js course notes

alert

we can use it to return something in th

e page by that

()window.alert

write

(”<document.write(“<h1> hello page </h1

is we want to print something in console we can use log

(”console.log(“hello from js

console

log	`console.log("hello world");` **to write in the console**
alert	
error	`console.error("error");`
table	`console.table(["osama" , "ahmed" , "sayed"]);`
C%	`console.log("hello form %c js %c 5 " , "color :red; fontsize:40px" , "color :blue; fontsize:40px")`

Type of operators

```
/*
  Data Types Intro
  - String
  - Number
  - Array => Object
  - Object
  - Boolean
*/


console.log("Osama Mohamed");
```

```
console.log(typeof "Osama Mohamed");

console.log(typeof 5000);

console.log(typeof 5000.99);

console.log(typeof [10, 15, 17]);

console.log(typeof { name: "Osama", age: 17, country:
"Eg" });

console.log(typeof true);

console.log(typeof false);

console.log(typeof undefined);

console.log(typeof null);
```

variable

when we put something in variable it calls declare

```
/*

  Variables Intro

  - What Is Variable ?

  - Why We Use Variables ?

  - Declare A Variable And Use

  - Syntax ( Keyword | Variable Name | Assignment
Operator | Variable Value )

  - Variable Without Var

  - Multiple Variables In The Same Line

  - Id And Global Variable

  - Loosely Typed vs Strongly Typed

*/
```

```
var user = "Sayed";
console.log(user);
```

the rules of declaring variables

we can’t start it with a number

$ we can’t put a special symbol in the variable else

we can’t put a space

```
/*
  Identifiers
  - Name Conventions And Rules
  - Reserved Words
*/

var userName = "Sayed";

console.log(userName);
```

Var, Let, Const Compare

```
/*
  Var
  - Redeclare (Yes)
  - Access Before Declare (Undefined)
  - Variable Scope Drama [Added To Window] ()
  - Block Or Scope Function
```

```
  Let
  - Redeclare (No => Error)
  - Access Before Declare (Error)
  - Variable Scope Drama ()
  - Block Or Scope Function

  Const
  - Redeclare (No => Error)
  - Access Before Declare (Error)
  - Variable Scope Drama ()
  - Block Or Scope Function
*/
```

014 - String Syntax And Characters Escape Sequences

```
/*
  String Syntax + Character Escape Sequences
  \ Escape + Line Continue
  \n
*/

console.log('Elzero Web "School"');
console.log("Elzero Web 'School'");
```

```
console.log("Elzero Web \"School\"");
console.log('Elzero \\ Web \'School\'');
console.log("Elzero \
Web \
School");
console.log("Elzero\nWeb\nSchool");
```

015 - Concatenation

ال Code الموجود في الدرس

```
/*
  Concatenation
*/

let a = "We Love";
let b = "JavaScript";

document.write(a + " " + b);

console.log(a, b);
```

016 - Template Literals (Template Strings)

ال Code الموجود في الدرس

```
/*
  Template Literals (Template Strings)
*/

// First Example

let a = "We Love";
let b = "JavaScript";
let c = "And";
let d = "Programming";

console.log(a = " \"\" " + b +
"\n" + c + " " + d);

console.log(`${a} "" '' \\ ${b}
${c} ${d}`);

// Second Example

let title = "Elzero";
let desc = "Elzero Web School";

let markup = `
```

```
  <div class="card">
    <div class="child">
      <h2>${title}</h2>
      <p>${desc}</p>
    </div>
  </div>
`;

document.write(markup);
```

018 - Arithmetic Operators

ال Code الموجود في الدرس

```
/*
  Arithmetic Operators
  + Addition
  - Subtraction
  * Multiplication
  / Division
  ** Exponentiation (ES7)
  % Modulus (Division Remainder)
  ++ Increment [ Post / Pre ]
  -- Decrement [ Post / Pre ]
```

```
*/

console.log(10 + 20);
console.log(10 + "Osama");

console.log(10 - 20);
console.log(10 - "Osama"); // NaN
console.log(typeof NaN);

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

console.log(20 / 5);
console.log(20 / 3);

console.log(2 ** 4);
console.log(2 * 2 * 2 * 2);

console.log(10 % 2);
console.log(11 % 2); // Remove 1
```

019 - Unary Plus And Negation Operators

ال Code الموجود في الدرس

```
/*
  - + Unary Plus [Return Number If Its Not Number]
  - - Unary Negation [Return Number If Its Not Number
+ Negates It]
  Tests
  - Normal Number
  - String Number
  - String Negative Number
  - String Text
  - Float
  - Hexadecimal Numeral System => 0xFF
  - null
  - false
  - true
*/

console.log(+100);
console.log(+"100");
console.log(+"-100");
console.log(+"Osama");
console.log(+"15.5");
console.log(+0xff);
console.log(+null);
```

```
console.log(+false);
console.log(+true);

console.log(-100);
console.log(-"100");
console.log(-"-100");
console.log(-"Osama");
console.log(-"15.5");
console.log(-0xff);
console.log(-null);
console.log(-false);
console.log(-true);

console.log(Number("100"));
```

020 - Type Coercion

ال Code الموجود في الدرس

```
/*
  Type Coercion (Type Casting)
  - +
  - -
  - "" - 2
```

```
  - false - true
*/

let a = "100";
let b = 20;
let c = true;

console.log(+a + b + c);
```

021 - Assignment Operators

ال Code الموجود في الدرس

```
/*
  Assignment Operators
*/

let a = 10;

a = a + 20;

a = a + 70;

a += 100; // a = a + 100
```

```
a -= 50; // a = a - 50

a /= 50; // a = a / 50
```

023 - Number

ال Code الموجود في الدرس

```
/*
  Number
  - Double Precision
  - Syntactic Sugar "_"
  - e
  - **
  - With Decimal
  - Number + BigInt
  - Number Min Value
  - Number Max Value
*/

console.log(1000000);
console.log(1_000_000);
console.log(1e6);
console.log(10 ** 6);
```

```
console.log(10 * 10 * 10 * 10 * 10 * 10);
console.log(1000000.0);

console.log(Number.MAX_SAFE_INTEGER);
console.log(Number.MAX_VALUE);
console.log(Number.MAX_VALUE + 23434343434);
```

024 - Number Methods

ال Code الموجود في الدرس

```
/*
  Number Methods
  - Two Dots To Call A Methods
  - toString()
  - toFixed()
  - parseInt()
  - parseFloat()
  - isInteger() [ES6]
  - isNaN() [ES6]
*/
//we use to string to turn int to string by the two ways
console.log((100).toString());
console.log(100.10.toString());
```

```
// we use it to adapt the long of a float number

console.log(100.554555.toFixed(2));

If we did that it will return NAN because it can’t
separate the string

console.log(Number("100 Osama"));

the sam

console.log(+"100 Osama");

//It will return 100 because (parseint have some
intelligent and and can separater the number from the
string )

console.log(parseInt("100 Osama"));

// it will return NUN because it too complicated

console.log(parseInt("Osama 100 Osama"));

// it will return 100 because it return just integar

console.log(parseInt("100.500 Osama"));

// it will return the number because it can recognize
the float number

console.log(parseFloat("100.500 Osama"));

// isIntegar will return int if the input is a number
of not

// it will return false

console.log(Number.isInteger("100"));

// it will return false because it is float

console.log(Number.isInteger(100.500));



// it will return true
```

```
console.log(Number.isInteger(100));

// isNan designed to recognize that the input isn’t number

console.log(Number.isNaN("Osama" / 20));
```

```
/*
  Math Object
  - round()
  - ceil()
  - floor()
  - min()
  - max()
  - pow()
  - random()
  - trunc() [Es6]
*/

// round , it round the number to the nearest integar

// it will return (99)

console.log(Math.round(99.2));

// it will return (100)

console.log(Math.round(99.5));


// The Math.ceil() method rounds a number UPWARDS to the nearest integer, and returns the result
```

```
console.log(Math.ceil(99.2));
```

The **`Math.floor()`** function returns the largest integer less than or equal to a given number.

```
console.log(Math.floor(99.9));
// Math.min return the smallest number in the list
console.log(Math.min(10, 20, 100, -100, 90));
// Math.min return the largest number in the list
console.log(Math.max(10, 20, 100, -100, 90));
// It like 2 ** 4
console.log(Math.pow(2, 4));
it will retrun a random number
console.log(Math.random());
// it ignore the float number and
console.log(Math.trunc(99.5));
```

number challenge

```
/*
  Number Challenge
*/

let a = 100;
let b = 2_00.5;
let c = 1e2;
let d = 2.4;

// Find Smallest Number In All Variables And Return Integer
console.log(Math.ceil(Math.min(a , b , c , d)));

// Use Variables a + d One Time To Get The Needed Output
console.log(100 * a + d*0); // 10000

// Get Integer "2" From d Variable With 4 Methods
console.log(Math.round(d));
```

```
console.log(Math.floor(d));
console.log(Math.trunc(d));
console.log(+ d.toFixed(0));

// Use Variables b + d To Get This Valus
console.log(+(b * 0 + (d * 27.7791)).toFixed(2)); // 66.67 => String
console.log(Math.ceil(+(b * 0 + (d * 27.7791)).toFixed(2))); // 67 => Number
```

027 - String Methods Part 1

ال Code الموجود في الدرس

```
/*
  String Methods
  - Access With Index
  - Access With charAt()
  - length
  - trim()
  - toUpperCase()
  - toLowerCase()
  - Chain Methods
*/

let theName = "  Ahmed  ";

console.log(theName);
console.log(theName[1]);
console.log(theName[5]);
```

```
we use cahr at to aim at a specific char in string
console.log(theName.charAt(1));
console.log(theName.charAt(5));
we use length to know the length of the string
console.log(theName.length);
we use trim to delete spaces
console.log(theName.trim());
to turn all the string to upper letters
console.log(theName.toUpperCase());
to turn all the string to lower letters
console.log(theName.toLowerCase());

console.log(theName.trim().charAt(2).toUpperCase());
```

028 - String Methods Part 2

ال Code الموجود في الدرس

```
/*
  String Methods
  - indexOf(Value [Mand], Start [Opt] 0)
  - lastIndexOf(Value [Mand], Start [Opt] Length)
  - slice(Start [Mand], End [Opt] Not Include End)
```

```
  - repeat(Times) [ES6]

  - split(Separator [Opt], Limit [Opt])

*/


let a = "Elzero Web School";

you put an input in index of function and it will return its place in the array

console.log(a.indexOf("Web"));

and you can also second input to tell where it must start from

console.log(a.indexOf("Web", 8));

console.log(a.indexOf("o")); // 5

when we use it start searching form the opposite side

console.log(a.lastIndexOf("o")); // 15

slice we use to cut a specific peace in the string

console.log(a.slice(2, 6));

console.log(a.slice(-5, -3));

we use repeat like for loop

console.log(a.repeat(5));

split turn the string to array and split any character here and it depend on the input and you can tell how many time you want to split

console.log(a.split("", 6));
```

029 - String Methods Part 3

ال Code الموجود في الدرس

```
/*
  String Methods
  - substring(Start [Mand], End [Opt] Not Including
End)
  --- Start > End Will Swap
  --- Start < 0 It Start From 0
  --- Use Length To Get Last Character
  - substr(Start [Mand], Characters To Extract)
  --- Start >= Length = ""
  --- Negative Start From End
  - includes(Value [Mand], Start [Opt] Default 0)
[ES6]
  - startsWith(Value [Mand], Start [Opt] Default 0)
[ES6]
  - endsWith(Value [Mand], Length [Opt] Default Full
Length) [ES6]
*/


let a = "Elzero Web School";


console.log(a.length);
it is like slide and if we put the start by a negative
number It will start from zero, if you put a start
bigger than end It will swap them(2 , 6) --   (6,2)
console.log(a.substring(2, 6));
```

```
console.log(a.substring(6, 2));

console.log(a.substring(-10, 6)); // 0 - 6

console.log(a.substring(a.length - 5, a.length - 3));
```

it doesn’t have end you just put to it the start and it will extract the string from this point , if the start is bigger than the length of the string it will return nothing

```
console.log(a.substr(0, 6));

console.log(a.substr(17));

console.log(a.substr(-3));

console.log(a.substr(-5, 2));
```

it check if the string include the input you can also tell it where it must start from

```
console.log(a.includes("Web"));

console.log(a.includes("Web", 8));
```

It check if the string start with a specific character you put

```
console.log(a.startsWith("E"));

console.log(a.startsWith("E", 2));

console.log(a.startsWith("zero", 2));
```

it checks if the string end with a specific character or not

```
console.log(a.endsWith("l"));
```

031 - Comparison Operators

ال Code الموجود في الدرس

```
/*
  Comparison Operators
  - == Equal
  - != Not Equal

  - === Identical
  - !== Not Identical

  - > Larger Than
  - >= Larger Than Or Equal

  - < Smaller Than
  - <= Smaller Than Or Equal
*/

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");
```

032 - Logical Operators

ال Code الموجود في الدرس

```
/*
  Logical Operators
  - ! Not
  - && And
  - || Or
*/

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

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

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

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

033 - If Conditions

ال Code الموجود في الدرس

```
/*
  Control Flow
  - if
  - else if
  - else

  if (Condition) {
    // Block Of Code
  }

*/

let price = 100;
```

```
let discount = true;

let discountAmount = 30;

let country = "KSA";


if (discount === true) {

  price -= discountAmount; // price = price -
discountAmount

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

  price -= 40;

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

  price -= 50;

} else {

  price -= 10;

}


console.log(price);
```

034 - Nested If Conditions

ال Code الموجود في الدرس

```
/*

  Nested If

*/
```

```
let price = 100;
let discount = false;
let discountAmount = 30;
let country = "Egypt";
let student = true;

if (discount === true) {

  price -= discountAmount;

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

  if (student === true) {

    price -= discountAmount + 30;

  } else {

    price -= discountAmount + 10;

  }

} else {
```

```
  price -= 10;


}


console.log(price);
```

035 - Conditional Ternary Operator

ال Code الموجود في الدرس

```
/*
  Conditional (Ternary) Operator
*/


let theName = "Mona";
let theGender = "Female";
let theAge = 30;


if (theGender === "Male") {
  console.log("Mr");
} else {
  console.log("Mrs");
}
```

```
// Condition ? If True : If False


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


let result = theGender === "Male" ? "Mr" : "Mrs";


document.write(result);


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


console.log(`Hello ${theGender === "Male" ? "Mr" :
"Mrs"} ${theName}`);


theAge < 20
  ? console.log(20)
  : theAge > 20 && theAge < 60
  ? console.log("20 To 60")
  : theAge > 60
  ? console.log("Larger Than 60")
  : console.log("Unknown");
```

036 - Nullish Coalescing Operator & Logical Or

ال Code الموجود في الدرس

```
/*
  Logical Or ||
  -- Null + Undefined + Any Falsy Value
  Nullish Coalescing Operator ??
  -- Null + Undefined
*/

console.log(Boolean(100));
console.log(Boolean(-100));
console.log(Boolean(0));
console.log(Boolean(""));
console.log(Boolean(null));

let price = 0;

console.log(`The Price Is ${price || 200}`);
console.log(`The Price Is ${price ?? 200}`);
```

037 - If Condition Challenge

التحدي الموجود في الدرس

```
/*
  If Condition Challenge
*/

let a = 10;

if (a < 10) {
  console.log(10);
} else if (a >= 10 && a <= 40) {
  console.log("10 To 40");
} else if (a > 40) {
  console.log("> 40");
} else {
  console.log("Unknown");
}

// Write Previous Condition With Ternary If Syntax

let st = "Elzero Web School";

// W Position May Change
if ("????" === "w") {
```

```
  console.log("Good");
}

if ("????" !== "string") {
  console.log("Good");
}

if ("????" === "number") {
  console.log("Good");
}

if ("????" === "ElzeroElzero") {
  console.log("Good");
}
```

038 - Switch Statement

ال Code الموجود في الدرس

```
/*
  Switch Statement
  switch(expression) {
    Case 1:
      // Code Block
```

