from elzeroo course-2021

javascript-book

**all of the reference is only syntax and some notes**

**basics**

**math**

**string**

**Statements**

**loops**

**Function**

**Array**

**high order function**

**object**

**dom**

**bom**

**destructing**

**set data-type**

**regex**

**oop**

**Date**

**json**

**generator**

**modules**

**ajax**

# basics

## Data-type & typeof-operator

**>type of:**

**to defined the data you give it**

**>data-type:**

**{ undefined , null , string , int & float & double (number),NaN ,**

**object ,**

**array**

**}**

**console.log(typeof data you wrote)**

**Ex:**

**console.log(typeof 85) -> number**

## Template-literals

**>let variable =`value` ->**

**>console.log(${variable}**

**Note: you can edit as normal like break or writing anything you want**

**Ex:**

**console.log(`${wizard} ${mans}**

## Assignment-operator

**variable (arthimetic-operator) = value**

**Ex:**

**a +=100 (a=a+100);**

## Unary PlusAnd Negation Operators

**>unary:**

**convert string or any string data to number in +ve way**

**>Negation:**

**convert string or any string data to number in -ve way**

**Ex:**

**console.log(-“10”)**

**console.log(+“10”)**

## Arithmetic-operators

**(+):Addition , (/):Division**

**(-):Subtraction , (%):Modulus**

**(\*):Multiplication , (\*\*):Exponentiation**

**(++):Increment , (--):Decrement**

**Ex:**

**console.log(2 (+ ,-,%,\*,^) 1) =>**

**variable = i**

**i ++ i—**

## logical-operators

**! = not**

**&&= and**

**Note:**

**all of the conditions must be acheived look in (Ex) to understad**

**|| = or**

**Note:**

**one or two of them is to acheive**

**Ex:**

**console.log(10 == "10" && 10 > 8 && 10 > 50);**

**console.log(!true);**

**console.log(10 == "10" || 10 > 80 || 10 > 50);**

**console.log(!(10 == "10"));**

## comparison-operator

**Mean: to compare between variables in date-types or values**

**== Equal**

**!= Not Equal**

**=== Identical**

**!== Not Identical**

**> Larger Than**

**>= Larger Than Or Equal**

**< Smaller Than**

**<= Smaller Than Or Equal**

**Ex:**

**console.log(10 == "10"); // Compare Value Only**

**console.log(-100 == "-100"); // Compare Value Only**

**console.log(10 != "10"); // Compare Value Only**

**console.log(10 === "10"); // Compare Value + Type**

**console.log(10 !== "10"); // Compare Value + Type**

**console.log(10 !== 10); // Compare Value + Type**

**console.log(10 > 20);**

**console.log(10 > 10);**

**console.log(10 >= 10);**

**console.log(10 < 20);**

**console.log(10 < 10);**

**console.log(10 <= 10);**

**console.log(typeof "Osama" === typeof "Ahmed");**

## nullish && logical-operators

**to return value in case of the value is null or undefined or falsey**

**>Note: falsey is 0**

**>Syntax:**

**-logical-> ||**

**-nullish-> ??**

**Ex:**

**let a = 0;**

**console.log(`the price is: ${a || 200 }`);**

**console.log(`the price is: ${a ?? 200 }`);**

# Dealing with numbers

## some-notes:

**syntactic-sugar:**

* **to wirte number in easy-way**

**note on fixed:**

* **if the decimal-number is more-than or equal real-number then it will inc by one**

## part\_one

**().tostring():to convert number to string-> {data-type\_only}**

**().toFixed():to make specific type of number appear after decimal-point**

**parseint(): to convert string to integer-number**

**parsefloat(): to convert string to float-number**

**isInteger():checks the number is integer or not**

**isNan():checks the number is Nan or not**

**Ex:**

**(100).toSstring();**

**(100).toFixed();**

**parseint(“100”);**

**parsefloat(“100.500”);**

**number. isInteger(100);**

**number. isnan(“osama”/100);**

## part\_two

**Math.round():remove decimal or add 1 on the number and It depend on decimal if it is half or more than half then it will inc 1 otherwise it will remove the fraction**

**Math.ceil():add one**

**Math.floor():minius-one**

**Math.min():choosing the small number and it will display it and you must give him group of numbers**

**Math.max():choosing the big number and it will display it and you must give him group of numbers**

**Math.pow():making number power number (\*\*)**

**Math.trunc():removing fraction**

**Math.random():will generate random number**

**Ex:**

**Math.round(45.2) , round(45.5) , round(45.8)**

**Math.floor(45.5)**

**Math.min(45,26,-1,200)**

**Math.max(45,26,-1,200)**

**Math.pow(4,5)**

**Math.trunc(45.5)**

**Math.random()**

# Strings

## Part\_ONE

* **-Access With Index->[]->to find specific-character with numbers (zero-based)**
* **-Access With->charAt()->same as index**
* **-.length->to know characters-number (count from one )**
* **-.toUpperCase()-> to make all string-uppercase**
* **-.toLowerCase()->to make all string-lowercase**
* **-.trim()->to remove the space before and after**

**Ex:**

**a[5]**

**a.charAt(5)**

**a. length**

**a.toUpperCase()**

**a.tolowerCase()**

**a.trim()**

**(a.toUpperCase.tolowercase().trim())**

## Part\_TWO

**-indexOf(“word to search”, begin-index )-> to find specific character or string (from begin to end)**

* **-1 if it don’t exist otherwise it will be resulted in index of element**

**-lastIndexOf(“word to search”, Start Number)**

* **to find specific character or string (from end to begin)**

**-slice(Start [number], End+1)**

* **to cut specific string or numbers of character**
* **if the end is 4 then he will stop at 3 :)(End+1)**

**-repeat(number) [ES6]->to repeat the string or num**

**-split(“separator”)=> (converting String to array)**

**Ex:**

**console.log(a.indexOf("iam"))**

**console.log(a.lastIndexOf("h",-5))**

**console.log(a.slice(1,10).toUpperCase().trim())**

**console.log(a.repeat(15))**

**console.log(a.split(""))**

## Part\_THREE

**-substring(Begin-index, End+1)->like slice**

* **if the begin > end => then it will swap**
* **if the begin < 0 => then it will begin from 0**

**-substr(Begin-index, End+1)**

* **if start\_value > length => it will resulted a Empty-string**
* **if start\_value < 0 => it will count from end**

**-includes(Value, Begin-index, Default 0)=>**

* **to check if the string or element you search for is exist or not**

**-startsWith(Value [word, index Default 0)=>**

* **to check if the string or element is the begining or not**

**-endsWith(Value, Length of value)=>**

* **to check if the string or element is the end or not**

**Ex:**

**console.log(b.length)**

**console.log(b.substring(8,4));**

**console.log(b.substr(5,21));**

**console.log(b.includes("e",8));**

**console.log(b.startsWith("r",10));**

**console.log(b.endsWith("d",3))**

# Statements

## If-introduction

**-syntax:**

**-if(condition is true){**

**Do this**

**}**

**-elseif(another condition is true){**

**Do this**

**}**

**-else(**

**Do this**

**)**

**Ex:**

**if (discount === true) { price -= discountAmount; }**

**else if (country === "Syria") {**

**price -= 50;**

**} else {price -= 10;}**

## Conditional Ternary Operator

**Uses:**

* **we can use it in small cases**

**Note:**

* **you can use it as variable**

**Syntax:**

* **condition ? true : false (if-state)**
* **condition ? true : condition-(any number of conditions you want) :false (if-state)**

**Ex:**

**console.log(theGender === "Male" ? "Mr" : "Mrs");**

**let name = theAge > 20 && theAge < 60 ? console.log("20 To 60")**

**:console.log("Larger Than 60")**

## switch-state

**Note:**

**it depend on value of varaible to test**

**cases:**

* **as you want**

**break:**

* **to prevent cases from printing with each other**

**default-value:**

* **like else in (if-state)**

**syntax:**

**swtich (var){ case value:**

**code as you want**

**break;**

**default:**

**break;**

**code as you want**

**Ex:**

**let day = 0;**

**switch (day) {**

**case 0:**

**console.log("saturday")**

**break;**

**case 1:**

**console.log("Unkown-day")**

**break;**

**}**

## Nested-condition

**if (discount === true) {price -= discountAmount;}**

**else if (country === "Egypt") {**

**if (student === true) {price -= discountAmount + 30;}**

**else**

**{**

**price -= discountAmount + 10;**

**}**

**}**

# Loops

## loop-introduction

**for(variable , condition , method-to \_count) {block of code}**

* for ( intialization , condition to stop , how it will count)
* intialization= like declaration
* condition-to-stop: when it achieve the condition it will stop-looping
* how it will count: like i++ it will count 1 , 2 , 3

**Ex:**

**for (let i = 0;  i < 1001;  i++ ){**

**console.log(`the number is ${i}`);**

**}**

## Nested-loop

Syntax:

