HTML , CSS, Javascript

-Sham Gaikwad

Goals

- Introduction to web technologies:
  - HTML to create the document structure and content
  - CSS to control is visual aspect
  - Javascript for interactivity

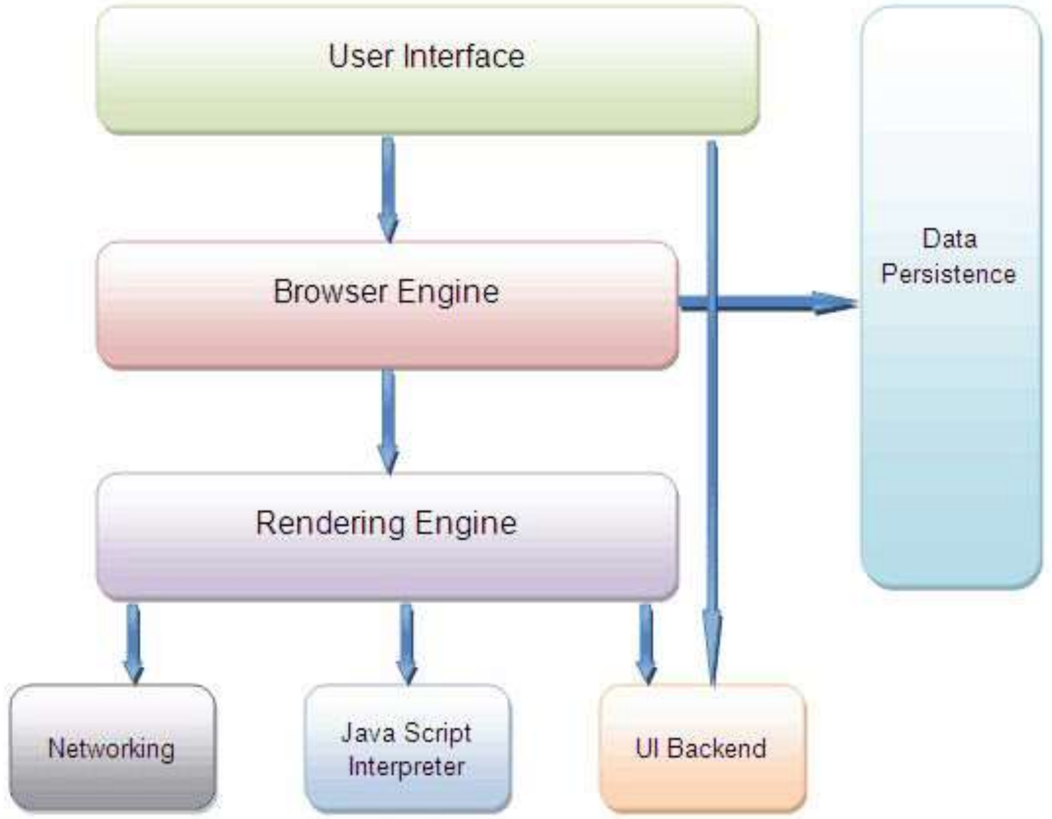
Test code

- How can I write test my code
- Just open the index.html from the template in your text editor and in your browser.
- Add simple html text and directly open in browser.
- When you do any change to the code, check it in the browser by pressing
  - F5 (refresh site)
- To open the developer tools press:
  - Windows: Control + Shift + I or F12

Inside a browser

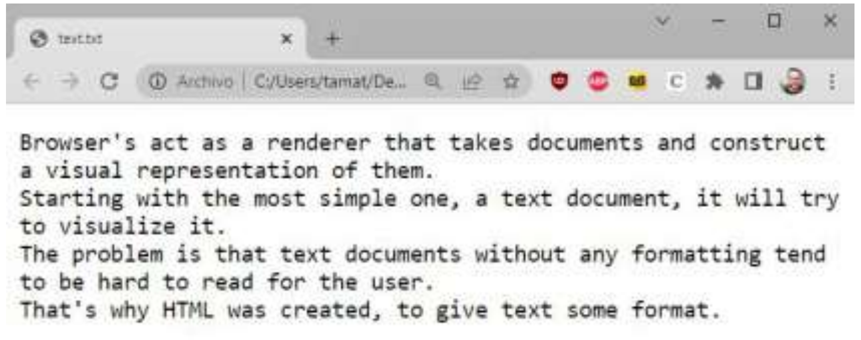
Browsers have very differentiate parts.
We are interested in two of them:

- the Rendering Engine (in charge of transforming our HTML+CSS in a visual image).
- The Javascript Interpreter (also known as VM), in charge of executing the Javascript code.



