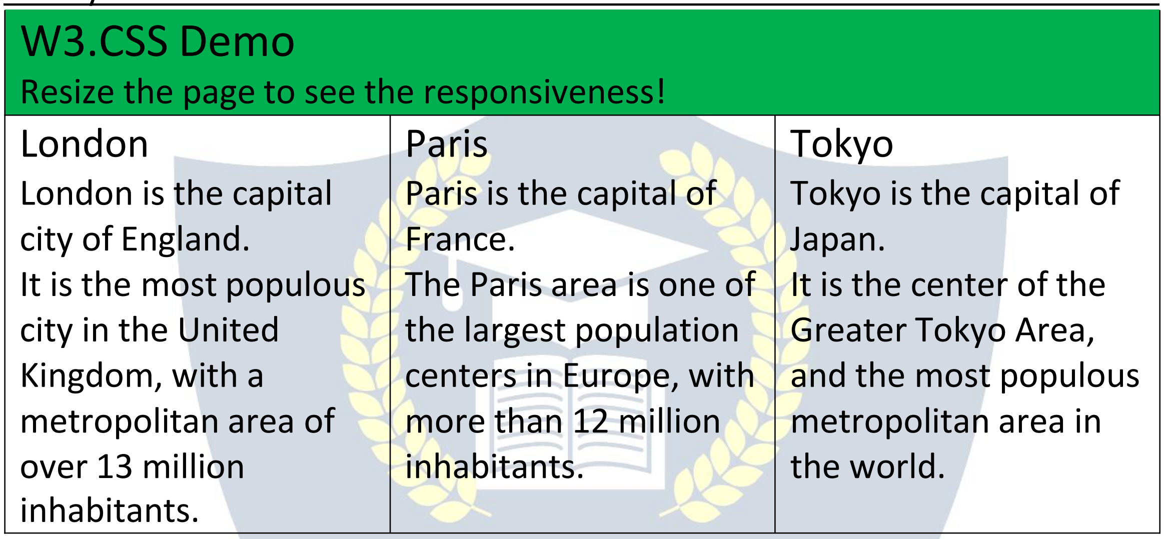


Responsive Web Design - Frameworks

All popular CSS Frameworks offer responsive design.
They are free, and easy to use.

W3.CSS

- ✓ W3.CSS is a modern CSS framework with support for desktop, tablet, and mobile design by default.
- ✓ W3.CSS is smaller and faster than similar CSS frameworks.
- ✓ W3.CSS is designed to be a high quality alternative to Bootstrap.
- ✓ W3.CSS is designed to be independent of jQuery or any other JavaScript library.



W3.CSS Demo

Resize the page to see the responsiveness!

London	Paris	Tokyo
London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.	Paris is the capital of France. The Paris area is one of the largest population centers in Europe, with more than 12 million inhabitants.	Tokyo is the capital of Japan. It is the center of the Greater Tokyo Area, and the most populous metropolitan area in the world.

Example:

```
<!DOCTYPE html>
<html>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<body>

<div class="w3-container w3-green">
  <h1>W3Schools Demo</h1>
  <p>Resize this responsive page!</p>
</div>

<div class="w3-row-padding">
  <div class="w3-third">
    <h2>London</h2>
    <p>London is the capital city of England.</p>
    <p>It is the most populous city in the United Kingdom,
    with a metropolitan area of over 13 million inhabitants.</p>
  </div>

  <div class="w3-third">
    <h2>Paris</h2>
    <p>Paris is the capital of France.</p>
    <p>The Paris area is one of the largest population centers in Europe,
    with more than 12 million inhabitants.</p>
  </div>

  <div class="w3-third">
    <h2>Tokyo</h2>
    <p>Tokyo is the capital of Japan.</p>
    <p>It is the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.</p>
  </div>
</div>

</body>
</html>
```

Bootstrap

Another popular CSS framework is Bootstrap. Bootstrap uses HTML, CSS and jQuery to make responsive web pages.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Bootstrap Example</title>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.
1/css/bootstrap.min.css">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js">
</script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.
min.js"></script>
</head>
<body>

<div class="container">
  <div class="jumbotron">
    <h1>My First Bootstrap Page</h1>
  </div>
  <div class="row">
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
      ...
    </div>
    <div class="col-sm-4">
    ...
    </div>
  </div>
</div>

</body>
</html>
```

COMPUTER CODE ELEMENTS

HTML contains several elements for defining user input and computer code.

Example:

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

HTML <kbd> For Keyboard Input

The HTML <kbd> element is used to define keyboard input. The content inside is displayed in the browser's default monospace font.

Example:
Define some text as keyboard input in a document:

```
<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>
```

Result:
Save the document by pressing Ctrl + S

HTML <samp> For Program Output

The HTML <samp> element is used to define sample output from a computer program. The content inside is displayed in the browser's default monospace font.

Example:
Define some text as sample output from a computer program in a document:

```
<p>Message from my computer:</p>
<p><samp>File not found.<br>Press F1 to continue</samp></p>
```

Result:
Message from my computer:
File not found.
Press F1 to continue

HTML <code> For Computer Code

The HTML <code> element is used to define a piece of computer code. The content inside is displayed in the browser's default monospace font.

Example:
Define some text as computer code in a document:

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

Result:
x = 5; y = 6; z = x + y;

✓ Notice that the <code> element does not preserve extra whitespace and line-breaks.

✓ To fix this, you can put the <code> element inside a <pre> element:

Example:

```
<pre>
<code>
x = 5;
y = 6;
z = x + y;
</code>
</pre>
```

Result:

```
x = 5;
y = 6;
z = x + y;
```

HTML <var> For Variables

The HTML <var> element is used to define a variable in programming or in a mathematical expression. The content inside is typically displayed in italic.

Example:

Define some text as variables in a document:

```
<p>The area of a triangle is: 1/2 x <var>b</var> x <var>h</var>,
where <var>b</var> is the base, and <var>h</var> is the vertical height.</p>
```

Result:

The area of a triangle is: 1/2 x *b* x *h*, where *b* is the base, and *h* is the vertical height.

SEMANTIC ELEMENTS

Semantic elements = elements with a meaning.

What are Semantic Elements?

- ✓ A semantic element clearly describes its meaning to both the browser and the developer.
- ✓ Examples of **non-semantic** elements: <div> and <span> - Tells nothing about its content.
- ✓ Examples of **semantic** elements: <form>, <table>, and <article> - Clearly defines its content.

Semantic Elements in HTML

- ✓ Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.
- ✓ In HTML there are some semantic elements that can be used to define different parts of a web page:
  - <article>
  - <aside>
  - <details>
  - <figcaption>
  - <figure>
  - <footer>
  - <header>
  - <main>
  - <mark>
  - <nav>
  - <section>
  - <summary>
  - <time>

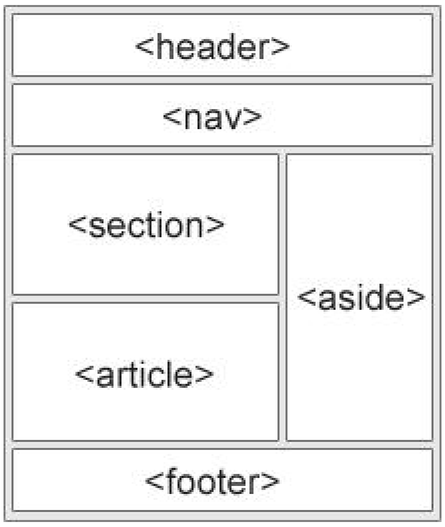
HTML <section> Element

- ✓ The <section> element defines a section in a document.
- ✓ According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."
- ✓ A web page could normally be split into sections for introduction, content, and contact information.

Example:

Two sections in a document:

```
<section>
<h1>WWF</h1>
<p>The World Wide Fund for Nature (WWF) is
an international organization working on issues
regarding the conservation, research and
restoration of the environment, formerly named
the World Wildlife Fund. WWF was founded in
1961.</p>
</section>

<section>
```



```
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of
WWF originated from a panda named Chi Chi that was transferred from the
Beijing Zoo to the London Zoo in the same year of the establishment of
WWF.</p>
</section>
```

HTML <article> Element

- ✓ The <article> element specifies independent, self-contained content.
- ✓ An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.

Examples of where an <article> element can be used:

- Forum post
- Blog post
- Newspaper article

Example:
Three articles with independent, self-contained content:

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008.
Chrome is the world's most popular web browser today!</p>
</article>

<article>
<h2>Mozilla Firefox</h2>
<p>Mozilla Firefox is an open-source web browser developed by Mozilla.
Firefox has been the second most popular web browser since January,
2018.</p>
</article>

<article>
<h2>Microsoft Edge</h2>
<p>Microsoft Edge is a web browser developed by Microsoft, released in 2015.
Microsoft Edge replaced Internet Explorer.</p>
</article>
```

Example 2:
Use CSS to style the <article> element:

```
<html>
<head>
<style>
.all-browsers
  {margin: 0;
  padding: 5px;
  background-color: lightgray;
}

.all-browsers > h1, .browser
  {margin: 10px;
  padding: 5px;
}

.browser
  { background:
  white;
}

.browser > h2, p
  {margin: 4px;
  font-size: 90%;
}
</style>
</head>
<body>

<article class="all-browsers">
  <h1>Most Popular Browsers</h1>
  <article class="browser">
    <h2>Google Chrome</h2>
    <p>Google Chrome is a web browser developed by Google, released in 2008.
Chrome is the world's most popular web browser today!</p>
  </article>
  <article class="browser">
    <h2>Mozilla Firefox</h2>
    <p>Mozilla Firefox is an open-source web browser developed by Mozilla.
Firefox has been the second most popular web browser since January,
2018.</p>
  </article>
  <article class="browser">
    <h2>Microsoft Edge</h2>
    <p>Microsoft Edge is a web browser developed by Microsoft, released in
2015. Microsoft Edge replaced Internet Explorer.</p>
  </article>
</article>

</body>
</html>
```

Nesting <article> in <section> or Vice Versa?

- ✓ The <article> element specifies independent, self-contained content.
- ✓ The <section> element defines section in a document.
- ✓ Can we use the definitions to decide how to nest those elements? No, we cannot!
- ✓ So, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.

HTML <header> Element

- ✓ The <header> element represents a container for introductory content or a set of navigational links.
- ✓ A <header> element typically contains:
  - one or more heading elements (<h1> - <h6>)
  - logo or icon
  - authorship information

Note: You can have several <header> elements in one HTML document. However, <header> cannot be placed within a <footer>, <address> or another <header> element.

Example:
A header for an <article>:

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural
```

```
environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

HTML <footer> Element

- ✓ The <footer> element defines a footer for a document or section.
- ✓ A <footer> element typically contains:
  - authorship information
  - copyright information
  - contact information
  - sitemap
  - back to top links
  - related documents

You can have several <footer> elements in one document.