**for (varaible = 0 , condition , varaible++)**

**{for (varaible = 0 , condition , varaible++){}}**

**Ex:**

**for (let i = 0; i < products.length; i++)**

**{**

**console.log(`#${products[i]}`)**

**for (let j = 0; j < colors.length; j++)**

**{**

**console.log(`-${colors[j]}`)**

**}**

## loop-control

**for (i,i,i){if (condition){break;}**

**for (i,i,i)**

**{**

**if (condition is true)**

**{ countiune; }**

**}**

**mainloop:for(i,i,i){if (condition){break mainloop;}**

**label:**

* **to give name for the loop**

**break:**

* **to stop the loop**

**coutiune:**

* **to resume the loop**

**Ex:**

**let products = ["Keyboard", "Mouse", "Pen", "Pad", "Monitor"];**

**let colors = ["Red", "Green", "Black"];**

**mainloop: for (let i = 0; i < products.length; i++) {**

**if (products[i] === "Pen") {**

**break;**

**}**

**console.log(products[i]);**

**nestedloop: for (let j = 0; j < colors.length; j++) {**

**if (colors[i] === "green") {break nestedloop;}}}**

## filling the Array (application on loop)

**for (let i = 0 i > array.length i++){**

**emptyarray=array.push()**

**}**

**Ex:**

**let frineds = [`mohamed`,`yasser`,`gharieb`,`sediek`,**

**`gouda`,];**

**let friendsarenumbers = [];**

**for (let i = 0; i < frineds.length; i++) {**

**if (typeof frineds[i] === "string") {**

**friendsarenumbers.push(frineds[i]);**

**console.log(frineds[i]);}**

**}**

## while-loop

**Syntax:**

**while(condition)**

**{**

**Repeat this**

**Increment**

**}**

**Ex:**

**let prodcuts = ["keyboard","ipad","iphone"]**

**let index =0;**

**while (index < prodcuts.length) {**

**console.log(`the name is ${prodcuts[index]} and his number ${index}`);**

**index+=1;**

**}**

## Do-while

**Note:**

* **It will be triggered one time even if the condition is false**

**Syntax:**

**do{**

**code we want to take specific things from it**

**increment**

**}while(condition)-> can be true or false**

**Ex:**

**let i = 0;**

**while (false) {console.log(`your number is ${i}`);**

**i++;}**

**do {console.log(i);**

**i+=4;}while (false){console.log(i)}**

# Function

## function-Introduction

**function name (parameters or parameterless)**

**{**

**task**

**}**

**Ex:**

**function saymyname (username)**

**{ console.log(`your name is ${username}`) }**

**saymyname(`hiesenberg`) => Callback**

**function print\_hi ()**

**{**

**console.log(`hi`)**

**}**

**print\_hi ();**

## Return And Use Cases

**function name (parameters)**

**{return task}**

**Note:**

**بالبلدي كدا ان هي هترجع قيمه ايا كان بقا هي ايه**

**Ex:**

**function generate(start, end) {**

**for (let i = start; i <= end; i++) {**

**console.log(i);**

**i === 15 ? `stop` : ‘contuine’**

**}**

**Return value;**

**}**

**generate(10, 20); Callback**

## Default Parameters

**2nd-way**

* **function (parameters)**

**{**

**parameters=parameters || “Backup-value” (nullish-operator)**

**}**

**3rd-way(Ecma-script)**

* **function (parameters=”value as you want”){}**

**Ex:**

**function sayhello (username, age=`unkown`)**

**{**

**console.log(`your name is ${username} and your age is ${age}`)**

**}**

**sayhello(`wiza`)**

## Rest Parameters

**Syntax:**

* **function (...parameter){task you want}**

**Note:**

**دور على استخدامتها**

**Ex:**

**function calc(...cero) {**

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

**console.log(calc(10, 20, 30, 45, 23, 234));**

## nested-function

**function name (parameters)**

**{**

**Task**

**function(){ task }**

**Return;**

**}**

**Ex:**

**function sayMessage(fName, lName) {**

**let message = `Hello`;**

**function concatMsg() {**

**message = `${message} ${fName} ${lName}`;**

**}**

**concatMsg();**

**return message;**

## arrow-function

**Note:**

* **Like annoymous maybe be existed in varaible from visual-studio code or in events**

**for-one-parameter :**

* **parameter => return**

**for-two-parameter:**

* **(parameter, parameter)=> return**

**Ex:**

**for-one-parameter :**

**let number = (num) => 100;**

**for-two-parameter:**

**(num1, num2)=> num1+num2**

## block-scope

**let name= value**

**if (condition)**

**{let name = different value}**

**Ex:**

**var x = 5;**

**if (x==5) { let x=10 console.log(`from the local-${x}`) }**

## local & global-scope variables

**(let varaible= value) ->global**

**{(let varaible= value) ->local}**

**Ex:**

**let a = 10;**

**let print = function (){let b = 10; console.log(b);};**

**console.log(b);**

**print();**

# Array

## Part\_one

**creating array:**

* **varaible = [data ]**

**creating nested array:**

* **varaible = [data,[data] ]**

**to access array:**

* **console.log(variable[0]);**

**to acess nested-array:**

* **console.log(array[nested array-index][any index of it])**

**overriding array value:**

* **varaible[index]=new-value;**

**change Nested array element or you can put an array inside it:**

* **varaible[Nested-array-number][any index of it ]=new-value->**

**to check if this array or not:**

* **Array.isArray(variable);**

**Ex:**

**ymn = [“y” , “m” , “n”];**

**ymn = [“y” , “m” , [“red”,”blue”]];**

**console.log(ymn[0]);**

**console.log(ymn[2][1]);**

**Array.isArray(ymn);**

**ymn[0]=”n”;**

**ymn[2][1]=[“blue”]; or ymn[2][1] = [“blue”,”green”];**

## Part\_two

**to know array\_length:**

* **array.length**

**to add the last value of length:**

* **array[array.length]= new element**

**to edit last value of index dynamically:**

* **array[array.length-1] = new element**

**Note:**

**you can add number more-than number of your element and your number is (3) array[4]=”wizard”; it will add it normally**

**Ex:**

**array[array.length] = “wizard”;**

**array[array.length-1] = “yorick”;**

## Part\_three

**Add Element To The First index:**

* **Array.unshift(“element”, “element”)**

**Add Element To The End:**

* **Array.push(“element”, “element”)**

**Remove First Element From Array:**

* **Array.shift()**

**Remove Last Element From Array:**

* **Array.pop()**

**Note:**

**(let first =friends.shift();) to add the removed element in varaible**

**Ex:**

**friends.unshift("zain" , "loay");**

**friends.push("ma8anem","omar ana asf")**

**friends.shift();**

**friends.pop();**

## Part\_four

**to search from begin to end:**

* **indexOf(“Search Element”, Begin-index [Opt]) => return index of element otherwise -1**

**to search from end to begin:**

* **lastIndexOf(“Search Element”, Begin-index [Opt])**

**here is checking only:**

* **includes(element, fromIndex [Opt]) [ES7]=> return boolean**

**Note:but the numbering isn’t change**

**Ex:**

**console.log(f.indexOf("red"));**

**console.log(f.lastIndexOf("red"));**

**console.log(f.includes("red"));**

## Part\_five

**To organize the array:**

**array.sort()->**

**To Reverse order of the array:**

**array.reverse()**

**To organize and Reverse:**

**array.sort.reverse()**

**Ex:**

**Console.log(a.sort())**

**Console.log(a.reverse())**

**Console.log(a.sort().reverse())**

## Part\_six

**cutting the arrays and makes new-array:**

**Array.slice(Start, End +1 )**

**to delete or add element in array:**

**Array.splice(begin-index, DeleteCount [0 No Remove], The Items To Add)**

**Ex:**

**console.log(myFriends.slice(1, 3));**

**console.log(myFriends.slice(-3));**

**console.log(myFriends);**

## Part\_seven

**add**ing all of the array **or** elements:

**New-array=array.concat(array , “number of arrays you want”)**

**converting from Array to string:**

**array.join(“separator”)**

**Ex:**

**let myFriends = ["1", "2", "3", "4”]**

**let myNewFriends = ["5", "6"];**

**let schoolFriends = ["7", "8"];**

**let allFriends =**

**myFriends.concat(myNewFriends, schoolFriends, "Gameel", [1, 2]);**

**console.log(allFriends.join(" @ "));**

**console.log(allFriends.join("|"));**

# High order function

## Notes:

* **-Element => The current element being processed in the array.**
* **-Index => The index of the current element being processed in the array.**
* **-Array => The Current Array**
* **-this => iteration(repeating)-count**
* **-callBackFunction(can be normal-function , annoymous-function , arrow-functionS)**

## M­­­­­­ap

**Notes:**

* **>Map Return A New Array**
* **>the most uses in array**

**Syntax:**

* **let varaible = array.Map(Function(Element, Index, Array){return task},iteration)**

**Ex:**

**let array=m.map(function(element, index , array){**

**return element + element;**

**},3)**

## filter

**Notes:**

* **it take specific element depending on specific condition**

**uses:**

* **for filtering array**

**Syntax:**

* **let varaible = array.filter(function(element , index , array){ return condition-as-you-want}),arg)**

**Ex:**

**let number = [4, 5, 6, 7, 8, 9];**

**let filterednumber = number.filter(function (element, index, array) {**

**return element % 2 === 0;**

**});**

## reduce

**Notes:**

* **>accumaltor=begin-value**
* **>current=end-value**
* **>intial-value=begin from value you will write**
* **>Intial-value: can bee { value in reduce-function,in variable,in array itself}**
* **>used in array**

**Syntax:**

* **let varaible = array.reduce(function(accumelator , current , index, array){condition},intial-value)**

**Ex:**

**let num = [1,2,3,4]**

**let addnums= num.reduce(function(acc,curr,index,arr){**

**return acc + curr},20)**

## For-each

**Notes:**

* **>looping and most uses in Dom**
* **>looping but in high order-function**

**Syntax:**

**forEach(callBackFunction(Element, Index, Array) { }, thisArg)**

**Ex:**

**allLis.forEach(function (ele) {**

**ele.onclick = function () {**

**allLis.forEach(function (ele) {**

**ele.classList.remove("active");**

**});**

**this.classList.add("active");**

**allDivs.forEach(function (ele) {**

**ele.style.display = "none";**

**});**

**};**

**});**

# object

## Object-introduction

**Varaible = {**

**Key:value,**

**Name:methods,**

**};**

**Ex:**

**let user ={**

**name:"yasser",**

**age:19,**

**home\_town:"portsaid",**

**saymyname:function ()**

**{return `my name is ${user.name}’}**

**};**

**console.log(user.age)**

**console.log(user.saymyname())**

## Dot Notation vs Bracket Notation

* **Dot-Notation:object.property**
* **Bracket Notation:object[“property”]**

**Ex:**

**let user ={**

**name:"mohamed",**

**age:19,**

**saymyname: function ()**

**{return `my name is ${user.name}`}};**

**console.log(user.age)**

**console.log(user[”name”]) , console.log(user[saymyname])**

## Nested Object

* **Varaible = {Key:value , Key:{} ,Methods:{} }**

**Ex:**

**addresses: {**

**spain: "barecolna",**

**egypt: {**

**one: "portsaid",**

**two: "6-october"},**

**checkAv: function () {**

**if (user.avalabel === true) {return `Free for work`}**

**else {return `Not free for work`}},**

**};**

## Object With New (constructor)

**Varaible = new object ({**

**Keys:Values,**

**Methods,})**

**Ex:**

**let user\_one = {**

**name:"yasser",**

**age:19,**

**};**

**creating an object-keyies with setting way**

**let user\_two = {};**

**user\_two.age = 38;**

**user\_two.name = "yasser"**

**user\_two.skill=["Html","Css","Java-script"]**

## This Keyword

**This**

**بتعود على الهي فيه يعني لو هي ف الجلوبال ف هي هتعود على الويندوز اسكوب**

**very-important-notes:**

**Ex:**

**let user = {**

**name: `yasser`,**

**age: 20,**

**double\_age: function () {**

**return this.age \* 2**

**}**

**هنا هتعود اليوزلا اوبجيكت**

## Create Object With Create Method

**varaible = {key:value};**

**Newvaraible = object.create(varaible)**

**بس هنا هتحط الحاجه زي منتا شايف ف المثال تحت**

**Note:**

**>taking keyies from another object**

**>to copy object but with different names**

**Ex:**

**const animal = { eats: true };**

**const rabbit = Object.create(animal, { jumps: { value: true } });**

## Create Object With Assign Method

* **Obj1**
* **Obj2**
* **Obj3**
* **Targeted\_Obj**
* **Newobj = object.create(target,soucres)**

**uses:**

* **>to create-(copying) object from other-objects**

**Ex:**

**let obj1 = {**

**prop\_one: `one`,**

**prop\_two: `two`,**

**prop\_three: `three`,**

**}**

**let obj2 = {**

**prop\_four: `four`,**

**prop\_five: `five`,**

**prop\_six: `six`,**

**}**

**let obj3 = {**

**prop\_seven: `seven`,**

**prop\_eight: `eight`,**

**prop\_nine: `nine`,**

**}**

**let targeted\_object={**

**prop\_one: `one`,**

**prop\_two: `two`,**

**prop\_three: `three`,**

**}**

**let object\_from\_copied = Object.assign(targeted\_object,obj1,obj2,obj3);**

# Document-object

## Select-Elements

**Find Element By ID:**

* **document.getElementById("id-name")**

**Find Element By Tag Name:**

* **document.getElementsByTagName("tag-name")**

**Find Element By Class Name:**

* **document.getElementsByClassName("class-name")**

**Find Element By CSS Selectors:**

* **document.querySelector("Css-selector")**
* **or**
* **document.querySelectorAll("Css-selectors")**

