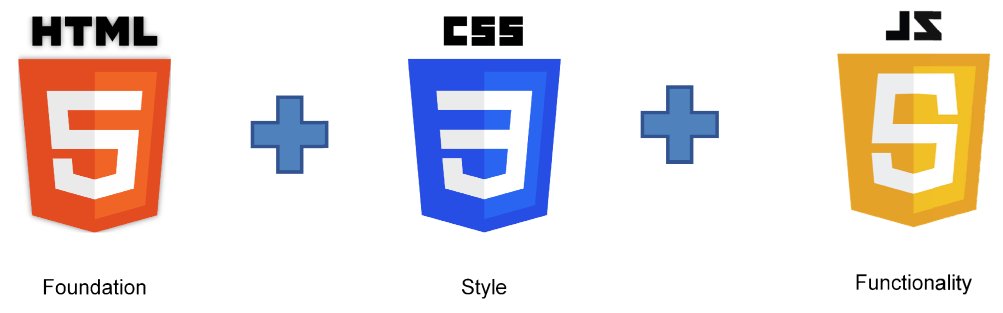
Major ingredients of the website



Any website is majorly built using

1. HTML
2. CSS
3. Java script

Website and pages can be divided into 2 types majorly:

Static Website	Dynamic Website
The content present on the web page cannot be modified at runtime.	The content present on the web page can be modified at runtime.
There is no interaction with the database.	The website can interact with the database.
It is cheaper to develop a static website as compared to a dynamic website.	A dynamic website requires more cost of development than a static website.
The same content gets loaded every time.	The content that gets loaded may vary as

	per requirements.

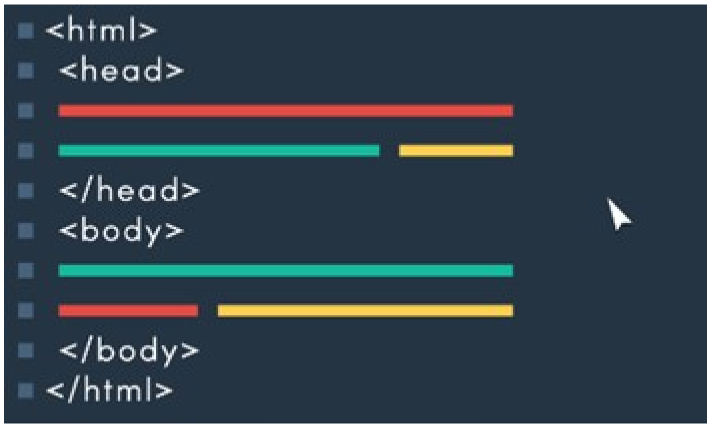
HTML:

- **HTML:** Hyper Text Mark up Language
    - **Hypertext:** Text which contains hyperlinks (or references) to other text.
    - **Mark up Language:** It is a way in which the documents are annotated such that it can be systematically distinguished from the normal text.
- **HTML** is a set of **mark up** symbols intended for displaying content on the Internet.
- **Mark up** tells browsers how to display the content of a webpage, that is, the words and images. Mark ups are commonly known as **tags**.
- The HTML file is broken into small parsing elements called Tokens, which are read by the browser.

<u>Structure of HTML</u>

HTML has majorly below parts:

- HTML (Container): A container that holds the content and the information about the content
- Head: Information about the content
- Body: Content

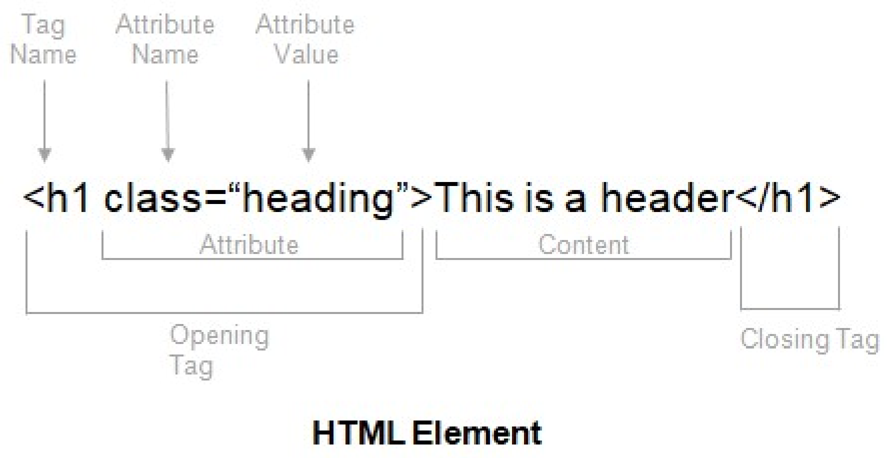


```
<html>
 <head>


 </head>
 <body>


 </body>
</html>
```

