PANJAB UNIVERSITY, CHANDIGARH-160014 (INDIA)

(Estd. under the Panjab University Act VII of 1947 – enacted by the Govt. of India)

Faculty of Science

Syllabi

For

BACHELOR OF COMPUTER APPLICATIONS (B.C.A)

(SEMESTER SYSTEM – 2022-23)

Second Semester – BCA-16-203

Fundamentals of Web Programming

By

Dr. Karuna Babber

Assistant Professor (IT)

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.

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

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

  </body>
  </html>
  ```

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>
 	none	*none*

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:
  ```

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	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1
2017	W3C Recommendation: HTML5.1 2nd Edition
2017	W3C Recommendation: HTML5.2

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**.

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

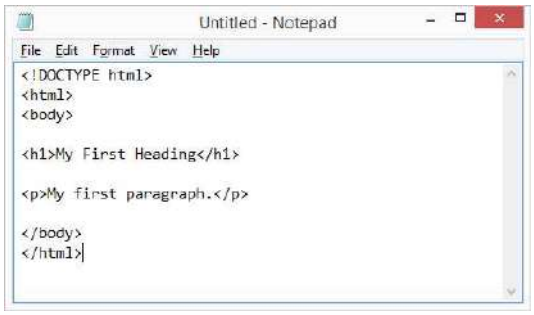
Write or copy the following HTML code into Notepad:

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

<h1>My First Heading</h1>

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

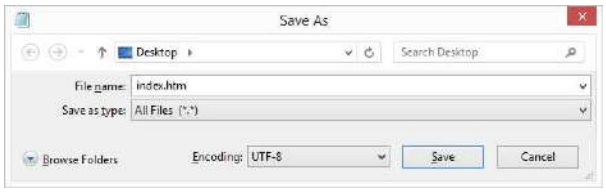
```
</body>
</html>
```



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).



You can either use .htm or .html as file extension. There is no difference.

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:

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>
```

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.tryschools.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="tryschools.jpg" alt="TRYSchools.com" width="104" height="142">
```

How to View HTML Source
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.

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>

My first paragraph.

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

Note: Empty elements have no end tags

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):

Example:

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

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

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

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 :

```
<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 element defines a paragraph.

It has a start tag and an end tag :

```
<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
 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 Case Sensitive

HTML tags are not case sensitive: <P> means the same as <p>.

The HTML standard does not require lowercase tags, but W3C **recommends** lowercase in HTML, and **demands** lowercase for stricter document types like XHTML.

HTML Tag Reference

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

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:

visit our school

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.tryschools.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 specify 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 a 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">

<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>
```

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**.

The HTML standard does not require quotes around attribute values.

Good:

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

Bad:

Visit our HTML tutorial

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

Example:

<p title=About My school>

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">

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>
```

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.

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>
```

HTML Line Breaks

The HTML
 element defines a line break.

Use
 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 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>
```

Tag	Description
<p>	Defines a paragraph

<hr>	Defines a thematic change in the content
 	Inserts a single line break
<pre>	Defines pre-formatted text

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;*">

Background Color

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

Example:

```
<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>
```

HTML Formatting Elements

Formatting elements were designed to display special types of text:

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

HTML <b> and Elements

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

Example:

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

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

Example:

This text is important!

HTML and Elements

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

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

Example:

This text is italic

The HTML element defines emphasized text. The content inside is typically displayed in italic.

Tip: A screen reader will pronounce the words in with an emphasis, using verbal stress.

Example:

This text is emphasized

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:

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

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:

My favorite color is ~~blue~~ red.

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>
```

HTML Text Formatting Elements

Tag	Description
<b>	Defines bold text
	Defines emphasized text
<i>	Defines a part of text in an alternate voice or mood
<small>	Defines smaller text
	Defines important text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

<mark>	Defines marked/highlighted text

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 60 years, WWF has worked to help people and nature thrive. As the world's leading conservation organization, WWF works in nearly 100 countries. At every level, we collaborate with people around the world to develop and deliver innovative solutions that protect communities, wildlife, and the places in which they live.
</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>
```

HTML Quotation and Citation Elements

Tag	Description
<abbr>	Defines an abbreviation or acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction
<blockquote>	Defines a section that is quoted from another source
<cite>	Defines the title of a work

<q>	Defines a short inline quotation

HTML Comment Tag

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.

Add Comments

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 -->
```

Hide Content

Comments can be used to hide content.

This can be helpful if you hide content temporarily:

Example

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

<!-- <p>This is another paragraph </p> -->

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

Example

Hide a section of HTML code:

```
<p>This is a paragraph.</p>
<!--
<p>Look at this cool image:</p>
<img border="0" src="pic_trulli.jpg" alt="Trulli">
-->
<p>This is a paragraph too.</p>
```

Hide Inline Content

Comments can be used to hide parts in the middle of the HTML code.

Example:

```
<p>This <!-- great text --> is a paragraph.</p>
```

Background Color

You can set the background color for HTML elements:

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:

Example

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

Example:

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">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 elements have their background color set with RGB, HEX, and HSL values:

```
<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 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).

What is CSS?

Cascading Style Sheets (CSS) is used to format the layout of a webpage.

With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colors are to be used, different displays for different devices and screen sizes, and much more!

Using CSS

CSS can be added to HTML documents in 3 ways:

- **Inline** - by using the style attribute inside HTML elements
- **Internal** - by using a <style> element in the <head> section
- **External** - by using a <link> element to link to an external CSS file

The most common way to add CSS, is to keep the styles in external CSS files. However, in this tutorial we will use inline and internal styles, because this is easier to demonstrate, and easier for you to try it yourself.

Inline CSS

An inline CSS is used to apply a unique style to a single HTML element.

An inline CSS uses the style attribute of an HTML element.

The following example sets the text color of the <h1> element to blue, and the text color of the <p> element to red:

Example:

```
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

Internal CSS

An internal CSS is used to define a style for a single HTML page.

An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

The following example sets the text color of ALL the <h1> elements (on that page) to blue, and the text color of ALL the <p> elements to red. In addition, the page will be displayed with a "powderblue" background color:

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
```

```
p    {color: red;}
</style>
</head>
<body>

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

</body>
</html>
```

External CSS

An external style sheet is used to define the style for many HTML pages.

To use an external style sheet, add a link to it in the <head> section of each HTML page:

Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

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

</body>
</html>
```

The external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is what the "styles.css" file looks like:

“styles.css”:

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

CSS Colors, Fonts and Sizes

Here, we will demonstrate some commonly used CSS properties. You will learn more about them later.

The CSS color property defines the text color to be used.

The CSS font-family property defines the font to be used.

The CSS font-size property defines the text size to be used.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
</head>
<body>

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

</body>
</html>
```

CSS Border

The CSS border property defines a border around an HTML element.

Tip: You can define a border for nearly all HTML elements.

Example:

```
p {
  border: 2px solid powderblue;
}
```

CSS Padding

The CSS padding property defines a padding (space) between the text and the border.

Example:

```
p {
  border: 2px solid powderblue;
  padding: 30px;
}
```

CSS Margin

The CSS margin property defines a margin (space) outside the border.

Example:

```
p {
  border: 2px solid powderblue;
  margin: 50px;
}
```

Link to External CSS

External style sheets can be referenced with a full URL or with a path relative to the current web page.

Example:

```
<link rel="stylesheet" href="https://www.tryschools.com/html/styles.css">
```

Example

This example links to a style sheet located in the html folder on the current web site:

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

Example

This example links to a style sheet located in the same folder as the current page:

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

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 a 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:

link text

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:

Visit my school.com!