**Find Element By Collection**

* **document.title**
* **document.body**
* **document.image**
* **document.forms**
* **document.link**

**Notes:**

* **>Indexing ([]) is very-important to acess on elements**
* **>putting them in varaible to make the access on them easier**

**Ex:**

**let a = document.getElementById("mydiv")**

**let b = document.getElementsByTagName("p")**

**let c = document.getElementsByClassName("my-article")**

**let d=document.querySelector(".special")**

**let e=document.querySelectorAll(".special")**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Get / Set Elements Content And Attributes

**Get-attrbute Directly**

* **Element.attribute**

**Method**

* **Element.getAttribute("attribute-name")**

**set-attrbute Directly**

* **Element.attribute= “value”**

**Method**

* **Element.setAttribute("attribute-name",”value)**

**Innerhtml**

* **Element.innerhtml=>printing**
* **Element.innerhtml = “value” => editing**

**textcontent**

* **Element.textcontent=>printing**
* **Element.textcontent= “value” => editing**

**Notes:**

**textContent:**

* **is for working with plain text, and it doesn't interpret or render HTML.**

**innerHTML:**

* **is for working with HTML content, allowing you to insert HTML tags and elements dynamically.**

**Ex:**

**get**

* **console.log(myelementbyid.id)**
* **console.log(myelementbyid.getAttribute("title"))**

**set**

* **myelementbyid.title="hello a5oya"**
* **console.log(myelementbyid.setAttribute("title","7ma"))**

**Innerhtml**

* **myelementbyid.innerHTML="hello from main.js file"**

**textcontent**

* **myelementbyclassname.textContent="hello from <span> main.js </span>"**

## Check Attributes

* **element.attributes**
* **element.hasAttribute(“attribute-name”)**
* **element.removeAttribute(“attribute-name”)**
* **element.hasAttributes()**

**Ex:**

**element.attributes:**

* **console.log(mydiv.attributes)**

**element.hasAttribute:**

**mydiv.hasAttribute("alt") ? mydiv.removeAttribute("alt")**

**: console.log("value of it existed")**

**mydiv.hasAttributes() ? console.log("kteer awy yaaa 3m") :**

**console.log(5555555555);**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Create And Append Elements

**createElement**

* **varaible=document.createelement(“tag-name”)**

**createComment**

**varaible= document.createcomment(“text”)**

**createTextNode**

* **varaible=document.createtextnode(“text”)**

**createAttribute**

* **varaible=document.createattribute**
* **element.setattributenode(varaible)**

**appendChild**

* **element.appendChild (“anything you want”)**

**Ex:**

**Let a =document.createElement("div");**

**let b= document.createComment("this div")**

**let c= document.createTextNode("hi")**

**let e = document.createAttribute("data-custom")**

**a.className="prodcut-one"**

**b.append(my\_comment)**

**c.append(my\_text)**

**d.setAttributeNode(my\_attributes)**

**document.body.append(my\_element)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Deal With Children's

* **Element.children**
* **Element.childnodes**
* **Element.firstchild**
* **Element.lastchild**
* **Element.firstElementChild**
* **Element. lastElementChild**

**Ex:**

**let testdiv = document.querySelector(`div`)**

**console.log(testdiv)**

**console.log(testdiv.children)**

**console.log(testdiv.childNodes)**

**console.log(testdiv.firstChild)**

**console.log(testdiv.lastChild)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## DOM Events

* **--- onclick (left-click\_mouse)**
* **--- oncontextmenu (right-click\_mouse)**
* **--- onmouseenter (hover in css)**
* **--- onmouseleave (hover in css)**
* **--- onload(page-load)**
* **--- onscroll**
* **--- onresize**
* **--- onfocus(foucs in Css)**
* **--- onblur(when you click outside)**
* **--- onsubmit(on clicking on submit button)**

**element.event = function**

**Note: function can be annoymous , arrow , normal-function**

**Ex:**

**let a = document.getElementById("button")**

**let b = document.getElementById("label")**

**let c = document.getElementsByTagName("form")**

**mybutton.onclick = function () {**

**mylabel.textContent = "good!"**

**}**

## Event Simulation

**element.event = function()**

**{another-element.Click() or .Focus() or .Blur()}**

**Ex:**

**let a = document.querySelector("[name=one]")**

**let b = document.querySelector("[name=two]")**

**let c = document.getElementsByClassName("link")**

**window.onload = function () {**

**input\_one.focus()**

**}**

**input\_two.onfocus = function()**

**{document.links[0].click()}**

## classlists

**varaible= element**

**element.classList**

**element.classList.length**

**element.classList.contains('class-name')**

**element.classList.item(0)**

**element.classList.add('new-class')**

**element.classList.remove('class-name')**

**element.classList.toggle('class-name')Notes:**

**classlists => to see what element contain classes**

* **.length => to know how many elements**
* **.contains => to show if specific class existed or no**
* **.add => to add class**
* **.remove => to remove class**
* **.item => like index of the array**

**toggle =>**

**if the class existed**

**=> he will remove it**

**if the class not existed**

**=> he will add it**

**Ex:**

**let a = document.getElementById("div")**

**console.log(a.classList.length)**

**console.log(a.classList.contains("six"))**

**a.onclick = function(){**

**this.classList.remove("one")**

**this.classList.toggle("blue")**

**this.classList.add("one")**

**}**

## CSS Styling And Stylesheets

**varaible = element**

* **element.style.property=”value”**
* **element.style.csstext=”property:value”**
* **element. removeProperty(Propertyname)**
* **element.setProperty(Propertyname , value , priortey {important}]**

**Note:**

**to access on css-file and it’s properties**

* **document.stylesheet[file-number].rules[property-number].**

**to access on css-file and it’s properties and edit them:**

* **document.stylesheet[file-number].rules[property-number].style**

**removeProperty(Propertyname) [inline , stylesheet]**

**setProperty(Propertyname , value , priortey {important}]**

**Ex:**

**let a = document.getElementById("div")**

**a.style.color="red";**

**a.style.backgroundColor="#eee"**

**a.style.cssText="font-size:100px; width:100%; text-align:center;"**

**a.style.removeProperty("font-size")**

**a.style.setProperty("font-size","100px")**

**console.log(document.styleSheets[0].rules[0].style.backgroundColor="red")**

## Before, After, Prepend, Append, Remove

**varaible = element**

* **element.before(Element || String)**
* **element.after(Element || String)**
* **element. append (Element || String)**
* **element.prepend(Element || String)**
* **element.remove()**

**Note:**

**append [Element || String] to add element or string in last of page**

**prepend [Element || String] => to add element or string first place in page**

**Ex:**

**let a = document.getElementById("tree")**

**let myparagraph = document.createElement("p");**

**a.before("hi")**

**a.after("yooo")**

**a.append(”wassup”)**

**a.prepend(myparagraph)**

**a.remove()**

## DOM Traversing

**varaible = element**

* **element.nextSibling**
* **element.previousSibling**
* **element.nextElementSibling**
* **element.previousElementSibling**
* **element.parantElement**

**Ex:**

**let span = document.querySelector(".two")**

**console.log(span.nextSibling) => الباقي زيه**

**span.onclick=function(){**

**span.parentElement.style.cssText="background-color:black; color:white;"**

**}**

## DOM Cloning

**varaible = element.cloneNode(true || false)**

**Note:**

**deep:mean anything inside the element and it has default value which is false**

**Ex:**

**let a =document.getElementById("my-p").cloneNode(true);**

**let b = document.getElementById("my-div");**

**b.append(my\_p);**

**a.id=`${my\_p.id}+cloned`**

## addEventListener

**variable = element**

* **element.addEventlistener(“event”,function)**

**Note:**

* **>addEventListener: to make more than one event at same time**
* **>event.target: mean when you make the event on element or specific thing**

**Ex:**

**document.body.addEventListener("click",function(e){**

**if (e.target.classname==="iam-cloned") {**

**console.log("this paragraph is cloned ala")**

**}**

**})**

# Bom

## **Alert, Confirm, Prompt**

* **.alert(Message or varaible)**
* **.confirm(Message or varaible)**
* **.prompt (Message or varaible) => for input test**

**Notes:**

**in global-scope you can any windows object write in these sytnaxes**

**-windows.object**

**-this.object**

**-object**

**Ex:**

**console.log(`hi wiza`);**

**let prompt = this.prompt(Msg,`no wiza`);**

## **setTimeout**

**variable = setTimeout(Function, Timeout, Additional Params)**

**event = clearTimeosut( variable)**

**Ex:**

**let btn = document.querySelector("button")**

**let counter = setTimeout(sayhello, 2000, `yasser`, `20`);**

**function sayhello(user, age) {**

**console.log(`hello ${user} with ${age} from js-file`);**

**}**

**btn.onclick = function(){**

**clearTimeout(counter)**

**}**

**Notes**

* **after specific time specific function will be triggered or executed**

## **setInterval and clearInterval**

**variable = setInterval(Function, Millseconds, Additional Params)**

**clearInterval(variable)**

**Ex:**

**let div = document.querySelector("div");**

**function countdown() {**

**div.innerHTML -= 1;**

**console.log(`text-1`)**

**if(div.innerText === `0`){**

**clearInterval(interval)**

**}**

**}**

**let interval = setInterval(countdown,1000,`yooo wiza `);**

## **location Object**

**location.href**

* **Get/Set: Allows you to get the current URL or set a new one to navigate to a different page.**

**Uses:**

* + **Get: Retrieve the full URL of the current page.**
  + **Set: Change the URL, which redirects the browser to the new URL.**

**location.reload()**

* **Reloads the Page: Reloads the current page.**

**Uses:**

* + **Useful for refreshing the content of the page. You can force a reload from the server by passing true as an argument: location.reload(true).**

**location.replace()**

* **Replace the Page in History: Replaces the current document with the one at the provided URL.**
* **Uses:**
  + **Navigates to a new page without leaving the current page in the session history. This means the user cannot go back to the original page using the back button.**

**location.hash**

* **Get/Set URL Fragment: Allows you to get or set the hash portion of the URL (the part after #).**

**Uses:**

* + **Navigate to a specific section within the current page or a different page using an anchor.**

**location.assign()**

* **Navigate to Another Page: Loads a new document and adds it to the session history.**

**Uses:**

* + **Similar to setting location.href, but keeps the current page in history, allowing the user to return to it using the back button.**

**location.protocol**

* **Get/Set Protocol: Allows you to get or set the protocol of the current URL (e.g., http:, https:).**
* **Uses:**
  + **You can switch between protocols, such as changing http to https, which will redirect the page.**

**location.host**

* **Get Host: Returns the hostname and port number of the URL.**
* **Uses:**
  + **Useful for extracting or verifying the host part of the URL, which might include the IP address or domain name along with the port.**

**location.hostname**

* **Get Hostname: Returns the domain name of the web host (e.g., www.example.com).**
* **Uses:**
  + **Useful for getting the domain without the port number, allowing you to check or manipulate the base domain of the current URL.**

