**HTML**

Hypertext Markup Language

HTML is not programming language

Markup for creating webpages/documents

HTML is extremely important & relevant but is also just the beggining

**DOCTYPE**

The HTML document type declaration, also known as DOCTYPE, is the first line of code required in every HTML or XHTML document. The DOCTYPE declaration is an instruction to the web browser about what version of HTML the page is written in. This ensures that the web page is parsed the same way by different web browsers.

In HTML 4.01, the DOCTYPE declaration refers to a document type definition (DTD). A DTD defines the structure and the legal elements of an XML document. Because HTML 4.01 was based on the Standard Generalised Markup Language (SGML), referring to a DTD in the DOCTYPE declaration was necessary.

Additionally, doctypes for HTML 4.01 required the declaration of either strict, transitional, or frameset DTD, each with a different use case as outlined below.

Strict DTD: Used for web pages that exclude attributes and elements that W3C expects to phase out as CSS support grows

Transitional DTD: Used for web pages that include attributes and elements that W3C expects to phase out as CSS support grows

Frameset DTD: Used for web pages with frames

In contrast, the declaration of HTML5 DOCTYPE is much simpler: it no longer requires a reference to DTDs as it is no longer based on SGML. See the examples below for a comparison between HTML 4.01 and HTML5 DOCTYPEs.

**Examples**

Doctype syntax for HTML5 and beyound:

<!DOCTYPE html>

Doctype syntax for strict HTML 4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">

Doctype syntax for transitional HTML 4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

Doctype syntax for frameset HTML 4.01:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">

**HTML Tag**

The <html> tag represents the root of an HTML document.

The <html> tag is the container for all other HTML elements (except for the <!DOCTYPE> tag).

**Head Tag**

The HTML <head> element is used as a container for metadata (data about data). It is used between <html> tag and <body> tag.

The head of an HTML document is a part whose content is not displayed in the browser on page loading. It just contains metadata about the HTML document which specifies data about the HTML document.

An HTML head can contain lots of metadata information or can have very less or no information, it depends on our requirement. But head part has a crucial role an HTML document while creating a website.

Metadata defines the document title, character set, styles, links, scripts, and other meta information.

**Body Tag**

HTML <body> tag defines the main content of an HTML document which displays on the browser. It can contain text content, paragraphs, headings, images, tables, links, videos, etc.

The <body> must be the second element after the <head> tag or it should be placed between </head> and </html> tags. This tag is required for every HTML document and should only use once in the whole HTML document.

**Meta Tag**

The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.

<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

Metadata will not be displayed on the page, but is machine parsable.

Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.

There is a method to let web designers take control over the viewport (the user's visible area of a web page), through the <meta> tag (See "Setting The Viewport" example below).

**HTML Block and Inline elements**

* **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<div>.

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

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

**Here are the block-level elements in HTML:**

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

* Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

**Here are the inline elements in HTML:**

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

NOTE: An inline element cannot contain a block-level element.

## **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

**Difference betweem div and span**

Span and div are both generic HTML elements that group together related parts of a web page. However, they serve different functions. A div element is used for block-level organization and styling of page elements, whereas a span element is used for inline organization and styling.

### **Div**

The [**div (division) element**](https://blog.hubspot.com/website/what-div-in-html)is a generic block-level element, most often used to divide page content into blocks. A block element is a page element that starts a new line and has a width equal to the entire page or the parent container.

You’ll very often see divs used to group related paragraphs, images, headings, and links. For example, a three-paragraph article may be enclosed in a div, and a navigation menu containing links might be enclosed in another div. Using divs this way makes it easier to identify different sections of a page and apply styling to them with [**CSS**](https://blog.hubspot.com/website/css-tutorial).

**Example**

<div id=“paragraphs”>  
 <p>This is my first paragraph.</p>  
 <p>This is my second paragraph.</p>  
 <p>This is my final paragraph.</p>   
</div>

### **Span** The span element is a generic inline element, typically used to apply styling to a portion of inline content. An inline element does not start a new line and only takes up as much space on the page as its content. Span tags are used on small segments of text, links, images, and other HTML elements that appear inline with the surrounding content.

**Example**

<p>This is a paragraph with <span id=“special-text”>a little something extra</span> inside it.</p>



**CSS**

Cascading Style Sheets

Styling/Stylesheet language

User for styling HTML elements

Can be extended with SASS/Less

**Basic selectors**

## **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.

#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.

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

## **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.

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

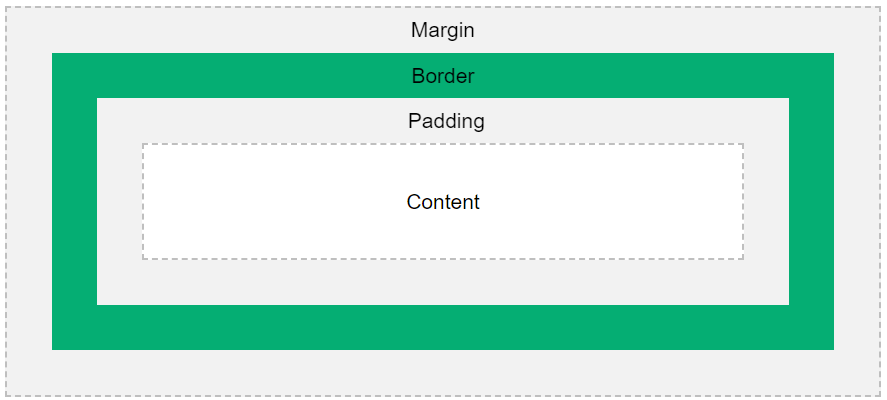
## **The CSS Universal Selector**

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

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

## **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.

**Demostratios of the box model:**

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

## **Width and Height of an Element**

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

**Important:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

**Example**  
This <div> element will have a total width of 350px:

div{  
 width: 320px;  
 padding: 10px;  
 border: 5px solid gray;  
 margin: 0;  
}

**Here is the calculation:**

320px (width)  
+ 20px (left + right padding)  
+ 10px (left + right border)  
+ 0px (left + right margin)  
= 350px

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

**The total height of an element should be calculated like this:**

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

**CSS Margins**

Margins are used to create space around elements, outside of any defined borders.

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

**Tip:** Negative values are allowed.

Example

**Set different margins for all four sides of a <p> 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

**If the margin property has three values:**

* **margin: 25px 50px 75px;**
  + top margin is 25px
  + right and left margins are 50px
  + bottom margin is 75px

**If the margin property has two values:**

* **margin: 25px 50px;**
  + top and bottom margins are 25px
  + right and left margins are 50px

**If the margin property has one value:**

* **margin: 25px;**
  + all four margins are 25px

## **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.

* Margin: auto;

## **The inherit Value**

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

margin-left: inherit;

**CSS Padding**

Padding is used to create space around an element's content, inside of any defined borders.

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).

**Display**

**Position**

Static: Not affected by tblr(top, bottom, left, right) properties/values (by default)

Relative: tblr values cause element to be move from its normal position

Absolute: Positioned relative to its parent elemtn that is positioned “relative”

Fixed: Positioned relative to the viewport

Sticky: Positioned based on scroll position