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

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:

Visit my schools!

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>
TRYC

```
<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.tryschools.com/html/" title="Go to TRYSchools HTML section">Visit our HTML Tutorial</a>
```

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 mouse is 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:

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>
```

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>
```

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>
```

HTML Images

Example:

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

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

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">
```

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;">
```

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="HTML5 Icon" 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.try.com/images/my_green.jpg" alt="try.com">
```

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="Smiley face" 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

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

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.

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:

Example:

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

Shape="circle"

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

Example:

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

Shape="poly"

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.

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">
```

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:

```
<map name="workmap">
  <area shape="circle" coords="337,300,44" href="coffee.htm" onclick="myFunction()">
</map>

<script>
function myFunction() {
  alert("You clicked the coffee cup!");
}
</script>
```

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:

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

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

Example:

```
<style>
p {
  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:

```
<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%:

Example

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

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:

```
<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>
```

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

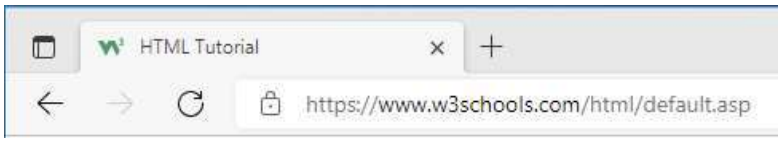
Some browsers or devices may not support all image formats. By using the <picture> element, you can add images of all formats, and the browser will use the first format it recognizes, and ignore any of the following elements.

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

How To Add a Favicon in HTML

You can use any image you like as your favicon. You can also create your own favicon on sites like https://www.favicon.cc.

A favicon image is displayed to the left of the page title in the browser tab, like this:



To add a favicon to your website, either save your favicon image to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".

Next, add a <link> element to your "index.html" file, after the <title> element, like this:

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page Title</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>

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

</body>
</html>
```

Favicon File Format Support

Define an HTML Table

A table in HTML consists of table cells inside rows and columns.

Example:

```
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
```

```
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
```

Table Cells

Each table cell is defined by a <td> and a </td> tag.

td stands for table data.

Everything between <td> and </td> are the content of the table cell.

Example:

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
</table>
```

Table Rows

Each table row starts with a <tr> and ends with a </tr> tag. tr stands for table row.

Example:

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
```

```
  </tr>
</table>
```

Table Headers

Sometimes you want your cells to be table header cells. In those cases use the <th> tag instead of the <td> tag: th stands for table header.

Example:

```
<table>
  <tr>
    <th>Person 1</th>
    <th>Person 2</th>
    <th>Person 3</th>
  </tr>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
  <tr>
    <td>16</td>
    <td>14</td>
    <td>10</td>
  </tr>
</table>
```

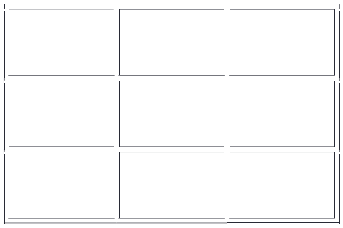
HTML Table Tags

Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting

<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

How To Add a Border

When you add a border to a table, you also add borders around each table cell:



To add a border, use the CSS border property on table, th, and td elements:

Example:

```
table, th, td {
  border: 1px solid black;
}
```

Collapsed Table Borders

To avoid having double borders like in the example above, set the CSS border-collapse property to collapse.

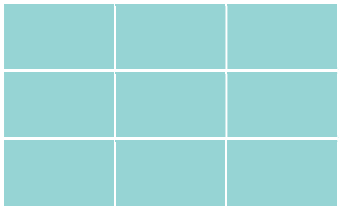
This will make the borders collapse into a single border:

Example:

```
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
```

Style Table Borders

If you set a background color of each cell, and give the border a white color (the same as the document background), you get the impression of an invisible border:

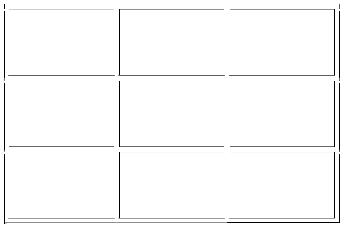


Example:

```
table, th, td {
  border: 1px solid white;
  border-collapse: collapse;
}
th, td {
  background-color: #96D4D4;
}
```

Round Table Borders

With the border-radius property, the borders get rounded corners:



Example:

```
table, th, td {
  border: 1px solid black;
  border-radius: 10px;
}
```

Dotted Table Borders

With the border-style property, you can set the appearance of the border.

The following values are allowed:

Example

```
th, td {
  border-style: dotted;
}
```

Border Color

With the border-color property, you can set the color of the border.

Example:

```
th, td {
  border-color: #96D4D4;
}
```

HTML Table Width

To set the width of a table, add the style attribute to the <table> element:

Example:

```
<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>
```

To set the size of a specific column, add the style attribute on a <th> or <td> element:

Example:

```
<table style="width:100%">
  <tr>
    <th style="width:70%">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>
```

HTML Table Row Height

To set the height of a specific row, add the style attribute on a table row element:

Example:

```
<table style="width:100%">
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Age</th>
  </tr>
  <tr style="height:200px">
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```

HTML Table Headers

Table headers are defined with th elements. Each th element represents a table cell.

Example:

```
<table>
  <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>
```

Vertical Table Headers

To use the first column as table headers, define the first cell in each row as a <th> element:

Example:

```
<table>
  <tr>
    <th>Firstname</th>
    <td>Jill</td>
    <td>Eve</td>
  </tr>
  <tr>
    <th>Lastname</th>
    <td>Smith</td>
    <td>Jackson</td>
  </tr>
  <tr>
    <th>Age</th>
    <td>94</td>
    <td>50</td>
  </tr>
</table>
```

Align Table Headers

By default, table headers are bold and centered:

Firstname	Lastname	Age
Jill	Smith	50
Eve	Jackson	94

To left-align the table headers, use the CSS text-align property:

Example:

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

Header for Multiple Columns

You can have a header that spans over two or more columns.

Name		Age
Jill	Smith	50
Eve	Jackson	94

To do this, use the colspan attribute on the <th> element:

Example:

```
<table>
  <tr>
    <th colspan="2">Name</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>
```

Table Caption

You can add a caption that serves as a heading for the entire table.

Monthly savings

Month	Savings
January	$100

February	$50

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>
```

HTML Table Padding & Spacing

HTML tables can adjust the padding inside the cells, and also the space between the cells.

With Padding		
hello	hello	hello
hello	hello	hello
hello	hello	hello

With Spacing		
hello	hello	hello
hello	hello	hello
hello	hello	hello

HTML Table - Cell Padding

Cell padding is the space between the cell edges and the cell content.

By default the padding is set to 0.

To add padding on table cells, use the CSS padding property:

Example:

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

To add padding only above the content, use the padding-top property.

And the others sides with the padding-bottom, padding-left, and padding-right properties:

Example:

```
th, td {
  padding-top: 10px;
  padding-bottom: 20px;
  padding-left: 30px;
  padding-right: 40px;
}
```

HTML Table - Cell Spacing

Cell spacing is the space between each cell.

By default the space is set to 2 pixels.