Example:
A footer section in a document:

```
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

HTML <nav> Element

- ✓ The <nav> element defines a set of navigation links.
- ✓ Notice that NOT all links of a document should be inside a <nav> element. The <nav> element is intended only for major block of navigation links.
- ✓ Browsers, such as screen readers for disabled users, can use this element to determine whether to omit the initial rendering of this content.

Example:
A set of navigation links:

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

HTML <aside> Element

- ✓ The <aside> element defines some content aside from the content it is placed in (like a sidebar).
- ✓ The <aside> content should be indirectly related to the surrounding content.

Example:
Display some content aside from the content it is placed in:

```
<p>My family and I visited The Epcot center this summer. The weather was
nice, and Epcot was amazing! I had a great summer together with my
family!</p>

<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting
attractions, international pavilions, award-winning fireworks and seasonal
special events.</p>
</aside>
```

Example 2:
Use CSS to style the <aside> element:

```
<html>
<head>
<style>
aside {
  width: 30%;
  padding-left: 15px;
  margin-left: 15px;
  float: right;
  font-style: italic;
  background-color: lightgray;
}
</style>
</head>
<body>

<p>My family and I visited The Epcot center this summer. The weather was
nice, and Epcot was amazing! I had a great summer together with my
family!</p>

<aside>
```

```
<p>The Epcot center is a theme park at Walt Disney World Resort featuring
exciting attractions, international pavilions, award-winning fireworks and
seasonal special events.</p>
</aside>

<p>My family and I visited The Epcot center this summer. The weather was
nice, and Epcot was amazing! I had a great summer together with my
family!</p>
<p>My family and I visited The Epcot center this summer. The weather was
nice, and Epcot was amazing! I had a great summer together with my
family!</p>

</body>
</html>
```

HTML <figure> and <figcaption> Elements

- ✓ The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
- ✓ The <figcaption> tag defines a caption for a <figure> element. The <figcaption> element can be placed as the first or as the last child of a <figure> element.
- ✓ The <img> element defines the actual image/illustration.

Example:

```
<figure>
  <img src="pic_trulli.jpg" alt="Trulli">
  <figcaption>Fig1. - Trulli, Puglia, Italy.</figcaption>
</figure>
```

Why Semantic Elements?

According to the W3C: "A semantic Web allows data to be shared and reused across applications, enterprises, and communities."

Semantic Elements in HTML

Below is a list of some of the semantic elements in HTML.

Tag	**Description**
<article>	Defines independent, self-contained content
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide

<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

STYLE GUIDE AND CODING CONVENTIONS

A consistent, clean, and tidy HTML code makes it easier for others to read and understand your code.
Here are some guidelines and tips for creating good HTML code.

Always Declare Document Type

- ✓ Always declare the document type as the first line in your document.
- ✓ The correct document type for HTML is:

```
<!DOCTYPE html>
```

Use Lowercase Element Names

- ✓ HTML allows mixing uppercase and lowercase letters in element names.
- ✓ However, we recommend using lowercase element names, because:
  - Mixing uppercase and lowercase names looks bad
  - Developers normally use lowercase names
  - Lowercase looks cleaner
  - Lowercase is easier to write

Good:
```
<body>
<p>This is a paragraph.</p>
</body>
```

Bad:
```
<BODY>
<P>This is a paragraph.</P>
</BODY>
```

Close All HTML Elements

- ✓ In HTML, you do not have to close all elements (for example the <p> element).
- ✓ However, we strongly recommend closing all HTML elements, like this:

Good:

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```

Bad:

```
<section>
  <p>This is a paragraph.
  <p>This is a paragraph.
</section>
```

Use Lowercase Attribute Names

- ✓ HTML allows mixing uppercase and lowercase letters in attribute names.
- ✓ However, we recommend using lowercase attribute names, because:
  - Mixing uppercase and lowercase names looks bad
  - Developers normally use lowercase names
  - Lowercase look cleaner
  - Lowercase are easier to write

Good:

```
<a href="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

Bad:

```
<a HREF="https://www.w3schools.com/html/">Visit our HTML tutorial</a>
```

Always Quote Attribute Values

- ✓ HTML allows attribute values without quotes.
- ✓ However, we recommend quoting attribute values, because:
  - Developers normally quote attribute values
  - Quoted values are easier to read
  - You MUST use quotes if the value contains spaces

Good:

```
<table class="striped">
```

Bad:

```
<table class=striped>
```

Very bad:
This will not work, because the value contains spaces:

```
<table class=table striped>
```

Always Specify alt, width, and height for Images

- ✓ Always specify the alt attribute for images. This attribute is important if the image for some reason cannot be displayed.
- ✓ Also, always define the width and height of images. This reduces flickering, because the browser can reserve space for the image before loading.

Good:

```
<img src="html5.gif" alt="HTML5" style="width:128px;height:128px">
```

Bad:

```
<img src="html5.gif">
```

Spaces and Equal Signs

HTML allows spaces around equal signs. But space-less is easier to read and groups entities better together.

Good:

```
<link rel="stylesheet" href="styles.css">
```

Bad:

```
<link rel = "stylesheet" href = "styles.css">
```

Avoid Long Code Lines

- ✓ When using an HTML editor, it is NOT convenient to scroll right and left to read the HTML code.
- ✓ Try to avoid too long code lines.

Blank Lines and Indentation

- ✓ Do not add blank lines, spaces, or indentations without a reason.
- ✓ For readability, add blank lines to separate large or logical code blocks.
- ✓ For readability, add two spaces of indentation. Do not use the tab key.

Good:

```
<body>

<h1>Famous Cities</h1>
```

```
<h2>Tokyo</h2>
<p>Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
and the most populous metropolitan area in the world.
It is the seat of the Japanese government and the Imperial Palace,
and the home of the Japanese Imperial Family.</p>

</body>
```

Bad:

```
<body>

  <h1>Famous Cities</h1>

  <h2>Tokyo</h2>

  <p>
    Tokyo is the capital of Japan, the center of the Greater Tokyo Area,
    and the most populous metropolitan area in the world.
    It is the seat of the Japanese government and the Imperial Palace,
    and the home of the Japanese Imperial Family.
  </p>

</body>
```

Good Table Example:

```
<table>
  <tr>
    <th>Name</th>
    <th>Description</th>
  </tr>
  <tr>
    <td>A</td>
    <td>Description of A</td>
  </tr>
  <tr>
    <td>B</td>
    <td>Description of B</td>
  </tr>
</table>
```

Good List Example:

```
<ul>
  <li>London</li>
  <li>Paris</li>
  <li>Tokyo</li>
</ul>
```

Never Skip the <title> Element

- ✓ The <title> element is required in HTML.
- ✓ The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.
- ✓ The <title> element:
    - defines a title in the browser toolbar
    - provides a title for the page when it is added to favorites
    - displays a title for the page in search-engine results

So, try to make the title as accurate and meaningful as possible:

```
<title>HTML Style Guide and Coding Conventions</title>
```

Omitting <html> and <body>?

An HTML page will validate without the <html> and <body> tags:

Example:

```
<!DOCTYPE html>
<head>
  <title>Page Title</title>
</head>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

- ✓ However, we strongly recommend to always add the <html> and <body> tags!
- ✓ Omitting <body> can produce errors in older browsers.
- ✓ Omitting <html> and <body> can also crash DOM and XML software.

Omitting <head>?

- ✓ The HTML <head> tag can also be omitted.
- ✓ Browsers will add all elements before <body>, to a default <head> element.

Example:

```
<!DOCTYPE html>
<html>
<title>Page Title</title>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

However, we recommend using the <head> tag.

Close Empty HTML Elements?

In HTML, it is optional to close empty elements.

Allowed:

```
<meta charset="utf-8">
```

Also Allowed:

```
<meta charset="utf-8" />
```

If you expect XML/XHTML software to access your page, keep the closing slash (/), because it is required in XML and XHTML.

Add the lang Attribute

You should always include the lang attribute inside the <html> tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

Example:

```
<!DOCTYPE html>
<html lang="en-us">
<head>
  <title>Page Title</title>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

```
</body>
</html>
```

Meta Data

To ensure proper interpretation and correct search engine indexing, both the language and the character encoding <meta charset="*charset*"> should be defined as early as possible in an HTML document:

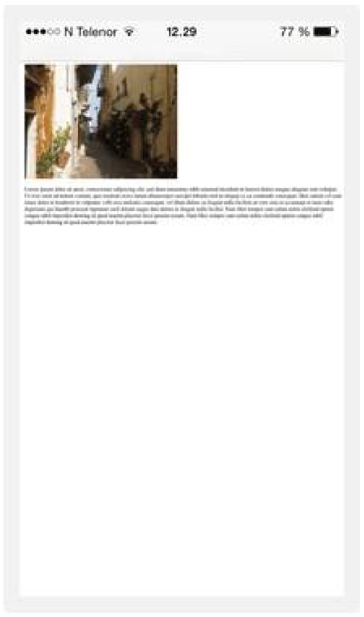
```
<!DOCTYPE html>
<html lang="en-us">
<head>
  <meta charset="UTF-8">
  <title>Page Title</title>
</head>
```

Setting The Viewport

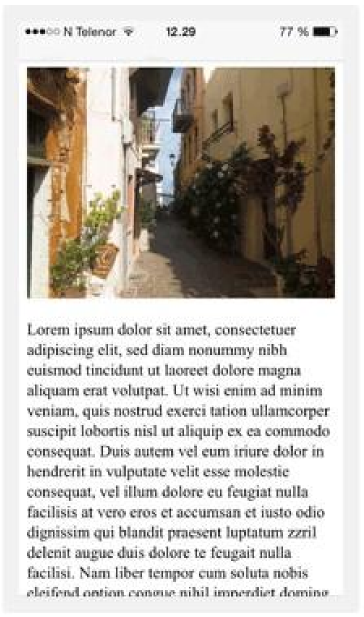
- ✓ The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.
- ✓ You should include the following <meta> element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

- ✓ This gives the browser instructions on how to control the page's dimensions and scaling.
- ✓ The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- ✓ The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.
- ✓ Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:
- ✓ **Tip:** If you are browsing this page with a phone or a tablet, you can click on the two links below to see the difference.



Without the viewport meta tag



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

With the viewport meta tag

HTML Comments

Short comments should be written on one line, like this:

```
<!-- This is a comment -->
```

Comments that spans more than one line, should be written like this:

```
<!--
  This is a long comment example. This is a long comment example.
  This is a long comment example. This is a long comment example.
-->
```

Long comments are easier to observe if they are indented with two spaces.

Using Style Sheets

Use simple syntax for linking to style sheets (the type attribute is not necessary):

```
<link rel="stylesheet" href="styles.css">
```

Short CSS rules can be written compressed, like this:

```
p.intro {font-family:Verdana;font-size:16em;}
```

Long CSS rules should be written over multiple lines:

```
body {
  background-color: lightgrey;
  font-family: "Arial Black", Helvetica, sans-serif;
  font-size: 16em;
  color: black;
}
```

- Place the opening bracket on the same line as the selector
- Use one space before the opening bracket
- Use two spaces of indentation
- Use semicolon after each property-value pair, including the last
- Only use quotes around values if the value contains spaces
- Place the closing bracket on a new line, without leading spaces

Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js">
```

Accessing HTML Elements with JavaScript

✓ Using "untidy" HTML code can result in JavaScript errors.
✓ These two JavaScript statements will produce different results:

Example:

```
getElementById("Demo").innerHTML = "Hello";

getElementById("demo").innerHTML = "Hello";
```

Use Lower Case File Names

✓ Some web servers (Apache, Unix) are case sensitive about file names: "london.jpg" cannot be accessed as "London.jpg".
✓ Other web servers (Microsoft, IIS) are not case sensitive: "london.jpg" can be accessed as "London.jpg".
✓ If you use a mix of uppercase and lowercase, you have to be aware of this.
✓ If you move from a case-insensitive to a case-sensitive server, even small errors will break your web!
✓ To avoid these problems, always use lowercase file names!

File Extensions

✓ HTML files should have a **.html** extension (**.htm** is allowed).
✓ CSS files should have a **.css** extension.
✓ JavaScript files should have a **.js** extension.

Differences Between .htm and .html?

- ✓ There is no difference between the .htm and .html file extensions!
- ✓ Both will be treated as HTML by any web browser and web server.

Default Filenames

- ✓ When a URL does not specify a filename at the end (like "https://www.w3schools.com/"), the server just adds a default filename, such as "index.html", "index.htm", "default.html", or "default.htm".
- ✓ If your server is configured only with "index.html" as the default filename, your file must be named "index.html", and not "default.html".
- ✓ However, servers can be configured with more than one default filename; usually you can set up as many default filenames as you want.

ENTITIES

Reserved characters in HTML must be replaced with character entities.

HTML Entities

- ✓ Some characters are reserved in HTML.
- ✓ If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.
- ✓ Character entities are used to display reserved characters in HTML.
- ✓ A character entity looks like this:

&*entity_name*;
OR
&#*entity_number*;

To display a less than sign (<) we must write: **&lt;** or **&#60;**

- ✓ **Advantage of using an entity name:** An entity name is easy to remember.
- ✓ **Disadvantage of using an entity name:** Browsers may not support all entity names, but the support for entity numbers is good.

Non-breaking Space

- ✓ A commonly used entity in HTML is the non-breaking space: **&nbsp;**
- ✓ A non-breaking space is a space that will not break into a new line.
- ✓ Two words separated by a non-breaking space will stick together (not break into a new line). This is handy when breaking the words might be disruptive.

Examples:

- § 10

- 10 km/h
- 10 PM

✓ Another common use of the non-breaking space is to prevent browsers from truncating spaces in HTML pages.

✓ If you write 10 spaces in your text, the browser will remove 9 of them. To add real spaces to your text, you can use the **&nbsp;** character entity.

Tip: The non-breaking hyphen (&#8209;) is used to define a hyphen character (-) that does not break into a new line.

Some Useful HTML Character Entities

Result	Description	Entity Name	Entity Number
	non-breaking space	&nbsp;	&#160;
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	double quotation mark	&quot;	&#34;
'	single quotation mark (apostrophe)	&apos;	&#39;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;

Note: Entity names are case sensitive.