```
      break;
    Case 2:
      // Code Block
      break;
    Default:
      // Code Block
  }
  - Default Ordering
  - Multiple Match
  - ===
*/

let day = "A";

switch (day) {
  default:
    console.log("Unknown Day");
    break;
  case 0:
    console.log("Saturday");
    break;
  case 1:
    console.log("Sunday");
    break;
```

```
  case 2:
  case 3:
    console.log("Monday");
    break;
}
```

The first challenge

```
/*
  If Condition Challenge
*/

let a = 10;

if (a < 10) {
  console.log(10);
} else if (a >= 10 && a <= 40) {
  console.log("10 To 40");
} else if (a > 40) {
  console.log("> 40");
} else {
  console.log("Unknown");
}

a < 10 ? console.log(10): a>=10 && a <= 40 ? console.log("10 To 40"):a > 40 ?
console.log(">40") : console.log("UnKnown");

let st = "Elzero Web School";

// W Position May Change
if (st.includes("W") ) {
  console.log("Good");
}


if (st !== "string") {
  console.log("Good");
}

if ( typeof st === typeof "number") {
  console.log("Good");
```

```
}

if (st.split(" " ,1) + st.split(" " ,1) === "ElzeroElzero") {
  console.log("Good");
}
```

039 - Switch And If Condition Challenge

التحدي الموجود في الدرس

```
/*
  Switch Challenge
*/


let job = "Manager";
let salary = 0;


if (job === "Manager") {
  salary = 8000;
} else if (job === "IT" || job === "Support") {
  salary = 6000;
} else if (job === "Developer" || job === "Designer")
{
  salary = 7000;
} else {
  salary = 4000;
}
```

```
/*
  If Challenge
*/

let holidays = 0;
let money = 0;

switch (holidays) {
  case 0:
    money = 5000;
    console.log(`My Money is ${money}`);
    break;
  case 1:
  case 2:
    money = 3000;
    console.log(`My Money is ${money}`);
    break;
  case 3:
    money = 2000;
    console.log(`My Money is ${money}`);
    break;
  case 4:
    money = 1000;
```

```
    console.log(`My Money is ${money}`);
    break;
  case 5:
    money = 0;
    console.log(`My Money is ${money}`);
    break;
  default:
    money = 0;
    console.log(`My Money is ${money}`);
}
```

040 - Array Big Introduction

ال Code الموجود في الدرس

```
/*
  Arrays
  - Create Arrays [Two Methods] new Array() + []
  - Access Arrays Elements
  - Nested Arrays
  - Change Arrays Elements
  - Check For Array Array.isArray(arr);
*/
```

```
let myFriends = ["Ahmed", "Mohamed", "Sayed",
["Marwan", "Ali"]];


console.log(`Hello ${myFriends[0]}`);

console.log(`Hello ${myFriends[2]}`);

console.log(`${myFriends[1][2]}`);

console.log(`Hello ${myFriends[3][1]}`);

console.log(`${myFriends[3][1][2]}`);


console.log(myFriends);

myFriends[1] = "Gamal";

console.log(myFriends);

myFriends[3] = ["Sameh", "Ameer"];

console.log(myFriends);


console.log(Array.isArray(myFriends));
```

041 - Using Length With Array

ال Code الموجود في الدرس

```
/*
  Array Methods
  - Length
*/
```

```
// Index Start From 0 [ 0, 1, 2, 3, 4 ]

let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa"];

myFriends.length = 2;

console.log(myFriends);
```

042 - Add And Remove From Array

ال Code الموجود في الدرس

```
/*
  Arrays Methods [Adding And Removing]
  - unshift("", "") Add Element To The First
  - push("", "") Add Element To The End
  - shift() Remove First Element From Array
  - pop() Remove Last Element From Array
*/

let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa"];

console.log(myFriends);
we use unshift to put things at the first of the array
```

```
myFriends.unshift("Osama", "Nabil");

console.log(myFriends);
we use it to put inputs at the end of the array
myFriends.push("Samah", "Eman");

console.log(myFriends);
we use shift to delete the first in put in the array
and it can store it
let first = myFriends.shift();

console.log(myFriends);

console.log(`First Name Is ${first}`);
we use pop to delete inputs from the end of the array
let last = myFriends.pop();

console.log(myFriends);

console.log(`Last Name Is ${last}`);
```

043 - Searching Array

ال Code الموجود في الدرس

```
/*
  Arrays Methods [Search]
  - indexOf(Search Element, From Index [Opt])
  - lastIndexOf(Search Element, From Index [Opt])
  - includes(valueToFind, fromIndex [Opt]) [ES7]
*/


let myFriends = ["Ahmed", "Mohamed", "Sayed", "Alaa",
"Ahmed"];


console.log(myFriends);

we use indexof to search in the array for a specific
input and It return its number in the array

console.log(myFriends.indexOf("Ahmed"));

console.log(myFriends.indexOf("Ahmed", 2));

the same of index of but it start from the opposite
side

console.log(myFriends.lastIndexOf("Ahmed"));

console.log(myFriends.lastIndexOf("Ahmed", -2));

includes checks if the it input are exist in the array
or not

console.log(myFriends.includes("Ahmed"));

console.log(myFriends.includes("Ahmed", 2));


if (myFriends.lastIndexOf("Osama") === -1) {
```

```
  console.log("Not Found");
}


console.log(myFriends.indexOf("Osama"));
console.log(myFriends.lastIndexOf("Osama"));
```

044 - Sorting Array

ال Code الموجود في الدرس

```
/*
  Arrays Methods [Sort]
  - sort(Function [Opt])
  - reverse
*/


let myFriends = [10, "Sayed", "Mohamed", "90", 9000,
100, 20, "10", -20, -10];


console.log(myFriends);
console.log(myFriends.sort().reverse());
```

045 - Slicing Array

ال Code الموجود في الدرس

```
/*
  Arrays Methods [Slicing]
  - slice(Start [Opt], End [Opt] Not Including End)
  --- slice() => All Array
  --- If Start Is Undefined => 0
  --- Negative Count From End
  --- If End Is Undefined || > Indexes => Slice To The End Array.length
  --- Return New Array
  - splice(Start [Mand], DeleteCount [Opt] [0 No Remove], The Items To Add [Opt])
  --- If Negative => Start From The End
*/


let myFriends = ["Ahmed", "Sayed", "Ali", "Osama", "Gamal", "Ameer"];
you can cut the array with slice method it apply two inputs
first : start
second: end (it isn’t include end)
console.log(myFriends);
console.log(myFriends.slice());
console.log(myFriends.slice(1));
console.log(myFriends.slice(1, 3));
console.log(myFriends.slice(-3));
```

```
console.log(myFriends.slice(1, -2));

console.log(myFriends.slice(-4, -2));

console.log(myFriends);

when we have to put three inputs

1) the start

2)the number of indexes you want to delete

3) inputs you want to put in the first

myFriends.splice(1, 2, "Sameer", "Samara");


console.log(myFriends);
```

046 - Joining Arrays

ال Code الموجود في الدرس

```
/*
  Arrays Methods [Joining]
  - concat(array, array) => Return A New Array
  - join(Separator)
*/


let myFriends = ["Ahmed", "Sayed", "Ali", "Osama",
"Gamal", "Ameer"];

let myNewFriends = ["Samar", "Sameh"];

let schoolFriends = ["Haytham", "Shady"];
```

```
concat can merge many arrays in one array

let allFriends = myFriends.concat(myNewFriends,
schoolFriends, "Gameel", [1, 2]);


console.log(allFriends);

join (can turn the whole array to string and you can
specify what is the separator )

console.log(allFriends.join());

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

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

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

console.log(allFriends.join("|").toUpperCase());
```

048 - Loop – For and The Concept Of Loop

ال Code الموجود في الدرس

```
/*
  Loop
  - For
  for ([1] [2] [3]) {
    // Block Of Code
  }
*/
```

```
for (let i = 0; i < 10; i++) {
  console.log(i);
}
```

049 - Loop On Sequences

ال Code الموجود في الدرس

```
/*
  Loop
  - Loop On Sequences
*/

let myFriends = [1, 2, "Osama", "Ahmed", 3, 4,
"Sayed", 6, "Ali"];

let onlyNames = [];

for (let i = 0; i < myFriends.length; i++) {
  if (typeof myFriends[i] === "string") {
    onlyNames.push(myFriends[i]);
  }
}
```

```
console.log(onlyNames);


// console.log(myFriends[0]);
// console.log(myFriends[1]);
// console.log(myFriends[2]);
// console.log(myFriends[3]);
// console.log(myFriends[4]);


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

050 - Nested Loops And Trainings

ال Code الموجود في الدرس

```
/*
  Loop
  - Nested Loops
*/


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

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

let models = [2020, 2021];

for (let i = 0; i < products.length; i++) {
  console.log("#".repeat(15));
  console.log(`# ${products[i]}`);
  console.log("#".repeat(15));
  console.log("Colors: ");
  for (let j = 0; j < colors.length; j++) {
    console.log(`- ${colors[j]}`);
  }
  console.log("Models: ");
  for (let k = 0; k < models.length; k++) {
    console.log(`- ${models[k]}`);
  }
}
```

052 - Loop For – Advanced Example

ال Code الموجود في الدرس

```
/*
  Loop For Advanced Example
```

```
*/


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

let i = 0;


for (;;) {

  console.log(products[i]);

  i++;

  if (i === products.length) break;

}
```

053 - Practice – Add Products To Page

ال Code الموجود في الدرس

```
/*

  Products Practice

*/


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

let colors = ["Red", "Green", "Blue"];

let showCount = 3;
```

```
document.write(`<h1>Show ${showCount} Products</h1>`);


for (let i = 0; i < showCount; i++) {
  document.write(`<div>`);
  document.write(`<h3>[${i + 1}] ${products[i]}</h3>`);
  for (let j = 0; j < colors.length; j++) {
    document.write(`<p>${colors[j]}</p>`);
  }
  document.write(`<p>${colors.join(" | ")}</p>`);
  document.write(`</div>`);
}
```

055 - Loop – Do, While

ال Code الموجود في الدرس

```
/*
  Loop
  - Do / While
*/


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

```
let i = 0;


do {
  console.log(i);
  i++;
} while (false);


console.log(i);
```

The challenge

```
/*
  Loop Challenge
*/

let myAdmins = ["Ahmed", "Osama", "Sayed", "Stop"];
let myEmployees = ["Amgad", "Samah", "Ameer", "Omar", "Othman", "Amany",
"Samia", "Anwar"];

document.write(`<div>We Have ${myAdmins.length} Admins</div>
<hr>
`)

for(i = 0 ; i < myAdmins.length ; i++){
document.write(`<p> Admin ${i + 1} For Team is ${myAdmins[i]}</p>`)
document.write(`<h3> Team Members </h3>`)
counter = 1
for(x = 0 ; x < myEmployees.length ; x++){
  if (myAdmins[i][0] == myEmployees[x][0]){
    document.write(`<p> - ${counter} ${myEmployees[x]} </p>`)
    counter += 1

  }
}
document.write(`<hr>`)
}
```

057 - Function Intro And Basic Usage

ال Code الموجود في الدرس

```
/*
  Function
  - What Is Function ?
  - User-Defined vs Built In
  - Syntax + Basic Usage
  - Example From Real Life
  - Parameter + Argument
  - Practical Example
*/

function sayHello(userName) {
  console.log(`Hi ${userName}`);
}

sayHello("Osama");
sayHello("Sayed");
sayHello("Ali");
```

058 - Function Advanced Example

ال Code الموجود في الدرس

```
/*
  Function Advanced Examples
*/


function sayHello(userName, age) {
  if (age < 20) {
    console.log(`App is Not Suitable For You`);
  } else {
    console.log(`Hello ${userName} Your Age is
${age}`);
  }
}


sayHello("Osama", 38);
sayHello("Sayed", 40);
sayHello("Ali", 18);


function generateYears(start, end, exclude) {
  for (let i = start; i <= end; i++) {
    if (i === exclude) {
      continue;
    }
    console.log(i);
```

```
  }
}


generateYears(1982, 2021, 2020);
```

059 - Function Return Statement And Use Cases

ال Code الموجود في الدرس

```
/*
  Function
  - Return
  - Automatic Semicolon Insertion [No Line Terminator Allowed]
  - Interrupting
*/


function generate(start, end) {
  for (let i = start; i <= end; i++) {
    if (i === 15) {
      return `Interrupting`;
    }
    console.log(i);
```

```
  }
}


generate(10, 20);
```

060 - Function Default Parameters

ال Code الموجود في الدرس

```
/*
  Function
  - Default Function Parameters
  - Function Parameters Default [Undefined]
  - Old Strategies [Condition + Logical Or]
  - ES6 Method
*/


function sayHello(username = "Unknown", age =
"Unknown") {
  // if (age === undefined) {
  //   age = "Unknown";
  // }
  // age = age || "Unknown";
  return `Hello ${username} Your Age Is ${age}`;
}
```

```
console.log(sayHello());
```

061 - Function Rest Parameters

ال Code الموجود في الدرس

```
/*
  Function
  - Rest Parameters
  - Only One Allowed
  - Must Be Last Element
*/

function calc(...numbers) {
  // console.log(Array.isArray(numbers));
  let result = 0;
  for (let i = 0; i < numbers.length; i++) {
    result += numbers[i]; // result = result +
numbers[i]
  }
  return `Final Result Is ${result}`;
}

console.log(calc(10, 20, 10, 30, 50, 30));
```

062 - Practice – Ultimate Function

ال Code الموجود في الدرس

```
/*
  Function Advanced Practice
  - Parameters
  - Default
  - Rest
  - Loop
  - Condition
*/


function showInfo(us = "Un", ag = "Un", rt = 0, show =
"Yes", ...sk) {
  document.write(`<div>`);
  document.write(`<h2>Welcome, ${us}</h2>`);
  document.write(`<p>Age: ${ag}</p>`);
  document.write(`<p>Hour Rate: $${rt}</p>`);
  if (show === "Yes") {
    if (sk.length > 0) {
      document.write(`<p>Skills: ${sk.join(" |
")}</p>`);
    } else {
```

```
      document.write(`<p>Skills: No Skills</p>`);
    }
  } else {
    document.write(`<p>Skills Is Hidden</p>`);
  }
  document.write(`</div>`);
}

showInfo("Osama", 38, 20, "No", "Html", "CSS");
```

064 - Anonymous Function and Use Cases

ال Code الموجود في الدرس

```
/*
  Function
  - Anonymous Function
  - Calling Named Function vs Anonymous Function
  - Argument To Other Function
```

```
  - Task Without Name
  - SetTimeout
*/

let calculator = function (num1, num2) {
  return num1 + num2;
};

console.log(calculator(10, 20));

function sayHello() {
  console.log("Hello Osama");
}

document.getElementById("show").onclick = sayHello;

setTimeout(function elzero() {
  console.log("Good");
}, 2000);
```

065 - Return Nested Function

ال Code الموجود في الدرس

```
/*
```

```
  Function
  - Function Inside Function
  - Return Function
*/

// Example 1

function sayMessage(fName, lName) {
  let message = `Hello`;
  // Nested Function
  function concatMsg() {
    message = `${message} ${fName} ${lName}`;
  }
  concatMsg();
  return message;
}

console.log(sayMessage("Osama", "Mohamed"));

// Example 2

function sayMessage(fName, lName) {
  let message = `Hello`;
  // Nested Function
```

```
  function concatMsg() {
    return `${message} ${fName} ${lName}`;
  }
  return concatMsg();
}

console.log(sayMessage("Osama", "Mohamed"));

// Example 3

function sayMessage(fName, lName) {
  let message = `Hello`;
  // Nested Function
  function concatMsg() {
    function getFullName() {
      return `${fName} ${lName}`;
    }
    return `${message} ${getFullName()}`;
  }
  return concatMsg();
}

console.log(sayMessage("Osama", "Mohamed"));
```

066 - Arrow Function Syntax

ال Code الموجود في الدرس

```
/*
  Function
  - Arrow Function
  -- Regular vs Arrow [Param + No Param]
  -- Multiple Lines
*/

let print = function () {
  return 10;
};

let print = () => 10;

let print = function (num) {
  return num;
};

let print = num => num;

let print = function (num1, num2) {
```

```
  return num1 + num2;
};

let print = (num1, num2) => num1 + num2;

console.log(print(100, 50));
```

067 - Scope – Global And Local

ال Code الموجود في الدرس

```
/*
  Scope
  - Global And Local Scope
*/

var a = 1;
let b = 2;

function showText() {
  var a = 10;
  let b = 20;
  console.log(`Function - From Local ${a}`);
  console.log(`Function - From Local ${b}`);
```

```
}

console.log(`From Global ${a}`);
console.log(`From Global ${b}`);

showText();
```

068 - Scope – Block

ال Code الموجود في الدرس

```
/*
  Scope
  - Block Scope [If, Switch, For]
*/

var x = 10;

if (10 === 10) {
  let x = 50;
  console.log(`From If Block ${x}`);
}

console.log(`From Global ${x}`);
```

```
/*
  Scope
  - Lexical Scope

  Search
  - Execution Context
  - Lexical Environment
*/

function parent() {
  let a = 10;

  function child() {
    console.log(a);
    console.log(`From Child ${b}`);

    function grand() {
      let b = 100;
      console.log(`From Grand ${a}`);
      console.log(`From Grand ${b}`);
    }
```

```
    grand();
  }
  child();
}
parent();
```

071 - Higher Order Functions – Map

ال Code الموجود في الدرس

```
/*
  Higher Order Functions
  ---> is a function that accepts functions as
parameters and/or returns a function.