Commonly used HTML Tags:

<h1></h1>...<h6></h6>
<p></p>
<a></a>
<span></span>
<div></div>
<select></select>
<table></table>
<img />
<br />
<input />
<hr />

Tag Structure:



Special Characters and Entities:

An HTML entity is basically a text that starts with an ampersand (&) and ends with a semicolon (;).

These entities are often used to display the reserved characters. I.e. the characters which otherwise would be interpreted as simple HTML: code.

Some commonly used characters and their entities are:

Character	Entity	Description
&	&amp;	It symbolizes the beginning of an character entity ‘and’.
<	&lt;	It is used to represent the opening tag or less than sign.
>	&gt;	It is used to represent the closing tag or greater than sign.
“	&quot;	It is used to represent the start and end of attribute value.
space	&nbsp;	It is used to represent the space.

Inline vs Block Elements:

You will notice few tags have end tag and few do not.

Few will allow next contents to be plotted beside and few do not.

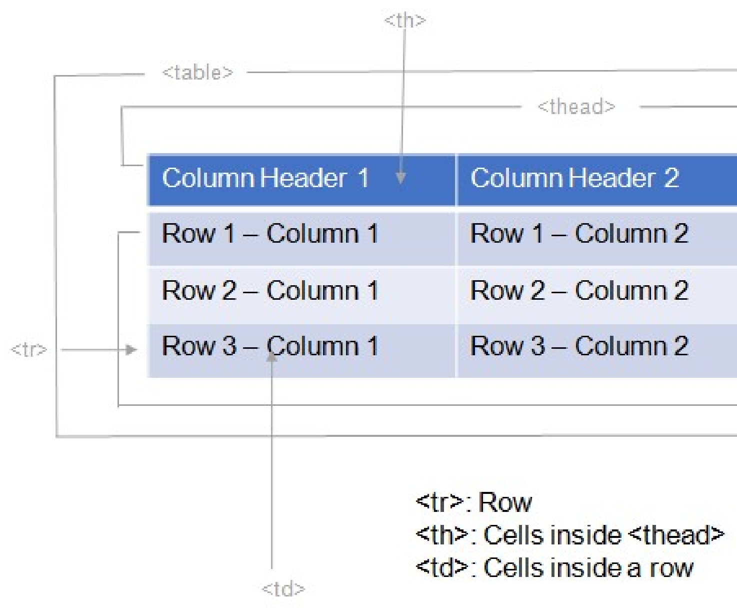
List of inline and block elements / tags:

Inline Elements	Block Elements
<span>	<div>
<b>	<p>
<strong>	<h1> …. <h6>
<i>	<hr>
<em>	<table>
<a>	<ul>
<img>	<ol>
<button> …	<form>
You can see the entire list here. https://developer.mozilla.org/en-US/docs/Web/HTML/Inline_elements	You can see the entire list here. https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

HTML Tables

Used for presenting data in tabular format:

Table Structure:



<tr>: Row
<th>: Cells inside <thead>
<td>: Cells inside a row

```
<body>
    <table align="right">
      <thead>
        <tr>
          <th>Fruit</th>
          <th>Quantity</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>Apples</td>
          <td>10</td>
```

```
            </tr>
             <tr>
               <td>Oranges</td>
               <td>15</td>
             </tr>
           </tbody>
        </table>
    </body>
```

Forms

Code Example:

```
<html>
  <head>
    <title>Log In</title>
  </head>
  <body>
    <form action="/action-page.php" method="get" target="_blank">

      <label for="name">Username:</label>
      <input type="text" name="name" id="name" />

      <label for="password">Password:</label>
      <input type="password" name="password" id="password" />

      <input type="checkbox" name="remember" id="remember" />

      <label for="remember">Keep me logged in.</label>

      <input type="submit" value="Login"/>
    </form>
  </body>
</html>
```

Form Tag Attributes and meaning:

action – It defines the action to be performed when the form is submitted.

method – It defines the HTTP method for submitting data.

target – It defines where the submitted results should open.

