***JAVASCRIPT***

***Welcome To Javascript Course :***

JavaScript is the world’s most popular programming language .

JavaScript is the programming language of the web .

JavaScript is easy to learn .

This turorial will teach you javascript from basic to advanced .

***Why study javascript :***

Javascript is one of **3 languages** all web developers must learn :

1**- HTML** : To Define the content of web pages

2- **CSS**: Syle Content of web pages

3- **JAVASCRIPT** : To program the behavior of web pages

***Learning Speed :***

In This tutorial , the learning speed is you choice .

Evrything is up to you .

Always make sure you understand all the « try-it-yourself » examples .

The only way to bacome a clever programmer is to ***:***

***practice, practice, practice. code.code.code !***

***Javascript Introduction :***

Here we will see some examples of what js can do

**1 – JavaScript can change HTML Content**

**2- JavaScript Can Change HTML Attribute Values**

**3- JavaScript Can Change HTML Styles (CSS)**

***JavaScript Where to Put the code***

In Html , JavaScript code is inserted between

<script> and </script>

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="UTF-8" />

    <meta http-equiv="X-UA-Compatible" content="IE=edge" />

    <title>Learn Javascript</title>

    <script>

      function change() {

        document.getElementById("demo").innerHTML = "MY First js";

      }

    </script>

  </head>

  <body>

    <h1 id="demo">test</h1>

    <button onclick="change()">try it</button>

  </body>

</html>

Now don’t focus on code , all this we will learn its dettails , so here we use script tag in head to write js code , we can also put it in body

<body>

    <h1 id="demo">test</h1>

    <button onclick="change()">try it</button>

    <script>

      function change() {

        document.getElementById("demo").style.color = "dodgerblue";

      }

    </script>

  </body>

NOTE : placing scripts at the bottom of the <body> element improves the display speed , because script interpretation slows down the display .

**External JavaScript :**

External Scripts are practical when the same code is used in many different web pages .

Javasript files have the file extension **.js**

To use an external script , put the name of the script file in the src (source) attribute of a <script> tag :

And also this script we can put it in body or head .

<body>

    <h1 id="demo">This Text is for Testing</h1>

    <script src="main.js"></script>

  </body>

But if we put it in head we will get an error . which says that there is no content in our HTML page . because as we know the computer read the code by a linear manner , and we put scrpt in head so computer read script before reading the content , and here we will get an error .

So here the solution is , in js file :

window.onload = function () {

  document.querySelector("h1").style.color = "red";

};

So Here we will make sure that the code will run after displaying page content , so here the computer will find the element .

NOTE : Best Practice is putting the code at the bottom of the page for avoid slowing displaying the content .

***Comment and bad practice***

1- why comment : imagine with me that you have a code that it understadable or not clear so here we use the comment to

Make clear by expalanig the code in the comments

*Comments types :*

// single line comment

/\* mulilline comment

    1

    2

    3

    4

    5

    6

    \*/

Use ctrl + / 🡺 for do a comment .

NOTE : Single Line Comment is the most popular .

Comments bad practice !! 🡪 using comments in clear things , in this step it’s okey , but when you become a pro developer bad practice using many comments

***Output To Screen***

**Javascript Display Possibilities :**

* Writing into an HTML element, using innerHTML.

<p id="demo">my name is :</p>

document.getElementById(« demo »).innerHTML = « Bilal » ;

innerHTML : for access to an html element , (old html content will be deleted

* Writing into an alert box, using window.alert().

window.alert("Message From JS File");

You can use an alert box to display data :

but in reality we don’t use alert , because it stop displaying html content , and cause many things , but we do it by html and syle it by css and program it by js .

you can skip window keyword .

In JavaScript, The window object is the global scope (النطاق ) object , that means variables , properties, and methods , by default belong to the window object , this also means that specifying the window keyword is optional :

alert("Without window ");

Writing into the HTML output using document.write().

For testing purposes , it is convenient to use document.write () .

The document.write() method should only be used for testing

Writing into the browser console using console.log().

console.log("Hi From Js File");

Writing an error message to the console console.error()

The error() method wirtes an error message to the console .

The console is useful for testing purposes.

console.error("you made a made a mistake ");

The console.table() method displays tabular data as a table.

The console.warn() method writes a warnign to the console.

console.warn(

  "%cWarning , be carfull !!",

  "font-size:30px; padding:20px;font-family:cursive"

);

Stop stop , first let’s speak about styling console.log or console.warn ….., first thing you write you comment then write (,)then do styling in double quotes , and in you message write dedictive c « %c » before message that you want to style it ,examples :

We will do facbook console as an example :

console.log(

  "%cStop !",

  "color:red ; font-size:80px;font-weight:bold; -webkit-text-stroke:black 2px"

);

console.log(

  "%cThis is a browser feature intended for developers. If someone told you to copy and paste something here to enable a Facebook feature or 'hack' someone's account, it is a scam and will give them access to your Facebook account.",

  "font-size:20px;font-family:verdana"

);

And sure you can syle to words by diffrent syling :

console.log(

  "Hello From %cJS %cFile !   ",

  "color:red ; font-size:80px;font-weight:bold; -webkit-text-stroke:black 2px",

  "color:dodgerblue ; font-size:80px;font-weight:bold; -webkit-text-stroke:black 2px"

);

And sure you can stop styling using this trick for avoid stylilng a specific word in medium

console.log(

  " Hello %cmy name %c is bilal    ",

  " color : olive;font-size:40px ; font-family:cursive",

  ""

);

But i want Tell you that console.log isn’t a javascript method but it from web api (application proramming interface), for understand what is an API : imagine that you are in a restaurent so you want take a pizza , sure in resaurant you reserve what you want and the waiter go the kitchen and bring you order and

Here the waiter represent the api and the client represent us and the pizza represent the data .

Use console.clear() for clear the console .

This method also write « console was cleared «  in the console .

Use console.count()method counts the number of times console.count() is called.

*The count() method send the number to the console .*

Note :

You can add a label that will be included in the console view .

As a default value the label « Default » is used .

Sure you can make many counters with diffrent labels

See the example below .

for (let i = 0; i < 5; i++) {

  console.count("times");

}

console.count("bilal");

console.count("bilal");

The console.group() method starts a message group .

All new messages after this method will written inside this group .

And for close use console.groupEnd()

Note :

You can add groups in a group and insed groups there are groups .