**http:**

* **hyper text transmit protocol**

**https:**

* **hyper text transmit protocol secured**

**Ex:**

### location.href

Get the current URL:

console.log(location.href); // Outputs: "https://www.example.com/page"

Set a new URL:

location.href = "https://www.example.com/new-page"; // Navigates to the new page

### location.reload()

Reload the current page:

location.reload(); // Reloads the page

Force reload from the server:

location.reload(true); // Forces a reload from the server, ignoring the cache

### location.replace()

Replace the current page with a new one:

location.replace("https://www.example.com/replacement-page"); // Replaces current page in history

### location.hash

Get the current hash:

console.log(location.hash); // Outputs: "#section1" if URL is "https://www.example.com/page#section1"

Set a new hash:

### location.hash = "#new-section"; // Navigates to the section with id="new-section"

location.assign()

### location.assign:

location.assign("https://www.example.com/another-page"); // Navigates to another-page, keeping current page in history

### location.protocol

Get the current protocol:

console.log(location.protocol); // Outputs: "https:" or "http:"

Set a new protocol:

location.protocol = "https:"; // Changes the protocol to HTTPS, reloading the page

### location.host

Get the host (including port if present):

console.log(location.host); // Outputs: "www.example.com" or "www.example.com:8080"

### location.hostname

Get just the hostname:

console.log(location.hostname); // Outputs: "www.example.com", without port number

## **Window Open And Close**

**window.open(URL , Window Name Or Attr, Win Features, History Replace)**

**window.close()**

**Notes:**

**open:(URL [Opt], Window Name Or Target Attr [Opt], Win Features [Opt], History Replace [Opt])**

**close()**

**Window Features**

**left [Num]**

**top [Num]**

**width [Num]**

**height [Num]**

**Ex:**

**let timeed\_page = setTimeout(() =>window.open("https://google.com","\_blank","width=600**

**height=600 top=500 left=600"), 2000);**

**btn[0].onclick = ()=>timeed\_page.close() , btn[0].style.backgroundColor="red"**

## **Window History Object**

**console.log(history)**

* **history.length**
* **history.back**
* **history.back()**
* **history.forward**
* **history.go()**

**Ex:**

**console.log(history) => properties**

**console.log(history.length) => length**

**console.log(history.back()) => back()**

**console.log(history.forward()) => forward()**

**history.go(-1) => go()**

## **Scroll, ScrollTo, ScrollBy, Focus, Print, Stop**

* **window.stop()**
* **window.scrollTo()**
* **window.print()**
* **window.focus()**
* **window.scrollBy()**
* **window.scroll()**

Notes:

**stop():**

**to stop content of page to load**

**print():**

**to print the page!**

**focus():**

**to foucs on opened-windows**

**scrollTo(x, y || Options):**

**scroll to specific section**

**scrollBy(x, y || Options):**

**mean scroll by this value**

**option-means: top ,left only**

**numbers is written without measuring-unit**

**Ex:**

**let my\_window = setTimeout(()=> window.open("https://google.com","\_blank","width=400 , height=500"),1000)**

**window.focus()**

**window.scrollBy({**

**left:200,**

**top:200,**

**behavior:"smooth"**

**})**

**window.scrollTo(1000,0)**

## **Local Storage**

**Set**

* **window.setitem(`key`,`value`)**
* **window.localstroage.key = ` value `**
* **window.localstroage[`key`] = ` value `**

**Get**

* **window.localstroage.getitem(`key`)**
* **window.localstroage.key**

**Other**

* **window.localstorage.removeItem(`key`)**
* **window.localstorage.clear(key)**

**console.log(window.localstorage.key(index))**

**Note:**

**we use key to print value**

**Private Tab => reseting in private-tab**

**Ex:**

**set**

**window.localStorage.setItem(`color`, `#eee`);**

**window.localStorage.setItem(`2nd-background-color`,`#def`);**

**window.localStorage[`font-weight`] = `bolder`**

**window.localStorage[`font-size`] = `20px`**

**get**

**window.localStorage.getItem(`color`);**

**Remove-one**

**// window.localStorage.removeItem("color");**

**Remove-all**

**window.localStorage.clear()**

**acessing-key**

**console.log(window.localStorage.key(2))**

## **seesion-storage**

* **window.SessionStorage.setItem**
* **window.SessionStorage.removeItem**
* **window.SessionStorage.getItem**
* **window.SessionStorage.clear**
* **window.SessionStorage.key**

**Ex:**

**window.localStorage.setItem("color", "red");**

**window.sessionStorage.setItem("color", "blue");**

**document.querySelector(".name").onblur = function () {**

**window.localStorage.setItem("input-name", this.value);**

**};**

# Destructing

## Destructuring Arrays Part 1

**array=[elements]**

**[varaibles]=array**

**[varaible, varaible=`backup-value`]=array**

**[varaible, , varaible]=array[3] => to skip one element of array**

**Ex:**

let my\_friends = [`mans`,`wiza`,`ahmed`,`mohmaed`,`akram`,`555555555`];

[,b,c,d,e,r,f=`wizard`] = my\_friends

## Destructuring Array Advanced Examples

**array = [1,2,3,4]­**

**[ , , , a ] = array**

**Ex:**

**let myFriends = ["1","2","3", "4", "5", ["6 ,"7"] ];**

**[, , , [a , , [ , b ]]] = myFriends**

## Destructuring Arrays Part 3

**swapping by destructing**

* **let a= 1**
* **let b =2**
* **[a,b] = [b , a ]**

**Ex**:

let book = `video`;

let video = `book`;

// ! by swapping by destructing  array

[book , video] = [video , book]

## Destructuring Objects Part 1

Syntax:

const { property1, property2 } = object;

Ex:

const person = {

name: "John",

age: 30,

occupation: "Engineer"

};

const { name, age, occupation } = person;

console.log(name); // Output: "John"

