JavaScript

mariaiuliana.dascalu@gmail.com

Définition

- Le Javascript est un langage de programmation de scripts orienté objet.
  - Un **langage de programmation** est un langage qui permet aux développeurs d'écrire du **code source** qui sera analysé par l'ordinateur.
  - Le Javascript permet de programmer des **scripts** =>un langage interprété
    - Chaque navigateur possède un interpréteur Javascript, qui diffère selon le navigateur. Si vous utilisez Internet Explorer, son interpréteur Javascript s'appelle *JScript* (l'interpréteur de la version 9 s'appelle *Chakra*), celui de Mozilla Firefox se nomme *SpiderMonkey* et celui de Google Chrome est *V8*.
  - Un langage de programmation **orienté objet** est un langage qui contient des éléments, appelés **objets**, et que ces différents objets possèdent des caractéristiques spécifiques ainsi que des manières différentes de les utiliser.

Utile pour

- ajouter de l'interactivité à vos pages Web (e.g. afficher/masquer du texte, faire défiler des images ,….)

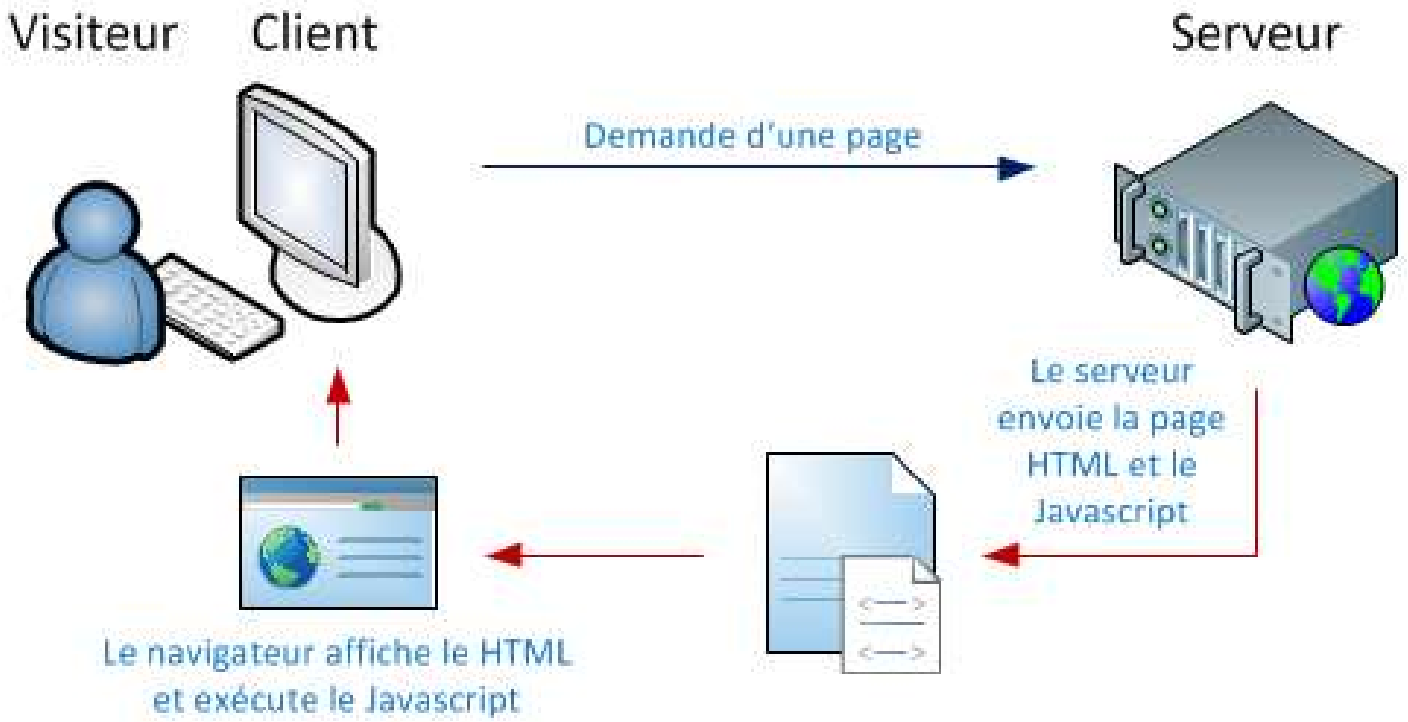
- détection des navigateurs (les mêmes éléments sont affichées de différentes manières, en fonction du type / version du navigateur)

- ajouter des cookies(= stocker des informations sur l'ordinateur du visiteur, puis récupérer ces informations automatiquement la prochaine fois que l'utilisateur visite votre page)

- valider les formulaires (e.g.: vérifiez si l'adresse e-mail indiquée a un @ en elle, parce que sinon, ce n'est pas une adresse valide)

Un langage client-side

- Le Javascript est un langage dit ***client-side***, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le **client**).



Petite historique

Brendan Eich, le papa de Javascript

- En 1995, Brendan Eich travaille chez Netscape Communication Corporation, la société qui éditait le célèbre navigateur Netscape Navigator, alors principal concurrent d'Internet Explorer.

- Brendan développe le LiveScript, un langage de script qui s'inspire du langage Java , et qui est destiné à être installé sur les serveurs développés par Netscape.

- Netscape se met à développer une version client du LiveScript, qui sera renommée JavaScript en hommage au langage Java créé par la société Sun Microsystems.

- Mais attention, au final, **ces deux langages sont radicalement différents** !

Editeurs de texte à utiliser

- Notepad++ pour Windows
- Vim pour Linux
- TextWrangler pour Mac
- …….

Où le placer?

