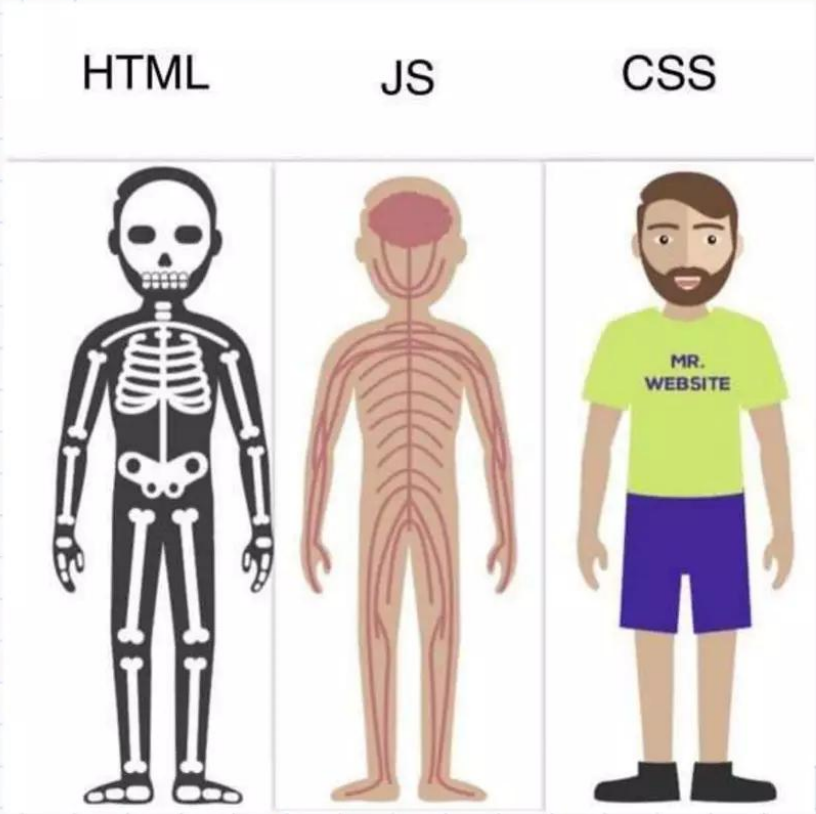
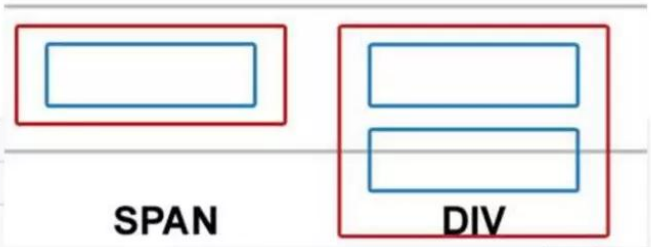

HTML
JS
CSS
MR.
WEBSITE


HTML
HTML+CSS

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
...	Set ... in boldface
...	Set ... in italics
...	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
...	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

Google Developer Student Clubs



Div Vs Span:

1.The tag is a block level element.	1.The tag is an inline element.
2. It is best to attach it to a section of a web page.	2. It is best to attach a CSS to a small section of a line in a web page.
3. It accepts align attribute.	3. It does not accept align attribute.
4. This tag should be used to wrap a section, for highlighting that section.	4. This tag should be used to wrap any specific word that you want to highlight in your webpage.

IMAGE:

```
<img src="VJTI.jpg" alt="VJTI" width="500" height="600">
```

HTML CSS

```
1 <img class="fit-picture"
2      src="/media/cc0-images/grapefruit-slice-332-
  332.jpg"
3      alt="Grapefruit slice atop a pile of other
  slices">
4
```

OUTPUT

Google Developer Student Clubs

LINKS:

3 Types of Links in HTML Tags:

1.
```
<a href="https://vjti.ac.in/">VJTI</a>
```

2.
```
<link rel="stylesheet" href="styles.css">
```

3.
```
<nav>
    <a href="/html/">HTML</a>
    <a href="/css/">CSS</a>
</nav>
```

LISTS:


ORDERED LIST

Apple
Mango
Grapes
Pineapple
Orange

1. Apple
2. Mango
3. Grapes
4. Pineapple
5. Orange


BULLETED LIST

Apple
Mango
Grapes
Pineapple
Orange

• Apple
• Mango
• Grapes
• Pineapple
• Orange


Working with HTML-forms
Why can't you just be simple?
Screams

FORMS:

- <form> is just another kind of HTML tag
- Usually the purpose is to ask the user for information. The information is then sent back to the server.
- Form elements include: buttons, checkboxes, text fields, radio buttons, drop-down menus, etc
- A form usually contains a Submit button to send the information in the form elements to the server

Google Developer Student Clubs

ATTRIBUTES OF FORMS:

It has two attributes commonly, method and action

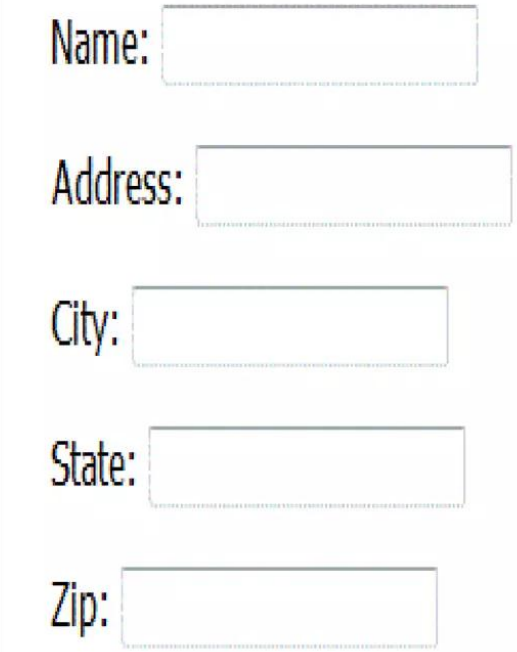
Action

- It is the url where the form data will be sent after the user clicks on submit button.

Method (GET/POST)

- Method can vary depending on the type of data you want to send.
- In get method form data is sent as URL variables.
- In post method form data is sent as HTTP post transaction.

Tag	Description
<form>	Defines an HTML form for user input
<input>	Defines an input control
<textarea>	Defines a multiline input control (text area)
<button>	Defines a clickable button
<select>	Defines a drop-down list
<optgroup>	Defines a group of related options in a drop-down list
<option>	Defines an option in a drop-down list
<label>	Defines a label for an <input> element
<fieldset>	Groups related elements in a form
<legend>	Defines a caption for a <fieldset> element
<datalist>	Specifies a list of pre-defined options for input controls
<output>	Defines the result of a calculation




```
<form action="formprocessor.html" method="get">
    <p>Name: <input type="text" name="Name" /></p>
    <p>Address: <input type="text" name="Address" /></p>
    <p>City: <input type="text" name="City" /></p>
    <p>State: <input type="text" name="State" /></p>
    <p>Zip: <input type="text" name="Zip" /></p>
</form>
```

Table Element

- The <table> HTML element represents **tabular** data — that is, information presented in a two-dimensional table comprised of rows and columns of cells containing data.