## Destructuring Objects Part 2

**change property-name**

**let [ property: name } = object**

**make new property and give it a value**

**let [ property: name =`value` } = object**

**destruct a nested-object**

**let [ nested\_object:{nested\_property} } = object**

**destruct a nested object-only**

**let {nested\_property}=object.nested\_object**

**Ex:**

**user*=*{**

**name:`yasser`,**

**age:19,**

**job:`freelancer`,**

**nested\_user:{**

**nested\_name:`yasser`,**

**nested\_age:19,**

**nested\_job:`freelancer`**

**}**

**}**

**// let {name: esm ,age: omr ,job: shoghl } = user**

**// let {name: esm ,age: omr ,job: shoghl , gender:gnsh = 'unkwon' } = user**

**// let {name: esm ,age: omr ,job: shoghl , gender:gnsh = 'unkwon',**

**nested\_user:{name} } = user**

**// ? destruct a nested object-only**

**let {nested\_age:age\_n} = user.nested\_user**

## Destructuring Function Parameters

**let object = property**

**function ({properties}=object){**

**console.log(`properties`)**

**function ({properties:name as you want}=object){**

**console.log(`name as you want `)**

**}**

**Ex:**

***let* user *=* {**

**name: `yasser`,**

**age: 19,**

**job: `freelance`,**

**};**

***// ? normal-function***

***// function show\_details(object){***

***//     console.log(`your name is "${object.name}"`)***

***//     console.log(`your Age is "${object.age}"`)***

***//     console.log(`your job is "${object.job}"`)***

***// }***

***// show\_details(user)***

***// function show\_details({ name, age, job } = user) {***

***//   console.log(`your name is "${name}"`);***

***//   console.log(`your Age is "${age}"`);***

***//   console.log(`your job is "${job}"`);***

***// }***

***// show\_details();***

***function* show\_details({**

**name: *your\_name*,**

**age: *your\_age*,**

**job: *your\_job*,**

**} *=* user) {**

**console.log(`your name is "${your\_name}"`);**

**console.log(`your Age is "${your\_age}"`);**

**console.log(`your job is "${your\_job}"`);**

**}**

**show\_details();**

# Set-data\_type

## Some-notes:

**it’s element can’t be accessed with indexing**

**add => to add elements**

**delete => to delete elements**

**clear => to clear the array**

**has => to check if the array has specific element or no**

## Set Data Types And Methods

**Decalaretion**

**let array = [elements]**

**let unique\_array = new Set (array) or (elements)**

**Methods**

**let unique\_array = new Set ().add(element) => repeat it as you want**

**let unique\_array = new Set ().delete(element-name)**

**let unique\_array = new Set ().clear()**

**let unique\_array = new Set ().has()**

**Ex:**

**let matrix = [1,2,3,4,5,6,7,8,9,10]**

**let unique\_matrix = new Set(matrix)**

**console.log(unique\_matrix)**

**console.log(unique\_matrix.delete(9))**

**console.log(unique\_matrix.has(`a`.toUpperCase()))**

**unique\_matrix.clear()**

## Set vs WeakSet And Garbage Collector

**Note: Garbage-collecter is an object without a reference**

**variable = array**

**unique\_varaible = new Set (array)**

**variable = object**

**let unique\_varaible = new WeakSet([{object }])**

**Ex:**

**let determinats =**

**[2,4,6,8,10,14,16,18,20]**

**let unique\_determinats = new Set(determinats)**

**console.log(unique\_determinats.size)**

**console.log(unique\_determinats.keys())**

**let object\_determinats = {A:1,B:2,C:3,D:4}**

**let unique\_object\_determinats = new WeakSet([{object\_determinats}])**

**console.log(unique\_object\_determinats.size)**

**console.log(unique\_object\_determinats.keys)**

## Map Data Type Vs Object

**Map**

**variable = new Masp**

**varaible.set(key , “value”)**

**console.log(varaible.get(key) )**

**Object**

**object.create(null) to remove default-keys**

**let variable = new object ({**

**key:`value`,**

**“key”:`value`**

**})**

**Ex:**

**let Maped\_Names = new Map()**

**Maped\_Names.set(`color`, `red`)**

**Maped\_Names.set(Boolean, true)**

**Maped\_Names.set(Object, { a: 1, b: 2, C: 3 })**

**console.log(Maped\_Names.get(`color`))**

**console.log(Maped\_Names.get(Boolean))**

**console.log(Maped\_Names.get(Object))**

**console.log(Maped\_Names.size)**

**let Old\_object = Object.create(null)**

**let Object\_Names = Object.create({**

**10: `number`,**

**11: 'red',**

**"12": 'BLUE'**

**})**

**console.log(Object\_Names[10])**

## Map-methods

**Map.set()**

**Map-variable = New Map(varaible)**

**let Map = new Map([**

**[key, "value"],**

**]);**

**console.log(Map.get(Key)) Map.clear()**

**map.delete(Key) Map.size**

**console.log(map.delete(Key)) Map.has()**

**Ex:**

**let Map\_Names = new Map([**

**[10,`Number`],**

**[12,`Number`],**

**[14,`Number`],**

**])**

**Map\_Names.set(20,`twenety`)**

**console.log(Map\_Names.get(20))**

**Map\_Names.delete(14)**

**console.log(Map\_Names.delete(14))**

**console.log(Map\_Names.has(10))**

**console.log(Map\_Names.size)**

**Map\_Names.clear**

## Map Vs WeakMap

**weak\_ Maped\_variable = new map()**

**Maped\_variable = map()**

**map.set()**

**weak\_ Maped.set()**

**Ex:**

**let MyData= {name:`wiza`,}**

**let My\_Map = new Map()**

**My\_Map.set( MyData , `object-value`);**

**MyData = null**

**let MyweakData= {name:`wiza`,}**

**let My\_weak\_Map = new WeakMap()**

**My\_weak\_Map.set( MyweakData , `object-value`);**

**MyweakData = null**

**console.log(My\_weak\_Map)**

## Array.from Method

**Array.from(iterable , function)**

**Ex:**

**console.log(**

**Array.from("12345", function (n) {**

**return +n + +n;**

**})**

**);**

**console.log(Array.from("12345", (n) => +n + +n));**

**let myArray = [1, 1, 1, 2, 3, 4];**

**let mySet = new Set(myArray);**

**console.log(Array.from(mySet));**

**function af() { return Array.from(arguments); }**

**console.log(af( "Osama", "Ahmed", "sayed", 1, 2, 3) );**

**let Names = `ahmedyassergharieb`;**

**console.log(Array.from(Names))**

## Array.copywithin

**Array.copyWithin(Target-index, Start-index, End-index)**

**Ex:**

**let array = [1,2,3,4,5,7,8]**

**array.copyWithin(0,9)**

**console.log(array)**

**Some-notes**

**length of array don’t change**

**the elements are copied not cutted careful-!**

**Any Negative Value Will Count From The End**

**in target If At Or Greater Than Array Length Nothing Will Be Copied**

**in index not including the end**

## Array.some Method

**Array.some(CallbackFunc(Element, Index, Array), This Argument)**

**Ex:**

**let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15];**

**let maximum = 15**

**let checking\_numbers = numbers.some(function (e) {**

**return e >= this**

**}, maximum)**

**let range = {min: 10,max: 20}**

**let check\_in\_range = numbers.some(**

**(e) => e > this.min && e < this.max, range)**

## Array.every Method

**Array.every(CallbackFunc(Element, Index, Array), This Argument)**

**Ex:**

**const locations = {**

**20: "place-one",**

**30: "place-two",**

**40: "place-three",**

**50: "place-four",**

**}**

**let Main\_location = 15;**

**let arrayed\_location = Object.keys(locations)**

**console.log(arrayed\_location)**

**let arrayed\_location\_numbers = arrayed\_location.map((n) => +n)**

**console.log(arrayed\_location\_numbers)**

**let checking = arrayed\_location\_numbers.every(function (e) {**

**return e > this**

**}, Main\_location)**

**console.log(checking)**

**Notes:**

**to store obejct-keys in array use variable =object.keys(object)**

## Spread Syntax And Use Cases

**Spread With String => Expand String**

**(..."datatype")=>to be spaced string;**

**[..."datatype"]=>to be array;**

**varaible = [...Array1, ...Array2]; => to add to arrays togther**

**copiedArray = [...myArray1]; => to copy arrays togther**

**array = ["Elements"]=> array ;**

**let pushing = [”Elements”];**

**pushing.push(...array); => to array in array with dynamic addition of element**

**variable= [Data-type];**

**Math.object(...variable)**

**{ ...object, ...object, to add-properties here if u want to }**

**Ex:**

**console.log("Osama");**

**console.log(..."Osama");**

**console.log([..."Osama"]);**

**let myArray1 = [1, 2, 3];**

**let myArray2 = [4, 5, 6];**

**let allArrays = [...myArray1, ...myArray2];**

**console.log(allArrays);**

**let copiedArray = [...myArray1];**

**console.log(copiedArray);**

**let allFriends = ["Osama", "Ahmed", "Sayed"];**

**let thisYearFriends = ["Sameh", "Mahmoud" ,"yasser"];**

**allFriends.push(...thisYearFriends);**

**console.log(allFriends);**

**let myNums = [10, 20, -100, 100, 1000, 500];**

**console.log(Math.max(...myNums));**

**let objOne = {**

**a: 1,**

**b: 2,**

**};**

**let objTwo = {**

**c: 3,**

**d: 4,**

**};**

**console.log({ ...objOne, ...objTwo, e: 5 });**

# Regular Expression

## Regular Expressions - Modifiers

* **i => case-insensitive (mean capital-letters)**
* **g => global**
* **m => Multilines**

**two ways to write it**

**1st-one**

**varaible = string**

**varaible\_pattern = /pattern/modifier(s);**

**console.log(varaible.match(patter))**

**2nd-one**

**varaible\_string = string**

**new RegExp("pattern", "modifier(s)")**

**let pattern = new RegExp(pattern,"modifiers")**

**let matching = varaible\_string.match(pattern)**

**Ex:**

**const name = `Yasser yasser YASSER YaSsEr yAsSeR`**

**let regex = `yasser`;**

**console.log(name.match(regex))**

**let new\_regex = new RegExp(regex,"i")**

**let matched\_experession = name.match(new\_regex)**

**console.log(matched\_experession)**

**console.log(new\_regex)**

## Regular Expression Ranges - Part 1

**(data|data) => data-one or data-two**

**[numbr-number] => 0 To 9**

**[^numbr-number] => Any Character Not number To number**

**Ex:**

**let urls = `.org .com .edu .net .custom-domain`**

**let urls\_Regex = /( .com |.org |.edu)/g;**

**console.log(urls.match(urls\_Regex))**

**let numbers = `1234567891011121314151617181920`;**

**let number\_regex = /[0-20]/g;**

**console.log(numbers.match(number\_regex));**

**let special\_numbers = `1@2@3#4$5%6^7&8\*9(0)`;**

**let special\_number\_regex = /[^0-9]/g;**

**console.log(special\_numbers.match(special\_number\_regex))**

**let text = `os1 os2os os3os os4os ososo5456235235os`;**

**let special\_text\_regex = /os[0-10]os/g;**

**console.log(text.match(special\_text\_regex))**

## Regular Expressions - Ranges Part 2

**[letter - letter]**

**[^letter - letter]**

**[letter\_cap-letter\_cap]**

**[^letter\_cap - letter\_cap]**

**[three-letters]**

**[^three-letters]**

**[^letter – letternumber – number cap\_letter – cap\_letter ]data don’t-contian anything in the brackets execpt one thing**

**Ex:**

**let myString = "AaBbcdefG123!234%^&\*";**

**let aToz= /[a-z]/g;**

**let NotaToz= /[^a-z]/g;**

**let AToZ= /[A-Z]/g;**

**let Abc= /[abc]/g;**

**let NotAbc= /[^abc]/g;**

**let special= /[^a-zA-z0-9]/g;**

**console.log(myString.match(aToz));**

**console.log(myString.match(NotaToz));**

**console.log(myString.match(AToZ));**

**console.log(myString.match(Abc));**

**console.log(myString.match(NotAbc));**

**console.log(myString.match(special));**

## Regular Expressions - Character Classes Part 1

**. => matches any character, except newline or other line terminators.**

**\w => matches word characters. [a-z, A-Z, 0-9 And Underscore]**

**\W => matches Non word characters**

**\d => matches digits from 0 to 9.**

**\D => matches non-digit characters.**

**\s => matches whitespace character.**

**\S => matches non whitespace character.**

**Ex:**

**let email = 'O@@@g...com O@g.com O@g.net A@Y.com O-g.com o@s.org 1@1.com';**

**let AllCharater = /./g;**

**let Wordonly = /\w/g;**

**let NonWordonly = /\W/g;**

**let Digits = /\d/g;**

**let NonDigits = /\D/g;**

**let WhiteSpaces = /\s/g;**

**let NonWhiteSpaces = /\S/g;**

**let Valid\_email = /\w@\w.(com|org)/g;**

**console.log(email.match(AllCharater))**

**console.log(email.match(Wordonly))**

**console.log(email.match(NonWordonly))**

**console.log(email.match(Digits)) console.log(email.match(NonWhiteSpaces))**

**console.log(email.match(NonDigits)) console.log(email.match(Valid\_email))**

**console.log(email.match(WhiteSpaces))**

## Character Classes Part 2

**\b:to check if word is starting or at the end**

**\B:to check if word is ending or at the end**

**Test-method: to test some experession on string**

**Ex:**

**let names = 'spamyasser5 yasserspam5 yasser'**

**let regex = /(\bspam|spam\b)/ig;**

**console.log(regex.test(names))**

**console.log(regex.test("names"))**

**console.log(regex.test("spam yasser"))**

**console.log(regex.test("yasser spam"))**

## Quantifiers Part 1

**varaible = / \w+ /ig => mean if there something of it or more (most comman uses for character)**

**varaible = / \w\* /ig => mean if there something of it or more (most comman uses for numbers)**

**varaible = / \w? /ig => mean if there something exsit or more (most comman uses for both)**

**Ex:**

**let mails = "o@nn.sa osama@gmail.com elzero@gmail.net osama@mail.ru"; let Mails\_Regex = /\w+@\w+.\w+/ig**

**console.log(mails.match(Mails\_Regex))**

**let nums = "0110 10 150 05120 0560 350 00"; // 0 Numbers Or No 0**

**let numc\_regex = /0\d\*0 /ig**

**console.log(nums.match(numc\_regex))**

**let urls = "https://google.com http://www.website.net web.com"; // http + https**

**let Urls\_regex = /https/ig**

**console.log(urls.match(Urls\_regex))**

## Quantifiers Part 2

**variable = //d{x}/ig => contain number x**

**variable = //d{x,y}/ig => contain from number x to number y**

**variable = //d{x,}/ig => contain at least number x**

**Ex:**

**let serials = "S10S S100S S3000S S50000S S950000S";**

**console.log(serials.match(/s\d{2,}s/ig))**

**console.log(serials.match(/s\d{3}s/ig))**

**console.log(serials.match(/s\d{4,6}s/ig))**

## Quantifiers Part 3

**word $ => End With Something**

**^ word => Start With Something**

**?= => Followed By Something**

**Note:**

**must be put between two brackets other will result in error**

**? ! => Not Followed By Something**

**Ex:**

**let myString = "We Love Programming";**

**console.log(/ing$/ig.test(myString))**

**let names = "1OsamaZ 2AhmedZ 3Mohammed 4MoustafaZ 5GamalZ";**

**console.log(/^\d/ig.test(names))**

**console.log(/(?=z)/ig.test(names))**

**console.log(/(?!z)/ig.test(names))**

## Regular Expressions - Replace With Pattern

**For Replace & ReplaceAll**

**replace&&replaceAll("word will be replaced","the repalcement-word")**

**1st-way**

**variable = string**

**console.log(variable.replace ("@","js"))**

**2nd-way**

**let regex = /@/ig**

**console.log(variable.replace (regex,"js"))**

**3rd-way**

**console.log(variable.replace (/@/ig))**

**Ex:**

**let txt = "We Love Programming And @ Because @ Is Amazing";**

**console.log(txt.replace("@", ""));**

**console.log(txt.replaceAll("@", ""));**

**let Txt\_Regex = /@/ig**

**console.log(txt.replace(Txt\_Regex, ""));**

**console.log(txt.replace(/@/g, ''));**

# Oriented

**Note:introduction all was about what’s the oriented about :)**

## Constructor Function Introduction

**function Name (parameters){  
this.access-name for parameters.parameters**

**}**

**variable = New Name(arguments )**

**Ex:**

**function User(id,username, salary) {**

**this.u = username**

**this.i = id;**

**this.s = salary**

**}**

**let New\_UserOne = new User(101, `mohamed`, 6000)**

**let New\_UserTwO = new User(202, `yasser`, 7000)**

**let New\_UserThree = new User(303, `gharieb`, 8000)**

**console.log(New\_UserOne.u)**

## Constructor Function New Syntax

**Class name{**

**constructor(parameters){  
tasks to perform}}**

**variable = New name(paremeters)**

**console.log(object instance of classname)**

**Note:**

**instance of: to know is the method or properties are part of the class or comes from it and the answer will boolean value**

