

# Réalisation de systèmes interactifs à manipulation directe en JavaScript

Ingénierie des systèmes interactifs  
L2 MIASHS Parcours informatique (UE MIB0405V)





# Tableaux

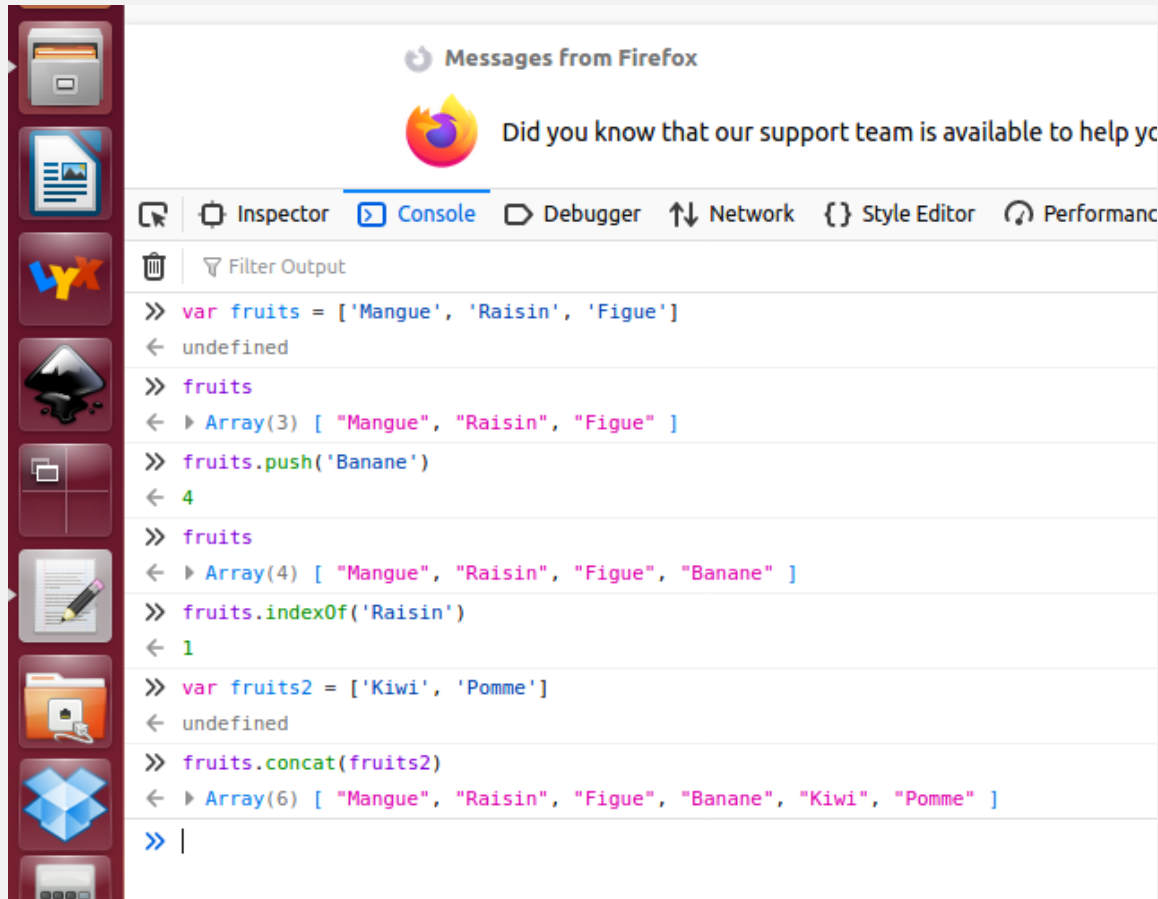
```
a = [1,2,3,5,8,13];  
for ( i=0; i < a.length ; i+=1 ) {  
    console.log (a[i]) ;  
} // affiche tous les éléments du tableau
```



```
>> var fruits = [ 'Mangue', 'Raisin', 'Figue' ];  
← undefined  
>> fruits.push('Banane')  
← 4  
>> fruits  
← ▶ Array(4) [ "Mangue", "Raisin", "Figue", "Banane" ]  
>> fruits.length  
← 4  
>> |
```