See The example Below :

console.group("first gourpe");

console.log(1);

console.log(2);

console.log(3);

console.group("names");

console.log("ossam");

console.log("Bilal");

console.log("Ahamd");

console.groupEnd();

console.groupEnd();

console.group("second gourpe");

console.log(10);

console.log(20);

console.log(30);

console.groupEnd();

console.group("third gourpe");

console.log(100);

console.log(200);

console.log(300);

console.groupEnd();

The console.g() method starts a message group .

console.groupCollapsed();

console.error("you did a mistake ");

console.log("hello");

console.groupEnd();

console.group();

console.error("you did a mistake ");

console.log("hello");

console.groupEnd();

console.warn("hello");

Note :

The only Diffrence (from my search) , between console.group() and console.groupcollapsed is console.group opend by default, which mean when you open the console you find console.group opened and sure you can close it .

but console.groupcollapsed is closed by default and sure you can open it .

The info() method writes a message to the console.

Note :

Console.info() and console.log are the same .

console.info("Hello from JS File");

*The* ***time()*** *method starts a timer in the console view* .

*The* ***time()*** *method allows you to time code for testing purposes .*

Note :

You can run many timers at the same time .

Use The label parameter to name diffrent timers .

Use console.timeEnd() method for stop the timer (don’t forget to specify the timer labal if you have many timers )

console.time("First timer");

for (let i = 0; i < 10000; i++) {

  if (i === 5000) {

    console.time("second timer");

  }

  console.log("A message");

}

console.timeEnd("second timer");

console.timeEnd("First timer");

*EcmaSecript / ES6*

Ecma is an international industry association dedicated to the standardization of information and communication systems .

Wich mean Es responsible fror setting appropriate standards for information and those informations can be programming language . and 6 mean ecmasecript version .

You can surf this website : es6-features.org this website give you all the newest features in es6 and give a small comparision with es5 , you can find there that some code was hard to do and know become easy , and you can find more than this like adding

A feature to js language and more ….. and know let speak about browsers support , are all browsers support these features ?

*Answear : no , not all (modern browsers only), but there are a website « babel js «  that compile modern js version code to legacy code ‘old code’ . and there many other wesite like es6console.com .*

***Knwo we will pass to practice .***

*Know focus you attention here ! ! !*

*Imagine with me that you work by many languages , among those languages ‘’python’’*

*Python code :*

myname = "Bilal"

print(f "Hello {myname}")

*let do that by js :*

var myname = "Bilal";

console.log("Hello" + myname);

*as you see There is no similarity between python code and js code , in python code is more comfortable ‘you dont have to close double quotes ..} , but in js the code isn’t comfortable .*

*know let use ecmasecript standards for print this message :*

var myname = "Bilal";

console.log(`Hello ${myname}`);

as you see codes are almost the same , and you can find some language wich has the same code for print messages .

finnaly : when language work by standards it start be more

Similar to other languages . There some features in other languages but not in js but with these standards these features become available in js programming too . and sure in this course we will learn the standards code and descover how legacy code is writted .

**Data Types And Typof Operator**

You can use The Javascript **typeof** operator to find the type of a javascript variable .

The **typeof** operator returns the type of a variable or an expression , so we will use it , to descover all data types :

**1- javascript numbers :**

Javascript has only one type of numbers . (not like C programming)

Numbers can be written with, or without decimals

console.log(typeof 17);     // Result : number

console.log(typeof 25.6);   // Result : number

**2- javascript Strings :**

A string (or a text string ) is a series of characters like « Bilal el « .

String are written with Quotes . you can use single or double quotes .

console.log(typeof "bilal"); // Result : String

console.log(typeof 'bilal'); // Result : String

**3- javascript Booleans :**

We use boolean data in yes no cases,maybe yes or no / true or false

console.log(typeof true);  // Result : Boolean

console.log(typeof false); // Result : Bolean

**4- Array 🡪 Object  :**

As we know array is like a bowl that contain value that are stored by a squential manner in memory .

Javascript arrays are written with square brackets [] .

Array items are sparated by commas . 🡺 ,

The following code declares (creates) an array called **cars ,** contains three item (cars names )

console.log(typeof (cars = ["BMW", "TESLA", "VOLVO"]));  // Result :Object

*Note :*

*1- typeof oparator need parentheses or round bracket the data is more that one word or it’s a string . (text with quotes symbol).*

*2- Array Indexes are Zero-based , which means that the first item is [0] , second is [1], and so on .*

**5- Object 🡪 Object  :**

JavaScript objects are written with curly braces { } ;

Object properties are written as **Key : Value .**

console.log(

  (Student = {

    firstname: "Bilal",

    lastname: "Elemrani",

    Rank: 1,

    age: 16,

    available: true,

  })

); // type object

In The example above The object (student) has 5 propereties .

**6- Undefind :**

In JavaScript , a variable without a value , has the value undefined.

The type is also undefined

let A;

let B = undefined;

console.log(typeof A);

console.log(typeof B);

// Result Undefind

Any variable can ve emptied , by setting the value to undefined , and sure the type also will be undefined .

let A = 5;

console.log(typeof A); //  5

A = undefined;

console.log(typeof A); // undefined

**Variables Introduction:**

Variables are named container , for storing data (storing data values) .

We use variable if i have a value that is used in all our system or it used many times , so I use variable and i put the value in it , and if a I want to change all , it enought to change variable value .

* **How to create a variable in js ?**

Creating a variable in javascript is called « declaring » a variable ,

You declare a js variable with **Var** keyword :

var user;

After the declaration the variable has no value (technically it is **undefined**) , to **assign** a value to the variable , use equal sign

In Javascript and in many of programming languages , the equal sign (=) is an ***‘’assingnment* operator’’** and not an ‘’equal operator ‘’

So this is different from algebra . The following does not make sense in algebra :

x = x + 5;

in js that has a sense 🡺 it assings x+5 to x , (the value of x incremented by 5 ) .

Note :

The « equal to operator is written like == in javascript

syntax :

keyword | variable nama | assignment operator | variable value .

Note :

*It’s not neccesary to use keyword* ***to declare*** *, but for now it good to use it .*

* **Multiple variables in The same Line**

Start the declaration by keyword , the separate the variables by **comma** :

var name = "Bilal",

    age = 17,

    rank = 5,

    country = "morocoo";

* **Id & global variable :**

When you create an element in Html , and you give it an Id , By example (carName) , the id become a global variable , so you can access to it on any other file .

 <div id="carName">Volvo</div>      <!-- Html -->

console.log(carName);             // in Javascript

so here we will print

<div id=’’carName ‘’> Volvo</div> 🡺 in console .

