**New Javascript codes**

document.getElementById("divParent") equal->  divParent.onlick

x == "someValue" &&  alert('hi')

**JavaScript** is an interpreted **language**, it is read **line by line**

**JavaScript Filter ()**

const words = [

"spray",

  "limit",

  "elite",

  "exuberant",

  "destruction",  
 "present"];

const result = words.filter(word => word.length > 6);  
console.log(result);  
*// expected output: Array ["exuberant", "destruction", "present"]*

**Match()**

var str = "The rain in SPAIN stays mainly in the plain";

var res = str.match(/ain/gi);

console.log(res);

*// expected output: Array ["ain", "AIN", "ain", "ain"]*

**Splice to Add or Remove element to Array**

Array.splice(start , deleteCount , addItem1, addItem2)

Note: deleteCount starts from 1, if 0 will not delete items

Note: start with minus => count from right side

Array.splice(0, 1) => Remove from 0 index one Item

Array.splice(1, 2, “name”) => Remove from 1 index two Items and insert name

Array.splice(-1, 1) => Remove last one item

Array.splice(-1, 1, ’ashta’) => Remove last Item and add ashta

Array.splice(Array.length, 0, ’ashta’) => Add ashta at the end of array

Array.splice(0, 0, ’ashta’) => Add ashta at the Begin of array

Array.splice(2) => Remove all elements after 2 index

Array.splice(0, Array.length) => Remove all elements

Array.splice(0, Array.length, ‘ashta’) => Remove all elements & add ashta

**Array.slice(begin, end) => Remove Elemnts From Array**

Note: begin start counting from 0

Note: end start counting from 1

Note: end with minus => count from right side

Array = ["Banana", "Orange", "Lemon", "Apple", "Mango"]

Array.slice(0, 2) => [Banana, Orange]

Array.slice(1, 4) => [Orange, Lemon, Apple]

Array.slice(0, -1) => [Banana, Orange, Lemon, Apple]

Array.slice(0, fruits.length)=> Get all Array

Array.slice(0, -fruits.length)=> Remove all Array

**Javascript event delegation**

**المشكلة:** لما بعمل اي event مثلا click علي element جديد اتضاف بعد ال load مش هيعمل الاكشن

**الحل Jquery :** استخدام on

**الحل JavaScript :**

document.getElementById("buttonParent") .onclick = function(event){

if (event.target.nodeName == "BUTTON" ) {  alert('hi') }

}

**JavaScript Hoisting**

console.log(x); *// Error because we not define x*

هو ده ال hoisting ان الجافا سكريبت بتاخد اللي انت معرفه تحت وتخليه فوق في ال top script

بتعرف ال  
value = undefined

Function: بتتحط بقيمتها اكنها اتعملها call فوق ف ال top script

console.log(x); *//will print undefined*

var x = 'value';

console.log(x); *//will print value*

حل الموضوع ده اننا نستخدم let, const, use strict

For example

console.log(x); *// Error and stop*

let x = 'value';

console.log(x); *//error too because the complier will stop*

**Generators (ES6)**

بنخزن قيم في functionعن طريق yieldوبنطلعها في خطوات اكنها checkpointsعن طريق next()

function\* genr (){

  yield 'one'

  yield 'two'

  yield 'three'

}

var x = genr()

console.log(x.next())

console.log(x.next())

console.log(x.next())

console.log(x.next()) // will be undefined because no fourth value

console:

{ value:"one", done:**false** }

{ value:"two", done:**false** }

{ value:"three", done:**false** }

{ value:**undefined**, done:**true** }

لوعملنا return بدل yield دي معناها ان ال checkpoints خلصوا وباقي ال value هيطلع undefined وباقي ال done هيطلع true

function\* genr (){

  yield 'one'

  return 'two'

  yield 'three'

}

var x = genr()

console.log(x.next())

console.log(x.next())

console.log(x.next())

console:

{ value:"one", done:**false** }

{ value:"two", done:**true** }

{ value:**undefined**, done:**true** }

Passing Value throw next

function\* genr (){

  var z = yield 'one'

  var y = yield 'two'

  var f = yield 'three'

  return z + y + f

}

var x = genr()

console.log(x.next())

console.log(x.next(5))

console.log(x.next(3))

console.log(x.next(4))

{ value:"one", done:**false** }

{ value:"two", done:**false** }

{ value:"three", done:**false** }

{ value:12, done:**true** }

**LocalStorge**

*//set item*

      localStorage.setItem('key', 'value');

*//get item*

      var cat = localStorage.getItem('key');

*//remove item*

      localStorage.removeItem('key');

*// Clear all items*

      localStorage.clear();

