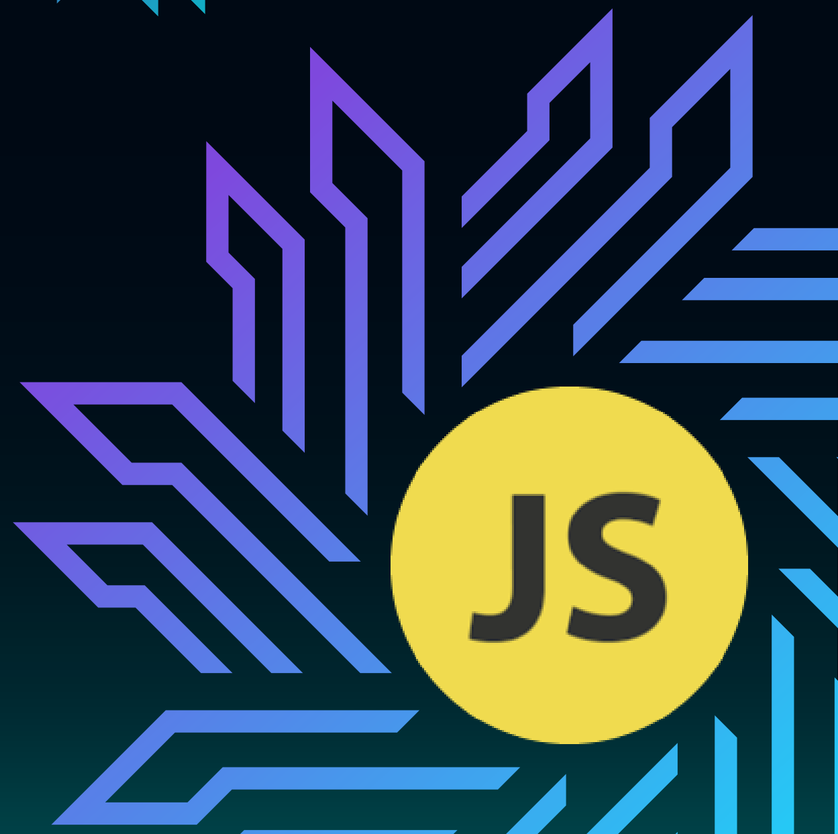
MNNIT CC

JAVASCRIPT

WEBSTER - 2nd Class

Created By : CC MNNIT



What is JS?

JavaScript is a scripting or programming language that allows you to implement complex features on web pages.

Visit : W3School

Scripting Language

- Basically, all scripting languages are programming languages.
- Scripting languages are interpreted e.g. JS, PHP, Python. Suitable for purposes where changes in code are more frequent.
- Programming languages are compiled e.g. Java, C, C++. Suitable for purposes where changes in code are less frequent.
- But Technically, Scripting languages are not fully interpreted theses days, they are compiled JIT(Just In Time).
- JIT Compilation -> Byte Code(Intermediate-level Non-Runnable Code) -> Interpretor -> Machine Code(Binary) -> CPU Execution.
- Types - Server-side scripting languages(JS, Python, PHP) and client-side scripting languages(JS).

How to add JS to Html Pages

Refer

- w3school
- MDN

<script> Tag

- The <script> tag is used to embed a client-side script (JavaScript).

- The <script> element either contains scripting statements, or it points to an external script file through the src attribute.

.