  - Map
  --- method creates a new array
  --- populated with the results of calling a provided
function on every element
  --- in the calling array.
```

```
  Syntax map(callBackFunction(Element, Index, Array) {
}, thisArg)

  - 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


  Notes

  - Map Return A New Array


  Examples

  - Anonymous Function

  - Named Function


*/


let myNums = [1, 2, 3, 4, 5, 6];


let newArray = [];


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

  newArray.push(myNums[i] + myNums[i]);

}
```

```
console.log(newArray);


// Same Idea With Map


// let addSelf = myNums.map(function (element, index,
arr) {
//   // console.log(`Current Element => ${element}`);
//   // console.log(`Current Index => ${index}`);
//   // console.log(`Array => ${arr}`);
//   // console.log(`This => ${this}`);
//   return element + element;
// }, 10);


// let addSelf = myNums.map((a) => a + a);


// console.log(addSelf);


function addition(ele) {
  return ele + ele;
}


let add = myNums.map(addition);


console.log(add);
```

072 - Higher Order Functions – Map Practice

ال Code الموجود في الدرس

```
/*
  Map
  - Swap Cases
  - Inverted Numbers
  - Ignore Boolean Value
*/

let swappingCases = "elZERo";
let invertedNumbers = [1, -10, -20, 15, 100, -30];
let ignoreNumbers = "Elz123er4o";

// let sw = swappingCases
//   .split("")
//   .map(function (ele) {
//     // Condition ? True : False
//     return ele === ele.toUpperCase() ?
ele.toLowerCase() : ele.toUpperCase();
//   })
//   .join("");
```

```
// Arrow Function Version


let sw = swappingCases
  .split("")
  .map((a) => (a === a.toUpperCase() ? a.toLowerCase()
: a.toUpperCase()))
  .join("");


console.log(sw);


let inv = invertedNumbers.map(function (ele) {
  return -ele;
});


console.log(inv);


let ign = ignoreNumbers
  .split("")
  .map(function (ele) {
    return isNaN(parseInt(ele)) ? ele : "";
  })
  .join("");


console.log(ign);
```

073 - Higher Order Functions – Filter

ال Code الموجود في الدرس

```
/*
  - Filter
  --- method creates a new array
  --- with all elements that pass the test implemented by the provided function.


  Syntax filter(callBackFunction(Element, Index, Array) { }, thisArg)
  - 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
*/


// Get Friends With Name Starts With A
let friends = ["Ahmed", "Sameh", "Sayed", "Asmaa", "Amgad", "Israa"];


let filterdFriends = friends.filter(function (el) {
  return el.startsWith("A") ? true : false;
});
```

```
console.log(filterdFriends);

// Get Even Numbers Only
let numbers = [11, 20, 2, 5, 17, 10];

let evenNumbers = numbers.filter(function (el) {
  return el % 2 === 0;
});

console.log(evenNumbers);

// Test Map vs Filter

// let addMap = numbers.map(function (el) {
//   return el + el;
// });

// console.log(addMap);

// let addFilter = numbers.filter(function (el) {
//   return el + el;
// });
```

```
// console.log(addFilter);
```

074 - Higher Order Functions – Filter Practice

ال Code الموجود في الدرس

```
/*
  Filter
  - Filter Longest Word By Number
*/

// Filter Words More Than 4 Characters
let sentence = "I Love Foood Code Too Playing Much";

let smallWords = sentence
  .split(" ")
  .filter(function (ele) {
    return ele.length <= 4;
  })
  .join(" ");

console.log(smallWords);
```

```
// Ignore Numbers
let ignoreNumbers = "Elz123er4o";

let ign = ignoreNumbers
  .split("")
  .filter(function (ele) {
    return isNaN(parseInt(ele));
  })
  .join("");

console.log(ign);

// Filter Strings + Multiply
let mix = "A13BS2ZX";

let mixedContent = mix
  .split("")
  .filter(function (ele) {
    return !isNaN(parseInt(ele));
  })
  .map(function (ele) {
    return ele * ele;
  })
  .join("");
```

```
console.log(mixedContent);
```

075 - Higher Order Functions – Reduce

ال Code الموجود في الدرس

```
/*
  - Reduce
  --- method executes a reducer function on each
element of the array,
  --- resulting in a single output value.


  Syntax
  reduce(callBackFunc(Accumulator, Current Val,
Current Index, Source Array) { }, initialValue)
  - Accumulator => the accumulated value previously
returned in the last invocation
  - Current Val => The current element being processed
in the array
  - Index => The index of the current element being
processed in the array.
  ---------- Starts from index 0 if an initialValue is
provided.
  ---------- Otherwise, it starts from index 1.
  - Array => The Current Array
*/
```

```
let nums = [10, 20, 15, 30];


let add = nums.reduce(function (acc, current, index,
arr) {
  console.log(`Acc => ${acc}`);
  console.log(`Current Element => ${current}`);
  console.log(`Current Element Index => ${index}`);
  console.log(`Array => ${arr}`);
  console.log(acc + current);
  console.log(`##############`);
  return acc + current;
}, 5);


console.log(add);
```

076 - Higher Order Functions – Reduce Practice

ال Code الموجود في الدرس

```
/*
  Reduce
  - Longest Word
```

```
  - Remove Characters + Use Reduce

*/


let theBiggest = ["Bla", "Propaganda", "Other", "AAA",
"Battery", "Test", "Propaganda_Two"];


let check = theBiggest.reduce(function (acc, current)
{
  console.log(`Acc => ${acc}`);

  console.log(`Current Element => ${current}`);

  console.log(acc.length > current.length ? acc :
current);

  console.log(`##############`);

  return acc.length > current.length ? acc : current;

});


console.log(check);


let removeChars = ["E", "@", "@", "L", "Z", "@", "@",
"E", "R", "@", "O"];


let finalString = removeChars

  .filter(function (ele) {

    return ele !== "@";

  })

  .reduce(function (acc, current) {
```

```
    return `${acc}${current}`;
  });


console.log(finalString);
```

My solution

```
let theBiggest = [
  "Bla",
  "Propaganda",
  "Other",
  "AAA",
  "Battery",
  "Test",
  "Propaganda_Two",
];
let check = theBiggest.reduce(function (ac, cr) {
  return ac.length > cr.length ? ac : cr;
});

console.log(check);

let removeChars = ["E", "@", "@", "L", "Z", "@", "@", "E", "R", "@", "O"];

let remove = removeChars.reduce(function (ac, cr) {
  return cr != "@" ? ac + cr : ac;
});

console.log(remove);
```

077 - Higher Order Functions – ForEach And Practice

ال Code الموجود في الدرس

```
<ul>
  <li class="active">One</li>
  <li>Two</li>
  <li>Three</li>
</ul>
<div class="content">
  <div>Div One</div>
  <div>Div Two</div>
  <div>Div Three</div>
</div>
/*
  - forEach
  --- method executes a provided function once for each array element.


  Syntax forEach(callBackFunction(Element, Index, Array) { }, thisArg)
  - 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


  Note
  - Doesnt Return Anything [Undefined]
```

```
  - Break Will Not Break The Loop
*/


let allLis = document.querySelectorAll("ul li");
let allDivs = document.querySelectorAll(".content
div");


allLis.forEach(function (ele) {
  ele.onclick = function () {
    // Remove Active Class From All Elements
    allLis.forEach(function (ele) {
      ele.classList.remove("active");
    });
    // Add Active Class To This Element
    this.classList.add("active");
    // Hide All Divs
    allDivs.forEach(function (ele) {
      ele.style.display = "none";
    });
  };
});
```

078 - Higher Order Functions Challenge

التحدي الموجود في الدرس

```
/*
  Higher Order Functions Challenges


  You Can Use

  - ,

  - _

  - Space

  - True => 1 => One Time Only In The Code


  You Cannot Use

  - Numbers

  - Letters


  - You Must Use [Filter + Map + Reduce + Your
Knowledge]

  - Order Is Not Important

  - All In One Chain


*/


let myString =
"1,2,3,EE,l,z,e,r,o,_,W,e,b,_,S,c,h,o,o,l,2,0,Z";
```

```
let solution = '?????';

console.log(solution); // Elzero Web School
```

079 - Object – Introduction

ال Code الموجود في الدرس

```
/*
  Object
  - Intro and What Is Object
  - Testing Window Object
  - Accessing Object
*/

let user = {
  // Properties
  theName: "Osama",
  theAge: 38,
  // Methods
```

```
  sayHello: function () {
    return `Hello`;
  },
};

console.log(user.theName);
console.log(user.theAge);
console.log(user.sayHello());
```

080 - Dot Notation vs Bracket Notation

ال Code الموجود في الدرس

```
/*
  Object
  - Dig Deeper
  - Dot Notation vs Bracket Notation
  - Dynamic Property Name
*/

let myVar = "country";

let user = {
  theName: "Osama",
  country: "Egypt",
```

```
};

console.log(user.theName);
console.log(user.country); // user.country
console.log(user.myVar); // user.country
console.log(user[myVar]); // user.country
```

081 - Nested Object And Advanced Examples

ال Code الموجود في الدرس

```
/*
  Object
  - Nested Object And Trainings
*/

let available = true;

let user = {
  name: "Osama",
  age: 38,
  skills: ["HTML", "CSS", "JS"],
  available: false,
```

```
  addresses: {
    ksa: "Riyadh",
    egypt: {
      one: "Cairo",
      two: "Giza",
    },
  },
  checkAv: function () {
    if (user.available === true) {
      return `Free For Work`;
    } else {
      return `Not Free`;
    }
  },
};

console.log(user.name);
console.log(user.age);
console.log(user.skills);
console.log(user.skills.join(" | "));
console.log(user.skills[2]); // Access With Index
console.log(user.addresses.ksa);
console.log(user.addresses.egypt.one);
console.log(user["addresses"].egypt.one);
```

```
console.log(user["addresses"]["egypt"]);
console.log(user["addresses"]["egypt"]["one"]);

console.log(user.checkAv());
```

082 - Create Object With New Keyword

ال Code الموجود في الدرس

```
/*
  Object
  - Create With New Keyword new Object();
*/

let user = new Object({
  age: 20,
});

console.log(user);

user.age = 38;
```

```
user["country"] = "Egypt";


user.sayHello = function () {
  return `Hello`;
};


console.log(user);
console.log(user.age);
console.log(user.country);
console.log(user.sayHello());
```

083 - This Keyword

ال Code الموجود في الدرس

```
/*
  Function this Keyword
  - this Introduction
  - this Inside Object Method
  --- When a function is called as a method of an
object,
  --- its this is set to the object the method is
called on.
  - Global Object
  - Test Variables With Window And This
  - Global Context
  - Function Context
```

```
  Search
  - Strict Mode
*/

console.log(this);
console.log(this === window);

myVar = 100;

console.log(window.myVar);
console.log(this.myVar);

function sayHello() {
  console.log(this);
  return this;
}
sayHello();

console.log(sayHello() === window);

document.getElementById("cl").onclick = function () {
  console.log(this);
};
```

```
let user = {
  age: 38,
  ageInDays: function () {
    console.log(this);
    return this.age * 365;
  },
};

console.log(user.age);
console.log(user.ageInDays());
```

---

084 - Create Object With Create Method

ال Code الموجود في الدرس

```
/*
  Object
  - Create Object With Create Method
*/
```

```
let user = {
  age: 20,
```

```
  doubleAge: function () {
    return this.age * 2;
  },
};

console.log(user);
console.log(user.age);
console.log(user.doubleAge());
```

```
let obj = Object.create({});

obj.a = 100;

console.log(obj);

let copyObj = Object.create(user);

copyObj.age = 50;

console.log(copyObj);
console.log(copyObj.age);
console.log(copyObj.doubleAge());
```

085 - Create Object With Assign Method

ال Code الموجود في الدرس

```
/*
  Object
  - Create Object With Assign Method
*/

let obj1 = {
  prop1: 1,
  meth1: function () {
    return this.prop1;
  },
};

let obj2 = {
  prop2: 2,
  meth2: function () {
    return this.prop2;
  },
};

let targetObject = {
  prop1: 100,
```

```
  prop3: 3,
};


let finalObject = Object.assign(targetObject, obj1,
obj2);


finalObject.prop1 = 200;
finalObject.prop4 = 4;


console.log(finalObject);


let newObject = Object.assign({}, obj1, { prop5: 5,
prop6: 6 });


console.log(newObject);
```

086 - What Is DOM ? And Select Elements

ال Code الموجود في الدرس

```
/*
  DOM
  - What Is DOM
  - DOM Selectors
```

```
  --- Find Element By ID
  --- Find Element By Tag Name
  --- Find Element By Class Name
  --- Find Element By CSS Selectors
  --- Find Element By Collection
  ------ title
  ------ body
  ------ images
  ------ forms
  ------ links
*/


let myIdElement = document.getElementById("my-div");
let myTagElements =
document.getElementsByTagName("p");
let myClassElement =
document.getElementsByClassName("my-span");
let myQueryElement =
document.querySelector(".my-span");
let myQueryAllElement =
document.querySelectorAll(".my-span");


console.log(myIdElement);
console.log(myTagElements[1]);
console.log(myClassElement[1]);
console.log(myQueryElement);
```

```
console.log(myQueryAllElement[1]);


console.log(document.title);
console.log(document.body);
console.log(document.forms[0].one.value);
console.log(document.links[1].href);
```

087 - Get, Set Elements Content And Attributes

ال Code الموجود في الدرس

```
/*
  DOM [Get / Set Elements Content And Attributes]
  - innerHTML
  - textContent
  - Change Attributes Directly
  - Change Attributes With Methods
  --- getAttribute
  --- setAttribute


  Search
  - innerText
*/
```

```
let myElement = document.querySelector(".js");


console.log(myElement.innerHTML);
console.log(myElement.textContent);


myElement.innerHTML = "Text From <span>Main.js</span>
File";
myElement.textContent = "Text From
<span>Main.js</span> File";


document.images[0].src = "https://google.com";
document.images[0].alt = "Alternate";
document.images[0].title = "Picture";
document.images[0].id = "pic";
document.images[0].className = "img";


let myLink = document.querySelector(".link");


console.log(myLink.getAttribute("class"));
console.log(myLink.getAttribute("href"));


myLink.setAttribute("href", "https://twitter.com");
myLink.setAttribute("title", "Twitter");
```

088 - Check Attributes And Examples

ال Code الموجود في الدرس

