



LE LANGAGE

JAVAScript

Module : Technologie Web

Année Universitaire : 2021 – 2022

Plan

- Partie 1

- JavaScript?
- HTML et JavaScript
- Les Variables
- Les Types
- Les Boites de dialogue
- Les Opérateurs Javascript
- Les Structures conditionnelle
- Les Itérations
- Hiérarchie d'objets en JavaScript
- Fonctions Prédéfinis

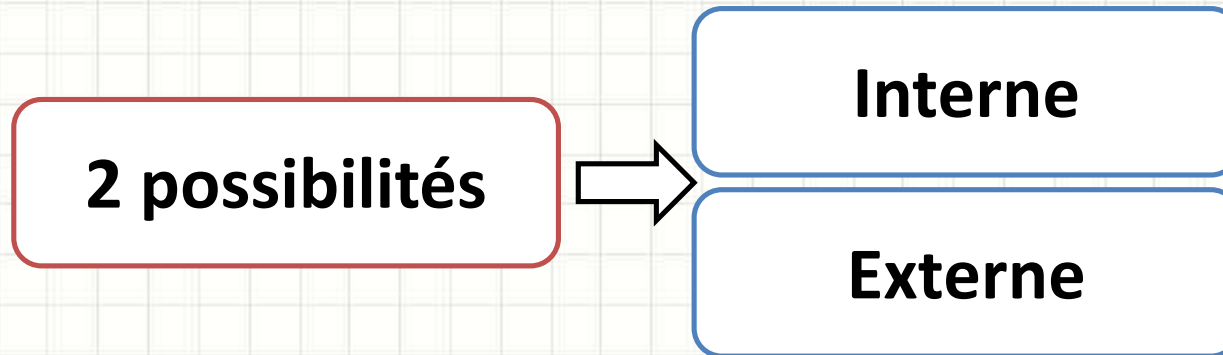
- Partie 2

- Déclaration des fonctions
- Méthodes utilisées
- L' Accès aux éléments

JavaScript?

- ✓ Javascript permet de rendre **intércatif** un site internet développé en HTML.
- ✓ Javascript permet de développer de véritables applications fonctionna nt exclusivement dans le cadre d'Internet.
- ✓ Javascript a été initialement élaboré par Netscape en association avec Sun Microsystem.
- ✓ Javascript est standardisé par un comité spécialisé, l'ECMA (European Computer Manufactures Association).
- ✓ Langage interprété au chargement de la page par le navigateur client

HTML et JavaScript



Interne: le code de JavaScript sera placé dans l'entête du document HTML

```
<head>
<meta http-equiv="Content-Type" content="text/html; cha
<title>Document sans nom</title>
<script language="javascript" type="text/javascript">

//instruction JavaScript

</script>
</head>
```

HTML et JavaScript

Externe: Regrouper les instructions de JavaScript dans un fichier (.js) externe au fichier HTML

```
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />  
<title>Document sans nom</title>  
<script language="javascript" type="text/javascript" src="FichierExterne.js">  
</script>  
</head>  
<body>
```

Les Variables

JavaScript est un langage **pauvrement typé**, il n'est pas indispensable de déclarer préalablement le type de variable.

- ➔ Utiliser ***var*** pour déclarer les variables
- ➔ Utiliser « ; » en fin d'instruction

Exemple:

```
var saluer="Bonjour";  
var nom ="Ali";
```


Les Types prédéfinis

- ✓ String, Number, Boolean, NaN, Null, etc.
- ➔ NaN est le sigle de "Not a Number", c'est à dire "Ce n'est pas un nombre !" C'est en fait le résultat d'une opération mathématique sans sens (la division de 0 par 0, la racine carrée d'un nombre négatif, etc.)
- ➔ NULL sert à spécifier une variable sans valeur

Les Boites de dialogue

PROMPT : `prompt('MSG','Valeur');`



Ex: `var x = prompt('veuillez mettre un message','valeur')`

➔ X prend la valeur entrée ou Null si aucune valeur n'est entrée

Les Boites de dialogue

Alert (retourne la valeur UNDEFINED)

EX: alert('Bienvenue à ESPRIT');



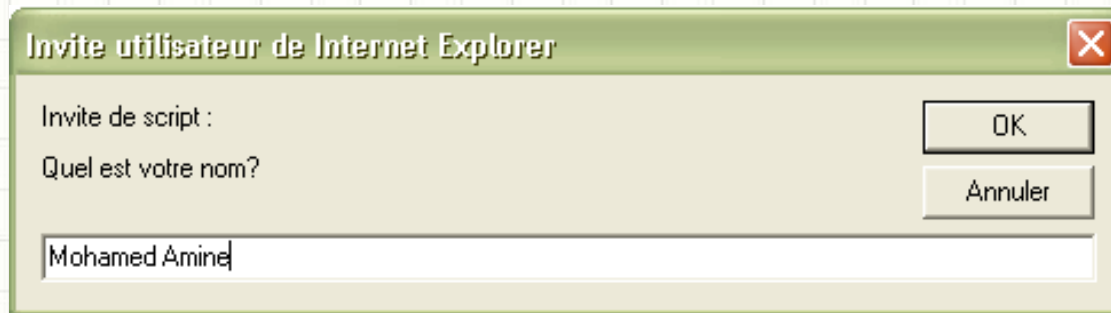
Confirm (retourne la valeur true ou false)

EX: confirm('Bienvenue à ESPRIT');



Exercice 1

Ecrire un script qui vous permet de réaliser les boîtes de dialogue suivantes:



Les Commentaires

Commentaires en JavaScript :

- Utiliser `//` pour les commentaires sur une ligne.
- Utiliser au début `/*` et à la fin `*/` pour les commentaires sur plusieurs lignes.

Les Opérateurs JavaScript

- Arithmétiques : + - / * ++ --

- Comparaison: < <= == != >= >

- Concaténation des chaînes : +

- Assignment : =

- Spéciales: == et != convertissent les
opérandes de même type avant la comparaison

La Structure Conditionnelle

- ✓ If (condition) instruction;
- ✓ If (condition) instruction else instruction;

Exemple:

```
<script type="text/javascript" language="javascript">  
var nom=prompt("Quel est votre nom?");  
if (nom=="Ali") alert("Bonjour "+nom);  
else if (nom == "Stephane") alert(" Hello "+nom);  
else alert ("Qui êtes-vous?");  
</script>
```

➔ REMARQUE: Mettre des { ...} pour plusieurs instructions dans un block conditionnel

Les Itérations

- ✓ While (condition) instruction;
- ✓ Do instruction While (condition);
- ✓ For (initialisation; condition; incrémentation) instructions;

Exemple:

```
<script type="text/javascript" language="javascript">  
var compteur;  
for(compteur=0;compteur<=3;compteur++)  
alert("Valeur: "+compteur);  
</script>
```

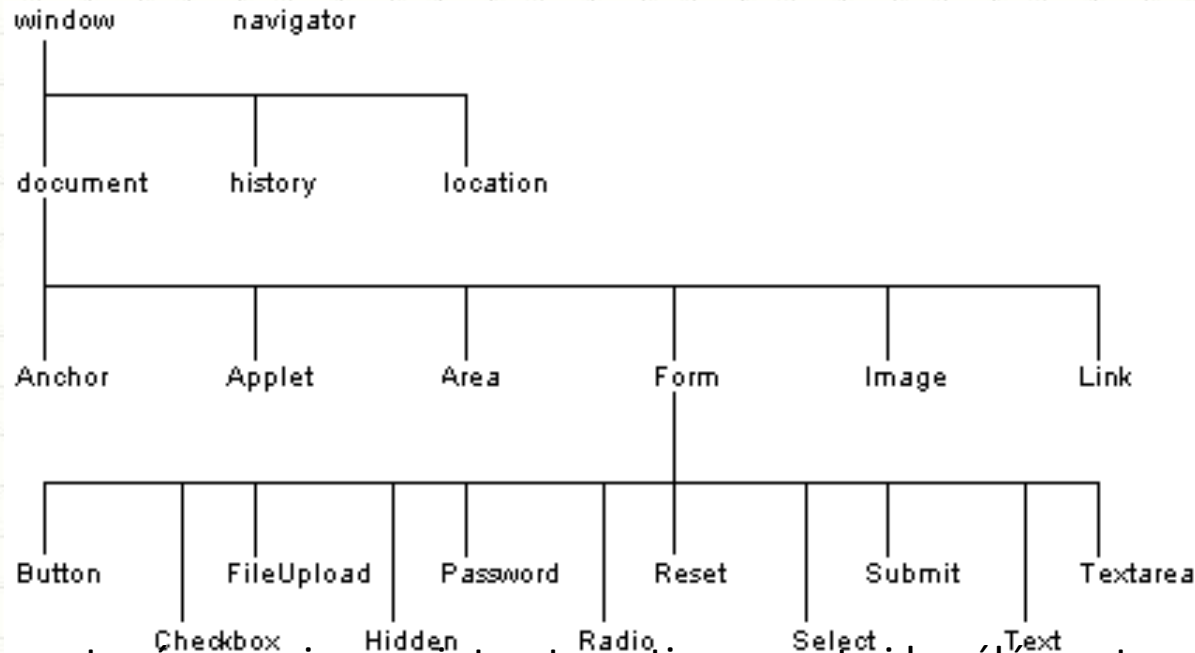
➔ REMARQUE: Mettre des { ...} pour plusieurs instructions dans un block itératif.

Notion d'hierarchie d'objets

- ✓ JavaScript n'est pas un langage orienté objet mais un langage basé sur les objets.
- ✓ Un objet est une entité prédéfinie propre à Javascript.