index.html > html

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1.0">
7       <title>Document</title>
8       <script src="./script.js"></script>
9   </head>
10  <body>
11      Hello World
12  <script>
13      console.log("Hello World");
14  </script>
15  </body>
16
17  </html>
```

Async and Defer Boolean attributes of script tag.

Without async or defer –

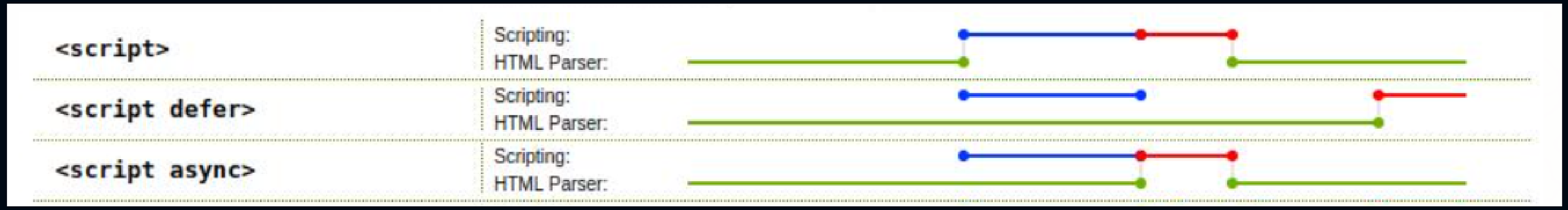
JavaScript files are fetched (downloaded) and executed in order of their presence in html file.

With defer –

The file gets downloaded asynchronously along with HTML Parsing, but executed only when the document parsing is completed.
They are executed in order of their presence in html file.

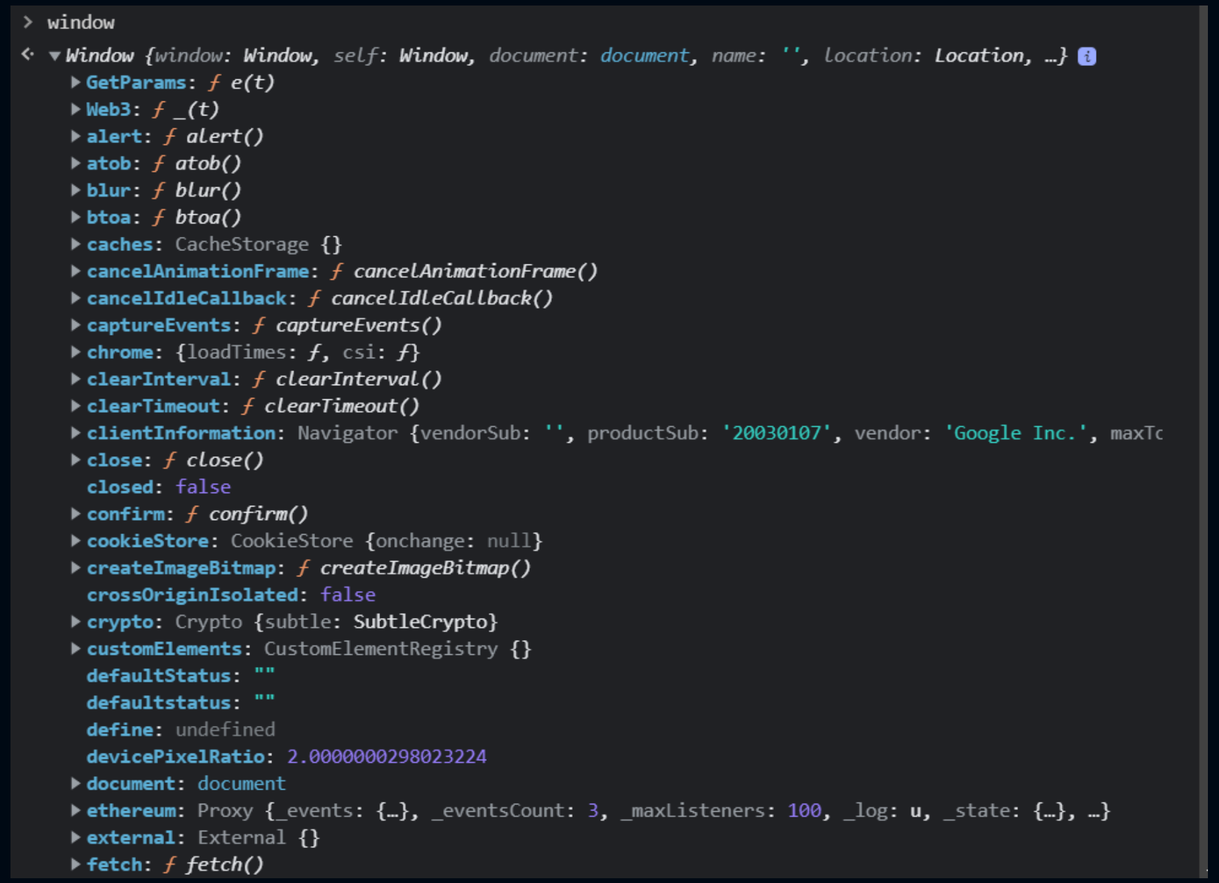
With async –

The file gets downloaded asynchronously along with HTML Parsing but executed as soon as it’s downloaded.
So, the order of execution is not fixed.



The Window object

Window object - Top of the hierarchy. It is the outmost element of the object hierarchy. Document object - Each HTML document that gets loaded into a window becomes a document object.



Refer

- w3school

Window Console Object

- The console object provides access to the browser's debugging console. The specifics of how it works varies from browser to browser, but there is a de facto set of features that are typically provided.
- The console object can be accessed from any global object. Window on browsing scopes and WorkerGlobalScope as specific variants in workers via the property console. It's exposed as Window.console, and can be referenced as console. For example:
- console.log("Failed to open the specified link")

Console.log()

Outputs a message to the console

Console.error()

Outputs an error message to the console

Console.warn()

Outputs a warning message to the console

Console.clear()

Clears the console

Refer

- w3school
- MDN

Refer : gfg

Variables

Variables are containers for storing data (storing data values).

1

var : The var is the oldest keyword to declare a variable in JavaScript.

Scope: Global scoped or function scoped. The scope of the var keyword is the global or function scope. It means variables defined outside the function can be accessed globally, and variables defined inside a particular function can be accessed within the function.

2

let : The let keyword is an improved version of the var keyword.

Scope: block scoped: The scope of a let variable is only block scoped. It can't be accessible outside the particular block ({block}). Let's see the below example.

3

const : The const keyword has all the properties that are the same as the let keyword, except the user cannot update it.

Scope: block scoped: When users declare a const variable, they need to initialize it, otherwise, it returns an error. The user cannot update the const variable once it is declared.

Refer : MDN

DataTypes in JS

Dynamic typing : JavaScript is a loosely typed and dynamic language. Variables in JavaScript are not directly associated with any particular value type, and any variable can be assigned (and re-assigned) values of all types:

Dynamic Typing

```
let foo = 42;    // foo is now a number
foo     = 'bar'; // foo is now a string
foo     = true;  // foo is now a boolean
```

Types

The set of types in the JavaScript language consists of primitive values and objects.

- Primitive values (immutable datum represented directly at the lowest level of the language)
  - Boolean type
  - Null type
  - Undefined type
  - Number type
  - BigInt type
  - String type
  - Symbol type
- Objects (collections of properties)

Strings in JS

Refer : w3shools

string.length