GET vs POST

GET	POST
Form data IS VISIBLE in the URL.	Form data is NOT VISIBLE in the URL.
Clicking the back button will not re-submit the data; hence, it is harmless.	Clicking the back button will re-submit the data; hence, you need to be cautious while using POST.
It can be bookmarked and cached.	It cannot be bookmarked and cached.
The length of the URL is restricted.	There are no restrictions on the length of the URL.
It is Not Secure.	It is Secure.

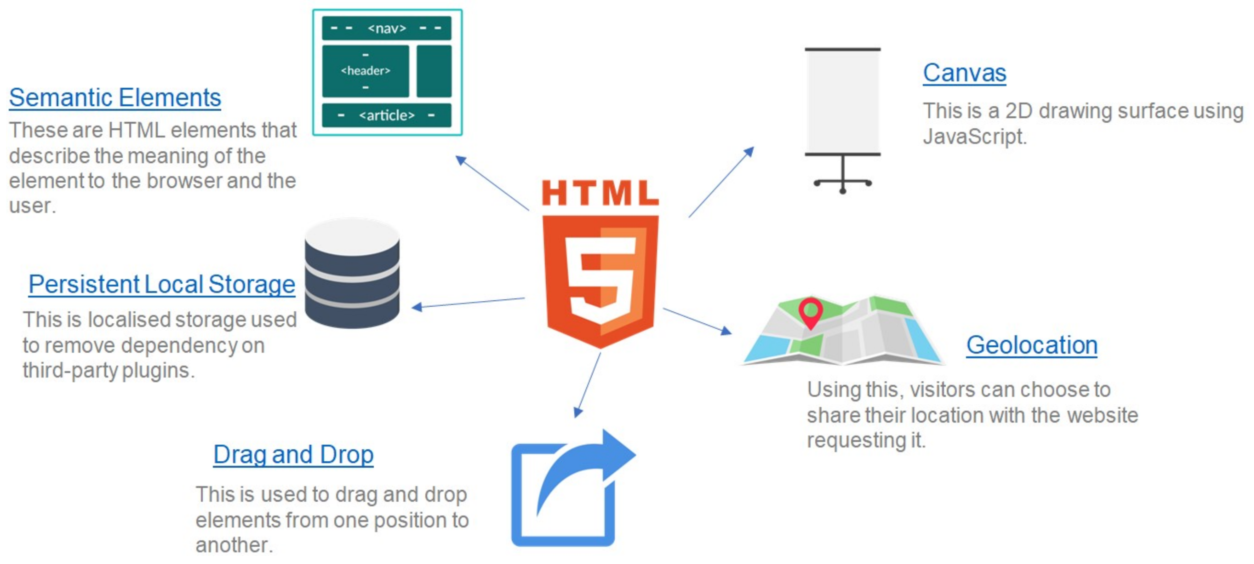
Different input types:

`<input type="text" />`	Allows users to type text.
`<input type="email" />`	Allows users to type text in email format, i.e. if the typed text does not contain @ or . as normal email formats do, it will throw up an error message on submitting.
`<input type="password" />`	Allows users to type text in password format. The text entered would look like this: ••••••
`<input type="number" />`	Allows users to type characters in number format. Users will not be able to type alphabets or any other special character if the input is defined as type number.
`<input type="radio" />`	Allows users to include a radio button selection, i.e., choose only one of multiple options provided.
`<input type="checkbox" />`	Allows users to include a checkbox selection, i.e., choose multiple options provided.
`<input type="file" />`	Allows users to upload files from their local machines.
`<input type="range" />`	Allows users to add a range slider between a minimum value and a maximum value.
`<input type="submit" />`	Allows users to submit a form.