To change the space between table cells, use the CSS border-spacing property on the table element:

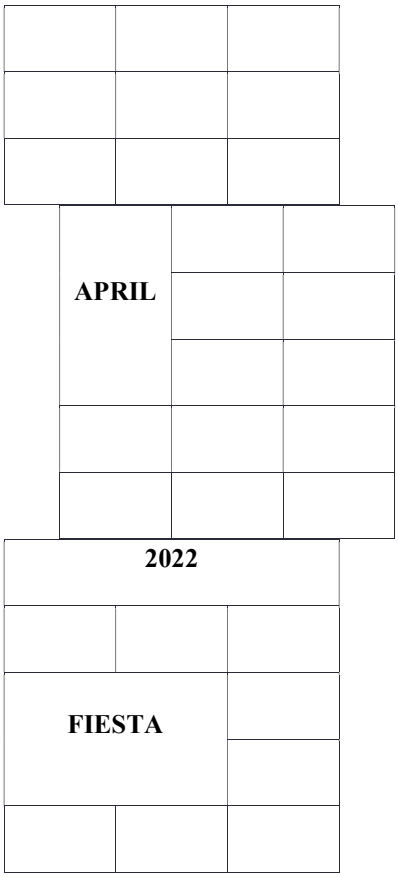
Example:

```
table {
  border-spacing: 30px;
}
```

HTML Table Colspan & Rowspan

HTML tables can have cells that span over multiple rows and/or columns.

NAME	



HTML Table - Colspan

To make a cell span over multiple columns, use the colspan attribute:

Example:

```
<table>
  <tr>
    <th colspan="2">Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>43</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>57</td>
  </tr>
</table>
```

HTML Table - Rowspan

To make a cell span over multiple rows, use the rowspan attribute:

Example:

```
<table>
  <tr>
    <th>Name</th>
    <td>Jill</td>
  </tr>
  <tr>
    <th rowspan="2">Phone</th>
    <td>555-1234</td>
  </tr>
  <tr>
    <td>555-8745</td>
</tr>
</table>
```

HTML Table - Zebra Stripes

If you add a background color on every other table row, you will get a nice zebra stripes effect.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

To style every other table row element, use the :nth-child(even) selector like this:

Example:

```
tr:nth-child(even) {
  background-color: #D6EEEE;
}
```

HTML Table - Vertical Zebra Stripes

To make vertical zebra stripes, style every other *column*, instead of every other *row*.

1	2	3	4

5	6	7	8
9	10	11	12
13	14	15	16
17	18	19	20

Set the :nth-child(even) for table data elements like this:

Example:

```
td:nth-child(even), th:nth-child(even) {
  background-color: #D6EEEE;
}
```

Combine Vertical and Horizontal Zebra Stripes

You can combine the styling from the two examples above and you will have stripes on every other row and every other column.

If you use a transparent color you will get an overlapping effect.

Use an rgba() color to specify the transparency of the color:

Example:

```
tr:nth-child(even) {
  background-color: rgba(150, 212, 212, 0.4);
}

th:nth-child(even),td:nth-child(even) {
  background-color: rgba(150, 212, 212, 0.4);
}
```

Horizontal Dividers

First Name	Last Name	Savings
Peter	Griffin	$100

If you specify borders only at the bottom of each table row, you will have a table with horizontal dividers.

Add the border-bottom property to all tr elements to get horizontal dividers:

```
tr {
  border-bottom: 1px solid #ddd;
}
```

Hoverable Table

Use the :hover selector on tr to highlight table rows on mouse over:

First Name	Last Name	Savings
Peter	Griffin	$100

```
tr:hover {background-color: #D6EEEE;}
```

HTML Table Colgroup

If you want to style the two first columns of a table, use the <colgroup> and <col> elements.

MON	TUE	WED	THU	FRI	SAT	SUN
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

The <colgroup> element should be used as a container for the column specifications.

Each group is specified with a <col> element.

The span attribute specifies how many columns that get the style.

The style attribute specifies the style to give the columns.

Example

```
<table>
  <colgroup>
    <col span="2" style="background-color: #D6EEEE">
  </colgroup>
  <tr>
    <th>MON</th>
    <th>TUE</th>
    <th>WED</th>
    <th>THU</th>
...
```

Multiple Col Elements

If you want to style more columns with different styles, use more <col> elements inside the <colgroup>:

```
<table>
  <colgroup>
    <col span="2" style="background-color: #D6EEEE">
    <col span="3" style="background-color: pink">
  </colgroup>
  <tr>
    <th>MON</th>
    <th>TUE</th>
    <th>WED</th>
    <th>THU</th>
...
```

Empty Colgroups

If you want to style columns in the middle of a table, insert a "empty" <col> element (with no styles) for the columns before:

Example:

```
<table>
  <colgroup>
    <col span="3">
    <col span="2" style="background-color: pink">
  </colgroup>
  <tr>
    <th>MON</th>
    <th>TUE</th>
    <th>WED</th>
    <th>THU</th>
...
```

Hide Columns

You can hide columns with the visibility: collapse property:

Example:

```
<table>
  <colgroup>
    <col span="2">
    <col span="3" style="visibility: collapse">
  </colgroup>
  <tr>
    <th>MON</th>
    <th>TUE</th>
    <th>WED</th>
    <th>THU</th>
...
```

HTML Lists

HTML lists allow web developers to group a set of related items in lists.

Example : an unordered HTML list:

- 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:

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

HTML List Tags

Tag	Description
	Defines an unordered list
	Defines an ordered list
	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

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:

```
<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):

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

```
    <li>Green tea</li>
   </ul>
  </li>
  <li>Milk</li>
</ul>
```

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:

```
<!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>
```

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:

```
<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

```
<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:

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

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:

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

```
  <dd>- white cold drink</dd>
</dl>
```

Block-level Elements

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Two commonly used block elements are: <p> and .

The <p> element defines a paragraph in an HTML document.

The element defines a division or a section in an HTML document.

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

The element is a block-level element.

Example:

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

Here are the block-level elements in HTML:

<address><article><aside><blockquote><canvas>

<dd> <dl><dt><fieldset><figcaption>

<figure><footer><form><h1>-<h6><header>

<hr>
<main><nav><noscript>

<p><pre><section><table><tfoot>
<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 element inside a paragraph.

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

Here are the inline elements in HTML:

<a><abbr><acronym><b><bdo><big>
<button>

<cite><code><dfn><i><img><input><kbd>
<label><map><object><output><q><samp>
<script><select><small>
<sub><sup><textarea><time><tt><var>

The Element

The element is often used as a container for other HTML elements.

The element has no required attributes, but style, class and id are common.

When used together with CSS, the element can be used to style blocks of content:

```
<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 Element

The element is an inline container used to mark up a part of a text, or a part of a document.

The element has no required attributes, but style, class and id are common.

When used together with CSS, the element can be used to style parts of the text:

```
<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>
```

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 elements with a class attribute with the value of "city". All of the three elements will be styled equally according to the .city style definition in the head section:

```
<!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 elements with a class attribute with the value of "note". Both elements will be styled equally according to the .note style definition in the head section:

```
<!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>
```

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. . 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:

```
<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> point to the "city" class and will share the same style:

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

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:

```
<!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>
```

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:

```
<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:

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

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.

```
<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:

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

```
<iframe src="demo_iframe.htm" style="height:200px;width:300px;" title="Iframe 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:

```
<iframe src="demo_iframe.htm" style="border:none;" 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:

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

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

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

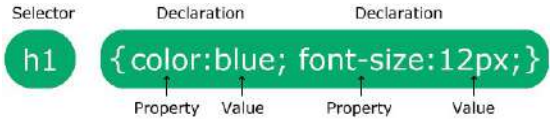
CSS Example

```
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

CSS Syntax



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

```
p {
  color: red;
  text-align: center;
}
```

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

The CSS element Selector

The element selector selects HTML elements based on the element name.