Combining Diacritical Marks

✓ A diacritical mark is a "glyph" added to a letter.

✓ Some diacritical marks, like grave (̀) and acute (́) are called accents.

✓ Diacritical marks can appear both above and below a letter, inside a letter, and between two letters.

✓ Diacritical marks can be used in combination with alphanumeric characters to produce a character that is not present in the character set (encoding) used in the page.

Here are some examples:

Mark	Character	Construct	Result
̀	a	a&#768;	à

́	a	a&#769;	á
̂	a	a&#770;	c
̃	a	a&#771;	ã
̀	O	O&#768;	Ò
́	O	O&#769;	Ó
̂	O	O&#770;	Ô
̃	O	O&#771;	Õ

SYMBOLS

Symbols that are not present on your keyboard can also be added by using entities.

HTML Symbol Entities

- ✓ HTML entities were described in the previous chapter.
- ✓ Many mathematical, technical, and currency symbols, are not present on a normal keyboard.
- ✓ To add such symbols to an HTML page, you can use the entity name or the entity number (a decimal or a hexadecimal reference) for the symbol.

Example:
Display the euro sign, €, with an entity name, a decimal, and a hexadecimal value:

```
<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>
```

Will display as:
I will display €
I will display €
I will display €

Some Mathematical Symbols Supported by HTML

Char	**Number**	**Entity**	**Description**
∀	&#8704;	&forall;	FOR ALL
∂	&#8706;	&part;	PARTIAL DIFFERENTIAL
∃	&#8707;	&exist;	THERE EXISTS
∅	&#8709;	&empty;	EMPTY SETS
∇	&#8711;	&nabla;	NABLA
∈	&#8712;	&isin;	ELEMENT OF

∉	&#8713;	&notin;	NOT AN ELEMENT OF
∋	&#8715;	&ni;	CONTAINS AS MEMBER
∏	&#8719;	&prod;	N-ARY PRODUCT
∑	&#8721;	&sum;	N-ARY SUMMATION

Some Greek Letters Supported by HTML

Char	**Number**	**Entity**	**Description**
Α	&#913;	&Alpha;	GREEK CAPITAL LETTER ALPHA
Β	&#914;	&Beta;	GREEK CAPITAL LETTER BETA
Γ	&#915;	&Gamma;	GREEK CAPITAL LETTER GAMMA
Δ	&#916;	&Delta;	GREEK CAPITAL LETTER DELTA
Ε	&#917;	&Epsilon;	GREEK CAPITAL LETTER EPSILON
Ζ	&#918;	&Zeta;	GREEK CAPITAL LETTER ZETA

Some Other Entities Supported by HTML

Char	**Number**	**Entity**	**Description**
©	&#169;	&copy;	COPYRIGHT SIGN
®	&#174;	&reg;	REGISTERED SIGN
€	&#8364;	&euro;	EURO SIGN
™	&#8482;	&trade;	TRADEMARK
←	&#8592;	&larr;	LEFTWARDS ARROW
↑	&#8593;	&uarr;	UPWARDS ARROW
→	&#8594;	&rarr;	RIGHTWARDS ARROW
↓	&#8595;	&darr;	DOWNWARDS ARROW
♠	&#9824;	&spades;	BLACK SPADE SUIT
♣	&#9827;	&clubs;	BLACK CLUB SUIT
♥	&#9829;	&hearts;	BLACK HEART SUIT
♦	&#9830;	&diams;	BLACK DIAMOND SUIT

EMOJIS IN HTML

Emoji's are characters from the UTF-8 character set: 😄 😍 ❤

What are Emoji's?

- ✓ Emojis look like images, or icons, but they are not.
- ✓ They are letters (characters) from the UTF-8 (Unicode) character set.

The HTML charset Attribute

To display an HTML page correctly, a web browser must know the character set used in the page.
This is specified in the <meta> tag:

```
<meta charset="UTF-8">
```

If not specified, UTF-8 is the default character set in HTML.

UTF-8 Characters

Many UTF-8 characters cannot be typed on a keyboard, but they can always be displayed using numbers (called entity numbers):

- A is 65
- B is 66
- C is 67

Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<p>I will display A B C</p>
<p>I will display &#65; &#66; &#67;</p>

</body>
</html>
```

Example Explained

- ✓ The <meta charset="UTF-8"> element defines the character set.
- ✓ The characters A, B, and C, are displayed by the numbers 65, 66, and 67.
- ✓ To let the browser understand that you are displaying a character, you must start the entity number with &# and end it with ; (semicolon).

Emoji Characters

Emojis are also characters from the UTF-8 alphabet:

- 😄 is 128516
- 😍 is 128525

- 💗 is 128151

Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<h1>My First Emoji</h1>

<p>&#128512;</p>

</body>
</html>
```

Since Emojis are characters, they can be copied, displayed, and sized just like any other character in HTML.

Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<h1>Sized Emojis</h1>

<p style="font-size:48px">
&#128512; &#128516; &#128525; &#128151;
</p>

</body>
</html>
```

Some Emoji Symbols in UTF-8

Emoji	Value

	&#128507;
	&#128508;
	&#128509;
	&#128510;
	&#128511;
	&#128512;
	&#128513;
	&#128514;
	&#128515;
	&#128516;
	&#128517;

ENCODING (CHARACTER SETS)

To display an HTML page correctly, a web browser must know which character set to use.

From ASCII to UTF-8

- ✓ ASCII was the first character encoding standard. ASCII defined 128 different characters that could be used on the internet: numbers (0-9), English letters (A-Z), and some special characters like! $ + - ( ) @ < > .
- ✓ ISO-8859-1 was the default character set for HTML 4. This character set supported 256 different character codes. HTML 4 also supported UTF-8.
- ✓ ANSI (Windows-1252) was the original Windows character set. ANSI is identical to ISO-8859-1, except that ANSI has 32 extra characters.
- ✓ The HTML5 specification encourages web developers to use the UTF-8 character set, which covers almost all of the characters and symbols in the world!

The HTML charset Attribute

To display an HTML page correctly, a web browser must know the character set used in the page.
This is specified in the <meta> tag:

```
<meta charset="UTF-8">
```

Differences Between Character Sets

The following table displays the differences between the character sets described above:

Numb	ASCII	ANSI	8859	UTF-8	Description
32					space
33	!	!	!	!	exclamation mark
34	"	"	"	"	quotation mark
35	#	#	#	#	number sign
36	$	$	$	$	dollar sign
37	%	%	%	%	percent sign
38	&	&	&	&	ampersand
39	'	'	'	'	apostrophe
40	(	(	(	(	left parenthesis
41	)	)	)	)	right parenthesis
42	*	*	*	*	asterisk
43	+	+	+	+	plus sign
44	,	,	,	,	comma
45	-	-	-	-	hyphen-minus
46	.	.	.	.	full stop
47	/	/	/	/	solidus
48	0	0	0	0	digit zero
49	1	1	1	1	digit one
50	2	2	2	2	digit two
51	3	3	3	3	digit three
52	4	4	4	4	digit four
53	5	5	5	5	digit five
54	6	6	6	6	digit six
55	7	7	7	7	digit seven
56	8	8	8	8	digit eight
57	9	9	9	9	digit nine
58	:	:	:	:	colon
59	;	;	;	;	semicolon
60	<	<	<	<	less-than sign
61	=	=	=	=	equals sign
62	>	>	>	>	greater-than sign
63	?	?	?	?	question mark
64	@	@	@	@	commercial at
65	A	A	A	A	Latin capital letter A
66	B	B	B	B	Latin capital letter B
67	C	C	C	C	Latin capital letter C

68	D	D	D	D	Latin capital letter D
69	E	E	E	E	Latin capital letter E
70	F	F	F	F	Latin capital letter F
71	G	G	G	G	Latin capital letter G
72	H	H	H	H	Latin capital letter H
73	I	I	I	I	Latin capital letter I
74	J	J	J	J	Latin capital letter J
75	K	K	K	K	Latin capital letter K
76	L	L	L	L	Latin capital letter L
77	M	M	M	M	Latin capital letter M
78	N	N	N	N	Latin capital letter N
79	O	O	O	O	Latin capital letter O
80	P	P	P	P	Latin capital letter P
81	Q	Q	Q	Q	Latin capital letter Q
82	R	R	R	R	Latin capital letter R
83	S	S	S	S	Latin capital letter S
84	T	T	T	T	Latin capital letter T
85	U	U	U	U	Latin capital letter U
86	V	V	V	V	Latin capital letter V
87	W	W	W	W	Latin capital letter W
88	X	X	X	X	Latin capital letter X
89	Y	Y	Y	Y	Latin capital letter Y
90	Z	Z	Z	Z	Latin capital letter Z
91	[	[	[	[	left square bracket
92	\	\	\	\	reverse solidus
93	]	]	]	]	right square bracket
94	^	^	^	^	circumflex accent
95	_	_	_	_	low line
96	`	`	`	`	grave accent
97	a	a	a	a	Latin small letter a
98	b	b	b	b	Latin small letter b
99	c	c	c	c	Latin small letter c
100	d	d	d	d	Latin small letter d
101	e	e	e	e	Latin small letter e
102	f	f	f	f	Latin small letter f
103	g	g	g	g	Latin small letter g
104	h	h	h	h	Latin small letter h
105	i	i	i	i	Latin small letter i
106	j	j	j	j	Latin small letter j

107	k	k	k	k	Latin small letter k
108	l	l	l	l	Latin small letter l
109	m	m	m	m	Latin small letter m
110	n	n	n	n	Latin small letter n
111	o	o	o	o	Latin small letter o
112	p	p	p	p	Latin small letter p
113	q	q	q	q	Latin small letter q
114	r	r	r	r	Latin small letter r
115	s	s	s	s	Latin small letter s
116	t	t	t	t	Latin small letter t
117	u	u	u	u	Latin small letter u
118	v	v	v	v	Latin small letter v
119	w	w	w	w	Latin small letter w
120	x	x	x	x	Latin small letter x
121	y	y	y	y	Latin small letter y
122	z	z	z	z	Latin small letter z
123	{	{	{	{	left curly bracket
124	|	|	|	|	vertical line
125	}	}	}	}	right curly bracket
126	~	~	~	~	tilde
127	DEL				
128		€			euro sign
129		•	•	•	NOT USED
130		‚			single low-9 quotation mark
131		ƒ			Latin small letter f with hook
132		„			double low-9 quotation mark
133		…			horizontal ellipsis
134		†			dagger
135		‡			double dagger
136		ˆ			modifier letter circumflex accent
137		‰			per mille sign
138		Š			Latin capital letter S with caron
139		‹			single left-pointing angle quotation mark
140		Œ			Latin capital ligature OE
141		•	•	•	NOT USED
142		Ž			Latin capital letter Z with caron
143		•	•	•	NOT USED
144		•	•	•	NOT USED

145		‘			left single quotation mark
146		’			right single quotation mark
147		“			left double quotation mark
148		”			right double quotation mark
149		•			bullet
150		–			en dash
151		—			em dash
152		˜			small tilde
153		™			trade mark sign
154		š			Latin small letter s with caron
155		›			single right-pointing angle quotation mark
156		œ			Latin small ligature oe
157		•	•	•	NOT USED
158		ž			Latin small letter z with caron
159		Ÿ			Latin capital letter Y with diaeresis
160					no-break space
161		¡	¡	¡	inverted exclamation mark
162		¢	¢	¢	cent sign
163		£	£	£	pound sign
164		¤	¤	¤	currency sign
165		¥	¥	¥	yen sign
166		¦	¦	¦	broken bar
167		§	§	§	section sign
168		¨	¨	¨	diaeresis
169		©	©	©	copyright sign
170		ª	ª	ª	feminine ordinal indicator
171		«	«	«	left-pointing double angle quotation mark
172		¬	¬	¬	not sign
173					soft hyphen
174		®	®	®	registered sign
175		¯	¯	¯	macron
176		°	°	°	degree sign
177		±	±	±	plus-minus sign
178		²	²	²	superscript two
179		³	³	³	superscript three
180		´	´	´	acute accent
181		µ	µ	µ	micro sign

182		¶	¶	¶	pilcrow sign
183		·	·	·	middle dot
184		¸	¸	¸	cedilla
185		¹	¹	¹	superscript one
186		º	º	º	masculine ordinal indicator
187		»	»	»	right-pointing double angle quotation mark
188		¼	¼	¼	vulgar fraction one quarter
189		½	½	½	vulgar fraction one half
190		¾	¾	¾	vulgar fraction three quarters
191		¿	¿	¿	inverted question mark
192		À	À	À	Latin capital letter A with grave
193		Á	Á	Á	Latin capital letter A with acute
194		Â	Â	Â	Latin capital letter A with circumflex
195		Ã	Ã	Ã	Latin capital letter A with tilde
196		Ä	Ä	Ä	Latin capital letter A with diaeresis
197		Å	Å	Å	Latin capital letter A with ring above
198		Æ	Æ	Æ	Latin capital letter AE
199		Ç	Ç	Ç	Latin capital letter C with cedilla
200		È	È	È	Latin capital letter E with grave
201		É	É	É	Latin capital letter E with acute
202		Ê	Ê	Ê	Latin capital letter E with circumflex
203		Ë	Ë	Ë	Latin capital letter E with diaeresis
204		Ì	Ì	Ì	Latin capital letter I with grave
205		Í	Í	Í	Latin capital letter I with acute
206		Î	Î	Î	Latin capital letter I with circumflex
207		Ï	Ï	Ï	Latin capital letter I with diaeresis
208		Ð	Ð	Ð	Latin capital letter Eth
209		Ñ	Ñ	Ñ	Latin capital letter N with tilde
210		Ò	Ò	Ò	Latin capital letter O with grave
211		Ó	Ó	Ó	Latin capital letter O with acute
212		Ô	Ô	Ô	Latin capital letter O with circumflex
213		Õ	Õ	Õ	Latin capital letter O with tilde
214		Ö	Ö	Ö	Latin capital letter O with diaeresis
215		×	×	×	multiplication sign
216		Ø	Ø	Ø	Latin capital letter O with stroke
217		Ù	Ù	Ù	Latin capital letter U with grave
218		Ú	Ú	Ú	Latin capital letter U with acute
219		Û	Û	Û	Latin capital letter U with circumflex

220		Ü	Ü	Ü	Latin capital letter U with diaeresis
221		Ý	Ý	Ý	Latin capital letter Y with acute
222		Þ	Þ	Þ	Latin capital letter Thorn
223		ß	ß	ß	Latin small letter sharp s
224		à	à	à	Latin small letter a with grave
225		á	á	á	Latin small letter a with acute
226		â	â	â	Latin small letter a with circumflex
227		ã	ã	ã	Latin small letter a with tilde
228		ä	ä	ä	Latin small letter a with diaeresis
229		å	å	å	Latin small letter a with ring above
230		æ	æ	æ	Latin small letter ae
231		ç	ç	ç	Latin small letter c with cedilla
232		è	è	è	Latin small letter e with grave
233		é	é	é	Latin small letter e with acute
234		ê	ê	ê	Latin small letter e with circumflex
235		ë	ë	ë	Latin small letter e with diaeresis
236		ì	ì	ì	Latin small letter i with grave
237		í	í	í	Latin small letter i with acute
238		î	î	î	Latin small letter i with circumflex
239		ï	ï	ï	Latin small letter i with diaeresis
240		ð	ð	ð	Latin small letter eth
241		ñ	ñ	ñ	Latin small letter n with tilde
242		ò	ò	ò	Latin small letter o with grave
243		ó	ó	ó	Latin small letter o with acute
244		ô	ô	ô	Latin small letter o with circumflex
245		õ	õ	õ	Latin small letter o with tilde
246		ö	ö	ö	Latin small letter o with diaeresis
247		÷	÷	÷	division sign
248		ø	ø	ø	Latin small letter o with stroke
249		ù	ù	ù	Latin small letter u with grave
250		ú	ú	ú	Latin small letter u with acute
251		û	û	û	Latin small letter with circumflex
252		ü	ü	ü	Latin small letter u with diaeresis
253		ý	ý	ý	Latin small letter y with acute
254		þ	þ	þ	Latin small letter thorn
255		ÿ	ÿ	ÿ	Latin small letter y with diaeresis

The ASCII Character Set

- ✓ ASCII uses the values from 0 to 31 (and 127) for control characters.

- ✓ ASCII uses the values from 32 to 126 for letters, digits, and symbols.
- ✓ ASCII does not use the values from 128 to 255.

The ANSI Character Set (Windows-1252)

- ✓ ANSI is identical to ASCII for the values from 0 to 127.
- ✓ ANSI has a proprietary set of characters for the values from 128 to 159.
- ✓ ANSI is identical to UTF-8 for the values from 160 to 255.

The ISO-8859-1 Character Set

- ✓ ISO-8859-1 is identical to ASCII for the values from 0 to 127.
- ✓ ISO-8859-1 does not use the values from 128 to 159.
- ✓ ISO-8859-1 is identical to UTF-8 for the values from 160 to 255.

The UTF-8 Character Set

- ✓ UTF-8 is identical to ASCII for the values from 0 to 127.
- ✓ UTF-8 does not use the values from 128 to 159.
- ✓ UTF-8 is identical to both ANSI and 8859-1 for the values from 160 to 255.
- ✓ UTF-8 continues from the value 256 with more than 10 000 different characters.

UNIFORM RESOURCE LOCATORS

- ✓ A URL is another word for a web address.
- ✓ A URL can be composed of words (e.g. w3schools.com), or an Internet Protocol (IP) address (e.g. 192.68.20.50).
- ✓ Most people enter the name when surfing, because names are easier to remember than numbers.

URL - Uniform Resource Locator

- ✓ Web browsers request pages from web servers by using a URL.
- ✓ A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.
- ✓ A web address like https://www.ditrp.com/html/default.asp follows these syntax rules:

scheme://prefix.domain:port/path/filename

Explanation:

- **scheme** - defines the **type** of Internet service (most common is **http or https**)
- **prefix** - defines a domain **prefix** (default for http is **www**)

- **domain** - defines the Internet **domain name** (like w3schools.com)
- **port** - defines the **port number** at the host (default for http is **80**)
- **path** - defines a **path** at the server (If omitted: the root directory of the site)
- **filename** - defines the name of a document or resource

Common URL Schemes

The table below lists some common schemes:

Scheme	**Short for**	**Used for**
http	HyperText Transfer Protocol	Common web pages. Not encrypted
https	Secure HyperText Transfer Protocol	Secure web pages. Encrypted
ftp	File Transfer Protocol	Downloading or uploading files
file		A file on your computer

URL Encoding

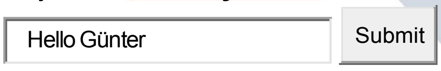
✓ URLs can only be sent over the Internet using the ASCII character-set. If a URL contains characters outside the ASCII set, the URL has to be converted.

✓ URL encoding converts non-ASCII characters into a format that can be transmitted over the Internet.

✓ URL encoding replaces non-ASCII characters with a "%" followed by hexadecimal digits.

✓ URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign, or %20.

Try It Yourself



Hello Günter

✓ If you click "Submit", the browser will URL encode the input before it is sent to the server.

✓ A page at the server will display the received input.

✓ Try some other input and click Submit again.

ASCII Encoding Examples

Your browser will encode input, according to the character-set used in your page.

The default character-set in HTML5 is UTF-8.

Character	**From Windows-1252**	**From UTF-8**
€	%80	%E2%82%AC

£	%A3	%C2%A3
©	%A9	%C2%A9
®	%AE	%C2%AE
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85

HTML Versus XHTML

XHTML is a stricter, more XML-based version of HTML.

What is XHTML?

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is a stricter, more XML-based version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

Why XHTML?

✓ XML is a markup language where all documents must be marked up correctly (be "well-formed").

✓ XHTML was developed to make HTML more extensible and flexible to work with other data formats (such as XML). In addition, browsers ignore errors in HTML pages, and try to display the website even if it has some errors in the markup. So XHTML comes with a much stricter error handling.

✓ If you want to study XML, please read our XML Tutorial.

The Most Important Differences from HTML

- <!DOCTYPE> is **mandatory**
- The xmlns attribute in <html> is **mandatory**
- <html>, <head>, <title>, and <body> are **mandatory**
- Elements must always be **properly nested**
- Elements must always be **closed**
- Elements must always be in **lowercase**
- Attribute names must always be in **lowercase**
- Attribute values must always be **quoted**
- Attribute minimization is **forbidden**

XHTML - <!DOCTYPE.....> Is Mandatory

- ✓ An XHTML document must have an XHTML <!DOCTYPE> declaration.
- ✓ The <html>, <head>, <title>, and <body> elements must also be present, and the xmlns attribute in <html> must specify the xml namespace for the document.

Example:
Here is an XHTML document with a minimum of required tags:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Title of document</title>
</head>
<body>

  some content here...

</body>
</html>
```

XHTML Elements Must be Properly Nested

In XHTML, elements must always be properly nested within each other, likethis:

Correct:

```
<b><i>Some text</i></b>
```

Wrong:

```
<b><i>Some text</b></i>
```

XHTML Elements Must Always be Closed

In XHTML, elements must always be closed, like this:

Correct:

```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```

Wrong:

```
<p>This is a paragraph
<p>This is another paragraph
```

XHTML Empty Elements Must Always be Closed

In XHTML, empty elements must always be closed, like this:

Correct:

```
A break: <br />
A horizontal rule: <hr />
An image: <img src="happy.gif" alt="Happy face" />
```

Wrong:

```
A break: <br>
A horizontal rule: <hr>
An image: <img src="happy.gif" alt="Happy face">
```

XHTML Elements Must be in Lowercase

In XHTML, element names must always be in lowercase, like this:

Correct:

```
<body>
<p>This is a paragraph</p>
</body>
```

Wrong:

```
<BODY>
<P>This is a paragraph</P>
</BODY>
```

XHTML Attribute Names Must be in Lowercase

In XHTML, attribute names must always be in lowercase, like this:

Correct:

```
<a href="https://www.ditrp.com/html/">Visit our HTML tutorial</a>
```

Wrong:

```
<a HREF="https://www.ditrp.com/html/">Visit our HTML tutorial</a>
```

XHTML Attribute Values Must be Quoted

In XHTML, attribute values must always be quoted, like this:

Correct:

```
<a href="https://www.ditrp.com/html/">Visit our HTML tutorial</a>
```

Wrong:
```
<a href=https://www.ditrp.com/html/>Visit our HTML tutorial</a>
```
XHTML Attribute Minimization is Forbidden
In XHTML, attribute minimization is forbidden:

Correct:
```
<input type="checkbox" name="vehicle" value="car" checked="checked" />
<input type="text" name="lastname" disabled="disabled" />
```
Wrong:
```
<input type="checkbox" name="vehicle" value="car" checked />
<input type="text" name="lastname" disabled />
```

HTML FORMS
An HTML form is used to collect user input. The user input is most often sent to a server for processing.

Example:

First name:

John

Last name:

Doe

Submit

The <form> Element
The HTML <form> element is used to create an HTML form for user input:
```
<form>
.
form elements
.
</form>
```
- ✓ The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.
- ✓ All the different form elements are covered in this chapter: HTML Form Elements.

The <input> Element
- ✓ The HTML <input> element is the most used form element.

- ✓ An <input> element can be displayed in many ways, depending on the type attribute.

Here are some examples:

Type	Description
<input type="text">	Displays a single-line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)
<input type="submit">	Displays a submit button (for submitting the form)
<input type="button">	Displays a clickable button

All the different input types are covered in this chapter: HTML Input Types.

Text Fields

The <input type="text"> defines a single-line input field for text input.

Example:

A form with input fields for text:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Note: The form itself is not visible. Also note that the default width of an input field is 20 characters.

The <label> Element

- ✓ Notice the use of the <label> element in the example above.
- ✓ The <label> tag defines a label for many form elements.
- ✓ The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.

- ✓ The <label> element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.
- ✓ The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

Radio Buttons

The <input type="radio"> defines a radio button.
Radio buttons let a user select ONE of a limited number of choices.

Example:
A form with radio buttons:

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

This is how the HTML code above will be displayed in a browser:
Choose your favorite Web language:

○ HTML

○ CSS

○ JavaScript

Checkboxes

The <input type="checkbox"> defines a **checkbox**.
Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example:
A form with checkboxes:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

☐ I have a bike

☐ I have a car

☐ I have a boat

The Submit Button

The <input type="submit"> defines a button for submitting the form data to a form-handler.
The form-handler is typically a file on the server with a script for processing input data.
The form-handler is specified in the form's action attribute.
Example:
A form with a submit button:

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

This is how the HTML code above will be displayed in a browser:
First name:

John

Last name:

Doe

Submit

The Name Attribute for <input>

✓ Notice that each input field must have a name attribute to be submitted.

- ✓ If the name attribute is omitted, the value of the input field will not be sent at all.

Example:

This example will not submit the value of the "First name" input field:

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" value="John"><br><br>
  <input type="submit" value="Submit">
</form>
```

FORM ATTRIBUTES

This chapter describes the different attributes for the HTML <form> element.

