# JavaScript

- **Create Js file and run it browser and with nodeJs**
- **Declare variable**
  - var

```
var ename;
ename = 'Smith';
console.log(ename);
var job = 'Manager';
```

  - Declare many variable

```
var ename = 'King',
    age = 50,
    job = 'President';
ename = 9999;
```

  - Different between undefined and not defined

```
var pname;
console.log(pname);
console.log(hname);
```

  - Local and global variable

```
function greet() {
    let message = 'Hello';
    console.log(message)
}
console.log(message)
```

  - How JS see variables

```
console.log(ename)
var ename;
ename = 'ali';
```

  - Different between let, var, const

```
{
    let name = 'ali'
}
console.log(name)
const name = 'ali'
name = 'hamid'
```

- data types
  - JS is weak data type checking language
  - Numbers, Strings, Booleans, null, undefined, Object

- **Numbers**

```
console.log( parseInt('100px') );
console.log( parseFloat('21.5$') );
console.log(Math.floor(15.8));
console.log(Math.ceil(15.8));
console.log(Math.round(15.8));
console.log(Math.trunc(15.8)); integer part of number
console.log(Math.random());
console.log(Math.max(7, -10, 0));
console.log(Math.min(7, -10, 0));
console.log(Math.pow(2, 10));
```

- **Strings**
  - Define

```
let str = 'Test'
```

  - String length

```
console.log(str.length);    // 4
```

  - Display value

```
console.log(str.valueOf());
console.log(str.toString());    // Test
```

  - Access string's index

```
console.log(str[0]);    // T
console.log(str.charAt(1));     // e
```

  - Concat strings

```
let s1 = "Paul ";
let s2 = "Miller";
let s3 = s1.concat(s2); // s1 + s2;
```

- SubStrings

```
// substr(startIndex, length)
str = "Chrome Browser";
console.log(str.substr(0, 8));      // Chrome B
console.log(str.substr(8, 4));      // rows

// substring(startIndex, endIndex)
str = "Chrome Browser";
console.log(str.substring(2, 13));  // rome Browse
```

- Some useful methods

```
console.log(str.toLowerCase());   // smith
console.log(str.toUpperCase());   // SMITH
// string.indexOf(substring, fromIndex);
str = "Chrome Browser";
console.log(str.indexOf('ro', 1));      // 2
console.log(str.indexOf('ro'));         // 2
console.log(str.indexOf('Google'));     // -1
// string.lastIndexOf(substring, fromIndex);
str = "Chrome Browser";
console.log(str.lastIndexOf('ro', 6));      // 2
console.log(str.lastIndexOf('ro'));         // 8
console.log(str.lastIndexOf('or'));         // -1

let greet = '   Hello ';
console.log(greet.trim());      // Hello

str = 'USE Chrome Browser';
let pos = str.search(/se/);
console.log(pos);       // 15

// str.replace(substr, newSubstr)
let url = 'abc.com/bird=fly';
let newUrl = url.replace(/bird/g, 'pigeon');

console.log(newUrl);
console.log(url);
```

- Template using ``

```
let msg3 = `This
is
multiline`;
console.log(msg3);
```

- Template using ${}

```
let greet = `Welcome, ${firstName} ${lastName} !!!`;
```

- **Operators**

```
console.log(10 / 2);    // 5
console.log(10 % 7);    // 3
console.log(2 ** 2);    // 4
var x = 3;
console.log( ++x ); // --
var y = 8;
console.log( y++ );
var x = 10;
console.log(-x);
var m = 10, n = 7;
console.log( m - n );
var str = 'alpha' + 'bet';
console.log(str);
console.log( '10' + 20 );   // 1020
console.log( 20 + '10' );   // 2010
console.log( 3 + 3 + '2' ); // ASK => 62
```

- **Condition Expression**

```
console.log(50 > 20);   // true
console.log(50 == 20);  // false
console.log(50 == 50);  // true
console.log(50 != 20);  // true
console.log('Z' > 'A');     // true
console.log('Tony' > 'Tiny');   // true
console.log('6' > 3); // ASK => true Because strings convert to integer
console.log('02' == 2); // ASK => true
console.log(true == 1);     // true
console.log(false == 0);    // true
console.log('' == false);   // true

console.log(0 === false);       // false
console.log('' === false);      // false
console.log('' !== false);      // true

console.log(false && false);    // false
console.log(false && true);     // false
console.log(true && false);     // false
console.log(true && true);      // true
```

```
var isAge = 30, isEyeSight = 0.75;
if (isAge >= 18 && isEyeSight >= 0.5) {
    console.log('Issue Driving Licence');
} else {
    console.log('Un-Authorized Candidate');
}

console.log(false || false);    // false
console.log(false || true);     // true
console.log(true || false);     // true
console.log(true || true);      // true

var x = 10, y = 30;
if ((x > 20) || (y > 20)) {
    console.log('Either x or y is greater than 20');
}

console.log( !true);    // false
console.log( !0 );      // true
console.log( !!true );  // console.log( true );
```