```
/*
  DOM [Check Attributes]
  - Element.attributes
  - Element.hasAttribute
  - Element.hasAttributes
  - Element.removeAttribute
*/


console.log(document.getElementsByTagName("p")[0].attr
ibutes);


let myP = document.getElementsByTagName("p")[0];


if (myP.hasAttribute("data-src")) {
  if (myP.getAttribute("data-src") === "") {
    myP.removeAttribute("data-src");
  } else {
    myP.setAttribute("data-src", "New Value");
  }
```

```
} else {
  console.log(`Not Found`);
}

if (myP.hasAttributes()) {
  console.log(`Has Attributes`);
}

if
(document.getElementsByTagName("div")[0].hasAttributes
()) {
  console.log(`Has Attributes`);
} else {
  console.log(`Div Has No Attributes`);
```

089 - Create And Append Elements

ال Code الموجود في الدرس

```
/*
  DOM [Create Elements]
  - createElement
  - createComment
```

```
  - createTextNode
  - createAttribute
  - appendChild
*/

let myElement = document.createElement("div");
let myAttr = document.createAttribute("data-custom");
let myText = document.createTextNode("Product One");
let myComment = document.createComment("This Is Div");

myElement.className = "product";
myElement.setAttributeNode(myAttr);
myElement.setAttribute("data-test", "Testing");

// Append Comment To Element
myElement.appendChild(myComment);

// Append Text To Element
myElement.appendChild(myText);

// Append Element To Body
document.body.appendChild(myElement);
```

090 - Product With Heading And Paragraph Practice

ال Code الموجود في الدرس

```
/*
  DOM [Create Elements]
  - Practice Product With Heading And Paragraph
*/


let myMainElement = document.createElement("div");
let myHeading = document.createElement("h2");
let myParagraph = document.createElement("p");


let myHeadingText = document.createTextNode("Product Title");
let myParagraphText = document.createTextNode("Product Description");


// Add Heading Text
myHeading.appendChild(myHeadingText);


// Add Heading To Main Element
myMainElement.appendChild(myHeading);
```

```
// Add Paragraph Text
myParagraph.appendChild(myParagraphText);

// Add Paragraph To Main Element
myMainElement.appendChild(myParagraph);

myMainElement.className = "product";

document.body.appendChild(myMainElement);
```

091 - Deal With Childrens

ال Code الموجود في الدرس

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge" />
    <title>Learn JavaScript</title>
  </head>
  <body>
    <div><!-- Osama -->Hello Div<p>Hello
P</p><span>Hello Span</span><!-- Comment
-->Hello</div>
```

```
    <script src="main.js"></script>
  </body>
</html>
/*
  DOM [Deal With Childrens]
  - children
  - childNodes
  - firstChild
  - lastChild
  - firstElementChild
  - lastElementChild
*/

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

console.log(myElement);
console.log(myElement.children);
console.log(myElement.children[0]);
console.log(myElement.childNodes);
console.log(myElement.childNodes[0]);

console.log(myElement.firstChild);
console.log(myElement.lastChild);
```

```
console.log(myElement.firstElementChild);
console.log(myElement.lastElementChild);
```

092 - DOM Events

ال Code الموجود في الدرس

```
/*
  DOM [Events]
  - Use Events On HTML
  - Use Events On JS
  --- onclick
  --- oncontextmenu
  --- onmouseenter
  --- onmouseleave

  --- onload
  --- onscroll
  --- onresize

  --- onfocus
  --- onblur
  --- onsubmit
*/

let myBtn = document.getElementById("btn");
```

```
myBtn.onmouseleave = function () {
  console.log("Clicked");
};

window.onresize = function () {
  console.log("Scroll");
};
```

093 - Validate Form And Prevent Default

ال Code الموجود في الدرس

```
/*
  DOM [Events]
  - Validate Form Practice
  - Prevent Default
*/

let userInput =
document.querySelector("[name='username']");
let ageInput = document.querySelector("[name='age']");

document.forms[0].onsubmit = function (e) {
```

```
  let userValid = false;
  let ageValid = false;

  if (userInput.value !== "" && userInput.value.length
<= 10) {
    userValid = true;
  }

  if (ageInput.value !== "") {
    ageValid = true;
  }

  if (userValid === false || ageValid === false) {
    e.preventDefault();
  }
};

document.links[0].onclick = function (event) {
  console.log(event);
  event.preventDefault();
};
```

094 - Event Simulation – Click, Focus, Blur

ال Code الموجود في الدرس

```
/*
  DOM [Events Simulation]
  - click
  - focus
  - blur
*/

let one = document.querySelector(".one");
let two = document.querySelector(".two");

window.onload = function () {
  two.focus();
};

one.onblur = function () {
  document.links[0].click();
};
```

095 - ClassList Object and Methods

ال Code الموجود في الدرس

```
/*
  DOM [Class List]
  - classList
  --- length
  --- contains
  --- item(index)
  --- add
  --- remove
  --- toggle
*/

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

console.log(element.classList);
console.log(typeof element.classList);
console.log(element.classList.contains("osama"));
console.log(element.classList.contains("show"));
console.log(element.classList.item("3"));

element.onclick = function () {
  element.classList.toggle("show");
};
```

096 - CSS Styling And Stylesheets

ال Code الموجود في الدرس

```
/*
  DOM [Class List]
  - classList
  --- length
  --- contains
  --- item(index)
  --- add
  --- remove
  --- toggle
*/

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

console.log(element.classList);
console.log(typeof element.classList);
console.log(element.classList.contains("osama"));
console.log(element.classList.contains("show"));
console.log(element.classList.item("3"));

element.onclick = function () {
```

```
  element.classList.toggle("show");
};
```

097 - Before, After, Prepend, Append, Remove

ال Code الموجود في الدرس

```
/*
  DOM [Deal With Elements]
  - before [Element || String]
  - after [Element || String]
  - append [Element || String]
  - prepend [Element || String]
  - remove
*/

let element = document.getElementById("my-div");
let createdP = document.createElement("p");

// element.remove();
```

098 - DOM Traversing

ال Code الموجود في الدرس

```
/*
  DOM [Traversing]
  - nextSibling
  - previousSibling
  - nextElementSibling
  - previousElementSibling
  - parentElement
*/

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

console.log(span.parentElement);

span.onclick = function () {
  span.parentElement.remove();
}
```

099 - DOM Cloning

ال Code الموجود في الدرس

```
/*
```

```
  DOM [Cloning]
  - cloneNode(Deep)
*/

let myP = document.querySelector("p").cloneNode(true);
let myDiv = document.querySelector("div");

myP.id = `${myP.id}-clone`;

myDiv.appendChild(myP);
```

100 - addEventListener

ال Code الموجود في الدرس

```
/*
  DOM [Add Event Listener]
  - addEventListener
  - Use Without On
  - Attach Multiple Events
  - Error Test

  Search
  - Capture & Bubbling JavaScript
  - removeEventListener
```

```
*/

let myP = document.querySelector("p");

// myP.onclick = one;
// myP.onclick = two;

// function one() {
//   console.log("Message From OnClick 1");
// }
// function two() {
//   console.log("Message From OnClick 2");
// }

// window.onload = "Osama";

// myP.addEventListener("click", function () {
//   console.log("Message From OnClick 1 Event");
// });

// myP.addEventListener("click", one);
// myP.addEventListener("click", two);

// myP.addEventListener("click", "String"); // Error
```

```
myP.onclick = function () {
  let newP = myP.cloneNode(true);
  newP.className = "clone";
  document.body.appendChild(newP);
};

// let cloned = document.querySelector(".clone"); //
Error

// cloned.onclick = function () {
//   console.log("Iam Cloned");
// };

document.addEventListener("click", function (e) {
  if (e.target.className === "clone") {
    console.log("Iam Cloned");
  }
});
```

101 Bom challenge

DOM Challenge | أكاديمية الزيرو (elzero.org)

102 - What Is BOM ?

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - Introduction
  --- Window Object Is The Browser Window
  --- Window Contain The Document Object
  --- All Global Variables And Objects And Functions
Are Members Of Window Object
  ------ Test Document And Console
  - What Can We Do With Window Object ?
  --- Open Window
  --- Close Window
  --- Move Window
  --- Resize Window
  --- Print Document
  --- Run Code After Period Of Time Once Or More
  --- Fully Control The URL
  --- Save Data Inside Browser To Use Later
*/


window.document.title = "Hello JS";
```

103 - Alert, Confirm, Prompt

ال Code الموجود في الدرس

```
/*

  BOM [Browser Object Model]

  - alert(Message) => Need No Response Only Ok
Available

  - confirm(Message) => Need Response And Return A
Boolean

  - prompt(Message, Default Message) => Collect Data

*/


// alert("Test");

// console.log("Test");


// let confirmMsg = confirm("Are You Sure?");


// console.log(confirmMsg);


// if (confirmMsg === true) {

//   console.log("Item Deleted");

// } else {
```

```
//   console.log("Item Not Deleted");
// }


let promptMsg = prompt("Good Day To You?", "Write Day
With 3 Characters");


console.log(promptMsg);
```

104 - setTimeout and clearTimeout

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - setTimeout(Function, Timeout, Additional Params)
  - clearTimeout(Identifier)
*/


// setTimeout(function () {
//   console.log("Msg");
// }, 3000);


// setTimeout(sayMsg, 3000);


// function sayMsg() {
```

```
//   console.log(`Iam Message`);
// }


// setTimeout(sayMsg, 3000, "Osama", 38);


// function sayMsg(user, age) {
//   console.log(`Iam Message For ${user} Age Is : ${age}`);
// }


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


btn.onclick = function () {
  clearTimeout(counter);
};


function sayMsg(user, age) {
  console.log(`Iam Message For ${user} Age Is : ${age}`);
}


let counter = setTimeout(sayMsg, 3000, "Osama", 38);
```

105 - setInterval and clearInterval

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - setInterval(Function, Millseconds, Additional
Params)
  - clearInterval(Identifier)
*/


// setInterval(function () {
//   console.log(`Msg`);
// }, 1000);


// setInterval(sayMsg, 1000);


// function sayMsg() {
//   console.log(`Iam Message`);
// }


// setInterval(sayMsg, 1000, "Osama", 38);


// function sayMsg(user, age) {
```

```
//   console.log(`Iam Message For ${user} His Age Is:
${age}`);
// }


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


function countdown() {
  div.innerHTML -= 1;
  if (div.innerHTML === "0") {
    clearInterval(counter);
  }
}


let counter = setInterval(countdown, 1000);
```

106 - Window Location Object

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - location Object
  --- href Get / Set [URL || Hash || File || Mail]
  --- host
```

```
  --- hash
  --- protocol
  --- reload()
  --- replace()
  --- assign()
*/


console.log(location);
console.log(location.href);


// location.href = "https://google.com";
// location.href = "/#sec02";
// location.href =
"https://developer.mozilla.org/en-US/docs/Web/JavaScri
pt#reference";


// console.log(location.host);
// console.log(location.hostname);


// console.log(location.protocol);


// console.log(location.hash);
```

107 - Window Open And Close

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - 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]
  --- menubar [yes || no]

  Search
  - Window.Open Window Features
*/

setTimeout(function () {
  window.open("", "_self", "", false);
}, 2000);

setTimeout(function () {
```

```
  window.open("https://google.com", "_blank",
"width=400,height=400,left=200,top=10");

}, 2000);
```

108 - Window History Object

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - History API
  --- Properties
  ------ length
  --- Methods
  ------ back()
  ------ forward()
  ------ go(Delta) => Position In History

  Search [For Advanced Knowledge]
  - pushState() + replaceState()
*/

console.log(history);
```

109 - Scroll, ScrollTo, ScrollBy, Focus, Print, Stop

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - stop()
  - print()
  - focus()
  - scrollTo(x, y || Options)
  - scroll(x, y || Options)
  - scrollBy(x, y || Options)
*/


// let myNewWindow = window.open("https://google.com",
"", "width=500,height=500");


// window.scrollTo({
//   left: 500,
//   top: 200,
//   behavior: "smooth"
// });
```

110 - Scroll To Top Using ScrollY Practice

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  - Practice => Scroll To Top
  - scrollX [Alias => PageXOffset]
  - scrollY [Alias => PageYOffset]
*/

// console.log(window.scrollX === window.pageXOffset);

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

window.onscroll = function () {
  if (window.scrollY >= 600) {
    btn.style.display = "block";
  } else {
    btn.style.display = "none";
  }
};

btn.onclick = function () {
  window.scrollTo({
    left: 0,
```

```
    top: 0,
    behavior: "smooth",
  });
};
```

111 - Local Storage

ال Code الموجود في الدرس

```
/*
  BOM [Browser Object Model]
  Local Storage
  - setItem
  - getItem
  - removeItem
  - clear
  - key

  Info
  - No Expiration Time
  - HTTP And HTTPS
  - Private Tab
*/

// Set
```

```
window.localStorage.setItem("color", "#F00");
window.localStorage.fontWeight = "bold";
window.localStorage["fontSize"] = "20px";

// Get
console.log(window.localStorage.getItem("color"));
console.log(window.localStorage.color);
console.log(window.localStorage["color"]);

// Remove One
// window.localStorage.removeItem("color");

// Remove All
// window.localStorage.clear();

// Get Key
console.log(window.localStorage.key(0));

// Set Color In Page
document.body.style.backgroundColor =
window.localStorage.getItem("color");

console.log(window.localStorage);
console.log(typeof window.localStorage);
```

112 - Local Storage Color App Practice

ال Code الموجود في الدرس

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <title>Learn JavaScript</title>
    <link rel="stylesheet" href="main.css" />
    <style>
      body {
        background-color: #eee;
      }
      ul {
        padding: 0;
        margin: 0;
        background-color: #ddd;
        margin: 20px auto;
        padding: 20px;
        width: 400px;
        list-style: none;
```

```
    display: flex;
    justify-content: space-between;
  }
  ul li {
    width: 60px;
    height: 60px;
    border: 2px solid transparent;
    opacity: 0.4;
    cursor: pointer;
    transition: 0.3s;
  }
  ul li.active,
  ul li:hover {
    opacity: 1;
  }
  ul li:first-child {
    background-color: red;
  }
  ul li:nth-child(2) {
    background-color: green;
  }
  ul li:nth-child(3) {
    background-color: blue;
  }
```

```
      ul li:nth-child(4) {
        background-color: black;
      }
      .experiment {
        background-color: red;
        width: 600px;
        height: 300px;
        margin: 20px auto;
      }
    </style>
  </head>
  <body>
    <ul>
      <li class="active" data-color="red"></li>
      <li data-color="green"></li>
      <li data-color="blue"></li>
      <li data-color="black"></li>
    </ul>
    <div class="experiment"></div>
    <script src="main.js"></script>
  </body>
</html>
```

نسخ الكود

```
/*
```

```
  BOM [Browser Object Model]
  Local Storage Practice
*/

let lis = document.querySelectorAll("ul li");
let exp = document.querySelector(".experiment");

if (window.localStorage.getItem("color")) {
  // If There Is Color In Local Storage
  // [1] Add Color To Div
  exp.style.backgroundColor =
window.localStorage.getItem("color");
  // [2] Remove Active Class From All Lis
  lis.forEach((li) => {
    li.classList.remove("active");
  });
  // [3] Add Active Class To Current Color

document.querySelector(`[data-color="${window.localSto
rage.getItem("color")}"]`).classList.add("active");
}

lis.forEach((li) => {
  li.addEventListener("click", (e) => {
    // console.log(e.currentTarget.dataset.color);
```

```
    // Remove Active Class From all Lis
    lis.forEach((li) => {
      li.classList.remove("active");
    });
    // Add Active Class To Current Element
    e.currentTarget.classList.add("active");
    // Add Current Color To Local Storage
    window.localStorage.setItem("color",
e.currentTarget.dataset.color);
    // Change Div Background Color
    exp.style.backgroundColor =
e.currentTarget.dataset.color;
  });
});
```

113 - Session Storage And Use Cases

ال Code الموجود في الدرس