And sure we can modify the text

carName.innerHTML = "BMW";

**Identifiers Name Conventions And Rules :**

Identifier is a variable name , and this name must respect some rules :

1. Identifier mustn’t have a space .
2. Identifier mustn’t start by a number
3. Identifier can start by $ or \_ and contain them
4. Identifier mustn’t contian special caracter {@ # ¡ ! ? , . …}
5. Identifier mustn’t be a reserved name , as in windows you can’t name any file by {con – aux – nul …} , in js variables also has reserved names , this reserved names are language keywords themself {function , var , if , else , let , const} and other words {enum ,delete ..}, don’t worry if you some of this word you code editor will alerts you .
6. We ue camel case in variables names , by example the name contain two words , the first word all small and first one of the second word is capital , ‘carName ‘

**Var , Let , Const compare :**

4 Ways To Declare a Javascript Variable :

* Using var
* Using let
* Using const
* Using nothing

**When to Use JavaScript var ?**

Always Declare JS variable with var , let or const

The var Keyword is used in all Javascript code from 1995 to 2015

The let and const keywords were added to JS in 2015

/\*

   Var

  - Redeclare(Yes)

  - Access Before Declare (Undefined)

  - Variable Scope Drama [added to Windows object]

   Let

  - Redeclare(No --> Error)

  - Access Before Declare (Error)

  - Variable Scope Drama [no]

   Const

  - Redeclare(NO --> Error)

  - Access Before Declare ( Error )

  - Variable Scope Drama [no]

\*/

Variable scop drama is a big quatity of variables in window object that make many problems ,we will see them after .

**String Syntax & chars Escape Sequences :**

For do this experment you have to turn of « format on save «

As we know string can writted in single or double quotes , and problem happen we you try to use a symbol that is espacialy for programming languages like double or single quotes .

*Example :* we want to hightlight ‘web’ word .

console.log("Elezro "web" school");    // Here i will get an error

and for avoid this problem , we have to write either single quotes in double quotes or double quotes in single quotes .

**Let try this :**

 console.log('Elezro "Web" School');      // Double inside single

 console.log("Elezro 'Web' School");      // single inside double

Hey just imagine with me Know , that you want to write double inside double or single inside single .

**Here we will use Escape Operator .**

Escape oprator tell the os , that the character after it has no relation with programming and it should behave with it as a text

**Let try this :**

 console.log("Elezero \"Web\" School");

 console.log('Elezero \'Web\' School');

But i Know that you are a special human , and know you think how to write \ it self , it is you have to use two backslash :

console.log("Learn Javacript Arabic \\ With ossama Elzero");

so you have to make onther backslash with each one .

and imagine too, that you will write message in diffrent lines in our editor .

 console.log("Hello

   from

   testing");   // a big Error

and form do that we have to use Escape operator , as we know when we want a new line we press enter an escape operator escape this pressing on enter key .

**Let try this :**

console.log("Hello \

   from \

   testing");    // now it’s good

and know we will learn how to print each part of the message in a line .

Here we will use ‘\n ‘ (this in all programming language not in js only )

**Let try this :**

console.log("Elzero \nWeb \nSchool ");

**Concatenation :**

All languages give to devs an operator that help them to link data ;

**Let try this :**

let a = "We love";

let b = "Javascript";

document.write(a + b);

so here we will get **‘we lovejavascript’** , the solution is making a space after love or adding another variable that contain space .

let a = "We love";

let b = "Javascript";

let c = " ";

document.write(a + c + b);

or simply we can do :

let a = "We love";

let b = "Javascript";

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

Note :

*If want to print in console multiple messages :*

let a = "We love";

let b = "Javascript";

console.log(a, b); // by default the os make a space between a and b

**Template Literals (Template Strings)**

Synonyms:

* Template Literals
* Template Strings
* String Templates
* Back-Tics Syntax

As we said in the last episode that there are many features for concatenation is Es6 , now we will descover all this , and we will do a small comparision between ES6 and legacy code . !!

let a = "We Love";

let b = "JavaScript";

let c = "And";

let d = "Programming";

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

the example above show the old method , and here we used new line operator , and know we use new method Es6 .

1- Template Literals use Back-ticks (``) rather than single quote and double quote , to define a string :

let text = `Hello World !`;

With Template Literals, you can use both single and double quotes inside a string :

console.log(`He's often called "Johny"`);

*Template Literals allow multiline strings :*

console.log(`

      The Fast

      Leon

      Eats

      Gazel `);

And know we will descover the new method for using variables :

As an example we will see the preveous problems :

let a = `We Love`;

let b = `JavaScript`;

let c = `And`;

let d = `Programming`;

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

Here we don’t have to do ‘’ ‘’ for make space , we make space direcly and if we want to pring each word in a line .

console.log(`${a}

${b}

${c}

${d}`);

Is quirly braces we can write any js exprassion or an operation , in the next example we will print in the console ‘we love javascript and programming for every day 100% ‘ :

console.log(`${a} ${b} ${c} ${d} For Evrey day ${25 \* 2 + 50}%`);

know we will do a small comparision betwenn Es6 and legacy , imagine with me that we have to make a card in js then put this card in the page :

**1- by Es**

let markUp = `

 <div class="card">

 <div class "child">

 <h2>Title</h2>

 <p> Paragraph </p>

 </div>

 </div>

 `;

document.write(markUp);

**2- legacy code : (you will get many problems and you will use many escape oprator and new line operator so simply with Es is more confortable .**

var Title = "Elzero";

var disc = "Elzero Web Shcool";

var markUp = "\n <div class=\"card\"> \n <div class \"child\"> \n <h2>".concat(Title, "</h2> \n <p> ").concat(disc, " </p>\n </div> \n </div>\n ");

document.write(markUp);

**Here is the same code using Es :**

let Title = `Elzero`;

let disc = `Elzero Web Shcool`;

let markUp = `

 <div class="card">

  <div class "child">

  <h2>${Title}</h2>

  <p> ${disc} </p>

  </div>

 </div>

 `;

document.write(markUp);

**Assignment Notes :**

1 – As we know for print an element :

  <p id="test">Hello From Html</p>       <!-- IN HTML -->

console.log(document.querySelector("p"));     // in Js

The result in the conosle : <p id="test">Hello From Html</p >

But imagine with me that you want to print the content «  hello from html « :

console.log(document.querySelector("p").innerHTML);

or we have an easy method , using id :

document.write(test.innerHTML); // in Js

as we know id consider as a gloabal scope variable , so i have access to use it where i want .