HTML

```
1
2  <table>
3    <tr>
4      <th>Column 1</th>
5      <th>Column 2</th>
6      <th>Column 3</th>
7    </tr>
8    <tr>
9      <td>Row 1 Column 1</td>
10     <td>Row 1 Column 2</td>
11     <td>Row 1 Column 3</td>
12   </tr>
13   <tr>
14     <td>Row 2 Column 1</td>
15     <td>Row 2 Column 2</td>
16     <td>Row 2 Column 3</td>
17   </tr>
18 </table>
```

Column 1	Column 2	Column 3
Row 1 Column 1	Row 1 Column 2	Row 1 Column 3
Row 2 Column 1	Row 2 Column 2	Row 2 Column 3

TABLES

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

DISPLAY: INLINE AND BLOCK

1.Block-level Elements

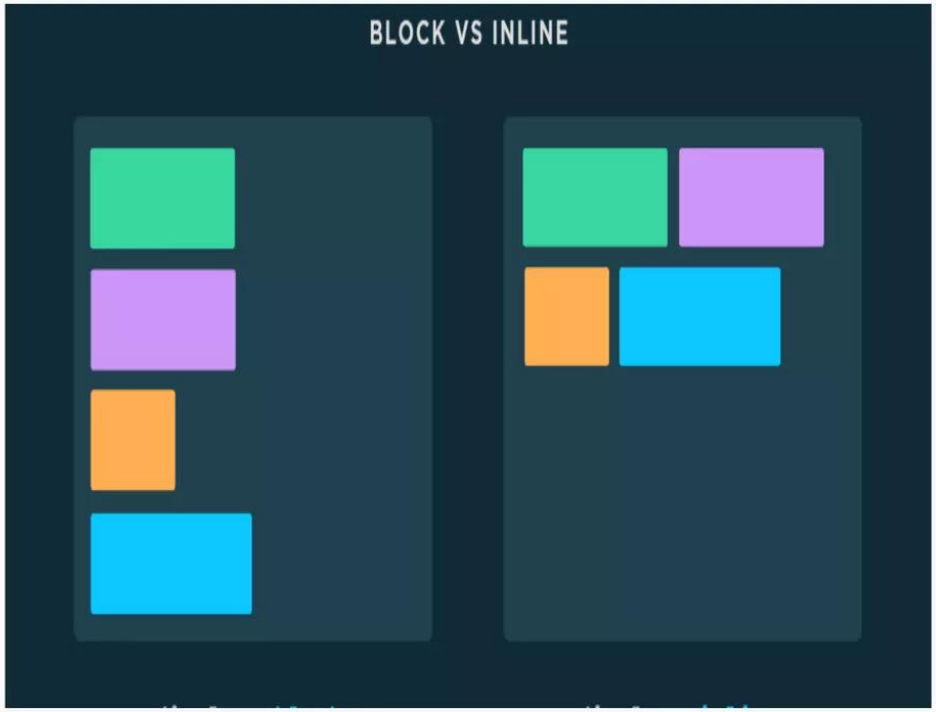
- Always starts on new line

Eg: <p> tag

1.Inline Elements

- Does not start on a new line.

Eg: tag inside a <p> tag



Block Elements	Inline Elements
• <p>	• <a>
• <h1>	• <b>
•	• <i>
• <hr>	• <img>

ID Attribute

- The HTML id attribute is used to specify a unique id for an HTML element.
- You cannot have more than one element with the same id in an HTML document.

Note :-

- The id name is case sensitive!
- The id name must contain at least one character, cannot start with a number, and must not contain whitespaces (spaces, tabs, etc.).

```
<h1 id="myHeader">My Header</h1>
```

Class Attribute

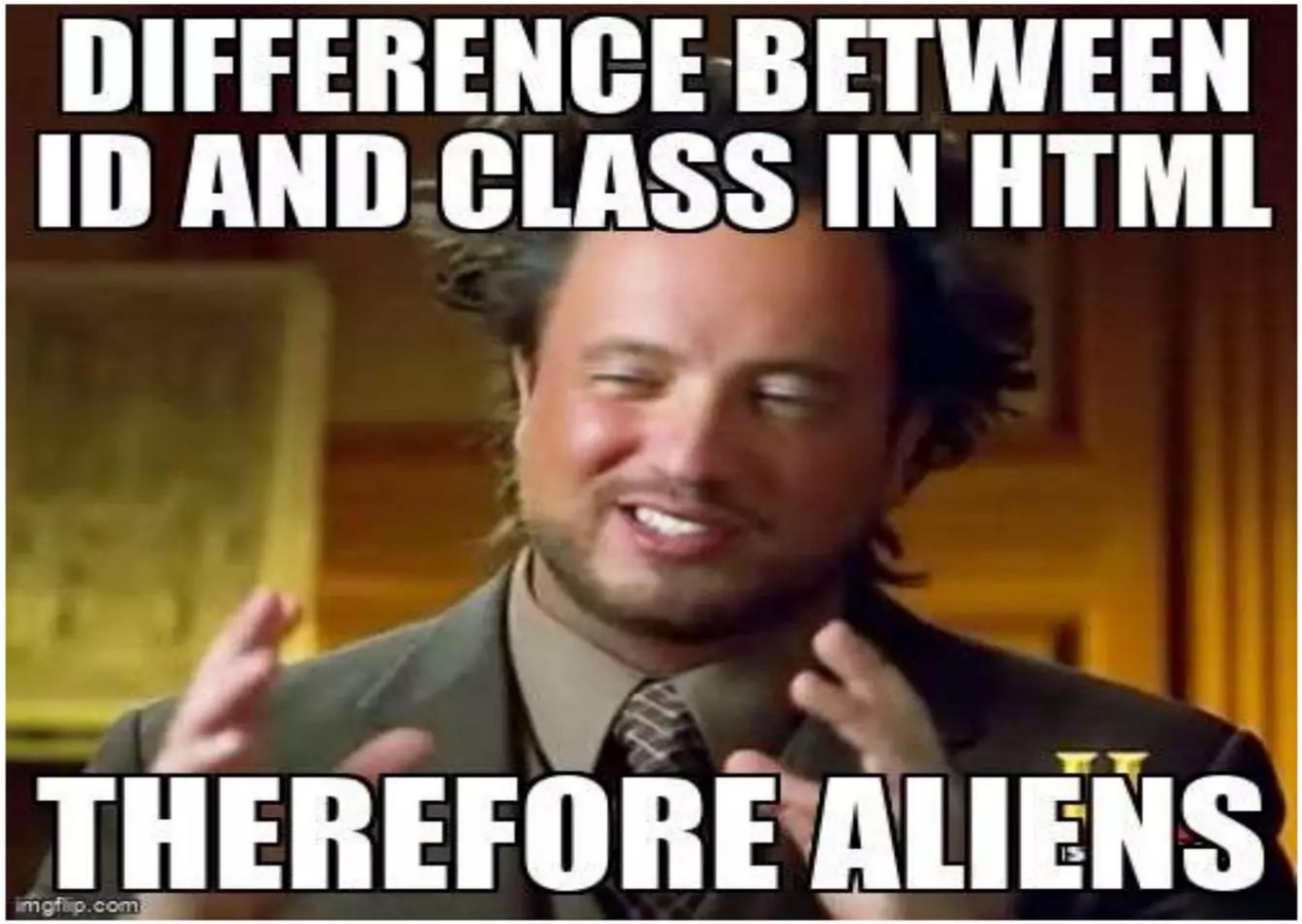
- The HTML class attribute is used to specify a class for an HTML element. Multiple HTML elements can share the same class.
- The class attribute is often used to point to a class name in a style sheet.

Tip: The class attribute can be used on any HTML element. Even if the two elements do not have the same tag name, they can have the same class name, and get the same styling.

Note: The class name is case sensitive!

```
<h1 class="intro">Header 1</h1>
```

```
<body>

    <!-- An element with unique id -->
    <h1 id="myHeader">My Cities</h1>

    <!-- Multiple Elements with same class -->
    <h3 class="city">Mumbai</h3>
    <h3 class="city">Thane</h3>
    <h3 class="city">Pune</h3>

</body>
```


DIFFERENCE BETWEEN
ID AND CLASS IN HTML
THEREFORE ALIENS
imgflip.com

HTML Entities

- Reserved characters in HTML must be replaced with character entities. Some characters are reserved in HTML.
- If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.
- Character entities are used to display reserved characters in HTML.