- en JavaScript, les tableaux sont considérés comme une forme particulière du type object (*typeof [] retourne "object"*)
- Comme toute variable en JavaScript, les valeurs stockées dans un tableau peuvent être **de n'importe quel type** (y compris un tableau), et **n'ont pas besoin d'être tous du même type**
- Fonctions avancées : *[ 'Mangue', 'Raisin' ].concat([ 'Figue', 'Kiwi' ])*  
*fruits.indexOf('Raisin')*  
*fruits.concat(fruits1)*

# Tableaux



The screenshot shows the Firefox Developer Console with the 'Console' tab selected. The console displays a series of JavaScript commands and their outputs, demonstrating array operations. The commands and outputs are as follows:

```
>> var fruits = ['Mangue', 'Raisin', 'Figue']  
← undefined  
>> fruits  
← ▶ Array(3) [ "Mangue", "Raisin", "Figue" ]  
>> fruits.push('Banane')  
← 4  
>> fruits  
← ▶ Array(4) [ "Mangue", "Raisin", "Figue", "Banane" ]  
>> fruits.indexOf('Raisin')  
← 1  
>> var fruits2 = ['Kiwi', 'Pomme']  
← undefined  
>> fruits.concat(fruits2)  
← ▶ Array(6) [ "Mangue", "Raisin", "Figue", "Banane", "Kiwi", "Pomme" ]  
>> |
```

- Plus sur les tableaux en JS :  
(length, sort, etc)

[https://www.w3schools.com/js/js\\_arrays.asp](https://www.w3schools.com/js/js_arrays.asp)

# Fonctions

```
function nomDeLaFonction ( param1, param2, param3 ...) {  
    // instructions javascript  
    return resultat;  
}
```

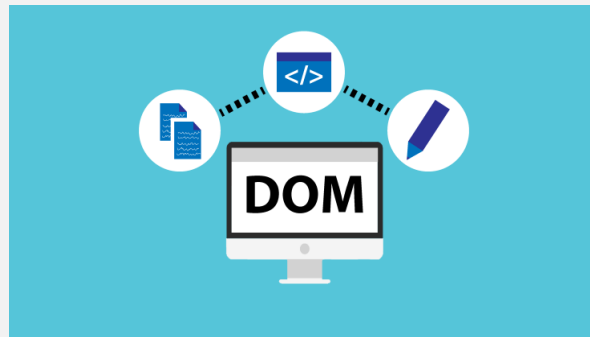
```
function multiplierParDeux (nombre) {  
    return nombre * 2;  
}
```

```
function estMultipleDeTrois (nombre) {  
    return nombre % 3 === 0;  
}
```

```
function estMultipleDeCinq(nombre) {  
    return nombre % 5 === 0;  
}
```

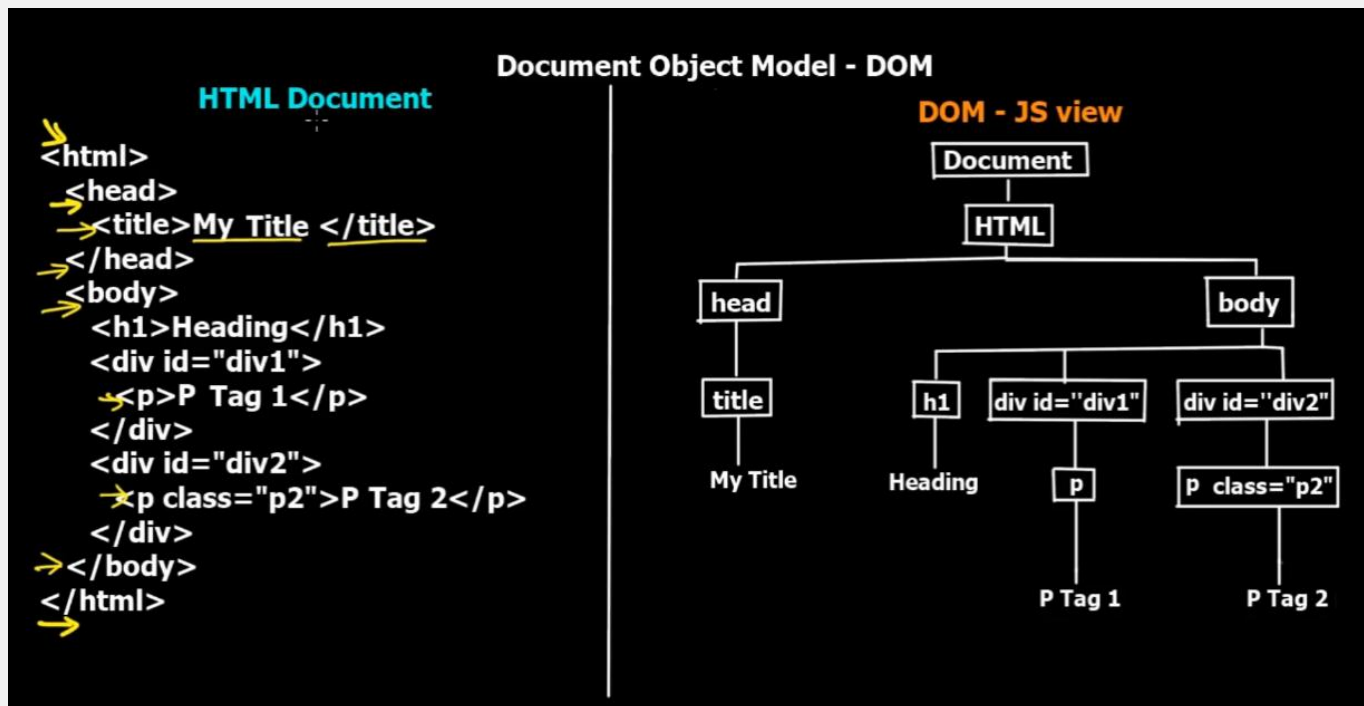
```
estMultipleDeTrois(2); // => retourne false, car 2 n'est pas multiple de 3  
var monNombre = 5;    // valeur fournie en guise d'exemple  
if ( estMultipleDeCinq(monNombre) ) {  
    console.log ('monNombre est multiple de 5');  
} else {  
    console.log ('monNombre n'est pas multiple de 5');  
}
```