```
<form action="">
  <input type="text" class="name" />
</form>
```

نسخ الكود

```
/*
  BOM [Browser Object Model]
```

```
  Session Storage
  - setItem
  - getItem
  - removeItem
  - clear
  - key

  Info
  - New Tab = New Session
  - Duplicate Tab = Copy Session
  - New Tab With Same Url = New Session
*/

// window.localStorage.setItem("color", "red");
// window.sessionStorage.setItem("color", "blue");

document.querySelector(".name").onblur = function () {
  // console.log(this.value);
  window.localStorage.setItem("input-name",
this.value);
};
```

My plan for the challenge on the video 114

What I am going to do?

First I will create the design which will be normal

Second the application will consist of a text input and a button when I click on the button It will take the value form the input I wrote and It put it in the local storage and display it below with a button it name is delete the will remove the Item from the local storage

And remove it from the page immediately

so, what am I going to do?

1) **First I will select the text input and I will select the button**
2) **I will make "on submit" on the button**
3) **This function will make the job**
   a) **It will put the value in the local storage**
   b) **I will create the element which contain the note I will be a div with a button**
4) **I will select this button by its class "delete"**
5) **I will make a function on it that will remove the div from the page and it will remove the text from the local storge**

How can I link between the delete button and the div?

I solved the problem and you will find it in the challenges solutions for this week in vs Code

---

115 - Destructuring Arrays Part 1

ال Code الموجود في الدرس

```
/*
  Destructuring
  " is a JavaScript expression that allows us to extract data from arrays,
    objects, and maps and set them into new, distinct variables. "
  - Destructuring Array
*/

let a = 1;
let b = 2;
let c = 3;
let d = 4;

let myFriends = ["Ahmed", "Sayed", "Ali", "Maysa"];

[a = "A", b, c, d, e = "Osama"] = myFriends;

console.log(a);
console.log(b);
console.log(c);
console.log(d);
console.log(e);
```

```
// console.log(myFriends[4]);

let [, y, , z] = myFriends;

console.log(y);
console.log(z);
```

116 - Destructuring Arrays Part 2

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Array Advanced Examples
*/

let myFriends = ["Ahmed", "Sayed", "Ali", ["Shady",
"Amr", ["Mohamed", "Gamal"]]];

// console.log(myFriends[3][2][1]);

// let [, , , [a, , [, b]]] = myFriends;

let [, , , [a, , [, b]]] = myFriends;
```

```
console.log(a); // Shady
console.log(b); // Gamal
```

117 - Destructuring Arrays Part 3 Swap Variables

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Array => Swapping Variables
*/

let book = "Video";
let video = "Book";

// // Save Book Value In Stash
// let stash = book; // Video

// // Change Book Value
// book = video; // Book

// // Change Video Value
// video = stash; // Video
```

```
[book, video] = [video, book];

console.log(book);
console.log(video);
```

118 - Destructuring Objects Part 1

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Object
*/

const user = {
  theName: "Osama",
  theAge: 39,
  theTitle: "Developer",
  theCountry: "Egypt",
};

// console.log(user.theName);
// console.log(user.theAge);
```

```
// console.log(user.theTitle);
// console.log(user.theCountry);

// let theName = user.theName;
// let theAge = user.theAge;
// let theTitle = user.theTitle;
// let theCountry = user.theCountry;

// console.log(theName);
// console.log(theAge);
// console.log(theTitle);
// console.log(theCountry);

// ({ theName, theAge, theTitle, theCountry } = user);
const { theName, theAge, theCountry } = user;

console.log(theName);
console.log(theAge);
console.log(theCountry);
```

119 - Destructuring Objects Part 2

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Object
```

```
  --- Naming The Variables
  --- Add New Property
  --- Nested Object
  --- Destructuring The Nested Object Only
*/

const user = {
  theName: "Osama",
  theAge: 39,
  theTitle: "Developer",
  theCountry: "Egypt",
  theColor: "Black",
  skills: {
    html: 70,
    css: 80,
  },
};

const {
  theName: n,
  theAge: a,
  theCountry,
  theColor: co = "Red",
  skills: { html: h, css },
```

```
} = user;


console.log(n);
console.log(a);
console.log(theCountry);
console.log(co);
console.log(`My HTML Skill Progress Is ${h}`);
console.log(`My CSS Skill Progress Is ${css}`);

const { html: skillOne, css: skillTwo } = user.skills;

console.log(`My HTML Skill Progress Is ${skillOne}`);
console.log(`My CSS Skill Progress Is ${skillTwo}`);
```

120 - Destructuring Function Parameters

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Function Parameters
```

```
*/


const user = {
  theName: "Osama",
  theAge: 39,
  skills: {
    html: 70,
    css: 80,
  },
};


showDetails(user);


// function showDetails(obj) {
//   console.log(`Your Name Is ${obj.theName}`);
//   console.log(`Your Age Is ${obj.theAge}`);
//   console.log(`Your CSS Skill Progress Is
${obj.skills.css}`);
// }


function showDetails({ theName: n, theAge: a, skills:
{ css: c } } = user) {
  console.log(`Your Name Is ${n}`);
  console.log(`Your Age Is ${a}`);
  console.log(`Your CSS Skill Progress Is ${c}`);
```

```
}
```

121 - Destructuring Mixed Content

ال Code الموجود في الدرس

```
/*
  Destructuring
  - Destructuring Mixed Content
*/

const user = {
  theName: "Osama",
  theAge: 39,
  skills: ["HTML", "CSS", "JavaScript"],
  addresses: {
    egypt: "Cairo",
    ksa: "Riyadh",
  },
};

const {
  theName: n,
  theAge: a,
```

```
  skills: [, , three],
  addresses: { egypt: e },
} = user;

console.log(`Your Name Is: ${n}`);
console.log(`Your Age Is: ${a}`);
console.log(`Your Last Skill Is: ${three}`);
console.log(`Your Live In: ${e}`);
```

122 - Destructuring Challenge

التحدي الموجود في الدرس

```
/*
  Destructuring
  - Challenge
*/

let chosen = 3;

let myFriends = [
  { title: "Osama", age: 39, available: true, skills:
["HTML", "CSS"] },
  { title: "Ahmed", age: 25, available: false, skills:
["Python", "Django"] },
```

```
  { title: "Sayed", age: 33, available: true, skills:
["PHP", "Laravel"] },

];
```

123 - Set Data Type And Methods

ال Code الموجود في الدرس

```
/*
  - Set Data Type
  Syntax: new Set(Iterable)
  -- Object To Store Unique Values
  -- Cannot Access Elements By Index

  Properties:
  - size

  Methods:
  - add
  - delete
  - clear
  - has
*/
```

```
let myData = [1, 1, 1, 2, 3, "A"];

// let myUniqueData = new Set([1, 1, 1, 2, 3]);

// let myUniqueData = new Set(myData);

// let myUniqueData = new
Set().add(1).add(1).add(1).add(2).add(3);

let myUniqueData = new Set();


myUniqueData.add(1).add(1).add(1);

myUniqueData.add(2).add(3).add("A");


console.log(`Is Set Has => A
${myUniqueData.has("a".toUpperCase())}`);


console.log(myData);

console.log(myUniqueData);


console.log(myUniqueData.size);


console.log(myData[0]);

console.log(myUniqueData[0]);


// myUniqueData.delete(2);

console.log(myUniqueData.delete(2));


console.log(myUniqueData);
```

```
console.log(myUniqueData.size);


myUniqueData.clear();


console.log(myUniqueData);
console.log(myUniqueData.size);
```

124 - Set vs WeakSet And Garbage Collector

ال Code الموجود في الدرس

```
/*
  - Set vs WeakSet
  "
    The WeakSet is weak,
    meaning references to objects in a WeakSet are held weakly.
    If no other references to an object stored in the WeakSet exist,
    those objects can be garbage collected.
  "
  --
  Set     => Can Store Any Data Values
  WeakSet => Collection Of Objects Only
```

```
  --

  Set     => Have Size Property

  WeakSet => Does Not Have Size Property

  --

  Set     => Have Keys, Values, Entries

  WeakSet => Does Not Have clear, Keys, Values And
Entries

  --

  Set     => Can Use forEach

  WeakSet => Cannot Use forEach


  Usage: Store objects and removes them once they
become inaccessible

*/


// Type Of Data


let mySet = new Set([1, 1, 1, 2, 3, "A", "A"]);


console.log(mySet);


// Size

console.log(`Size Of Elements Inside Set Is:
${mySet.size}`);
```

```
// Values + Keys [Alias For Values]
let iterator = mySet.keys();

console.log(iterator.next().value);
console.log(iterator.next().value);
console.log(iterator.next().value);
console.log(iterator.next().value);
console.log(iterator.next());

// forEach

mySet.forEach((el) => console.log(el));

console.log("#".repeat(20));

// Type Of Data

let myws = new WeakSet([{ A: 1, B: 2 }]);

console.log(myws);
```

125 - Map Data Type vs Object

ال Code الموجود في الدرس

```
/*
  - Map Data Type
```

```
  Syntax: new Map(Iterable With Key/Value)

  -- Map vs Object

  --

  ------ Map => Does Not Contain Key By Default

  ------ Object => Has Default Keys

  --

  ------ Map => Key Can Be Anything [Function, Object,
Any Primitive Data Types]

  ------ Object => String Or Symbol

  --

  ------ Map => Ordered By Insertion

  ------ Object => Not 100% Till Now

  --

  ------ Map => Get Items By Size

  ------ Object => Need To Do Manually

  --

  ------ Map => Can Be Directly Iterated

  ------ Object => Not Directly And Need To Use
Object.keys() And So On

  --

  ------ Map => Better Performance When Add Or Remove
Data

  ------ Object => Low Performance When Comparing To
Map

*/
```

```
let myObject = {};
let myEmptyObject = Object.create(null);
let myMap = new Map();

console.log(myObject);
console.log(myEmptyObject);
console.log(myMap);

let myNewObject = {
  10: "Number",
  "10": "String",
};

console.log(myNewObject[10]);

let myNewMap = new Map();
myNewMap.set(10, "Number");
myNewMap.set("10", "String");
myNewMap.set(true, "Boolean");
myNewMap.set({a: 1, b: 2}, "Object");
myNewMap.set(function doSomething() {}, "Function");

console.log(myNewMap.get(10));
console.log(myNewMap.get("10"));
```

```
console.log("####");

console.log(myNewObject);
console.log(myNewMap);
```

126 - Map Methods

ال Code الموجود في الدرس

```
/*
  - Map Data Type
  Methods
  --- set
  --- get
  --- delete
  --- clear
  --- has

  Properties
  --- size
```

```
*/

let myMap = new Map([
  [10, "Number"],
  ["Name", "String"],
  [false, "Boolean"],
]);

// myMap.set(10, "Number");
// myMap.set("Name", "String");

console.log(myMap);

console.log(myMap.get(10));
console.log(myMap.get("Name"));
console.log(myMap.get(false));

console.log("####");

console.log(myMap.has("Name"));

console.log("####");

console.log(myMap.size);

console.log(myMap.delete("Name"));
```

```
console.log(myMap.size);

myMap.clear();

console.log(myMap.size);
```

127 - Map vs WeakMap

ال Code الموجود في الدرس

```
/*
  - Map vs WeakMap
  "
    WeakMap Allows Garbage Collector To Do Its Task But Not
Map.
  "
  --
  Map     => Key Can Be Anything
  WeakMap => Key Can Be Object Only
  --
*/
```

```
let mapUser = { theName: "Elzero" };

let myMap = new Map();

myMap.set(mapUser, "Object Value");

mapUser = null; // Override The Reference

console.log(myMap);

console.log("#".repeat(20));

let wMapUser = { theName: "Elzero" };

let myWeakMap = new WeakMap();

myWeakMap.set(wMapUser, "Object Value");

wMapUser = null; // Override The Reference
```

```
console.log(myWeakMap);
```

128 - Array.from Method

ال Code الموجود في الدرس

```
/*
  Array Methods
  - Array.from(Iterable, MapFunc, This)
  --- Variable
  --- String Numbers
  --- Set
  --- Using The Map Function
  --- Arrow Function
  --- Shorten The Method + Use arguments
*/

console.log(Array.from("Osama"));
```

```
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));

// console.log([...new Set(myArray)]); // Future

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

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

129 - Array.copyWithin Method

ال Code الموجود في الدرس

```
/*

  Array Methods

  - Array.copyWithin(Target, Start => Optional, End =>
Optional)

  "Copy Part Of An Array To Another Location in The Same
Array"

  -- Any Negative Value Will Count From The End

  -- Target

  ---- Index To Copy Part To

  ---- If At Or Greater Than Array Length Nothing Will Be
Copied

  -- Start

  ---- Index To Start Copying From
```

```
  ---- If Ommited = Start From Index 0

  -- End

  ---- Index To End Copying From

  ---- Not Including End

  ---- If Ommited = Reach The End

*/


let myArray = [10, 20, 30, 40, 50, "A", "B"];


// myArray.copyWithin(3); // [10, 20, 30, 10, 20, 30, 40]


// myArray.copyWithin(4, 6); // [10, 20, 30, 40, "B", "A",
"B"]


// myArray.copyWithin(4, -1); // [10, 20, 30, 40, "B", "A",
"B"]


// myArray.copyWithin(1, -2); // [10, "A", "B", 40, 50, "A",
"B"]


myArray.copyWithin(1, -2, -1); // [10, "A", 30, 40, 50, "A",
"B"]


console.log(myArray);
```

130 - Array.some Method

ال Code الموجود في الدرس

```
/*

  Array Methods

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

  --- CallbackFunc => Function To Run On Every Element On
The Given Array

  ------ Element => The Current Element To Process

  ------ Index => Index Of Current Element

  ------ Array => The Current Array Working With

  --- This Argument => Value To Use As This When Executing
CallbackFunc

  --

  Using
```

```
  - Check if Element Exists In Array
  - Check If Number In Range
*/

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

let myNumber = 10;

// let check = nums.some(function (e) {
//   console.log("Test");
//   return e > 5;
// });

let check = nums.some(function (e) {
  return e > this;
}, myNumber);

// let check = nums.some((e) => e > 5);

console.log(check);

function checkValues(arr, val) {
  return arr.some(function (e) {
    return e === val;
  });
}
```

```
console.log(checkValues(nums, 20));
console.log(checkValues(nums, 5));

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

let checkNumberInRange = nums.some(function (e) {
  // console.log(this.min);
  // console.log(this.max);
  return e >= this.min && e <= this.max;
}, range);
```

```
console.log(checkNumberInRange);
```

131 - Array.every Method

ال Code الموجود في الدرس

```
/*

  Array Methods

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

  --- CallbackFunc => Function To Run On Every Element On
The Given Array

  ------ Element => The Current Element To Process

  ------ Index => Index Of Current Element

  ------ Array => The Current Array Working With

  --- This Argument => Value To Use As This When Executing
CallbackFunc

  --

*/


const locations = {

  20: "Place 1",

  30: "Place 2",

  50: "Place 3",

  40: "Place 4",

};


let mainLocation = 15;


let locationsArray = Object.keys(locations);


console.log(locationsArray);
```

```
let locationArrayNumbers = locationsArray.map((n) => +n);


console.log(locationArrayNumbers);


let check = locationArrayNumbers.every(function (e) {
  return e > this;
}, mainLocation);


console.log(check);
```

132 - Spread Syntax And Use Cases

ال Code الموجود في الدرس

```
/*
  Spread Operator => ...Iterable
  "Allow Iterable To Expand In Place"
*/


// Spread With String => Expand String
```

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

// Concatenate Arrays

let myArray1 = [1, 2, 3];
let myArray2 = [4, 5, 6];

let allArrays = [...myArray1, ...myArray2];
console.log(allArrays);

// Copy Array

let copiedArray = [...myArray1];
console.log(copiedArray);

// Push Inside Array

let allFriends = ["Osama", "Ahmed", "Sayed"];
let thisYearFriends = ["Sameh", "Mahmoud"];

allFriends.push(...thisYearFriends);

console.log(allFriends);
```

```
// Use With Math Object

let myNums = [10, 20, -100, 100, 1000, 500];
console.log(Math.max(...myNums));

// Spread With Objects => Merge Objects

let objOne = {
  a: 1,
  b: 2,
};
let objTwo = {
  c: 3,
  d: 4,
};
```

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

133 - Map And Set Challenge

التحدي الموجود في الدرس

```
/*
  Map And Set + What You Learn => Challenge
  Requirements
  - You Cant Use Numbers Or True Or False
  - Don't Use Array Indexes
  - You Cant Use Loop
  - You Cant Use Any Higher Order Function
  - Only One Line Solution Inside Console
  - If You Use Length => Then Only Time Only
  Hints
  - You Can Use * Operator Only In Calculation
  - Set
  - Spread Operator
  - Math Object Methods
*/

let n1 = [10, 30, 10, 20];
let n2 = [30, 20, 10];

console.log("Your Solution Here"); // 210
```

134 - Intro And What Is Regular Expression?

