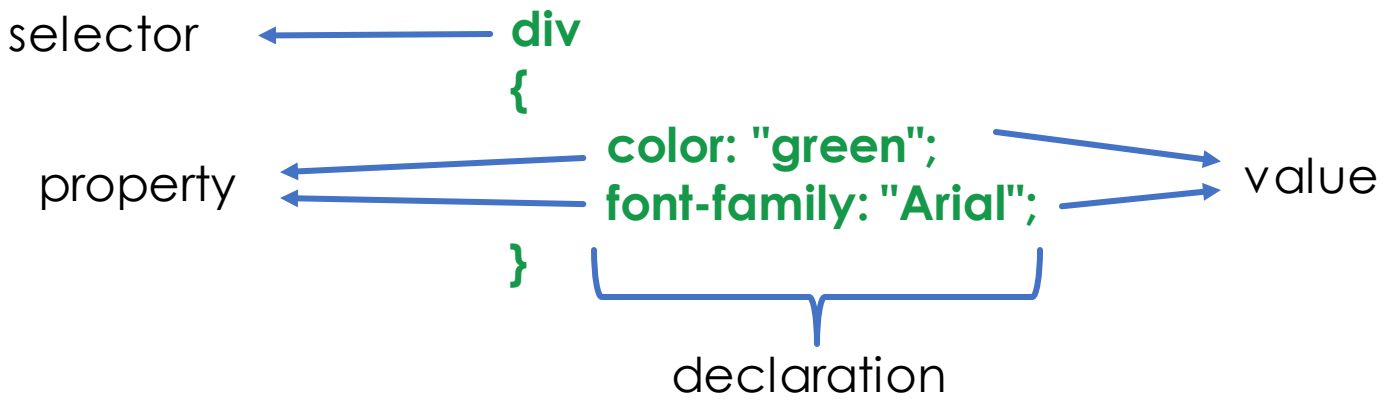
CSS & HTML DOM

CSS

Introduction

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- It is used to style HTML elements.

Syntax



How to add CSS

- **Inline CSS**

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

- **Internal CSS**

```
<head>
  <style type = text/css>
    body {background-color: blue;}
    p { color: yellow;}
  </style>
</head>
```

- **External CSS**

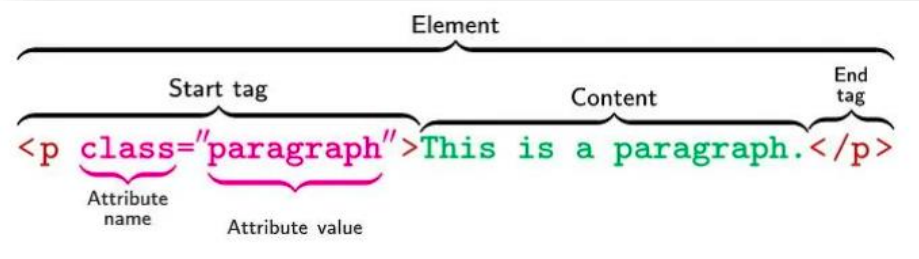
```
<head>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

This is a paragraph

This is a paragraph

Classes

A CSS class is an attribute used to define a group of HTML elements in order to apply unique styling and formatting to those elements with CSS.



HTML

```
<h2>This is my first heading.</h2>
<p>This is my first paragraph</p>
<h2 class="bright">This is my second
heading.</h2>
<p>This is my second paragraph</p>
<h2 class="bright">This is my third
heading.</h2>
<p class="bright">This is my third
paragraph</p>
```

CSS

```
.bright {
  color: orange;
  font-family: Avenir;
}
```

This is my first heading.

This is my first paragraph

This is my second heading.

This is my second paragraph

This is my third heading.

This is my third paragraph

Selectors

A CSS **selector** is the first part of a CSS Rule.

It is a pattern of elements and other terms that tell the browser which HTML elements should be selected to have the CSS property values inside the rule applied to them.