- vous aurez besoin de laisser le navigateur sait à l'avance quand vous entrez dans le langage Javascript, en utilisant la balise <script>
- **Exemple: la boîte de dialogue alert()**

```
<html>
    <head>
        <title>My Javascript Page</title>
        <script type="text/javascript">
            alert("Welcome!");
        </script>
    </head>
    <body>
        <div>TEST</div>
    </body>
</html>
```

- vous pouvez entrer le Javascript dans les deux sections du document (<head> and <body>).

Le Javascript externe

- Il est possible, et même conseillé, d'écrire le code Javascript dans un fichier externe, portant l'extension.js.

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Hello World!</title>
  </head>

  <body>

    <script src="hello.js"></script>

  </body>
</html>
```

On suppose ici que le fichier *hello.js* se trouve dans le même répertoire que la page Web.

La syntaxe du Javascript

- les instructions doivent être séparées par un point-virgule que l'on place à la fin de chaque instruction

- il y a des commentaires

javascript

```javascript
/* Ce script comporte 3 instructions :
      - Instruction 1 qui fait telle chose
      - Instruction 2 qui fait autre chose
      - Instruction 3 qui termine le script
*/
instruction_1;
instruction_2;
instruction_3; // Fin du script
```

Les variables

- le nom d'une variable ne peut contenir que des lettres de A à Z, a à z et les chiffres de 0 à 9 ; l'underscore **(_)** et le dollar **($)** sont aussi acceptés

- le nom de la variable ne peut pas commencer par un chiffre et ne peut pas être constitué uniquement de mots-clés utilisés par le Javascript

- vous trouverez sur cette page (en anglais) tous les mots réservés par le Javascript: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Reserved_Words

- Javascript est un langage typé *dynamiquement*: cela veut dire, généralement, que toute déclaration de variable se fait avec le mot-clé var sans distinction du contenu

- example: *var* myVariable;

Tester l'existence de variables avec l'instruction typeof

```javascript
var number = 2;
alert(typeof number); // Affiche : « number »

var text = 'Mon texte';
alert(typeof text); // Affiche : « string »

var aBoolean = false;
alert(typeof aBoolean); // Affiche : « boolean »
```

```javascript
alert(typeof nothing); // Affiche : « undefined »
```

Si l'instruction typeof vous renvoie undefined, c'est soit que votre variable est inexistante, soit qu'elle est déclarée mais ne contient rien.

Les opérateurs arithmétiques

Opérateur	Signe
addition	+
soustraction	-
multiplication	*
division	/
modulo	%

```
1 var number = 3;
2 number += 5;
3 alert(number); // Affiche : « 8 »
```

javascript

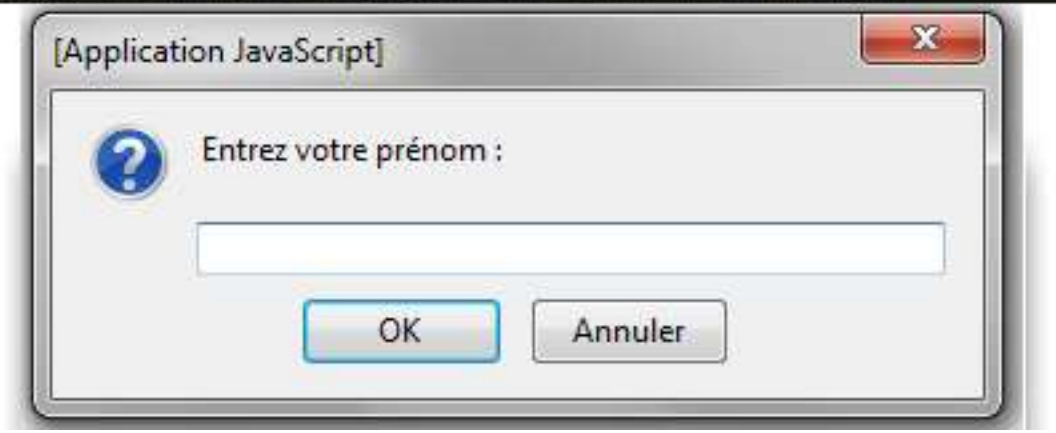
```
1 var divisor = 3, result1, result2, result3;
2
3 result1 = (16 + 8) / 2 - 2 ; // 10
4 result2 = result1 / divisor;
5 result3 = result1 % divisor;
6
7 alert(result2); // Résultat de la division : 3,33
8 alert(result3); // Reste de la division : 1
```

- La concaténation: une concaténation consiste à ajouter une chaîne de caractères à la fin d'une autre

```javascript
var text = 'Bonjour ';
text += 'toi';
alert(text); // Affiche « Bonjour toi ».
```

- Interagir avec l'utilisateur: la fonction prompt() s'utilise comme alert() mais a une petite particularité; elle renvoie ce que l'utilisateur a écrit sous forme d'une chaîne de caractères

```
var userName = prompt('Entrez votre prénom :');
alert(userName); // Affiche le prénom entré par l'utilisateur
```



- Convertir une chaîne de caractères en nombre

```javascript
var first, second, result;

first = prompt('Entrez le premier chiffre :');
second = prompt('Entrez le second chiffre :');
result = parseInt(first) + parseInt(second);

alert(result);
```

- Convertir un nombre en chaîne de caractères

```javascript
var text, number1 = 4, number2 = 2;
text = number1 + '' + number2;
alert(text); // Affiche : « 42 »
```

Les opérateurs

- Les opérateurs de comparaison: ==, !=,>,>=,<,<=
- Les opérateurs logiques: &&, ||, !