*// get all Data from localStorage*

      const getAllStorge = JSON.stringify(localStorage)

      console.log(JSON.parse(getAllStorge))

*// Local storage store only String so if we need to store Obeject, convert to string*

      window.localStorage.setItem('user', JSON.stringify(person));

*// To get this object as an object*

       JSON.parse(window.localStorage.getItem('user'));

**Pop, Push, Shift, Unshift Array**

Pop & push (the end of an array)

**Pop**:

let cats = ['Bob', 'Willy', 'Mini']

cats.pop(); *// ['Bob', 'Willy']*

**Push**:

let cats = ["Bob"];

  cats.push("Willy"); *// ['Bob', 'Willy']*

  cats.push("Puff", "George"); *// ['Bob', 'Willy', 'Puff', 'George']*

shift & unshift (the beginning of an array)

**Shift**

let cats = ['Bob', 'Willy', 'Mini'];

cats.shift(); *// ['Willy', 'Mini']*

**Unshift**

let cats = ['Bob'];

cats.unshift('Willy'); *// ['Willy', 'Bob']*

cats.unshift('Puff', 'George'); *// ['Puff', 'George', 'Willy', 'Bob']*

**JavaScript Objects**

**Get Object Keys**

const fruits = {

  apple: 28,

  orange: 17,

  pear: 54,

};

const keys = Object.keys(fruits);

console.log(keys); *// [apple, orange, pear]*

**Get Object Values**

const values = Object.values(fruits);

console.log(values); *// [28, 17, 54]*

**Looping through the object**

for (const property in fruits) {

  console.log(`${property}: ${fruits[property]}`);

}

*// Result*

*// "apple: 28"*

*// "orange: 17"*

*// "pear: 54"*

**Delete**

The JavaScript **delete operator** removes a property from an object

const Employee = {

   firstname: 'John',

   lastname: 'Doe'

 };

 console.log(Employee.firstname);

*//  output: "John"*

 delete Employee.firstname;

 console.log(Employee.firstname);

*// expected output: undefined*

*// to make the property undeletable*

 Object.defineProperty(Employee, 'lastname', {configurable: false});

**Array/Object Destructuring**

**Array:**

const names = ["ali", "ashraf"];

const [a, b] = names;

console.log(a,b);

*// a = ali*

*// b = ashraf*

const names = ["ali", "ashraf"];

const [a, b, c = 'muhammad'] = names;

console.log(a,b,c);

*// a = ali*

*// b = ashraf*

*// c = muhammad*

const names = ["ali", "ashraf", "ashta"];

const [a, b, c = "muhammad"] = names;

console.log(a, b, c);

*// a = ali*

*// b = ashraf*

*// c = ashta*

const names = ["ali", "ashraf", "ashta"];

const [a,, b] = names;

console.log(a, b);

*// a = ali*

*// b = ashta*

const numbers = ["1", "2", "3", "4", "5", "6"]

const [a,b, ...rest] = numbers

console.log(a, b, rest);

*// a = 1*

*// b = 2*

*// rest = ["3", "4", "5", "6"]*

*//Merge two arrays*

let one = ["A", "B"];

let two = ["1", "2"];

const three = [ ...one, ...two]

*// OR*

let three = one.concat(two)

console.log(three);

*//["A", "B", "1", "2"]*

**Object:**

const personOne = {

name: "Ashta",

age: 26,

job: "frontend",

addr: {

city: "Cairo",

cont: "Egypt"

}

};

const {name, job} = personOne

console.log(name, job)

*// Ashta frontend*

const personOne = { same the previous };

const {name:firstName, job} = personOne

console.log(firstName, job)

*// Ashta frontend*

const personOne = {same the previous}

const {name:firstName, ...rest} = personOne

console.log(firstName, rest)

*// Ashta Object {age: 26, job: "frontend", addr: Object}*

const personOne = { same the previous };

const {name:firstName, addr:{city}} = personOne

console.log(firstName, city)

*// Ashta Cairo*

**Import & Export**

each file have just one export default

the same file can have export & export default

export => import {name}

export default => name

In source JS file

export const name = 'Muhammad'

export const age = 26

const job = 'frontend'

export default job

In receiving file

import job, {name, age}  from '@/assets/js/someJS'

*// we can import the default we any name*

import wazefa from '@/assets/js/someJS'

*//wazefa = frontend*

*// we can import the non default with another name by using (as)*

import {name as esm, age as omr} from '@/assets/js/someJS'

*//esm = Muhammad, omr = 26*

import \* as allValues from '@/assets/js/someJS'

*//import all the file as object and name it allValues*