```
let txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
let length = txt.length;
>> 26
```

string.slice

```
let str = "Apple, Banana, Kiwi";
let part = str.slice(7, 13);
>>  Banana
```

string.substr

```
let str = "Apple, Banana, Kiwi";
let part = str.substr(7, 6);
>> Banana
```

string.replace

```
let text = "Hello Earth!";
let newText = text.replace("Earth", "World");
>> Hello World
```

string.concat

```
let text1 = "Hello";
let text2 = "World";
let text3 = text1.concat(" ", text2);
>> Hello World!
```

Ab aage khud se explore kar lena baaki ke strings function.....sab mai hi padha doonga to tum log kya karoge!

Template literals in JS

Template literals (template strings) are literals delimited with backtick (`) characters, allowing for multi-line strings, for string interpolation with embedded expressions, and for special constructs called tagged templates.

Quotes Inside Strings

```
let text = `He's often called "Ronny"`;
>> He's often called "Ronny"
```

Refer : w3schools

Multiline Strings

```
let text =
`The quick
brown fox
jumps over
the lazy dog`;
>> The quick
 brown fox
jumps over
the lazy dog
```

Variable Substitutions

```
let firstName = "John";
let lastName = "Doe";
let text = `Welcome ${firstName}, ${lastName}!`;
>> Welcome John, Doe!
```

Expression Substitution

```
let price = 10;
let VAT = 0.25;
let total = `Total: ${(price * (1 + VAT)).toFixed(2)}`;
>> Total: 12.50
```

Refer : W3Schools

Objects in JS

Objects are variables too. But objects can contain many values

const car = {type:"Fiat", model:"500", color:"white"};

const person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

Object Methods

```
const person = {
 firstName: "John",
 lastName : "Doe",
 id      : 5566,
 fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Array in Objects

```
const person = {
 firstName: "John",
 lastName : "Doe",
 id : 5566,
 name: ['John', 'Doe']
};
```

Arrays in JS

Refer : MDN

- const cars = ["Saab", "Volvo", "BMW"];

- const cars = [];
  - cars[0]= "Saab";
  - cars[1]= "Volvo";
  - cars[2]= "BMW";

- const cars = new Array("Saab", "Volvo", "BMW");

Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.
But, JavaScript arrays are best described as arrays.

Objects in Array

foo = ['hello', 20, {f: "a", s: "b"}]

JavaScript variables can be objects. Arrays are special kinds of objects.
Because of this, you can have variables of different types in the same Array.
You can have objects in an Array. You can have functions in an Array. You can have arrays in an Array:

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

Array Methods

```
cars.length   // Returns the number of elements
cars.sort()   // Sorts the array
```

Destructuring assignment

Refer : MDN

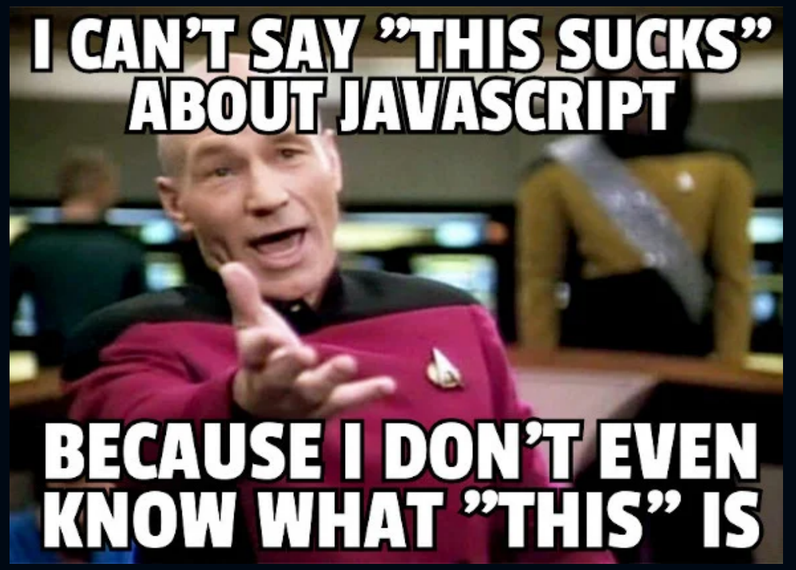
```
let a, b, rest;
[a, b] = [10, 20];

console.log(a);
// expected output: 10


console.log(b);
// expected output: 20

[a, b, ...rest] = [10, 20, 30, 40, 50];

console.log(rest);
// expected output: Array [30,40,50]
```

```
var colors = {
    color1: 'red',
    color2: 'blue',
    color3: 'yellow'
};

// Before ES6:
var color1 = colors.color1;
var color2 = colors.color2;
var color3 = colors.color3;

console.log(color1, color2, color3); // 'red', 'blue', 'yellow'

// After ES6:
var {color1, color2, color3} = colors;

console.log(color1, color2, color3); // 'red', 'blue', 'yellow'
```