**Ex:**

**class User{**

**constructor(id,userName,salary){**

**this.id=id;**

**this.username=userName;**

**this.salary=salary+5000;**

**}**

**}**

**let User\_One = new User (101,`yasser`,5000)**

**console.log(User\_One.username)**

**console.log(User\_One instacne of User)**

## **Deal With Properties And** Methods

**Note:**

**method:function inside the the class but the difference between it and normal function is we don’t wrtie keyword”function”**

**class name{**

**constructor(parameters){**

**}**

**name(parameters){**

**}**

**}**

**Ex:**

**class User{**

**constructor(UserName,ID, salary){**

**this.u=UserName**

**this.I=ID**

**this.S=salary <5000 ? salary+500: `your salary is ${salary} `}**

**writemsg(){**

**return`hello "${this.u}" your salary is ${this.S}`}**

**}**

**let user\_mohamed = new User(`mohamed`,102,4000)**

**console.log(user\_mohamed.u)**

## Update Properties And Built In Constructors

**class name {**

**constructor (parameters){**

**this.parameters (properties)**

**}**

**method (paremeters){**

**this.parameters = method-paremeters}**

**}**

**Notes:**

**just like normal input function**

**to compare between data-type and instacne of classes use:**

**typeof data**

**instanceof data**

**Ex:**

**class User {**

**constructor(Username, Id, state, Salary) {**

**this.U = Username || `unkown`**

**this.I = Id || `undefined`**

**this.Sa = Salary < 5000 ? `here's your bonus ${Salary + 5000}` : `no bonus ala `}**

**Msg() {return `welcome back ${this.U} here's your ID ${this.I}`}**

**Update\_Name(Name){this.U = Name}}**

**let User\_One = new User(`ahmed`, 101, 5000)**

**console.log(User\_One.U)**

**User\_One.Update\_Name(`yasser`)**

**console.log(`the updated name is "${User\_One.U}"`)**

**console.log(User\_Two.Sa)**

**let Strone = `yasser`**

**let strtwo = new String("mohamed")**

## Class Static Properties And Methods

**class name {**

**constructor(){**

**static property**

**}**

**static method**

**}**

**console.log(classname.property[which have keyword-static])**

**Note:**

**it means there’s no one can have access on it execpt the class only**

**Ex:**

**class User {**

**static count = 0**

**constructor(name, id, salary) {**

**this.n = name**

**this.i = id**

**this.s = salary**

**User.count++**

**}**

**msg() {**

**return `hello from user\_Class`**

**}**

**static countmember() {**

**return `${count} member created`**

**}**

**}**

**let user\_one = new User("yasser", 101, 5000)**

**let user\_two = new User(`ahmed`, 104, 6000)**

**let user\_three = new User(`mohamed`, 102, 6000)**

**let user\_four = new User(`gharieb`, 104, 6000)**

**console.log(user\_one.msg())**

**console.log(user\_two.n)**

**console.log(user\_three.n)**

**console.log(user\_four.n)**

**console.log(`${user\_one.count} bec it's a static property`)**

**console.log(User.count)**

## Class Inheritance

class one {

constructor(parameters-one){

properties

}

methods(parameters){}

class-two extends class-one{

constructor(parameters-one , parameters-two){

super(inherited-properties)}

Note:methods will be inherited automacallay}

Ex**:**

**class User {**

**static accces;**

**constructor(username, id, age) {**

**this.u = username;**

**this.i = id;**

**this.a = age;**

**}**

**saya7a() {**

**return `a7a ya ${this.u} ya abo id ${this.i}`**

**}**

**}**

**let user\_one = new User(`yasser`, 200036827, 19)**

**console.log(user\_one.u)**

**console.log(user\_one.saya7a())**

**class admin extends User {**

**word Extends**

**constructor(username, id, age, permission) {**

**super(username, id, age)**

**this.p = permission**

**}**

**}**

**let admin\_one = new admin(`magdy`, 76827, 32, 1)**

**console.log(admin\_one.u)**

**console.log(admin\_one.saya7a())**

## Class Encapsulation

**class name {**

**#(property-name)**

**constructor(parameters){**

**this.properties**

**this.#(property-name)**

**}**

**method to access the privatie method**

**}**

**Ex:**

**class User {**

**#e**

**constructor(username, id, esalary) {**

**this.u = username;**

**this.i = id;**

**this.#e = esalary;    }**

**getSalary() {**

**return `your salary is ${parseInt(this.#e \* 0.3)}`**

**}**

**}**

**let u1 = new User(`yasser`, 200036827, 5000)**

**console.log(u1.u);**

**console.log(u1.i);**

**console.log(u1.getSalary());**

**console.log(u1.#e)**

**class Employee extends User {**

**constructor(username, id, esalary) {**

**super(username, id, esalary)**

**}**

**}**

**let E1 = new Employee(`yasser`, 102, 200)**

**console.log(E1.getSalary())**

## Prototype Introduction

**console.log(class.prototype) // features**

**Ex:**

**class User {**

**constructor(name, age) {**

**this.us = name**

**this.a = age**

**}**

**sayhello() {**

**return `hello "${this.us}"`**

**}**

**}**

**let U1 = new User(`yasser`, 19);**

**console.log(U1.sayhello())**

**console.log(U1.us)**

**let str\_one = `yasser`**

**console.log(String.prototype)**

**console.log(Number.prototype)**

## Add To Prototype Chain And Extend Constructors Features

**to add method in specific class**

**class\_name.prototype.method = function (parameter) {method}**

**to add property in specific class**

**class\_name.prototype.property = data as you want**

**to add to all classes method**

**Object.prototype.method = function (parameter) {method}**

**to add to all classes properties**

**Object.prototype.method = function (parameter) {method}**

**Ex:**

**class user {**

**constructor(name,age){**

**this.u=name**

**this.a=age**

**}**

**sayhello(){**

**return` ${this.u}**

**}**

**}**

**let U1 = new user(`yasser`,19)**

**console.log(U1.u)**

**console.log(U1.sayhello())**

**user.prototype.Expected\_Salary= function (eSalary){**

**return`the expected salray is ${eSalary\*0.5}`**

**}**

**Object.prototype.Expected\_Salary = function (eSalary){**

**return`the expected salray is ${eSalary\*0.5}`**

**}**

**Object.prototype.Hello = `hello from object constuctor`**

**console.log(U1.Expected\_Salary(5000)) console.log(U1.Hello)**

## Object Meta Data And Descriptor Part 1

**object {**

**properties:value**

**}**

**Object(keyword).defineProperty(object ,”property-name”,{**

**writable:boolean-value ,**

**enumerable:boolean-value ,**

**configurable:boolean-value,**

**value: any data you want**

**}**

**)**

**Ex:**

**let myobject = {**

**a: 1,**

**b: 2**

**};**

**Object.defineProperty(myobject, "c", {**

**writable: true ,**

**configurable: true,**

**value:3)**

**})**

**Object.defineProperty(myobject, "c", {**

**writable: false ,**

**enumerable: false,**

**configurable: true,**

**value:3000**

**})**

**console.log(myobject.c)**

**myobject.c = 3000;**

**console.log(myobject.c)**

**for (let prop in myobject){**

**console.log(prop , myobject[prop])}console.log(myobject)**

## Object Meta Data And Descriptor Part 2

**object = {properties}**

**Object(keyword).defineproperties(object-name,{properties{descriptors}})**

**to check the descriptor of the property**

**console.log(object.getowndescriptor(object,”property-name”))**

**to check the descriptors of the properties**

**console.log(object.getowndescriptors(object-only))**

**Ex:**

**const MyObject = {**

**a: 1,**

**b: 2**

**}**

**Object.defineProperty(MyObject, "c", {**

**writable: true,**

**enumerable: true,**

**configurable: false,**

**value: 50**

**});**

**console.log(MyObject)**

**Object.defineProperties(MyObject, {**

**d: {**

**configurable: true,**

**value: 1000**

**},**

**E: {**

**configurable: true,**

**value: 1001**

**},**

**F:{**

**configurable: true,**

**value: 1002**

**}**

**})**

**console.log(MyObject);**

**console.log(Object.getOwnPropertyDescriptor(MyObject, 'd'));**

**console.log(Object.getOwnPropertyDescriptors(MyObject));**

# Data&Time

## Notes:

**to get date of the place in seconds ,minutes , hours , days , years**

**getTime():it will get time in ms**

**getDate():it will get date of the but in case of january it will zero due to epoch-time**

**getDay():if it was sunday ti will be zero due to epoch-time**

**you can write date in any form as the following**

## Data-introduction

**to get today’s date :**

**varaible = new Date()**

**to get date in milleseconds**

**console.log(Date.now())**

**to get date in seconds:**

**seconds = Date.now()) / 1000**

**to get date in hours:**

**hours = seconds / 60**

**to get date in days:**

**days = hours / 24**

**to get date in years:**

**years = days / 365**

**Ex:**

**declaring data constructor**

**let date = new Date()**

**printing date in ms(milleseconds)**

**console.log(`today's datae:${date}`)**

**declaring date in millseconds**

**console.log(`mille-seconds:${Date.now()}`)**

**declaring date in seconds**

**let seconds = Date.now() / 1000**

**printing seconds**

**console.log(`seconds:${seconds}`)**

**declaring date in minutes**

**let minutes = seconds / 60**

**printing seconds**

**console.log(`minutes:${minutes}`)**

**declaring date in hours**

**let hours = minutes / 60**

**printing hours**

**console.log(`hours:${hours}`)**

**declaring date in days**

**let days = hours / 24**

**printing days**

**console.log(`days:${days}`)**

**declaring date in years**

**let years = days / 365**

**printing years**

**console.log(`years:${years}`)**

## Get Date And Time

**Declaring-date**

**variable = new date()**

**GetTime**

**console.log(variable.getTime())**

**GetDate**

**console.log(variable.getDate()) // ? output will be day of the month**

**Getyear s**

**console.log(variable.getFullYear()) // ? getting year**

**Getmonth**

**console.log(variable.getMonth())**

**Getting day**

**console.log(variable.getDay())**

**Gethours**

**console.log(variable.getHours())**

**Getminutes**

**console.log(variable.getMinutes())**