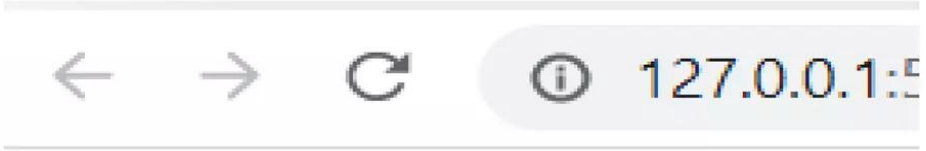
A character entity looks like -

&entity_name;

OR

&#entity_number;

```
<body>

    <p>Hello this is a paragraph</p>

</body>
```



Hello this is a paragraph

```
<body>

    <p>Hello this is a                    paragraph</p>

</body>
```

Hello this is a paragraph

Google Developer Student Clubs

```
<body>

    <!-- A paragraph with 3 spaces -->
    <p>Hello this is a       paragraph</p>

</body>
```

Hello this is a paragraph

```
<body>

    <!-- paragraph tag representation -->
    <p>Paragraph in HTML is <p> </p>
    <p>Paragraph in HTML is &lt;p&gt;</p>

</body>
```

Paragraph in HTML is

Paragraph in HTML is <p>

Google Developer Student Clubs

Result	Description	Entity Name	Entity Number
	non-breaking space		
<	less than	&lt;	&#60;
>	greater than	&gt;	&#62;
&	ampersand	&amp;	&#38;
"	double quotation mark	&quot;	&#34;
'	single quotation mark (apostrophe)	&apos;	&#39;
¢	cent	&cent;	&#162;
£	pound	&pound;	&#163;
¥	yen	&yen;	&#165;
€	euro	&euro;	&#8364;
©	copyright	&copy;	&#169;
®	registered trademark	&reg;	&#174;

S

Semantic Elements

- Semantic elements = elements with a **meaning**.
- A semantic element clearly describes its meaning to both the browser and the developer.
- Examples of non-semantic elements: and - Tells nothing about its content.
- Examples of semantic elements: <form>, <table>, and <article> - Clearly defines its content.
- Many web sites contain HTML code like: to indicate navigation, header, and footer.

Some more Semantic Elements

- <article>
- <aside>
- <details>
- <figure>
- <footer>
- <header>
- <nav>
- <summary>
- <main>
- <section>

Tag	Description
<article>	Defines independent, self-contained content
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text
<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

```
<body>
    <details>
    <summary>GDSC</summary>
    Google Developer Students Club!!
    </details>
</body>
```

▶ GDSC

▼ GDSC
Google Developer Students Club!!

Final Slide HTML

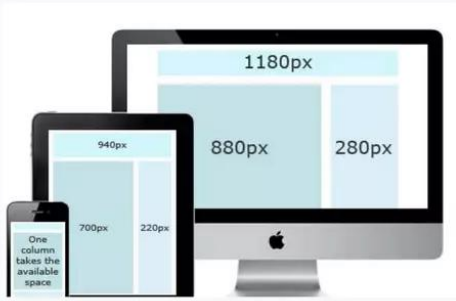
Google Developer Student Clubs

Google Developer Student Clubs

CSS

Introduction

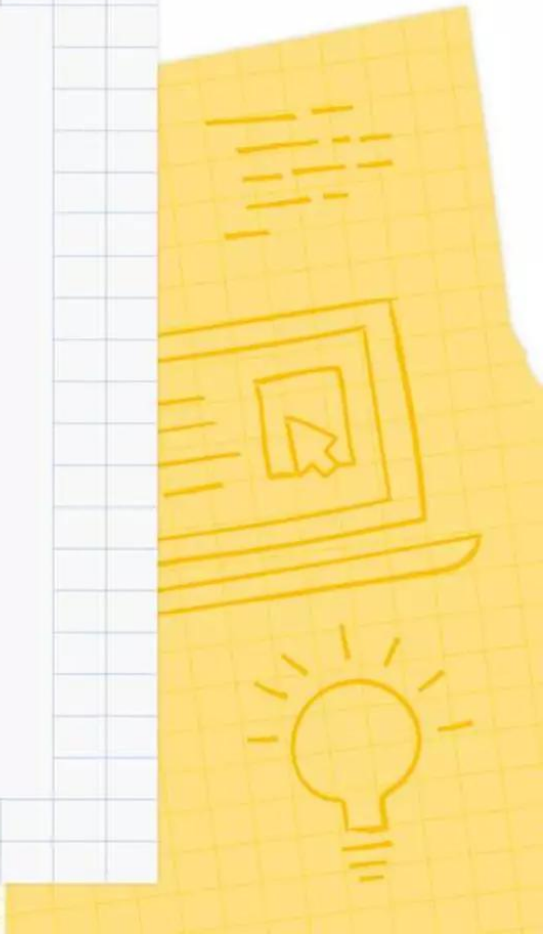
- **C**ascading **S**tyle **S**heets
- Defines styles for web pages including design and layout
- Responsible for variations in display for different screens and devices



Types

Inline CSS

- CSS property is in the body section attached with element.
- It is specified within HTML tag using style attribute.



Google Developer Student Clubs

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

<h1 style="color:blue;text-align:center;">GDSC VJTI</h1>
<p style="color:red;">Learning Inline CSS.</p>

</body>
</html>
```

GDSC VJTI

Learning Inline CSS.

Internal or Embedded CSS

- Used when single HTML document should be styled uniquely.
- CSS rule set should be within HTML file in the head section.

Google Developer Student Clubs

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

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

<h1>GDSC VJTI</h1>
<p>Learning Internal CSS.</p>

</body>
</html>
```

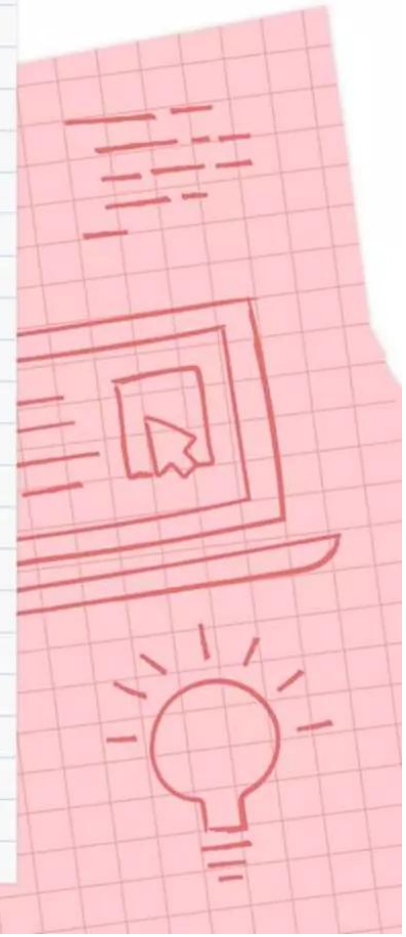
GDSC VJTI

Learning Internal CSS.

Google Developer Student Clubs

External CSS

- CSS property written in separate file with **.css** extension.
- Should be linked to HTML using **link** tag.
- File contains only style property with the help of tag attributes.(example: class,id,heading,etc.)

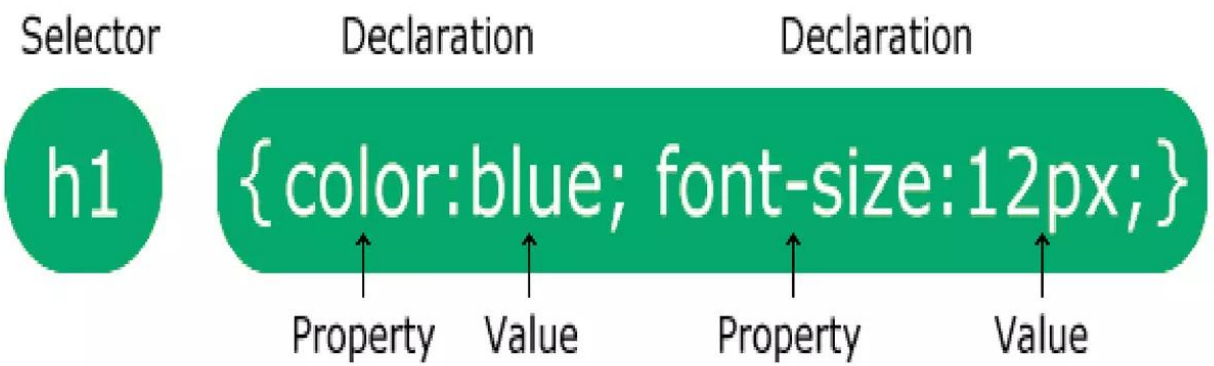


```html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>GDSC VJTI</h1>
<p>Learning External CSS</p>

</body>
</html>
```

"mystyle.css"

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

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

SYNTAX



- Selector points to HTML element you want to style
- Declaration block contains one or more declarations separated by semicolons
- Each declaration includes CSS property name and value, separated by colon
- Declaration blocks surrounded by curly braces

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

- **p** is selector in CSS(points to HTML element : **<p>**)
- **color** is property and **red** is a property value
- **text-align** is property and **center** is property value

Selectors

- Used to “find”(or select) the HTML elements you want to style.