La page Web est considérée comme un
ENSEMBLE D'OBJETS

Notion d'hierarchie d'objets



- Ces objets sont créés par javascript automatiquement si les éléments correspondants existent dans la page
- Certains existent toujours
 - navigator
 - window
 - document
 - location
 - history

Notion d'hierarchie d'objets

- navigator
 - Contient le nom et la version du navigateur, les plugins installés...
- window
 - Propriétés qui s'appliquent à la fenêtre tout entière
- document
 - Propriétés sur le contenu du document (titre, couleur...)
- location
 - URL actuelle
- history
 - URLs visitées

Objet Array

Création de l'objet :

```
var mycars=new Array() ;  
mycars[0]="Saab" ;  
mycars[1]="Volvo" ;  
mycars[2]="BMW" ;
```

Ou bien :

```
var mycars=new Array("Saab" ,"Volvo" ,"BMW") ;
```

➔ Pour contrôler la grandeur du tableau :

```
var myArray=new Array(3) ;
```

Exercice: Affichage de la date du jour

```
<HTML>

<HEAD>  <TITLE> Exemple Date </TITLE> </HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
var dt=new Date;
var jour=dt.getDay( );      // renvoi un jour [0..6]
var numero=dt.getDate( );   // renvoi le numéro dans le mois
var mois=dt.getMonth( );    // renvoi le mois [0..11]

var tab_jour=new Array("Dimanche","Lundi","Mardi",
                        "Mercredi","Jeudi","Vendredi","Samedi");
var tab_mois=new Array("Janvier","Février","Mars","Avril","Mai",
                        "Juin","Juillet","Août","Septembre",
                        "Octobre","Novembre","Décembre");

document.write("Nous sommes le "+tab_jour[jour]+" "
               +numero+" "+tab_mois[mois]);

</SCRIPT>
</BODY>

</HTML>
```

Fonctions Prédéfinies

- **Parsefloat ()**
- **Parseint ()**
- **String ()**
- **Number ()**
- **Length()**

Fonctions Parsefloat

- ✓ `parseFloat(une_chaine);`
 - ➔ Convertit une chaîne en nombre à virgule flottante si la chaîne commence par un caractère numérique.

Exemple:

```
<script type="text/javascript" language="javascript">
var a = "54 a";
var b = "d 54";
var c = 12.5;
var d = "toto";
window.document.write(parseFloat(a));
window.document.write("<br/>");
window.document.write(parseFloat(b));
window.document.write("<br/>");
window.document.write(parseFloat(c));
window.document.write("<br/>");
window.document.write(parseFloat(d));
</script>
```




54
NaN
12.5
NaN

Fonctions Parseint

- ✓ `parseInt(chaine_de_caractère);`
 - ➔ Convertit l'argument en un nombre entier
 - ➔ Renvoie NaN si la conversion est impossible

Exemple:

```
<script type="text/javascript" language="javascript">  
var a = "54 a";  
var b = "d 54";  
var c = 12.5;  
var d = "toto";  
window.document.write(parseInt(a));  
window.document.write("<br/>");  
window.document.write(parseInt(b));  
window.document.write("<br/>");  
window.document.write(parseInt(c));  
window.document.write("<br/>");  
window.document.write(parseInt(d));  
</script>
```



```
54  
NaN  
12  
NaN
```

Fonctions String

✓ String (une_chose);

→ Convertit l'argument en une chaîne

Exemple:

```
<script type="text/javascript" language="javascript">  
var A = 55;  
window.document.write(" 1/ Le type de A est : " +typeof(A));  
window.document.write("<br/>");  
A = String (A);  
window.document.write("2/ Le type de A est : " +typeof(A));  
</script>
```

Resultat :

```
1/ Le type de A est : number  
2/ Le type de A est : string
```

Fonctions Number

- ✓ `Number(une_valeur);`
 - ➔ Convertit l'argument en un nombre
 - ➔ Renvoie NaN si la conversion est impossible

Exemple:

```
<script type="text/javascript" language="javascript">  
var a = "54";  
var b = "d54";  
var c = "54d";  
window.document.write(Number(a));  
window.document.write("<br/>");  
window.document.write(Number(b));  
window.document.write("<br/>");  
window.document.write(Number(c));  
</script>
```



54

NaN

NaN

Les Fonctions

Définies dans la partie <head> et appelé dans la partie <body>

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1"
<title>Document sans nom</title>
<script language="javascript" type="text/javascript">
function nom_de_la_fonction () {
//Instructions de la fonction;
}
</script>
</head>
<body>

<form>
<!--! Instruction du formulaire -->

<input type="button" value="valider" onclick="nom_de_la_fonction ()" />
</form>

</body>
</html>
```



Les Méthodes Utilisées

Instruction	Description
length	C'est un entier qui indique la longueur de la chaîne de caractères.
charAt()	Méthode qui permet d'accéder à un caractère isolé d'une chaîne.
indexOf()	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée. (en commençant au début de la chaîne principale soit en position 0).
lastIndexOf()	Méthode qui renvoie la position d'une chaîne partielle à partir d'une position déterminée. (en commençant à la fin soit en position length moins 1).
substring(x,y)	Méthode qui renvoie un string partiel situé entre la position x et la position y-1.
toLowerCase()	Transforme toutes les lettres en minuscules.
toUpperCase()	Transforme toutes les lettres en Majuscules.

La Propriété Length

- La propriété `length` retourne un entier qui indique le nombre d'éléments dans une chaîne de caractères. Si la chaîne est vide (""), le nombre est zéro.

La syntaxe est simple :

```
x=variable.length;
```

```
x=("chaîne de caractères").length;
```

La Propriété Length

- La propriété length ne sert pas que pour les Strings, mais aussi pour connaître la longueur ou le nombre d'éléments :
 - ✓ Formulaires: Combien a-t-il de formulaires différents ?
 - ✓ Boutons radio: Combien a-t-il de boutons radio dans un groupe ?
 - ✓ Cases à cocher: Combien a-t-il de cases à cocher dans un groupe ?
 - ✓ Options: Combien a-t-il d'options dans un Select ?

La Méthode CharAt()

Il faut d'abord bien noter que les caractères sont comptés de gauche à droite et que la position du premier caractère est 0. La position du dernier caractère est donc la longueur (length) de la chaîne de caractère moins 1.

chaîne : Javascript (longueur = 10)

| | | | | | | |

position : 0123456789 (longueur - 1)

Si la position que vous indiquer est inférieure à zéro ou plus grande que la longueur moins 1, Javascript retourne une chaîne vide.

La Méthode CharAt()

La syntaxe de charAt() est :

chaîne_réponse = chaîne_départ.charAt(x);

NB : où x est un entier compris entre 0 et la longueur de la chaîne à analyser moins 1

Notez l'exemple suivant :

```
var str="Javascript";  
var chr=str.charAt(0);  
var chr="Javascript".charAt(0);  
ou var chr=charAt(str,0);  
ou var  
chr=charAt("Javascript",0);  
➔ La réponse est "J".
```

La Méthode IndexOf()

Cette méthode renvoie la position, soit x, d'un string partiel (lettre unique, groupe de lettres ou mot) dans une chaîne de caractères en commençant à la position indiquée par y. Cela vous permet, par exemple, de voir si une lettre, un groupe de lettres ou un mot existe dans une phrase.

```
variable="chaîne_de_caractères";  
var="string_partiel";  
x=variable.indexOf(var,y);
```

La Méthode IndexOf()

✓ Où y est la position à partir de laquelle la recherche (de gauche vers la droite) doit commencer. Cela peut être tout entier compris entre 0 et la longueur - 1 de la chaîne de caractères à analyser.

Si y n'est pas spécifié, la recherche commencera par défaut à la position 0.

➔ Si le string partiel n'est pas trouvé dans la chaîne de caractères à analyser, la valeur retournée sera égale à -1.

La Méthode IndexOf()



Exemple:

```
variable="Javascript"  
var="script"  
x=variable.indexOf(var,0);
```

➔ x vaut 4

```
variable="VanlanckerLuc&ccim.be"  
var="@"  
x=variable.indexOf(var);
```

➔ x vaut -1

La Méthode substring()

La méthode substring() sera particulièrement utile, par exemple, pour prendre différentes données dans une longue chaîne de caractères.

variable = "chaîne de caractères"

resultat=variable.substring(x,y)

➔ Résultat est un sous ensemble de la chaîne de caractère (ou de la variable).

Les x et y sont des entiers compris entre 0 et la longueur moins 1 de la chaîne de caractères.

NB : Si x est égal à y, substring() retourne une chaîne vide.

La Méthode substring()

Javascript

| | | | | | | |

0 1 2 3 4 5 6 7 8 9

Exemple :

```
str="Javascript";
```

```
str1=str.substring(0,4);
```

```
str2="Javascript".substring(0,4);
```

```
str3=str.substring(6,9);
```

Les résultats sont :

```
str1="Java"; soit les positions 0,1,2 et 3.
```

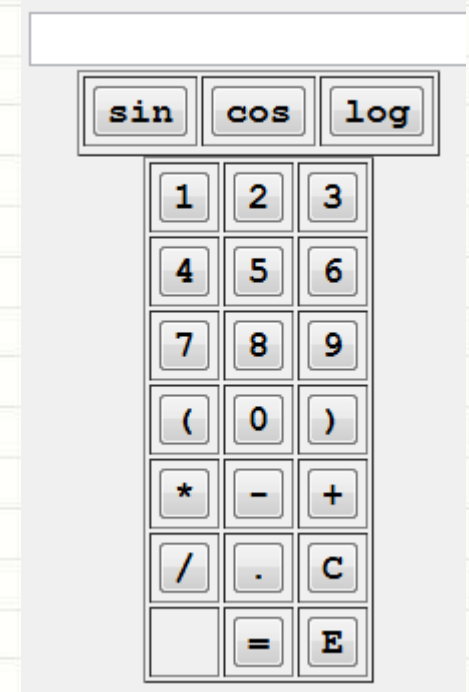
```
str2="Java"; soit les positions 0,1,2 et 3.
```