2- When adding a number to a string JS will treat the numbe a string :

let x = 16 + " Volvo ";

so know js treat 16 as a string ‘’16 ‘’ 🡺 **16 Volvo**

let y = 16 + 4 + " volvo ";

as we know js evalute code from left to right so 16 + 4 will be treated as number then the total with ‘’volvo ‘’ will be treated as string sure :

result 🡺 **20 volvo**

let a = "volvo " + 16 + 4;

left to right , so in the begging a take string type : so restult : **volvo 164**

3- Repeat() : returns a string with a number of copies

**Arithmetic Operators**

Arithemetic Opereotr :

1-The **addition** Operator **(+)** adds numbers :

console.log(10 + 20);

and also + operator can also used as **concatenate operator**

console.log(100 + "bilal");

2- The **Substraction** Operator (-) for subsract numbers :

console.log(100 - 20);

But here there no + operator to consider it as a concatenate operator , so Here the os sure that inputs will be numbers , so the result here is *NaN 🡺 Not a Number* and the weird thing Here That **NaN has number type .**

3- The **multiplication** operator (\*) multipliesf numbers :

console.log(100 \* 20);

4- The Division operator (/) for divise numbers :

console.log(100 / 20);

5- The **Modulus (Division Remainder / rest )** (%) for rest of the division :

console.log(37 % 20);

The Result is : 17 , wich mean 37 isn’t divisable by 20 , and for become divisable , we have to subsruct 17 from 37 so we will get 20 and the rest of divise 20 by 20 is 0 sure !!

6- The **exponentiation** operator (\*\*) , first operand is the base and the second on is the power : 3\*2 = 3²

console.log(2 \*\* 3);  // 2\*2\*2 = 8

And this is the same result using Math.pow(x ,y) function :

console.log(Math.pow(2, 3)); // 2\*2\*2 = 87 -

7**- ++Increment [Post / Pree ] :**

Post increment : first print the number then increase the number by 1

Pree increment : first increase the number by 1 then print the number so here the printed the nubmer is increased on , and this is the opposit of Pree wich print the old number before increasing :

let a = 1;

a++;      // post increament

let b= 1;

++b;      // Pre increament

**8- decrement [post / pree]**

The same increment’s speech !!!

**Unary Plus and Negation Operators**

* + Unary Plus [Return Number If its not Number ]
* - Unary Nagation [Return Number If it not number + Negates It ]

First we will start with unary plus .

console.log(100);

console.log(+"100"); // return 100 as a number not string

For make sure that ‘’100’’ become a number we will use **typeof** operator

console.log(typeof +"-100"); // number

console.log(+"Bilal");

now we tell the os to convert bilal wich is a text to a number sure the computer will reply by ‘’NaN’’ 🡺 Not a number

console.log(+"0x11"); 🡪 17 as a number

unary + can also convert hexa to a number

console.log(+"15.5"); 🡪 15.5 as a number

and sure we can do some experiments :

console.log(+false); 🡪 0

console.log(+true); 🡪 1

console.log(+null); 🡪 0

and now we will do some experiment by unary negation :

console.log(-100); 🡪

console.log(-"-100"); 🡪 negative and negative = positive = 100

as we know unary nagation convert text to a numbe and negate it

console.log(-"Bilal"); // not a number

and at the end :

console.log(-"0x11"); 🡪 -17

console.log(-false); 🡪 -0

console.log(-true); 🡪 -1

console.log(-null); 🡪 -0

we have another method to convert string to a number , Number constructor :

let myAge = "15";

let numAge = **Number**(myAge);

console.log(typeof myAge);

console.log(typeof numAge);

The **Number()** method converts a value to a number

If the value cannot be converted , NaN is returned , as we have seen in unaray opratores .

**Type Coercion(Type Casting)**

Type coercion ( التحويل بالاكراه ) is converiting data types proccess during code running .

let a = "10";

let b = 20;

let c = true;

console.log(a + b); 🡪 1020

how to slove this problem if i want really to add them as number , so here we will use unary operator .

console.log(+a + b); 🡪 30

Here we will see how type coercion will behave if we use ‘– ‘or make substruction .

console.log(a - b); 🡪 -10

type coercion convert « a » to a number , because when we use minus operator it necessay to be an arithmetic operation between numbers so here type coericion convert « a » automatically from string to a number , but if « a » doesn’t accept converting so here the result will be NaN .

console.log("" - 2); 🡪 -2

because if we convert ‘’ ‘’ to a number we will get 0

console.log(0 – 2 ) sure will equal -2

console.log(false - true); 🡪 0 – 1 = -1

console.log(b + c); 🡪 20 + 1 = 21

pay attention here :

console.log(a + b + c); 🡪 1020true

true is a boolean value , here type coercion convert it to a string

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

**Assignment Operators :**

let a = 10;

but now i want to add 10 to « a «

a = 20 ;

but me i don’t want this , i want to add 10 to original variable

a = a + 20;

and the assignment operator make it easy for us , instead of a = a + 20 ,

a = a + 20; // a = a + 20

and this method is valaible for all all arithmetic operator :

let a = 10;

a += 20; 30

a \*= 3; 30\* 3 = 90

a += 10; 90 + 10 = 100

a /= 2; 100 / 2 = 50

a %= 3; The rest (50/ 3 ) mutiply \* 3 (when we substruct

the rest getted here we will get 0 like a rest

console.log(a);

**Assignment Note :**

console.log(b++ + a + ++b + a--);

console.log(6 + 5 + 8 + 5);

// a will become : 4

// b will become : 8

The value returned is the value that should be printed

**Number**

**1. Syntactic Sugar**

Somethings that make your code beautfull and understandable :

console.log(1000000);

let use Syntactic sugar :

console.log(1\_000\_000);

or you can use onther syntactic sugar :

console.log(1e6); 🡪 1000000

or we can use this :

console.log(10 \*\* 6); 10 power 6 / which mean 6 zeroes

*floting number : number that contains digits*

console.log(10810.0);

if the digits are zeroes so the formatter will remove them automatically , because they don’t have any value , and if you have one zero or many zeroes the formatter will save one zero only and the console won’t print any zero

Hey , we have used Number in converting from string to number

And this number has many other uses , type Number. For see all properties .

Among this property :

console.log(Number.MIN\_SAFE\_INTEGER); 🡪 9007199254740991

The Number.MAX\_SAFE\_INTEGER\*\* constant represents the maximum safe integer in JavaScript

For larger Integers , use *BigIng .*

Number.MAX\_VALUE 🡪 returns the largest number possible in Javascript .