ال Code

```
/*
  Regular Expression
  - Email
  - IP
  - Phone
  - URL
*/

let str1 = '10 20 100 1000 5000';
let str2 = 'Os1 Os12 Os123 Os123Os Os12312Os123';

let invalidEmail = 'Osama@@@gmail....com';
let validEmail = 'o@nn.sa';

let ip = '192.168.2.1'; // IPv4

let url = 'elzero.org';
let url = 'elzero.org/';
let url = 'http://elzero.org/';
```

```
let url = 'http://www.elzero.org/';
let url = 'https://.elzero.org/';
let url = 'https://www.elzero.org/';
```

```
let url = 'https://www.elzero.org/?facebookid=asdasdasd';
```

```
135 - Regular Expression – Modifiers
```


ال Code

```
/*
  Regular Expression

  Syntax
  /pattern/modifier(s);
  new RegExp("pattern", "modifier(s)")

  Modifiers => Flags
  i => case-insensitive
  g => global
  m => Multilines
```

```
  Search Methods
  - match(Pattern)

  Match
  -- Matches A String Against a Regular Expression Pattern
  -- Returns An Array With The Matches
  -- Returns null If No Match Is Found.
*/

let myString = "Hello Elzero Web School I Love elzero";

let regex = /elzero/ig;

console.log(myString.match(regex));
```

136 - Regular Expression – Ranges Part 1

الـ Code

```
/*
```

```
  Regular Expression

  Ranges

  - Part 1
  (X|Y) => X Or Y
  [0-9] => 0 To 9
  [^0-9] => Any Character Not 0 To 9
  Practice

  - Part 2
  [a-z]
  [^a-z]
  [A-Z]
  [^A-Z]
  [abc]
  [^abc]

*/

let tld = "Com Net Org Info Code Io";
let tldRe = /(info|org|io)/ig;
console.log(tld.match(tldRe));

let nums = "12345678910";
let numsRe = /[0-2]/g;
```

```
console.log(nums.match(numsRe));

let notNums = "12345678910";
let notNsRe = /[^0-2]/g;
console.log(notNums.match(notNsRe));

let specialNums = "1!2@3#4$5%678910";
let specialNumsRe = /[^0-9]/g;
console.log(specialNums.match(specialNumsRe));

let practice = "Os1 Os1Os Os2 Os8 Os8Os";
let practiceRe = /Os[5-9]Os/g;
console.log(practice.match(practiceRe));
```

137 - Regular Expression – Ranges Part 2

ال Code

```
/*
  Regular Expression
```

```
  Ranges

  - Part 1
  (X|Y) => X Or Y
  [0-9] => 0 To 9
  [^0-9] => Any Character Not 0 To 9
  Practice

  - Part 2
  [a-z]
  [^a-z]
  [A-Z]
  [^A-Z]
  [abc]
  [^abc]

*/

let myString = "AaBbcdefG123!234%^&*";
let atozSmall = /[a-z]/g;
let NotAtozSmall = /[^a-z]/g;
let atozCapital = /[A-Z]/g;
let NotAtozCapital = /[^A-Z]/g;
let aAndcAnde = /[ace]/g;
let NotaAndcAnde = /[^ace]/g;
let lettersCapsAndSmall = /[a-zA-Z]/g;
```

```
let numsAndSpecials = /[^a-zA-Z]/g;
let specials = /[^a-zA-Z0-9]/g;
console.log(myString.match(atozSmall));
console.log(myString.match(NotAtozSmall));
console.log(myString.match(atozCapital));
console.log(myString.match(NotAtozCapital));
console.log(myString.match(aAndcAnde));
console.log(myString.match(NotaAndcAnde));
console.log(myString.match(lettersCapsAndSmall));
console.log(myString.match(numsAndSpecials));
```

```
console.log(myString.match(specials));
```

138 - Regular Expression – Character Classes Part 1

ال Code

```
/*
  Regular Expression
  Character Classes
```

```
  . => 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.

*/


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

let dot = /./g;

let word = /\w/g;

let valid = /\w@\w.(com|net)/g;

console.log(email.match(dot));

console.log(email.match(word));
```

```
console.log(email.match(valid));
```

Code ال

```
/*
  Regular Expression
  Character Classes
  \b => matches at the beginning or end of a word.
  \B => matches NOT at the beginning/end of a word.

  Test Method
  pattern.test(input)
*/

let names = "Sayed 1Spam 2Spam 3Spam Spam4 Spam5 Osama Ahmed Aspamo";
let re = /(\bspam|spam\b)/ig;
console.log(names.match(re));

console.log(re.test(names));
console.log(/(\bspam|spam\b)/ig.test("Osama"));
console.log(/(\bspam|spam\b)/ig.test("1Spam"));
console.log(/(\bspam|spam\b)/ig.test("Spam1"));
```

140 - Regular Expression – Quantifiers Part 1

Code الـ

```
/*
  Regular Expression


  Quantifiers
  n+    => One Or More
  n*    => zero or more
  n?    => zero or one
*/


let mails = "o@nn.sa osama@gmail.com elzero@gmail.net
osama@mail.ru"; // All Em      ails
// let mailsRe = /\w+@\w+.(com|net)/ig;
let mailsRe = /\w+@\w+.\w+/ig;
console.log(mails.match(mailsRe));


let nums = "0110 10 150 05120 0560 350 00"; // 0 Numbers Or
No 0
let numsRe = /0\d*0/ig;
console.log(nums.match(numsRe));
```

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

let urlsRe = /(https?:\/\/)?(www.)?\w+.\w+/ig;

console.log(urls.match(urlsRe));
```

141 - Regular Expression – Quantifiers Part 2

Code ال

```
/*
  Regular Expression

  Quantifiers
  n{x}   => Number of
  n{x,y} => Range
  n{x,}  => At Least x
*/

let serials = "S100S S3000S S50000S S9500000S";

console.log(serials.match(/s\d{3}s/ig)); // S[Three Number]S
```

```
console.log(serials.match(/s\d{4,5}s/ig)); // S[Four Or Five Number]S
```

```
console.log(serials.match(/s\d{4,}s/ig)); // S[At Least Four]S
```

142 - Regular Expression – Quantifiers Part 3

Code الـ

```
/*
  Regular Expression

  Quantifiers
  $  => End With Something
  ^  => Start With Something
  ?= => Followed By Something
  ?! => Not Followed By Something
*/
```

```
let myString = "We Love Programming";
let names = "1OsamaZ 2AhmedZ 3Mohammed 4MoustafaZ 5GamalZ";

console.log(/ing$/ig.test(myString));
console.log(/^we/ig.test(myString));
console.log(/lz$/ig.test(names));
console.log(/^\d/ig.test(names));

console.log(names.match(/\d\w{5}(?=Z)/ig));
```

```
console.log(names.match(/\d\w{8}(?!Z)/ig));
```

143 - Regular Expression – Replace With Pattern

Code ال

```
/*
  Regular Expression

  - replace
  - replaceAll
*/

let txt = "We Love Programming And @ Because @ Is Amazing";
console.log(txt.replace("@", "JavaScript"));
console.log(txt.replaceAll("@", "JavaScript"));
```

```
let re = /@/ig;

console.log(txt.replaceAll(re, "JavaScript"));
```

```
console.log(txt.replaceAll(/@/ig, "JavaScript"));
```

```

```

144 - Regular Expression – Form Validation

ال Code

```
<form id="register" action="" method="get">

  <input type="text" id="phone" name="the-phone"
maxlength="15" />

  <input type="submit" value="Register" />

</form>
```

```
/*

  Regular Expression

  - Input Form Validation Practice

*/


document.getElementById("register").onsubmit = function () {
```

```
let phoneInput = document.getElementById("phone").value;
let phoneRe = /\(\d{4}\)\s\d{3}-\d{4}/; // (1234) 567-8910
let validationResult = phoneRe.test(phoneInput);
if (validationResult === false) {
  return false;
}
return true;
```

}

145 - Test Your Regular Expression And Discussions

المواقع الخاصة بإختبار ال Regular Expression

- Regexr
- RegExTester
- RegEx101

146 - Regular Expression Challenge

التحدي

```
/*
  Regular Expression
  - Challenge
*/


let url1 = 'elzero.org';
let url2 = 'http://elzero.org';
let url3 = 'https://elzero.org';
let url4 = 'https://www.elzero.org';
```

```
let url5 =
'https://www.elzero.org:8080/articles.php?id=100&cat=topics'
;


let re = //;


console.log(url1.match(re));

console.log(url2.match(re));

console.log(url3.match(re));

console.log(url4.match(re));

console.log(url5.match(re));
```

147 - OOP Introduction

ابدأ الآن

محتوى الدرس

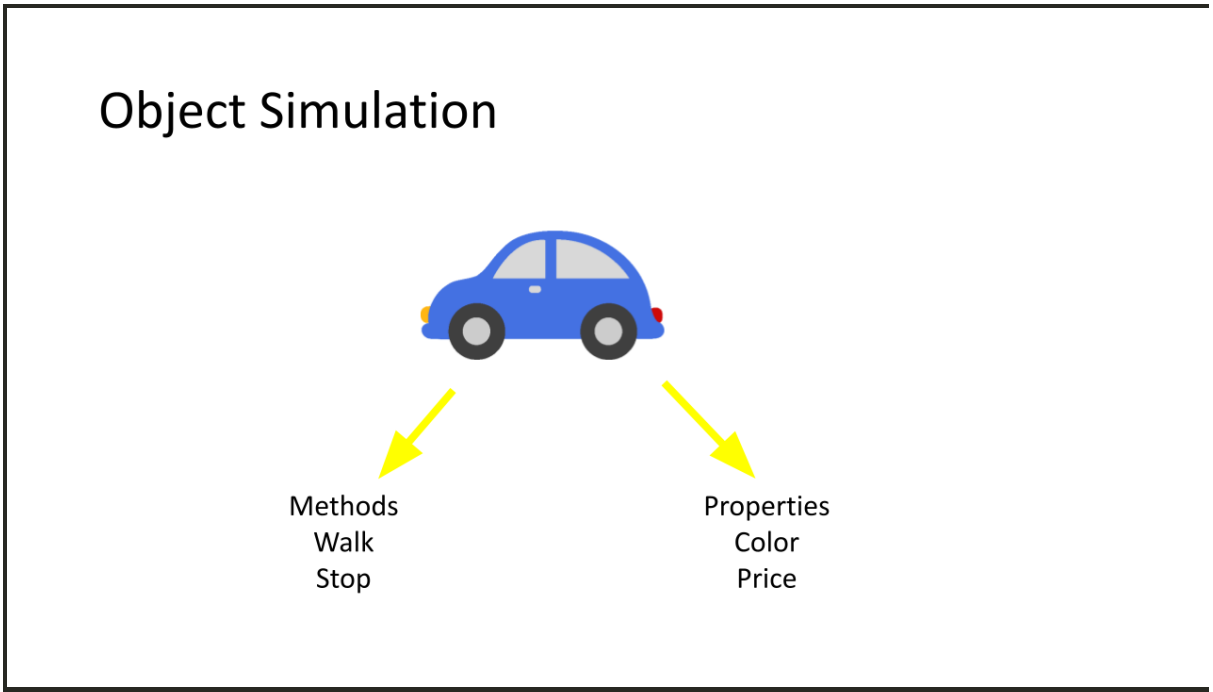
ألدرس يحتوي فقط على Slide يمكنك تحميله من **هنا**

JavaScript OOP

Elzero Web School

What Is OOP ?

- OOP Stand For => Object Oriented Programming
- OOP Is A Paradigm or Style Of Code
- OOP Use The Concept Of Object To Design A Computer Program
- Its Not New => Simula Is The First OOP Programming Language At 1960
- Some Languages Support OOP and Some Not
- Object Is A Package Contains Properties and Functions That Work Together To Produce Something in Your Application. Functions Here Called Methods



148 - Constructor Function Introduction

ابدأ الآن

محتوى الدرس

```
/*
  Constructor Function
*/

function User(id, username, salary) {
  this.i = id;
  this.u = username;
  this.s = salary + 1000;
}

let userOne = new User(100, "Elzero", 5000);
let userTwo = new User(101, "Hassan", 6000);
let userThree = new User(102, "Sayed", 7000);

console.log(userOne.i);
console.log(userOne.u);
console.log(userOne.s);

console.log(userTwo.i);
console.log(userTwo.u);
console.log(userTwo.s);
```

```
console.log(userThree.i);
console.log(userThree.u);
console.log(userThree.s);

// const userOne = {
//   id: 100,
//   username: "Elzero",
//   salary: 5000,
// };

// const userTwo = {
//   id: 101,
//   username: "Hassan",
//   salary: 6000,
// };

// const userThree = {
//   id: 102,
//   username: "Sayed",
//   salary: 7000,
// };
```

149 - Constructor Function New Syntax

محتوى الدرس

```
/*
  Constructor Function
  - New Syntax
*/

class User {
  constructor(id, username, salary) {
    this.i = id;
```

```
    this.u = username;
    this.s = salary + 1000;
  }
}

let userOne = new User(100, "Elzero", 5000);

console.log(userOne.i);
console.log(userOne.u);
console.log(userOne.s);

console.log(userOne instanceof User);
```

```
console.log(userOne.constructor === User);
```

150 - Deal With Properties And Methods

محتوى الدرس

```
/*
  Constructor Function
  - Deal With Properties And Methods
*/

class User {
  constructor(id, username, salary) {
    // Properties
    this.i = id;
    this.u = username || "Unknown";
    this.s = salary < 6000 ? salary + 500 : salary;
    this.msg = function () {
      return `Hello ${this.u} Your Salary Is ${this.s}`;
    };
  }
  // Methods
```

```
  writeMsg() {
    return `Hello ${this.u} Your Salary Is ${this.s}`;
  }
}

let userOne = new User(100, "Elzero", 5000);
let userTwo = new User(101, "", 6000);

console.log(userOne.u);
console.log(userOne.s);
console.log(userOne.msg());
console.log(userOne.writeMsg());

console.log(userTwo.u);
console.log(userTwo.s);
console.log(userTwo.msg); // Native Code

console.log(userTwo.writeMsg); // Native Code
```

151 - Update Properties & Built In Constructors

محتوى الدرس

```
/*
  Constructor Function
  - Update Properties
  - Built In Constructors
*/

class User {
  constructor(id, username, salary) {
    this.i = id;
    this.u = username;
    this.s = salary;
  }
  updateName(newName) {
    this.u = newName;
  }
}

let userOne = new User(100, "Elzero", 5000);
```

```
console.log(userOne.u);
userOne.updateName("Osama");
console.log(userOne.u);

let strOne = "Elzero";
let strTwo = new String("Elzero");

console.log(typeof strOne);
console.log(typeof strTwo);

console.log(strOne instanceof String);
console.log(strTwo instanceof String);

console.log(strOne.constructor === String);
console.log(strTwo.constructor === String);
```

152 - Class Static Properties And Methods

محتوى الدرس

```
/*
  Class
  - Static Properties And Methods
*/

class User {
  // Static Property
  static count = 0;

  constructor(id, username, salary) {
    this.i = id;
    this.u = username;
    this.s = salary;
    User.count++;
```

```
  }

  // Static Methods
  static sayHello() {
    return `Hello From Class`;
  }
  static countMembers() {
    return `${this.count} Members Created`;
  }
}

let userOne = new User(100, "Elzero", 5000);
let userTwo = new User(101, "Ahmed", 5000);
let userThree = new User(102, "Sayed", 5000);

console.log(userOne.u);
console.log(userTwo.u);
console.log(userOne.count);
console.log(User.count);
console.log(User.sayHello());
```

```
console.log(User.countMembers());
```

153 - Class Inheritance

محتوى الدرس

```
/*
  Class
  - Inheritance
*/

// Parent Class
class User {
  constructor(id, username) {
    this.i = id;
    this.u = username;
```

```
  }
  sayHello() {
    return `Hello ${this.u}`;
  }
}

// Derived Class
class Admin extends User {
  constructor(id, username, permissions) {
    super(id, username);
    this.p = permissions;
  }
}

class Superman extends Admin {
  constructor(id, username, permissions, ability) {
    super(id, username, permissions);
    this.a = ability;
  }
}

let userOne = new User(100, "Elzero");
let adminOne = new Admin(110, "Mahmoud", 1);

console.log(userOne.u);
console.log(userOne.sayHello());
```

```
console.log("####");
console.log(adminOne.i);
console.log(adminOne.u);
console.log(adminOne.p);

console.log(adminOne.sayHello());
```

154 - Class Encapsulation

محتوى الدرس

```
/*
  Encapsulation
```

```
  - Class Fields Are Public By Default

  - Guards The Data Against Illegal Access.