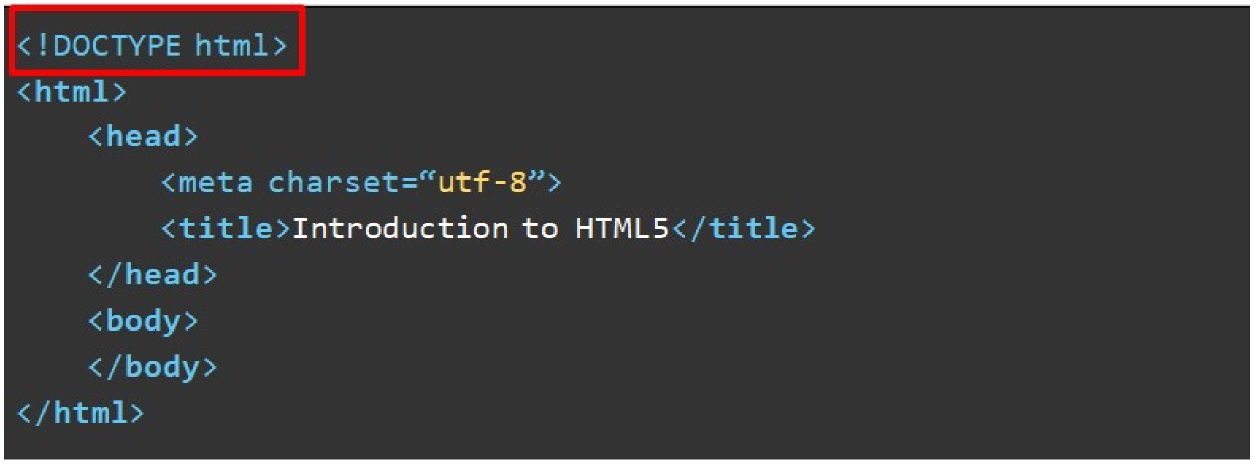
Common Input tag attributes:

`value`	It can be used to specify an initial value for an input field.
`size`	It can used to specify the visible width of the input field box.
`min`	It specifies the minimum value of the input field.
`max`	It specifies the maximum value of the input field.
`maxlength`	It specifies the maximum allowable number of characters.
`readonly`	It is used to ensure that the input field becomes read-only.
`disabled`	It is used to disable the field
`placeholder`	Text written inside the input field when it is empty
`required`	It is used to mark the input fields as compulsorily before submitting the form.

What is new in HTML 5?


Semantic Elements
These are HTML elements that describe the meaning of the element to the browser and the user.
<nav>
<header>
<article>
Canvas
This is a 2D drawing surface using JavaScript.
HTML
5
Persistent Local Storage
This is localised storage used to remove dependency on third-party plugins.
Geolocation
Using this, visitors can choose to share their location with the website requesting it.
Drag and Drop
This is used to drag and drop elements from one position to another.

How do we tell browser that content is HTML 5?



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Introduction to HTML5</title>
    </head>
    <body>
    </body>
</html>
```

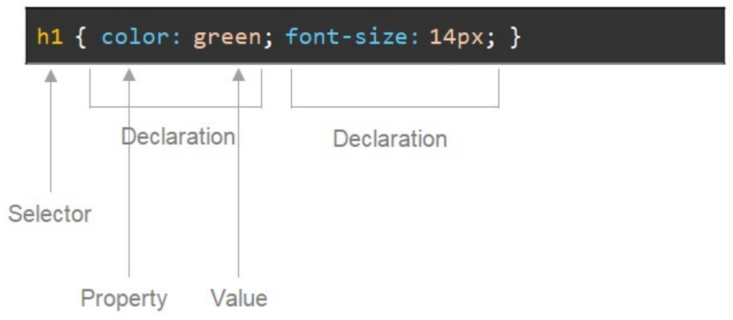
Simple HTML 5 Tags Example:

```
<!DOCTYPE html>

<html>
  <head>

    <title>Introduction to HTML5</title>
  </head>
  <body>
    <header role="banner">