```
p {
  text-align: center;
  color: red;
}
```

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example

The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

Example

In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {
  text-align: center;
  color: red;
}
```

Example

In this example only <p> elements with class="center" will be red and center-aligned:

```
p.center {
  text-align: center;
  color: red;
}
```

HTML elements can also refer to more than one class.

Example

In this example the <p> element will be styled according to class="center" and to class="large":

```
<p class="center large">This paragraph refers to two classes.</p>
```

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

Example

The CSS rule below will affect every HTML element on the page:

```
* {
  text-align: center;
  color: blue;
}
```

The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

All CSS Simple Selectors

Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,..	div, p	Selects all elements and all <p> elements

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

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

</body>
</html>
```

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Example

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}
```

```
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

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

</body>
</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example

Inline styles are defined within the "style" attribute of the relevant element:

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

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

```
/* This is a single-line comment */
p {
  color: red;
}
```

Example

```
p {
  color: red;  /* Set text color to red */
}
```

Example

```
/* This is
a multi-line
comment */

p {
  color: red;
}
```

HTML and CSS Comments

From the HTML tutorial, you learned that you can add comments to your HTML source by using the <!--...--> syntax.

In the following example, we use a combination of HTML and CSS comments:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red; /* Set text color to red */
}
</style>
</head>
<body>

<h2>My Heading</h2>

<!-- These paragraphs will be red -->
<p>Hello World!</p>
<p>This paragraph is styled with CSS.</p>
<p>CSS comments are not shown in the output.</p>
```

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

CSS Background Color

You can set the background color for HTML elements:

Example

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

CSS Text Color

You can set the color of text:

Hello World

Example

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

CSS Border Color

You can set the color of borders:

Hello World

Hello World

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

CSS Color Values

In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

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>
```

The CSS background properties are used to add background effects for elements.

In these chapters, you will learn about the following CSS background properties:

background-color

background-image

background-repeat

background-attachment

background-position

background (shorthand property)

CSS background-color

The background-color property specifies the background color of an element.

Example

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

With CSS, a color is most often specified by:

- a valid color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

Other Elements

You can set the background color for any HTML elements:

Example

Here, the <h1>, <p>, and elements will have different background colors:

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:

opacity 1, opacity 0.6, opacity 0.3, opacity 0.1

Example

```
div {
  background-color: green;
  opacity: 0.3;
}
```

Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:

100% opacity, 60% opacity, 30% opacity, 10% opacity

You learned from our CSS Colors Chapter, that you can use RGB as a color value. In addition to RGB, you can use an RGB color value with an **alpha** channel (RGB**A**) - 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 (fully opaque).

```
div {
  background: rgba(0, 128, 0, 0.3) /* Green background with 30% opacity */
}
```

CSS background-image

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url("paper.gif");
}
```

CSS background-repeat

By default, the background-image property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

```
body {
  background-image: url("gradient_bg.png");
}
```

If the image above is repeated only horizontally (background-repeat: repeat-x;), the background will look better:

```
body {
  background-image: url("gradient_bg.png");
```

```
  background-repeat: repeat-x;
}
```

CSS background-repeat: no-repeat

Showing the background image only once is also specified by the background-repeat property:

Example

Show the background image only once:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

CSS background-position

The background-position property is used to specify the position of the background image.

Example

Position the background image in the top-right corner:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

Example

Specify that the background image should be fixed:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Example

Specify that the background image should scroll with the rest of the page:

```
body {
  background-image: url("img_tree.png");
```

```
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

CSS background - Shorthand property

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

Instead of writing:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-clip	Specifies the painting area of the background
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-origin	Specifies where the background image(s) is/are positioned
background-position	Sets the starting position of a background image

background-repeat	Sets how a background image will be repeated
background-size	Specifies the size of the background image(s)

CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

Example

Demonstration of the different border widths:

```
p.one {
  border-style: solid;
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: dotted;
  border-width: 2px;
}

p.four {
  border-style: dotted;
  border-width: thick;
}
```

Specific Side Widths

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border):

```
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}

p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}

p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
}
```

CSS Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

Note: If border-color is not set, it inherits the color of the element.

Example

Demonstration of the different border colors:

```
p.one {
  border-style: solid;
  border-color: red;
}

p.two {
  border-style: solid;
  border-color: green;
}

p.three {
  border-style: dotted;
  border-color: blue;
}
```

If the border-style property has four values:

- **border-style: dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

If the border-style property has three values:

- **border-style: dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double

If the border-style property has two values:

- **border-style: dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid

If the border-style property has one value:

- **border-style: dotted;**

- all four borders are dotted

```
/* Four values */
p {
  border-style: dotted solid double dashed;
}

/* Three values */
p {
  border-style: dotted solid double;
}

/* Two values */
p {
  border-style: dotted solid;
}

/* One value */
p {
  border-style: dotted;
}
```

CSS Border - Shorthand Property

Like you saw in the previous page, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

```
p {
  border: 5px solid red;
}
```

CSS Rounded Borders

The border-radius property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roundest border

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

All CSS Border Properties

Property	Description
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-	Sets the width of the left border

width	
border-radius	Sets all the four border-*-radius properties for rounded corners
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders

CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

Margin - Shorthand Property

To shorten the code, it is possible to specify all the margin properties in one property.

The margin property is a shorthand property for the following individual margin properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

So, here is how it works:

If the margin property has four values:

- **margin: 25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px

```
p {
  margin: 25px 50px 75px 100px;
}
```

Example

Use the margin shorthand property with three values:

```
p {
  margin: 25px 50px 75px;
}
```

The auto Value

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

Example

Use margin: auto:

```
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
```

The inherit Value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element ( ):

Example

Use of the inherit value:

```
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
}
```

All CSS Margin Properties

Property	Description

margin	A shorthand property for setting all the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

Look at the following example:

Example

Demonstration of margin collapse:

```
h1 {
  margin: 0 0 50px 0;
}

h2 {
  margin: 20px 0 0 0;
}
```

All CSS Margin Properties

Property	Description
margin	A shorthand property for setting all the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element

margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

Example

Set different padding for all four sides of a element:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The padding property is a shorthand property for the following individual padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left

So, here is how it works:

If the padding property has four values:

- **padding: 25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

Example

Use the padding shorthand property with four values:

```
div {
  padding: 25px 50px 75px 100px;
}
```

Example

Use the padding shorthand property with three values:

```
div {
  padding: 25px 50px 75px;
}
```

Example

Use the padding shorthand property with one value:

```
div {
  padding: 25px;
}
```

Padding and Element Width

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element (the box model).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

Example

Here, the element is given a width of 300px. However, the actual width of the element will be 350px (300px + 25px of left padding + 25px of right padding):

```
div {
  width: 300px;
  padding: 25px;
}
```

Example

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

```
div {
  width: 300px;
  padding: 25px;
  box-sizing: border-box;
}
```

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

The CSS height and width properties are used to set the height and width of an element.

The CSS max-width property is used to set the maximum width of an element.

CSS Setting height and width

The height and width properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm, etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

Example

Set the height and width of another element:

```
div {
  height: 100px;
  width: 500px;
  background-color: powderblue;
}
```

Setting max-width

The max-width property is used to set the maximum width of an element.

The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows.

The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Example

Demonstration of the box model:

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

CSS Outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

CSS has the following outline properties:

- outline-style
- outline-color
- outline-width
- outline-offset
- outline

CSS Outline Style

The outline-style property specifies the style of the outline, and can have one of the following values:

- dotted - Defines a dotted outline
- dashed - Defines a dashed outline
- solid - Defines a solid outline
- double - Defines a double outline
- groove - Defines a 3D grooved outline
- ridge - Defines a 3D ridged outline
- inset - Defines a 3D inset outline
- outset - Defines a 3D outset outline
- none - Defines no outline
- hidden - Defines a hidden outline

The following example shows the different outline-style values:

Example

Demonstration of the different outline styles:

```
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
```

```
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

CSS Outline Width

The outline-width property specifies the width of the outline, and can have one of the following values:

- thin (typically 1px)
- medium (typically 3px)
- thick (typically 5px)
- A specific size (in px, pt, cm, em, etc)

Example:

```
p.ex1 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thin;
}

p.ex2 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: medium;
}

p.ex3 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thick;
}

p.ex4 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: 4px;
}
```

CSS Outline Color

The outline-color property is used to set the color of the outline.