The Action Attribute

- ✓ The action attribute defines the action to be performed when the form is submitted.
- ✓ Usually, the form data is sent to a file on the server when the user clicks on the submit button.
- ✓ In the example below, the form data is sent to a file called "action_page.php". This file contains a server-side script that handles the form data:

Example:

On submit, send form data to "action_page.php":

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

Tip: If the action attribute is omitted, the action is set to the current page.

The Target Attribute

- ✓ The target attribute specifies where to display the response that is received after submitting the form.
- ✓ The target attribute can have one of the following values:

Value	Description

_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

The default value is _self which means that the response will open in the current window.

Example:

Here, the submitted result will open in a new browser tab:

```
<form action="/action_page.php" target="_blank">
```

The Method Attribute

- ✓ The method attribute specifies the HTTP method to be used when submitting the form data.
- ✓ The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
- ✓ The default HTTP method when submitting form data is GET.

Example:

This example uses the GET method when submitting the form data:

```
<form action="/action_page.php" method="get">
```

Example:

This example uses the POST method when submitting the form data:

```
<form action="/action_page.php" method="post">
```

Notes on GET:

- Appends the form data to the URL, in name/value pairs
- NEVER use GET to send sensitive data! (the submitted form data is visible in the URL!)
- The length of a URL is limited (2048 characters)
- Useful for form submissions where a user wants to bookmark the result
- GET is good for non-secure data, like query strings in Google

Notes on POST:

- Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)
- POST has no size limitations, and can be used to send large amounts of data.
- Form submissions with POST cannot be bookmarked

Tip: Always use POST if the form data contains sensitive or personal information!

The Autocomplete Attribute

- ✓ The autocomplete attribute specifies whether a form should have autocomplete on or off.
- ✓ When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

Example:
A form with autocomplete on:

```
<form action="/action_page.php" autocomplete="on">
```

The Novalidate Attribute

- ✓ The novalidate attribute is a boolean attribute.
- ✓ When present, it specifies that the form-data (input) should not be validated when submitted.

Example:
A form with a novalidate attribute:

```
<form action="/action_page.php" novalidate>
```

FORM ELEMENTS

This chapter describes all the different HTML form elements.

The HTML <form> Elements

The HTML <form> element can contain one or more of the following form elements:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

The <input> Element

- ✓ One of the most used form element is the <input> element.

- ✓ The <input> element can be displayed in several ways, depending on the type attribute.

Example:

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

All the different values of the type attribute are covered in the next chapter: HTML Input Types.

The <label> Element

- ✓ The <label> element defines a label for several form elements.
- ✓ The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.
- ✓ The <label> element also help users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.
- ✓ The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

The <select> Element

The <select> element defines a drop-down list:

Example:

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

- ✓ The <option> elements defines an option that can be selected.
- ✓ By default, the first item in the drop-down list is selected.
- ✓ To define a pre-selected option, add the selected attribute to the option:

Example:

```
<option value="fiat" selected>Fiat</option>
```

Visible Values:

Use the size attribute to specify the number of visible values:

Example:

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="3">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

Allow Multiple Selections:

Use the multiple attribute to allow the user to select more than one value:

Example:

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

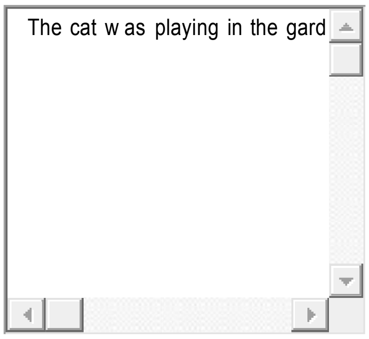
The <textarea> Element

The <textarea> element defines a multi-line input field (a text area):

Example:

```
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
```

- ✓ The rows attribute specifies the visible number of lines in a text area.
- ✓ The cols attribute specifies the visible width of a text area.
- ✓ This is how the HTML code above will be displayed in a browser:



You can also define the size of the text area by using CSS:
Example:

```
<textarea name="message" style="width:200px; height:600px;">
The cat was playing in the garden.
</textarea>
```

The <button> Element

The <button> element defines a clickable button:
Example:

```
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
```

This is how the HTML code above will be displayed in a browser:
Click Me!
Note: Always specify the type attribute for the button element. Different browsers may use different default types for the button element.

The <fieldset> and <legend> Elements

- ✓ The <fieldset> element is used to group related data in a form.
- ✓ The <legend> element defines a caption for the <fieldset> element.

Example:

```
<form action="/action_page.php">
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
```

This is how the HTML code above will be displayed in a browser:

Personalia:

First name:

John

Last name:

Doe

Submit

The <datalist> Element

- ✓ The <datalist> element specifies a list of pre-defined options for an <input> element.
- ✓ Users will see a drop-down list of the pre-defined options as they input data.
- ✓ The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

Example:

```
<form action="/action_page.php">
 <input list="browsers">
 <datalist id="browsers">
  <option value="Internet Explorer">
  <option value="Firefox">
  <option value="Chrome">
  <option value="Opera">
  <option value="Safari">
 </datalist>
</form>
```

The <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

Example:

Perform a calculation and show the result in an <output> element:

```
<form action="/action_page.php"
 oninput="x.value=parseInt(a.value)+parseInt(b.value)">
 0
 <input type="range" id="a" name="a" value="50">
 100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
  <input type="submit">
</form>
```

HTML Form Elements

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<button>	Defines a clickable button
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation

HTML INPUT TYPES

This chapter describes the different types for the HTML <input> element.

HTML Input Types

Here are the different input types you can use in HTML:

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">

- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

Tip: The default value of the type attribute is "text".

Input Type Text

<input type="text"> defines a **single-line text input field**:

Example:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

This is how the HTML code above will be displayed in a browser:

First name:

Last name:

Input Type Password

<input type="password"> defines a **password field**:

Example:

```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
</form>
```

This is how the HTML code above will be displayed in a browser:
Username:

Password:

The characters in a password field are masked (shown as asterisks or circles).

Input Type Submit

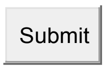
- ✓ <input type="submit"> defines a button for **submitting** form data to a **form-handler**.
- ✓ The form-handler is typically a server page with a script for processing input data.
- ✓ The form-handler is specified in the form's action attribute:

Example:
```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

This is how the HTML code above will be displayed in a browser:
First name:

John
Last name:

Doe


Submit
If you omit the submit button's value attribute, the button will get a default text:

Example:
```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit">
</form>
```

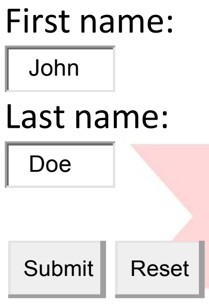
Input Type Reset

<input type="reset"> defines a **reset button** that will reset all form values to their default values:

Example:

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```

This is how the HTML code above will be displayed in a browser:



First name:

John

Last name:

Doe

Submit Reset

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

Input Type Radio

<input type="radio"> defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices:

Example:

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

○ HTML

○ CSS

○ JavaScript

Input Type Checkbox

<input type="checkbox"> defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example:

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

☐ I have a bike

☐ I have a car

☐ I have a boat

Input Type Button

<input type="button"> defines a **button**:

Example:

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```

This is how the HTML code above will be displayed in a browser:

Input Type Color

The <input type="color"> is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

Example:
```
<form>
  <label for="favcolor">Select your favorite color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

Input Type Date

The <input type="date"> is used for input fields that should contain a date. Depending on browser support, a date picker can show up in the input field.
Example:
```
<form>
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday">
</form>
```

You can also use the min and max attributes to add restrictions to dates:

Example:
```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-
31"><br><br>
  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-02">
</form>
```

Input Type Datetime-local

- ✓ The <input type="datetime-local"> specifies a date and time input field, with no time zone.
- ✓ Depending on browser support, a date picker can show up in the input field.

Example:
```
<form>
  <label for="birthdaytime">Birthday (date and time):</label>
  <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```

Input Type Email

- ✓ The <input type="email"> is used for input fields that should contain an e-mail address.
- ✓ Depending on browser support, the e-mail address can be automatically validated when submitted.
- ✓ Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

Example:

```
<form>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email">
</form>
```

Input Type File

The <input type="file"> defines a file-select field and a "Browse" button for file uploads.

Example:

```
<form>
  <label for="myfile">Select a file:</label>
  <input type="file" id="myfile" name="myfile">
</form>
```

Input Type Hidden

- ✓ The <input type="hidden"> defines a hidden input field (not visible to a user).
- ✓ A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.
- ✓ A hidden field often stores what database record that needs to be updated when the form is submitted.

Note: While the value is not displayed to the user in the page's content, it is visible (and can be edited) using any browser's developer tools or "View Source" functionality. Do not use hidden inputs as a form of security!

Example:

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
```

Input Type Month

- ✓ The <input type="month"> allows the user to select a month and year.
- ✓ Depending on browser support, a date picker can show up in the input field.

Example:

```
<form>
  <label for="bdaymonth">Birthday (month and year):</label>
  <input type="month" id="bdaymonth" name="bdaymonth">
</form>
```

Input Type Number

- ✓ The <input type="number"> defines a **numeric** input field.
- ✓ You can also set restrictions on what numbers are accepted.
- ✓ The following example displays a numeric input field, where you can enter a value from 1 to 5:

Example:

```
<form>
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

Input Restrictions

Here is a list of some common input restrictions:

Attribute	**Description**
checked	Specifies that an input field should be pre-selected when the page loads (for type="checkbox" or type="radio")
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field
pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

You will learn more about input restrictions in the next chapter.

The following example displays a numeric input field, where you can enter a value from 0 to 100, in steps of 10. The default value is 30:

Example:
```
<form>
  <label for="quantity">Quantity:</label>
  <input type="number" id="quantity" name="quantity" min="0" max="100" st
ep="10" value="30">
</form>
```

Input Type Range
The <input type="range"> defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the min, max, and step attributes:

Example:
```
<form>
  <label for="vol">Volume (between 0 and 50):</label>
  <input type="range" id="vol" name="vol" min="0" max="50">
</form>
```

Input Type Search
The <input type="search"> is used for search fields (a search field behaves like a regular text field).

Example:
```
<form>
  <label for="gsearch">Search Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```

Input Type Tel
The <input type="tel"> is used for input fields that should contain a telephone number.

Example:
```
<form>
  <label for="phone">Enter your phone number:</label>
  <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

Input Type Time

The <input type="time"> allows the user to select a time (no time zone). Depending on browser support, a time picker can show up in the input field.

Example:

```
<form>
  <label for="appt">Select a time:</label>
  <input type="time" id="appt" name="appt">
</form>
```

Input Type Url

- ✓ The <input type="url"> is used for input fields that should contain a URL address.
- ✓ Depending on browser support, the url field can be automatically validated when submitted.
- ✓ Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

Example:

```
<form>
  <label for="homepage">Add your homepage:</label>
  <input type="url" id="homepage" name="homepage">
</form>
```

Input Type Week

- ✓ The <input type="week"> allows the user to select a week and year.
- ✓ Depending on browser support, a date picker can show up in the input field.

Example:

```
<form>
  <label for="week">Select a week:</label>
  <input type="week" id="week" name="week">
</form>
```

HTML INPUT ATTRIBUTES

This chapter describes the different attributes for the HTML <input> element.

The value Attribute

The input value attribute specifies an initial value for an input field:

Example:

Input fields with initial (default) values:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The readonly Attribute

- ✓ The input readonly attribute specifies that an input field is read-only.
- ✓ A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).
- ✓ The value of a read-only input field will be sent when submitting the form!

Example:

A read-only input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" readonly><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The disabled Attribute

- ✓ The input disabled attribute specifies that an input field should be disabled.
- ✓ A disabled input field is unusable and un-clickable.
- ✓ The value of a disabled input field will not be sent when submitting the form!

Example:

A disabled input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```

The size Attribute

- ✓ The input size attribute specifies the visible width, in characters, of an input field.
- ✓ The default value for size is 20.

Note: The size attribute works with the following input types: text, search, tel, url, email, and password.

Example:
Set a width for an input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" size="4">
</form>
```

The maxlength Attribute

The input maxlength attribute specifies the maximum number of characters allowed in an input field.
Note: When a maxlength is set, the input field will not accept more than the specified number of characters. However, this attribute does not provide any feedback. So, if you want to alert the user, you must write JavaScript code.