Number.MAX\_VALUE 🡪 has the value of 1.7976931348623157e+308

Based on this , there are : ***Number.MIN\_SAFE\_INTEGER*** and

**Number.MIN\_VALUE**

Number.MIN\_VALUE : returns the smallest number possible in javascript ;

Number.MIN\_VALUE has a value of 5e-324

The MIN\_VALUE in positive not negative

MIN\_VALUE is the value closet to 0 .

Numbers smaller than this converted to 0

The most negative number is the negative MAX\_Value

The smallest negative number is the negative of MIN\_VALUE

**Number Methods**

1- how to call a method

We will learn how to call a method with to string method

console.log(100..toString());

The toString() method returns a number as a string.

We appended two dots after the number for telling os that there is no digits , and if the number contain digits

console.log(100.25.toString());

the first dot for digit and the second one for method , the cause behind doing two dots because there is no digit after dot

console.log(100.0.toString());

but prettier , makes it like this  :

console.log((100).toString());

**toFixed :** returns a string, with the number written with a specified number of decimals:

let x = 9.6;

console.log(x.toFixed()); // 10 (because 6 >= 5 )

let x = 3.35;

console.log(x.toFixed(1)); // 3.4 (because 5 >= 5 )

if you want 2 dicimals so the computer go to check the third dicimal if it’s smaller that 5 , so the secod dicimal will still as it’s ,but if the third decimal = or greater that 5 so the computer will add 1 to the second digit , i hope that you understand .

**toPrecision() :**  returns a string, with a number written with a specified length:

let x = 30.35;

console.log(x.toPrecision(5)); // 30.350

Here we specified the length to 5 , which mean it should be 5 numbers , we start from left to right , and sure it can’t clip integers side , it can clip dicimals side , and we can add zeros if we need them for complete the number requested , and sure this method work with toFixed logic , so it can add one to a dicimal when , it remove one .

**Converting Variables to Numbers**

There are 3 JavaScript methods that can be used to convert variables to numbers :

* The Number() method
* The parseInt() method
* The parseFloat() method

**parseInt()** parses a string and returns a whole number. Spaces are allowed. Only the first number is returned .

console.log(parseInt("  10 22 bilal ")); 🡪 10

Note , the text can start by spaces but if the number is after somthig like words or characters , NaN will be returned value.

console.log(parseInt("age:  10  ")); 🡪 NaN

Make in mind : That parseInt return integar so no dicimals

console.log(parseInt(" 120.5$ 35 ")); 🡪 120

**ParseFloat() 🡪** parses a string and returns a number, if the number has decimals , pareFloat() can return the number with its decimals .

console.log(parseFloat(" 120.5$  ")); 🡪 120.5

The **Number.isInteger()** method returns true if a value is an integer of the datatype number .

Otherwese it returns false .

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

console.log(Number.isInteger(100.25));           // false

console.log(Number.isInteger(100.0));            // true

The **Number.isNan()** method returns true if The value is Nan and return false if the value returned is other value .

Make in mind : that Not a number we don’t mean by it datatype , but we mean value returned .

console.log(Number.isNaN("ossama")); 🡪 false

for return Nan , we will try to divide ossam / 100

console.log(Number.isNaN("ossama" / 10)); 🡪 True

**Math Object**

First for enter to any math Method : syntax

**The syntax for Math any methods is : Math.*method*(*number*)**

1- **Round**:

Math.round(x) : Returns X rounded to it nearset integer

console.log(Math.round(12.5));    // 13

console.log(Math.round(12.4));    // 12

2- **ceil**: (سقف )

Math.ceil(x) : Returns X rounded up to it nearset integer

console.log(Math.ceil(12.5)); // 13

console.log(Math.ceil(12.4)); // 13

3- **floor**:

Math.floor(x) : Returns X rounded down to it nearset integer

console.log(Math.floor(12.5)); // 12

console.log(Math.floor(12.4)); // 12

4- **trunc**: {new in Es6 }

Math.trunc(x) : Returns The integer part of x

console.log(Math.floor(12.9)); // 12

console.log(Math.floor(12.4)); // 12

4- **Sign**: {new in Es6 }

Math.sign(x) : Returns -1 if x is negative , or it returns 1 if x is positive , or returns 0 if x = 0 (no positive no negative)

console.log(Math.sign(-4)); // -1

console.log(Math.sign(0)); // 0

console.log(Math.sign(4)); // 1

5- **Math.min() and Math.max()**

*Math.min()* and *Math.max()* can be used to find the lowest or hightest value in a list of arguments :

console.log(Math.max(100, 25, -65, -82, 547, 1245, 0, 5, 588));

// 1245

console.log(Math.min(100, 25, -65, -82, 547, 1245, 0, 5, 588));

// -82

6-***Math.random() :***

*Math.random()* returns a random number between 0 (inclusive), and 1 (exclusive) . (output from 0 to < 1 )

console.log(Math.random());

7-***Math.pow(x,y) :***

*Math.pow(x,y)* returns the value of x to the power of y :

console.log(Math.pow(7, 2)); // 7²

console.log(Math.pow(4, 5));  // 4 os 5 ==> 1024

8-***Math.sqrt(x) :***

*Math.sqrt(x)* returns the square root of x :

console.log(Math.sqrt(49)); // recine carre de 49 est 7

console.log(Math.sqrt(16)); // recine carre de 16 est 4

*Assignments Note*

1- For print the number of the digits in a number :

Syntax : `${num`}.length

console.log(Number.MAX\_SAFE\_INTEGER.toFixed(0).length); // 16

console.log(`${Number.MAX\_SAFE\_INTEGER}`.length); // 16

adding to these 2 methods , third method , is converting text to string then use length , (best one is the second)

2- how to exctract a random value from a list of numbers

Return a random number between 0 and 100 :

console.log(Math.random() \* 100);

and know we will make it returns a number between 50 and 55 :

console.log(Math.round(Math.random() \* 5) + 50);

**String Methods**

String methods help you to work with strings .

**String Methods and Properties .**

Primitive values, like ‘hello world’ or any other text , cannot have properties or methods (because they are not object)

But with the javascript, methods and properties are also available to primitive values, because js treats primitve values as objects when excuting methods and properties .

*Extracting String Characters*

There are 2 methods for extracting string characters :

* charAt(*position*)
* Access with Index

The Charat() method returns the character at a specified index(position) in a sring :

let theName = "Bilal";

console.log(theName.charAt(3)); **a**🡪(Because Counting starts from 0)

Remember c language .