```javascript
var result = false;

result = !result; // On stocke dans « result » l'inverse de « result », c'est parfai
alert(result); // Affiche « true » car on voulait l'inverse de « false »

result = !result;
alert(result); // Affiche « false » car on a inversé de nouveau « result », on est d
```

Les conditions

- La condition « if else »

```javascript
if (2 < 8 && 8 >= 4) { // Cette condition renvoie « true », le code est donc exécuté
    alert('La condition est bien vérifiée.');
}

if (2 > 8 || 8 <= 4) { // Cette condition renvoie « false », le code n'est donc pas
    alert("La condition n'est pas vérifiée mais vous ne le saurez pas vu que ce code
}
```

```javascript
var floor = parseInt(prompt("Entrez l'étage où l'ascenseur doit se rendre (de -2 à 30)

if (floor == 0) {

    alert('Vous vous trouvez déjà au rez-de-chaussée.');

} else if (-2 <= floor && floor <= 30) {

    alert("Direction l'étage n°" + floor + ' !');

} else {

    alert("L'étage spécifié n'existe pas.");

}
```

switch

```
var drawer = parseInt(prompt('Choisissez le tiroir à ouvrir (1 à 4) :'));

switch (drawer) {
    case 1:
        alert('Contient divers outils pour dessiner : du papier, des crayons, etc.')
    break;

    case 2:
        alert('Contient du matériel informatique : des câbles, des composants, etc.'
    break;

    case 3:
        alert('Ah ? Ce tiroir est fermé à clé ! Dommage !');
    break;

    case 4:
        alert('Contient des vêtements : des chemises, des pantalons, etc.');
    break;

    default:
        alert("Info du jour : le meuble ne contient que 4 tiroirs et, jusqu'à preuve
}
```

Les ternaires

```
var startMessage = 'Votre catégorie : ',
    endMessage,
    adult = confirm('Êtes-vous majeur ?');

if (adult) { // La variable « adult » conti
    endMessage = '18+';
} else {
    endMessage = '-18';
}

alert(startMessage + endMessage);
```

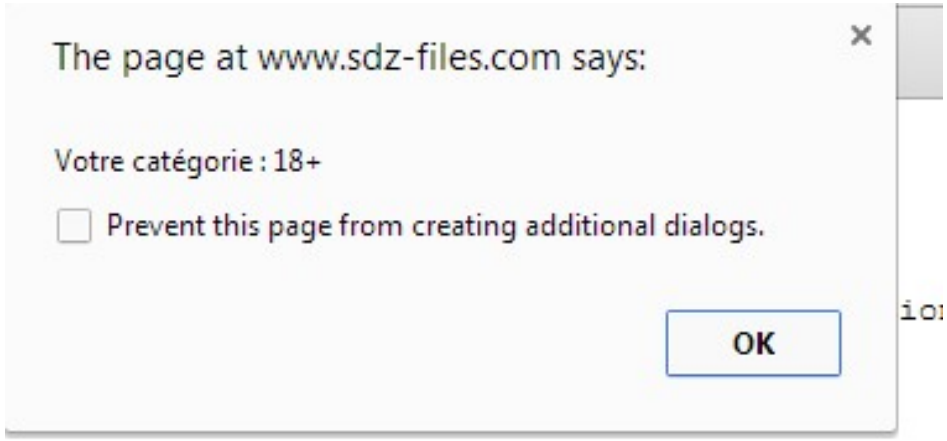
```
var startMessage = 'Votre catégorie : ',
    endMessage,
    adult = confirm('Êtes-vous majeur ?');

endMessage = adult ? '18+' : '-18';

alert(startMessage + endMessage);
```

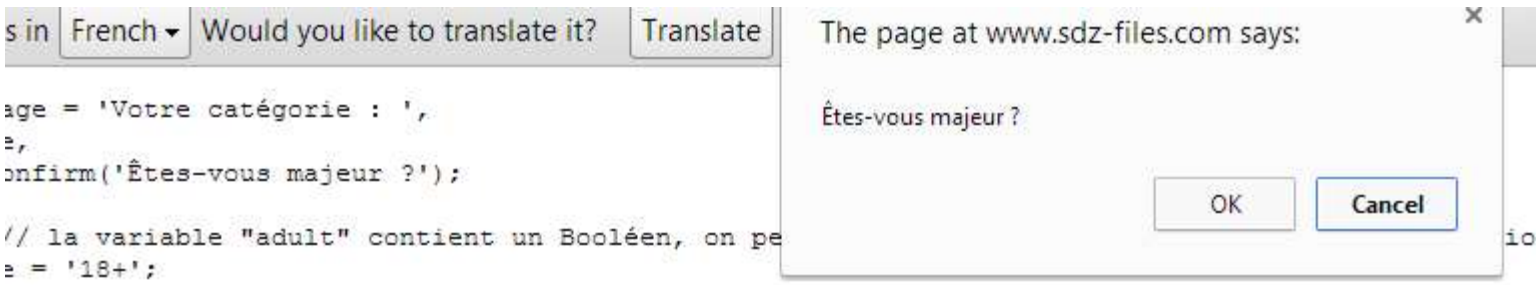
endMessage = adult ? '18+' : '-18';

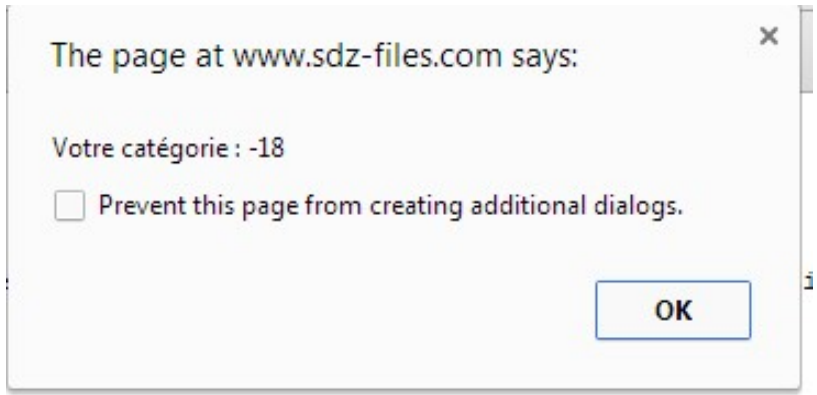
Si l'on décompose cette ligne on peut voir :

- La variable endMessage qui va accueillir le résultat de la ternaire ;
- La variable adult qui va être analysée par la ternaire ;
- Un point d'interrogation suivi d'une valeur (un nombre, du texte, etc.) ;
- Deux points suivis d'une deuxième valeur et enfin le point-virgule marquant la fin de la ligne d'instructions.