Browsers as a renderer

- Browser's act as a renderer that takes documents and construct a visual representation of them.
- Starting with the most simple one, a text document, it will try to visualize it.
- The problem is that text documents without any formatting tend to be hard to read for the user.
- That's why HTML was created, to give text some format.



```
Browser's act as a renderer that takes documents and construct
a visual representation of them.
Starting with the most simple one, a text document, it will try
to visualize it.
The problem is that text documents without any formatting tend
to be hard to read for the user.
That's why HTML was created, to give text some format.
```

Markup language

- There are many markup languages that add special tags into the text that the renderer wont show but use to know how to display the text.

e.g. ,

My name is <b>Sham Gaikwad</b>

here <b> and </b> tags are not shown by renderer but used to make the text bold.

HTML

- HTML means Hyper Text Markup Language.
- The HTML allow us to define the structure of a document or a
website.
- HTML is **NOT** a programming language, it’s a markup language,
which means its purpose is to give structure to the content of the
website, not to define an algorithm.
- It is a series of nested tags (it is a subset of XML) that contain all
the website information (like texts, images and videos). Here is an
- example of tags:

<title>This is a title</title>

- The HTML defines the page structure. A website can have several
HTMLs to different pages.

```
<html>
   <head>
   </head>
   <body>
      <div>
       <p>Hi</p>
      </div>
   </body>
</html>
```

HTML: basic rules

Some rules about HTML:

● It uses XML syntax (tags with attributes, can contain other tags).

```
<tag_name attribute="value"> content </tag_name>
```

● It stores all the information that must be shown to the user.

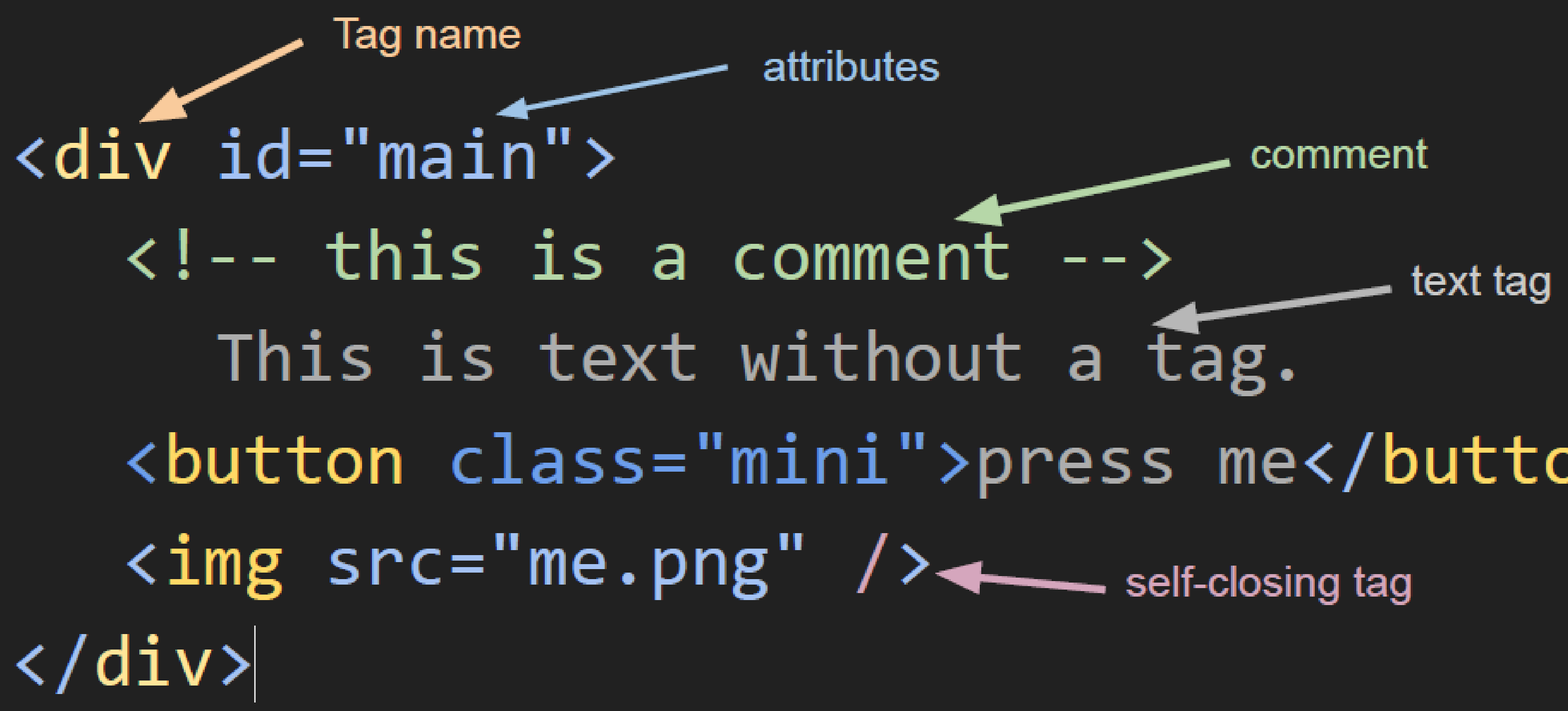
● There are different HTML elements for different types of information and behaviour.