- **One line condition**

```
condition ? whenIsTrue : whenIsFalse
```

- **Else if**

```
if (testScore >= 90) {
    console.log('Grade A');
} else if (testScore >= 70) {
    console.log('Grade B');
} else if (testScore >= 50) {
    console.log('Grade C');
} else {
    console.log('Grade D');
}
// ASK
var grade = (testScore >= 90) ? console.log('Grade A') :
                (testScore >= 70) ? console.log('Grade B') :
                (testScore >= 50) ? console.log('Grade C') :
                            console.log('Grade D');
```

- **Switch case**

```
let grade = 'B';
switch (grade) {
    case 'A':
        console.log('Excellent');
        break;
    case 'B':
        console.log('Well Done');
        break;
    case 'C':
        console.log('Just Passed');
        break;
    default:
        console.log('Better Try Again');
}
let animal = 'Tiger';
switch (animal) {
    case 'Lion':
    case 'Tiger':
        console.log('Wild Animals.');
        break;
    case 'Sheep':
    default:
        console.log('Domestic Animals');
}
```

- **While**

```
var count = 0;
var total = 0;
while (count < 3) {
    count++;
    total += count;     // total = total + count;
    console.log(count + '\t\t' + total);
}
```

- **Do while**

```
var count = 0;
do {
    count += 1;     // count = count + 1;
    console.log(count);
} while (count < 5);
```

- **For**

```
for (let count = 0; count < 3; count++) {
    console.log('Value: ' + count);
}
```

- **Break**

```
for (var count = 1; count < 10; count++) {
    if (count % 5 == 0) {
        break;
    }
    console.log('Value: ' + count);
}
```

- **Continue**

```
for (var count = 1; count < 10; count++) {
    if (count % 2 == 0) {
        continue;
    }
    console.log('Value: ' + count);
}
```

- **Named Loop**

```
markLoop:
for (var count = 0; count < 3; count++) {
    for (var num = 0; num < 5; num++) {
        if (num == 1) {
            continue markLoop;
        }
        console.log(count + '\t\t' + num);
        if (num == 2) {
            break markLoop;
        }
    }
}
console.log('Completed !!!');
```

- **Functions**
  - Define

```
function addition(num1, num2) {
    var sum = num1 + num2;
    return sum;
}
var result = addition(10, 5);
console.log(result);        // 15
```

  - Access arguments

```
function addition() {
    var sum = 0;
    for (var i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;     // 15
}
var result = addition(10, 5);
console.log(result);        // 15
```

  - Execute without call

```
(function greet() {
    console.log('gg')
}) ();
```

  - Arrow functions

```
const arrowFunc = (x, y) => {
    return x + y;
}
```

  - Default value

```
function sum(x = 5, y = 2) {
    return x + y;
}
```

- Bind

```
function func() {
    console.log(this.x)
}
const obj = {
    x: 10
}
const newFunc = func.bind(obj)
newFunc()
```

- Rest params

```
function sum(...args) {
    console.log('Parameter\'s length: ' + args.length);
    console.log('Parameters: ' + args);
    let total = 0;
    for(let count = 0; count < args.length; count++) {
        total = total + args[count];
    }
    return total;
}
console.log('Addition = ' + sum(10, 20, 30));
```

- **Arrays**
  - Define

```
var colors = new Array();
var depts = [];
var fruits = ['Apple', 'Banana', 'Mango'];
```

  - Access array member

```
console.log(fruits[0]);
console.log(fruits[1]);
console.log(fruits[2]);
```

  - Change array

```
fruits[1] = 'Guava';
```

  - Array length

```
console.log(fruits.length);     // 4
```

  - Display array members

```
console.log(fruits.toString());
console.log(fruits.valueOf());
console.log(fruits.join(' $ '));
console.log(fruits.join(' - '));
console.log(...[1,2,3]) // ES6
```

  - Array with different member data type

```
var mixArr = [    {ename: 'Smith'},
                  false,
                  function(){console.log('Hello')},
                  'Apple'
];
console.log(mixArr[0].ename);   // Smith
mixArr[2]();
```

- Two demonical array

```
var cubes = [
    [1, 2, 3],
    [4, 5, 6]
];
for (var i = 0; i < cubes.length; i++) {
    for (var j = 0; j < cubes[i].length; j++)
        console.log('cube[' + i + '][' + j + ']  = ' + cube[j]);
}
```

- Inset in array

```
fruits[3] = 'Coconut';
// StackTori
var stack = [];
stack.push(1);
console.log(stack);     // [1]
// QueueTori
var queue = [];
queue.unshift(1]);
```