# Manipuler le Web avec DOM



# DOM

- Lorsqu'une page web est chargée, le navigateur crée un **modèle d'objet de document** de la page (DOM : Document Object Model).
- Construit comme un arbre d'Objets :

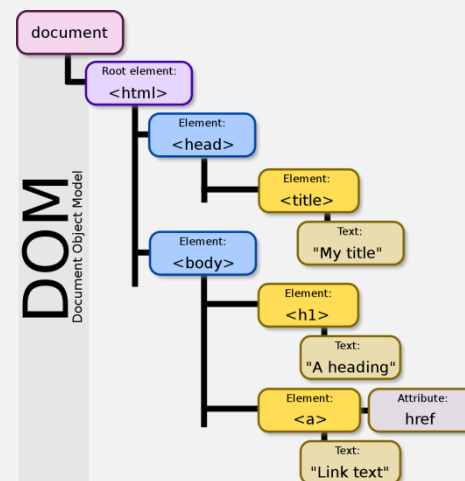


# DOM (Document Object Model)

DOM est une *interface standard* (neutre point de vue de la plate-forme et du langage) qui permet aux *programmes d'accéder dynamiquement* au contenu, à la structure et au style d'un document et de *les mettre à jour*.

w3schools.com

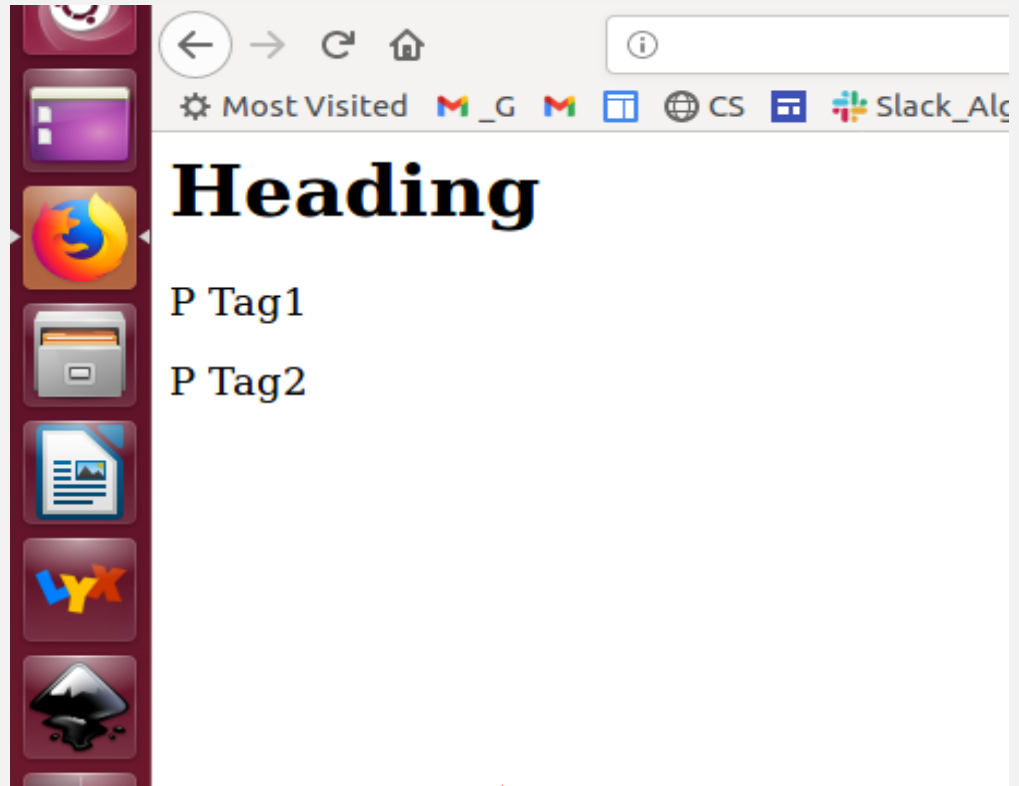
- La norme est divisée en trois parties différentes :
  - *Core DOM* - modèle standard pour tous les types de documents
  - *XML DOM* - modèle standard pour les documents XML