● The information is stored in a tree-like structure (nodes that contain nodes inside) called DOM (Document Object Model).

● It gives the document some semantic structure (this is a title, this is a section, this is a form) which is helpful for computers to understand websites content.

● It must not contain information related to how it should be displayed (that information belongs to the CSS), so no color information, font size, position, etc.

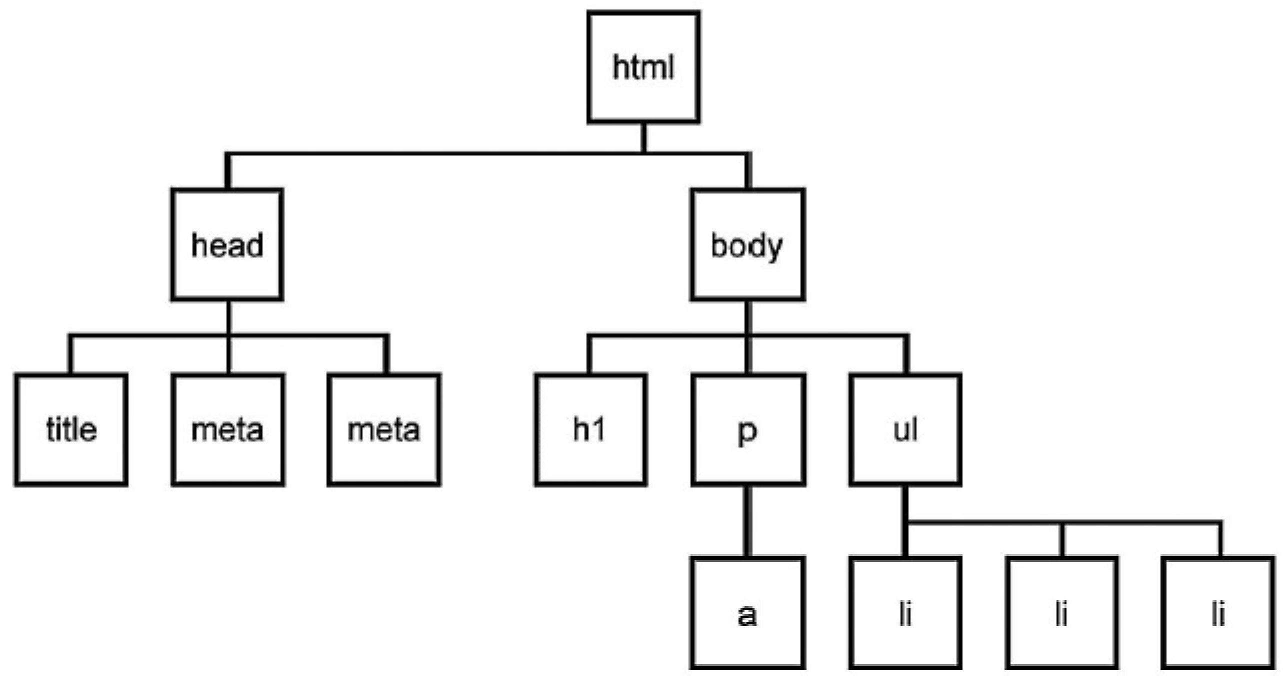
HTML: syntax example



```html
<div id="main">
   <!-- this is a comment -->
     This is text without a tag.
   <button class="mini">press me</button>
   <img src="me.png" />
</div>
```

DOM is a tree

Every node can only have one parent, and every node can have several children, so the structure looks like a tree.



HTML: main tags

Although there are lots of tags in the HTML specification, 99% of the webs use a subset of HTML tags with less that 10 tags, the most important are:

- <div>: a container, usually represents a rectangular area with information inside.
- <img/>: an image
- <a>: a clickable link to go to another URL
- <p>: a text paragraph
- <h1>: a title (h2,h3,h4 are titles of less importance)
- <input>: a widget to let the user introduce information
- <style>: to insert CSS rules
- <script>: to execute Javascript
- <span>: a null tag (doesn't do anything)

HTML: other interesting tags

There are some tags that could be useful sometimes:

- <button>: to create a button
- <audio>: for playing audio
- <video>: to play video
- <canvas>: to draw graphics from javascript
- <iframe>: to put another website inside ours

HTML: tagging correctly

Try to avoid doing this:

```
<div>
Title

Here is some content
Here is more content
</div>
```

DONT DO THIS

Do this instead

```
<div>
    <h1>Title</h1>
    <p>Here is content.</p>
    <p>Here is more content</p>
</div>
```

HTML good use

It is good to have all the information properly wrapped in tags that give it some semantics.