```
var startMessage = 'Votre catégorie : ',
    endMessage,
    adult = confirm('Êtes-vous majeur ?');

if (adult) { // la variable "adult" contient un Booléen, on peut donc directement la soumettr
    endMessage = '18+';
}

else {
    endMessage = '-18';
}

alert(startMessage + endMessage);
```




s in French Would you like to translate it? Translate
age = 'Votre catégorie : ',
e,
onfirm('Êtes-vous majeur ?');
// la variable "adult" contient un Booléen, on pe
io
e = '18+';
e = '-18';
ssage + endMessage);
The page at www.sdz-files.com says:
Êtes-vous majeur ?
OK
Cancel


The page at www.sdz-files.com says:
Votre catégorie : -18
Prevent this page from creating additional dialogs.
OK

Les boucles

```
while (condition) {
    instruction_1;
    instruction_2;
    instruction_3;
}
```

```
var number = 1;

while (number < 10) {
    number++;
}

alert(number); // Affiche : « 10 »
```

```
do {
    instruction_1;
    instruction_2;
    instruction_3;
} while (condition);
```

Attention à la syntaxe de la boucle `do while` : il y a un point-virgule après la parenthèse fermante du `while` !

```
for (initialisation; condition; incrémentation) {
    instruction_1;
    instruction_2;
    instruction_3;
}
```

```
for (var iter = 0; iter < 5; iter++) {
    alert('Itération n°' + iter);
}
```

Les fonctions

- Jusqu'ici:alert(), prompt(), confirm()et parseInt()

```
function myFunction(arguments) {
    // Le code que la fonction va devoir exécuter
}
```

```
function showMsg() {
    alert('Et une première fonction, une !');
}

showMsg(); // On exécute ici le code contenu dans la fonction
```

La portée des variables

```
var message = 'Ici la variable globale !';

function showMsg() {
    var message = 'Ici la variable locale !';
    alert(message);
}

showMsg();

alert(message);
```

Les arguments

```
function moar(first, second) {
    // On peut maintenant utiliser les variables « f
    alert('Votre premier argument : ' + first);
    alert('Votre deuxième argument : ' + second);
}
```

```
function sayHello() {
    return 'Bonjour !';
    alert('Attention ! Le texte arrive !');
}

alert(sayHello());
```

Les fonctions anonymes

- Des fonctions qui ne portent pas de nom sont des fonctions anonymes et servent à isoler une partie du code.

```
var sayHello = (function() {

    return 'Yop !';

})();

alert(sayHello); // Affiche : « Yop ! »
```

Les objets

- Un objet est un concept, une idée ou une chose.
- Un objet possède une structure qui lui permet de pouvoir fonctionner et d'interagir avec d'autres objets.
- Le Javascript met à notre disposition des objets natifs, c'est-à-dire des objets directement utilisables.
- Examples: Number, Boolean, String, Array

```
var myString = 'Ceci est une chaîne de caractères'; // On crée un objet String

alert(myString.length); // On affiche le nombre de caractères, au moyen de la propri

alert(myString.toUpperCase()); // On récupère la chaîne en majuscules, avec la métho
```

Les tableaux

- Un tableau, ou plutôt un *array* en anglais, est une variable qui contient plusieurs valeurs, appelées **items**. Chaque item est accessible au moyen d'un **indice** (*index* en anglais) et dont la numérotation commence à partir de 0.

```
1 var myArray = ['Sébastien', 'Laurence', 'Ludovic', 'Pauline', 'Guillaume'];
```

```
1 var myArray = new Array('Sébastien', 'Laurence', 'Ludovic', 'Pauline', 'Guillaume');
```

Indice	0	1	2	3	4
Donnée	Sébastien	Laurence	Ludovic	Pauline	Guillaume

```
1 var myArray_a = [42, 12, 6, 3];
2 var myArray_b = [42, 'Sébastien', 12, 'Laurence'];
```

Récupérer et modifier des valeurs

```
1 var myArray = ['Sébastien', 'Laurence', 'Ludovic', 'Pauline', 'Guillaume'];
2
3 alert(myArray[1]); // Affiche : « Laurence »
```

```
1 var myArray = ['Sébastien', 'Laurence', 'Ludovic', 'Pauline', 'Guillaume'];
2
3 myArray[1] = 'Clarisse';
4
5 alert(myArray[1]); // Affiche : « Clarisse »
```

Ajouter et supprimer des items dans un tableau

- La méthode push() permet d'ajouter un ou plusieurs items à un tableau.
- La méthode unshift() fonctionne comme push(), excepté que les items sont ajoutés au début du tableau.
- Les méthodes shift() et pop() retirent respectivement le premier et le dernier élément du tableau.

```
var myArray = ['Sébastien', 'Laurence'];

myArray.push('Ludovic'); // Ajoute « Ludovic » à la fin du tableau
myArray.push('Pauline', 'Guillaume'); // Ajoute « Pauline » et « Guillaume » à la fin du
```

```
1 var myArray = ['Sébastien', 'Laurence', 'Ludovic', 'Pauline', 'Guillaume'];
2
3 myArray.shift(); // Retire « Sébastien »
4 myArray.pop(); // Retire « Guillaume »
5
6 alert(myArray); // Affiche « Laurence,Ludovic,Pauline »
```

Chaînes de caractères et tableaux

- Les chaînes de caractères possèdent une méthode split() qui permet de les découper en un tableau, en fonction d'un séparateur.