```
str3="rip"; soit les positions 6,7 et 8
```

Exercice: calculatrice javascript

Dans cette réalisation, le calcul repose sur la fonction **eval()** de *javascript*.

Attention, comme les fonctions mathématiques de javascript appartiennent à la classe Math, il faut, par exemple, évaluer `Math.sin(x)` pour obtenir `sin(x)`.



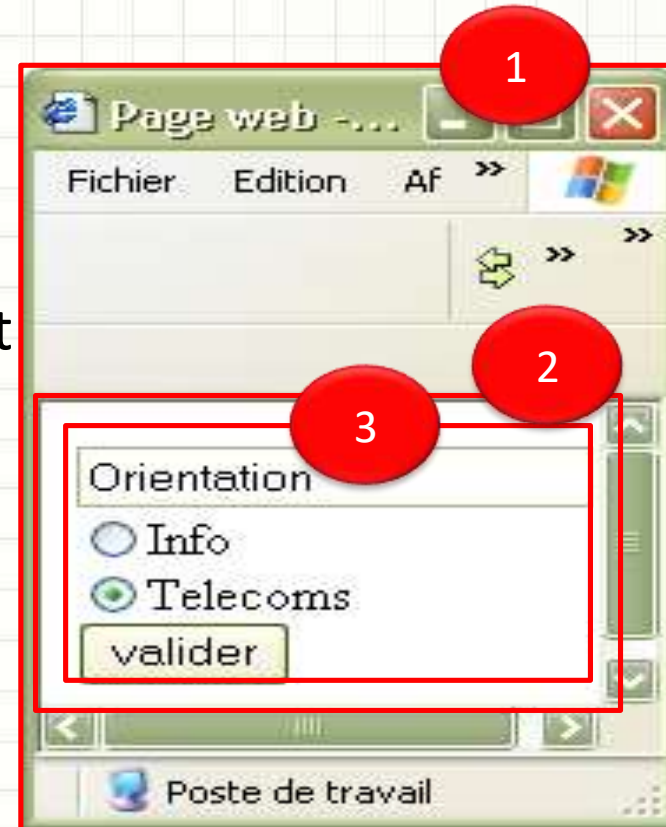
Accès aux objets du Formulaire

Window.document.forms[n].name_d'objet

1

2

3



- forms[n] (c'est le tableau des formulaires) peut être remplacé par le **nom** de la balise form ou par **getElementById(id de la balise form)**.
- Puisque Window occupe la première place dans l'hierarchie, il devient facultatif.

Accès aux éléments de Type Input

Les zones de texte:

La principale action en javascript sur une zone de texte est de manipuler son contenu.

➤ Il faut bien penser à ajouter la propriété **.value** pour accéder au contenu.

```
<script language="javascript" type="text/javascript">
function acced(){
var InputText = window.document.MonForm.MonChamp.value ;
alert("Le contenu du champ est : "+InputText);
}
</script>
</head>
<body>
<form name="MonForm">
Nom : <input type="text" name="MonChamp" />
<input type="button" value="Acceder" onclick="acced()" />
</form>
```


Exercice 5

Nom :

CIN:

1. Ecrire une fonction qui permet de vérifier les champs nom et CIN ne sont pas vide.
2. Ecrire une fonction qui permet de vérifier que le champs CIN est numérique et possède une longueur égale à 8 caractères.

Accès aux éléments de Type Input

Les radio boutons:

Pour détecter qu'une case est cochée, il faut utiliser sa propriété checked

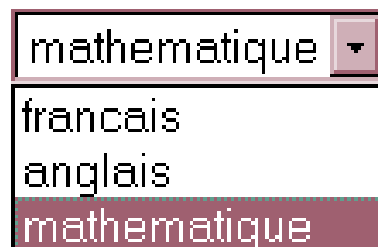
```
<script language="javascript" type="text/javascript">
function cocher(){
if(window.document.forms[0].choix[0].checked)
alert(window.document.forms[0].choix[0].value+" est cocher");
if(window.document.forms[0].choix[1].checked)
alert(window.document.forms[0].choix[1].value+"radio non cocher");
}
</script>
</head>

<body>
<form>
<input type="radio" name="choix" value="Esprit"/>Esprit<br />
<input type="radio" name="choix" value="Autres"/>Autres<br />
<input type="button" value="check" onclick="cocher()" />
```

L'accès aux éléments de types Select

```
//nous donne l'indice d'option sélectionné  
var t1 = window.document.forms[0].choix.selectedIndex;  
//nous donne la longueur de la liste  
var t2 = window.document.forms[0].choix.length;  
//La valeur du champs selectionner  
var t3 = window.document.forms[0].choix.options[t1].value;  
//Le libelle d'option selectionner  
var t4 = window.document.forms[0].choix.options[t1].text;
```

Résultat



mathematique ▼

français
anglais
mathematique

valider

La valeur de t1 est :2

La valeur de t2 est :3

La valeur de t3 est :math

La valeur de t4 est :mathematique



MERCI POUR VOTRE ATTENTION



LE LANGAGE

JAVAScript

Module : Technologie Web

Année Universitaire : 2021 – 2022

Les interfaces du DOM

Ce qu'on appelle « DOM » est en fait un ensemble d'interfaces qui vont pouvoir fonctionner ensemble et nous permettre notamment d'accéder à et de manipuler divers éléments de nos documents en JavaScript.

Les interfaces du DOM

Parmi les interfaces du DOM:

- L'interface **Window** qui est liée au DOM
- L'interface **Event** qui représente tout événement qui a lieu dans le DOM
- L'interface **EventTarget** qui est une interface que vont implémenter les objets qui peuvent recevoir des événements
- L'interface **Node** qui est l'interface de base pour une grande partie des objets du DOM
- L'interface **Document** qui représente le document actuel et qui va être l'interface la plus utilisée
- L'interface **Element** qui est l'interface de base pour tous les objets d'un document

Accéder aux éléments dans un document avec JavaScript et modifier leur contenu

L'interface DOM va nous permettre de manipuler le contenu HTML et les styles d'un document.

Accéder à un élément à partir de son sélecteur CSS associé

querySelector() : une méthode qui retourne un objet `Element` représentant le premier élément dans le document correspondant au sélecteur (ou au groupe de sélecteurs) CSS passé en argument ou la valeur `null` si aucun élément correspondant n'est trouvé.

querySelectorAll() : renvoie un objet appartenant à l'interface `NodeList` (les objets `NodeList` sont des collections (des listes) de nœuds).

- La méthode `querySelectorAll()` renvoie un objet appartenant à l'interface `NodeList`.
- L'objet `NodeList` renvoyé est une liste statique
- Pour itérer dans cette liste d'objets `NodeList` et accéder à un élément en particulier, on utilise la méthode `forEach()`. Cette méthode prend une fonction de **rappel** en argument qui peut prendre jusqu'à trois arguments optionnels qui représentent :
 - ✓ L'élément en cours de traitement dans la `NodeList`
 - ✓ L'index de l'élément en cours de traitement dans la `NodeList`
 - ✓ L'objet `NodeList` auquel `forEach()` est appliqué.

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours JavaScript</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="cours.css">
    <script src='cours.js' async></script>
  </head>

  <body>
    <h1 class='bleu'>Titre principal</h1>
    <p id='p1'>Un paragraphe</p>
    <div>
      <p>Un paragraphe dans le div</p>
      <p class='bleu'>Un autre paragraphe dans le div</p>
    </div>
    <p>Un autre paragraphe</p>
  </body>
</html>
```

Exemple

```
/*Sélectionne le premier paragraphe du document et change son texte avec la
 *propriété textContent que nous étudierons plus tard dans cette partie*/
document.querySelector('p').textContent = '1er paragraphe du document';

let documentDiv = document.querySelector('div'); //1er div du document
//Sélectionne le premier paragraphe du premier div du document et modifie son texte
documentDiv.querySelector('p').textContent = '1er paragraphe du premier div';

/*Sélectionne le premier paragraphe du document avec un attribut class='bleu'
 *et change sa couleur en bleu avec la propriété style que nous étudierons
 *plus tard dans cette partie*/
document.querySelector('p.bleu').style.color = 'blue';

//Sélectionne tous les paragraphes du document
let documentParas = document.querySelectorAll('p');

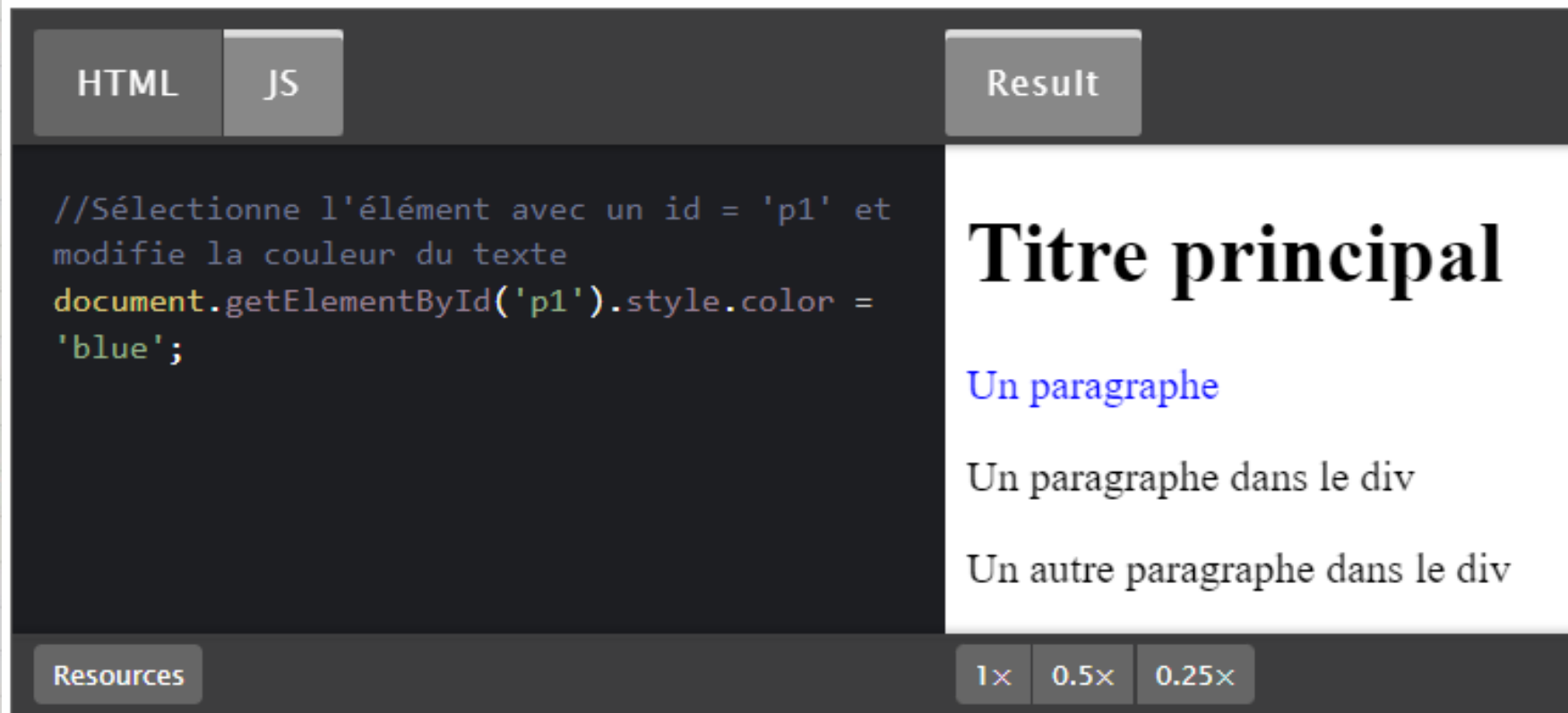
//Sélectionne tous les paragraphes du premier div
let divParas = documentDiv.querySelectorAll('p');

/*On utilise forEach() sur notre objet NodeList documentParas pour rajouter du
 *texte dans chaque paragraphe de notre document*/
documentParas.forEach(function(nom, index){
    nom.textContent += ' (paragraphe n°:' + index + ')';
});
```