Index out of range (cross the length of the string at all), *emty string* is returned in the case :

console.log(theName.charAt(5)); // nothing , return emty string (the last index is 4)

Note : Invalid index converts to 0 (4.5 = 4 , 2.9 = 2, 1.1=1)

console.log(theName.charAt(3.9)); // invalid index 3.9 = 3

removing the dicimals part)

Access with index :

let theList = [1, 2, 3, 4, 5];

index is the box that contain a value in squence .

each value value are in an index , the first index is 0

for access the index :

console.log(theList[3]); // 4

as we see in c programming string is an array of characters, so string is also an array :

console.log(theName[3]); // a

NOTE :

If the index is out of range , [ ]returns undefined, while chartAt() retunrs an empty string .

If the index is invalid , [] returns undefinded while charAt() try to fix it .

console.log(theName[1.5]); // undefined

index out of range :

console.log(theName[5]); // undefined

**2- length**

The length property returns the length of a string :

console.log(theName.length); // 5

**3- trim()**

Imagine with me that we add 2 spaces before and after the text , and we will check length , if the withspaces increse or no .

let theName = "  Bilal  ";

console.log(theName.length); // 9

so witespaces are character and it has a value and it cost us memory, so for remove these witespaces we use trim() .

The trim() method removes whitespace from both sides of a string:

console.log(theName);

console.log(theName.length); // 9

console.log(theName.trim());

console.log(theName.trim().length); // 5

***4- toUpperCase(), toLowerCase()***

A string is converted to upper case with toUpperCase():

A string is converted to lower case with toLowerCase():

console.log(theName); // the original one : Bilal

console.log(theName.toLowerCase()); // lowercase : bilal

console.log(theName.toUpperCase()); // uppercase : BILAL

***4- Chain Methods***

Chain methods is making a chain contains all this mehtods .

Imagine with me that i told to remove left and rigtht whitespaces from a text and return to me a special character uppercase .

So here you should use chain methods :

console.log(theName.trim().charAt(2).toUpperCase()); // L

***5- indexOf()***

The indexOf() method returns the index of (the position of) the first character in the spcefied text in the string

let theTest = "i hope to learn javascript and learn english ";

console.log(theTest.indexOf("learn")); // 10

it gave us the index of the first character ‘l’ .

and sure it gave us the position of the first « learn »

***6-lastIndexOf()***

the lastIndexOf() method searches backwards (from the end to the beginning) , meaning : if the second parameter is 15 , the search starts at postion 15 , and searches to the beginning of the stirng .

**NOTE :**

Both indexOf(), and lastIndexOf() return -1 if the text is not found:

let theTest = "i hope to learn javascript and learn english ";

console.log(theTest.indexOf("learn")); // 10

console.log(theTest.lastIndexOf("learn")); // 31

**indexOf()** starts the search from the beggining of the text, this is why it return the position of the first « learn »

**lastIndexOf()** start the search from the end of the text and this is why it return the postion of the last « learn »

let theTest = "i hope to learn javascript and learn english ";

console.log(theTest.lastIndexOf("learn", 30)); // 10

Here it start from the start point to the begging of the string .

Both indexOf(), and lastIndexOf() return -1 if the text is not found:

console.log(theTest.indexOf("bilal")); // -1

console.log(theTest.lastIndexOf("bilal")); // -1

Both methods accept a second parameter **as the starting position for the search**:

By example i search for one word, and i find, but i need to check if there is another one so here we use the second parameter for tell js to start search after the founded word .

console.log(theTest.indexOf("learn")); // 10 (knowing learn take 5 index so we will start seaching after these 5 indexes)

console.log(theTest.indexOf("learn", 16)); // 31

second parameter is ***optional*** , 0 is the default value for indexOf(), and length - 1 is default value for lastIndexOf()

***7-Slice()***

Extracting String Parts

There are 3 methods for extracting a part of a string .

* slice(*start*, *end*)
* substring(*start*, *end*)
* substr(*start*, *length*)

Slice() extracts a part of a string and returns the extracted part in a new string .

The method takes 2 parameters: *the start position*, and *the end position (****end not included and it’s opitional****)*.

Slice out a portion of a string from position 15 to position 20

(20 not included) .

let theTest = "I will make my dream come true";

console.log(theTest.slice(15, 20)); // dream

if the second parmeter is not specified , default is ‘length ‘ wich mean it will continue to the end of a string .

console.log(theTest.slice(12)); // my dream come true

so if you omit (حدفت ) the second parameter, the method will slice out the rest of the string .

if a parameter is negative, the position is counted from the end of the string .

The example slices out a portion of the stirng text from position -15 to position -10

console.log(theTest.slice(-15, -10)) ; -10 not included

and this without count from the end .

console.log(theTest.slice(-18)); //my dream come true

***8-repeat()***

The repeat() method returns a string with a number of copies of a string.

console.log(theTest.slice(15, 21).repeat(4)); // dream dream dream dream

***9-Split()***

**Converting a String To an Array .**

a String can be converted to an array with the split() method .

let theTest = "My Name Is Bilal";

console.log(theTest.split(" ")); // split on spaces

result 🡪 [‘My’, ‘Name’, ‘Is’, ‘Bilal’ ]

1. **0**: "My"
2. **1**: "Name"
3. **2**: "Is"
4. **3**: "Bilal"

If the separator is omitted, the returned array will contain the whole string in index [0]

If the separator is "", the returned array will be an array of single characters

Spilit() take two parameters so the second one is the limit of cutting, it’s an opetional parameter , default one is cutting alll :

let theTest = "My Name Is Bilal";

console.log(theTest.slice(3).split("", 4));

['N', 'a', 'm', 'e'] 🡪 i set the limit of cutting so i will have 4 character .

***10-Substring()***

substring() is similar to slice().

The difference is that start and end values less than 0 are treated as 0 in substring().

let theTest = "My Name Is Bilal";

console.log(theTest.substring(0, 10));

Contrary to this , slice() nagative values start counting from the end of the string.

If you omit the second parameter, substring() will slice out the rest of the string.

console.log(theTest.slice(-5));         // Bilal

console.log(theTest.substring(-5));     // My Name Is Bilal

so as you see -5 is treated as 0 .

Substring() has another property, if you make a mistake, and put the starting point insead of end point and vice versa ,

*Slice()* will return nothing ‘emty’

*Substring()* will detect the probelm and the swap the values .

console.log(theTest.slice(7, 3));    //

console.log(theTest.substring(7, 3)); // Name

if i want to arrive at the last caracter :

console.log(theTest.substring(theTest.length - 1));

so if i want count from the end of the string, you can use these method .

console.log(theTest.substring(theTest.length - 1 - 12, theTest.length - 1 - 5)); // Name is

***11-Substr()***

substr() is similar to slice().