```
var cousinsString = 'Pauline Guillaume Clarisse',
cousinsArray  = cousinsString.split(' ');
```

- L'inverse de split(), c'est-à-dire créer une chaîne de caractères depuis un tableau, se nomme join().

```
var cousinsString_2 = cousinsArray.join('-');//Pauline-Guillaume-Clarisse
```

Parcourir avec for un tableau

```
var myArray = ['Sébastien', 'Laurence', 'Ludovic',
   'Pauline', 'Guillaume'];

for (var i = 0; i < myArray.length; i++)
{
   alert(myArray[i]);
 }
```

Les objets littéraux

```
var family = {
    self:     'Sébastien',
    sister:   'Laurence',
    brother:  'Ludovic',
    cousin_1: 'Pauline',
    cousin_2: 'Guillaume'
};
```

Identifiant	self	sister	brother	cousin_1	cousin_2
Donnée	Sébastien	Laurence	Ludovic	Pauline	Guillaume

- **Accès aux items:** family.sister; family['sister'];
- **Ajouter des items:** family['uncle'] = 'Didier'; family.uncle = 'Didier';
- **Parcourir un objet avec for in:**

```
for (var id in family) { // On stocke l'identifiant dans « id » pour parcourir l'objet « famil
    y »
  alert(family[id]);
}
```

Document Object Model

- Le **Document Object Model** (abrégé **DOM**) est une interface de programmation pour les documents XML et HTML, via le JavaScript

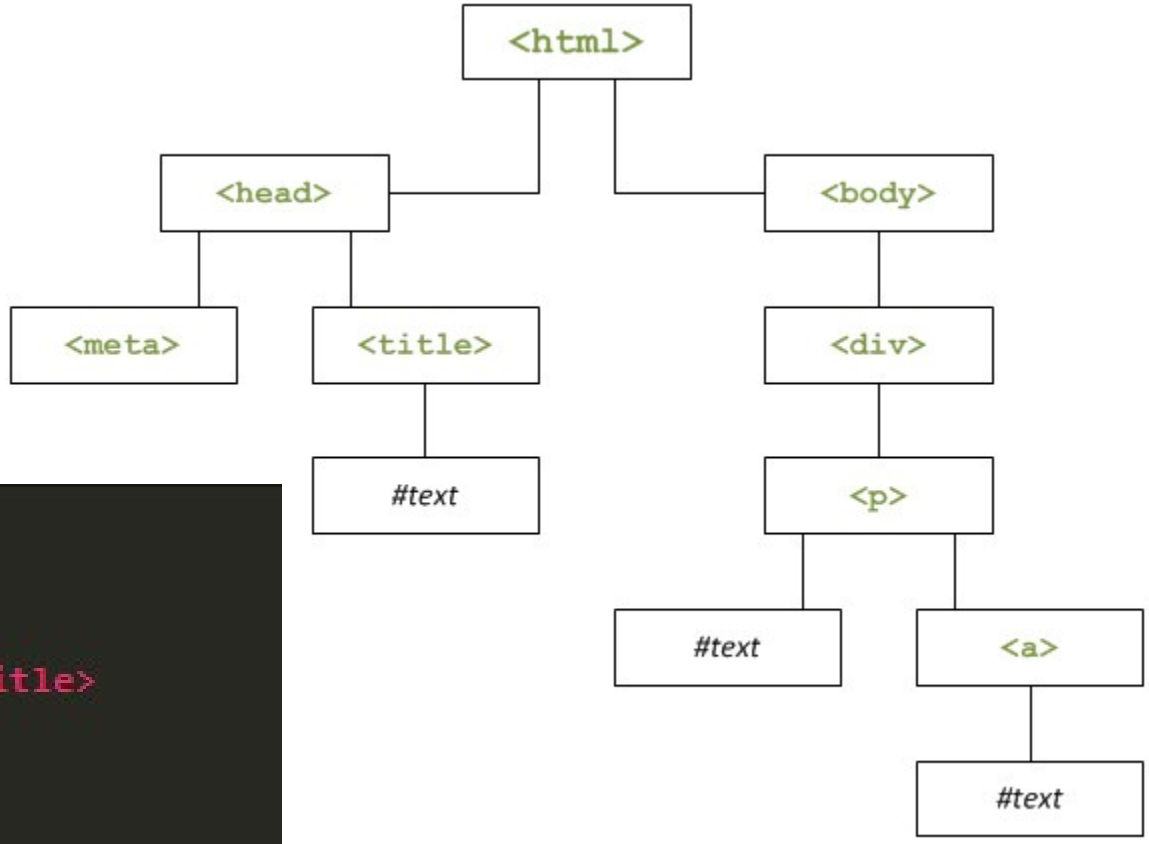
- Une interface de programmation = une **API** (**Application Programming Interface)**

- Un document HTML ou XML est représenté sous la forme d'un arbre, ou plutôt hiérarchiquement. Ainsi, l'élément <html> contient deux éléments enfants: <head> et <body>, qui à leur tour contiennent d'autres éléments enfants.

- L'objet document est un sous-objet de window (window.alert('Hello world!'); ⇔ alert('Hello world!');)

La structure DOM

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
  </head>

  <body>
    <div>
      <p>Un peu de texte <a>et un lien</a></p>
    </div>
  </body>
</html>
```