This Keyword in JS

Refer : MDN

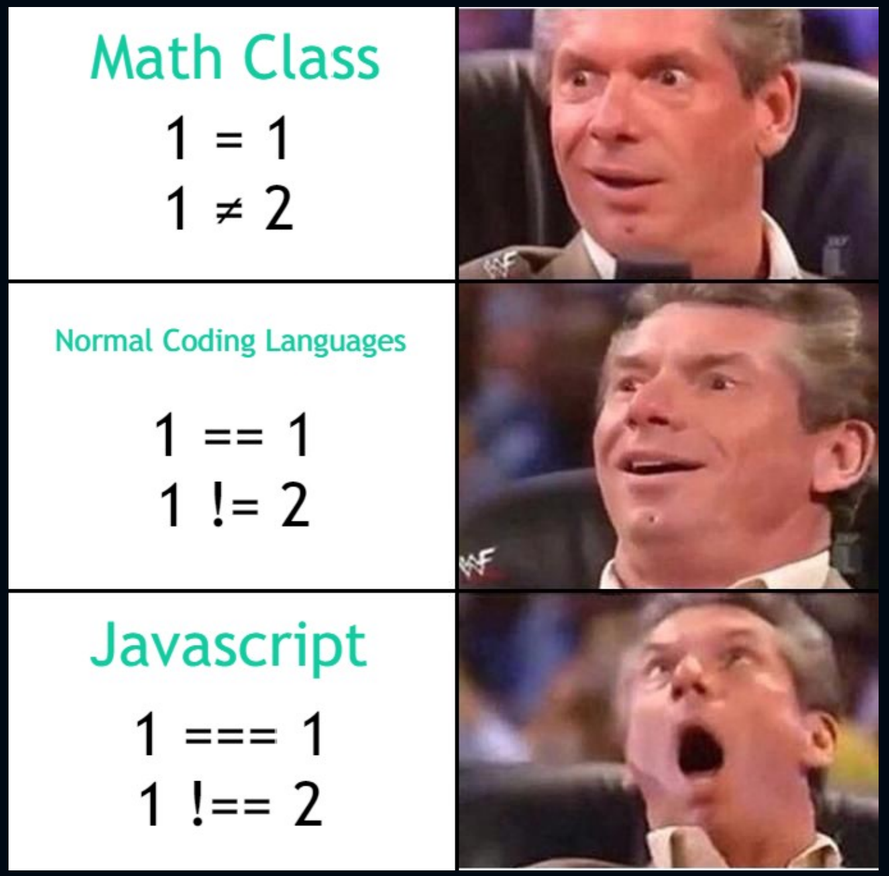
In JavaScript, the 'this' keyword refers to an object.
Which object depends on how this is being invoked (used or called).
The this keyword refers to different objects depending on how it is used:



- In an object method, this refers to the object.
- Alone, this refers to the global object.
- In a function, this refers to the global object.
- In a function, in strict mode, this is undefined.
- In an event, this refers to the element that received the event.
- Methods like call(), apply(), and bind() can refer this to any object.

Comparison and Logical Operators JS

- equal to
- equal value and equal type
- not equal
- not equal value or not equal type
- greater than
- less than
- greater than or equal to
- less than or equal to



Refer : W3Schools

Conditional Statements

Refer : MDN

- In JavaScript we have the following conditional statements:
- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

Syntax

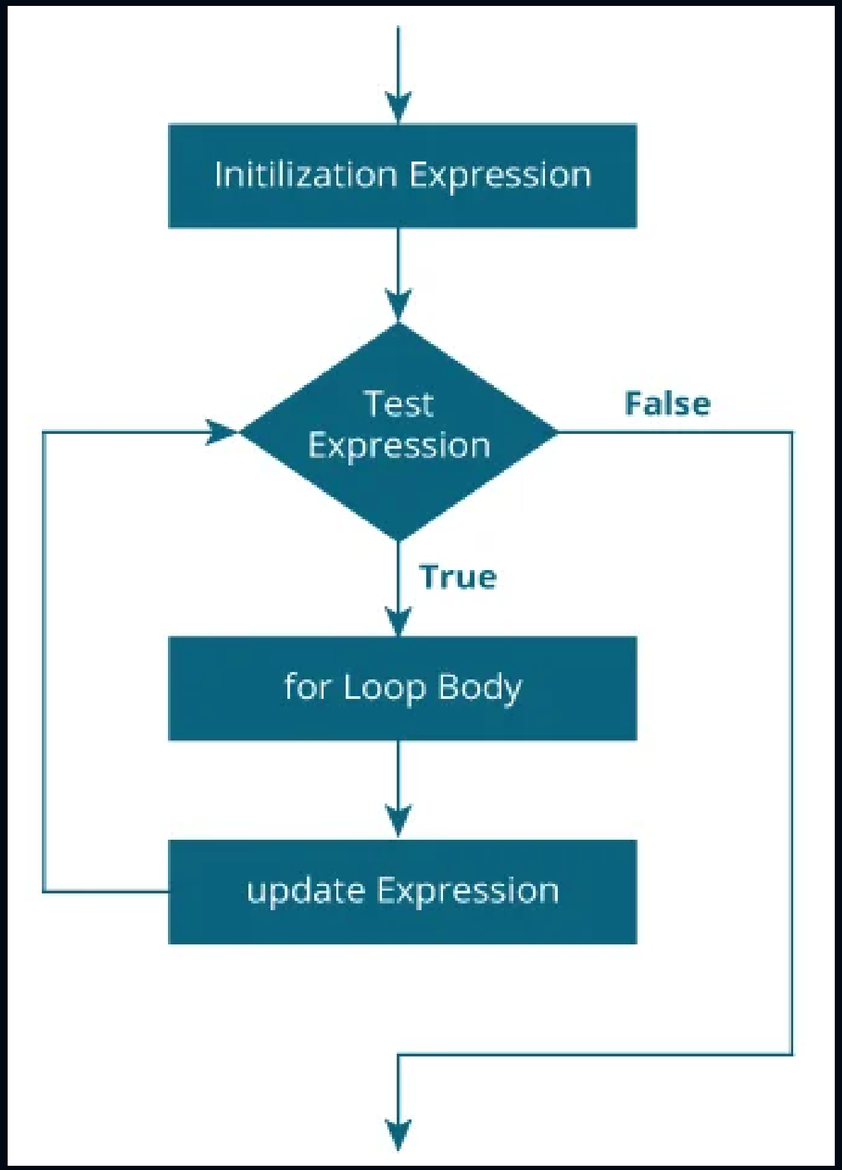
```
if (condition1) {
  //  block of code to be executed if condition1 is true
} else if (condition2) {
  //  block of code to be executed if the condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the condition1 is false and condition2 is false
}
```