        <h1>Introduction to HTML5</h1>

      </header>

      <article>

          <section>

            <p>One article can have multiple sections.</p>

          </section>

      </article>

      <footer>Created by Bonaventure Systems.</footer>
    </body>
  </html>
```

Basics of CSS

- CSS stands for **C**ascading **S**tyle **S**heets.
- It makes an HTML website presentable.
- It adds style to various HTML elements.
- It helps you to define how the elements should look, where they should be placed and whether they should be displayed or not.
- It also helps you to define how elements should look on different devices.

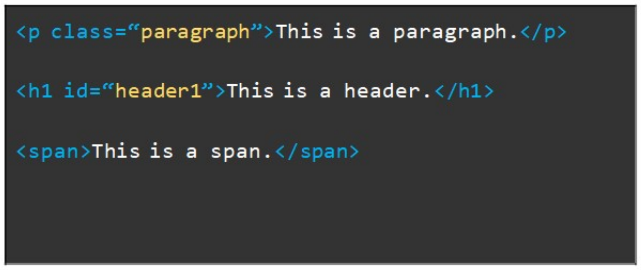
Syntax:



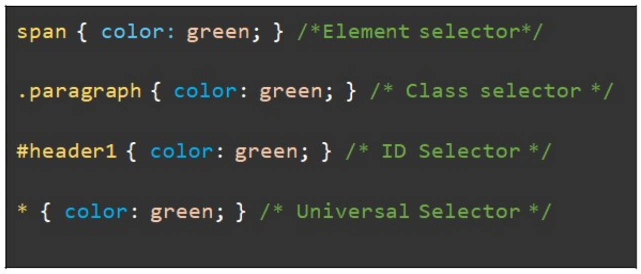
```
h1 { color: green; font-size: 14px; }
```

- A selector can be an element itself such as **p**, **h1** or the **class** or the **id**.
- A property signifies the property of an element, such as the **color** of a text element.
- A value signifies the value that should be assigned to a property, such as **red** for the **color** property.

Ways to apply Style:



```
<p class="paragraph">This is a paragraph.</p>

<h1 id="header1">This is a header.</h1>

<span>This is a span.</span>
```

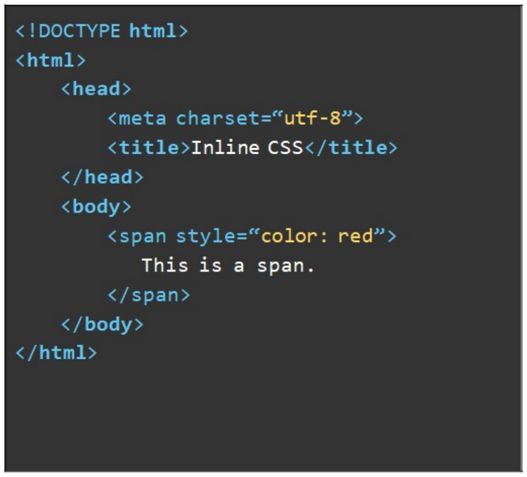


```
span { color: green; } /*Element selector*/

.paragraph { color: green; } /* Class selector */

#header1 { color: green; } /* ID Selector */

* { color: green; } /* Universal Selector */
```

- **Element selectors** only require the element name.
- **Class selectors** require '**.**' before the class name given to an element.
- **ID selectors** require '**#**' before the ID name given to an element.
- **Universal selector** require '*'. It signifies all the elements in a given file.

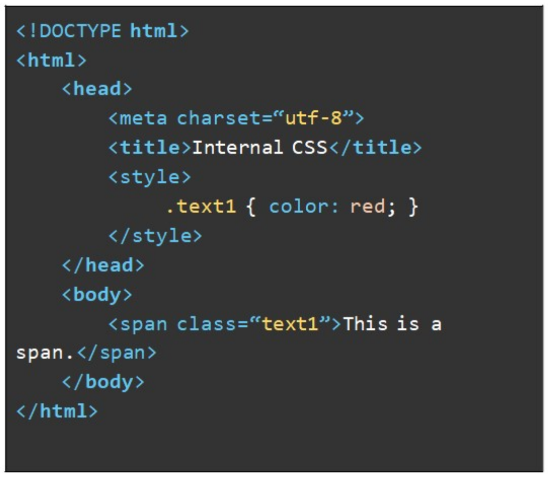
Inline vs Internal vs External:

- Inline stylesheet: This is a CSS that is directly applied to an element inside HTML code.
- Internal stylesheet: This is a CSS that is applied inside a file but not inside an element. It is generally added inside the <style> tag, which is contained within the <head> tag.
- External stylesheet: This is a CSS that is applied from an external file. This external file is referred to inside the HTML code using the <link> tag.
- The following is the order of priority: Inline > Internal > External.



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Inline CSS</title>
    </head>
    <body>
        <span style="color: red">
           This is a span.
        </span>
    </body>
</html>
```