We also can extend the code semantics by adding extra attributes to the tags:

- id: tells a unique identifier for this tag
- class: tells a generic identifier for this tag

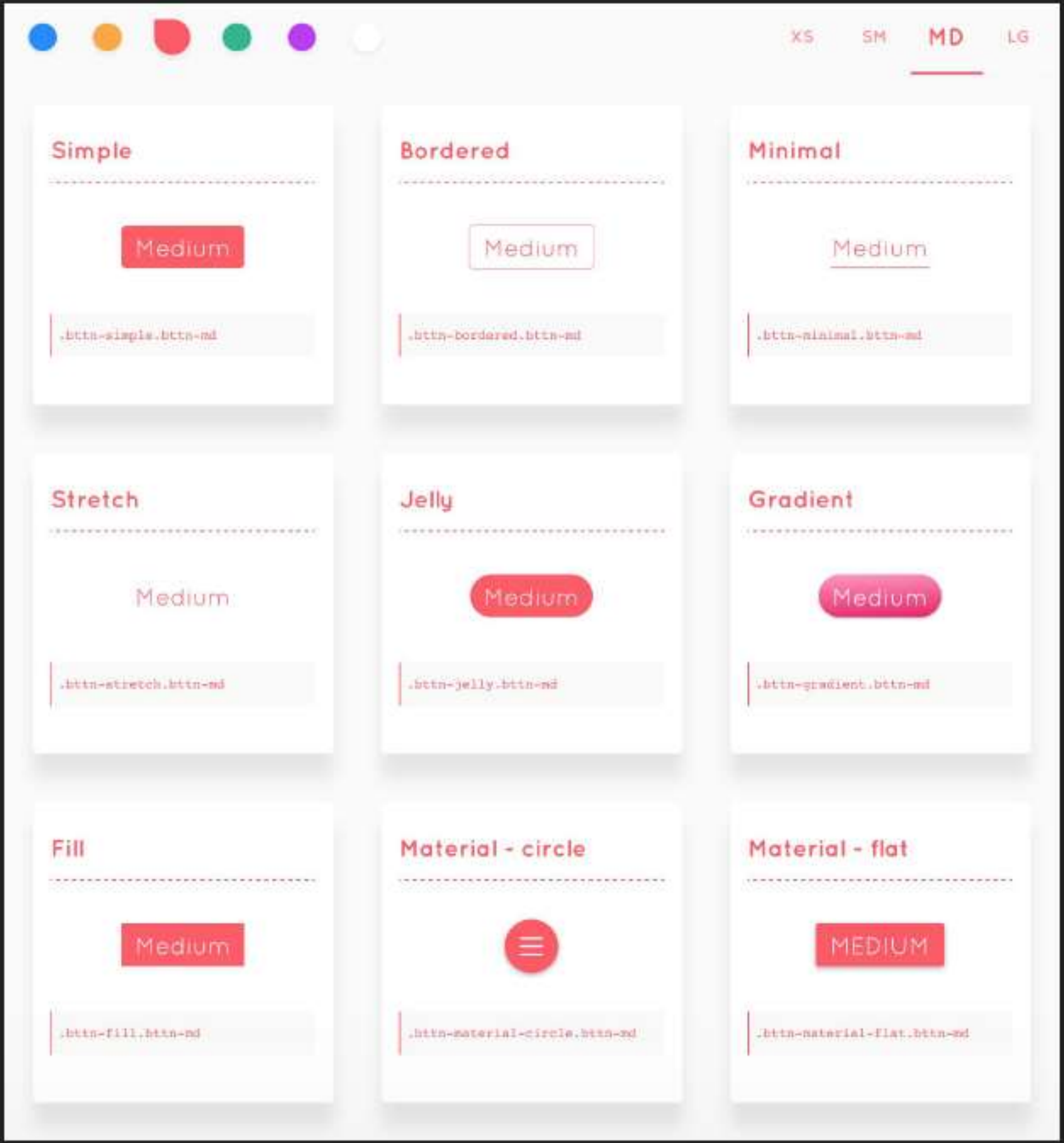
```
<div id="profile-picture" class="mini-image">...</div>
```

CSS

CSS allows us to specify how to present (render) the document info stored in the HTML.

Thanks to CSS we can control all the aspects of the visualization and some other features:

- Colors: content, background, borders
- Margins: interior margin, exterior margin
- Position: where to put it
- Sizes: width, height
- Behaviour: changes on mouse over



```
.bttn-simple.bttn-md
.bttn-bordered.bttn-md
.bttn-minimal.bttn-md
.bttn-stretch.bttn-md
.bttn-jelly.bttn-md
.bttn-gradient.bttn-md
.bttn-fill.bttn-md
.bttn-material-circle.bttn-md
.bttn-material-flat.bttn-md
```

CSS example

```
* {
    color: blue; /*a comment */
    margin: 10px;
    font: 14px Tahoma;
}
```

This will change all the tags in my web (‘*‘ means all) to look blue with font Tahoma with 14px, and leaving a margin of 10px around.