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

Element selector

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

ID Selector

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

Class selector

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

Universal selector

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

Grouping selector

Colors

- Can be defined using three main types:
- **Name** of colors (Eg. Orange,Tomato,Gray)
- **RGB** values (rgb(255,99,71))
- **HEX** values (#ff6347)

Color name	Hexadecimal Value	Decimal Value or rgb() value
Red	#FF0000	rgb(255,0,0)
Orange	#FFA500	rgb(255,165,0)
Yellow	#FFFF00	rgb(255,255,0)
Pink	#FFC0CB	rgb(255,192,203)

```
<h3 style="color:Tomato;">Is Tomato a fruit or vegetable?</h3>
<h3 style="color:rgb(153, 204, 255);">It is a fruit</h1>
<h3 style="color:#af38eb;">No, It is a vegetable</h1>
```

Is Tomato a fruit or vegetable?

It is a fruit

No, It is a vegetable

Borders

- Allows to specify following things of element:
- **Style:** dotted,dashed,solid,none,etc
- **Width:** 5px,thick,medium,etc
- **Color:** can be specified using rgb,hex or color names
- **Radius:** adds rounded border to element

```
<p style="border-top-style: dotted;">A dotted border.</p>
<p style="border-style: solid; border-width: 5px;border-color: red;border-
radius: 10px;">A solid border.</p>
```

A dotted border.

A solid border.

Fonts

- Adds value to your text
- Has huge impact on how reader's experience a website
- **font-family:** specifies font of a text
- **font-style:** normal,italic,oblique
- **font-size:** sets size of font

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
<style>
body {
  font-family: "Sofia", sans-serif;
   font-size: 15px;
   color: gray;
   font-style: italic;
}
</style>
```

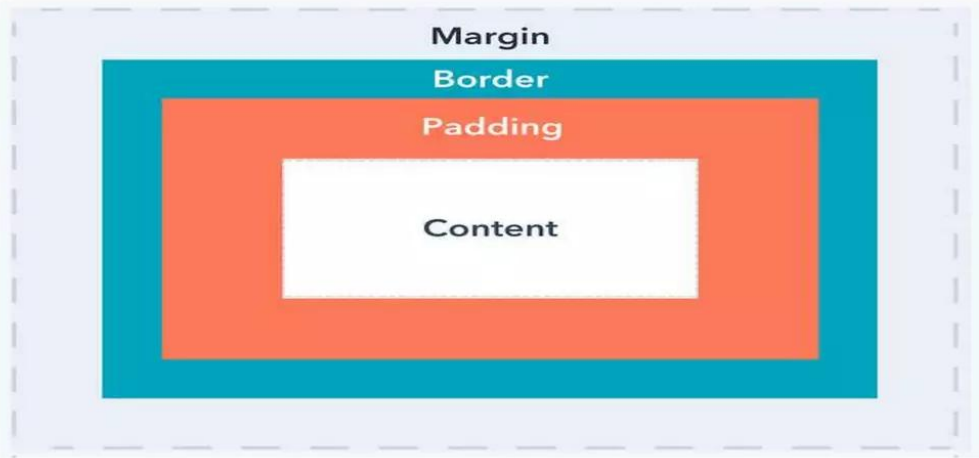
> Hello folks!
>
> How's the josh?

```css
div {
  border: 1px solid black;
  margin-top: 100px;
  margin: 25px 50px 75px 100px;
  padding: 25px 50px 25px 10px;
  background-color: lightblue;
}
```

This div element has a top margin of 25px, a right margin of 50px, a bottom margin of 75px, and a left margin of 100px. This div element has a top padding of 25px, a right padding of 50px, a bottom padding of 25px, and a left padding of 10px.

Margin & Padding

- Margins creates space around elements,outside of any defined borders
- Padding creates space around elements,inside of any defined borders
- **length:** specifies in px,cm
- **percentage:** specifies in % of width containing element



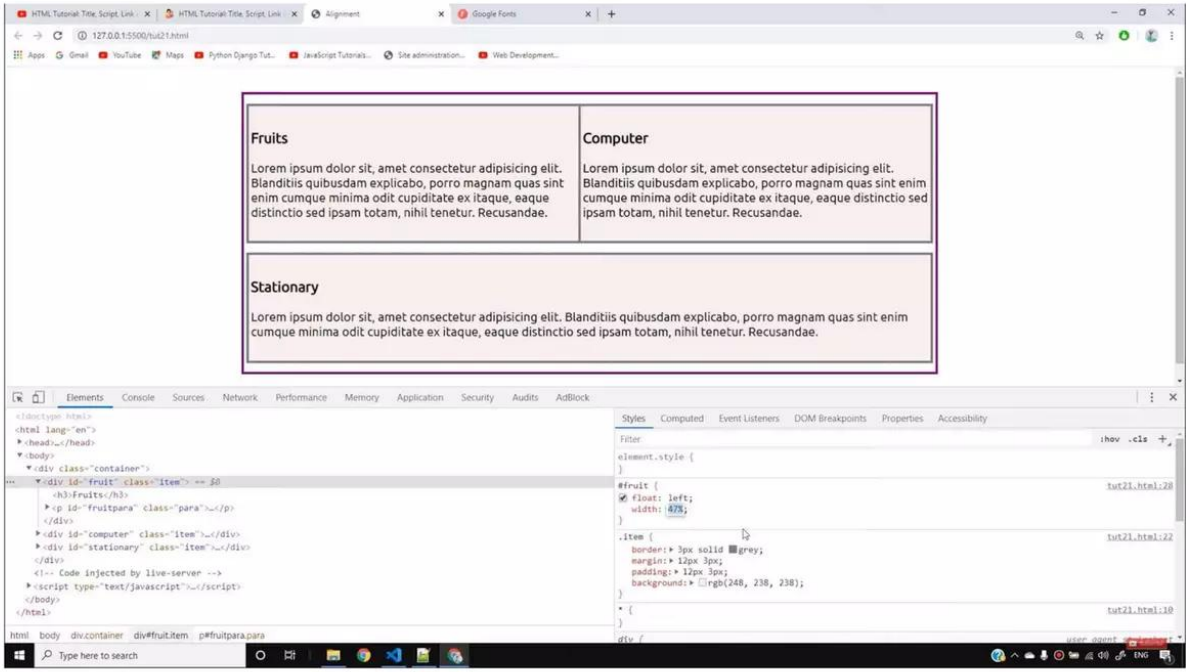
Float & Clear

Float specifies how an element should float.
Clear specifies what elements can float beside the cleared element and on which side.

Properties :

- Left- The elements floats to the left of its container.
- Right- The elements floats to the right of its container.
- None- The element does not float (it will be displayed just where it occurs in the text). This is default.
- Inherit- The element inherits the float value of its parent.