  - *HTML DOM* - modèle standard pour les documents HTML
- L'objectif est d'avoir la possibilité créer du HTML dynamique :
  - Modifier *tous les éléments HTML* (les attributs) de la page,
  - Modifier les *styles CSS* de la page
  - *Supprimer* les éléments et attributs HTML existants
  - *Ajouter de nouveaux éléments* et attributs HTML
  - *Réagir* à tous les événements HTML existants dans la page
  - Créer de *nouveaux événements* HTML dans la page





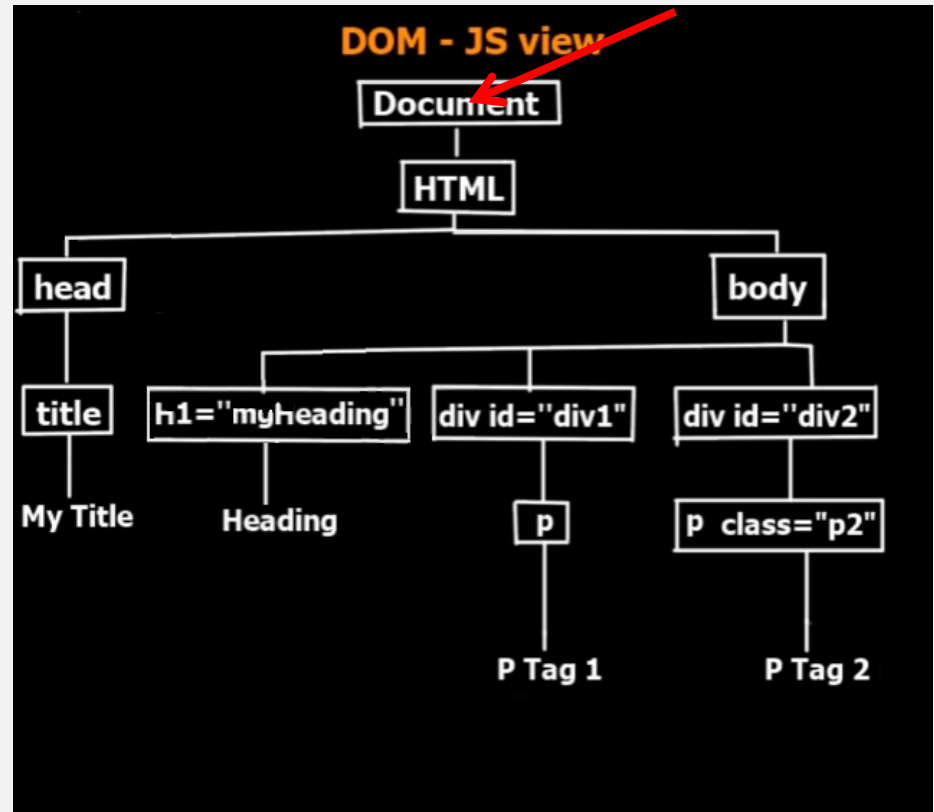
# DOM

```
<html>
  <head>
    <title>My Title </title>
  </head>
  <body>
    <h1 id="myheading">
      Heading
    </h1>
    <div id="div1">
      <p> P Tag1</p>
    </div>
    <div id="div2">
      <p classe="p2"> P Tag2</p>
    </div>
  </body>
</html>
```