CSS fields

Here is a list of the most common CSS fields and an example:

- `color: #FF0000;    red;     rgba(255,00,100,1.0); //different ways to specify colors`
- `background-color: red;`
- `background-image: url('file.png');`
- `font: 18px 'Tahoma';`
- `border: 2px solid black;`
- `border-top: 2px solid red;`
- `border-radius: 2px; //to remove corners and make them more round`
- `margin: 10px; //distance from the border to the outer elements`
- `padding: 2px; //distance from the border to the inner elements`
- `width: 100%;    300px;    1.3em;  //many different ways to specify distances`
- `height: 200px;`
- `text-align: center;`
- `box-shadow: 3px 3px 5px black;`
- `cursor: pointer;`
- `display: inline-block;`
- `overflow: hidden;`

CSS how to add it

There are four ways to add CSS rules to your website:

- Inserting the code inside a style tag
```
<style>
    p { color: blue }
</style>
```
- Referencing an external CSS file
```
<link href="style.css" rel="stylesheet" />
```
- Using the attribute style on a tag
```
<p style="color: blue; margin: 10px">
```
- Using Javascript (we will see this one later).

CSS selectors

Let's start by changing the background color of one tag of our website:

```
div {
    background-color: red;
}
```

This CSS rule means that every tag DIV found in our website should have a red background color. Remember that DIVs are used mostly to represent areas of our website.

We could also change the whole website background by affecting the tag body:

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

CSS selectors

What if we want to change one specific tag (not all the tags of the same type).

We can specify more precise selectors besides the name of the tag. For instance, by class or id. To specify a tag with a given class name, we use the dot:

```
p.intro {
    color: red;
}
```

This will affect only the tags p with class name intro:

```
<p class="intro">
```

CSS Selectors

There are several selectors we can use to narrow our rules to very specific tags of our website.

The main selectors are:

- tag name: just the name of the tag
  - `p { ... }  //affects to all <p> tags`
- dot (.): affects to tags with that class
  - `p.highlight { ... } //affects all <p> tags with class="highlight"`
- sharp character (#): specifies tags with that id
  - `p#intro { ... } //affects to the <p> tag with the id="intro"`
- two dots (:): behaviour states (mouse on top)
  - `p:hover { ... } //affects to <p> tags with the mouse over`
- brackets ([attr='value']): tags with the attribute attr with the value 'value'
  - `input[type="text"] {...} // affects to the input tags of the type text`

CSS Selectors

You can also specify tags by its context, for example: tags that are inside of tags matching a selector. Just separate the selectors by an space:

```
div#main p.intro { ... }
```

This will affect to the p tags of class intro that are inside the tag div of id main

```
<div id="main">
    <p class="intro">....</p>  ← Affects this one
</div>

<p class="intro">....</p>  ← but not this one
```

CSS Selectors

And you can combine selectors to narrow it down more.

```
div#main.intro:hover { ... }
```

will apply the CSS to the any tag div with id main and class intro if the mouse is over.

And you do not need to specify a tag, you can use the class or id selectors without tag, this means it will affect to any node of id main

```
#main { ... }
```

CSS Selectors

If you want to select only elements that are direct child of one element (not that have an ancestor with that rule), use the `>` character:

```
ul.menu > li { ... }
```

Finally, if you want to use the same CSS actions to several selectors, you can use the comma `,` character:

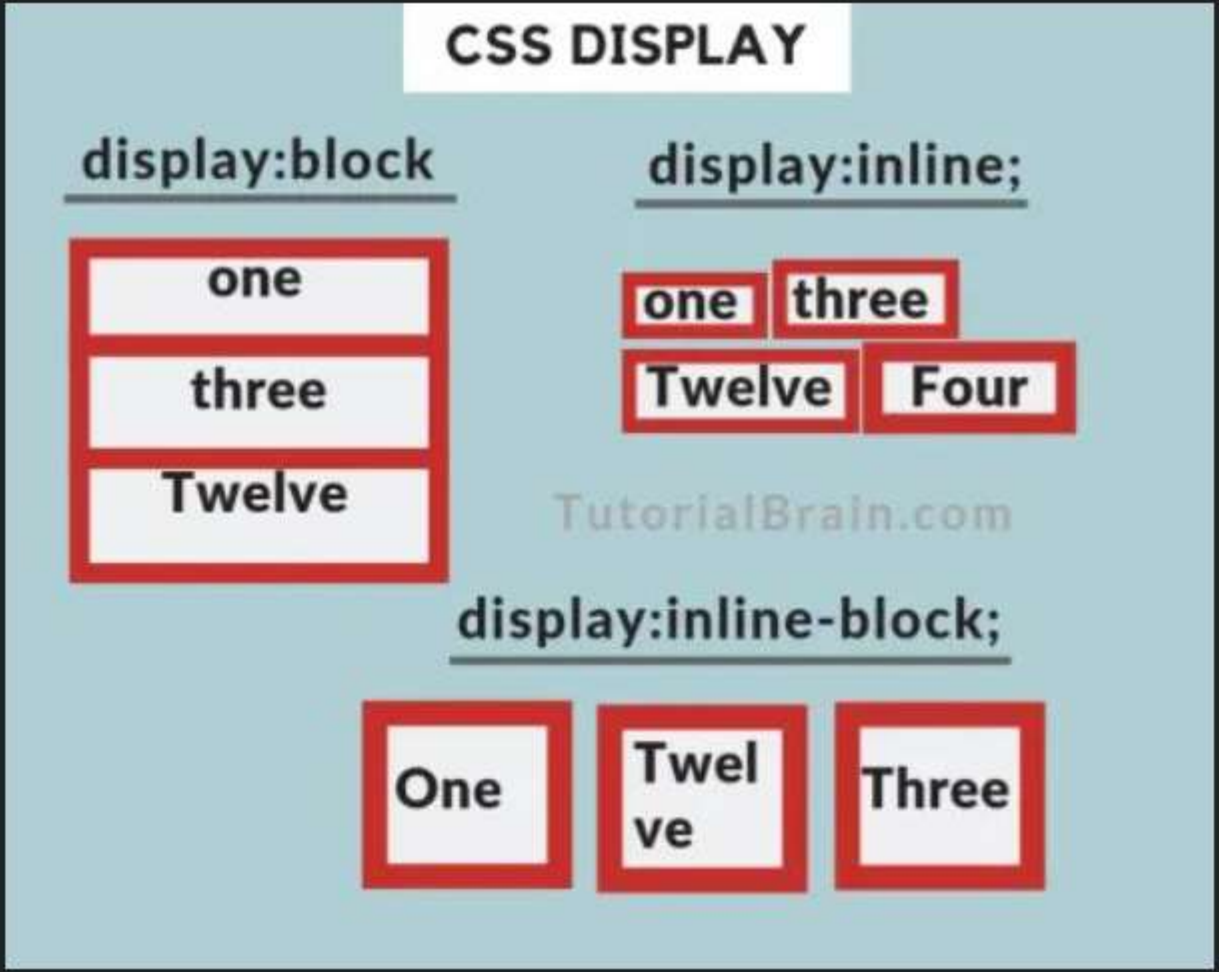
```
div, p { … }  ← this will apply to all divs and p tags
```

HTML arrange

It is important to understand how the browser arranges the elements on the screen.

You can change the way elements are arranged using the display property:

```
div { display: inline-block; }
```