Loops

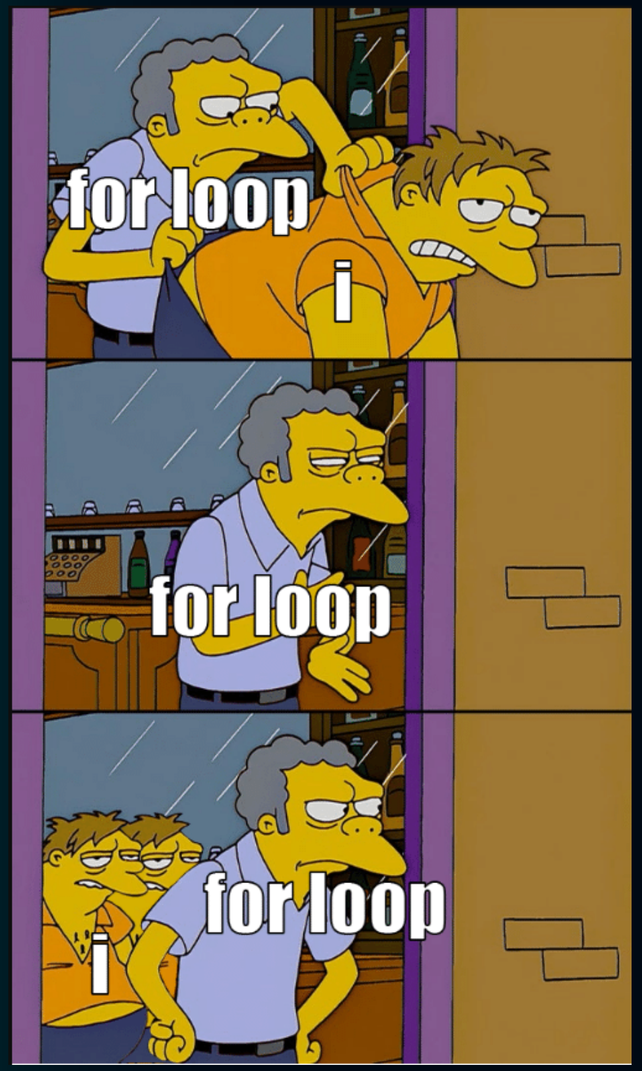
JavaScript supports different kinds of loops:

- for - loops through a block of code a number of times
- for/in - loops through the properties of an object
- for/of - loops through the values of an iterable object
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

Refer : MDN



for loops



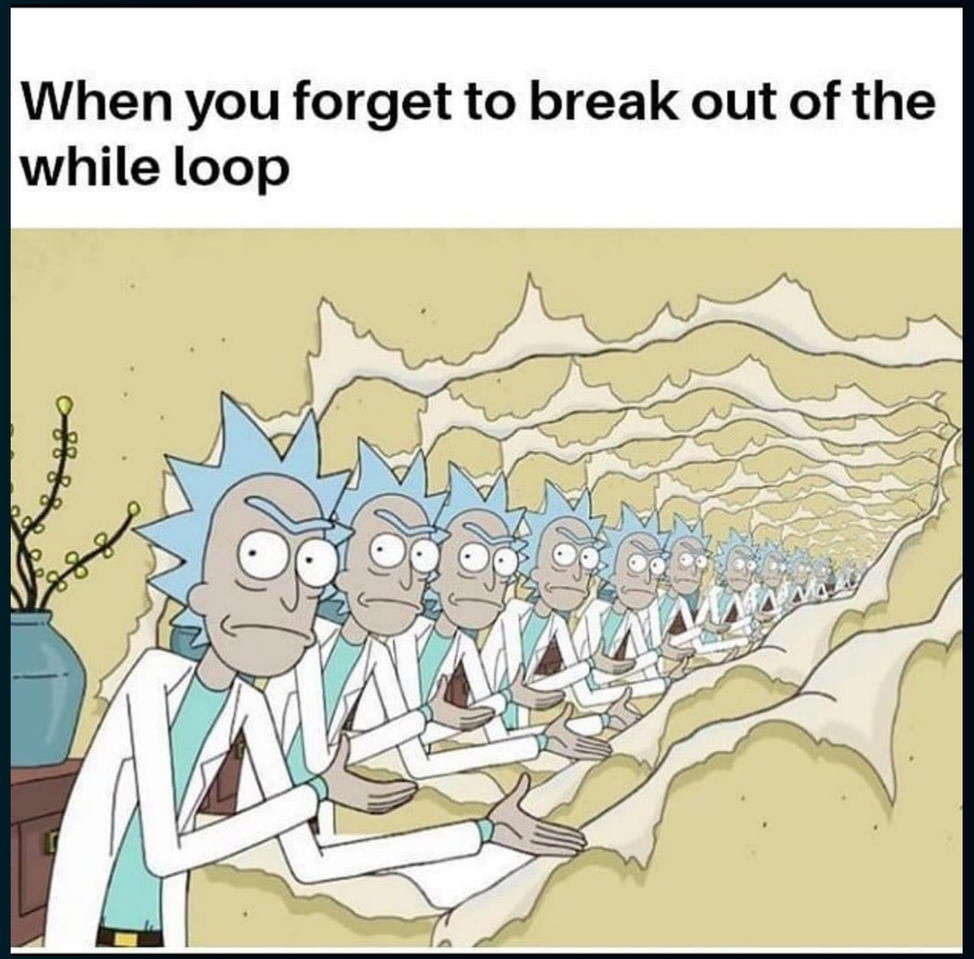

```
const numbers = [45, 4, 9, 16, 25];

for (let i = 0; i < numbers.length; i++) {
 console.log(numbers[i]);
}
```

```
for (let x in numbers) {
  console.log(numbers[x]);
}
```

```
numbers.forEach(myFunction);

function myFunction(value) {
 console.log(value)
}
```

```
for(let x of numbers){
    console.log(x);
}
```