# DOM

```
<html>
  <head>
    <title>My Title </title>
  </head>
  <body>
    <h1 id="myheading">
      Heading
    </h1>
    <div id="div1">
      <p> P Tag1</p>
    </div>
    <div id="div2">
      <p classe="p2"> P Tag2</p>
    </div>
  </body>
</html>
```



# accéder à un élément HTML

- La façon la plus courante d'accéder à un élément HTML est d'utiliser l'identifiant de l'élément.

```
...  
<h1 id="myheading"> Heading </h1>  
<div id="div1">  
...  

```

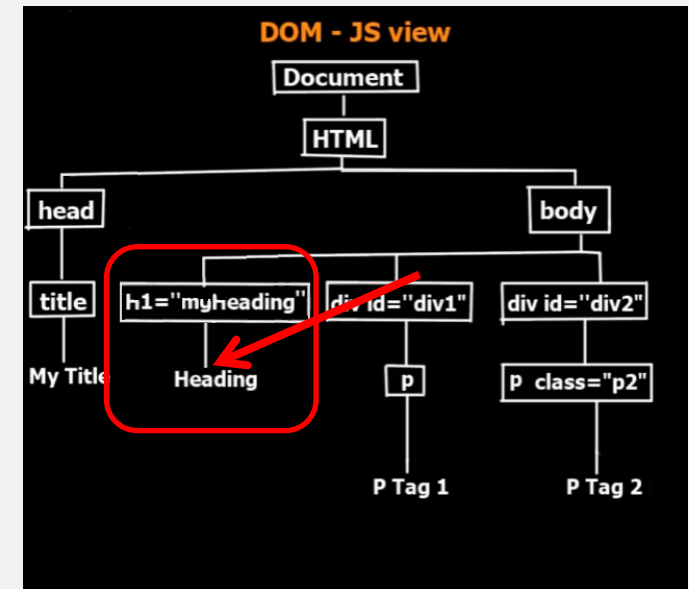
- Dans l'exemple ci-après, la méthode `getElementById` utilise `id="myheading"` pour trouver l'élément (*return object*).

```
document.getElementById('myheading');
```

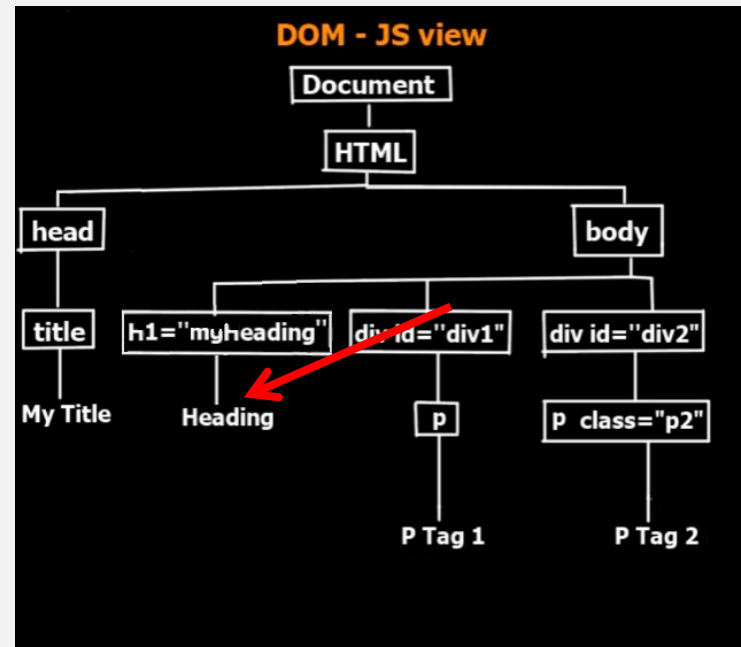
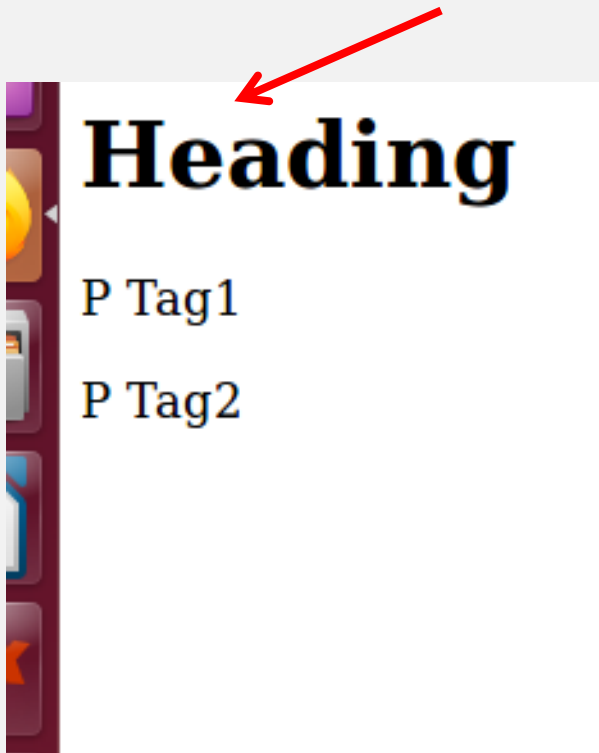
- La façon la plus simple d'obtenir le contenu d'un élément est d'utiliser la propriété `innerHTML` (utile pour obtenir mais aussi pour remplacer le contenu d'un éléments HTML).

```
document.getElementById('myheading').innerHTML();
```