```
#
#fruit   {
        float: right;
        width: 48%;
    }

    #computer
        float: left;
        width: 48%;
    }

    #stationary {
        /* float: left; */
        clear: both;
        clear: left;
        width: 100%;

     }
```



Styling links & buttons

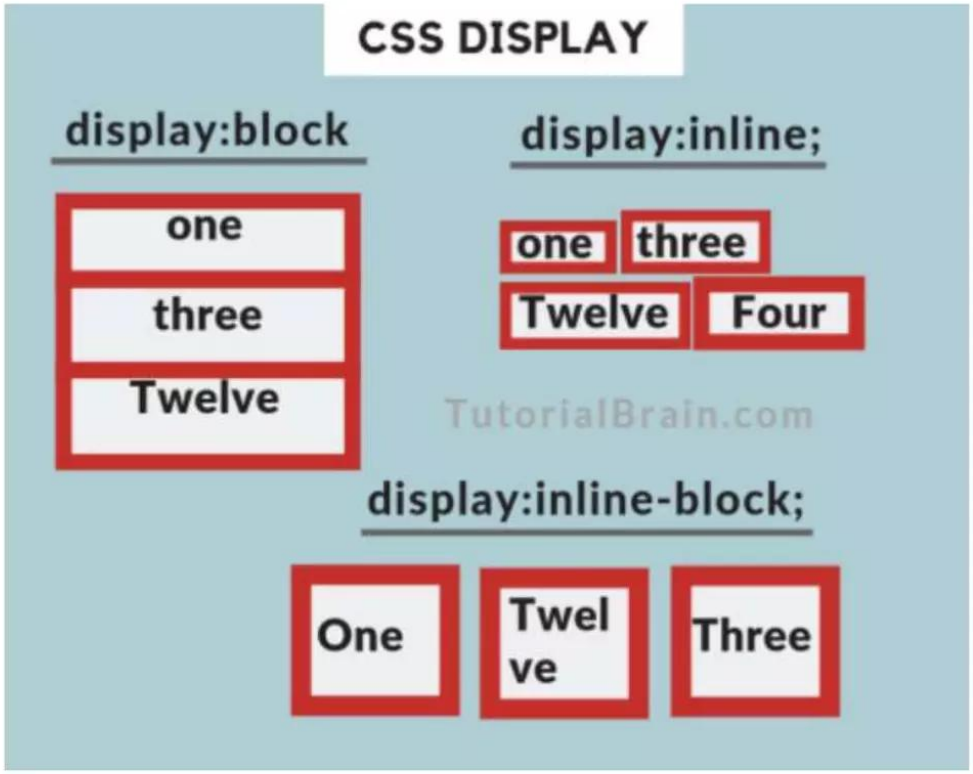
```
<a href="https://yahoo.com" class="btn">Read more</a>
<button class="btn">Contact us</button>
```

SIMPLIFIED

```
.btn{
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        font-weight: bold;
        background-color: crimson;
        padding:6px;
        border: none;
        cursor:pointer;
        font-size: 13px;
        border-radius: 4px;
    }
```

MODIFIED

Displays:



Google Developer Student Clubs

Syntax of Display Inline :
display: inline;

Syntax of Display Block :
display: block;

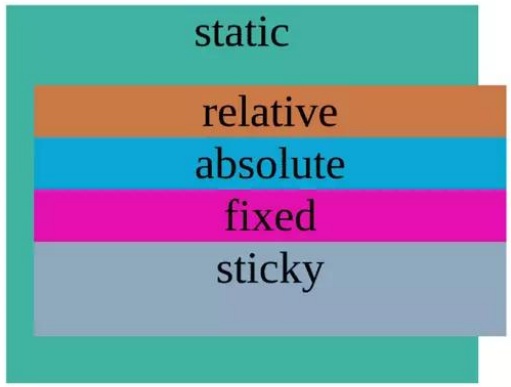
Syntax of Display inline-block :
display: inline-block;

Position property

Specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`



Navigation menu

An organised list of links to other webpages, usually internal pages

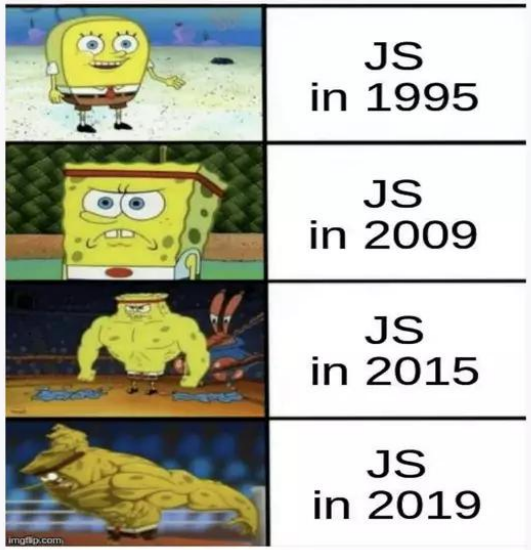
Syntax