while loops




```
const numbers = [45, 4, 9, 16, 25];

while( i < numbers.length){
  console.log(numbers[i]);
  i++;
}
```

```
do {
 console.log(numbers[i]);
}
while (i < numbers.length);
```

Functions in JS

Refer : MDN

A JavaScript function is a block of code designed to perform a particular task.
A JavaScript function is executed when "something" invokes it (calls it).

```
let x = square(4);   // Function is called, return value will end up in x

function square(a) {
  return a * a;           // Function returns the product of a and b
}
```

Why Functions?

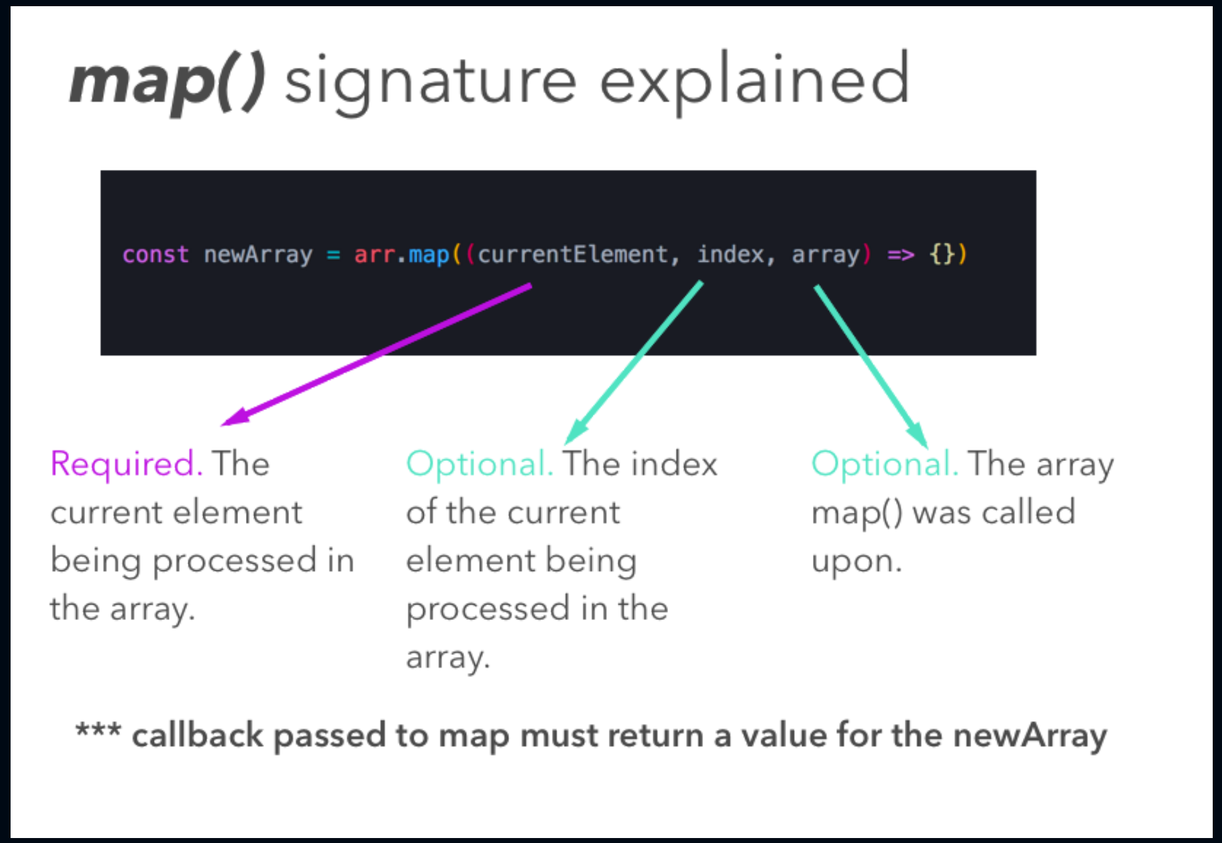
You can reuse code: Define the code once, and use it many times.
You can use the same code many times with different arguments, to produce different results.