**GetSeconds**

**console.log(variable.getSeconds())**

**Ex:**

**declaring date**

**let date = new Date()**

**declaring birthdate**

**let birthdate = new Date("Aug 12,2004")**

**get the difference between them**

**let dateDiff = date - birthdate**

**printing date-differnece**

**console.log(dateDiff / 1000 / 60 / 60 / 24 / 365)**

**get time using methods of Date-constructor**

**console.log(date.getTime())**

**getting day of the month using GetDate()**

**console.log(date.getDate())**

**Getting year**

**console.log(date.getFullYear())**

**Getting month**

**console.log(date.getMonth())**

**Getting day**

**console.log(date.getDay())**

**Getting hours**

**console.log(date.getHours())**

**Getting minutes**

**console.log(date.getMinutes())**

**Getting Seconds**

**console.log(date.getSeconds())**

## Set Date And Time

**Declare Date constructor**

**Variable = new Date()**

**setTime**

**Variable.setTime(Milliseconds)**

**setDate**

**Variable.setDate(Day Of The Month [Negative And Positive])**

**setFullYear**

**Variable.setFullYear(year, month ,day)**

**setmonths**

**Variable.setMonth(Month ,Day)**

**setHours**

**Variable.setHours(Hours , Minutes, Seconds, MilleSecond)**

**setMinutes**

**Variable.setMinutes(Minutes, Seconds ,MS)**

**console.log(Variable)**

**setSeconds**

**Variable.setSeconds(seconds,milleseconds)**

**console.log(Variable)**

**Ex:**

**setting time**

**date.setTime(10000) ? in ms (mille\_second)**

**setting Date in Days**

**date.setDate(20000)**

**setting setFullYear in years & months & Days**

**date.setFullYear(2,0,1)**

**setting date in months**

**date.setMonth(0)**

**setting Hours in minutes & sec & ms**

**date.setHours(2,20)**

**setting Minutes in minutes & seconds & mille-second**

**date.setMinutes(59,59)**

**setting Seconds in second & mille-seconds**

**date.setSeconds(50)**

## Formatting Date And Time

**to convert string to number in Date use this**

**Date.parse("String")**

**Declare date (string-method)**

**varaible= new date(formatter)**

**1st- varaible= new date("month(key-word) day year")**

**2nd- varaible= new date("month(number)/day/year”)**

**3rd- varaible= new date(“year-month-day”) =>in iso-form**

**4th- varaible= new date(”year-month”)**

**4th- varaible= new date(“year”)**

**5th- varaible= new date(year,month,day,hours,minutes,seconds)**

**6th- varaible= new date(year,month,day)**

**7th- varaible= new date( year-month-daytimezone)**

**8th- varaible= new date(mille-seconds)**

**Ex:**

**console.log(Date.parse(`aug 12 2004`))**

**declaring & Testing different dates**

**let date1 = new Date(1092258000000)**

**let date2 = new Date(`aug 12 2004`)**

**let date3 = new Date("8/12/2004")**

**let date4 = new Date("2004 8")**

**let date5 = new Date("2004")**

**let date6 = new Date("82")**

**let date7 = new Date(2004,7,12,3,20,0)**

**let date8 = new Date(2000,6,25)**

**let date9 = new Date("1982-10-25T06:10:00Z")**

# Json

## Note:

**-Json= object notation**

**-introduction has no content**

## JSON Syntax And Compare With JS Object

**{“key”:value**

**Note: value can be double-qouted & maybe not depend on type of the data}**

**Ex:**

**{{**

**"string": "yasser",**

**"Numbers": 12,**

**"object": {**

**"EG": "portsaid",**

**"6th-october": "degla-palms"**

**},**

**"Array":["Spain","Germany","Egypt"],**

**"null":null**

**}**

## What Is Application Programming Interface

**vip-note:**

**any api is made in json-object**

**>application programming inteface**

**>control what you see and you can’t see**

**>two types of api**

**public:**

**what all people see and can edit and if they have access on specific something no problem if something happend**

**Ex:**

**search about github-api**

**private:**

**what people can’t see and can’t edit and if they have access on specific something some problems if something happend and it have secert key which the owner only generate it**

**for-Ex:**

**youtube channel owner they can (upload , Edit , delete , adding personal-information)**

## Parse And Stringify

**sNote:**

**string to (JSON)>parse**

**(JSON) to string>stringify**

**Ex:**

**Declaring string but in form of JSON**

**const MyobjectFromServer = `{"Username":"mohamed","Age":19}`**

**printing string-object**

**console.log(typeof MyobjectFromServer)**

**console.log(MyobjectFromServer)**

**converting from string to object notation (JSON)**

**let Myobject = JSON.parse(MyobjectFromServer)**

**console.log(typeof Myobject)**

**console.log(Myobject)**

**changing values**

**Myobject["Username"] = `yasser `**

**Myobject["Age"] = 20**

**printing changed values**

**console.log(typeof Myobject)**

**console.log(Myobject)**

**converting from object notation (JSON) to string**

**let converted\_object = JSON.stringify(Myobject)**

**printing converted\_object**

**console.log(typeof converted\_object)**

**console.log(converted\_object)**

## Asynchronous && Synchronous Programming

**Note:**

**كله نظري و تطبيق علي حجات قديمه**

**synchronous**

**- opearations run in sequences**

**- to run another opearation the previous one must be finished**

**Asynchronous**

**- opearations run in parallel**

**- another opearation can run while the previous one is still processed**

**Ex:**

**Example for synchronous**

**console.log(1); console.log(2);**

**window.alert("hi") console.log(3);**

**Example for Asynchronous**

**console.log(1);**

**console.log(2);**

**setTimeout(()=>console.log(`hi`),1000)**

**console.log(3);**

## Call Stack And Web API

**>--Code Execution Is Synchronous.**

**>--Work Using LIFO Principle => Last In First Out**

**>--lifo-means:**

**L:last , I:in , F:first , O:out means: last one enter he will work first**

**شوف المثال غشان تفهم**

**بص برضو في البيدجات بتاعت الويب لما تدوس علي حاجه تلاقي جوها حاجه و تدوس عليها ف توديك علي حاجه تاينه (جرب عشان تفهم)**

**Ex:**

**setTimeout(() => {**

**console.log("webstack api")**

**}, 0);**

**function one(){**

**console.log("one");**

**}**

**function two(){**

**one()**

**console.log("two");**

**}**

**function three(){**

**two()**

**console.log("three");**

**}**

**function four() {**

**three()**

**console.log("four");**

**}**

**four()**

**/\***

**setTimeout(() => {**

**console.log()**

**}, 0);**

**function one(){**

**console.log("one");**

**}  // ! last will run**

**function four(){**

**three()**

**console.log("four");**

**}**

**function three(){**

**two()**

**console.log("three");**

**}**

**function two(){**

**one();**

**console.log("two");}**

**function one(){**

**console.log("one");**

**} // ! 1st will run**

**console.log(`$`.repeat(10))**

**// ! 1st-will run**

**console.log("one");**

**// ! 2nd-will run**

**console.log("two");**

**// ! 3rd-will run**

**console.log("three");**

**// ! 4th-will run**

**console.log("four");**

## Event Loop And Callback Queue

**-Threads mean:**

**- Is A Single Threaded Language "All Operations Executed in Single Thread"**

**-Call Stack Track All Calls ( to see if any function left to call or all function is called)**

**-Every Function Is Done Its Poped Out**

**-When You Call Asynchronous Function It Sent To Browser API // ! Browser API Act As A Second Thread**

**-Asynchronous Function Like Settimeout Start Its Own Thread // ! bec it’s asynchoronous and Browser-api is working as asynchoronous**

**- API Finish Waiting And Send Back The Function For Processing // ! synchoronous**

**-Browser API Add The Callback To Callback Queue // ! added as asynchoronus**

**-Event Loop Wait For Call Stack To Be Empty // ! means there's no function to call (all functions finished running)**

**-Event Loop Get Callback From Callback Queue And Add It To Call Stack**

**-Callback Queue Follow FIFO "First In First Out" Rule // ! means 1st wil be written it will be 1st to run**

**Ex:**

**console.log("one")**

**will run 1st**

**setTimeout(function () { console.log("three") },**

**setTimeout(function () { console.log("four") },**

**console.log("two")**

**will run 2nd**

**one -> two -> three -> four and this prove that asynchoronus First In First Out**

**testing undeclared variable in asynchoronus function**

**Note:**

**if you print variable made by let it will result as error**

**console.log(myvar) // ? output = error(undeclared variable)**

**let myvar = 100;**

**myvar+=100;**

**using asynchoronus**

**setTimeout(() => {**

**console.log(myvar) //  ? output = 200**

**}, 0);**

**let myvar = 100;**

**myvar+=100;**

## What Is AJAX || Network Information

**-Asynchronous And XML**

**-Approach To Use Many Technologies Together [HTML, CSS, Js, DOM] means Asynchronous**

**-It Use "XMLHttpRequest" Object To Interact With The Server**

**-You Can Fetch Data Or Send Data Without Page Refresh // ! for example: writing something on google drive and he save it without refresh**

**Status Code:**

**to make sure the response is okay for Ex: Status Code:200 this means ok & Status Code:404 this means not-found**

**Syntax:**

**variable = new XMLHttpRequest();**

**Note: it's an object**

**console.log(variable);**

**Ex:**

**let myreq = new XMLHttpRequest();**

**console.log(myreq);**

**Status-code link:**

[**https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#information\_responses**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#information_responses)

## Request || Response From Real API

**Ready-state-code:**

**[0] Request Not Initialized**

**[1] Server Connection Established**

**[2] Request Received**

**[3] Processing Request**

**[4] Request Is Finished And Response Is Ready**

**Note:**

**the request can have error and finished so how to check see the status code then if he is 200 then it’s ok the requset is compeleted if it 404 then it’s an error :)**