The difference is that the second parameter specifies the **length** of the extracted part.

console.log(theTest.substr(3, 7)); // Name is

**If you omit the second parameter, substr() will slice out the rest of the string.**

**If the first parameter is negative, the position counts from the end of the string.**

***12-includes()***

The includes() method returns true if a string contains a specified string.

Otherwise it returns false.

The includes() method is case sensitive.

Syntax

String.includes(searchvalue,start)

*SearchValue* : required

*Start*: optional , the position to start searching from , default 0

console.log(theTest.includes("Bi")); // true

let’s specify the start points :

let theTest = "NO fap from today ";

console.log(theTest.includes("fap", 5)); // false

***13-startWith()***

The startsWith() method returns true if a string starts with a specified string.

Otherwise it returns false.

The startsWith() method is case sensitive.

let theTest = "NO fap from today ";

console.log(theTest.startsWith("NO")); // true

we can specify the starting searching point .

console.log(theTest.startsWith("f", 7)); // true

***14-endWith()***

The endsWith() method returns true if a string ends with a specified string.

Otherwise it returns false.

The endsWith() method is case sensitive.

Syntax : *string*.endsWith(searchvalue, length)

let theTest = "NO fap from today";

console.log(theTest.endsWith("y")); // true

and let’s specify the second parameter :

console.log(theTest.endsWith("p", 6)); // true

NOTE : the second parameter in the length crossed no indexes

***15-replace ()***

**The replace() method searches a string for a value or a regular expression.**

**The replace() method returns a new string with the value(s) replaced.**

**The replace() method does not change the original string.**

Syntax : *string*.replace(*searchValue, newValue*)

All parameters are required

let theTest = "I'm Bilal";

console.log(theTest.replace("Bilal", "Ahmad")); // I'm Ahmad

Note :

 you replace a value, only the first instance will be replaced. To replace all instances

let theTest = "I'm Bilal, and my favorite name is Bilal";

console.log(theTest.replace("Bilal", "Ahmad"));

// I'm Ahmad, and my favorite name is Bilal

See this :

console.log(theTest.replace(/Bilal/g, "Ahmad"));

// I'm Ahmad, and my favorite name is Ahmad

But see this :

let theTest = "I'm Bilal, and my favorite name is bilal";

console.log(theTest.replace(/Bilal/g, "Ahmad"));

// I'm Ahmad, and my favorite name is bilal

That happen because this method is case-sensitive , for make it case-insensitive :

console.log(theTest.replace(/Bilal/i, "Ahmad"));

and for replace all instaces also :

console.log(theTest.replace(/Bilal/gi, "Ahmad"));

// I'm Ahmad, and my favorite name is Ahmad

JavaScript Comparison and Logical Operators

Comparison and Logical are used to test for True or False

**1-Comparison Operators**

Comparison Operators are used in logical statements to determine equality or difference between variable or values .

Given That Y=8 .

* Equal to == Y == 8 true

console.log(y == 10); // false

console.log(y == "8"); // true

- Equal value & equal type === y === "8" false

console.log(y === 10); // false

console.log(y === "8"); // false

console.log(y === 8); // true

- Not equal != Y != 8 false

console.log(y != 10); // true

console.log(y != "8"); // false

console.log(y != 8); // false

- Not equal value or not equal type  !== Y !== “8“ true

console.log(y !== 10); // true

console.log(y !== "8"); // true

console.log(y !== 8); // false

- less than <

console.log(y < 10); // true

console.log(y < "8"); // false

- greater than >

console.log(y > 7); // true

console.log(y > "8"); // false

console.log(y < "bi"); // false

- greater than or equal & less than or equal

console.log(y >= "8"); // true

console.log(y <= "8"); // true

comparison operators can be used in conditional statements to compare values and take action depending on the result :

let age = 17;

if (age >= 18) text = "you can drive cars ";

if (age < 18) text = "you cannot drive cars ";

console.log(text);

a small challenge in this episode is return true from this code without touching the values :

console.log("ossama" === "ahmad");

the solution is :

console.log(typeof "ossama" === typeof "ahmad"); // true

**2-Logical Operators**

Logical operators are used to determine the logic between variables or values.

Given that x = 6 and y = 3 .

**Not  !**

console.log(true); // True

console.log(!true); // not true ==> False

let do annother test with variables :

console.log(!(x == y)); // not false = true

so this logical operator return the opposit of the origin result .

**And &&**

For do many conditions we need to use this logical operator .

console.log(x > y && y < x); // true

console.log(x > y && y < x && x == "6" && y != "6"); // true

if one condition is false to the result will be false .

console.log(x > y && y < x && x == "6" && y != "3"); // flase

**or ||**

For do many conditions we need to use this logical operator .

But the deffrence here that one condition true is enought to make all the result **true** .

console.log(x == "3" || y == "3"); // true (because the second one is true)

**if conditions**

Conditional statements are used to perform different actions based on different conditions.

Conditional Statements :

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this .

In javascript we have the following conditional statements :

* Use if to specify a block of code to be executed, if a specified condition is true
* Use else to specify a block of code to be executed, if the same condition is false
* Use else if to specify a new condition to test, if the first condition is false

**1- the if Statement :**

Use the if statement to specify a block of javaScript code to be excuted if a condition is true .

Syntax :

if (condition) {

  // block of code to be excuted if the condition is true

}

Example :

if (age >= 18) {

  console.log("Search For Job");

} else console.log("Just focus on Stydying");

Note :

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

If (age >= 18) {

  console.log("Search For Job");

}

Here we will get error .

2- The else Statement :

Use the else statement to specify a block of code to be executed if the condition is false .

if (condition) {

        // block of code to be excuted if the condition is true

 }else{

        // block of code to be excuted if the condition is false

 }

Example :

let Price = "5000dh";

if (parseInt(`${Price}`) > 5000) {

  console.log("sorry i can afford buy it for you ");

} else console.log("Ok, inshallah i'll by it for you ");

2- The else if Statement :

Use the else if statement to specify a new condition if the first condition is false .