```
<ul cl<ul class="menu Navigation-Menu-Classes">
    ....
</ul>
```

Google Developer Student Clubs

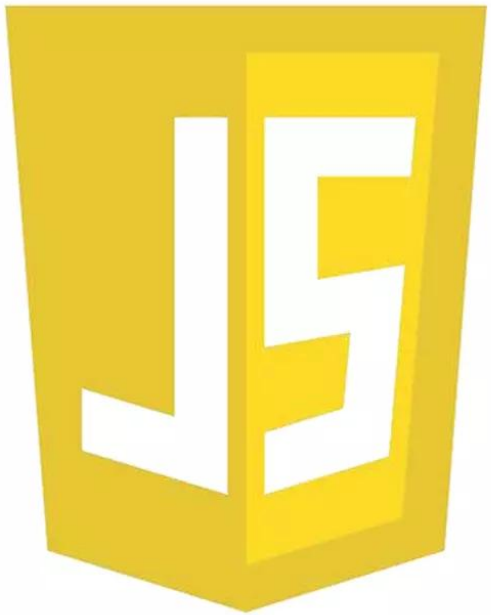
Google Developer Student Clubs

JAVASCRIPT



WHAT IS JAVASCRIPT??

- JavaScript is a client-side cross-platform, object-oriented scripting language.
- JS is the soul of every webpage. It is used to programmatically perform actions within the page.
- It’s a great language to use to help build dynamic and interactive web pages.
- JavaScript runs directly in the web browser, which means we don’t need any additional resources to execute our JavaScript code.
- It can also run on a server via Node.js
- So, for now what we only need is any text editor such as VSCode to write JS code, execution would be handled by browser itself



How to integrate JS to HTML??

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

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

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

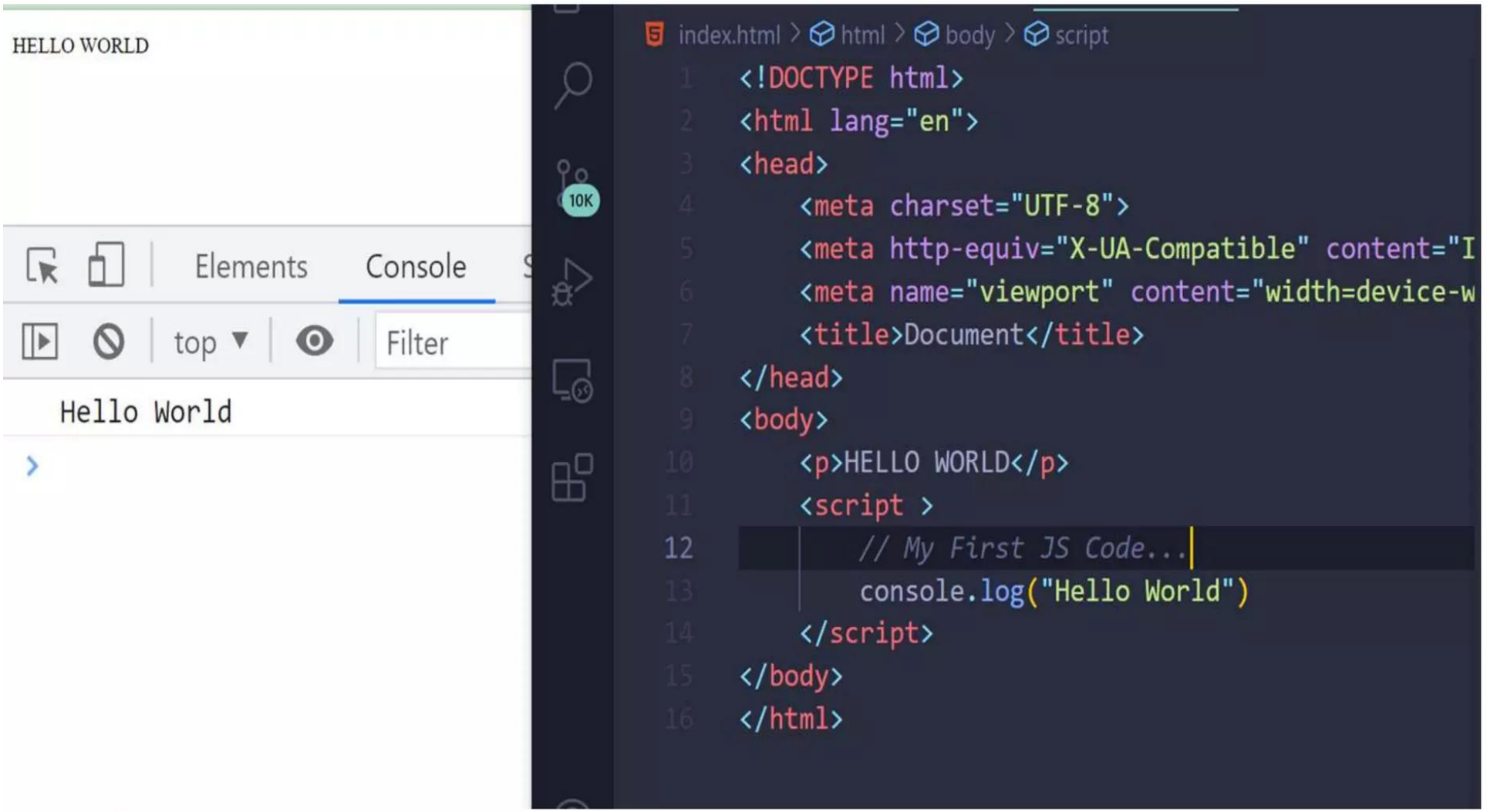
- Import a Javascript file using the <script> tag:

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

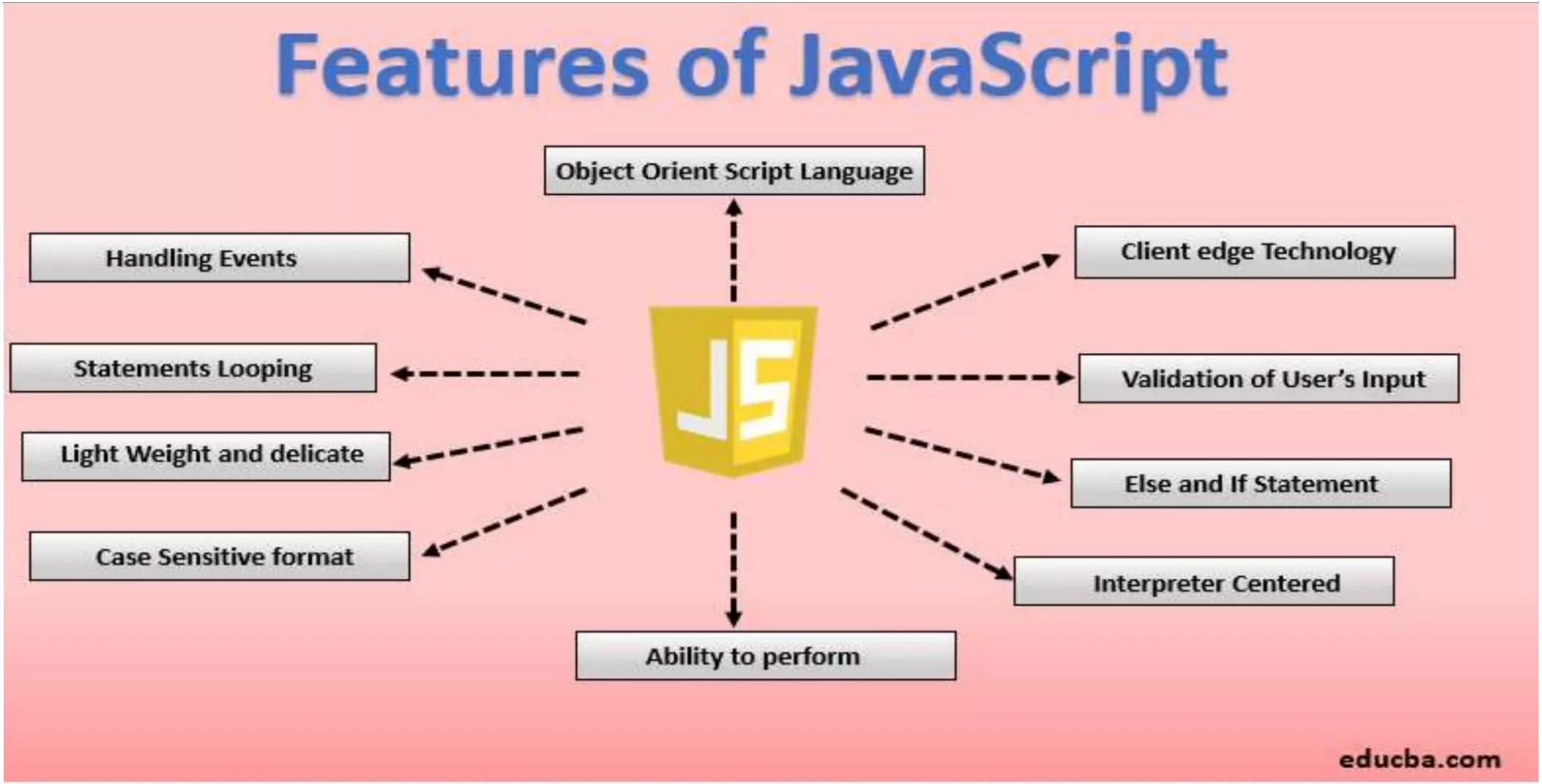
- Inject the code on an event inside a tag:

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

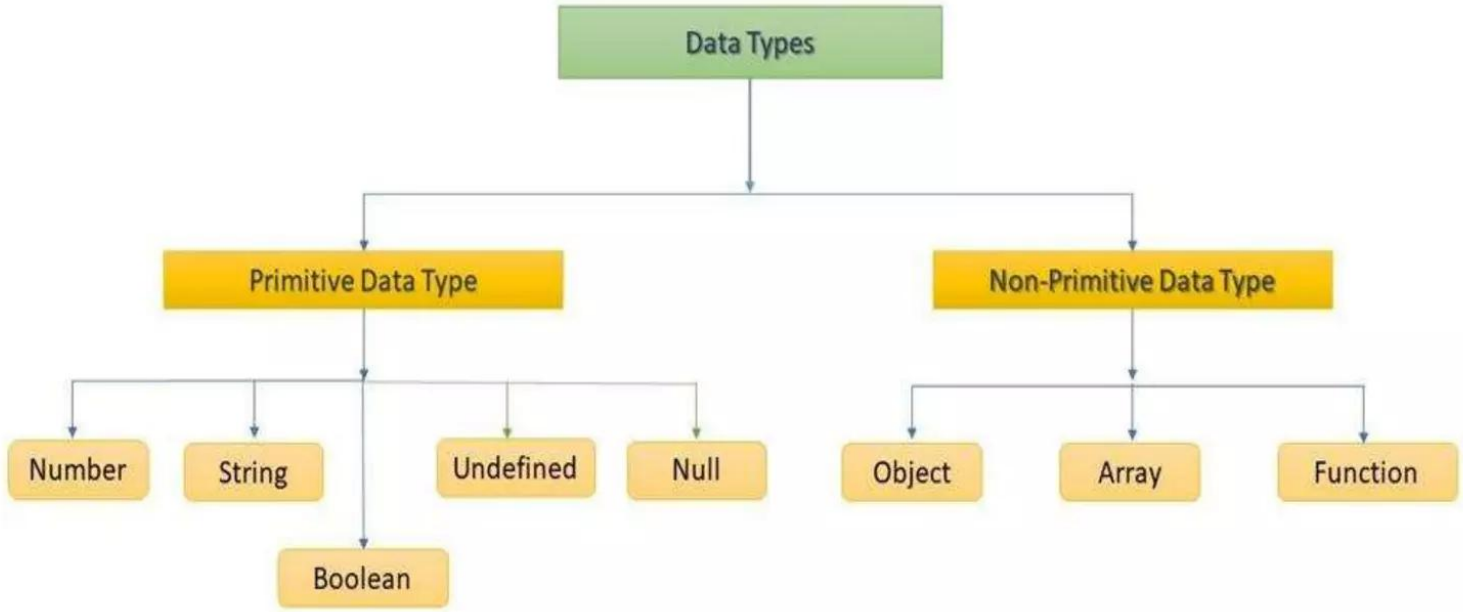
Google Developer Student Clubs


HELLO WORLD
Elements
Console
top
Filter
Hello World
index.html > html > body > script
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="I
<meta name="viewport" content="width=device-w
<title>Document</title>
</head>
<body>
HELLO WORLD
<script >
// My First JS Code...
console.log("Hello World")
</script>
</body>
</html>

Features of JavaScript



DATA TYPES



Google Developer Student Clubs

Numbers