JavaScript Array map()

Refer : MDN

The map() method creates a new array populated with the results of calling a provided function on every element in the calling array.

```
const numbers = [65, 44, 12, 4];
const newArr = numbers.map(myFunction)

function myFunction(num) {
  return num * 10;
}

console.log(numbers);
>> [65,44,12,4]
console.log(newArr);
>> [650,440,120,40]
```

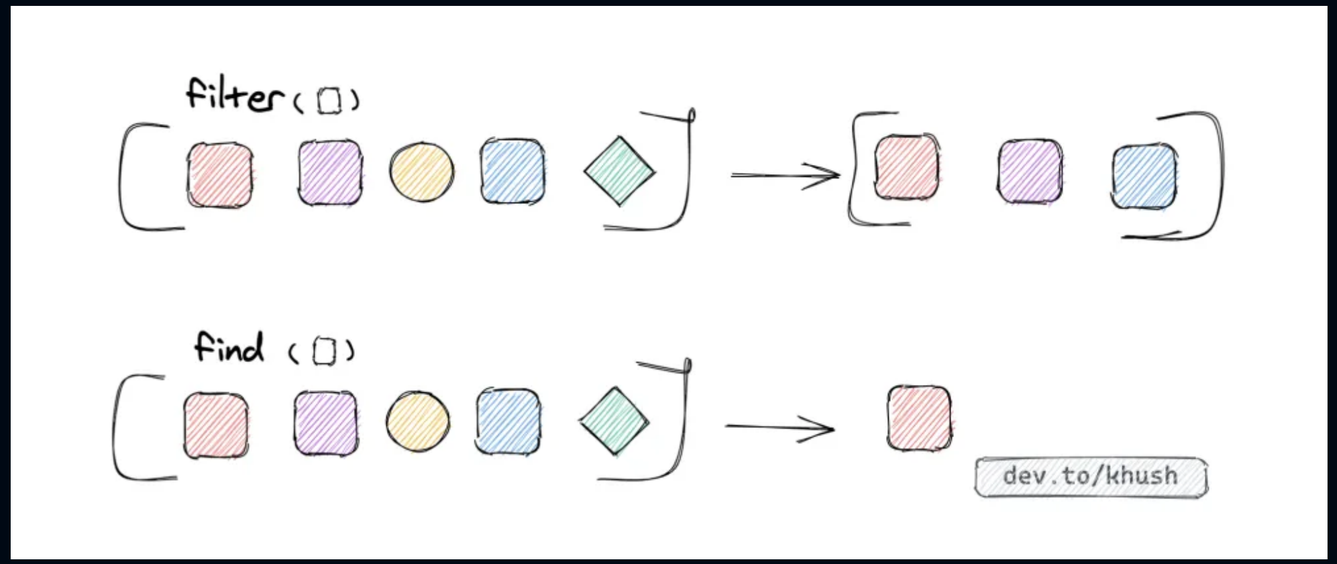


JavaScript Array filter()

Refer : MDN

The filter() method creates a shallow copy of a portion of a given array, filtered down to just the elements from the given array that pass the test implemented by the provided function.

```
const words = ['spray', 'limit', 'elite',
'exuberant', 'destruction', 'present'];

const result = words.filter(word =>
word.length > 6);

console.log(result);
>>["exuberant", "destruction", "present"]

console.log(words);
>>["spray", "limit", "elite", "exuberant",
"destruction", "present"]
```

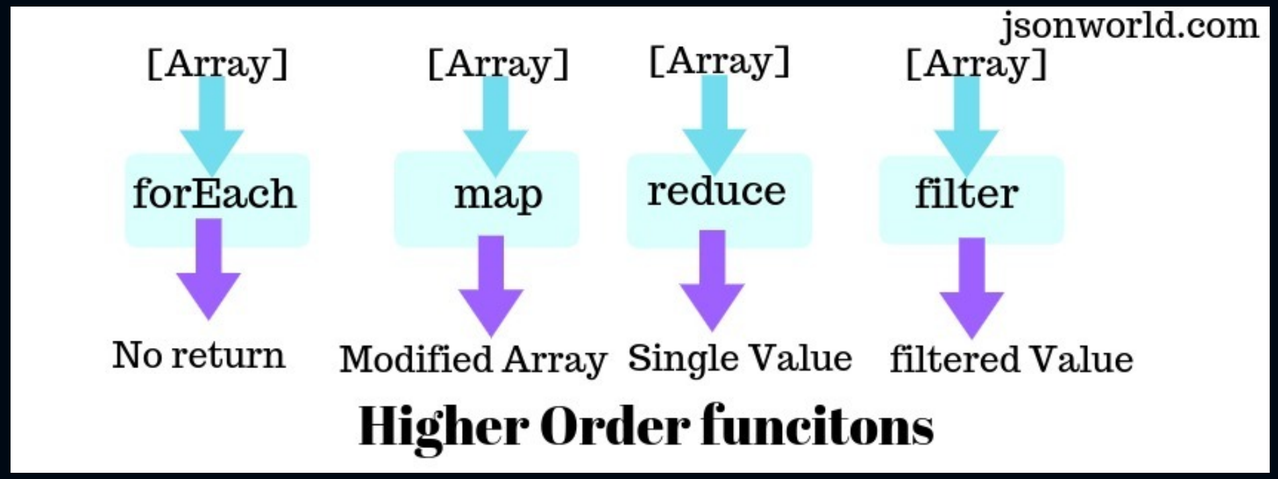


JavaScript Array reduce()

Refer : MDN

The reduce() method executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value.