The color can be set by:

- name - specify a color name, like "red"

- HEX - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

Example:

```
p.ex1 {
  border: 2px solid black;
  outline-style: solid;
  outline-color: red;
}

p.ex2 {
  border: 2px solid black;
  outline-style: dotted;
  outline-color: blue;
}

p.ex3 {
  border: 2px solid black;
  outline-style: outset;
  outline-color: grey;
}
```

CSS Outline - Shorthand property

The outline property is a shorthand property for setting the following individual outline properties:

- outline-width
- outline-style (required)
- outline-color

The outline property is specified as one, two, or three values from the list above. The order of the values does not matter.

The following example shows some outlines specified with the shorthand outline property:

A dashed outline.

A dotted red outline.

A 5px solid yellow outline.

A thick ridge pink outline.

```
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
```

```
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
```

CSS Outline Offset

The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

Example:

```
p {
  margin: 30px;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
```

All CSS Outline Properties

Property	Description
outline	A shorthand property for setting outline-width, outline-style, and outline-color in one declaration
outline-color	Sets the color of an outline
outline-offset	Specifies the space between an outline and the edge or border of an element
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline

CSS Text

Text Color

The color property is used to set the color of the text. The color is specified by:

- a color name - like "red"
- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"

The default text color for a page is defined in the body selector.

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

Text Color and Background Color

In this example, we define both the background-color property and the color property:

```
body {
  background-color: lightgrey;
  color: blue;
}

h1 {
  background-color: black;
  color: white;
}

div {
  background-color: blue;
  color: white;
}
```

Text Alignment and Text Direction

In this chapter you will learn about the following properties:

- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

```
h1 {
  text-align: center;
}
```

```
h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

```
div {
  text-align: justify;
}
```

Text Align Last

The text-align-last property specifies how to align the last line of a text.

```
p.a {
  text-align-last: right;
}

p.b {
  text-align-last: center;
}

p.c {
  text-align-last: justify;
}
```

Text Direction

The direction and unicode-bidi properties can be used to change the text direction of an element:

```
p {
  direction: rtl;
  unicode-bidi: bidi-override;
}
```

Vertical Alignment

The vertical-align property sets the vertical alignment of an element.

Example

Set the vertical alignment of an image in a text:

```
img.a {
  vertical-align: baseline;
```

```
}

img.b {
  vertical-align: text-top;
}

img.c {
  vertical-align: text-bottom;
}

img.d {
  vertical-align: sub;
}

img.e {
  vertical-align: super;
}
```

The CSS Text Alignment/Direction Properties

Property	Description
direction	Specifies the text direction/writing direction
text-align	Specifies the horizontal alignment of text
text-align-last	Specifies how to align the last line of a text
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element

Text Decoration

In this chapter you will learn about the following properties:

- text-decoration-line

- text-decoration-color
- text-decoration-style
- text-decoration-thickness
- text-decoration

Add a Decoration Line to Text

The text-decoration-line property is used to add a decoration line to text.

Tip: You can combine more than one value, like overline and underline to display lines both over and under a text.

```
h1 {
  text-decoration-line: overline;
}

h2 {
  text-decoration-line: line-through;
}

h3 {
  text-decoration-line: underline;
}

p {
  text-decoration-line: overline underline;
}
```

Specify a Color for the Decoration Line

The text-decoration-color property is used to set the color of the decoration line.

```
h1 {
  text-decoration-line: overline;
  text-decoration-color: red;
}

h2 {
  text-decoration-line: line-through;
  text-decoration-color: blue;
}

h3 {
  text-decoration-line: underline;
  text-decoration-color: green;
}

p {
  text-decoration-line: overline underline;
```

```
  text-decoration-color: purple;
}
```

Specify a Style for the Decoration Line

The text-decoration-style property is used to set the style of the decoration line.

```
h1 {
  text-decoration-line: underline;
  text-decoration-style: solid;
}

h2 {
  text-decoration-line: underline;
  text-decoration-style: double;
}

h3 {
  text-decoration-line: underline;
  text-decoration-style: dotted;
}

p.ex1 {
  text-decoration-line: underline;
  text-decoration-style: dashed;
}

p.ex2 {
  text-decoration-line: underline;
  text-decoration-style: wavy;
}

p.ex3 {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: wavy;
}
```

Specify the Thickness for the Decoration Line

The text-decoration-thickness property is used to set the thickness of the decoration line.

```
h1 {
  text-decoration-line: underline;
  text-decoration-thickness: auto;
}

h2 {
  text-decoration-line: underline;
```

```
  text-decoration-thickness: 5px;
}

h3 {
  text-decoration-line: underline;
  text-decoration-thickness: 25%;
}

p {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: double;
  text-decoration-thickness: 5px;
}
```

The Shorthand Property

The text-decoration property is a shorthand property for:

- text-decoration-line (required)
- text-decoration-color (optional)
- text-decoration-style (optional)
- text-decoration-thickness (optional)

```
h1 {
  text-decoration: underline;
}

h2 {
  text-decoration: underline red;
}

h3 {
  text-decoration: underline red double;
}

p {
  text-decoration: underline red double 5px;
}
```

All CSS text-decoration Properties

Property	Description
text-decoration	Sets all the text-decoration properties in one declaration

text-decoration-color	Specifies the color of the text-decoration
text-decoration-line	Specifies the kind of text decoration to be used (underline, overline, etc.)
text-decoration-style	Specifies the style of the text decoration (solid, dotted, etc.)
text-decoration-thickness	Specifies the thickness of the text decoration line

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

```
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
```

Text Spacing

In this chapter you will learn about the following properties:

- text-indent
- letter-spacing
- line-height
- word-spacing
- white-space

Text Indentation

The text-indent property is used to specify the indentation of the first line of a text:

```
p {
  text-indent: 50px;
}
```

Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

```
h1 {
  letter-spacing: 5px;
}

h2 {
  letter-spacing: -2px;
}
```

Line Height

The line-height property is used to specify the space between lines:

```
p.small {
  line-height: 0.8;
}

p.big {
  line-height: 1.8;
}
```

Word Spacing

The word-spacing property is used to specify the space between the words in a text.

The following example demonstrates how to increase or decrease the space between words:

```
p.one {
  word-spacing: 10px;
}

p.two {
  word-spacing: -2px;
}
```

White Space

The white-space property specifies how white-space inside an element is handled.

This example demonstrates how to disable text wrapping inside an element:

```
p {
  white-space: nowrap;
}
```

The CSS Text Spacing Properties

Property	Description
letter-spacing	Specifies the space between characters in a text
line-height	Specifies the line height
text-indent	Specifies the indentation of the first line in a text-block
white-space	Specifies how to handle white-space inside an element
word-spacing	Specifies the space between words in a text

Text Shadow

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Text shadow effect!