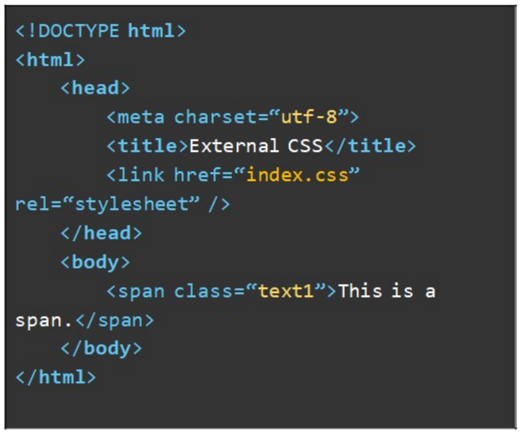
Inline CSS



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Internal CSS</title>
        <style>
             .text1 { color: red; }
        </style>
    </head>
    <body>
        <span class="text1">This is a
span.</span>
    </body>
</html>
```

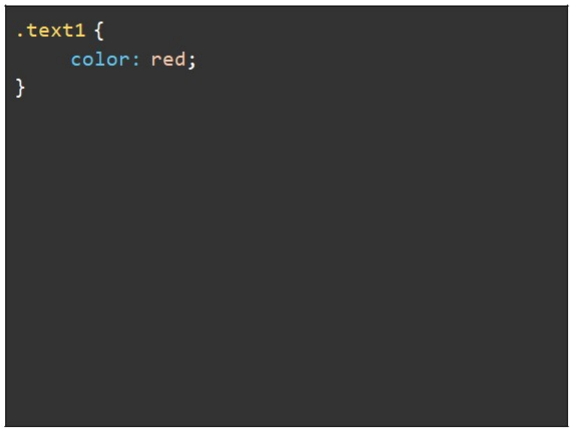
Internal CSS

External CSS:



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>External CSS</title>
        <link href="index.css"
rel="stylesheet" />
    </head>
    <body>
        <span class="text1">This is a
span.</span>
    </body>
</html>
```

index.html

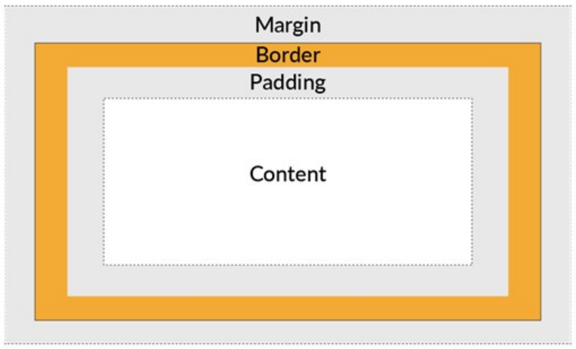


```
.text1 {
     color: red;
}
```

index.css

CSS Box Model:

The CSS box model is a box that surrounds every HTML element. It comprises the following: **content + padding + border + margin.**



- Therefore, the total width of element = Actual width + Left padding + Right padding + Left border + Right border + Left margin + Right margin.

- The total height of element = Actual height + Top padding + Bottom padding + Top border + Bottom border + Top margin + Bottom margin.

Basics of Java script

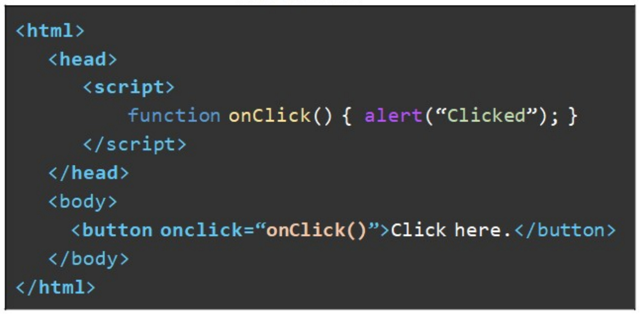
- JavaScript (JS) is the programming language of web development. It was initially used to ‘make web pages alive’.
- JS programs are called scripts.
- Unlike Java, scripts do not require any compilation to run. JS can execute on a browser, a server and any device that has a ‘JavaScript Engine’.
- JS does not provide low-level access to memory or CPU, which is why it is a ‘safe’ programming language.
- With the help of JS, you can perform the following:
- Adding or modifying HTML content on a page, including, but not restricted to, showing/hiding elements as well as changing the size, style and HTML attributes
- Reacting to user actions such as hovering, mouse clicks, key presses, etc.
- Sending requests to remote servers over the network and saving the data on the client side.

Where to put java script?