Example:
Set a maximum length for an input field:

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>
```

The min and max Attributes

- ✓ The input min and max attributes specify the minimum and maximum values for an input field.
- ✓ The min and max attributes work with the following input types: number, range, date, datetime-local, month, time and week.

Tip: Use the max and min attributes together to create a range of legal values.

Example:
Set a max date, a min date, and a range of legal values:

```
<form>
  <label for="datemax">Enter a date before 1980-01-01:</label>
  <input type="date" id="datemax" name="datemax" max="1979-12-
31"><br><br>

  <label for="datemin">Enter a date after 2000-01-01:</label>
  <input type="date" id="datemin" name="datemin" min="2000-01-
02"><br><br>

  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
</form>
```

The multiple Attribute

- ✓ The input multiple attribute specifies that the user is allowed to enter more than one value in an input field.
- ✓ The multiple attribute works with the following input types: email, and file.

Example:
A file upload field that accepts multiple values:

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

The pattern Attribute

- ✓ The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.
- ✓ The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

Tip: Use the global title attribute to describe the pattern to help the user.
Tip: Learn more about regular expressions in our JavaScript tutorial.

Example:
An input field that can contain only three letters (no numbers or special characters):

```
<form>
  <label for="country_code">Country code:</label>
  <input type="text" id="country_code" name="country_code"
  pattern="[A-Za-z]{3}" title="Three letter country code">
</form>
```

The placeholder Attribute

- ✓ The input placeholder attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).
- ✓ The short hint is displayed in the input field before the user enters a value.
- ✓ The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

Example:

An input field with a placeholder text:

```
<form>
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone"
  placeholder="123-45-678"
  pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```

The required Attribute

- ✓ The input required attribute specifies that an input field must be filled out before submitting the form.
- ✓ The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

Example:

A required input field:

```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
</form>
```

The step Attribute

- ✓ The input step attribute specifies the legal number intervals for an input field.
- ✓ Example: if step="3", legal numbers could be -3, 0, 3, 6, etc.

Tip: This attribute can be used together with the max and min attributes to create a range of legal values.
The step attribute works with the following input types: number, range, date, datetime-local, month, time and week.

Example:
An input field with a specified legal number intervals:
```
<form>
  <label for="points">Points:</label>
  <input type="number" id="points" name="points" step="3">
</form>
```

Note: Input restrictions are not foolproof, and JavaScript provides many ways to add illegal input. To safely restrict input, it must also be checked by the receiver (the server)!

The autofocus Attribute
The input autofocus attribute specifies that an input field should automatically get focus when the page loads.

Example:
Let the "First name" input field automatically get focus when the page loads:
```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" autofocus><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname">
</form>
```

The height and width Attributes
The input height and width attributes specify the height and width of an <input type="image"> element.

Tip: Always specify both the height and width attributes for images. If height and width are set, the space required for the image is reserved when the page is loaded. Without these attributes, the browser does not know the size of the image, and cannot reserve the appropriate space to it. The effect will be that the page layout will change during loading (while the images load).

Example:
Define an image as the submit button, with height and width attributes:

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="image" src="img_submit.gif" alt="Submit" width="48" height="4
8">
</form>
```

The list Attribute

The input list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

Example:

An <input> element with pre-defined values in a <datalist>:

```
<form>
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

The autocomplete Attribute

- ✓ The input autocomplete attribute specifies whether a form or an input field should have autocomplete on or off.
- ✓ Autocomplete allows the browser to predict the value. When a user starts to type in a field, the browser should display options to fill in the field, based on earlier typed values.
- ✓ The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

Example:

An HTML form with autocomplete on, and off for one input field:

```
<form action="/action_page.php" autocomplete="on">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" autocomplete="off"><br><br>
  <input type="submit" value="Submit">
</form>
```

HTML INPUT FORM* ATTRIBUTES

This chapter describes the different form* attributes for the HTML <input> element.

The form Attribute

- ✓ The input form attribute specifies the form the <input> element belongs to.
- ✓ The value of this attribute must be equal to the id attribute of the <form> element it belongs to.

Example:

An input field located outside of the HTML form (but still a part of the form):

```
<form action="/action_page.php" id="form1">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>

<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname" form="form1">
```

The formaction Attribute

- ✓ The input formaction attribute specifies the URL of the file that will process the input when the form is submitted.
- ✓ **Note:** This attribute overrides the action attribute of the <form> element.
- ✓ The formaction attribute works with the following input types: submit and image.

Example:

An HTML form with two submit buttons, with different actions:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formaction="/action_page2.php" value="Submit as
Admin">
</form>
```

The formenctype Attribute

- ✓ The input formenctype attribute specifies how the form-data should be encoded when submitted (only for forms with method="post").
- ✓ **Note:** This attribute overrides the enctype attribute of the <form> element.
- ✓ The formenctype attribute works with the following input types: submit and image.

Example:
A form with two submit buttons. The first sends the form-data with default encoding, the second sends the form-data encoded as "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

The formmethod Attribute

- ✓ The input formmethod attribute defines the HTTP method for sending form-data to the action URL.
- ✓ **Note:** This attribute overrides the method attribute of the <form> element.
- ✓ The formmethod attribute works with the following input types: submit and image.
- ✓ The form-data can be sent as URL variables (method="get") or as an HTTP post transaction (method="post").

Notes on the "get" method:

- This method appends the form-data to the URL in name/value pairs
- This method is useful for form submissions where a user want to bookmark the result

- There is a limit to how much data you can place in a URL (varies between browsers), therefore, you cannot be sure that all of the form-data will be correctly transferred
- Never use the "get" method to pass sensitive information! (password or other sensitive information will be visible in the browser's address bar)

Notes on the "post" method:

- This method sends the form-data as an HTTP post transaction
- Form submissions with the "post" method cannot be bookmarked
- The "post" method is more robust and secure than "get", and "post" does not have size limitations

Example:
A form with two submit buttons. The first sends the form-data with method="get". The second sends the form-data with method="post":

```
<form action="/action_page.php" method="get">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit using GET">
  <input type="submit" formmethod="post" value="Submit using POST">
</form>
```

The formtarget Attribute

- ✓ The input formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.
- ✓ **Note:** This attribute overrides the target attribute of the <form> element.
- ✓ The formtarget attribute works with the following input types: submit and image.

Example:
A form with two submit buttons, with different target windows:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formtarget="_blank" value="Submit to a new
```

```
window/tab">
</form>
```

The formnovalidate Attribute

- ✓ The input formnovalidate attribute specifies that an <input> element should not be validated when submitted.
- ✓ **Note:** This attribute overrides the novalidate attribute of the <form> element.
- ✓ The formnovalidate attribute works with the following input types: submit.

Example:
A form with two submit buttons (with and without validation):

```
<form action="/action_page.php">
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formnovalidate="formnovalidate"
  value="Submit without validation">
</form>
```

The novalidate Attribute

- ✓ The novalidate attribute is a <form> attribute.
- ✓ When present, novalidate specifies that all of the form-data should not be validated when submitted.

Example:
Specify that no form-data should be validated on submit:

```
<form action="/action_page.php" novalidate>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
</form>
```

HTML Form and Input Elements

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control

HTML CANVAS GRAPHICS

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

What is HTML Canvas?

- ✓ The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- ✓ The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- ✓ Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

Browser Support

The numbers in the table specify the first browser version that fully supports the <canvas> element.

Element					
<canvas>	4.0	9.0	2.0	3.1	9.0

Canvas Examples

A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.

The markup looks like this:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Note: Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

Here is an example of a basic, empty canvas:

Example:

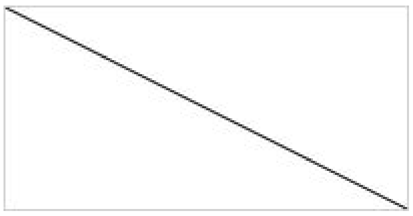
```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>
```

Add a JavaScript

After creating the rectangular canvas area, you must add a JavaScript to do the drawing.

Here are some examples:

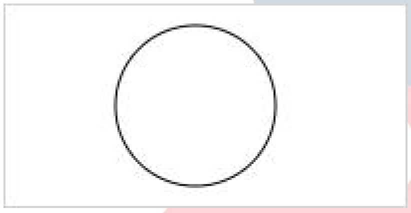
Draw a Line



Example:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0, 0);
ctx.lineTo(200, 100);
ctx.stroke();
</script>
```

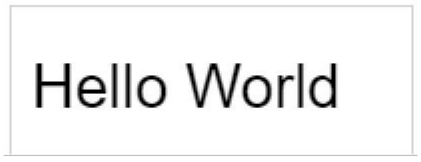
Draw a Circle



Example:

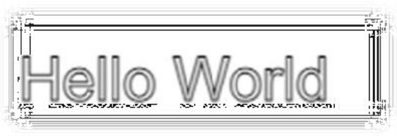
```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95, 50, 40, 0, 2 * Math.PI);
ctx.stroke();
</script>
```

Draw a Text



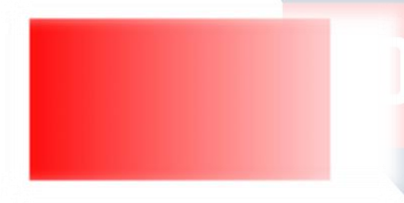
Example:
```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World", 10, 50);
</script>
```

Stroke Text



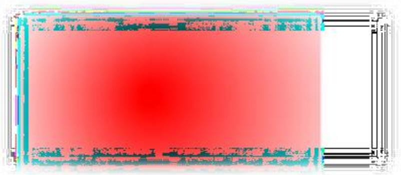
Example:
```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World", 10, 50);
</script>
```

Draw Linear Gradient



Example:
```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createLinearGradient(0, 0, 200, 0);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

Draw Circular Gradient



Example:

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");

// Create gradient
var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
grd.addColorStop(0, "red");
grd.addColorStop(1, "white");

// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10, 10, 150, 80);
</script>
```

Draw Image

```
<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
var img = document.getElementById("scream");
ctx.drawImage(img, 10, 10);
</script>
```

HTML SVG GRAPHICS

SVG defines vector-based graphics in XML format.

What is SVG?

- SVG stands for Scalable Vector Graphics

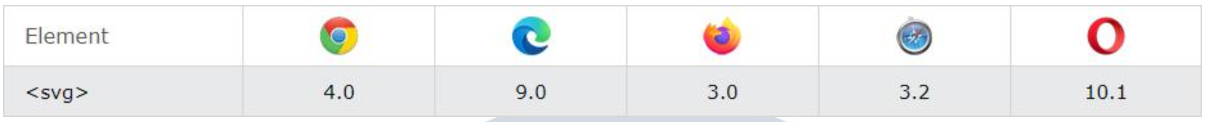
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

The HTML <svg> Element

The HTML <svg> element is a container for SVG graphics.
SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

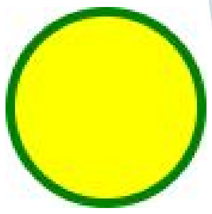
Browser Support

The numbers in the table specify the first browser version that fully supports the <svg> element.



Element					
<svg>	4.0	9.0	3.0	3.2	10.1

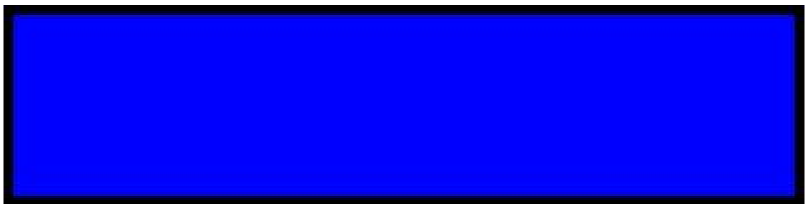
SVG Circle



Example:

```
<!DOCTYPE html>
<html>
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" />
</svg>

</body>
</html>
```

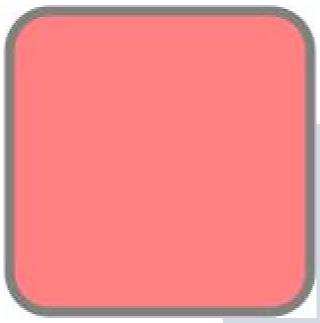
SVG Rectangle



Example:

```
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
width:10;stroke:rgb(0,0,0)" />
</svg>
```

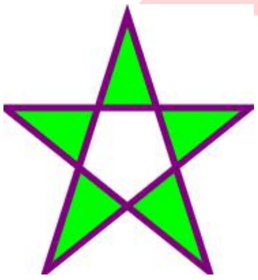
SVG Rounded Rectangle



Example:

```
<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
</svg>
```

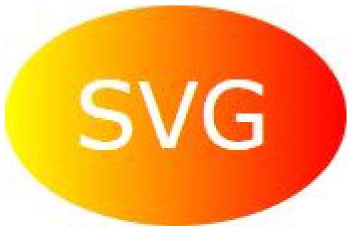
SVG Star



Example:

```
<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>
```

SVG Logo



Example:

```
<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%" style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%" style="stop-color:rgb(255,0,0);stop-opacity:1" />
    </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-
family="Verdana" x="50" y="86">SVG</text>
Sorry, your browser does not support inline SVG.
</svg>
```

Differences Between SVG and Canvas

- ✓ SVG is a language for describing 2D graphics in XML.
- ✓ Canvas draws 2D graphics, on the fly (with a JavaScript).
- ✓ SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- ✓ In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- ✓ Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

Canvas	**SVG**

• Resolution dependent • No support for event handlers • Poor text rendering capabilities • You can save the resulting image as .png or .jpg • Well suited for graphic-intensive games	• Resolution independent • Support for event handlers • Best suited for applications with large rendering areas (Google Maps) • Slow rendering if complex (anything that uses the DOM a lot will be slow) • Not suited for game applications

HTML MULTIMEDIA

Multimedia on the web is sound, music, videos, movies, and animations.

What is Multimedia?

- ✓ Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.
- ✓ Web pages often contain multimedia elements of different types and formats.

Browser Support

- ✓ The first web browsers had support for text only, limited to a single font in a single color.
- ✓ Later came browsers with support for colors, fonts, images, and multimedia!

Multimedia Formats

- ✓ Multimedia elements (like audio or video) are stored in media files.
- ✓ The most common way to discover the type of a file, is to look at the file extension.
- ✓ Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.

Common Video Formats

- There are many video formats out there.
- The MP4, WebM, and Ogg formats are supported by HTML.
- The MP4 format is recommended by YouTube.

Format	**File**	**Description**
MPEG	.mpg .mpeg	MPEG. Developed by the Moving Pictures Expert Group. The first popular video format on the web. Not supported anymore in HTML.
AVI	.avi	AVI (Audio Video Interleave). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
WMV	.wmv	WMV (Windows Media Video). Developed by Microsoft. Commonly used in video cameras and TV hardware. Plays well on Windows computers, but not in web browsers.
QuickTime	.mov	QuickTime. Developed by Apple. Commonly used in video cameras and TV hardware. Plays well on Apple computers, but not in web browsers.
RealVideo	.rm .ram	RealVideo. Developed by Real Media to allow video streaming with low bandwidths. Does not play in web browsers.
Flash	.swf .flv	Flash. Developed by Macromedia. Often requires an extra component (plug-in) to play in web browsers.
Ogg	.ogg	Theora Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
WebM	.webm	WebM. Developed by Mozilla, Opera, Adobe, and Google. Supported by HTML.
MPEG-4 or MP4	.mp4	MP4. Developed by the Moving Pictures Expert Group. Commonly used in video cameras and TV hardware. Supported by all browsers and recommended by YouTube.

Note: Only MP4, WebM, and Ogg video are supported by the HTML standard.

Common Audio Formats

✓ MP3 is the best format for compressed recorded music. The term MP3 has become synonymous with digital music.

If your website is about recorded music, MP3 is the choice.

Format	**File**	**Description**
MIDI	.mid .midi	MIDI (Musical Instrument Digital Interface). Main format for all electronic music devices like synthesizers and PC sound cards. MIDI files do not contain sound, but digital notes that can be played by electronics. Plays well on all computers and music hardware, but not in web browsers.
RealAudio	.rm .ram	RealAudio. Developed by Real Media to allow streaming of audio with low bandwidths. Does not play in web browsers.
WMA	.wma	WMA (Windows Media Audio). Developed by Microsoft. Plays well on Windows computers, but not in web browsers.
AAC	.aac	AAC (Advanced Audio Coding). Developed by Apple as the default format for iTunes. Plays well on Apple computers, but not in web browsers.
WAV	.wav	WAV. Developed by IBM and Microsoft. Plays well on Windows, Macintosh, and Linux operating systems. Supported by HTML.
Ogg	.ogg	Ogg. Developed by the Xiph.Org Foundation. Supported by HTML.
MP3	.mp3	MP3 files are actually the sound part of MPEG files. MP3 is the most popular format for music players. Combines good compression (small files) with high quality. Supported by all browsers.
MP4	.mp4	MP4 is a video format, but can also be used for audio. Supported by all browsers.

HTML VIDEO

The HTML <video> element is used to show a video on a web page.

Example
Courtesy of Big Buck Bunny:

The HTML <video> Element

To show a video in HTML, use the <video> element:

Example:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

How it Works

- ✓ The controls attribute adds video controls, like play, pause, and volume.
- ✓ It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.
- ✓ The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
- ✓ The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

HTML <video> Autoplay

To start a video automatically, use the autoplay attribute:

Example:

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
```

```
Your browser does not support the video tag.
</video>
```

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

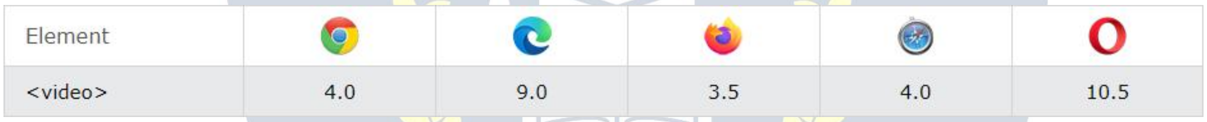
Add muted after autoplay to let your video start playing automatically (but muted):

Example:

```
<video width="320" height="240" autoplay muted>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

Browser Support

The numbers in the table specify the first browser version that fully supports the <video> element.



Element					
<video>	4.0	9.0	3.5	4.0	10.5

HTML Video Formats

There are three supported video formats: MP4, WebM, and Ogg. The browser support for the different formats is:

Browser	**MP4**	**WebM**	**Ogg**
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

HTML Video - Media Types

File Format	**Media Type**
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

HTML Video - Methods, Properties, and Events

- ✓ The HTML DOM defines methods, properties, and events for the <video> element.
- ✓ This allows you to load, play, and pause videos, as well as setting duration and volume.
- ✓ There are also DOM events that can notify you when a video begins to play, is paused, etc.

Example: Using JavaScript

Video courtesy of Big Buck Bunny.

For a full DOM reference, go to our HTML Audio/Video DOM Reference.

HTML AUDIO

The HTML <audio> element is used to play an audio file on a web page.

The HTML <audio> Element

To play an audio file in HTML, use the <audio> element:

Example:

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Audio - How It Works

- ✓ The controls attribute adds audio controls, like play, pause, and volume.
- ✓ The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.
- ✓ The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

HTML <audio> Autoplay

To start an audio file automatically, use the autoplay attribute:

Example:

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Note: Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.

Add muted after autoplay to let your audio file start playing automatically (but muted):

Example:

```
<audio controls autoplay muted>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

Browser Support

The numbers in the table specify the first browser version that fully supports the <audio> element.

Element					
<audio>	4.0	9.0	3.5	4.0	10.5

HTML Audio Formats

There are three supported audio formats: MP3, WAV, and OGG. The browser support for the different formats is:

Browser	**MP3**	**WAV**	**OGG**
Edge/IE	YES	YES*	YES*
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

*From Edge 79

HTML Audio - Media Types

File Format	**Media Type**
MP3	audio/mpeg
OGG	audio/ogg
WAV	audio/wav

HTML Audio - Methods, Properties, and Events

- ✓ The HTML DOM defines methods, properties, and events for the <audio> element.
- ✓ This allows you to load, play, and pause audios, as well as set duration and volume.
- ✓ There are also DOM events that can notify you when an audio begins to play, is paused, etc.
- ✓ For a full DOM reference, go to our HTML Audio/Video DOM Reference.

HTML Audio Tags

Tag	**Description**
<audio>	Defines sound content
<source>	Defines multiple media resources for media elements, such as <video> and <audio>

HTML PLUG-INS

Plug-ins are computer programs that extend the standard functionality of the browser.

Plug-ins

Plug-ins were designed to be used for many different purposes:

- To run Java applets
- To run Microsoft ActiveX controls
- To display Flash movies

- To display maps
- To scan for viruses
- To verify a bank id

Warning!

- ✓ Most browsers no longer support Java Applets and Plug-ins.
- ✓ ActiveX controls are no longer supported in any browsers.
- ✓ The support for Shockwave Flash has also been turned off in modern browsers.

The <object> Element

- ✓ The <object> element is supported by all browsers.
- ✓ The <object> element defines an embedded object within an HTML document.
- ✓ It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:

Example:

```
<object width="100%" height="500px" data="snippet.html"></object>
```

Or images if you like:

Example:

```
<object data="audi.jpeg"></object>
```

The <embed> Element

- ✓ The <embed> element is supported in all major browsers.
- ✓ The <embed> element also defines an embedded object within an HTML document.
- ✓ Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

Example:

```
<embed src="audi.jpeg">
```

Note that the <embed> element does not have a closing tag. It can not contain alternative text.

The <embed> element can also be used to include HTML in HTML:

Example:

```
<embed width="100%" height="500px" src="snippet.html">
```

HTML YOUTUBE VIDEOS

The easiest way to play videos in HTML, is to use YouTube.

Struggling with Video Formats?

- ✓ Converting videos to different formats can be difficult and time-consuming.
- ✓ An easier solution is to let YouTube play the videos in your web page.

YouTube Video Id

- ✓ YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.
- ✓ You can use this id, and refer to your video in the HTML code.

Playing a YouTube Video in HTML

To play your video on a web page, do the following:

- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player
- Add any other parameters to the URL (see below)

Example:

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

YouTube Autoplay + Mute

- ✓ You can let your video start playing automatically when a user visits the page, by adding autoplay=1 to the YouTube URL. However, automatically starting a video is annoying for your visitors!
- ✓ **Note:** Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.
- ✓ Add mute=1 after autoplay=1 to let your video start playing automatically (but muted).

<u>YouTube - Autoplay + Muted</u>

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
```

YouTube Playlist

A comma separated list of videos to play (in addition to the original URL).

YouTube Loop

- ✓ Add loop=1 to let your video loop forever.
- ✓ Value 0 (default): The video will play only once.
- ✓ Value 1: The video will loop (forever).

<u>YouTube - Loop</u>

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY
&loop=1">
</iframe>
```

YouTube Controls

- ✓ Add controls=0 to not display controls in the video player.
- ✓ Value 0: Player controls does not display.
- ✓ Value 1 (default): Player controls display.

<u>YouTube - Controls</u>

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

HTML GEOLOCATION API

The HTML Geolocation API is used to locate a user's position.

Locate the User's Position

- ✓ The HTML Geolocation API is used to get the geographical position of a user.
- ✓ Since this can compromise privacy, the position is not available unless the user approves it.

Note: Geolocation is most accurate for devices with GPS, like smartphones.

Browser Support

The numbers in the table specify the first browser version that fully supports Geolocation.

API					
Geolocation	5.0 - 49.0 (http) 50.0 (https)	9.0	3.5	5.0	16.0

Note: As of Chrome 50, the Geolocation API will only work on secure contexts such as HTTPS. If your site is hosted on an non-secure origin (such as HTTP) the requests to get the users location will no longer function.