- Delete from array

```
console.log(stack.pop());
console.log(stack);     // [1,2]
queue.shift();
```

- Sort array

```
var depts = ['Sales', 'IT', 'R&D', 'Management'];
depts.sort();
console.log(depts); // ['IT', 'Management', 'R&D', 'Sales']
```

- Insert and Delete from index of array

```
// spilce(start, delectCount, …items)
var months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun'];
var deleteMonths = months.splice(0, 4);
console.log(months);    // [May, Jun]
console.log(deleteMonths);  // [Jan, Feb, Mar, Apr]
months.splice(2, 0, 'Sep', 'Oct');
console.log(months);    // [Jul, Aug, Sep, Oct, Nov, Dec];
var depts = ['R&D', 'Sales', 'IT'];
depts.splice(1, 1, 'HR', 'Finance');
console.log(depts);     // [R&D, HR, Finance, IT];
```

- Copy array

```
// arr.slice(start, stop);
var depts = [10, 20, 30, 40, 50];
var newDepts = depts.slice(0, 3);
console.log('depts: ' + depts); // [10, 20, 30, 40, 50]
console.log('newDepts: ' + newDepts);   // [10, 20, 30]
```

- Find index of an element

```
// arr.indexOf(searchElement, fromIndex);
var depts = [40, 30, 20, 40, 20];
console.log(depts.indexOf(30));     // 1
console.log(depts.indexOf(40, 2));  // 3
console.log(depts.indexOf(20));     // 2
console.log(depts.indexOf(50));     // -1
// ASK
// locate all indexes of an element
function locateAll(element, arr) {
    var results = [];
    var idx = arr.indexOf(element);
    while (idx != -1) {
        results.push(idx);
        idx = arr.indexOf(element, idx + 1);
    }
    return results;
}

var depts = [40, 30, 20, 40, 20];
console.log(locateAll(20, depts));  // [2, 4]
```

- Find last index

```
// arr.lastIndexOf(searchElement, fromIndex);
var depts = [40, 30, 20, 40, 20];
console.log(depts.lastIndexOf(40)); // 3
console.log(depts.lastIndexOf(30)); // 1
console.log(depts.lastIndexOf(20)); // 4
console.log(depts.lastIndexOf(50)); // -1
```

- Concat arrays

```
// concat()
var alpha = ['a', 'b'];
var num1 = [1, 2];
var num2 = [3, 4];
var alphaNumeric = alpha.concat(num1, num2, 5, [6, 7]);
console.log(alphaNumeric);  // [a, b, 1, 2, 3, 4, 5, 6, 7]
console.log([...num1, ...num2]) // ES6
```

- Search in array

```
// find()
var gadgets = [
    {name: 'Apple', quantity: 3},
    {name: 'Nokia', quantity: 0},
    {name: 'Mi', quantity: 7}
];
function isMi(mobile) {
    return mobile.name === 'Mi';
}
var result = gadgets.find(isMi);
// var result = gadgets.find(mobile => mobile.name === 'Mi');
console.log(result);    // {name: 'Mi', quantity: 7}

// findIndex()
function isMi(mobile) {
    return mobile.name === 'Mi';
}
var result = gadgets.findIndex(isMi);
// var result = gadgets.findIndex(mobile => mobile.name === 'Mi');
console.log(result);    // 2
```

- Filter array

```
// filter()
var languages = ['java', 'oracle', 'javascript', 'kotlin', 'hibernate'];
function desiredLanguage(language) {
    return language.length > 6;
}
var result = languages.filter(desiredLanguage);
console.log(result);
```

- Reverse array

```
// reverse()
var array1 = [3, 4, 5, 6, 7];
var reversed = array1.reverse();
console.log(array1);
console.log(reversed);   // [7, 6, 5, 4, 3]
```

- Iterate on array's members

```
// map()
var arr = [1, 4, 9, 16], result = [], mutiply = 0;
for (var i = 0; i < arr.length; i++) {
    mutiply = arr[i] * arr[i];
    result.push(mutiply);
}
console.log(result);    // [1, 16, 81, 256]

function square(num) {
    return num * num;
}
var result = arr.map(square);
// var result = arr.map(num => num * num);
console.log(result);    // [1, 16, 81, 256]

// forEach()
var grades = ['a', 'b', 'c'];
for (var i = 0;i < grades.length; i++) {
    console.log(grades[i]);
}
var grades = ['a', 'b', 'c'];
grades.forEach(function(element) {
    console.log(element);
});

// for...of
for (let grade of grades) {
    console.log(grade);
}
```

- **Map**
  - Define

```
let departements = new Map([
    [30, 'SALES'],
    [40, 'R&D'],
    [50, 'Finance']
]);

for (let [key, value] of departements) {
    console.log(`Key: ${key} & Value: ${value}`);
}
```