```
h1 {
  text-shadow: 2px 2px;
}
```

Text shadow effect!

```
h1 {
  text-shadow: 2px 2px red;
}
```

Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Example

Specify some different fonts for three paragraphs:

```
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}

p {
font-family: Tahoma, Verdana, sans-serif;
}
```

Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
```

Font Weight

The font-weight property specifies the weight of a font:

```
p.normal {
  font-weight: normal;
}

p.thick {
  font-weight: bold;
}
```

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

```
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
```

Font Size

The font-size property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

```
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
}

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
```

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

```
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
```

```
  font-size: 0.875em;
}
```

How To Use Google Fonts

Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

Example

Here, we want to use a font named "Sofia" from Google Fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>
```

Use Multiple Google Fonts

To use multiple Google fonts, just separate the font names with a pipe character (|), like this:

Example

Request multiple fonts:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">
<style>
h1.a {font-family: "Audiowide", sans-serif;}
h1.b {font-family: "Sofia", sans-serif;}
h1.c {font-family: "Trirong", serif;}
</style>
</head>
```

Styling Google Fonts

Of course you can style Google Fonts as you like, with CSS!

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
  font-size: 30px;
  text-shadow: 3px 3px 3px #abababab;
}
</style>
</head>
```

Example

Add multiple effects to the "Sofia" font:

```
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia&effect=neon|outline|emboss|shadow-multiple">
<style>
body {
  font-family: "Sofia", sans-serif;
  font-size: 30px;
}
</style>
</head>
<body>

<h1 class="font-effect-neon">Neon Effect</h1>
<h1 class="font-effect-outline">Outline Effect</h1>
<h1 class="font-effect-emboss">Emboss Effect</h1>
<h1 class="font-effect-shadow-multiple">Multiple Shadow Effect</h1>

</body>
```

Font Pairing Rules

Here are some basic rules to create great font pairings:

1. Complement

It is always safe to find font pairings that complement one another.

A great font combination should harmonize, without being too similar or too different.

2. Use Font Superfamilies

A font superfamily is a set of fonts designed to work well together. So, using different fonts within the same superfamily is safe.

For example, the Lucida superfamily contains the following fonts: Lucida Sans, Lucida Serif, Lucida Typewriter Sans, Lucida Typewriter Serif and Lucida Math.

3. Contrast is King

Two fonts that are too similar will often conflict. However, contrasts, done the right way, brings out the best in each font.

Example: Combining serif with sans serif is a well known combination.

A strong superfamily includes both serif and sans serif variations of the same font (e.g. Lucida and Lucida Sans).

4. Choose Only One Boss

One font should be the boss. This establishes a hierarchy for the fonts on your page. This can be achieved by varying the size, weight and color.

```
body {
  background-color: black;
  font-family: Verdana, sans-serif;
  font-size: 16px;
  color: gray;
}

h1 {
  font-family: Georgia, serif;
  font-size: 60px;
  color: white;
}
```

The CSS Font Property

To shorten the code, it is also possible to specify all the individual font properties in one property.

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

Note: The font-size and font-family values are required. If one of the other values is missing, their default value are used.

Example

Use font to set several font properties in one declaration:

```
p.a {
  font: 20px Arial, sans-serif;
}

p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}
```

How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like <i> or).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Awesome Icons

To use the Font Awesome icons, go to fontawesome.com, sign in, and get a code to add in the <head> section of your HTML page:

```
<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>
```

```
<!DOCTYPE html>
<html>
<head>
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
</head>
<body>

<i class="fas fa-cloud"></i>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>

</body>
</html>
```

Bootstrap Icons

To use the Bootstrap icons, add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>

<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>

</body>
</html>
```

Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>

</body>
</html>
```

CSS Links

With CSS, links can be styled in many different ways.

Text Link Text Link Link Button Link Button

Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

```
a {
  color: hotpink;
}
```

In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

```
/* unvisited link */
a:link {
  color: red;
}
```

```
/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

```
a:link {
  text-decoration: none;
}

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

a:active {
  text-decoration: underline;
}
```

Background Color

The background-color property can be used to specify a background color for links:

```
a:link {
  background-color: yellow;
}
```

```
a:visited {
  background-color: cyan;
}

a:hover {
  background-color: lightgreen;
}

a:active {
  background-color: hotpink;
}
```

Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

```
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 14px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}

a:hover, a:active {
  background-color: red;
}
```

Example

This example demonstrates how to add other styles to hyperlinks:

```
a.one:link {color: #ff0000;}
a.one:visited {color: #0000ff;}
a.one:hover {color: #ffcc00;}

a.two:link {color: #ff0000;}
a.two:visited {color: #0000ff;}
a.two:hover {font-size: 150%;}

a.three:link {color: #ff0000;}
a.three:visited {color: #0000ff;}
a.three:hover {background: #66ff66;}

a.four:link {color: #ff0000;}
a.four:visited {color: #0000ff;}
a.four:hover {font-family: monospace;}
```

```
a.five:link {color: #ff0000; text-decoration: none;}
a.five:visited {color: #0000ff; text-decoration: none;}
a.five:hover {text-decoration: underline;}
```

JavaScript

JavaScript is the world's most popular programming language. JavaScript is the programming language of the Web. JavaScript is easy to learn. This tutorial will teach you JavaScript from basic to advanced.

JavaScript and Java are completely different languages, both in concept and design. JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997.

JavaScript Can Change HTML Content

One of many JavaScript HTML methods is getElementById().

The example below "finds" an HTML element (with id="demo"), and changes the element content (innerHTML) to "Hello JavaScript":

Example

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

JavaScript accepts both double and single quotes

JavaScript Can Change HTML Styles (CSS)

Changing the style of an HTML element, is a variant of changing an HTML attribute:

Example

```
document.getElementById("demo").style.fontSize = "35px";
```

JavaScript Can Hide HTML Elements

Hiding HTML elements can be done by changing the display style:

Example

```
document.getElementById("demo").style.display = "none";
```

JavaScript Can Show HTML Elements

Showing hidden HTML elements can also be done by changing the display style:

Example

```
document.getElementById("demo").style.display = "block";
```

The <script> Tag

In HTML, JavaScript code is inserted between <script> and </script> tags.

Example

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

JavaScript Functions and Events

A JavaScript `function` is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.

Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.

The function is invoked (called) when a button is clicked:

Example

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

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

External JavaScript

Scripts can also be placed in external files:

External file: myScript.js

```
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
```

External scripts are practical when the same code is used in many different web pages.

JavaScript files have the file extension **.js**.

To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

Example

```
<script src="myScript.js"></script>
```

You can place an external script reference in <head> or <body> as you like.

The script will behave as if it was located exactly where the <script> tag is located.

External scripts cannot contain <script> tags.

External JavaScript Advantages

Placing scripts in external files has some advantages:

- It separates HTML and code
- It makes HTML and JavaScript easier to read and maintain
- Cached JavaScript files can speed up page loads

To add several script files to one page - use several script tags:

Example

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

External References

An external script can be referenced in 3 different ways:

- With a full URL (a full web address)
- With a file path (like /js/)
- Without any path

This example uses a **full URL** to link to myScript.js:

Example

```
<script src="https://www.w3schools.com/js/myScript.js"></script>
```

This example uses a **file path** to link to myScript.js:

Example

```
<script src="/js/myScript.js"></script>
```

This example uses no path to link to myScript.js:

Example

```
<script src="myScript.js"></script>
```

JavaScript Display Possibilities

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using inner HTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content:

Example

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

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Using document.write()

For testing purposes, it is convenient to use document.write():

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<button type="button" onclick="document.write(5 + 6)">Try it</button>

</body>
</html>
```

Using window.alert()

You can use an alert box to display data:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

You can skip the window keyword.

In JavaScript, the window object is the global scope object. This means that variables, properties, and methods by default belong to the window object. This also means that specifying the window keyword is optional:

Example

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

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
alert(5 + 6);
</script>

</body>
</html>
```

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

Example

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

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript Print: JavaScript does not have any print object or print methods. You cannot access output devices from JavaScript. The only exception is that you can call the window.print() method in the browser to print the content of the current window.