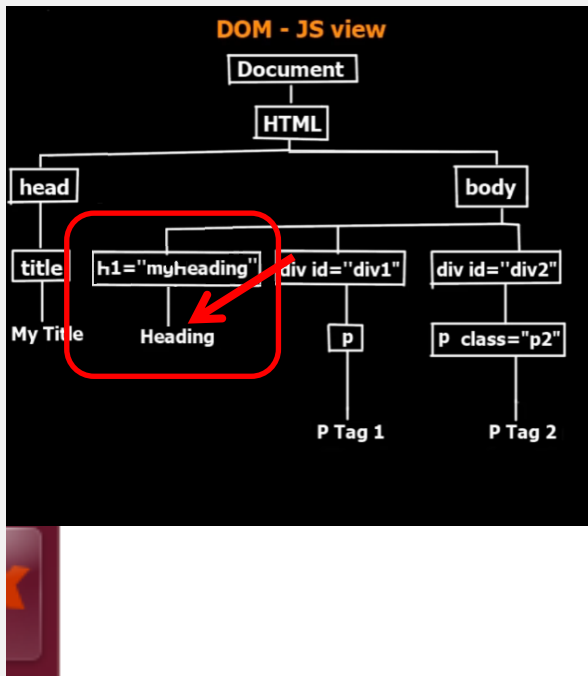
- manip ...



# getElementById



# getElementById



```
Inspector Console Debugger Style Editor
Filter output

>> document
< HTMLDocument file:///home/

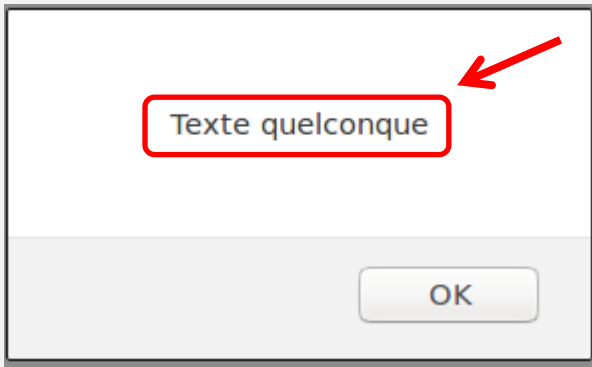
>> document.getElementById("myheading")
< <h1 id="myheading">

>> typeof(document.getElementById("myheading"))
< "object"

>> document.getElementById("myheading").innerHTML
< " Heading "

>> typeof(document.getElementById("myheading").innerHTML)
< "string"
```

# getElementById (manip 2)



```
<!DOCTYPE html>
<html>
  <head>
    <title> getElementById </title>
  </head>
  <body>
    <button>Cliquez ici</button>
    <h2 id="myheading">Texte quelconque</h2>
  </body>
</html>
```

# getElementById (manip 2)

```
<!DOCTYPE html>
<html>
<head>
  <title>getElementById</title>
</head>
<body>
  <script>
    function cliqueBouton (even) {
      var str = document.getElementById("myheading").innerHTML;
      alert(str);
      console.log(str);
    }
  </script>
  <button onclick="cliqueBouton()">Cliquez ici</button>
  <h2 id="myheading">Texte quelconque</h2>
</body>
</html>
```

## Plus d'exercices ... *innerHTML*

**Objectif :** Manipulation de boucle et de *innerHTML*

**Énoncé :**

- Écrivez une fonction JavaScript qui *crée un tableau*,
- acceptez les *numéros de ligne* et de *colonne* de l'utilisateur et
- entrez le numéro de ligne et de colonne comme contenu (par exemple, *Ligne-0 Colonne-0*) d'une cellule.

