JavaScript Programmation

Out put to screen

```
>window.alert('bonjour')
>document.write('bonjour')
>console.log('bonjour')

// console styling
>console.error('Error')
>console.table(['one','two'])
// Directive %c
>console.log('Hello from %c js file','color:red';-)
>defer // indiquer au navigateur que le fichier js ne doit etre executer d'une fois que le code HTML afin d'etre analyser , utuliser seulement avec scr
>Async // script completely independant

// Data type
>console.log(typeof(500);
// string ,Number ,object,boolean,undefined , null(object)
```

Variables

```
// declaration variable
> var user='sahbi'
>document.write(user)

// declaration multi
> var user='sahbi' , age = 30 ;
// chaque htmlID est un variable
```

Concatenation

```
> let a = 'one'
> let b = 'two'
> document.write(a,b)
> document.write(a+b)
```

Literals Template string

```
> ` ` // backticks
> let a = 'Hello'
> let b = ' world'
> console.log(`${a}`)
```

Operators Aritmetique

```
> / // division
> ** // puissance
> % // modulo
> ++ // increment [post/pre]
> -- // decrement [post/pre]
// post :
> num =1; num + + // afficher valeur et incrementer
// pre :
> num =1; + + num // incrementer puis afficher
```

Unarry Plus and Negation Operators

```
> console.log(+'100'); // 100 return number
> console.log(-'100'); // -100 return negation number
> console.log(number('100')); // convertir nombre
```

Type coercion / type casting

```
> let a='10', b=10
> console.log(a+b); // retourner concatenation
> console.log(+a+b); // retourner number
> console.log(a-b); // retourner number
```

Put function

```
// OnClick event
<input type='button' value='calcul' onClick='calculer()'>
// pseudo Protocole
<a href ='javascript:void(calculer())'>lien</a
```

Comparaison Operators

```
== :equal != : not equal === identicals !=== not ident
> :larger > : smaller >= larger equal <= smaller equal
!= : not     && : and       || : Or
```

If condition

```
>if (condition){
      Instruction ;
}else if {
    Instruction;
  }else {  instruction ; }

// Tenarry Operator
> condition ? If true : if false
```

Nullish coalescing Operator

```
>console.log('the price ${price || 200'};
// return 200 si price = null or undefined
```

Switch statement

```
>switch(expression){
   Case 1 :
         Code ; break ;
    Case 2 :
         Code ; break ;
     Defaults :
          Code ;
}
// if not match : default
//if multi match :  case 2 :
                    Case 3 :
                         Code ; break;
```

Loop For

```
>for (let i=0 ; i<10 ; i++){
    Instruction ;
 }

>for (let fruit of fruitsList ){
    Instruction ;
 }

// label : identifier loop
>mainloop : for (let i=0 ; i<10 ; i++){
                    Instruction ;
              }
```

While Loop

```
>while(condition){
      Instruction ;
   }
```

Do While Loop

```
> do {
      Instruction ;
   } while (condition);
```

Function

```
>function sayHello(){ instruction ;}
>function sayHello(){
                        instruction ;
                        Return
                      };
>function sayHello(user , age){
                  If (age===undefined){
                       age = 'unknow';
                   }
          };
> age = age || 'unknow';

// default value : unknow is default value pour age

> function sayHello(username , age='unknow'){
                Instruction
                  };
```

Rest Parameter

```
// si je ne connais pas nombre arguments
// doit etre unique ou le derniers arguments
// transforme les parameters on  array

>function calculer(… numbers){
                For (let i=0,i<numbers.length;i++){

                      }
                Return ` resultat finale ${resultat}`
                  };
```

Anonymous function

```
// definir function dans un variable

> let calculer = function calculer(num1,num2){
                Return  num1+num2;
                  };
```

Arrow function

```
// one line function

> let print  = function (){
                Return  10 ;
                  };

> let print  = ( ) => 10 ;

> let print  = _  => 10 ; // sans arguments
```

Higher Order function

```
// is a function that accepts functions a parameters and or
return a function

// Map : array methodes

// Filters  : array  method with all elements that pass the
test implemented by the provided function

// Reduce : array methodes , execute a provided function
once for each array element
```

————-

Numbers

```
// tous les nombres sont double précision

// syntaxe sugar
>console.log(1000000)
>console.log(1_000_000)
>console.log(1e6)

>console.log((100).tostring());
>console.log(100..tostring());// convertir to string

>num.toFixed(2) //fixer chiff apres virgule,retourn string
>num=parseInt(numb) //analyser pour trouver entier
>num=parsefloat(numb) //analyser pour trouver reel

>num.isInteger() //verifier si entier
>num.isNaN() //is not a number
```