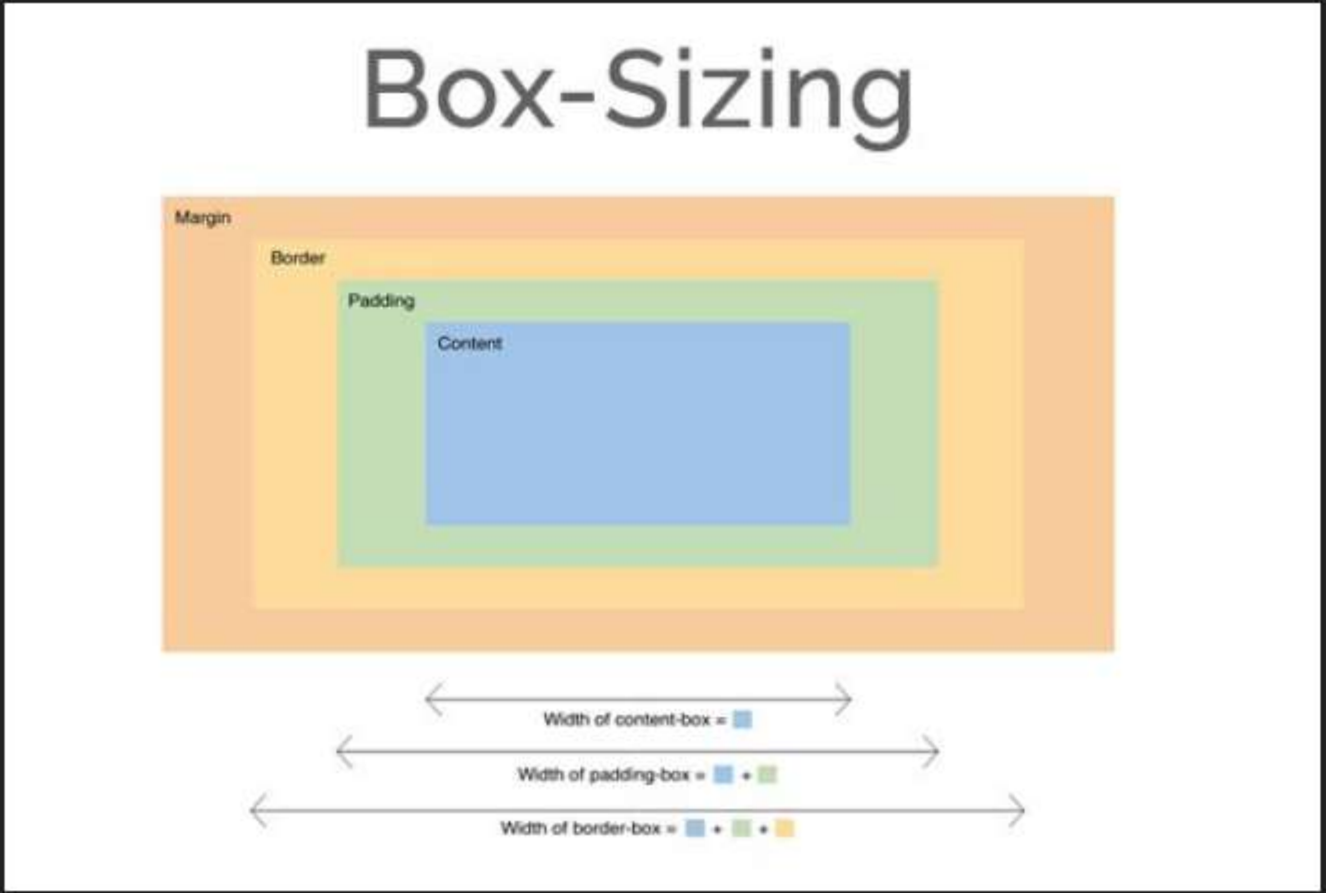
Box Model

It is important to note that by default any width and height specified to an element will not take into account its margin, so a div with width 100px and margin 10px will measure 120px on the screen, not 100px.

This could be a problem breaking your layout.

You can change this behaviour changing the box model of the element so the width uses the outmost border:

```
div { box-sizing: border; }
```

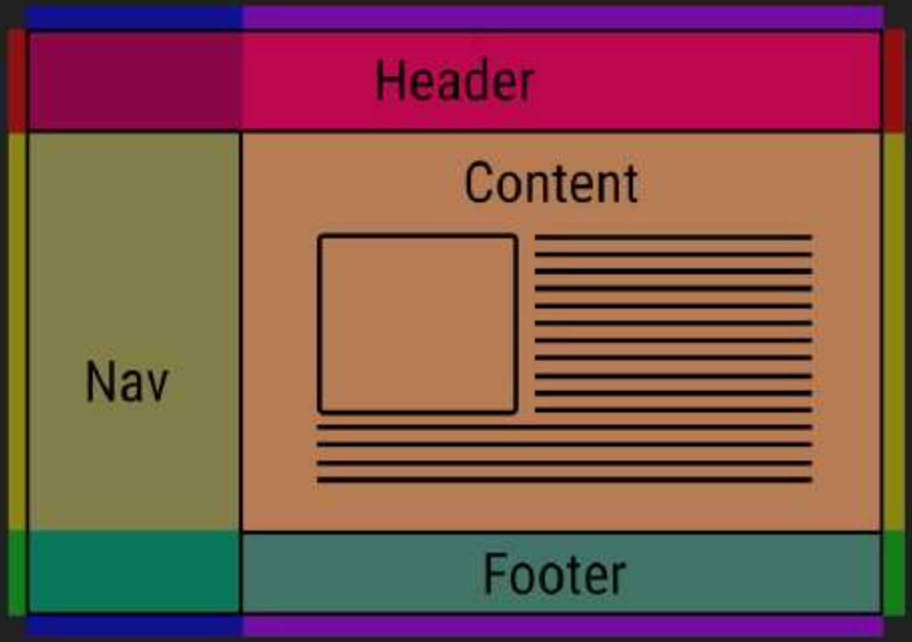


Layout

One of the hardest parts of CSS is construing the layout of your website (the structure inside the window) .

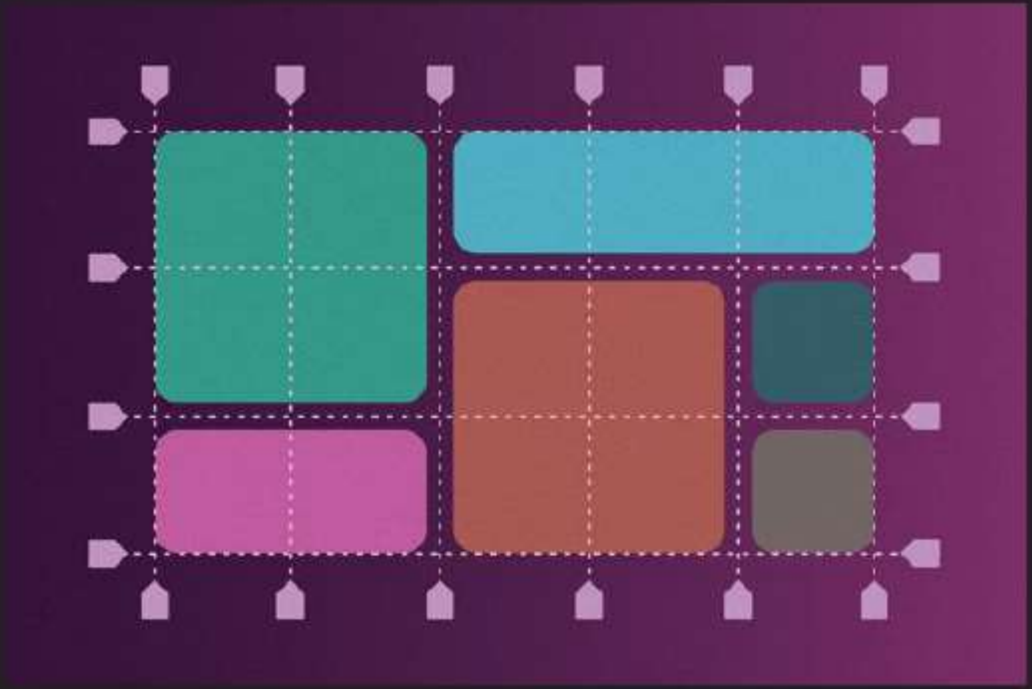
By default HTML tends to put everything in one column, which is not ideal.