```
50
3.874
-2
1.9999293999
NaN
NaN
number
>
```

```
1  console.log(50); //number
2  console.log( 3.874 );//number
3  console.log( -2 ); //number
4  console.log( 1.9999293999); //number
5
6
7  /* Not a number(NaN) is a number that is
8  not a legal number */
9  console.log(0 / 0 );//Number
10 console.log(1 + NaN); //Number
11 console.log(typeof (0 / 0));
```

Null and Undefined

Null is a special value that represents an empty or unknown value. For example, let number = null;

The code above suggests that the number variable is empty at the moment and may have a value later.

Undefined means the value does not exist in the compiler. It is the global object.


```
> typeof null
< 'object'
> typeof undefined
< 'undefined'
>
```

0

null

undefined

Google Developer Student Clubs

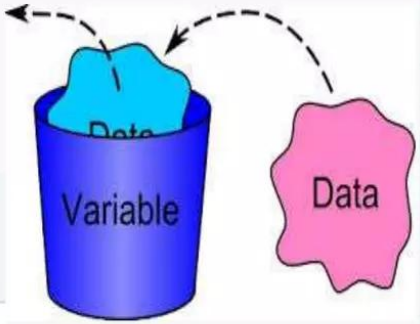
VARIABLES

Variables are like labels for values.We use variables to store data temporarily.

For declaring variables, we use the let keyword.
let someName = value;

```
index.js > ...
1  let a = 78;
2  let b = "GDSC";
3  console.log(a);
4  console.log(b);
```

```
Filter   Default levels ▼
No Issues
78                index.js:3
GDSC              index.js:4
>
```



Rules for declaring variables

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

```
> let GDSC_1 = 1
< undefined
> let 1a = 1
✖ Uncaught SyntaxError: Invalid or unexpected token
> let _GDSC = "fun"
< undefined
```

```
> let gdsc = 1
<· undefined
> let GDSC = 2
<· undefined
> console.log(gdsc,GDSC)
  1 2
<· undefined
> let if = 2
✕ Uncaught SyntaxError: Unexpected token 'if'
>
```

Google Developer Student Clubs

Dynamic Typing

JavaScript is dynamically typed language. We don't need to define the type of variable, JS interprets it.

Google Developer Student Clubs

Operators

JavaScript Operators are symbols that are used to perform operations on operands.

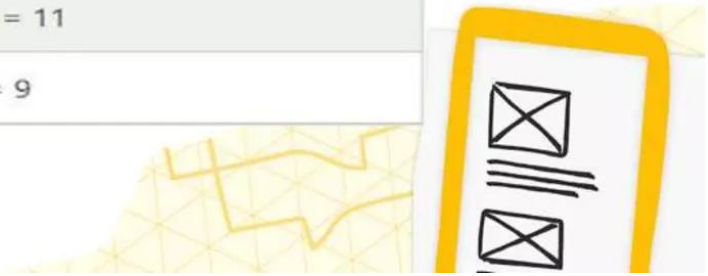
There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9



Comparison Operators

The JavaScript comparison operator compares the two operands.

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	'1' === 1 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

Google Developer Student Clubs

Math Class

1 = 1

1 ≠ 2

Normal Coding Languages

1 == 1

1 != 2

Javascript

1 === 1

1 !== 2

BitWise Operators

The bitwise operators perform bitwise operations on operands.

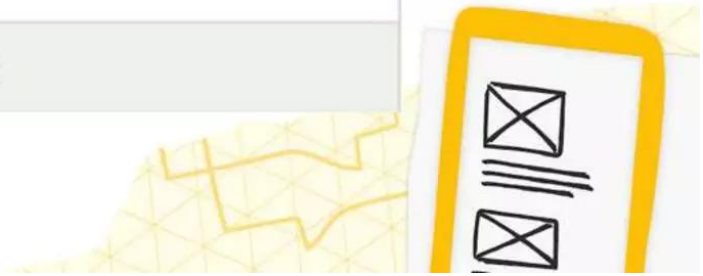
Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40
>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

Logical Operators

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

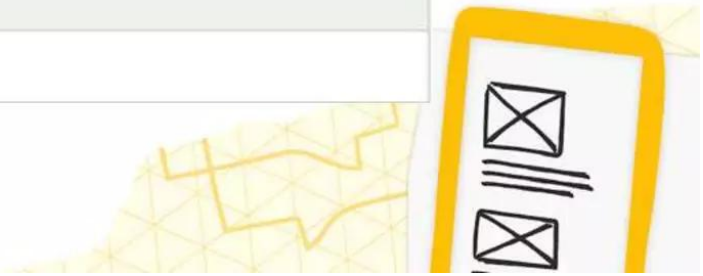
Assignment Operators

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0



Special Operators

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator.



JavaScript Strings

A string is a sequence of one or more characters that may consist of letters, numbers, or symbols. Strings in JavaScript are primitive data types and immutable.

- In JavaScript, there are three ways to write a string: they can be written inside single quotes (' '), double quotes (" "), or backticks (` `).
  The type of quote used must match on both sides, however it is possible that all three styles can be used throughout the same script.

```
'This string uses single quotes.';
"This string uses double quotes.";
```

- The third and newest way to create a string is called a template literal. Template literals use the backtick (also known as a grave accent) and work the same way as regular strings with a few additional bonuses,

```
`This string uses backticks.`;
```

One special feature of the template literal feature is the ability to include expressions and variables within a string. Instead of having to use concatenation, we can use the ${} syntax to insert a variable.