Accéder à un élément en fonction de la valeur de son attribut id

- La méthode `getElementById()` est une méthode du mixin `NonElementParentNode` et qu'on va implémenter à partir d'un objet `Document`.
- Cette méthode renvoie un objet `Element` qui représente l'élément dont la valeur de l'attribut `id` correspond à la valeur spécifiée en argument.
- La méthode `getElementById()` est un moyen simple d'accéder à un élément en particulier (si celui-ci possède un `id`) puisque les `id` sont uniques dans un document.

Exemple



The screenshot shows a web development tool interface with a dark theme. At the top, there are three tabs: 'HTML', 'JS', and 'Result'. The 'JS' tab is currently selected. Below the tabs, the 'JS' panel contains a JavaScript code snippet. The 'Result' panel displays the rendered output of the code. At the bottom, there is a 'Resources' panel and a zoom control area with buttons for '1x', '0.5x', and '0.25x'.

```
//Sélectionne l'élément avec un id = 'p1' et  
modifie la couleur du texte  
document.getElementById('p1').style.color =  
'blue';
```

Titre principal

Un paragraphe

Un paragraphe dans le div

Un autre paragraphe dans le div

Resources

1x 0.5x 0.25x

Accéder à un élément en fonction de la valeur de son attribut class

`getElementsByClassName()` renvoie une liste des éléments possédant un attribut class avec la valeur spécifiée en argument.

La liste renvoyée est un objet de l'interface `HTMLCollection` qu'on va pouvoir traiter quasiment comme un tableau.

Exemple

```
//Sélectionne les éléments avec une class = 'bleu'  
let bleu = document.getElementsByClassName('bleu');  
  
//"bleu" est un objet de HTMLCollection qu'on va manipuler comme un tableau  
for(valeur of bleu){  
    valeur.style.color = 'blue';  
}
```

Accéder à un élément en fonction de son identité

`getElementsByName()` permet de sélectionner des éléments en fonction de leur nom et renvoie un objet `HTMLCollection` qui consiste en une liste d'éléments correspondant au nom de balise passé en argument.

Lorsqu'on utilise `getElementsByName()` avec un objet `Document`, la recherche se fait dans tout le document

Lorsqu'on utilise `getElementsByName()` avec un objet `Element`, la recherche se fera dans l'élément en question seulement.

Exemple

```
//Sélectionne tous les éléments p du document
let paras = document.getElementsByTagName('p');

// "paras" est un objet de HTMLCollection qu'on va manipuler comme un tableau
for(valeur of paras){
    valeur.style.color = 'blue';
}
```

Accéder au contenu des éléments et les modifier

Pour récupérer le contenu d'un élément ou le modifier, nous allons pouvoir utiliser l'une des propriétés suivantes :

- La propriété **innerHTML** permet de récupérer ou de redéfinir la syntaxe HTML interne à un élément
- La propriété **outerHTML** permet de récupérer ou de redéfinir l'ensemble de la syntaxe HTML interne d'un élément et de l'élément en soi
- La propriété **textContent** représente le contenu textuel d'un nœud et de ses descendants.
- La propriété **innerText** représente le contenu textuel visible sur le document final d'un nœud et de ses descendants.

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours JavaScript</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="cours.css">
    <script src='cours.js' async></script>
  </head>

  <body>
    <h1>Titre principal</h1>
    <p id='p1'>Un paragraphe</p>
    <div>
      <p>Un paragraphe dans le div</p>
      <p id='texte'>Un autre paragraphe dans le div</p>
    </div>
    <p id='p2'>Un autre paragraphe
      <span style='visibility: hidden'>avec du contenu caché</span>
    </p>
    <p id='p3'></p>
  </body>
</html>
```

Exemple

```
//Accède au contenu HTML interne du div et le modifie
document.querySelector('div').innerHTML +=
    '<ul><li>Elément n°1</li><li>Elément n°2</li></ul>';

//Accède au HTML du 1er paragraphe du document et le modifie
document.querySelector('p').outerHTML = '<h2>Je suis un titre h2</h2>';

/*Accède au contenu textuel de l'élément avec un id='texte' et le modifie.
 *Les balises HTML vont ici être considérées comme du texte*/
document.getElementById('texte').textContent = '<span>Texte modifié</span>';

//Accède au texte visible de l'élément avec l'id = 'p2'
let texteVisible = document.getElementById('p2').innerText;
//Accède au texte (visible ou non) de l'élément avec l'id = 'p2'
let texteEntier = document.getElementById('p2').textContent;

//Affiche les résultats du dessus dans l'élément avec l'id = 'p3'
document.getElementById('p3').innerHTML =
    'Texte visible : ' + texteVisible + '<br>Texte complet : ' + texteEntier;
```


Accéder au parent ou à la liste des enfants d'un nœud

La propriété **parentNode** de l'interface Node renvoie le parent du nœud spécifié dans l'arborescence du DOM ou null si le nœud ne possède pas de parent.

La propriété **childNodes** de cette même interface renvoie une liste des nœuds enfants de l'élément donné.

A noter que la propriété **childNodes** renvoie tous les nœuds enfants et cela quels que soient leurs types : nœuds élément, nœuds texte, nœuds commentaire, etc.

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours JavaScript</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="cours.css">
    <script src='cours.js' async></script>
  </head>

  <body>
    <h1>Titre principal</h1>
    <p id='p1'>Un paragraphe <span>avec un span</span></p>
    <div>
      <p id='p2'>Un paragraphe dans le div</p>
      <p>Un autre paragraphe dans le div</p>
    </div>
    <p>Un autre paragraphe</p>
  </body>
</html>
```

Exemple

```
let p1 = document.getElementById('p1');
let p2 = document.getElementById('p2');

p2.parentNode.style.backgroundColor = 'Rgba(240,160,000,0.5)'; //Orange

//On accède à tous les noeuds enfants de p1. childNodes renvoie une NodeList
let enfantsP1 = p1.childNodes;

/*On peut ensuite utiliser une boucle forEach() pour tous les manipuler ou
 *un indice comme pour les tableaux pour manipuler un noeud enfant en
 *particulier (le premier enfant a l'indice 0, le deuxième l'indice 1, etc.)*/
enfantsP1[1].style.fontWeight = 'bold';

/*On accède aux noeuds enfants éléments seulement de p1.
 *children renvoie une HTMLCollection*/
let enfantsEltP1 = p1.children;

//On peut ensuite accéder aux différents enfants comme on le ferait avec un tableau
enfantsEltP1[0].style.textDecoration = 'underline';
```

Accéder à un nœud enfant en particulier à partir d'un nœud parent

- La propriété **firstChild** de l'interface Node renvoie le premier nœud enfant direct d'un certain nœud ou null s'il n'en a pas.
- La propriété **lastChild**, au contraire, renvoie le dernier nœud enfant direct d'un certain nœud ou null s'il n'en a pas.
- Notez que ces deux propriétés vont renvoyer les premiers et derniers nœuds enfants quels que soient leurs types (nœuds élément, nœuds texte ou nœuds commentaire).
- Pour renvoyer le premier et le dernier nœud enfant de type élément seulement d'un certain nœud, on utilisera plutôt les propriétés **firstElementChild** et **lastElementChild**

Exemple

```
//On accède au premier noeud enfant de body
let bodyFirstChild = document.body.firstChild;

//On accède au dernier noeud enfant de body
let bodyLastChild = document.body.lastChild;

//On accède au premier noeud enfant élément de body
let bodyFirstChildElement = document.body.firstChildElement;

//On accède au dernier noeud enfant élément de body
let bodyLastElementChild = document.body.lastElementChild;

alert(
    'Premier noeud enfant de body : ' + bodyFirstChild +
    '\nPremier noeud enfant élément de body : ' + bodyFirstChildElement +
    '\nDernier noeud enfant de body : ' + bodyLastChild +
    '\nDernier noeud enfant élément de body : ' + bodyLastElementChild
);
```



Ajouter, modifier ou supprimer des éléments du DOM avec JavaScript

Créer de nouveaux nœuds et les ajouter dans l'arborescence du DOM

- Pour créer un nouvel élément HTML en JavaScript, on utilise la méthode `createElement()` de l'interface Document.

```
let newP = document.createElement('p');  
newP.textContent = 'Paragraphe créé et inséré grâce au JavaScript';
```

- Pour créer un nœud texte

```
let newP = document.createElement('p');  
let newTexte = document.createTextNode('Texte écrit en JavaScript');
```

Insérer un nœud dans le DOM

- Il existe différentes méthodes qui nous permettent d'insérer des nœuds dans d'autres nœuds.
- La différence entre ces méthodes va souvent consister dans la position où le nœud va être inséré.
- Les méthodes `prepend()` et `append()` du mixin permettent respectivement d'insérer un nœud ou du texte avant le premier enfant d'un certain nœud ou après le dernier enfant de ce nœud.