- JavaScript (JS) can be placed either internally in an HTML file using the <script> tag, where you can write all the scripts inside the <script> tag, or externally in a JavaScript file (that ends with a ‘.js’ extension), which can be added as a source in the <script> tag.
- The <script> tag can be placed either in the <head> or in the <body> part of the HTML document.
- However, because JS is a render blocker, it is preferable to call these scripts in the <body> tag after all the HTML elements are rendered.

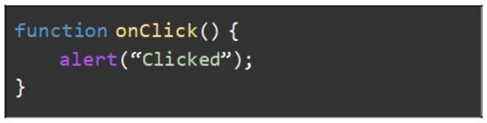
Internal vs External:

index.html

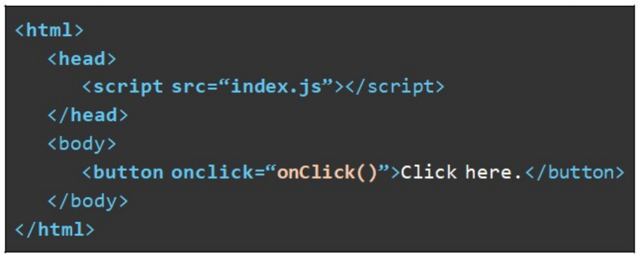


```
<html>
   <head>
      <script>
         function onClick() { alert(“Clicked”); }
      </script>
   </head>
   <body>
     <button onclick=“onClick()”>Click here.</button>
   </body>
</html>
```

index.js



```
function onClick() {
    alert(“Clicked”);
}
```



```
<html>
   <head>
      <script src=“index.js”></script>
   </head>
   <body>
      <button onclick=“onClick()”>Click here.</button>
   </body>
</html>
```

Syntax and Terms:

- JavaScript (JS) programs are a list of programming instructions called **statements** that are executed by web browsers.
- JS statements consist of values, operators, expressions, keywords and comments. They are executed in a waterfall fashion, that is, one by one, in the order in which they are written.
- Generally, a JavaScript (JS) code consists of statements with variables that store values or expressions.

```
var x, y; //Declaring variables

var num1 = 24; //Assigning values to variable

var num2 = 30; //Assigning values to variable

var total = num1 + num2; //Computing expressions and storing result in variable
```

- In a programming language, **variables** are used to store data values. In JS, variables are declared using the **var** keyword. An equal sign (‘=’) is used to assign value to the variables.
- JS **values** can be strings, numbers, arrays, objects, booleans or expressions. String values need to be written within single or double quotes, for example, ‘John Doe’ or “John Doe”, and number values should not be written within quotes, for example, 2 or 200.45.

- JS operators can be used to compute values; e.g.,

  var sum = (24 + 200) * 50;

- JS **expressions** are a combination of values, variables and operators

  var mul = 10 * 10;

  var x = num1 * 10;

  var name = "John" + " " + "Doe";

- For single-line JS comments, you can use “//”, and for multiline comments, you can use “/* */”.

- JavaScript (JS) programs are a list of programming instructions called **statements** that are executed by web browsers.
- JS statements consist of values, operators, expressions, keywords and comments. They are executed in a waterfall fashion, that is, one by one, in the order in which they are written.
- JS statements end with a semicolon (‘;’). Semicolons essentially separate JS statements. It is not mandatory to add a semicolon, but it is highly recommended. Using semicolons, you can add multiple JS statements in one line. For example,

  var name = “John Doe”;

  var age = 24;

- JS ignores multiple spaces. However, it is recommended to add white spaces to your code to make it more readable. For example,
- var name = “John”;
- A best practice is to avoid having more than 80 characters in one line for better readability. The best place to break a JS code to a new line is immediately after the operator.
- JavaScript (JS) variables are containers for storing data values. In earlier examples, you learnt that variables can store values and expressions.
- All JS variables must be identified with a unique name called **identifier**. The rules for writing an identifier name are as follows:
- Names can contain letters, digits, the dollar sign (‘$’) and the underscore sign (‘_’).
- Names should always begin with a letter.
- JavaScript is case-sensitive, i.e., *name* and *Name* are different variables.

- Reserved JS words cannot be used for an identifier.
- Get the details about reserved words from:
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar
- The best practice is to **not** keep a name, such as x, y, z, that does not indicate the meaning or significance of that variable. The identifier name should signify the meaning of the variable; for example, *name* and *age*.

- In JS, you can assign a value to the variable using the ‘=’ operator.