**status-code:**

**200->mean it’s ok**

**404->mean not found**

**syntax:**

**declaring the request**

**variable = new XMLHttpRequest();**

**to get metods**

**console.log(variable);**

**to get data from api**

**variable.open("GET", "api\_url");**

**to send the requset**

**variable.send();**

**to print the data**

**console.log(varaible.response);**

**to print how the ready-state change**

**variable.onreadystatechange = function (){task}**

**for more method print the variable**

**console.log(variable);**

**Ex:**

**let MyReq = new XMLHttpRequest();**

**console.log(MyReq);**

**MyReq.open("Get","https://api.github.com/users/elzerowebschool/repos", true);**

**MyReq.send()**

**console.log(MyReq)**

**MyReq.onreadystatechange = function () {**

**console.log(this.readyState)**

**console.log(this.response)**

**console.log(this.status);**

**if (this.readyState === 4 && this.status === 200) {**

**document.body.innerHTML = this.response}}**

# Generator

## Generator Function Introduction

**function\* name(){yield }**

**varaible = function\_name**

**console.log(varaible.next());=> to go to the next data**

**Note:**

**when Done = true & value = undefined => generator-function is compeleted**

**otherwise it will still generate**

**Ex:**

**function\* GenerateNumber() {**

**yield 1**

**console.log(`yield-1 finished`);**

**yield 2**

**console.log(`yield-2 finished`);**

**yield 3**

**console.log(`yield-3 finished`);**

**yield 4**

**console.log(`yield-4 finished`);**

**yield 5**

**console.log(`yield-5 finished`);**

**yield 6**

**console.log(`yield-6 finished`);**

**}**

**creating a varible of generator function**

**let gen1 = GenerateNumber()**

**Yielding (moving to the next)**

**console.log(gen1.next())**

**console.log(gen1.next())**

**console.log(gen1.next())**

**console.log(gen1.next())**

**console.log(gen1.next())**

**console.log(gen1.next())**

**console.log(gen1.next())**

**// ! iterating(looping) on the data of gen\_function**

**for (let value of GenerateNumber()){**

**console.log(value);**

**}**

## Delegate Generator Function

**function\*name-1(){**

**yield data**

**}**

**function\*name-2(){**

**yield data**

**}**

**function\*name-3(){**

**yield\*name-one()**

**yield\*name-2()**

**yield\*data || array**

**}**

**variable = name-3()**

**console.log(name-3().next);**

**console.log(name-3().return("string"));**

**Ex:**

**Declaring Number\_generator function**

**function\*generateNumbers(){**

**yield 1**

**yield 2**

**yield 3**

**}**

**Declaring letter\_generator function**

**function\*generateLetters(){**

**yield "A"**

**yield "B"**

**yield "C"**

**}**

**Declaring generate\_all function**

**function\*generateAll(){**

**yield\*generateNumbers()**

**yield\*generateLetters()**

**yield\*[4,5,6]**

**}**

**declaring-variable for the function**

**let generator = generateAll()**

**console.log(generator.next())**

**console.log(generator.next())**

**console.log(generator.next())**

**console.log(generator.next())**

**console.log(generator.next())**

**console.log(generator.return(`a7a kfaya`))**

**console.log(generator.next())**

**console.log(generator.next())**

**console.log(generator.next())**

## Generate Infinite Numbers

**function\* InfiniteNumber(){**

**let counter = 0**

**while(true){yield index ++}**

**}**

**depend on what number you will generate**

**for ( let i = 0; i < number you want; i++ ){**

**console.log(generator.next())**

**}**

**Ex:**

**function\* InfiniteNumber(){**

**intiate-counter**

**let index = 0**

**infinte-generator**

**while(true){**

**yield index ++**

**}**

**}**

**creating variable-generator**

**let generator = InfiniteNumber()**

**Generating numbers**

**for ( let i = 0; i < 11; i++ )**

**{ console.log(generator.next())}**

# Modules

## very important Note:

**to work with modules use live-server to avoid any error**

**before anything go to html file specifically in script-tag and write type=”modules”**

**exporting:**

**mean to extract data to put it in another file**

**importing:**

**mean to use data from another file**

## Modules Import And Export

**Exporting**

**1st-method**

**export variable=data**

**export function (){}; export array=[];**

**2nd-method**

**variable=data**

**function (){}**

**array=[]**

**export{variable, function, array }**

**importing**

**import{data you export it as name you like} from “./file-name.js”**

**Ex:**

**1st-method**

**export let a = 10;**

**export let b = 20;**

**export function add(a,b){**

**return `the total is ${a + b}`;**

**}**

**2nd-method:**

**let a = 10;**

**let b = 20;**

**function add(a,b){**

**return `the total is ${a + b}`;**

**}**

**export {a,b,add}**

## Named vs Default Export And Import All

**Export Alias & Named Export:**

**variable= data**

**arr = []**

**function(){}**

**export { variable as name as you want, arr as name as you want function as name as you want }**

## Default Export & importing it

**export default data**

**import default=>(name as you want) , {other-data} from “./another-file.js”**

## Import All

**Import \* as any-name from “./another-file.js”**

**console.log(any-name from.it’s-data)**

**Ex:**

**main.js**

**let a = 10;**

**let arr = [1, 2, 3, 4];**

**function saySomething() {return `Something`;}**

**export { a as x, arr as y, saySomething as say }**

**export default function ana (){**

**return`ana t3bt wallahy`**

**}**

**app.js**

**import ana , {x,y,say} from "./main.js"**

**console.log(x)**

**console.log(y)**

**console.log(say())**

**console.log(ana());**

**import \* as all from "./main.js"**

**console.log(all);**

**console.log(all.x);**

**console.log(all.y);**

**console.log(all.say());**

**console.log(all.default());**

# Promise and last part

## Callback Hell Or Pyramid Of Doom

**-What Is Callback:A Function That Is Passed Into Another One As An Argument To Be Executed Later**

**-Callback Hell Or Pyramid Of Doom: too many function calledback inside each other && and it’s solution is to use promise**

**-Note:there order won't affect the operation and by this it will run one by one (synchoronus)**

**Note: it’s just a problem**

**Ex:**

**Callback Hell Example**

**setTimeout(() => {**

**console.log("Download Photo From URL");**

**setTimeout(() => {**

**console.log("Resize Photo");**

**setTimeout(() => {**

**console.log("Add Logo To The Photo");**

**setTimeout(() => {**

**console.log("Show The Photo In Website");**

**}, 4000);**

**},3000);**

**}, 2000);**

**}, 1000);**

## Promise Intro || Syntax

**Note:for theoritcal go to the folder**

**Promise Status**

**Note:**

**Pending: Initial State**

**Fulfilled: Completed Successfully**

**Rejected: Failed**

**Syntax:**

**variable = new promise (function(resolved(name as you want) , rejected(name as you want)){**

**condition**

**true-> resolved**

**false-> rejected**

**})).then(function(in case of fulfilled),function(in case of rejected))**

**let in case of fulfilled = function(){specific task}**

**let in case of rejected = function(){specific task}**

**Ex:**

**let fulfilled = function () {**

**console.log(`promise is {fullfiled}`) }**

**let rejected = function () {**

**console.log(`promise is rejected`) }**

**let promise = new Promise(function (succss\_function, failure\_function) {**

**let connect = true**

**if (connect) {**

**succss\_function("yabn el mara")**

**} else {**

**failure\_function("braaaaaaa")**

**}**

**}).then(fulfilled, rejected)**

**console.log(promise)**

## Promise - Then, Catch And Finally

**Note:For theoritcal go to it’s folder**

**Syntax:**

**variable = new Promise((resolve,reject)**

**{**

**condition }).then(fulfilled,rejected){**

**}.catch(reason=>()).finally({message})**

**Ex:**

**let empolyees = []**

**let promise = new Promise(function (resolved, rejected) {**

**if (empolyees.length === 4) {**

**resolved(`the empolyees is 4`);**

**} else {**

**rejected(Error`the empolyees is not 4`);**

**}**

**});**

**promise.then(function fulfilled() {**

**empolyees.length = 2**

**console.log(`the chosen two is ${empolyees.length}`)**

**}).then(function fulfilled() {**

**empolyees.length = 1**

**console.log(`the chosen one is ${empolyees.length}`)**

**return empolyees**

**}).catch(reason=>("the empolyee is not 4")).finally(console.log(`the operation is done**

**without error `))**

## Promise And XHR:

**Just a practical video about promise && request and response of api**

## Fetch API

**>composed of Xmlhttprequest() && promise**

**>it return a new promise and the entire response**

**Fetch(“api-link).then((function-as you want){task})**

**Ex:**

**fetch("https://api.github.com/users/elzerowebschool/repos")**

**.then((result)=>{**

**let data = result.json()**

**return data**

**}).then((data)=>{**

**data.length =10**

**console.log(data)**

**return data**

**})**

**.then((data)=>{**

**document.write(data[0].name);})**

## Promise All And All Settled And Race

**all:**

**when he gp through give promisies he coutinue and if he found one is rejected he will stop and return it alone as a rejected promise**

**all-settled:**

**he coutinue and if he found one is rejected doesn't affected by it and when he finish he will return all neither rejected or resolved**

**race:  
the 1st one he see return**

**Syntax:**

**promise.all([promises]).then()**

**promise.allsettled([promises]).then()**

**promise.Race([promises]).then()**

**Ex:**

**let mypromiseone = new Promise((resolved,rejected)=>{**

**setTimeout(() => {**

**resolved("iam the 1st promise ")**

**}, 5000);**

**})**

**let mypromisetwo = new Promise((resolved,rejected)=>{**

**resolved("iam the two promise ")**

**})**

**let mypromisethree = new Promise((resolved,rejected)=>{**

**resolved("iam the three promise ")**

**})**

**Promise.all([mypromiseone,mypromisetwo,mypromisethree]).then((result)=>{**

**console.log(result)**

**})**

**Promise.allSettled([mypromiseone,mypromisetwo,mypromisethree]).then((result)=>{**

**console.log(result)**

**})**

**Promise.race([mypromiseone,mypromisetwo,mypromisethree]).then((result)=>{**

**console.log(result)**

**})**

**Ex:**

**function Get\_data() {**

**return new Promise((res, rej) => {**

**let user = ["mohamed"]**

**if (user.length > 0) {res("Users found")**

**}else{**

**rej(Error`no users found`)**

**}})}**

**function Get\_data() {**

**let user = ["mohamed"]**

**if (user.length > 0) {**

**return Promise.resolve("Users found")**

**} else {**

**return Promise.reject(Error`no users found`)}}**

**console.log(Get\_data())**

**async function Get\_data() {**

**let user = ["mohamed"]**

**if (user.length > 0) {**

**return "Users found"} else {throw new Error`no users found`}}**

**console.log(Get\_data())**

## Async And Training

**-make the promise in two ways like in the videpo**

**-returning a new promsie**

**-return promise but in a chain-method => you will see bith ways in the Examples**

**-async return from the function promise and for the resolve && rejected it depend on the condition if the condition is true or not**

**-async it's written before the function**

**async function name(params) {**

**condition**

**incase of resolved => return "anything as you want"**

**incase of rejected => throw new Error "anything as you want"**

**}**

**Ex:**

**function Get\_data() {**

**return new Promise((res, rej) => {**

**let user = ["mohamed"]**

**if (user.length > 0) {res("Users found")**

**}else{**

**rej(Error`no users found`)**

**}})}**

**function Get\_data() {**

**let user = ["mohamed"]**

**if (user.length > 0) {**

**return Promise.resolve("Users found")**

**} else {**

**return Promise.reject(Error`no users found`)}}**

**console.log(Get\_data())**

**async function Get\_data() {**

**let user = ["mohamed"]**

**if (user.length > 0) {**

**return "Users found"} else {throw new Error`no users found`}}**

**console.log(Get\_data())**

## Await And Training

**Await it's work only in case of async is existed**

**Await make the function works after promise works (synchoronou)**

**Syntax:**

**variable = new promise()**

**async function name(params) {**

**condition**

**console.log(Await promise ) this means it will retunr promise 1st then the lines after it**

**task**

**}**

**Ex:**

**let My\_Promise = new Promise((resolved, rejected) => {**

**setTimeout(() => {**

**resolved("iam the good promise ") // in case of resolved**

**}, 3000);**

**})**

**async function Get\_Data() {**

**console.log("iam here before the promise") console.log(await My\_Promise)**

**console.log("iam here before the promise")**

**}**

**Get\_Data()**

## Try, Catch And Finally With Fetch

**>Try:**

**to try something you want**

**>catch:**

**to print the error message if it happend**

**>finally:**

**to print message after all finished**

**Syntaxs:**

**Try{task (as you want)}**

**catch(reason){console.log("Error: " + reason);}**

**finally{message(as you want)}**

**Ex:**

**async function Fetch\_data() {**

**console.log(`iam before the fetching`);**

**try {**

**let My\_Data = await fetch("https://api.github.com/users/elzerowebschool/repos")**

**console.log(await My\_Data.json());**

**} catch (reason) {**

**console.log(Error(reason))**

**} finally {**

**setTimeout(() => {**

**console.log("operation is finished")**

**}, 3000);**

**}**

**}**

**Fetch\_data()**