```css
/* universal selector */
* {
  margin: 0px;
  padding: 0px;
}
/* element selector */
h1 {
  color: red;
  font-size: 16px;
}

/* class selector */
.profile {
  /* property: value*/
  color: red;
  font-size: 16px;
}

/* id selector */
#profile {
  color: red;
  font-size: 16px;
}
```

Combinators

A CSS selector can contain more than one simple selector.

Between the simple selectors, we can include a **combinator**.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

```css
/* Descendant Selector */
/* Select <p> inside <div>*/
div p {
    background-color: yellow;
  }

/* Child Selector (>) */
/* Select <p> with direct parent <div>*/
div > p {
    background-color: red;
  }

/* Adjacent Sibling Selector (+) */
/* Select adjacent <p> which is a sibling of <div>*/
div + p {
    background-color: blue;
}

/* General Sibling Selector (~) */
/* Select all <p> which are siblings of <div>*/
div ~ p {
    background-color: green;
  }
```

Pseudo Classes

- A CSS ***pseudo-class*** is a keyword added to a selector that specifies a special state of the selected element(s).
- There are many such pseudo classes, namely, *:active, :disabled, :focus, :hover, :first-child, :last-child, :nth-child,* etc.
- For example, **:hover** can be used to change a button's color when the user's pointer hovers over it.

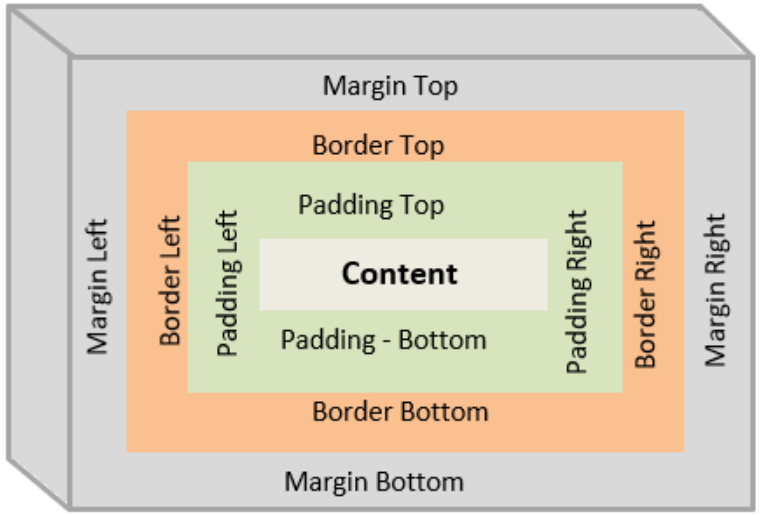
```
/* Any button over which the user's pointer is hovering */
button:hover {
  color: blue;
}
```

Padding, Margin

CSS **paddings** are used to create space around the element, inside any defined border.

CSS **margins** are used to create space around the element.

We can set the different sizes of paddings and margins for individual sides (top, right, bottom, left).



Example: padding: 10px;, padding: 5px 2px 5px 2 px; , padding: 5px 10px;