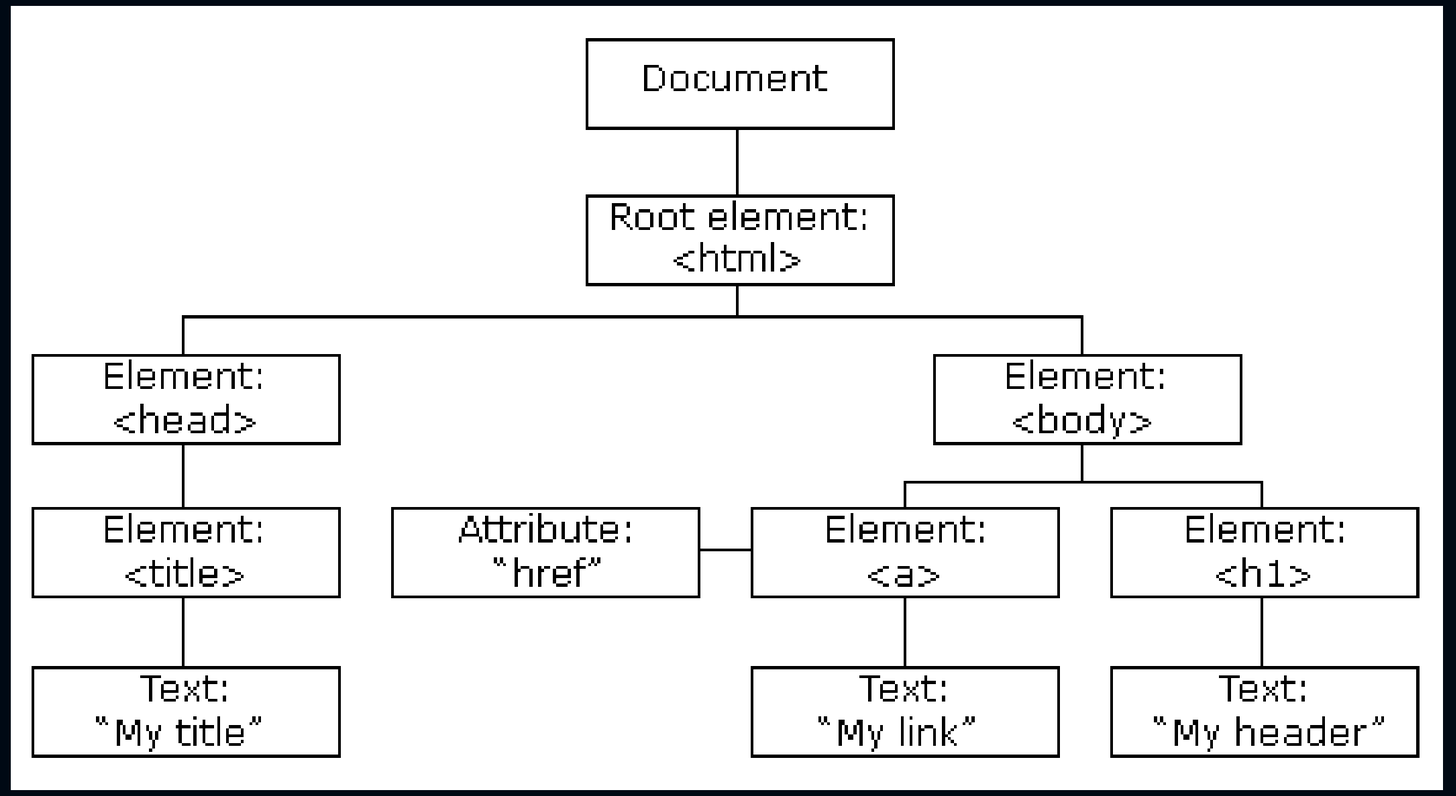
```
const numbers = [15.5, 2.3, 1.1, 4.7];

let sum = numbers.reduce(getSum, 0);

function getSum(total, num) {
  return total + Math.round(num);
}

console.log(sum);
>> 24
```



DOM

Refer : GFG

When a web page is loaded, the browser creates a Document Object Model of the page.
The HTML DOM model is constructed as a tree of Objects:



DOM Events in JS

Refer : MDN

Javascript events provides functionalities to HTML elements when something is invoked in the form of mouse click, scroll, key press, etc.

- document.createElement
- document.getElemetById
- document.getElementByClassName
- document.appendChild
- document.querySelector
- document.querySelectorAll
- element.innerHTML
- element.setAttribute

and many more....

Classes in JS

Refer : MDN

Use the keyword class to create a class.
Always add a method named constructor():

```
class Car {
 constructor(name, year) {
 this.name = name;
 this.year = year;
 }

age(x) {
    return x - this.year;
 }
}

const myCar = new Car("Ford", 2014);
```