Avancer, reculer et actualiser



1. Accès à la page précédente : `history.back()`

2. Accès à la page suivante : `history.forward()`

3. `go()`

Avance ou recule du nombre de pages désiré dans l'historique sauvegardé. Attend comme paramètre le nombre de pages à sauter. Un nombre négatif recule (autant de fois page précédente que mentionné). Un nombre positif avance (autant de fois page suivante que mentionné).

4. Rafraîchir la page: `history.go(0)`

`Précédente`

`Suivante`

`Actualiser`

` Précédente`

Fonctions mathématiques



- **Math.abs(nombre)**
La méthode abs() renvoie la valeur absolue du nombre.
- **Math.ceil(nombre)**
La méthode ceil() renvoie l'entier supérieur ou égal au nombre.
- **Math.floor(nombre)**
La méthode floor() renvoie l'entier inférieur ou égal au nombre.
- **Math.round(nombre)**
La méthode round() arrondit le nombre à l'entier le plus proche.
- **Math.max(nombre1, nombre2)**
La méthode max(nombre1, nombre2) renvoie le plus grand des 2 nombres.
- **Math.min(nombre1, nombre2)**
La méthode min(nombre1, nombre2) renvoie le plus petit des 2 nombres.
- **Math.pow(nombre1, nombre2)**
La méthode pow(nombre1, nombre2) calcule la valeur d'un nombre nombre1 à la puissance nombre2.
- **Math.random()**
La méthode random() renvoie la valeur d'un nombre aléatoire choisi entre 0 et 1.
- **Math.sqrt(nombre)**
La méthode sqrt(nombre) renvoie la racine carrée du nombre.

Fonctions de contrôle



parseInt(string) ou parseInt(string, radix)

Cette fonction convertit une chaîne contenant un nombre en une valeur entière.

```
str='1.2345';
```

```
    x=parseInt(str);
```

```
document.write(x + "<br />");
```

```
document.write(parseInt("A", 16) + "<br />");
```

```
document.write(parseInt("A", 10) + "<br />");
```

```
document.write(parseInt("A") + "<br />");
```

```
document.write(parseInt("15", 10) + "<br />");
```

```
document.write(parseInt(15.99, 10)+ "<br />");
```

```
document.write(parseInt(15.99)+ "<br />");
```

```
document.write(parseInt("FXX123", 16)+ "<br />");
```

```
document.write(parseInt("1111", 2)+ "<br />");
```

```
document.write(parseInt("15*3", 10)+ "<br />");
```

```
document.write(parseInt("XX")+ "<br />");
```

```
document.write(parseInt("123XX")+ "<br />");
```

1
10
NaN
NaN
15
15
15
15
15
15
NaN
123

Fonctions de contrôle

parseFloat(string)

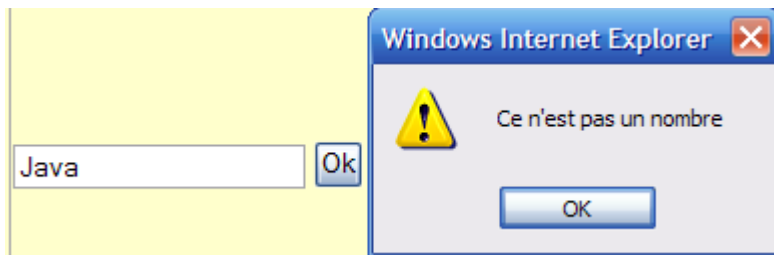
Cette fonction convertit une chaîne contenant un nombre en une valeur à virgule flottante.

```
document.write(parseFloat("3.14")+ "<br />");  
document.write(parseFloat("314e-2")+ "<br />");  
document.write(parseFloat("0.0314E+2")+ "<br />");  
var x = "3.14";  
document.write(parseFloat(x)+ "<br />");  
x='-0.123';  
document.write(parseFloat(x)+ "<br />");  
x='$123';  
document.write(parseFloat(x)+ "<br />");
```

3.14
3.14
3.14
3.14
-0.123
NaN

isNaN(paramètre)

isNaN() détermine si le résultat de la fonction parseFloat ou parseInt est un “NaN” : Not-a-Number.



```
function numerique(nb){  
  var x = parseFloat(nb.value);  
  //ou encore var x = parseInt(nb.value);  
  if (isNaN(x)) {  
    alert('Ce n\'est pas un nombre');  
    nb.value="";  
  } else {alert(x +' : C\'est un nombre');}  
}
```


Exercice



Récupérer dans la variable i, un nombre entier aléatoire entre 0 et 10. pour les cas 0 à 5, afficher bonjour à tous !! nombre de fois. Dans les autres cas, afficher bonjour à tous !!! plusieurs fois

Le nombre choisi est 2, alors
Bonjour à tous !!! deux fois

```
var i = parseInt(Math.random() * 10);  
var s = 'Bonjour à tous !!!';  
document.write('Le nombre choisi est ' + i + ',alors <br />');  
switch (i)  
{  
  case 0 :document.write(s + 'zéro fois'); break;  
  case 1 :document.write(s + 'une fois'); break;  
  case 2 :document.write(s + 'deux fois'); break;  
  case 3 :document.write(s + 'trois fois'); break;  
  case 4 :document.write(s + 'quatre fois'); break;  
  case 5 :document.write(s + 'cinq fois'); break;  
  default :document.write(s + 'plusieurs fois');  
}
```

Chaînes de caractères



1. La propriété length

La propriété length retourne un entier qui indique le **nombre de caractères** dans une chaîne de caractères. Si la chaîne est vide (""), le nombre est zéro.

Syntaxe :

```
x=variable.length;
```

```
x="chaîne de caractères".length;
```

2. La méthode CharAt()

Les caractères sont comptés de gauche à droite: **la position du premier caractère est 0**. La position du dernier caractère est donc la longueur (length) de la chaîne de caractères moins 1.

Si la position que vous indiquez est inférieure à zéro ou plus grande que la longueur moins 1, Javascript retourne une chaîne vide.

La syntaxe de charAt() est :

```
chaîne_réponse = chaîne_départ.charAt(x);
```

où x est un entier compris entre 0 et la longueur de la chaîne à analyser moins 1.

3. La méthode indexOf()

Cette méthode renvoie la position, soit x, d'un string partiel **ss** (lettre unique, groupe de lettres ou mot) dans une chaîne de caractères en commençant à la position indiquée par **position**. Cela vous permet, par exemple, de voir si une lettre, un groupe de lettres ou un mot existe dans une phrase.

```
variable="chaîne_de_caractères";
```

```
ss="string_partiel";
```

```
x=variable.indexOf(ss, position);
```

où position est la position à partir de laquelle la recherche (de gauche vers la droite) doit commencer. Cela peut être tout entier compris entre 0 et la longueur - 1 de la chaîne de caractères à analyser.

Si position n'est pas spécifiée, la recherche commencera par défaut à la position 0. Si le string partiel n'est **pas trouvé dans la chaîne de caractères à analyser, la valeur retournée sera égale à -1.**

Chaînes de caractères



4. La méthode `lastIndexOf()`

Méthode semblable à `indexOf()` sauf que la recherche va cette fois de droite à gauche (en commençant donc par la fin).

```
position=variable.lastIndexOf(ss);
```

Notons que même lorsqu'on commence à lire de la fin de la chaîne, la position retournée est comptée depuis le début de la chaîne avec le comptage commençant à zéro.

5. La méthode `substring()`

Retourne une sous-chaîne de la chaîne initiale comprise entre la position x et la position y.

```
variable = "chaîne de caractères"
```

```
resultat=variable.substring(x,y)
```

Les x et y sont des entiers compris entre 0 et la longueur moins 1 de la chaîne de caractères.

Si x est inférieur à y, la valeur retournée commence à la position x et se termine à la position y-1.

Si x est supérieur à y, la valeur retournée commence à la position y et se termine à la position x-1.

En fait, ceci donne le même résultat et il est équivalent d'écrire par exemple **`substring(3,6)`** ou **`substring(6,3)`**.

Si x est égal à y, `substring()` retourne une chaîne vide.

6. La méthode `toLowerCase()`

Cette méthode affiche toutes les majuscules d'une chaîne de caractères variable2 en minuscules.

```
variable2="chaîne de caractères";
```

```
variable1=variable2.toLowerCase();
```

7. La méthode `toUpperCase()`

Cette méthode affiche toutes les minuscules d'une chaîne de caractères variable2 en majuscules.

```
variable2="chaîne de caractères";
```

```
variable1=variable2.toUpperCase();
```

Chaînes de caractères- Exemples



```
var chaine="chaîne_de_caractères";
var longueur=chaine.length;
document.write("Longueur de '" + chaine + "' : " + longueur + "<br/>");

document.write("Caractère à la position 3 : " + chaine.charAt(3) + "<br/>");
var ss="de" ;    var pos=chaine.indexOf(ss, 0);
document.write("Position de '" + ss+ "' : " + pos + "<br/>");
ss="ded" ;    pos=chaine.indexOf(ss, 0);
document.write("Position de '" + ss+ "' : " + pos + "<br/>");

ss= chaine.substring(3,6);    document.write("Sous chaine '" + ss + "'<br/>");
ss= chaine.substring(6,3);    document.write("Sous chaine '" + ss + "'<br/>");
ss= chaine.substring(3,3);    document.write("Sous chaine '" + ss + "'<br/>");

var minuscule=chaine.toLowerCase();
document.write("Chaine en miniscule'" + minuscule + "'<br/>");
var majuscule=chaine.toUpperCase();
document.write("Chaine en majuscule '" + majuscule + "'<br/>");
```

Chaînes de caractères- Exemples



Longueur de 'chaîne_de_caractères' : 20

Caractère à la position 3 : î

Position de 'de' : 7

Position de 'ded' : -1

Sous chaine 'îne'

Sous chaine 'îne'

Sous chaine ''

Chaine en miniscule 'chaîne_de_caractères'

Chaine en majuscule 'CHAÎNE_DE_CARACTÈRES'

Exercice



- Ecrire une fonction **compterNbVoyelles()** qui prend un mot en paramètre et renvoie son nombre de voyelles. Utilisez cette fonction pour afficher le nombre de voyelles et de consonnes du mot de la manière suivante:

Entrez un mot :

ESEN

☐ Empêcher cette page de générer des boîtes de dialogue supplémentaires

OK Annuler

'ESEN' contient 2 voyelle(s) et 2 consonne(s)