Example

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

<button onclick="window.print()">Print this page</button>

</body>
</html>
```

JavaScript Statements

Example

```
let x, y, z;   // Statement 1
x = 5;         // Statement 2
y = 6;         // Statement 3
z = x + y;     // Statement 4
```

JavaScript Programs

A **computer program** is a list of "instructions" to be "executed" by a computer.

In a programming language, these programming instructions are called **statements**.

A **JavaScript program** is a list of programming **statements**.

In HTML, JavaScript programs are executed by the web browser.

JavaScript Statements

JavaScript statements are composed of:

Values, Operators, Expressions, Keywords, and Comments.

This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

Example

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

Semicolons ;

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

Examples

```
let a, b, c;  // Declare 3 variables
a = 5;        // Assign the value 5 to a
b = 6;        // Assign the value 6 to b
c = a + b;    // Assign the sum of a and b to c
```

JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

```
let person = "Hege";
let person="Hege";
```

JavaScript Line Length and Line Breaks

For best readability, programmers often like to avoid code lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it is after an operator:

Example

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

JavaScript Code Blocks

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.

The purpose of code blocks is to define statements to be executed together.

One place you will find statements grouped together in blocks, is in JavaScript functions:

Example

```
function myFunction() {
  document.getElementById("demo1").innerHTML = "Hello Dolly!";
  document.getElementById("demo2").innerHTML = "How are you?";
}
```

JavaScript Keywords

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant

if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
// How to create variables:
var x;
let y;

// How to use variables:
x = 5;
y = 6;
let z = x + y;
```

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called **Literals**. Variable values are called **Variables**.

JavaScript Literals

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

10.50

1001

2. **Strings** are text, written within double or single quotes:

"John Doe"

'John Doe'

JavaScript Variables

In a programming language, **variables** are used to **store** data values. JavaScript uses the keywords var, let and const to **declare** variables.

An **equal sign** is used to **assign values** to variables. In this example, x is defined as a variable. Then, x is assigned (given) the value 6:

```
let x;
x = 6;
```

JavaScript Operators

JavaScript uses **arithmetic operators** (+ - * /) to **compute** values:

```
(5 + 6) * 10
```

JavaScript uses an **assignment operator** (=) to **assign** values to variables:

```
let x, y;
x = 5;
y = 6;
```

JavaScript Expressions

An expression is a combination of values, variables, and operators, which computes to a value. The computation is called an evaluation.

For example, 5 * 10 evaluates to 50:

```
5 * 10
```

The values can be of various types, such as numbers and strings.

For example, "John" + " " + "Doe", evaluates to "John Doe":

```
"John" + " " + "Doe"
```

JavaScript Keywords

JavaScript **keywords** are used to identify actions to be performed.

The let keyword tells the browser to create variables:

```
let x, y;
x = 5 + 6;
y = x * 10;
```

The var keyword also tells the browser to create variables:

```
var x, y;
x = 5 + 6;
y = x * 10;
```

JavaScript Comments

Not all JavaScript statements are "executed".

Code after double slashes // or between /* and */ is treated as a **comment**.

Comments are ignored, and will not be executed:

```
let x = 5;   // I will be executed

// x = 6;   I will NOT be executed
```

JavaScript Identifiers / Names

Identifiers are JavaScript names. Identifiers are used to name variables and keywords, and functions. The rules for legal names are the same in most programming languages.

A JavaScript name must begin with:

- A letter (A-Z or a-z)
- A dollar sign ($)
- Or an underscore (_)

Subsequent characters may be letters, digits, underscores, or dollar signs.

JavaScript is Case Sensitive

All JavaScript identifiers are **case sensitive**.

The variables lastName and lastname, are two different variables:

```
let lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
```

Hyphens are not allowed in JavaScript. They are reserved for subtractions.

JavaScript Character Set

JavaScript uses the **Unicode** character set. Unicode covers (almost) all the characters, punctuations, and symbols in the world.

JavaScript Comments

Single Line Comments

Single line comments start with / /.

Any text between / / and the end of the line will be ignored by JavaScript (will not be executed).

This example uses a single-line comment before each code line:

Example

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";

// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";
```

This example uses a single line comment at the end of each line to explain the code:

Example

```
let x = 5;      // Declare x, give it the value of 5
let y = x + 2;  // Declare y, give it the value of x + 2
```

Multi-line Comments

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

This example uses a multi-line comment (a comment block) to explain the code:

Example

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

JavaScript Variables

4 Ways to Declare a JavaScript Variable:

- Using var
- Using let
- Using const
- Using nothing

- What are Variables?
- Variables are containers for storing data (storing data values).
- In this example, x, y, and z, are variables, declared with the var keyword:

Example

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

In this example, x, y, and z, are variables, declared with the let keyword:

Example

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

In this example, x, y, and z, are undeclared variables:

Example

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

When to Use JavaScript var?

Always declare JavaScript variables with var, let, or const. The var keyword is used in all JavaScript code from 1995 to 2015. The let and const keywords were added to JavaScript in 2015. If you want your code to run in older browsers, you must use var.

When to Use JavaScript const?

If you want a general rule: always declare variables with const. If you think the value of the variable can change, use let. In this example, price1, price2, and total, are variables:

Example

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

The two variables `price1` and `price2` are declared with the `const` keyword. These are constant values and cannot be changed. The variable `total` is declared with the `let` keyword. This is a value that can be changed.

JavaScript Identifiers

All JavaScript **variables** must be **identified** with **unique names**. These unique names are called **identifiers**. Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter.
- Names can also begin with $ and _ (but we will not use it in this tutorial).
- Names are case sensitive (y and Y are different variables).
- Reserved words (like JavaScript keywords) cannot be used as names.

JavaScript Dollar Sign $

Since JavaScript treats a dollar sign as a letter, identifiers containing $ are valid variable names:

Example

```
let $ = "Hello World";
let $$$ = 2;
let $myMoney = 5;
```

JavaScript Let

The `let` keyword was introduced in 2015.

Variables defined with `let` cannot be Re-declared. Variables defined with `let` must be Declared before use. Variables defined with `let` have Block Scope. Variables defined with let cannot be **redeclared**.

With let you can not do this:

Example

```
let x = "John Doe";

let x = 0;

// SyntaxError: 'x' has already been declared
```

With var you can:

Example

```
var x = "John Doe";

var x = 0;
```

JavaScript Const

Variables defined with `const` cannot be Re-declared. Variables defined with `const` cannot be Re-assigned. A const variable cannot be reassigned:

Example

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

When to use JavaScript const?

Always declare a variable with const when you know that the value should not be changed.

Use const when you declare:

- A new Array
- A new Object
- A new Function
- A new RegExp

Constant Arrays

You can change the elements of a constant array:

Example

```
// You can create a constant array:
const cars = ["Saab", "Volvo", "BMW"];

// You can change an element:
cars[0] = "Toyota";

// You can add an element:
cars.push("Audi");
```

Constant Objects

You can change the properties of a constant object:

Example