Example: margin: 10px; , margin: 5px 2px 5px 2 px; , margin: 5px 10px;

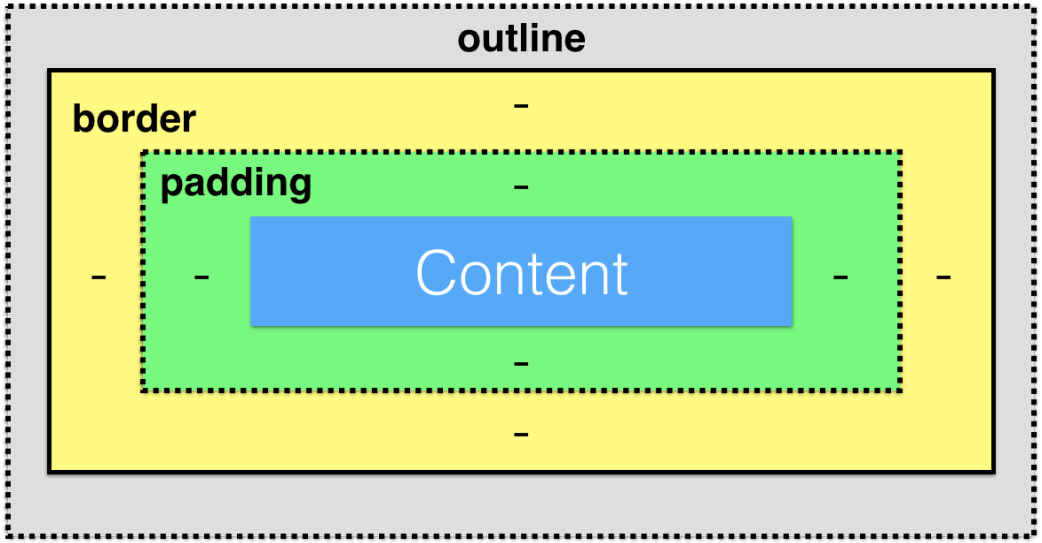
Border, Outline

The **border** property sets an element's border. It sets the values of border-width, border-style, and border-color.

An **outline** is a line drawn outside the element's border.

Example: *border: 1px solid black;*

Example: *outline: 2px solid green;*



This element has a black border and a green outline with a width of 10px.

Shadow-Box, Z-Index

- **box-shadow** attaches shadows to the elements.

Syntax: box-shadow: <*h-offset v-offset blur spread color*> | none;

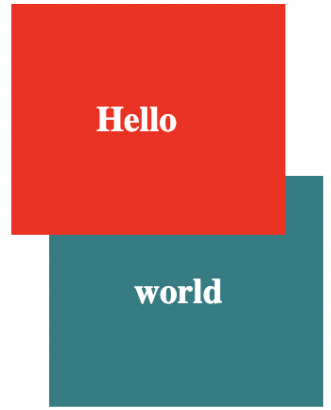
Example: *box-shadow: 5px 10px 8px 10px #888888;*

box-shadow

- **z-index** specifies the stack order of an element. Elements with greater value will be in front of the elements with lower value.

Example: *z-index: 100;*

z-index



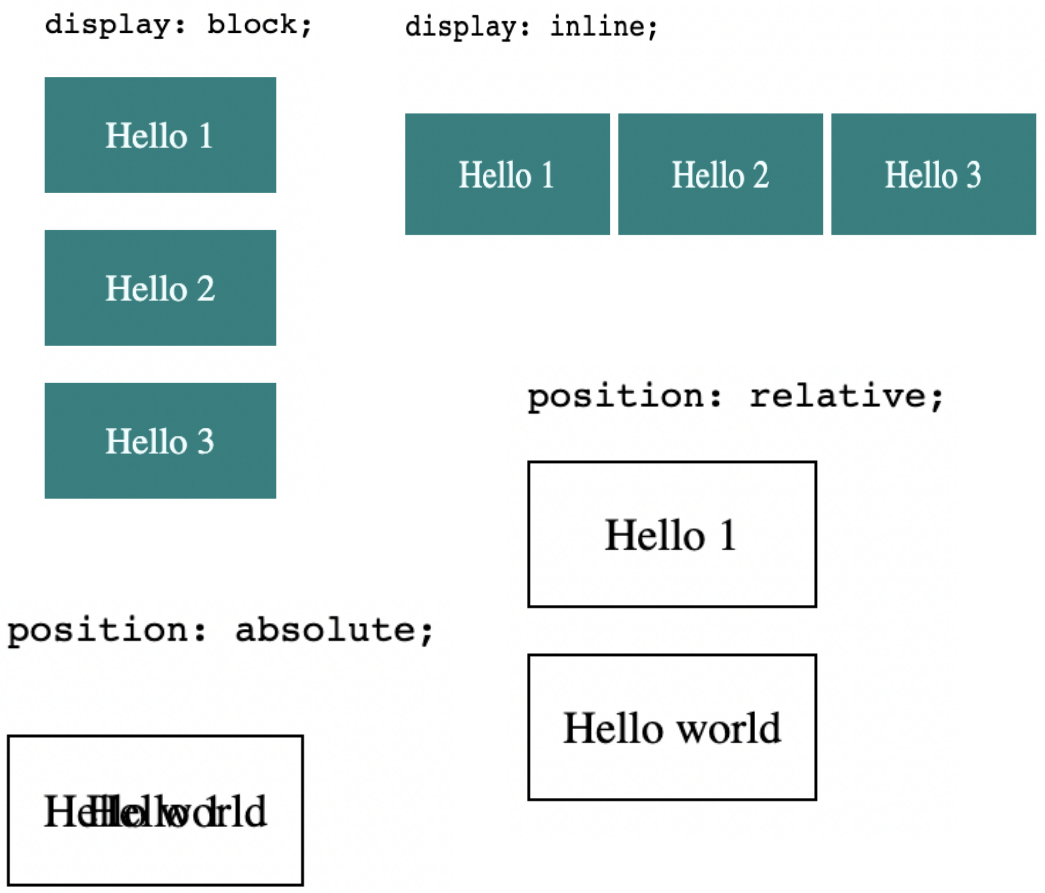
Displays, Position

- **Display**: specifies the display behavior of an element. You can hide or show the specific element.

Example: display: block;, display: none; , display: inline;

- **Position**: specifies the positioning behavior of an element.

Example: position: relative; , position: absolute; , position: sticky;



Attribute Selectors

To style elements having some specific attribute or attributes value.

Example:

input**[type="text"]** *// changes all input element having type="text"*
div **[title~="hello"]** // selects elements *having "hello"* ***word*** *present in their attribute's value*
div **[title^="hello"]** // selects elements having attribute's value **starting** with "*hello*"
div **[title$="hello"]** // selects elements having attribute's value **ending** with "*hello*"
div **[title*="hello"]** // selects elements having "*hello*" as **substring** in its value