- **Objects**
  - Define

```
let person = {
    firstName: 'ali',
    lastName: 'modiri',
    fullName() {
        return this.firstName + ' ' + this.lastName
    }
}
console.log(person.fullName())
```

  - Delete propery

```
delete person.firstName
```

  - Loop on object

```
for(let property in person) console.log(property)
```

  - Function instead of class

```
function person(fname, lname) {
    return {
        firstName: fname,
        lastName: lname,
        fullName: function() {
            return this.firstName + ' ' + this.lastName;
        }
    }
}
let person1 = person('ali', 'modiri');
```

  - Add new property

```
let person = {
    fname: 'ali',
    lname: 'modiri'
}
person.age = 10;
let newPreson = {
    age: 10,
    ...person
}
console.log(newPreson)
```

- Select properties from object

```
const person = {
    nameInfo: {
        firstName: 'ali',
        lastName: 'modiri'
    },
    age: 10
}

const {nameInfo: {firstName, lastName}, age} = person

console.log(firstName)
```

- **Error handling**

```
try {
    console.log('Try: Starts');
    test();
    console.log('Try: Ends');
} catch(error) {
    console.log(error.name + ': ' + error.message);
} finally {

    console.log('finally block');
}
```

- **Classes**
  - Define

```
class Person {
    constructor(fname, lname) {
        this.firstName = fname;
        this.lastName = lname;
    }

    fullName() {
        console.log(`fullName: ${this.firstName} ${this.lastName}`);
    }
}
let person1 = new Person('ali', 'modiri');
console.log(person1);
person1.fullName();
```

  - Get and set

```
class Person {

    firstName;
    set firstName(fname) {
        this.firstName = fname
    }
    get firstName() {
        return this.firstName
    }
}

let person1 = new Person()
console.log(person1.firstName)
person1.firstName = 'ali'
console.log(person1.firstName)
```

  - Static method

```
class ClassWithStaticMethod {
    static staticMethod() {
        return 'static method has been called.';
    }
}
console.log(ClassWithStaticMethod.staticMethod());
```

- Inheritance

```
class Animal {
    foo = () => console.log('foo')
}
class Dog extends Animal {
    bark = () => console.log('bark')
}
let dog = new Dog()
dog.foo()
dog.bark()

class Animal {
    constructor(name) {
        this.animalName = name;
    }
    foo = () => console.log(`foo from ${this.animalName}`)
}
class Dog extends Animal {
    constructor(aname) {
        super(aname)
    }
    bark = () => console.log('bark')
}
let dog = new Dog('poppy')
dog.foo()
dog.bark()

class Animal {
    bar() {console.log('bar')}
}
class Dog extends Animal {
    fromParent() {
        console.log('from child')
        super.bar()
    }
}
let dog = new Dog()
dog.fromParent()
```

- **Strict Mode**
  - Throw error

```
'use strict'

x = 10;
```

  - It's ok

```
x = 10;
```

- **Promises**

```
let check = new Promise((resolve, reject) => {
    let isComplete = true;
    if(isComplete) {
        resolve('Delivered.');
    } else {
        reject('Not Delivered.');
    }
});

check.then(function(fromResOrRej) {
    console.log('Project is: ' + fromResOrRej);
}).catch(function(fromResOrRej) {
    console.log('Project is: ' + fromResOrRej);
});

const request = () => {
    return new Promise((resolve, reject) => {
        setTimeout(() => resolve('Result'), 2000)
    })
}

const result = request()
    .then((respones) => {
        console.log(respones)
    })
    .catch((err) => {
        console.log(err)
    })
```

- Promise.all

```
let promise1 = Promise.resolve(3);
let promise2 = 42;
let promise3 = new Promise(function(resolve, reject) {
    setTimeout(resolve, 10000, 'foo');
});

Promise.all([promise1, promise2, promise3]).then(function(values) {
  console.log(values);
});
```

- **Async / await**

```
const delayFunc = () => {
    return new Promise((resolve, reject) => {
        setTimeout(() => resolve('message'), 1000)
    })
}

async function api() {
    const response = await delayFunc()
    console.log(response)
}

api()
```

- **Modules**

```
let message = 'Hey Guys !';
function getMsg() {
    return message;
}
function setMsg(msg) {
    message = msg;
}
export {message, getMsg, setMsg};
```

```
import { message, getMsg, setMsg } from './utils.mjs'
const mes = getMsg()
console.log(mes)
setMsg('new')
console.log(message)
console.log(getMsg())
```

```
import * as utils from './utils.mjs'
const mes = utils.getMsg()
console.log(mes)
utils.setMsg('new')
console.log(utils.message)
console.log(utils.getMsg())
```

```
export let message = 'Hey Guys !';
export function getMsg() {
    return message;
}
```

```
const message = 'hello'
export default message
```

```
import utils from './utils.mjs'

console.log(utils)
```