Ligne-0 Colonne-0	Ligne-0 Colonne-1
Ligne-1 Colonne-0	Ligne-1 Colonne-1

Créer le tableau



## Plus d'exercices ... *innerHTML*

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8 />
  <title>Créer un tableau</title>
  <style type="text/css">
    body {margin: 30px;}
  </style>
</head>
<body>
  <table id="maTable" border="1"></table>
  <form>
    <input type="button" onclick="creerTable()" value="Créer le tableau">
  </form>
</body>
</html>
```

## Plus d'exercices ... *Math et formulaire*

**Objectif :** Utilisation de Javascript dans un *formulaire*.

**Énoncé :** Écrivez un programme JavaScript pour *calculer le volume d'une sphère*.

Entrez la valeur du rayon et obtenez le volume d'une sphère.

Rayon

Volume

## Plus d'exercices ... *Math et formulaire*

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Volume d'une sphère.</title>
    <style>
      body{padding-top:30px;}
      label,input{display:block;}
    </style>
  </head>
  <body>
    <p>Entrez la valeur du rayon et obtenez le volume d'une sphère.</p>
    <form action="" method="post" id="MonFormulaire">
      <label for="rayon">Rayon</label>
      <input type="text" name="rayon" id="rayon" required>
      <label for="volume">Volume</label><input type="text" name="volume" id="volume">
      <input type="submit" value="Calculer" id="submit">
    </form>
  </body>
</html>
```

## Plus d'exercices ... Chaînes de caractères

**Objectif** : Manipuler les chaînes de caractères.

**Énoncé** :

1. Ecrire une fonction qui permet de **tester** si une chaîne de caractère passée en paramètre **commence** par une lettre majuscule ou minuscule *entre 'a' et 'd'*.
2. Ecrire une fonction qui renvoie **'true'** lorsque'une chaîne contient **un seul caractère '@'** sinon renvoie 'false'.
3. Ecrire une fonction qui renvoie **'true'** lorsque'une chaîne contient **au moins un chiffre** sinon renvoie 'false'.
4. Ecrire une fonction qui **remplace** les chiffre par le caractère **'\*'** .
5. **Colorer** un message en rouge si l'entrée est invalide et en vert sinon.

Bonjour	Tester
adresse@domaine.com	Tester
12test	Tester
at31400toulouse	Tester

## Plus d'exercices ... Chaînes de caractères

```
<!doctype html>
<html>
<head>
  <title>Manipuler les chaines de caractères</title>
  <meta charset="utf-8">
  <style type="text/css">
    div{
      border: 1px solid black;
    }
    .error{
      color: green;
    }
  </style>
</head>
```

```

<body>
  <div >
    <input type="text" value="Bonjour" id="ch1"/>
    <button onclick="str_firts_letter(document.getElementById('ch1').value)"> Tester
    </button>
    <p id="res1" class="error"></p>
  </div>
  <div >
    <input type="text" value="adresse@domaine.com" id="ch2"/>
    <button onclick="str_email(document.getElementById('ch2').value)">Tester</button>
    <p id="res2" class="error"></p>
  </div>
  <div >
    <input type="text" value="12test" id="ch3"/>
    <button onclick="str_number(document.getElementById('ch3').value)"> Tester
    </button>
    <p id="res3" class="error"></p>
  </div>
  <div >
    <input type="text" value="at31400toulouse" id="ch4"/>
    <button onclick="str_replace(document.getElementById('ch4').value)"> Tester
    </button>
    <p id="res4" class="error"></p>
  </div>
</body>

```

## *getElementsByTagName*

- La méthode *getElementsByTagName* permet de sélectionner les éléments portant un nom de balise donné dans une page.

*Énoncé :*

- Écrivez un programme JavaScript pour *mettre en évidence* les mots en gras du paragraphe suivant, *en passant la souris* sur le paragraphe. Définir les fonctions *mise\_en\_evidence()* et *retour\_normal()*.