Une page Web peut être vue comme un arbre

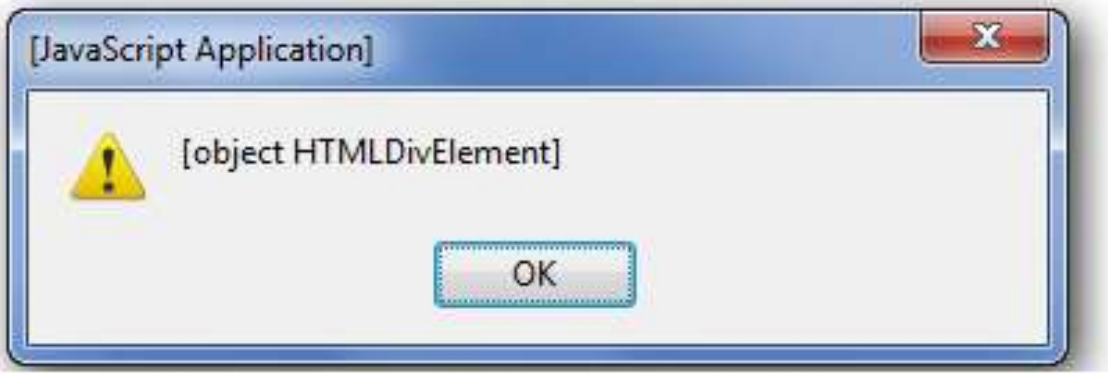
Accéder aux éléments

- L'objet document possède trois méthodes principales:
  - getElementById():
    - cette méthode permet d'accéder à un élément en connaissant son ID qui est simplement l'attribut id de l'élément
  - getElementsByTagName():
    - cette méthode permet de récupérer, sous la forme d'un tableau, tous les éléments de la famille
    - cette méthode est accessible sur n'importe quel élément HTML et pas seulement sur l'objet document
  - getElementsByName():
    - cette méthode est semblable à getElementsByTagName() et permet de ne récupérer que les éléments qui possèdent un attribut name que vous spécifiez
    - cette méthode est dépréciée en XHTML mais est maintenant standardisée pour l'HTML5

getElementById()

```
1 <div id="myDiv">
2   <p>Un peu de texte <a>et un lien</a></p>
3 </div>
4
5 <script>
6   var div = document.getElementById('myDiv');
7
8   alert(div);
9 </script>
```

En exécutant ce code, le navigateur affiche ceci :



getElementsByTagName()

```
1 var divs = document.getElementsByTagName('div');
2
3 for (var i = 0, c = divs.length ; i < c ; i++) {
4     alert('Element n° ' + (i + 1) + ' : ' + divs[i]);
5 }
```

- En paramètre de cette méthode vous pouvez mettre une chaîne de caractères contenant un astérisque * qui récupérera tous les éléments HTML contenus dans l'élément ciblé.

querySelector() et querySelectorAll()

- Ces deux méthodes prennent pour paramètre un seul argument : une chaîne de caractères !

- Cette chaîne de caractères doit être un sélecteur CSS comme ceux que vous utilisez dans vos feuilles de style.

- querySelector() renvoie le premier élément trouvé correspondant au sélecteur CSS

- querySelectorAll() va renvoyer *tous* les éléments (sous forme de tableau) correspondant au sélecteur CSS

- Exemples:

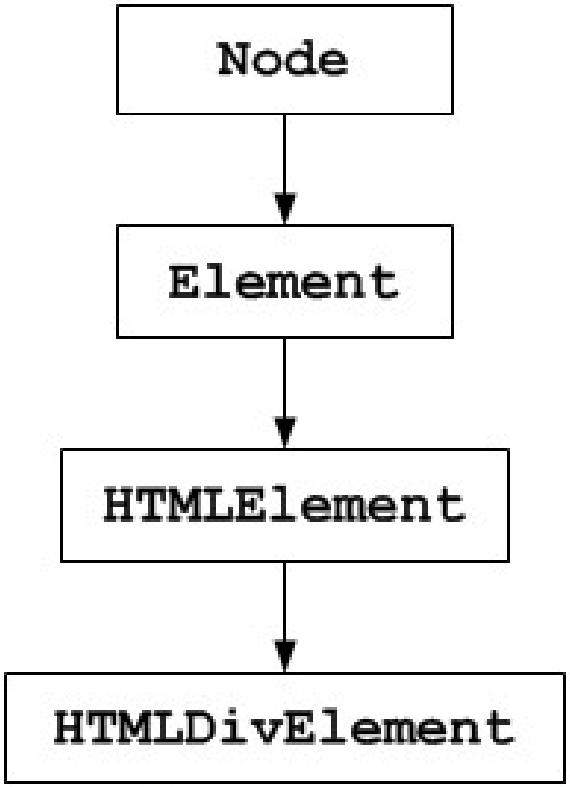
  var query = document.querySelector('#menu .item span'), queryAll = document.querySelectorAll('#menu .item span');// Ce sélecteur CSS stipule que l'on souhaite sélectionner les balises de type contenues dans les classes .item elles-mêmes contenues dans un élément dont l'identifiant est #menu.

```html
<div id="menu">

  <div class="item">
    <span>Élément 1</span>
    <span>Élément 2</span>
  </div>

  <div class="publicite">
    <span>Élément 3</span>
    <span>Élément 4</span>
  </div>

</div>

<div id="contenu">
  <span>Introduction au contenu de la page...</span>
</div>
```

```
var query = document.querySelector('#menu .item span'),
    queryAll = document.querySelectorAll('#menu .item span');

alert(query.innerHTML); // Affiche : "Élément 1"

alert(queryAll.length); // Affiche : "2"
alert(queryAll[0].innerHTML + ' - ' + queryAll[1].innerHTML); // Affiche :
    "Élément 1 - Élément 2"
```

L'héritage des propriétés et des méthodes

- Le Javascript voit les éléments HTML comme étant des objets, cela veut donc dire que chaque élément HTML possède des propriétés et des méthodes.