```
<h3 class="top header">Hello</h3>
<h3 class="top-top">world</h3>
<h3 class="content">CSS</h3>
```

CSS [attribute^="value"] Selector

Hello

world

CSS

CSS [attribute~="value"] Selector

Hello

world

CSS

CSS [attribute~="value"] Selector

Hello

world

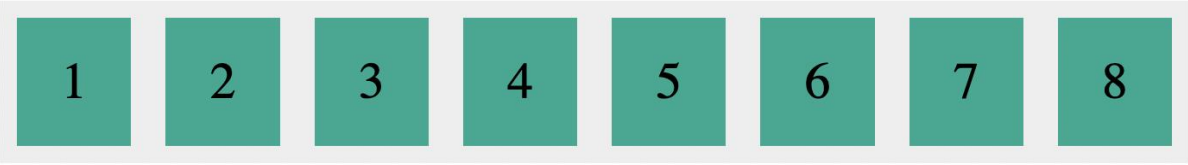
CSS

Flex Box

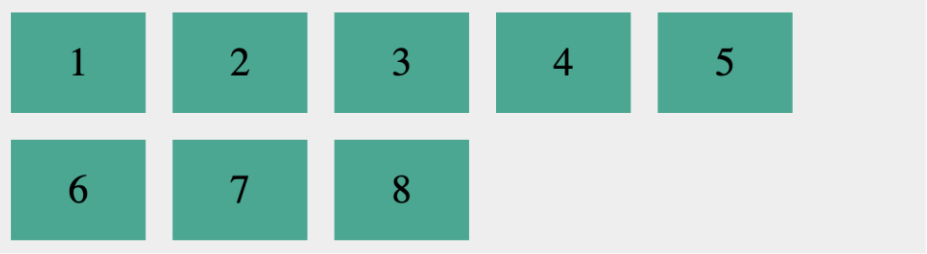
- Easy way to build flexible responsive layout structure.

- Properties:
  - ***flex-direction***: direction to stack the items.
  - ***flex-wrap***: whether to wrap items.
  - ***justify-content***: align items horizontally if flex-direction is 'row'.
  - ***align-items***: align items vertically if flex-direction is 'row'.

Wider Screen



Smaller Screen



Animations

- It allows elements to gradually change from one set of styles to another. It is a combination of multiple animation properties.

- *Properties:*
  - ***@keyframes**: define set of styles to change for an animation.*
  - ***animation-duration**: time animation takes to complete its cycle.*
  - ***animation-delay**: delay for the animation to start.*
  - ***animation-iteration-count**: number of times an animation must run.*
  - ***animation-direction**: whether animation to be displayed forward, backward or alternate cycles.*
  - ***animation-timing-function**: specifies speed curve of animation.*
- *Example:*