En passant la souris sur le paragraphe suivant, les mots en gras seront mis en évidence:

**Nous** venons de commencer **cette** section pour les utilisateurs (**débutants** à intermédiaires) qui **veulent** travailler avec **divers problèmes** de JavaScript et écrire des scripts en ligne pour **tester** leurs **compétences** en JavaScript.

## getElementsByTagName

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Obtenir et styliser tous les tags</title>
</head>
<body>
    <p>En passant la souris sur le paragraphe suivant, les mots en gras seront mis en
    évidence:</p>
    <p onMouseOver="mise_en_evidence()" onMouseOut="retour_normal()">
        <strong>Nous</strong> venons de commencer <strong>cette</strong> section pour les
        utilisateurs (<strong>débutants</strong> à intermédiaires) qui <strong>veulent</strong>
        travailler avec <strong>divers problèmes</strong> de JavaScript et écrire des scripts en ligne
        pour <strong>tester</strong> leurs <strong>compétences</strong> en JavaScript.</p>
</body>
</html>
```

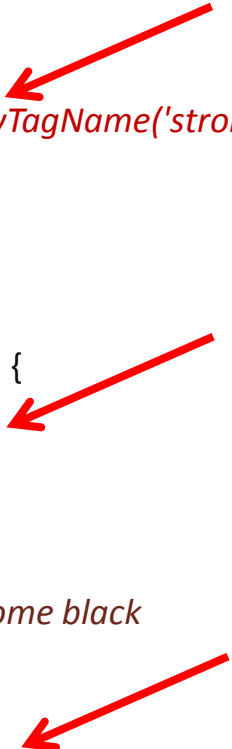


```
<script type="text/javascript">
  //First create a list of all bold items
  var bold_Items;
  window.onload = getBold_items();

  // Collect all <strong> tags
  function getBold_items() {
    bold_Items = document.getElementsByTagName('strong');
  }

  // iterate all bold tags and change color
  function mise_en_evidence() {
    for (var i=0; i<bold_Items.length; i++) {
      bold_Items[i].style.color = "green";
    }
  }

  // On mouse out highlighted words become black
  function retour_normal(){
    for (var i=0; i<bold_Items.length; i++) {
      bold_Items[i].style.color = "black";
    }
  }
}
</script>
```



# accéder à un élément HTML

## ○ Accès direct

- `getElementById` (*déjà vu*)
- `getElementsByTagName` (*déjà vu*)
- `document.getElementsByClassName(<classe>)`
- `document.querySelector()`

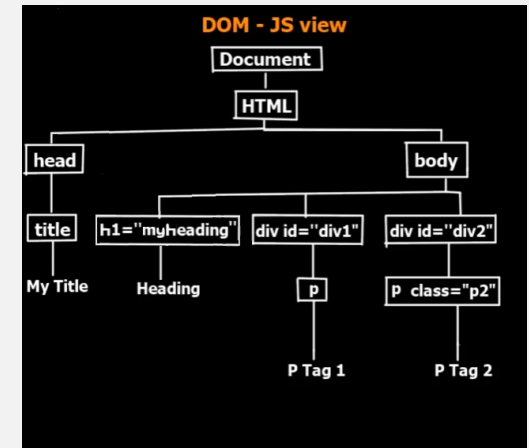
## ○ Accès relatif

- Accès aux nœuds ancêtres : *parentNode*
- Accès aux nœuds enfants : *elt.childNodes*, *elt.firstChild*, *elt.lastChild*, *elt.lastElementChild*
- Accès aux nœuds-frères : *previousSibling*, *nextSibling*

## ○ Accès par les collections


- Quatre collections donnent accès à différents éléments du document : *document.forms*, *document.images*, *document.links* et *window.frames*.

## ○ Plus d'exemples ... (1), (2)



(1) <https://www.gchagnon.fr/cours/dhtml/introdom.html#directdocument>

(2) <https://openclassrooms.com/fr/courses/5543061-ecrire-du-javascript-pour-le-web/5577476-accédez-aux-éléments-du-dom>

> typeof NaN	> true==1
< "number"	< true
> 9999999999999999	> true===1
< 10000000000000000	< false
> 0.5+0.1==0.6	> (!+[[]]+[![]]).length
< true	< 9
> 0.1+0.2==0.3	> 9+"1"
< false	< "91"
> Math.max()	> 91-"1"
< -Infinity	< 90
> Math.min()	> []==0
< Infinity	< true
> []+[]	
< ""	
> []+{}	
< "[object Object]"	
> {}+[]	
< 0	
> true+true+true===3	
< true	
> true-true	
< 0	