Math object

```
>num=Math.round(x)// arrondir x
>num=Math.Ceil(x)// arrondir  up
>num=Math.floor(x)// arrondir  down
>num=Math.min(x)// minimum
>num=Math.max(x)// maximum
>num=Math.pow(x)// puissance
>num=Math.random()// random number 0 – 1
>num=Math.trunc(x)// integer part
```

String object

```
>name[1] = name.charArt(1)
>n.charCodeAt(0)// retourner code assci
>name.length()// retourner longueur de la chaine
>name.trim()// eleminer les espaces
>name.toUpperCase()// rendre maj
>name.toLowerCase()// rendre minus
>name.indexOf('sa')// index substring
>name.ilastIndexOf('sa')// le dernier index substring
>name.slice(0,3)// slice d'une chaine
>name.split(' , ')// split string return array
>name.substring(2,5)// return substring
>name.substr(2,10)// return substring debut , nmbr char

>name.includes('sah',2,5)// verifier si substring existe
>name.startWith('string',2,5)//  true or false
>name.endWith('string',2,5)//  true or false
```

Array object

```
>let arrays =['one','two']// create array
>arrays [1]// access array

>tab.isarray()// verifier type
>tab.length()// nombre element d'un tableau

>tab.unshif(element)// ajouter element au debut
>tab.push(element)// ajouter element au dernier
>tab.splice(index,howmany ,item 1 )// add with index

>tab.shift()// supprimer premier element
>tab.pop()// supprimer dernier element

>tab.indexof(element,2)// return index , search from 2
>tab.lastIndexof(element,-1)//return index,search from last

>tab.includes(element)// true or false
>tab.sort()// trie tableau
>tab.reverse()// retourner tab reverser

>tab.slice(debut , fin)// retourner sous tableau
>tab.concat(tab1)// join two array
```

Object

```
// declaration

> let user{
    theName : 'sahbi'
    theAge : '38'

    SayHello:function(){
              Console.log('hello');
             }
  }

> let user{ }

// Access
>user.theName / with dot
>user[theName ]/ with bracket
 // pour dynamic property use bracket notation

// New Object

> let user{ };
> user.age=39;
> let user= newobject()
> let user= newobject({          ,
                                   ,
                          };        )


// this

> console.log(this); // this object actuelle


> let obj = object.create({});
// object to use as prototype ( heriter propriete)


// Assign Methode

> let finalobject = object.assign();
```

Select Elements

>document.getElementById()
>document.getelementsByTagName()
>document.getelemntsByclassName()
>document.Queryselector('.css selector')
>document.QueryselectorAll()
>document.Title
>document.body
>document.forms
>document.forms[0].one
>document.links
>document.links[0].href

Get/Set Elements

>document.Queryslector('.class).innerHtml
>document.Queryslector('.class).innerText
>document.Queryslector('.class).textContent

>myElement.getAttribute('href')
//retourner la valeur de l'attribut href

>myElement.setAttribute('href','www.google.com')
// changer ou ajouter l'attribut href

>myElement.attributes
// retourner une liste de tous les attributs

>myElement.hasAttribute('href')
>myElement.hasAttributes
// retourner true or false

>myElement.removeAttribute('data-src')
// supprimer l'attribut de l'élément

>myElement.createAttribute('data-src')
// créer attribut parsonnaliser

Create Elements

>document.createElement('div')
>document.createTextNode('lorem')
>document.createComment('lorem')
>MyElement.appendChild('myElement2')
>MyElement.className='prod'

Childrens

>MyElement.childrens
// retourner une liste des enfants
>MyElement.firstElementChild
>MyElement.lastElementChild
// retourner element tags

>MyElement.childNodes
>MyElement.firstChild
>MyElement.lastChild;

DOM Traversing

>MyElement.nextSibling
// retourner nœuds suivant
>MyElement.nextElementSibling
>MyElement.previousSibling
>MyElement.previousElementSibling
>MyElement.parentElement

Deals With Elements

>MyElement.before(variable)
// ajouter avant MyElement
>MyElement.after(variable)
>MyElement.append(variable)
// ajouter a l'intérieur en dernier
>MyElement.prepend(variable)
// ajouter a l'intérieur en premier
>MyElement.remove()

>MyElement.cloneNode(false)
// créer copie , true : copier avec tous les interieurs