```
// You can create a const object:
const car = {type:"Fiat", model:"500", color:"white"};

// You can change a property:
car.color = "red";

// You can add a property:
car.owner = "Johnson";
```

JavaScript Operators

Assignment Examples

```
let x = 10;

// Assign the value 5 to x
let x = 5;
// Assign the value 2 to y
let y = 2;
// Assign the value x + y to z:
let z = x + y;
```

Adding

```
let x = 5;
let y = 2;
let z = x + y;
```

The **Multiplication Operator** (*) multiplies numbers.

Multiplying

```
let x = 5;
let y = 2;
let z = x * y;
```

Types of JavaScript Operators

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Conditional Operators
- Type Operators

JavaScript Arithmetic Operators

Arithmetic Operators are used to perform arithmetic on numbers:

Arthmetic Operators Example

```
let a = 3;
let x = (100 + 50) * a;
```

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

**=	x **= y	x = x ** y

Adding JavaScript Strings

The + operator can also be used to add (concatenate) strings.

Example

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

The result of text3 will be:

John Doe

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript Logical Operators

Operator	Description

&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

JavaScript Arithmetic Operators

Example

```
let x = 100 + 50;
```

Example

```
let x = (100 + 50) * a;
```

Adding

The **addition** operator (+) adds numbers:

Example

```
let x = 5;
let y = 2;
let z = x + y;
```

Subtracting

The **subtraction** operator (-) subtracts numbers.

Example

```
let x = 5;
let y = 2;
let z = x - y;
```

Multiplying

The **multiplication** operator (*) multiplies numbers.

Example

```
let x = 5;
let y = 2;
let z = x * y;
```

Dividing

The **division** operator (/) divides numbers.

Example

```
let x = 5;
let y = 2;
let z = x / y;
```

Remainder

The **modulus** operator (%) returns the division remainder.

Example

```
let x = 5;
let y = 2;
let z = x % y;
```

Note: In arithmetic, the division of two integers produces a **quotient** and a **remainder**. In mathematics, the result of a **modulo operation** is the **remainder** of an arithmetic division.

Incrementing

The **increment** operator (++) increments numbers.

Example

```
let x = 5;
x++;
let z = x;
```

Decrementing

The **decrement** operator (--) decrements numbers.

Example

```
let x = 5;
x--;
let z = x;
```

Exponentiation

The **exponentiation** operator (**) raises the first operand to the power of the second operand.

Example

```
let x = 5;
let z = x ** 2;
```

x ** y produces the same result as `Math.pow(x,y)`:

Example

```
let x = 5;
let z = Math.pow(x,2);
```

The += Operator

The **Addition Assignment Operator** adds a value to a variable.

Addition Assignment Examples

```
let x = 10;
x += 5;
```

The -= Operator

The **Subtraction Assignment Operator** subtracts a value from a variable.

Subtraction Assignment Example

```
let x = 10;
x -= 5;
```

The *= Operator

The **Multiplication Assignment Operator** multiplies a variable.

Multiplication Assignment Example

```
let x = 10;
x *= 5;
```

The **= Operator

The **Exponentiation Assignment Operator** raises a variable to the power of the operand.

Exponentiation Assignment Example

```
let x = 10;
x **= 5;
```

The /= Operator

The **Division Assignment Operator** divides a variable.

Division Assignment Example

```
let x = 10;
x /= 5;
```

The %= Operator

The **Remainder Assignment Operator** assigns a remainder to a variable.

Remainder Assignment Example

```
let x = 10;
x %= 5;
```

The <<= Operator

The **Left Shift Assignment Operator** left shifts a variable.

Left Shift Assignment Example

```
let x = -100;
x <<= 5;
```

The >>= Operator

The **Right Shift Assignment Operator** right shifts a variable (signed).

Right Shift Assignment Example

```
let x = -100;
x >>= 5;
```

The >>>= Operator

The **Unsigned Right Shift Assignment Operator** right shifts a variable (unsigned).

Unsigned Right Shift Assignment Example

```
let x = -100;
x >>>= 5;
```

The &= Operator

The **Bitwise AND Assignment Operator** does a bitwise AND operation on two operands and assigns the result to the the variable.

Bitwise AND Assignment Example

```
let x = 10;
x &= 5;
```

The |= Operator

The **Bitwise OR Assignment Operator** does a bitwise OR operation on two operands and assigns the result to the variable.

Bitwise OR Assignment Example

```
let x = 10;
x |= 5;
```

The ^= Operator

The **Bitwise XOR Assignment Operator** does a bitwise XOR operation on two operands and assigns the result to the variable.

Bitwise XOR Assignment Example

```
let x = 10;
x ^= 5;
```

The &&= Operator

The **Logical AND assignment operator** is used between two values.

If the first value is true, the second value is assigned.

Logical AND Assignment Example

```
let x = 10;
x &&= 5;
```

The ||= Operator

The **Logical OR assignment operator** is used between two values.

If the first value is false, the second value is assigned.

Logical OR Assignment Example

```
let x = 10;
x ||= 5;
```

The ??= Operator

The **Nullish coalescing assignment operator** is used between two values.

If the first value is undefined or null, the second value is assigned.

Nullish Coalescing Assignment Example

```
let x = 10;
x ??= 5;
```

JavaScript has 8 Datatypes

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

The Object Datatype

The object data type can contain:

1. An object
2. An array
3. A date

Examples

```
// Numbers:
let length = 16;
```

```
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
let x = "16" + "Volvo";
```

Note

When adding a number and a string, JavaScript will treat the number as a string.

Example

```
let x = 16 + "Volvo";
```

Example

```
let x = "Volvo" + 16;
```

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

Example:

```
let x = 16 + 4 + "Volvo";
```

JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

Example

```
let x;       // Now x is undefined
x = 5;       // Now x is a Number
x = "John";  // Now x is a String
```

JavaScript Strings

A string (or a text string) is a series of characters like "John Doe". Strings are written with quotes. You can use single or double quotes:

Example

```
// Using double quotes:
let carName1 = "Volvo XC60";

// Using single quotes:
let carName2 = 'Volvo XC60';
```

You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

Example

```
// Single quote inside double quotes:
let answer1 = "It's alright";

// Single quotes inside double quotes:
let answer2 = "He is called 'Johnny'";

// Double quotes inside single quotes:
let answer3 = 'He is called "Johnny"';
```

JavaScript Numbers

All JavaScript numbers are stored as decimal numbers (floating point). Numbers can be written with, or without decimals:

Example

```
// With decimals:
let x1 = 34.00;

// Without decimals:
let x2 = 34;
```

Exponential Notation

Extra large or extra small numbers can be written with scientific (exponential) notation:

Example

```
let y = 123e5;    // 12300000
let z = 123e-5;   // 0.00123
```

Javascript are always one type:
double (64-bit floating point).

JavaScript Arrays

JavaScript arrays are written with square brackets. Array items are separated by commas. The following code declares (creates) an array called cars, containing three items (car names):

Example

```
const cars = ["Saab", "Volvo", "BMW"];
```

JavaScript Objects

JavaScript objects are written with curly braces { }. Object properties are written as name:value pairs, separated by commas.

Example

```
const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

Example

```
typeof ""              // Returns "string"
typeof "John"          // Returns "string"
typeof "John Doe"      // Returns "string"
```

Undefined

In JavaScript, a variable without a value, has the value undefined. The type is also undefined.

Example

```
let car;    // Value is undefined, type is undefined
```

Any variable can be emptied, by setting the value to undefined. The type will also be undefined.

Example

```
car = undefined;    // Value is undefined, type is undefined
```