Syntax :

const time = new Date().getHours();

if (time < 10) {

  console.log(`Good Morning`);

} else if (time < 20) {

  console.log(`Good Day `);

} else console.log(`Good Evening`);

Example with all that we learned :

let price = 200;

let discount = false;

let discountAmount = 0.3;

let country = "Morrocco";

let clientSituation = "dev";

if (discount === true) {

  price -= price \* discountAmount;

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

  if (clientSituation == "student") {

    price -= price \* 0.2;

  } else {

    price -= price \* 0.1;

  }

}

console.log(`${price} Dh`);

**Conditinal (ternary) operator**

**JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.**

Syntax

*variablename*= (*condition*) ?*value1*:*value2*

// condition ? if true : if false

age = 17;

let voteAble = age < 18 ? "Too young" : "Old enough";

console.log(voteAble);

we will do a project using simple condition and we will do the same with ternary opearator :

// data

let theName = "Inas";

let theGender = "Female";

let theAge = "17";

// Fist method

if (theGender === "Male") {

  document.write(`Hello, Mr ${theName}. \nYour Age now is ${theAge}`);

} else if (theGender === "Female") {

  document.write(`Hello, Mrs ${theName}. \nYour Age now is ${theAge}`);

} else {

  document.write("Please,check your gender spelling");

}

// second method

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

console.log(`Hello ${gender} ${theName}, you age now is ${theAge}`);

in ternary operator we can also make many else case as if condition

//Fist method

if (theGender === "Male") {

  if (theAge <= 11) {

    document.write(`Hi ${theName}, you are in the childhood`);

  } else if (theAge > 11 && theAge <= 21) {

    document.write(`Hi ${theName}, you are in the Adolescense`);

  } else if (theAge > 21 && theAge <= 30) {

    document.write(`Hi Mr ${theName}, you are in the maturity`);

  } else if (theAge > 40) {

  } else if (theAge > 30 && theAge <= 40) {

    document.write(`Hi Mr ${theName}, you are in the youths`);

  } else if (theAge > 40) {

    document.write(`Hi Mr ${theName}, you are in the old age`);

  }

} else if (theGender === "Female") {

  if (theAge <= 11) {

    document.write(`Hi ${theName}, you are in the childhood`);

  } else if (theAge > 11 && theAge <= 21) {

    document.write(`Hi ${theName}, you are in the Adolescense`);

  } else if (theAge > 21 && theAge <= 30) {

    document.write(`Hi Mr ${theName}, you are in the maturity`);

  } else if (theAge > 30 && theAge <= 40) {

    document.write(`Hi Mrs ${theName}, you are in the yoths`);

  } else if (theAge > 40) {

    document.write(`Hi Mrs ${theName}, you are in the old age`);

  }

}

// second method

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

let lifeGender =

  theAge <= 10

    ? "childhood"

    : theAge > 10 && theAge <= 21

    ? "adolescense"

    : theAge > 21 && theAge <= 30

    ? "maturity"

    : theAge > 30 && theAge <= 40

    ? "youths"

    : "old age";

console.log(`Hello ${gender} ${theName}, you are now in ${lifeGender}`);

Nullish coalescing operator and logical or

Imagine with that you add a product in a website but you make a mistake in variable’s value .

So here you need, always to write an alternate value .

let price = null;

console.log(price || 20); // null + undefined or any other falsy value

console.log(price ?? 20); // null + undefined only

falsy values : is a value that is considerd false is boolean contexts

for test any value if it is or not , use this ,

console.log(Boolean(-1)); // true so it's not a falsy value

console.log(Boolean("")); // false so it's a falsy value

console.log(Boolean()); // false so it's a falsy value

console.log(Boolean(0)); // false so it's a falsy value

console.log(Boolean(undefined)); // false so it's a falsy value

console.log(Boolean(null)); // false so it's a falsy value

Switch

**The switch statement is used to perform different actions based on different conditions.**

**Syntax :**

switch(expression) {

  case x:

    // code block

    break;

  case y:

    // code block

    break;

  default:

    // code block

}

**When JavaScript reaches a break keyword, it breaks out of the switch block.**

**This will stop the execution inside the switch block.**

**It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.**

**Note: If you omit the break statement, the next case will be executed even if the evaluation does not match the case.**

**Because it’s that are the same cases and sure the variable preseve the value from the last modifing.**

**This is the true way by putting break after each case , for tell him that this case ends here .**

let day;

switch (new Date().getDay()) {

  case 0:

    day = "Sunday";

    break;

  case 1:

    day = "Monday";

    break;

  case 2:

    day = "Tuesday";

    break;

  case 3:

    day = "Wednesday";

    break;

  case 4:

    day = "Thursday";

    break;

  case 5:

    day = "Friday";

    break;

  case 6:

    day = "Saturday";

}

console.log(day);

**but i know i will not tell about that . !**

let day;

switch (new Date().getDay()) {

  case 0:

    day = "Sunday";

  case 1:

    day = "Monday";

  case 2:

    day = "Tuesday";

    break;

  case 3:

    day = "Wednesday";

  case 4:

    day = "Thursday";

  case 5:

    day = "Friday";

    break;

  case 6:

    day = "Saturday";

}

console.log(day);

**today is Wednesday but there no break to tell , about the limit of this case , so the code will be excuted even if case doesn’t match the value , because it think that they are all this cases share the same , becaus there is no breaks .**

**The default keyword specifies the code to run if there is no case match:**

let flowerColor = "white";

let availability;

switch (flowerColor) {

  case "red":

  case "white":

  case "yellow":

  case "purple":

  case "pink":

  case "blue":

  case "green":

    availability = `Available`;

    break;

  default:

    availability = `Sorry, it's not available`;

}

if (availability == `Available`) {

  console.log(`%c${availability}`, "color : green;font-size:18px");

} else {

  console.log(`%c${availability}`, "color :red;font-size:18px");

}

**If default is not the last case in the switch block, remember to end the default case with a break.**

switch (flowerColor) {

  default:

    availability = `Sorry, it's not available`;

    break;

  case "red":

  case "white":

  case "yellow":

  case "purple":

  case "pink":

  case "blue":

  case "green":

    availability = `Available`;

}

**And sure we can remove the break from the last cas because the block breaks there anyway .**

// Game State Variables

let playerHealth = 100;

let playerScore = 0;

let currentLevel = 1;

function handlePlayerAction(action) {

  switch (action) {

    case "attack":

      playerScore += 10;

      console.log("You are attacking the enemy !!");

      break;

    case "attacked":

      playerScore -= 10;

      console.log("You are attacked by the enemy !!");

      break;

    case "heal":

      playerHealth += 20;

      console.log("You helped yourself");

      break;

    case "level Up":

      ++currentLevel;

      console.log(

        "Congratulations !! You passed to the next level \nlevel: " +

          currentLevel

      );

      break;

  }

}

handlePlayerAction("heal"); // You helped yourself

handlePlayerAction("attack"); // You are attacking the enemy !!

handlePlayerAction(

  "level Up"

); /\*Congratulations !! You passed to the next level

level: 2 \*/

handlePlayerAction("level Up");

/\*Congratulations !! You passed to the next level

level: 3\*/