  - Helps To Achieve The Target Without Revealing Its
Complex Details.

  - Will Reduce Human Errors.

  - Make The App More Flexible And Manageable.

  - Simplifies The App.
*/


class User {

  // Private Property

  #e;

  constructor(id, username, eSalary) {

    this.i = id;

    this.u = username;

    this.#e = eSalary;

  }

  getSalary() {

    return parseInt(this.#e);

  }

}


let userOne = new User(100, "Elzero", "5000 Gneh");


console.log(userOne.u);
```

```
console.log(userOne.getSalary() * 0.3);
```

155 - Prototype Introduction

محتوى الدرس

```
/*
  Prototype
  - Introduction
  - Prototypes are the mechanism by which JavaScript objects
    inherit features from one another.
*/
```

```
class User {
  constructor(id, username) {
    this.i = id;
    this.u = username;
  }
  sayHello() {
    return `Hello ${this.u}`;
  }
}

let userOne = new User(100, "Elzero");
console.log(userOne.u);

console.log(User.prototype);

let strOne = "Elzero";

console.log(String.prototype);
```

محتوى الدرس

```
/*
  Prototype
  - Add To Prototype Chain
  - Extend Built In Constructors Features
*/


class User {
  constructor(id, username) {
    this.i = id;
    this.u = username;
  }
  sayHello() {
    return `Hello ${this.u}`;
  }
}

let userOne = new User(100, "Elzero");
console.log(userOne.u);
```

```
console.log(User.prototype);
console.log(userOne);

User.prototype.sayWelcome = function () {
  return `Welcome ${this.u}`;
};

Object.prototype.love = "Elzero Web School";

String.prototype.addDotBeforeAndAfter = function (val) {
  return `.${this}.`;
};
```

```
let myString = "Elzero";
```

157 - Object Meta Data And Descriptor Part 1

محتوى الدرس

```
/*
  Object Meta Data And Descriptor
  - writable
  - enumerable
  - configurable [Cannot Delete Or Reconfigure]
*/

const myObject = {
  a: 1,
  b: 2,
};

Object.defineProperty(myObject, "c", {
  writable: false,
  enumerable: true,
  configurable: false,
```

```
  value: 3,
});

// Object.defineProperty(myObject, "c", {
//   writable: false,
//   enumerable: true,
//   configurable: true, <= Cannot redefine property
//   value: 3,
// });

myObject.c = 100;

console.log(delete myObject.c);

for (let prop in myObject) {
  console.log(prop, myObject[prop]);
}
```

```
console.log(myObject);
```

158 - Object Meta Data And Descriptor Part 2

محتوى الدرس

```
/*
  Object Meta Data And Descriptor
  - Define Multiple Properties
  - Check Descriptors
*/

const myObject = {
  a: 1,
  b: 2,
};

Object.defineProperties(myObject, {
  c: {
    configurable: true,
```

```
    value: 3,
  },
  d: {
    configurable: true,
    value: 4,
  },
  e: {
    configurable: true,
    value: 5,
  },
});

console.log(myObject);

console.log(Object.getOwnPropertyDescriptor(myObject, "d"));
```

```
console.log(Object.getOwnPropertyDescriptors(myObject));
```

159 - Date And Time Introduction

محتوى الدرس

```
/*
  Date And Time
  - Date Constructor

  Static Methods
  - Date.now()

  - To Track Time You Need Starting Point
  - Epoch Time Or Unix Time In Computer Science Is The
Number of Seconds Since January 1, 1970.
  - Why 1970 [829 Days To 136 Years]

  Search For
  - Year 2038 Problem in Computer Science.
*/
```

```
let dateNow = new Date();

console.log(dateNow);

console.log(Date.now()); // 1000 Mill = 1 Second

let seconds = Date.now() / 1000; // Number Of Seconds
console.log(`Seconds ${seconds}`);

let minutes = seconds / 60; // Number Of Minutes
console.log(`Minutes ${minutes}`);

let hours = minutes / 60; // Number Of Hours
console.log(`Hours ${hours}`);

let days = hours / 24; // Number Of Days
console.log(`Days ${days}`);

let years = days / 365; // Number Of Years
console.log(`Years ${years}`);
```

160 - Get Date And Time

محتوى الدرس

```
/*
  Date And Time
  - getTime() => Number Of Milliseconds
  - getDate() => Day Of The Month
  - getFullYear()
  - getMonth() => Zero Based
  - getDay() => Day Of The Week
  - getHours()
  - getMinutes()
  - getSeconds()
*/

let dateNow = new Date();
let birthday = new Date("Oct 25, 82");
let dateDiff = dateNow - birthday;

console.log(dateDiff);
console.log(dateDiff / 1000 / 60 / 60 / 24 / 365);
```

```
console.log(dateNow);
console.log(dateNow.getTime());
console.log(dateNow.getDate());
console.log(dateNow.getFullYear());
console.log(dateNow.getMonth());
console.log(dateNow.getDay());
console.log(dateNow.getHours());
console.log(dateNow.getMinutes());
```

```
console.log(dateNow.getSeconds());
```

161 - Set Date And Time

محتوى الدرس

```
/*

  Date And Time

  - setTime(Milliseconds)

  - setDate() => Day Of The Month [Negative And Positive]

  - setFullYear(year, month => Optional [0-11], day =>
Optional [1-31])

  - setMonth(Month [0-11], Day => Optional [1-31]) [Negative
And Positive]

  - setHours(Hours [0-23], Minutes => Optional [0-59],
Seconds => Optional [0-59], MS => Optional [0-999])

  - setMinutes(Minutes [0-59], Seconds => Optional [0-59],
MS => Optional [0-999])

  - setSeconds(Seconds => [0-59], MS => Optional [0-999])

*/


let dateNow = new Date();

console.log(dateNow);
```

```
console.log("#".repeat(66));


// dateNow.setTime(0);
// console.log(dateNow);


// console.log("#".repeat(66));


// dateNow.setTime(10000);
// console.log(dateNow);


// console.log("#".repeat(66));


// dateNow.setDate(35);
// console.log(dateNow);


// dateNow.setFullYear(2020, 13);
// console.log(dateNow);


dateNow.setMonth(15);

console.log(dateNow);
```

162 - Formatting Date And Time

محتوى الدرس

```
/*
  Date And Time


  new Date(timestamp)
  new Date(Date String)
  new Date(Numeric Values)


  Format
  - "Oct 25 1982"
```

```
  - "10/25/1982"
  - "1982-10-25" => ISO International Standard
  - "1982 10"
  - "1982"
  - "82"
  - 1982, 9, 25, 2, 10, 0
  - 1982, 9, 25
  - "1982-10-25T06:10:00Z"

  Date.parse("String") // Read Date From A String
*/

console.log(Date.parse("Oct 25 1982"));

let date1 = new Date(0);
console.log(date1);

let date2 = new Date(404344800000);
console.log(date2);

let date3 = new Date("10-25-1982");
console.log(date3);

let date4 = new Date("1982-10-25");
console.log(date4);
```

```
let date5 = new Date("1982-10");
console.log(date5);

let date6 = new Date("82");
console.log(date6);

let date7 = new Date(1982, 9, 25, 2, 10, 0);
console.log(date7);

let date8 = new Date(1982, 9, 25);
console.log(date8);

let date9 = new Date("1982-10-25T06:10:00Z");
console.log(date9);
```

ابدأ الآن

محتوى الدرس

```
/*
  Date And Time
  - Track Operations Time

  Search
  - performance.now()
  - performance.mark()
*/

// Start Time
let start = new Date();

// Operation
for (let i = 0; i < 100000; i++) {
  // document.write(`<div>${i}</div>`);
  let div = document.createElement("div");
  div.appendChild(document.createTextNode(i));
  document.body.appendChild(div);
}

// Time End
let end = new Date();
```

```
// Operation Duration
let duration = end - start;
```

```
console.log(duration);
```

```
```

164 - Generator Function Introduction

محتوى الدرس

```
/*
  Generators
```

```
  - Generator Function Run Its Code When Required.

  - Generator Function Return Special Object [Generator
Object]

  - Generators Are Iterable

*/


function* generateNumbers() {

  yield 1;

  console.log("Hello After Yield 1");

  yield 2;

  yield 3;

  yield 4;

}


let generator = generateNumbers();


console.log(typeof generator);

console.log(generator);


console.log(generator.next());

console.log(generator.next());

console.log(generator.next());

console.log(generator.next());

console.log(generator.next());


for (let value of generateNumbers()) {
```

```
  console.log(value);
}

for (let value of generator) {
  console.log(value);
}
```

```
```

165 - Delegate Generator Function

محتوى الدرس

```
/*
```

```
  Generators
  - Delegate Generator
*/

function* generateNums() {
  yield 1;
  yield 2;
  yield 3;
}

function* generateLetters() {
  yield "A";
  yield "B";
  yield "C";
}

function* generateAll() {
  yield* generateNums();
  yield* generateLetters();
  yield* [4, 5, 6];
}

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.next());
console.log(generator.return("Z"));
console.log(generator.next());
console.log(generator.next());
```

```
console.log(generator.next());
```


محتوى الدرس

```
/*
  Generators
```

```
  - Generate Infinite Numbers
  - Use Return Inside Generators
*/

function* generateNumbers() {
  // yield 1;
  // yield 2;
  // return "A";
  // yield 3;
  // yield 4;
  let index = 0;

  while (true) {
    yield index++;
  }
}

let generator = generateNumbers();

console.log(generator.next());
console.log(generator.next());
console.log(generator.next());

console.log(generator.next());
```

167 - Modules Import And Export

محتوى الدرس

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
```

```
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <link rel="stylesheet" href="main.css" />
    <title>Learn JavaScript</title>
  </head>
  <body>
    <script src="main.js" type="module"></script>
    <script src="app.js" type="module"></script>
  </body>
</html>
```

main.js

```
/*
  Modules
  - Import And Export
*/

let a = 10;
let arr = [1, 2, 3, 4];

function saySomething() {
  return `Something`;
}
```

```
export { a, arr, saySomething };
```

app.js

```
import { a, arr, saySomething as s } from "./main.js";

console.log(a);
console.log(arr);
console.log(s());
```

168 - Named vs Default Export And Import All

ابدأ الآن

محتوى الدرس

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
    <link rel="stylesheet" href="main.css" />
    <title>Learn JavaScript</title>
  </head>
  <body>
    <script src="main.js" type="module"></script>
    <script src="app.js" type="module"></script>
  </body>
</html>
```

main.js

```
/*
```

```
  Modules
  - Export Alias
  - Named Export
  - Default Export
  - Import All
*/

let a = 10;
let arr = [1, 2, 3, 4];

function saySomething() {
  return `Something`;
}

export { a as myNumber, arr, saySomething };

export default function () {
  return `Hello`;
}
```

app.js

```
// import elzero, { myNumber, arr, saySomething as s } from
"./main.js";
```

```
// console.log(myNumber);
// console.log(arr);
// console.log(s());
// console.log(elzero());

import * as all from "./main.js";

console.log(all);

console.log(all.myNumber);
console.log(all.arr);
```

169 - What Is JSON

تعلم الآن

محتوى الدرس

main.js

```
/*
  What Is JSON ?
  - JavaScript Object Notation
  - Format For Sharing Data Between Server And Client
  - JSON Derived From JavaScript
  - Alternative To XML
  - File Extension Is .json

  Why JSON ?
  - Easy To Use And Read
  - Used By Most Programming Languages And Its Frameworks
  - You Can Convert JSON Object To JS Object And Vice Versa

  JSON vs XML
  ====================================================
  = Text Based Format      = Markup Language         =
  = Lightweight            = Heavier                 =
  = Does Not Use Tags      = Using Tags              =
  = Shorter                = Not Short               =
  = Can Use Arrays         = Cannot Use Arrays       =
```

```
  = Not Support Comments    = Support Comments       =

  ==========================================================
```

```
*/
```

test.json

```
{
  "widget": {
    "debug": "on",
    "window": {
      "title": "Sample Konfabulator Widget",
      "name": "main_window",
      "width": 500,
      "height": 500
    },
    "image": {
      "src": "Images/Sun.png",
      "name": "sun1",
      "hOffset": 250,
      "vOffset": 250,
      "alignment": "center"
    },
    "text": {
      "data": "Click Here",
      "size": 36,
```

```
      "style": "bold",
      "name": "text1",
      "hOffset": 250,
      "vOffset": 100,
      "alignment": "center",
      "onMouseUp": "sun1.opacity = (sun1.opacity / 100) *
90;"
    }
  }
}
```

test.xml

```
<widget>
  <debug>on</debug>
  <window title="Sample Konfabulator Widget">
    <name>main_window</name>
    <width>500</width>
    <height>500</height>
  </window>
  <image src="Images/Sun.png" name="sun1">
    <hOffset>250</hOffset>
    <vOffset>250</vOffset>
    <alignment>center</alignment>
  </image>
```

```
  <text data="Click Here" size="36" style="bold">
    <name>text1</name>
    <hOffset>250</hOffset>
    <vOffset>100</vOffset>
    <alignment>center</alignment>
    <onMouseUp>
      sun1.opacity = (sun1.opacity / 100) * 90;
    </onMouseUp>
  </text>
```

```
</widget>
```

170 - JSON Syntax And Compare With JS Object

محتوى الدرس

main.js

```
/*
  JSON Syntax
  - Data Added Inside Curly Braces {  }
  - Data Added With Key : Value
  - Key Should Be String Wrapped In Double Quotes
  - Data Separated By Comma
  - Square Brackets [] For Arrays
  - Curly Braces {} For Objects

  Available Data Types
  - String
  - Number
  - Object
  - Array
  - Boolean Values
  - null
```

```
*/
```

test.json

```
{
  "string": "Elzero",
  "number": 100,
  "object": { "EG": "Giza", "KSA": "Riyadh" },
  "array": ["HTML", "CSS", "JS"],
  "boolean": true,
  "null": null
}
```

171 - What Is API

تعلم الآن

محتوى الدرس

```
/*
  JSON
  - API Overview
  - Tools To Test API
  - Preview Github API
*/
```

```
```

172 - Parse And Stringify

تعلم الآن

محتوى الدرس

```
/*
  JSON
  - JSON.parse => Convert Text Data To JS Object
  - JSON.stringify => Convert JS Object To JSON
*/


// Get From Server
const myJsonObjectFromServer = '{"Username": "Osama", "Age": 39}';
console.log(typeof myJsonObjectFromServer);
console.log(myJsonObjectFromServer);


// Convert To JS Object
const myJsObject = JSON.parse(myJsonObjectFromServer);
console.log(typeof myJsObject);
console.log(myJsObject);