Exercice- A améliorer

```
function compterNbVoyelles(mot) {  
  var nbVoyelles = 0;  
  var motConverti= mot.toLowerCase();  
  for (var i = 0; i < motConverti.length; i++) {  
    var lettre = motConverti[i];  
    if ((lettre === 'a') || (lettre === 'e') || (lettre === 'i') ||  
        (lettre === 'o') || (lettre === 'u') || (lettre === 'y')) {  
      nbVoyelles++;  
    }  
  }  
  return nbVoyelles;  
}  
  
var motSaisi = prompt("Entrez un mot :");  
var nbVoyelles = compterNbVoyelles(motSaisi);  
document.write("''" + motSaisi + "''" + " contient " + nbVoyelles + "  
voyelle(s) et " + (motSaisi.length - nbVoyelles) + " consonne(s)"); 91
```

Date et heure



```
date1=new Date();
```

```
document.write('La date actuelle est : ' + date1+'<br />');
```

```
date2=new Date("Mar 26,2004 00:00:00");
```

```
document.write('La date crée est : ' + date2 + '<br />');
```

```
document.write('L\'année en cours est : ' + date1.getFullYear()+'<br />');
```

```
document.write('L\'année de la date crée est : ' + date2.getFullYear()+'<br />');
```

```
mois = date1.getMonth() + 1;
```

```
document.write
```

```
('Le mois en cours est : ' + mois+'<br />');
```

```
jour = date1.getDate();
```

```
document.write
```

```
('Le jour du mois est : ' + jour+'<br />');
```

```
jour = date1.getDay();
```

```
document.write
```

```
('Le jour de la semaine est : ' + jour+'<br />');
```

```
heure = date1.getHours();
```

```
document.write('L\'heure en cours est : ' + heure+'<br />');
```

```
minutes = date1.getMinutes();
```

```
document.write('Les minutes sont actuellement : ' + minutes+'<br />');
```

```
secondes = date1.getSeconds();
```

```
document.write('Les secondes sont actuellement : ' + secondes+'<br />');
```

La date actuelle est : Tue Dec 05 2017 20:03:35
GMT+0100 (Paris, Madrid)

La date crée est : Fri Mar 26 2004 00:00:00
GMT+0100 (Paris, Madrid)

L'année en cours est : 2017

L'année de la date crée est : 2004

Le mois en cours est : 12

Le jour du mois est : 5

Le jour de la semaine est : 2

L'heure en cours est : 20

Les minutes sont actuellement : 3

Les secondes sont actuellement : 35

Tableaux



L'objet **Array** (ou tableau) est une liste d'éléments indexés, de différents types.

Construction: Approche 1

On commence par définir le tableau :

```
nom_du_tableau = new Array (x);
```

où x est le nombre d'éléments du tableau.

Ensuite, on va alimenter la structure ainsi définie :

```
nom_du_tableau[i] = "élément";
```

où i est un nombre compris entre 0 et x moins 1.

Exemple :

```
couleur = new Array(3);
```

```
// ou encore couleur = new Array();
```

```
couleur [0]="Rouge";
```

```
couleur [1]="Vert";
```

```
couleur [2]="Bleu";
```

```
for(i=0; i < couleur.length ; i++)
```

```
    document.write('La couleur numéro ' + i + ' est ' + couleur[i] + '<br />');
```

La couleur numéro 0 est Rouge

La couleur numéro 1 est Vert

La couleur numéro 2 est Bleu

Construction: Approche 2

```
var tabMois = new
```

```
Array("Janvier","Fevrier","Mars","Avril","Mai","Juin","Juillet","Août",  
"Septembre","Octobre","Novembre","Decembre");
```

Construction: Approche 3

```
tableau = [1,0.25, true, "Ahmed"];
```

```
document.write(tableau+"<br/>");
```

1,0.25,true,Ahmed

Tableaux

Propriétés et méthodes



- **Propriété: length** : Retourne le nombre d'éléments du tableau.
- **sort()** : Trie les éléments par ordre croissant ou alphabétique croissant
- **reverse()** : Inverse l'ordre des éléments dans le tableau (ne les trie pas)
- **push()** : Ajouter des éléments en fin de tableau.
- **pop()** : Supprime le dernier élément du tableau.
- **concat()** : Renvoie un tableau concaténant le tableau initial avec les tableaux passés en paramètre (concaténation de tableaux)
- **slice()** : Renvoie les éléments d'un tableau de l'index "debut" (inclus) à l'index "fin" (exclus et facultatif) concaténés par une virgule (,)
- **unshift()** : Ajoute des éléments au début du tableau.
- **shift()** : Supprime le premier élément du tableau.

- **join()** : Regroupe tous les éléments du tableau dans une seule chaîne. Les différents éléments sont séparés par un caractère séparateur spécifié en argument. Par défaut, ce séparateur est une virgule.
- **toString()** : Renvoie une chaîne de caractère contenant les éléments du tableau

Tableaux

Propriétés et méthodes- Exemples



```
couleur = ["Rouge","Vert","Bleu"]
document.write("Tableau initial: " + couleur+"<br />");
document.write("Longueur du tableau: "+ couleur.length+"<br />");
couleur.sort();
document.write("Tableau Trié: "+ couleur + "<br />");
couleur.reverse();
document.write("Tableau inversé: "+ couleur + "<br />");
couleur.push("Rouge", "Gris");
document.write("Tableau après push: " + couleur+"<br />");
couleur.pop();
document.write("Tableau après pop: " + couleur+"<br />");
t=new Array("Jaune","Orange")
tfinal=couleur.concat(t);
document.write("Nouveau tableau après concat: "+ tfinal + "<br />");

document.write("Une partie du tableau de 2 à 4: " + tfinal.slice(2,5)+ "<br />");
tfinal.unshift("Noir","Blanc");
document.write("Tableau après unshift: " +tfinal+ "<br />");
tfinal.shift();
document.write("Tableau après shift: " +tfinal+ "<br />");
document.write("Tableau après join: " +tfinal.join(" - ")+ "<br />");
var ch = tfinal.toString();
document.write("Chaine du tableau: "+ ch + "<br />");
```

Tableaux

Propriétés et méthodes- Exemples



Tableau initial: Rouge, Vert, Bleu

Longueur du tableau: 3

Tableau Trié: Bleu, Rouge, Vert

Tableau inversé: Vert, Rouge, Bleu

Tableau après push: Vert, Rouge, Bleu, Rouge, Gris

Tableau après pop: Vert, Rouge, Bleu, Rouge

Nouveau tableau après concat: Vert, Rouge, Bleu, Rouge, Jaune, Orange

Une partie du tableau de 2 à 4: Bleu, Rouge, Jaune

Tableau après unshift: Noir, Blanc, Vert, Rouge, Bleu, Rouge, Jaune, Orange

Tableau après shift: Blanc, Vert, Rouge, Bleu, Rouge, Jaune, Orange

Tableau après join: Blanc - Vert - Rouge - Bleu - Rouge - Jaune - Orange

Chaine du tableau: Blanc, Vert, Rouge, Bleu, Rouge, Jaune, Orange

Tableau à deux dimensions



On peut créer des tableaux à deux dimensions (et plus encore). On déclare d'abord un tableau à 1 dimension de façon classique :

```
nom_du_tableau = new Array (x);
```

Ensuite, on déclare chaque élément du tableau comme un tableau à 1 dimension :

```
nom_du_tableau[i] = new Array(y);
```

Ou encore, par exemple: **Matrice[i][j] = ...**

(Matrice c'est le nom du tableau dans l'exemple)

Ou encore:

```
matrice=[["Rouge","Vert","Bleu"],["Jaune","Orange"],["Noir","Blanc"]];
```

```
document.write(matrice[1][0]+"<br />");
```

Jaune

Tableau à deux dimensions- Exemple



```
tableau = new Array (3);  
    tableau[0] = new Array(3);  
    tableau[1] = new Array(2);  
    tableau[2] = new Array(2);  
    tableau[0][0]="Rouge";  
    tableau[0][1]="Vert";  
    tableau[0][2]="Bleu";  
    tableau[1][0]="Jaune";  
    tableau[1][1]="Orange";  
    tableau[2][0]="Noir";  
    tableau[2][1]="Blanc";  
    document.write(tableau+"<br />");  
    matrice=[  
        ["Rouge","Vert","Bleu"],  
        ["Jaune","Orange"],  
        ["Noir","Blanc"]  
    ];  
    document.write(matrice+"<br />");  
    document.write(matrice[1][0]+"<br />");
```

Rouge,Vert,Bleu,Jaune,Orange,Noir,Blanc
Rouge,Vert,Bleu,Jaune,Orange,Noir,Blanc
Jaune

Définir ses propres objets



- Vous pouvez définir vos propres objets si vous voulez programmer en JavaScript de façon strictement orientée objet.
- Différentes approches sont possibles:
 - Création d'un objet littéral
 - Création d'un objet par prototypage
 - Création d'un objet à l'aide d'un constructeur

Création d'un objet littéral



- La programmation orientée objet consiste à écrire des programmes en utilisant des **objets** qui représentent les éléments du domaine étudié.
- En JavaScript, un objet est constitué d'un ensemble de **propriétés**, définies à l'intérieur d'une paire d'accolades : `{ ... }`;
- Chaque propriété possède un nom et une valeur, et est séparée des autres par une virgule. Une propriété peut être vue comme une sorte de variable attachée à un objet. □
- Lorsque la valeur d'une propriété est une **fonction**, on dit que cette propriété est une **méthode** de l'objet.
- **NB: Le mot-clé *this*** est défini automatiquement par JavaScript à l'intérieur d'une méthode et représente **l'objet sur lequel la méthode a été appelée**.