Using HTML Geolocation

- ✓ The getCurrentPosition() method is used to return the user's position.
- ✓ The example below returns the latitude and longitude of the user's position:

Example:

```
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation)
    { navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

Example explained:

- Check if Geolocation is supported
- If supported, run the getCurrentPosition() method. If not, display a message to the user
- If the getCurrentPosition() method is successful, it returns a coordinates object to the function specified in the parameter (showPosition)
- The showPosition() function outputs the Latitude and Longitude

The example above is a very basic Geolocation script, with no error handling.

Handling Errors and Rejections

The second parameter of the getCurrentPosition() method is used to handle errors. It specifies a function to run if it fails to get the user's location:

Example:

```
function showError(error)
 {switch(error.code) {
   case error.PERMISSION_DENIED:
     x.innerHTML = "User denied the request for Geolocation."
     break;
   case error.POSITION_UNAVAILABLE:
     x.innerHTML = "Location information is unavailable."
     break;
   case error.TIMEOUT:
     x.innerHTML = "The request to get user location timed out."
     break;
   case error.UNKNOWN_ERROR:
     x.innerHTML = "An unknown error occurred."
     break;
 }
}
```

Displaying the Result in a Map

- ✓ To display the result in a map, you need access to a map service, like Google Maps.
- ✓ In the example below, the returned latitude and longitude is used to show the location in a Google Map (using a static image):

Example:

```
function showPosition(position) {
 var latlon = position.coords.latitude + "," + position.coords.longitude;

 var img_url = "https://maps.googleapis.com/maps/api/staticmap?center=
 "+latlon+"&zoom=14&size=400x300&sensor=false&key=YOUR_KEY";

 document.getElementById("mapholder").innerHTML = "<img
src='"+img_url+"'>";
```

}

Location-specific Information

- ✓ This page has demonstrated how to show a user's position on a map.
- ✓ Geolocation is also very useful for location-specific information, like:
  - Up-to-date local information
  - Showing Points-of-interest near the user
  - Turn-by-turn navigation (GPS)

The get Current Position () Method - Return Data

The get Current Position () method returns an object on success. The latitude, longitude and accuracy properties are always returned. The other properties are returned if available:

Property	**Returns**
coords.latitude	The latitude as a decimal number (always returned)
coords.longitude	The longitude as a decimal number (always returned)
coords.accuracy	The accuracy of position (always returned)
coords.altitude	The altitude in meters above the mean sea level (returned if available)
coords.altitudeAccuracy	The altitude accuracy of position (returned if available)
coords.heading	The heading as degrees clockwise from North (returned if available)
coords.speed	The speed in meters per second (returned if available)
timestamp	The date/time of the response (returned if available)

Geolocation Object - Other interesting Methods

The Geolocation object also has other interesting methods:

- Watch Position () - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
- Clear Watch () - Stops the watch Position () method.

The example below shows the watch Position() method. You need an accurate GPS device to test this (like smartphone):

Example:

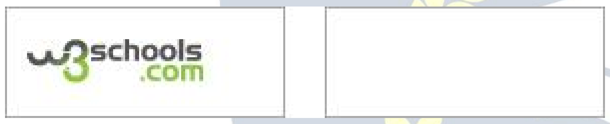
```
<script>
var x = document.getElementById("demo");
function getLocation() {
 if (navigator.geolocation)
   { navigator.geolocation.watchPosition(showPosition);
```

```
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
```

HTML DRAG AND DROP API

In HTML, any element can be dragged and dropped.

Example



Drag the W3Schools image into the rectangle.

Drag and Drop

Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.

Browser Support

The numbers in the table specify the first browser version that fully supports Drag and Drop.

API					
Drag and Drop	4.0	9.0	3.5	6.0	12.0

HTML Drag and Drop Example

The example below is a simple drag and drop example:

Example:

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}

function drag(ev)
  { ev.dataTransfer.setData("text", ev.target.id);
}

function drop(ev)
  { ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>

<img id="drag1" src="img_logo.gif" draggable="true" ondragstart="drag(event)
" width="336" height="69">

</body>
</html>
```

It might seem complicated, but lets go through all the different parts of a drag and drop event.

Make an Element Draggable

First of all: To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

What to Drag - ondragstart and setData()

- ✓ Then, specify what should happen when the element is dragged.
- ✓ In the example above, the ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
- ✓ The dataTransfer.setData() method sets the data type and the value of the dragged data:

```
function drag(ev)
  { ev.dataTransfer.setData("text", ev.target.id);
}
```

In this case, the data type is "text" and the value is the id of the draggable element ("drag1").

Where to Drop - ondragover

- ✓ The ondragover event specifies where the dragged data can be dropped.
- ✓ By default, data/elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element.
- ✓ This is done by calling the event.preventDefault() method for the ondragover event:

```
event.preventDefault()
```

Do the Drop - ondrop

When the dragged data is dropped, a drop event occurs.
In the example above, the ondrop attribute calls a function, drop(event):

```
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

Code explained:

- Call preventDefault() to prevent the browser default handling of the data (default is open as link on drop)
- Get the dragged data with the dataTransfer.getData() method. This method will return any data that was set to the same type in the setData() method
- The dragged data is the id of the dragged element ("drag1")
- Append the dragged element into the drop element

More Examples

Example
How to drag (and drop) an image back and forth between two <div> elements:

HTML WEB STORAGE API

HTML web storage; better than cookies.

What is HTML Web Storage?

- ✓ With web storage, web applications can store data locally within the user's browser.
- ✓ Before HTML5, application data had to be stored in cookies, included in every server request. Web storage is more secure, and large amounts of data can be stored locally, without affecting website performance.
- ✓ Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.
- ✓ Web storage is per origin (per domain and protocol). All pages, from one origin, can store and access the same data.

Browser Support

The numbers in the table specify the first browser version that fully supports Web Storage.

API					
Web Storage	4.0	8.0	3.5	4.0	11.5

HTML Web Storage Objects

HTML web storage provides two objects for storing data on the client:

- window.localStorage - stores data with no expiration date
- window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)

Before using web storage, check browser support for localStorage and sessionStorage:

```
if (typeof(Storage) !== "undefined") {
  // Code for localStorage/sessionStorage.
} else {
  // Sorry! No Web Storage support..
}
```

The local Storage Object

The local Storage object stores the data with no expiration date. The data will not be deleted when the browser is closed, and will be available the next day, week, or year.

Example:

```
// Store
localStorage.setItem("lastname", "Smith");

// Retrieve
document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
```

Example explained:

- Create a localStorage name/value pair with name="lastname" and value="Smith"
- Retrieve the value of "lastname" and insert it into the element with id="result"

The example above could also be written like this:

```
// Store
localStorage.lastname = "Smith";
// Retrieve
document.getElementById("result").innerHTML = localStorage.lastname;
```

The syntax for removing the "lastname" localStorage item is as follows:

```
localStorage.removeItem("lastname");
```

Note: Name/value pairs are always stored as strings. Remember to convert them to another format when needed!

The following example counts the number of times a user has clicked a button. In this code the value string is converted to a number to be able to increase the counter:

Example:

```
if (localStorage.clickcount) {
  localStorage.clickcount = Number(localStorage.clickcount) + 1;
} else
  { localStorage.clickcount =
  1;
}
document.getElementById("result").innerHTML = "You have clicked the button
" +
localStorage.clickcount + " time(s).";
```

The sessionStorage Object

- ✓ The sessionStorage object is equal to the localStorage object, **except** that it stores the data for only one session. The data is deleted when the user closes the specific browser tab.
- ✓ The following example counts the number of times a user has clicked a button, in the current session:

Example:

```
if (sessionStorage.clickcount) {
  sessionStorage.clickcount = Number(sessionStorage.clickcount) + 1;
} else
  { sessionStorage.clickcount =
  1;
}
document.getElementById("result").innerHTML = "You have clicked the button
" +
sessionStorage.clickcount + " time(s) in this session.";
```

HTML WEB WORKERS API

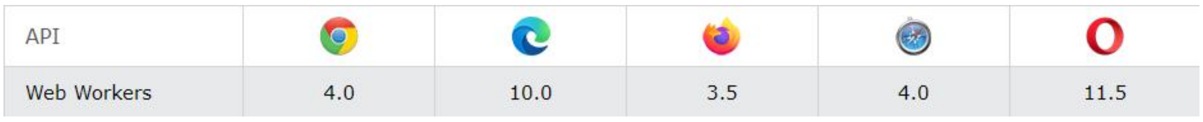
A web worker is a JavaScript running in the background, without affecting the performance of the page.

What is a Web Worker?

- ✓ When executing scripts in an HTML page, the page becomes unresponsive until the script is finished.
- ✓ A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

Browser Support

The numbers in the table specify the first browser version that fully support Web Workers.



API					
Web Workers	4.0	10.0	3.5	4.0	11.5

HTML Web Workers Example

The example below creates a simple web worker that count numbers in the background:

Example:
Count numbers:

Start Worker   Stop Worker

Check Web Worker Support

Before creating a web worker, check whether the user's browser supports it:

```
if (typeof(Worker) !== "undefined") {
  // Yes! Web worker support!
  // Some code.....
} else {
  // Sorry! No Web Worker support..
}
```

Create a Web Worker File

- ✓ Now, let's create our web worker in an external JavaScript.
- ✓ Here, we create a script that counts. The script is stored in the "demo_workers.js" file:

```
var i = 0;

function timedCount()
  {i = i + 1;
  postMessage(i);
  setTimeout("timedCount()",500);
}

timedCount();
```

The important part of the code above is the postMessage() method - which is used to post a message back to the HTML page.

Note: Normally web workers are not used for such simple scripts, but for more CPU intensive tasks.

Create a Web Worker Object

Now that we have the web worker file, we need to call it from an HTML page. The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo_workers.js":

```
if (typeof(w) == "undefined") {
  w = new Worker("demo_workers.js");
}
```

Then we can send and receive messages from the web worker.

Add an "onmessage" event listener to the web worker.

```
w.onmessage =
  function(event){ document.getElementById("result").innerHTML = event.data;
};
```

When the web worker posts a message, the code within the event listener is executed. The data from the web worker is stored in event.data.

Terminate a Web Worker

- ✓ When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.
- ✓ To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

Reuse the Web Worker

If you set the worker variable to undefined, after it has been terminated, you can reuse the code:

```
w = undefined;
```

Full Web Worker Example Code

We have already seen the Worker code in the .js file. Below is the code for the HTML page:

Example:

```
<!DOCTYPE html>
<html>
<body>

<p>Count numbers: <output id="result"></output></p>
<button onclick="startWorker()">Start Worker</button>
<button onclick="stopWorker()">Stop Worker</button>

<script>
```

```
var w;

function startWorker() {
 if (typeof(Worker) !== "undefined")
  {if (typeof(w) == "undefined") {
   w = new Worker("demo_workers.js");
  }
  w.onmessage = function(event)
   { document.getElementById("result").innerHTML = event.data;
  };
 } else {
  document.getElementById("result").innerHTML = "Sorry! No Web Worker
support.";
 }
}

function stopWorker()
 {w.terminate();
 w = undefined;
}
</script>

</body>
</html>
```

Web Workers and the DOM

Since web workers are in external files, they do not have access to the following JavaScript objects:

- The window object
- The document object
- The parent object

HTML SSE API

Server-Sent Events (SSE) allow a web page to get updates from a server.

Server-Sent Events - One Way Messaging

- ✓ A server-sent event is when a web page automatically gets updates from a server.
- ✓ This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.
- ✓ Examples: Facebook/Twitter updates, stock price updates, news feeds, sport results, etc.

Browser Support

The numbers in the table specify the first browser version that fully support server-sent events.

API					
SSE	6.0	79.0	6.0	5.0	11.5

Receive Server-Sent Event Notifications

The EventSource object is used to receive server-sent event notifications:

Example:

```
var source = new EventSource("demo_sse.php");
source.onmessage = function(event) {
  document.getElementById("result").innerHTML += event.data + "<br>";
};
```

Example explained:

- Create a new EventSource object, and specify the URL of the page sending the updates (in this example "demo_sse.php")
- Each time an update is received, the onmessage event occurs
- When an onmessage event occurs, put the received data into the element with id="result"

Check Server-Sent Events Support

In the tryit example above there were some extra lines of code to check browser support for server-sent events:

```
if(typeof(EventSource) !== "undefined") {
  // Yes! Server-sent events support!
  // Some code.....
} else {
  // Sorry! No server-sent events support..
}
```

Server-Side Code Example

- ✓ For the example above to work, you need a server capable of sending data updates (like PHP or ASP).
- ✓ The server-side event stream syntax is simple. Set the "Content-Type" header to "text/event-stream". Now you can start sending event streams.
- ✓ Code in PHP (demo_sse.php):

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

Code in ASP (VB) (demo_sse.asp):

```
<%
Response.ContentType = "text/event-stream"
Response.Expires = -1
Response.Write("data: The server time is: " & now())
Response.Flush()
%>
```

Code explained:

- Set the "Content-Type" header to "text/event-stream"
- Specify that the page should not cache
- Output the data to send (**Always** start with "data: ")
- Flush the output data back to the web page

The Event Source Object

In the examples above we used the onmessage event to get messages. But other events are also available:

Events	**Description**
onopen	When a connection to the server is opened
onmessage	When a message is received
onerror	When an error occurs

VIDYAPITH ACADEMY

A unit of **AITDC (OPC) PVT. LTD**.

IAF Accredited An ISO 9001:2015 Certified Institute.

Registered Under Ministry of Corporate Affairs

(CIN U80904AS2020OPC020468)

Registered Under MSME, Govt. of India. (UAN- AS04D0000207).

Registered Under MHRD (CR act) Govt. of India.