// Update Data
myJsObject["Username"] = "Elzero";
myJsObject["Age"] = 40;


// Send To Server
```

```
const myJsonObjectToServer = JSON.stringify(myJsObject);
console.log(typeof myJsonObjectToServer);
```

```
console.log(myJsonObjectToServer);
```

173 - Asynchronous vs Synchronous Programming

محتوى الدرس

```
/*
  To Understand Ajax, Fetch, Promises


  Asynchronous vs Synchronous Programming
  - Meaning
```

```
  Synchronous

  - Operations Runs in Sequence

  - Each Operation Must Wait For The Previous One To
Complete

  - Story From Real Life


  Asynchronous

  - Operations Runs In Parallel

  - This Means That An Operation Can Occur while Another One
Is Still Being Processed

  - Story From Real Life


  - Facebook As Example

  - Simulation


  Search

  - JavaScript Is A Single-Threaded

  - Multi Threaded Languages
*/


// Synchronous


// console.log("1");

// console.log("2");

// window.alert("Operation");

// console.log("3");
```

```
// Asynchronous
console.log("1");
console.log("2");
setTimeout(() => console.log("Operation"), 3000);
console.log("3");
```

174 - Call Stack And Web API

محتوى الدرس

```
/*
  To Understand Ajax, Fetch, Promises


  Call Stack || Stack Trace
  -- JavaScript Engine Uses A Call Stack To Manage Execution Contexts
  -- Mechanism To Make The Interpreter Track Your Calls
  -- When Function Called It Added To The Stack
  -- When Function Executed It Removed From The Stack
  -- After Function Is Finished Executing The Interpreter Continue From The Last Point
  -- Work Using LIFO Principle => Last In First Out
  -- Code Execution Is Synchronous.
  -- Call Stack Detect Web API Methods And Leave It To The Browser To Handle It


  Web API
  -- Methods Available From The Environment => Browser
*/


setTimeout(() => {
  console.log("Web API");
}, 0);
```

```
function one() {
  console.log("One");
}
function two() {
  one();
  console.log("Two");
}
function three() {
  two();
  console.log("Three");
}

three();

/*
==================================
console.log("One");
==================================
function one() {
  console.log("One");
}
==================================
function two() {
  one();
  console.log("Two");
}
```

```
==================================
function three() {
  two();
  console.log("Three");
}
==================================
*/

console.log("#####");
console.log("One");
console.log("Two");

console.log("Three");
```

175 - Event Loop And Callback Queue

محتوى الدرس

```
/*
  To Understand Ajax, Fetch, Promises


  Event Loop + Callback Queue


  Story
  - JavaScript Is A Single Threaded Language "All Operations
Executed in Single Thread"
  - Call Stack Track All Calls
  - Every Function Is Done Its Poped Out
  - When You Call Asynchronous Function It Sent To Browser
API
  - Asynchronous Function Like Settimeout Start Its Own
Thread
  - Browser API Act As A Second Thread
  - API Finish Waiting And Send Back The Function For
Processing
  - Browser API Add The Callback To Callback Queue
```

```
  - Event Loop Wait For Call Stack To Be Empty
  - Event Loop Get Callback From Callback Queue And Add It To Call Stack
  - Callback Queue Follow FIFO "First In First Out" Rule
*/

console.log("One");
setTimeout(() => {
  console.log("Three");
}, 0);
setTimeout(() => {
  console.log("Four");
}, 0);
console.log("Two");

setTimeout(() => {
  console.log(myVar);
}, 0);

let myVar = 100;
myVar += 100;
```

176 - What Is AJAX And Network Informations

محتوى الدرس

```
/*

  AJAX

  - Asynchronous JavaScript And XML

  - Approach To Use Many Technologies Together [HTML, CSS,
Js, DOM]

  - It Use "XMLHttpRequest" Object To Interact With The
Server

  - You Can Fetch Data Or Send Data Without Page Refresh

  - Examples

  --- Youtube Studio
```

```
  --- Google Drive
  --- Upload Article Photo
  --- Form Check Name

  Test new XMLHttpRequest();
  Request And Response
  Status Code
*/

let req = new XMLHttpRequest();

console.log(req);
```

177 - Request And Response From Real API

before you start you should know ready state numbers meanings

	UNSENT	Client has been created. open() not called yet.
1	OPENED	open() has been called.
2	HEADERS_RECEIVED	send() has been called, and headers and status are available.
3	LOADING	Downloading; responseText holds partial data.
4	DONE	The operation is complete.

status code mean the message that describe the status of the server’s response for example 200 means done

404 means not found and so on

محتوى الدرس

```
/*
  Ajax
  - Ready State => Status Of The Request
```

```
  [0] Request Not Initialized
  [1] Server Connection Established
  [2] Request Received
  [3] Processing Request
  [4] Request Is Finished And Response Is Ready
  - Status
  [200] Response Is Successful
  [404] Not Found
*/

let myRequest = new XMLHttpRequest();
myRequest.open("GET",
"https://api.github.com/users/elzerowebschool/repos");
myRequest.send();
console.log(myRequest);

myRequest.onreadystatechange = function () {
  // console.log(myRequest.readyState);
  // console.log(myRequest.status);
  if (this.readyState === 4 && this.status === 200) {
    console.log(this.responseText);
  }
};
```

178 - Loop On Data

محتوى الدرس

```
/*
  Ajax
  Loop On Data

  Search
  - Cross Origin API [CORS]
```

```
  - API Authentication
*/


let myRequest = new XMLHttpRequest();
myRequest.open("GET",
"https://api.github.com/users/elzerowebschool/repos");
myRequest.send();
myRequest.onreadystatechange = function () {
  if (this.readyState === 4 && this.status === 200) {
    // console.log(this.responseText);
    let jsData = JSON.parse(this.responseText);
    // console.log(jsData);
    for (let i = 0; i < jsData.length; i++) {
      let div = document.createElement("div");
      let repoName =
document.createTextNode(jsData[i].full_name);
      div.appendChild(repoName);
      document.body.appendChild(div);
    }
  }
};
```

179 - Callback Hell Or Pyramid Of Doom

محتوى الدرس

```
/*
  To Understand Ajax, Fetch, Promises

  Pyramid Of Doom || Callback Hell

  - What Is Callback
  - Callback Hell Example

  What Is Callback
```

```
  - A Function That Is Passed Into Another One As An
Argument To Be Executed Later

  - Function To Handle Photos

  --- [1] Download Photo From URL

  --- [2] Resize Photo

  --- [3] Add Logo To The Photo

  --- [4] Show The Photo In Website
*/


function makeItRed(e) {
  e.target.style.color = "red";
}


let p = document.querySelector(".test");
p.addEventListener("click", makeItRed);


function iamACallback() {
  console.log("Iam A Callback Function");
}


setTimeout(iamACallback, 2000);


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");
      }, 1000);
    }, 1000);
  }, 1000);
}, 1000);
```

180 - Promise Intro And Syntax

محتوى الدرس

```
/*

  Promise Intro And Syntax

  - Promise In JavaScript Is Like Promise In Real Life

  - Promise Is Something That Will Happen In The Future

  - Promise Avoid Callback Hell

  - Promise Is The Object That Represent The Status Of An
Asynchronous Operation And Its Resulting Value



  - Promise Status

  --- Pending: Initial State

  --- Fulfilled: Completed Successfully

  --- Rejected: Failed



  Story

  - Once A Promise Has Been Called, It Will Start In A
Pending State

  - The Created Promise Will Eventually End In A Resolved
State Or In A Rejected State

  - Calling The Callback Functions (Passed To Then And
Catch) Upon Finishing.



  - Then

  --- Takes 2 Optional Arguments [Callback For Success Or
Failure]

*/
```

```
// const myPromise = new Promise((resolveFunction,
rejectFunction) => {
//   let connect = false;
//   if (connect) {
//     resolveFunction("Connection Established");
//   } else {
//     rejectFunction(Error("Connection Failed"));
//   }
// }).then(
//   (resolveValue) => console.log(`Good ${resolveValue}`),
//   (rejectValue) => console.log(`Bad ${rejectValue}`)
// );


const myPromise = new Promise((resolveFunction,
rejectFunction) => {
  let connect = true;
  if (connect) {
    resolveFunction("Connection Established");
  } else {
    rejectFunction(Error("Connection Failed"));
  }
});


console.log(myPromise);
```

```
let resolver = (resolveValue) => console.log(`Good
${resolveValue}`);

let rejecter = (rejectValue) => console.log(`Bad
${rejectValue}`);


myPromise.then(resolver, rejecter);


myPromise.then(

  (resolveValue) => console.log(`Good ${resolveValue}`),

  (rejectValue) => console.log(`Bad ${rejectValue}`)

);


myPromise.then(

  (resolveValue) => console.log(`Good ${resolveValue}`),

  (rejectValue) => console.log(`Bad ${rejectValue}`)

);
```

181 - Promise – Then, Catch And Finally

محتوى الدرس

```
/*
  Promise Training


  We Will Go To The Meeting, Promise Me That We Will Find
The 4 Employees
  .then(We Will Choose Two People)
  .then(We Will Test Them Then Get One Of Them)
  .catch(No One Came)


  Then    => Promise Is Successfull Use The Resolved Data
  Catch   => Promise Is Failed, Catch The Error
  Finally => Promise Successfull Or Failed Finally Do
Something
*/


const myPromise = new Promise((resolveFunction,
rejectFunction) => {
```

```
  let employees = [];
  if (employees.length === 4) {
    resolveFunction(employees);
  } else {
    rejectFunction(Error("Number Of Employees Is Not 4"));
  }
});

myPromise
  .then((resolveValue) => {
    resolveValue.length = 2;
    return resolveValue;
  })
  .then((resolveValue) => {
    resolveValue.length = 1;
    return resolveValue;
  })
  .then((resolveValue) => {
    console.log(`The Choosen Emplyee Is ${resolveValue}`);
  })
  .catch((rejectedReason) => console.log(rejectedReason))
  .finally(console.log("The Operation Is Done"));
```

182 - Promise And XHR

محتوى الدرس

```
/*
  Promise And XHR
*/


const getData = (apiLink) => {
  return new Promise((resolve, reject) => {
    let myRequest = new XMLHttpRequest();
    myRequest.onload = function () {
      if (this.readyState === 4 && this.status === 200) {
        resolve(JSON.parse(this.responseText));
```

```
      } else {
        reject(Error("No Data Found"));
      }
    };

    myRequest.open("GET", apiLink);
    myRequest.send();
  });
};

getData("https://api.github.com/users/elzerowebschool/repos"
)
  .then((result) => {
    result.length = 10;
    return result;
  })
  .then((result) => console.log(result[0].name))
  .catch((rej) => console.log(rej));
```

محتوى الدرس

```
/*
  Fetch API
  - Return A Representation Of the Entire HTTP Response
*/

fetch("https://api.github.com/users/elzerowebschool/repos")
  .then((result) => {
    console.log(result);
    let myData = result.json();
    console.log(myData);
    return myData;
  })
  .then((full) => {
    full.length = 10;
    return full;
  })
  .then((ten) => {
    console.log(ten[0].name);
```

```
  });


// const getData = (apiLink) => {
//   return new Promise((resolve, reject) => {
//     let myRequest = new XMLHttpRequest();
//     myRequest.onload = function () {
//       if (this.readyState === 4 && this.status === 200) {
//         resolve(JSON.parse(this.responseText));
//       } else {
//         reject(Error("No Data Found"));
//       }
//     };

//     myRequest.open("GET", apiLink);
//     myRequest.send();
//   });
// };


//
getData("https://api.github.com/users/elzerowebschool/repos"
)
//   .then((result) => {
//     result.length = 10;
//     return result;
//   })
//   .then((result) => console.log(result[0].name))
```

```
//    .catch((rej) => console.log(rej));
```

184 - Promise – All, All Settled, Race

محتوى الدرس

```
/*
  Promise
```

```
  - All
  - All Settled
  - Race
*/

const myFirstPromise = new Promise((res, rej) => {
  setTimeout(() => {
    res("Iam The First Promise");
  }, 5000);
});

const mySecondPromise = new Promise((res, rej) => {
  setTimeout(() => {
    rej("Iam The Second Promise");
  }, 1000);
});

const myThirdPromise = new Promise((res, rej) => {
  setTimeout(() => {
    res("Iam The Third Promise");
  }, 2000);
});

// Promise.all([myFirstPromise, mySecondPromise,
myThirdPromise]).then(
//   (resolvedValues) => console.log(resolvedValues),
```

```
//   (rejectedValue) => console.log(`Rejected:
${rejectedValue}`)

// );



// Promise.allSettled([myFirstPromise, mySecondPromise,
myThirdPromise]).then(

//   (resolvedValues) => console.log(resolvedValues),

//   (rejectedValue) => console.log(`Rejected:
${rejectedValue}`)

// );



Promise.race([myFirstPromise, mySecondPromise,
myThirdPromise]).then(

  (resolvedValues) => console.log(resolvedValues),

  (rejectedValue) => console.log(`Rejected:
${rejectedValue}`)

);
```

185 - Async And Training

محتوى الدرس

```
/*

  Async

  - Async Before Function Mean This Function Return A
Promise

  - Async And Await Help In Creating Asynchronous Promise
Behavior With Cleaner Style

*/


// function getData() {

//   return new Promise((res, rej) => {

//     let users = [];

//     if (users.length > 0) {

//       res("Users Found");

//     } else {

//       rej("No Users Found");
```

```
//     }
//   });
// }


// getData().then(
//   (resolvedValue) => console.log(resolvedValue),
//   (rejectedValue) => console.log("Rejected " +
rejectedValue)
// );


// function getData() {
//   let users = ["Osama"];
//   if (users.length > 0) {
//     return Promise.resolve("Users Found");
//   } else {
//     return Promise.reject("No Users Found");
//   }
// }


// getData().then(
//   (resolvedValue) => console.log(resolvedValue),
//   (rejectedValue) => console.log("Rejected " +
rejectedValue)
// );


async function getData() {
  let users = [];
```

```
  if (users.length > 0) {
    return "Users Found";
  } else {
    throw new Error("No Users Found");
  }
}

console.log(getData());

getData().then(
  (resolvedValue) => console.log(resolvedValue),
  (rejectedValue) => console.log("Rejected " +
rejectedValue)
);
```

تعلم الآن

محتوى الدرس

```
/*
  Await
  - Await Works Only Inside Asnyc Functions
  - Await Make JavaScript Wait For The Promise Result
  - Await Is More Elegant Syntax Of Getting Promise Result
*/

const myPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    // resolve("Iam The Good Promise");
    reject(Error("Iam The Bad Promise"));
  }, 3000);
});

async function readData() {
  console.log("Before Promise");
  // myPromise.then((resolvedValue) =>
console.log(resolvedValue));
  // console.log(await myPromise);
  console.log(await myPromise.catch((err) => err));
  console.log("After Promise");
}
```

```
readData();
```

187 - Try, Catch, Finally With Fetch

محتوى الدرس

```
/*
  Async & Await With Try, Catch, Finally
*/

const myPromise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Iam The Good Promise");
  }, 3000);
});

// async function readData() {
//   console.log("Before Promise");
//   try {
//     console.log(await myPromise);
//   } catch (reason) {
//     console.log(`Reason: ${reason}`);
//   } finally {
//     console.log("After Promise");
//   }
// }

// readData();

// "https://api.github.com/users/elzerowebschool/repos"

async function fetchData() {
```

```
  console.log("Before Fetch");
  try {
    let myData = await
fetch("https://api.github.com/users/elzerowebschool/repos");
    console.log(await myData.json());
  } catch (reason) {
    console.log(`Reason: ${reason}`);
  } finally {
    console.log("After Fetch");
  }
}
```

```
fetchData();
```

188 - The End And Advices

على خصم 100 دولار (لأول 50 طالب) على أي دورة تعليمية باستعمال الكود

ELZERO100C1

تعلم الآن

محتوى الدرس

```
/*
  The End
  - Other Information => Practice + Tutorials
  - Problem Solving
  - Search In Lessons
  - Advanced Books
*/
```

تمت بحمد الله