Création d'un objet littéral



Voici la syntaxe générale de création et d'utilisation d'un objet.

```
var monObjet = {  
  propriete1: valeur1,  
  propriete2: valeur2,  
  // ... ,  
  methode1: function(/* ... */) {  
    // ...  
  },  
  methode2: function(/* ... */) {  
    // ...  
  },  
  // ...  
};
```

Création d'un objet littéral- Exemple



```
var voiture = {  
  couleur: "bleu",  
  marque: "Peugeot",  
  type: "206",
```

```
//Ajout d'une méthode à un objet
```

```
  // Renvoie la description de l'objet
```

```
  decrire: function () {
```

```
    var description = "La couleur de la voiture est: " + this.couleur + "  
    Sa marque est " + this.marque + ". Elle est de type " + this.type;
```

```
    return description;  
  }
```

```
};
```

Création d'un objet littéral- Exemple



//Utilisation d'un objet

```
document.write(voiture.couleur+"<br/>");  
document.write(voiture.marque+"<br/>");  
document.write(voiture.type+"<br/>");
```

//Modification d'un objet

```
voiture.couleur="rouge"  
document.write(voiture.couleur+"<br/>");
```

//Ajout d'une nouvelle propriété

```
voiture.prix=23000;  
document.write(voiture.prix+"<br/>");
```

//Appel d'une méthode de l'objet

```
document.write(voiture.decrire()+"<br/>");
```

```
bleu  
Peugeot  
206  
rouge  
23000
```

La couleur de la voiture est: rouge. Sa marque est Peugeot. Elle est de type 206

Exercice- Modélisation d'un compte bancaire



Ecrivez un programme `compte.js` qui crée un objet compte ayant les propriétés suivantes :

- Une propriété titulaire valant "ISG".
- Une propriété solde valant initialement 0.
- Une méthode `crediter()` ajoutant le montant passé en paramètre au solde du compte.
- Une méthode `debiter()` retirant le montant passé en paramètre du solde du compte.
- Une méthode `decrire()` renvoyant la description du compte.
- Utilisez cet objet pour afficher sa description, le créditer puis le débiter de montants saisis successivement par l'utilisateur, puis afficher de nouveau sa description.

Exercice- Modélisation d'un compte bancaire



```
var compte = {  
  titulaire: "ISG",  
  solde: 0,  
  
  // Crédite le compte d'un certain montant  
  crediter: function (montant) {  
    this.solde = this.solde + montant;  
  },  
  // Débite le compte d'un certain montant  
  debiter: function (montant) {  
    this.solde = this.solde - montant;  
  },  
  // Renvoie la description du compte  
  decrire: function () {  
    var description = "Titulaire : " + this.titulaire +  
      ", solde : " + this.solde + " DT";  
    return description;  
  }  
};
```

Exercice- Modélisation d'un compte bancaire



```
alert(compte.decrire());  
var credit = Number(prompt("Entrez le montant à créditer :"));  
compte.crediter(credit);  
var debit = Number(prompt("Entrez le montant à débiter :"));  
compte.debiter(debit);  
alert(compte.decrire());
```

Prototypage



Pour créer des modèles d'objets en JavaScript, on utilise les prototypes. En plus de ses propriétés particulières, tout objet JavaScript possède une propriété interne appelée **prototype**.

Prototypage- Exemple



```
var ModeleVoiture = {  
  couleur: "",  
  marque: "",  
  type: "",  
  decrire: function () {  
    var description = "La couleur de la voiture est: " + this.couleur + ".  
    Sa marque est " + this.marque + ". Elle est de type " + this.type;  
    return description;  
  }  
};
```

Prototypage- Exemple



```
var voiture1 = Object.create(ModeleVoiture);  
voiture1.couleur = "grise";  
voiture1.marque = "Peugeot";  
voiture1.type = "206";
```

```
var voiture2 = Object.create(ModeleVoiture);  
voiture2.couleur = "noire";  
voiture2.marque = "Audi";  
voiture2.type = "Q5";
```

```
document.write(voiture1.decrire()+"<br/>");  
document.write(voiture2.decrire()+"<br/>");
```

La couleur de la voiture est: grise. Sa marque est Peugeot. Elle est de type 206
La couleur de la voiture est: noire. Sa marque est Audi. Elle est de type Q5

Initialisation



- Ajouter au modèle de voiture, la méthode **init()** qui prend en paramètres les valeurs initiales des propriétés d'une voiture, et définit les propriétés associées.
- A l'intérieur de cette méthode, il faut bien distinguer les propriétés (préfixées par le mot-clé **this**) des paramètres (non préfixés). Par exemple, **this.couleur** représente la propriété **couleur** de l'objet et **couleur** correspond à l'un des paramètres de la méthode.

```
var ModeleVoiture = {  
  couleur: "",  
  marque: "",  
  type: "",  
  init: function (couleur, marque, type) {  
    this.couleur = couleur;  
    this.marque = marque;  
    this.type = type;  
  },  
  decrire: function () {  
    var description = "La couleur de la voiture est: " + this.couleur + ". Sa marque est " + this.marque + ".  
    Elle est de type " + this.type;  
    return description;  
  }  
};
```


Initialisation



```
var voiture3 = Object.create(ModeleVoiture);  
voiture3.init("verte","Renault","ZX");  
document.write(voiture3.decrire()+"<br/>");
```

La couleur de la voiture est: verte. Sa marque est Renault. Elle est de type ZX

Création d'un objet à l'aide d'un constructeur



- Un constructeur est une fonction particulière dont le rôle est d'initialiser un nouvel objet. Son nom commence souvent par une lettre majuscule
- La création de l'objet à partir du constructeur est appelée **l'instanciation**. Elle s'effectue à l'aide du mot-clé **new**. □

// Constructeur MonObjet

function MonObjet() {

// Initialisation de l'objet

// ...

}

// Instanciation d'un objet à partir du constructeur

var monObj = new MonObjet();

Création d'un objet à l'aide d'un constructeur: Exemple

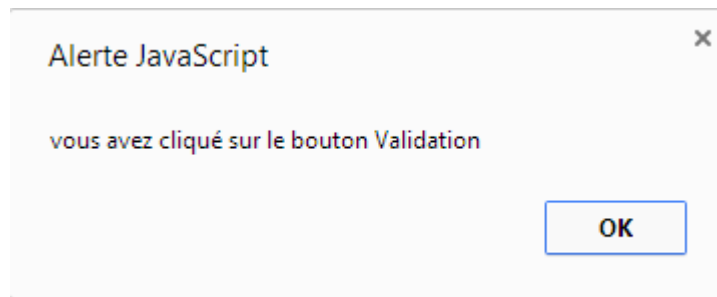


```
<!DOCTYPE html>
<html>
<head>
<title>Création d'un objet JS</title>
<script >
//fonction constructeur
function Couleur(R, G, B) {
    this.R = R; this.G = G; this.B = B; this.hex="#";
}
function changer_arriere_plan() {
    var arriere_plan = new Couleur("E0","FF","E0");
    document.bgColor = arriere_plan.hex + arriere_plan.R + arriere_plan.G +
    arriere_plan.B;
}
</script>
</head>
<body>
    <h1>Son propre objet couleur avec JavaScript</h1>
    <p><a href="javascript:changer_arriere_plan()">changer la couleur d'arrière-
    plan</a> </p>
</body>
</html>
```

Se référer à l'objet actuel (this)



```
<form name="formulaire">  
  <input type="button" value="Validation"  
    onClick="alert('vous avez cliqué sur le bouton ' + this.value)" />  
</form>
```



Le mot clé with



- Le mot clé **with** évite l'emploi répété du nom d'un objet pour accéder à ses propriétés et à ses méthodes.

```
<script>
with (window.navigator) {
    document.write("userAgent: " + userAgent + "<br />");
    document.write("platform: " + platform + "<br />");
}
</script >
```

```
userAgent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36
platform: Win32
```

DOM, Document Object Model



- Une page web est un document structuré contenant à la fois du texte et des balises HTML. Grâce à JavaScript, on peut accéder à la structure d'une page affichée dans un navigateur, et même la modifier.
- La représentation de la structure d'une page web offerte par un navigateur et exploitable via JavaScript est appelée **DOM**, pour **Document Object Model**.
- Le DOM représente une page web comme une hiérarchie d'objets reflétant sa structure. Chaque objet du DOM correspond à un nœud dans l'arborescence de la page web.
- La variable JavaScript **document** permet d'accéder à la racine de l'arborescence du DOM et correspond à l'élément `<html>` de la page.
- Les objets du DOM disposent de propriétés et de méthodes utilisables avec JavaScript. Parmi ces propriétés, **nodeType** renvoie le type de nœud, **childNodes** contient une collection de nœuds enfants et **parentNode** renvoie le nœud parent.
- **Consulter:** http://www.w3schools.com/jsref/dom_obj_document.asp

DOM, Document Object Model



- **Accéder au DOM avec la variable document**
- Lorsqu'un programme JavaScript s'exécute dans le contexte d'un navigateur web, il peut accéder à la racine du DOM en utilisant la variable document.
- **La variable document correspond à l'élément <html>.**
- Cette variable est un objet et dispose des propriétés **head** et **body** qui permettent d'accéder respectivement aux éléments <head> et <body> de la page. □
- **Exemple:**

var h = document.head; // La variable h contient l'objet head du DOM

document.write(h);

document.write("<hr/>");

var b = document.body; // La variable b contient l'objet body du DOM

document.write(b);

[object HTMLHeadElement]

[object HTMLBodyElement]

Découvrir le type d'un nœud



- Chaque objet du DOM a une propriété **nodeType** qui indique son type. La valeur de cette propriété est **document.ELEMENT_NODE** pour un nœud "élément" (balise HTML) et **document.TEXT_NODE** pour un nœud textuel.
- Les nœuds de type élément peuvent avoir des sous-nœuds (appelés enfants).