```
animation: example 5s linear 2s infinite alternate;
@keyframes example {
from { background-color: red;  },
to { background-color: yellow;  }
}
```

Miscellaneous

- **!important**
  - It is used to add *more importance* to a property/value than normal.
  - It will **override ALL** previous styling rules for that specific property on that element.

  Example: width: calc(100% - 100px);

- **calc() function**
  - The calc() function performs a calculation to be used as the property value.

  Example: *width: calc(100% - 100px);*

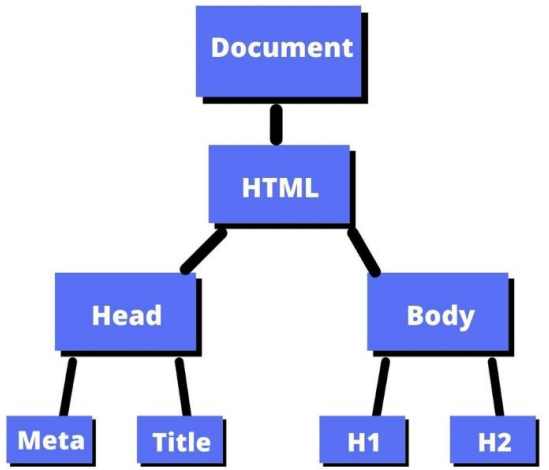
HTML DOM

- **DOM** stands for Document Object Model.
- **W3C** (*World Wide Web Consortium*) standard.
- A Programming Interface that allows us to
  - create,
  - change, or
  - remove elements from the document, and
  - even associate events to these elements.
- The DOM views an HTML document as a tree of nodes. A node represents an HTML element.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>DOM tree structure</title>
  </head>

  <body>
    <h1>DOM tree structure</h1>
    <h2>Learn about the DOM</h2>
  </body>

</html>
```



HTML DOM Methods

- HTML DOM methods are **actions** you can perform on HTML Elements.
- HTML DOM properties are **values** of HTML Elements that you can set or change.

- **getElementById(id)**
  - Returns the element that has the ID attribute with the specified value.
  - Returns **null** if no matching element found.
  - Syntax – ***document.getElementById(elementID)***
  - Example:
    - var elem = document.getElementById("myList");
    - This will access the element having value of its ID attribute as "myList".

HTML DOM Methods (Cont.)

- **getElementsByTagName(tag_name)**
  - Returns the collection of all the elements in the document with the given tag name.
  - Syntax – ***document.getElementsByTagName(tagname)***
  - Example:
    - var itemsLength = document.getElementsByTagName("li").length;
    - This will find out the number of list items present in the document.

- **querySelectorAll(pattern)**
  - Returns a **NodeList** object, representing all elements that matches the specified CSS selector(s).
  - Syntax - **document.querySelectorAll(CSS selectors)**
  - Example:
    - var para = document.querySelectorAll("p.example");
    - This will get all <p> elements having class as "example".

HTML DOM Methods

- **getElementsByClassName(class_name)**
  - Returns an HTMLCollection object, representing a collection of elements with the specified class name.
  - To search for multiple class names, separate them with spaces, like "class1 class2"
  - Syntax – ***document.getElementsByClassName(classname1 classname1 ...)***
  - Example:
    - To change the font color of all the element with class "example" to blue:

```
var elements = document.getElementsByClassName("example");
    for (let i = 0; i < elements .length; i++) {
        elements [i].color = "blue";
    }
```

- **createElement(name)**
  - Creates an Element Node with the specified name.
  - Syntax – ***document.createElement(name)***
  - Example:
    - var newElem = document.createElement("p");
      newElem.innerHTML = "Hello World...";

HTML DOM Methods

- **appendChild(node)**
  - Adds a node to the end of the list of children of a specified parent node.
  - Syntax – ***element.appendChild(node)***
  - Example:
    - Add a new list item to the existing list:

```
var item = document.createElement("li");
item.textContent = "item2";
document.getElementById("list").appendChild(item);
```

- **replaceChild(node, node)**
  - Replaces a child node with a new node.
  - Syntax – ***element.replaceChild(newElem, oldElem)***
  - Example:
    - Replace first item in a list with another item.

```
var list = document.getElementById('list');
var newItem = document.createElement('li');
newItem.textContent = 'New item';
list.replaceChild(newItem, menu.firstElementChild);
```

HTML DOM Methods

- **removeChild(node)**
  - Removes a specified child node of the specified element.
  - Syntax – ***element.removeChild(node)***
  - Example:
    - Remove all the items in a list:

```
var list = document.getElementById('list');
while (list.firstChild) {
        list.removeChild(list.firstChild);
}
```

Example

Adding a new child <p> element to a element on click of a button.

```
<html>
   <head>
      <script>
         function addNewParaElement(){
            var mainDiv= document.getElementById("main");
            var newPara = document.createElement("p");
            newPara.innerHTML = "New Child";
            mainDiv.appendChild(newPara );
         }
      </script>
   </head>
   <body>
         <div id="main">
         </div>
         <button onclick="addNewParaElement()"> Add Child</button>
   </body>
</html>
```