En Javascript, un objet peut appartenir à plusieurs groupes

Éditer les éléments HTML

- Les éléments HTML sont composés:
  - d'attributs
  - d'un contenu, qui est de type #text: le contenu peut aussi être un autre élément HTML
- Pour interagir avec les attributs, l'objet Element nous fournit deux méthodes:
  - getAttribute() : récupérer un attribut
  - setAttribute(): éditer un attribut

- Exemple:

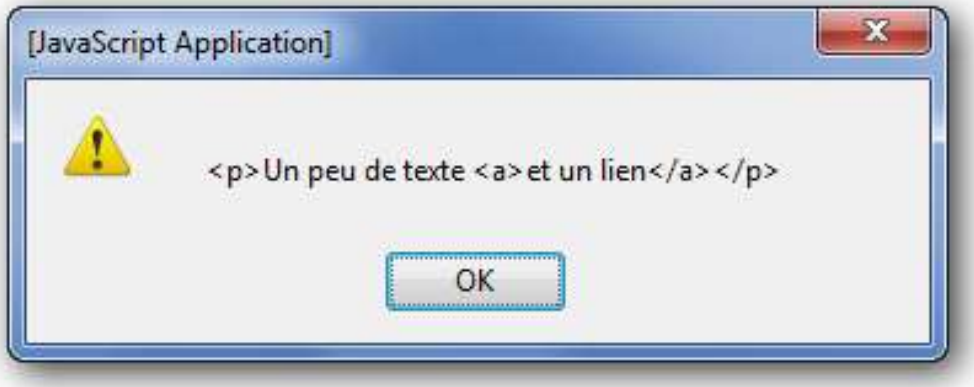
```
<body>
    <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié dyna
    miquement</a>
    <script>
            var link = document.getElementById('myLink');
            var href = link.getAttribute('href'); // On récupère l'attribut « href »
            //or il est possible d'accéder à un attribut via une propriété  link.href
            alert(href);
            link.setAttribute('href', 'http://www.siteduzero.com'); // On édite l'attribut «
    href »
    </script>
</body>
```

Récupérer du HTML avec innerHTML()

- innerHTML permet de récupérer le code HTML enfant d'un élément sous forme de texte. Ainsi, si des balises sont présentes, innerHTML les retournera sous forme de texte

```
<body>
  <div id="myDiv">
    <p>Un peu de texte <a>et un lien</a></p>
  </div>

  <script>
    var div = document.getElementById('myDiv');

    alert(div.innerHTML);
  </script>
</body>
```

Nous avons donc bien une boîte de dialogue qui affiche le contenu de `myDiv`, sous forme de texte :



Le contenu de myDiv est bien affiché

Ajouter ou éditer du HTML avec innerHTML()

- Editer:

```
document.getElementById('myDiv').innerHTML = '<blockquote>Je mets une citation à la place du paragraphe</blockquote>';
```

- Ajouter:

```
document.getElementById('myDiv').innerHTML += ' et <strong>une portion mise en emphase</strong>.';
```

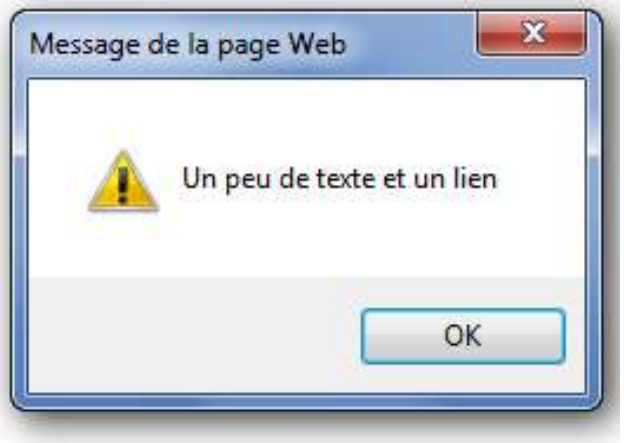
- Si un jour il vous prend l'envie d'ajouter une balise <script> à votre page par le biais de la propriété innerHTML, sachez que ceci ne fonctionne pas ! Il est toutefois possible de créer cette balise par le biais de la méthode createElement()

innerText et textContent

- innerText pour Internet Explorer et textContent pour les autres navigateurs

- le fonctionnement d'innerText est le même qu'innerHTML excepté le fait que seul le texte est récupéré, et non les balises.

```
<body>
    <div id="myDiv">
        <p>Un peu de texte <a>et un lien</a></p>
     </div>
    <script>
        var div = document.getElementById('myDiv');
        alert(div.innerText || div.textConent);
</script>
</body>
```



Le texte est affiché sans les balises HTML

Naviguer entre les nœuds HTML

- La propriété parentNode permet d'accéder à l'élément parent d'un élément

```
<blockquote>
   <p id="myP">Ceci est un paragraphe !</p>
</blockquote>
var paragraph = document.getElementById('myP');
var blockquote = paragraph.parentNode;
```

- nodeType et nodeName servent respectivement à vérifier le *type* d'un nœud et le *nom* d'un nœud

- nodeType retourne un nombre, qui correspond à un type de nœud

```
<blockquote>
    <p id="myP">Ceci est un paragraphe !</p>
</blockquote>

var paragraph = document.getElementById('myP');
alert(paragraph.nodeType + '\n\n' + paragraph.nodeName.toLowerCase());
```

Numéro	Type de nœud
1	Nœud élément
2	Nœud attribut
3	Nœud texte
4	Nœud pour passage CDATA (relatif au XML)
5	Nœud pour référence d'entité
6	Nœud pour entité
7	Nœud pour instruction de traitement
8	Nœud pour commentaire
9	Nœud document
10	Nœud type de document
11	Nœud de fragment de document
12	Nœud pour notation