- **Exemple:**

```
if (document.body.nodeType === document.ELEMENT_NODE) {  
    document.write("Body est un noeud élément");  
} else {  
    document.write("Body est un noeud textuel");  
}
```

Body est un noeud élément

<!DOCTYPE html> Structure de la page Web

<html>

<head>

<meta charset="utf-8">

<title>Introduction à JavaScript</title>

<link rel="stylesheet" href="../css/style.css"/>

</head>

<body>

<h1 class="Titre" id="TitrePage">Ma première page web</h1>

<h2 class="Titre">Ma première section</h2>

<p>Mon premier paragraphe

Site de l'ESEN</p>

<div id="contenu">

<h2 class="Titre">Ma deuxième section</h2>

<p id="second" class="Rouge second">Mon second paragraphe</p>

</div>

<script>

```
for (var i = 0; i < document.body.childNodes.length; i++) {  
    document.write(document.body.childNodes[i]);
```

```
} </script> </body> </html>
```

childNodes



- Chaque objet du DOM de type élément possède une propriété **childNodes**. Il s'agit d'une collection ordonnée regroupant tous ses nœuds enfants sous la forme d'objets DOM. On peut utiliser cette collection un peu comme un tableau pour accéder aux différents enfants d'un nœud.
- NB: Le premier enfant du nœud **body** est un nœud textuel. Les espaces entre les balises ainsi que les retours à la ligne dans le code HTML sont considérés par le navigateur comme des nœuds textuels.
- **document.write(document.body.childNodes[0]); =>[object Text]**
- **document.write(document.body.childNodes[1]); =>[object HTMLHeadingElement]**

```
for (var i = 0; i < document.body.childNodes.length; i++) {  
    document.write(document.body.childNodes[i]);  
}
```

```
=>[object Text][object HTMLHeadingElement][object Text][object  
HTMLHeadingElement][object Text][object  
HTMLParagraphElement][object Text][object HTMLDivElement][object  
Text][object HTMLScriptElement][object Text]
```

Accéder au parent d'un nœud



- Chaque objet du DOM possède une propriété **parentNode** qui renvoie son nœud parent sous la forme d'un objet DOM.
- **Pour le nœud racine du DOM (la variable document), la valeur de parentNode est null: document n'a aucun nœud parent.**

```
var h1 = document.body.childNodes[1];  
document.write(h1.parentNode);  
    // Affiche [object HTMLBodyElement]  
document.write(document.parentNode);  
    // Affiche null : document n'a aucun nœud parent
```

Les limites du parcours du DOM nœud par nœud et solutions



- Le code de sélection devient difficile à lire lorsque la structure de la page se complexifie. De plus, il risque de devenir erroné en cas de modification ultérieure de la structure. Les nœuds textuels vides associés aux espaces et aux retours à la ligne rendent difficile l'écriture du code.

Solutions:

- Sélection d'éléments selon leur balise:

Exp: **document.getElementsByTagName("h2")**

```
var titresElts = document.getElementsByTagName("h2");
```

```
console.log(titresElts[0]);
```

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

```
console.log(titresElts.length);
```

```
<h2 class="Titre">Ma première section</h2>
```

```
<h2 class="Titre">Ma deuxième section</h2>
```

```
2
```

- Sélection d'éléments selon leur classe:

Exp: **document.getElementsByClassName("Titre")**

```
var TitreElts = document.getElementsByClassName("Titre");
```

```
for (var i = 0; i < TitreElts.length; i++) {
```

```
    console.log(TitreElts[i]);
```

```
}
```

```
<h1 class="Titre" id="TitrePage">Ma première page web</h1>
```

```
<h2 class="Titre">Ma première section</h2>
```

```
<h2 class="Titre">Ma deuxième section</h2>
```

Les limites du parcours du DOM nœud par nœud et solutions



- Sélection d'un élément selon son identifiant

Exp: `console.log(document.getElementById('TitrePage'));`

```
<h1 class="Titre" id="TitrePage">Ma première page web</h1>
```

- Sélection d'éléments à partir d'un sélecteur CSS
- **querySelectorAll**, permet de rechercher des éléments à partir d'un sélecteur CSS.

```
console.log(document.querySelectorAll('p').length); // Affiche 2
```

```
console.log(document.querySelectorAll('#contenu p').length); // Affiche 1
```

```
console.log(document.querySelectorAll('.Titre').length); // Affiche 3
```

- **querySelector**. renvoie uniquement le premier élément correspondant au sélecteur passé en paramètre. □

```
console.log(document.querySelector('p'));
```

```
<p>
  "Mon premier paragraphe "
  <a href="http://esen.tn" target="_blank">Site de l'ESEN</a>
</p>
```

Obtenir des informations sur les éléments



- **Le contenu HTML**

La propriété **innerHTML** permet de récupérer tout le contenu HTML d'un élément du DOM.

```
console.log(document.getElementById('contenu').innerHTML);
```

```
<h2 class="Titre">Ma deuxième section</h2>
<p id="second" class="Rouge second">Mon second paragraphe</p>
```

- **Le contenu textuel**

La propriété **textContent** renvoie tout le contenu textuel d'un élément du DOM, sans le balisage HTML □

```
console.log(document.getElementById('contenu').textContent);
```

```
Ma deuxième section
Mon second paragraphe
```


Les attributs



- La méthode **getAttribute** renvoie la valeur de l'attribut passé en paramètre. □

```
console.log(document.querySelector("a").getAttribute("href"));
```

Affiche <http://esen.tn>

- **Nb:** Certains attributs sont directement accessibles sous la forme de propriétés. C'est le cas des attributs **id**, **href** et **value**.

```
console.log(document.querySelector("div").id);
```

// L'identifiant de la première division, affiche contenu

```
console.log(document.querySelector("a").href);
```

// L'attribut href du premier lien, affiche <http://esen.tn/>

- On peut vérifier la présence d'un attribut sur un élément grâce à la méthode **hasAttribute**, comme dans l'exemple ci-après.

```
if (document.querySelector("a").hasAttribute("target")) {
```

```
    console.log("Le premier lien possède l'attribut target");
```

```
} else {    console.log("Le premier lien ne possède pas l'attribut target");}
```

//Affiche Le premier lien possède l'attribut target

Les classes



- Dans une page web, une balise peut posséder plusieurs classes. La propriété **classList** permet de récupérer la liste des classes d'un élément du DOM. Elle s'utilise comme un tableau.

```
var classes = document.getElementById("second").classList;  
console.log(classes.length); // Affiche 2 : l'élément possède deux classes  
console.log(classes[0]); // Affiche "Rouge"
```

- Pour tester la présence d'une classe dans un élément en appelant la méthode **contains** sur la liste des classes. □□

```
if (document.getElementById("second").classList.contains("second")) {  
    console.log("L'élément identifié par second possède la classe second");  
} else {  
    console.log("L'élément identifié par second ne possède pas la classe  
    second");  
}
```



Autres Exemples

La propriété innerHTML



- La propriété **innerHTML** permet de sauvegarder le contenu d'un élément HTML.

```
<p id="Test"  
onmouseover="this.innerHTML='<em>Vous voyez?</em>'"  
onmouseout="this.innerHTML='<strong>Je suis  
dynamique</strong>'">Je suis dynamique  
</p>
```

Je suis dynamique

Vous voyez?

Je suis dynamique

Méthode : `window.document.getElementById()`



- Retourne un objet HTML à partir de son id

```
<form name="formulaire1">
```

```
<input type="text" id="champ_input" placeholder="Saisir une valeur" />  
<br/>
```

```
<input type="button" onclick="f('champ_input')" value="Vérifier" />
```

```
<script>
```

```
</form>
```

```
function f(identifiant)
```

```
{
```

```
    var obj = document.getElementById(identifiant);
```

```
    alert('le champ a pour valeur : '"+obj.value+'");
```

```
    obj.value='autre valeur';
```

```
    alert('maintenant il contient : '"+obj.value+'");
```

```
}
```

```
</script>
```

Méthode : window.document.getElementsByTagName



```
<script>
```

```
function check() {
```

```
    var inputs = document.getElementsByTagName('input');
```

```
    inputsLength = inputs.length;
```

```
    for (var i = 0 ; i < inputsLength ; i++) {
```

```
        if (inputs[i].type == 'radio' && inputs[i].checked) {
```

```
            alert('La case cochée est la n°'+ inputs[i].value); } } }
```

```
</script>
```

```
<form name="FormChec">
```

```
<input type= "radio" name="chec" value="1" id="ch1"/> <label  
    for="ch1">Case n°1</label><br />
```

```
<input type= "radio" name="chec" value="2" id="ch2"/> <label  
    for="ch2">Case n°2</label><br />
```

```
<input type= "radio" name="chec" value="3" id="ch3"/> <label  
    for="ch3">Case n°3</label><br />
```

```
<input type= "radio" name="chec" value="4" id="ch4"/> <label  
    for="ch4">Case n°4</label>
```

```
<br /><br />
```

```
<input type="button" value="Afficher la case cochée" onclick="check();" />
```

```
</form>
```

Exercice: Modifier le code suivant. Le résultat demandé est affiché dans les interfaces suivantes.

```
<script>
function check() {
    var inputs = document.getElementsByTagName('input');
    inputsLength = inputs.length;
    for (var i = 0 ; i < inputsLength ; i++) {
        if (inputs[i].type == 'radio' && inputs[i].checked) {
            alert('La case cochée est la n°'+ inputs[i].value); } } }
</script>

<form name="FormChec">
<input type="radio" name="chec" value="1" id="ch1"/> <label for="ch1">Case
    n°1</label><br />
<input type="radio" name="chec" value="2" id="ch2"/> <label for="ch2">Case
    n°2</label><br />
<input type="radio" name="chec" value="3" id="ch3"/> <label for="ch3">Case
    n°3</label><br />
<input type="radio" name="chec" value="4" id="ch4"/> <label for="ch4">Case
    n°4</label>
<br /><br />
<input type="button" value="Afficher la case cochée" onclick="check();" />
</form>
```

Résultat demandé



- ☐ Case n°1
- ☐ Case n°2
- ☐ Case n°3
- ☐ Case n°4

Afficher les cases cochées

Aucune case n'est cochée

☐ Empêcher cette page de générer des boîtes de dialogue supplémentaires

OK

- ☐ Case n°1
- ☒ Case n°2
- ☒ Case n°3
- ☒ Case n°4

Afficher les cases cochées

Les cases cochées sont les cases n°:

2
3
4

☐ Empêcher cette page de générer des boîtes de dialogue supplémentaires

OK

Correction- Exercice du cours



```
<script>
function check2() {
    var inputs = document.getElementsByTagName('input');
    inputsLength = inputs.length;
    resultat="";
    for (var i = 0 ; i < inputsLength ; i++) {
        if (inputs[i].type == 'checkbox' && inputs[i].checked) {
            resultat+=inputs[i].value +"\n";
        }
    }
    if (resultat=="")
        alert('Aucune case n'est cochée');
    else alert('Les cases cochées sont les cases n°:\n'+ resultat);
}
</script>
```