There has been many proposals in CSS to address this issue (tables, fixed divs, flex, grid, …).



Grid system

Because most sites are structured in a grid, I recommend to use the CSS Grid system.



HTML

```html
<div class="grid-container">
  <div class="grid-item1">1</div>
  <div class="grid-item2">2</div>
</div>
```

CSS

```css
.grid-container {
  display: grid;
  grid-template-rows: 100px; 100px;
  grid-template-columns: 100px; 100px; 100px;
  grid-gap: 5px;
}

.grid-item1 {
  background: blue;
  border: black 5px solid;
  grid-column-start: 1;
  grid-column-end: 5;
  grid-row-start: 1;
  grid-row-end: 3;
}
```

Fullscreen divs

Sometimes we want to have a div that covers the whole screen (to make a webapp), instead of a scrolling website (more like regular documents).

In that case remember to use percentages to define the size of elements, but keep in mind that percentages are relative to the element's parent size, so you must set the size to the <body> element to use 100%.

CSS

```
html, body {
      width: 100%;
      height: 100%;
}

div {
      margin: 0;
      padding: 0;
}

#main {
      width: 100%;
      height: 100%;
}
```

Trick to center

Centering divs can be hard sometimes, use this trick:

```
.horizontal-and-vertical-centering {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Javascript

A regular programming language, easy to start, hard to master.

Allows to give some interactivity to the elements on the web.

Syntax similar to C or Java but with no types.

You can change the content of the HTML or the CSS applied to an element.

You can even send or retrieve information from the internet to update the content of the web without reloading the page.

History

- **1992**

  Oak, Gosling at Sun & FirstPerson
- **1995**

  HotJava

  LiveScript, Eich at Netscape
- **1996**

  JScript at Microsoft
- **1998**

  ECMAScript

Javascript: insert code

There is three ways to execute javascript code in a website:

- Embed the code in the HTML using the <script> tag.

```
<script> /* some code */ </script>
```

- Import a Javascript file using the <script> tag:

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

- Inject the code on an event inside a tag:

```
<button onclick="javascript: /*code*/">press me</button>
```

JavaScript Syntax - Variables and Literals

- Declaration
  - Explicit: var i = 12; // no ‘var’ in declaration
  - Implicit: i = 12;
- Variable Scope
  - Global
    - Declared outside functions
    - Any variable implicitly defined
  - Local
    - *Explicit* declarations inside functions

Dynamic Typing - Variables can hold any valid type of value:

– Number … var myInt = 7;

– Boolean … var myBool = true;

– Array … var myArr = new Array();

– String … var myString = “abc”;

  – … and can hold values of different types at different times during execution

Javascript: Syntax

Very similar to C++ or Java but much simpler.

```
var my_number = 10; //this is a comment
var my_string = "hello";
var my_array = [10,20,"name",true];
var my_object = { name: "javi", city: "Barcelona" };

function say( str )
{
    for(var i = 0; i < 10; ++i)
        console.log(" say: " + str );
}
```

Math

- Math object is modeled on Java's Math class.
- It contains

`abs`	absolute value
`floor`	integer
`log`	logarithm
`max`	maximum
`pow`	raise to a power
`random`	random number
`round`	nearest integer
`sin`	sine
`sqrt`	square root

String Methods

- `charAt`
- `concat`
- `indexOf`
- `lastIndexOf`
- `match`
- `replace`
- `search`
- `slice`
- `split`
- `substring`
- `toLowerCase`
- `toUpperCase`

Key Comparison Operators

- == the numbers or objects or values must be equal
- ! Logical NOT
- || Logical OR
- != the numbers or objects or values must not be equal
- >= number on the right must be less than or equal to the number on the left
- && Logical AND

  number on the right must be greater than or equal to the number on the left
- <=
- < number on the right must be greater than the number on the left
- > number on the left must be greater than the number on the right

JavaScript Output

- The document objects allows printing directly into the browser page (amongst other things)
- • window object is implied
- • Writing in text or HTML with script
- – No line-break
- document.write(“I am <B>BOLD</B>”);
- – With line-break
- document.writeln(“I am <U>underlined</U>”);