```
const poem = "The Wide Ocean";
const author = "Pablo Neruda";
const favePoem = `My favorite poem is ${poem}
  by ${author}.`;
```

output:
My favorite poem is The Wide Ocean by Pablo Neruda

String Methods

Methods	Description
charAt()	It provides the char value present at the specified index.
charCodeAt()	It provides the Unicode value of a character present at the specified index.
concat()	It provides a combination of two or more strings.
indexOf()	It provides the position of a char value present in the given string.
lastIndexOf()	It provides the position of a char value present in the given string by searching a character from the last position.
search()	It searches a specified regular expression in a given string and returns its position if a match occurs.
match()	It searches a specified regular expression in a given string and returns that regular expression if a match occurs.
replace()	It replaces a given string with the specified replacement.
substr()	It is used to fetch the part of the given string on the basis of the specified starting position and length.
substring()	It is used to fetch the part of the given string on the basis of the specified index.
slice()	It is used to fetch the part of the given string. It allows us to assign positive as well negative index.

Google Developer Student Clubs

ARRAYS

In JavaScript, array is a single variable that is used to store different elements. It is often used when we want to store list of elements and access them by a single variable.

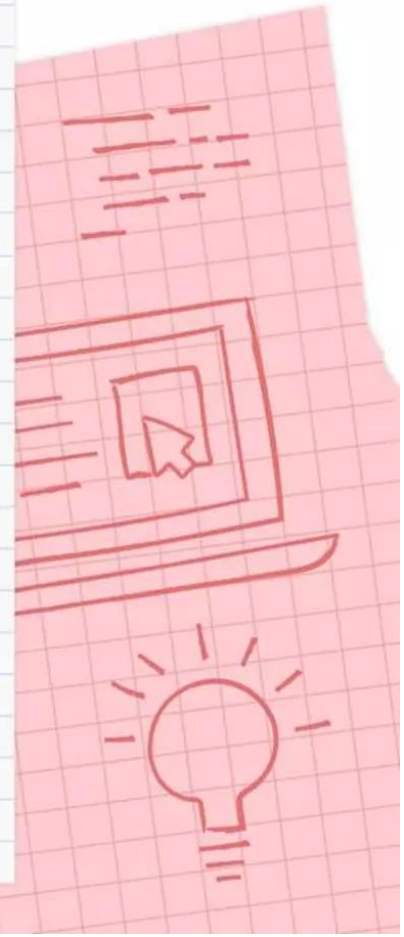
Declaration of an Array

There are basically two ways to declare an array.

Example:

```
var House = [ ]; // method 1
var House = new Array(); // method 2
```

JavaScript arrays are resizable and can contain a mix of different data types.



Initialisation of Array

Example 1:

// Initializing while declaring

```
var house = ["1BHK", "2BHK", "3BHK", "4BHK"];
```

//Accessing array elements

```
var house1=house[0]
```

Example 2:

// Creates an array of 4 undefined elements

```
var house = new Array(4);
```

// Now assign values

```
house[0] = "1BHK"
house[1] = “2BHK
house[2]  = "3BHK"
house[3] = "4BHK"
```

Example 3:

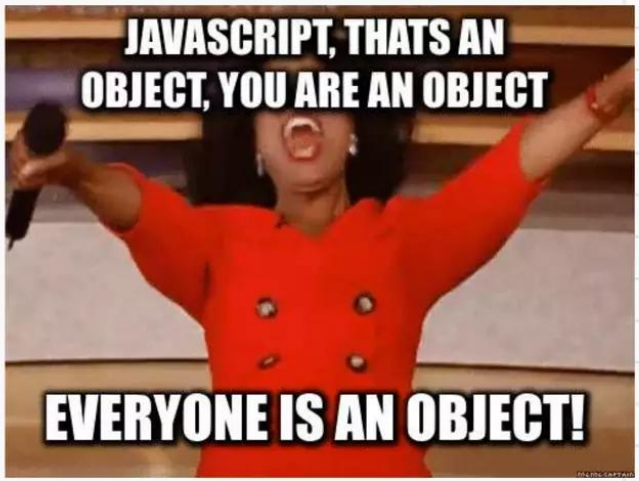
We can store Numbers, Strings and Boolean in a single array.

// Storing number, boolean, strings in an Array

```
var house = ["1BHK", 25000, "2BHK", 50000, "Rent", true];
```

OBJECTS

A javaScript object is an entity having state and behavior (properties and method). For example: car, pen, bike, chair, glass, keyboard, monitor etc.



Creating Objects in JavaScript

There are 3 ways to create objects:

1.By object literal
2.By creating instance of Object directly (using new keyword)
3.By using an object constructor (using new keyword)
4.Using es6 classes:

To explore more,click here

Google Developer Student Clubs

Functions

Suppose you need to create a program to create a circle and color it. You can create two functions to solve this problem:
a function to draw the circle
a function to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable. JavaScript also has a huge number of inbuilt functions. For example, Math.sqrt() is a function to calculate the square root of a number.

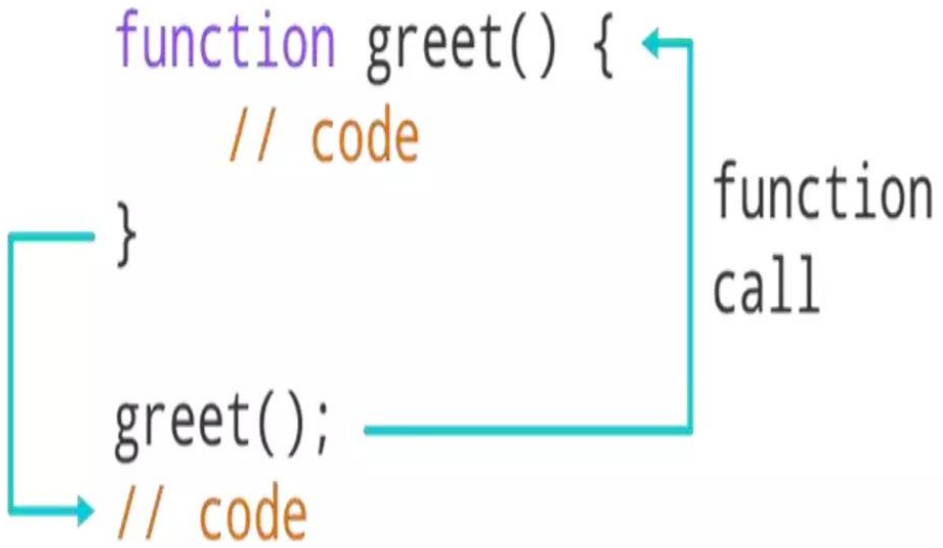
```
The syntax to declare a function is:
function nameOfFunction () {
    // function body
}
```

Example

```
// declaring a function named greet()
function greet() {
   console.log("Hello there");
}
```

Calling a function

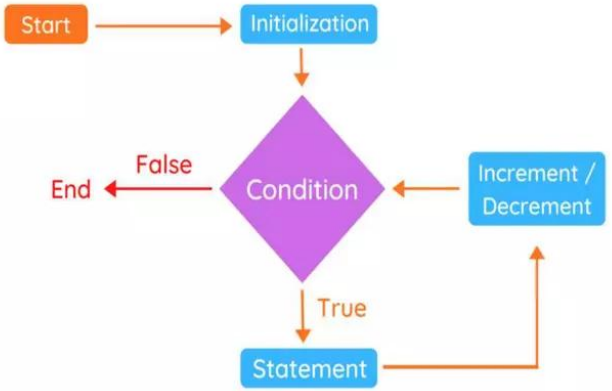
```
// function call
greet();
```



Google Developer Student Clubs

LOOPS

The JavaScript loops are used to iterate the piece of code using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.



There are four types of loops in JavaScript.
1.for loop
2.while loop
3.do-while loop
4.for-in loop

FOR LOOP :

```
<script>
document.write("The For
loop<br>")
for (i=1; i<=5; i++)
{
document.write(i + "<br/>")
}
</script>
```

The For loop
1
2
3
4
5

For….In loop

CODE

```
<script>
    document.write("For...in loop<br>")
    const numbers=[10,20,30,40,50];
    let txt="";
    for(let x in numbers){
        document.write(x+"<br>");
    }
</script>
```

RESULT

For...in loop
10
20
30
40
50

While Loop

CODE

```
<script>
document.write("The while loop<br>")
let i=11;
while (i<=15)
{
document.write(i + "<br/>");
i++;
}
</script>
```

RESULT

The while loop
11
12
13
14
15

Do While Loop

CODE

```
<script>
document.write("The Do..while
loop<br>")
let i=21;
do{
document.write(i + "<br/>");
i++;
}while (i<=25);
</script>
```

RESULT

The Do..while loop
21
22
23
24

EVENTS

Common HTML Events

Here is a list of some common HTML events:

Event	Description
onchange	An HTML element has been changed
onclick	The user clicks an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeydown	The user pushes a keyboard key
onload	The browser has finished loading the page

On Click Event

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to display the date.</p>
<button onclick="displayDate()">The time is?
</button>
<script>
function displayDate() {
  document.getElementById("demo").innerHTML
= Date();
}
</script>
<p id="demo"></p>
</body>
</html>
```