- firstChild et lastChild servent respectivement à accéder au premier et au dernier enfant d'un nœud

- nodeValue et data: on récupère le contenu

- childNodes retourne un tableau contenant la liste des enfants d'un élément

- nextSibling et previousSibling sont deux propriétés qui permettent d'accéder respectivement au nœud suivant et au nœud précédent

```
<body>
  <div>
    <p id="myP">Un peu de texte <a>et un lien</a></p>
  </div>

  <script>
    var paragraph = document.getElementById('myP');
    var children  = paragraph.childNodes;

    for (var i = 0, c = children.length; i < c; i++) {

      if (children[i].nodeType === 1) { // C'est un élément HTML
        alert(children[i].firstChild.data);
      } else { // C'est certainement un nœud textuel
        alert(children[i].data);
      }

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

Exemple

```
<!doctype html>
2<html>
3  <head>
4    <meta charset="utf-8" />
5    <title>Le titre de la page</title>
6  </head>
7
8  <body>
9    <div>
10     <p id="myP">Un peu de texte, <a>un lien</a> et
       <strong>une portion en emphase</strong></p>
11   </div>
12
13   <script>
14     var paragraph = document.getElementById('myP');
15     var first = paragraph.firstChild;
16     var last  = paragraph.lastChild;
17
18     alert(first.nodeName.toLowerCase());
19     alert(last.nodeName.toLowerCase());
20   </script>
21 </body>
22</html>
```

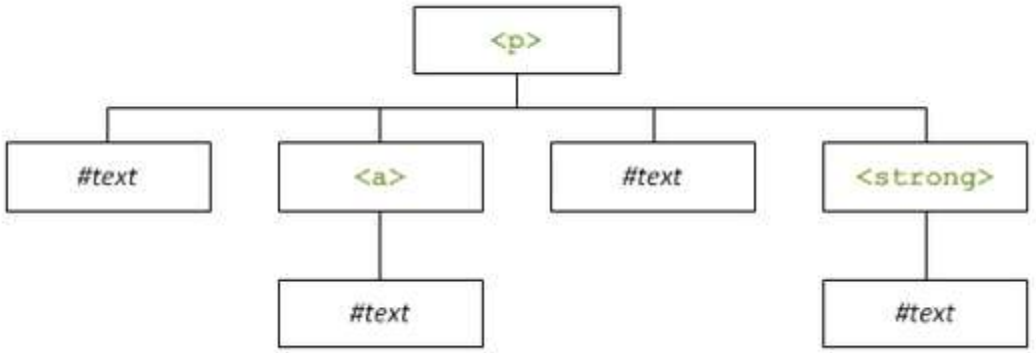


Schéma de l'élément myP

Résultat=?

Ajouter des éléments HTML

- Avec le DOM, l'ajout d'un élément HTML se fait en trois temps :
  - On crée l'élément

```
var newLink = document.createElement('a');
var newLinkText = document.createTextNode("Le Site du Zéro");
```

  - On lui affecte des attributs

```
newLink.id    = 'sdz_link';
newLink.href  = 'http://www.siteduzero.com';
newLink.title = 'Découvrez le Site du Zéro !';
```

  - On l'insère dans le document

```
newLink.appendChild(newLinkText);
document.getElementById('myP').appendChild(newLink);
```

- **Remplacer un élément par un autre**

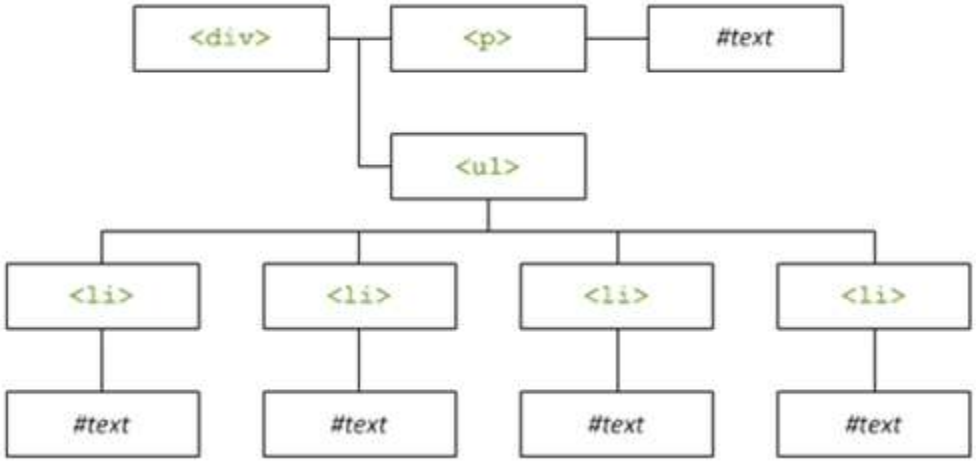
```
<body>
  <div>
    <p id="myP">Un peu de texte <a>et un lien</a></p>
  </div>

  <script>
    var link = document.getElementsByTagName('a')[0];
    var newLabel= document.createTextNode('et un hyperlien');

    link.replaceChild(newLabel, link.firstChild);
  </script>
</body>
```

- **Supprimer un élément**

```
var link = document.getElementsByTagName('a')[0];

link.parentNode.removeChild(link);
```

Il n'y a pas besoin de récupérer myP (l'élément parent) avec getElementById(), autant le faire directement avec parentNode.

```html
<div id="divTP2">
  <p>Langages basés sur ECMAScript :</p>

  <ul>
    <li>JavaScript</li>
    <li>JScript</li>
    <li>ActionScript</li>
    <li>EX4</li>
  </ul>
</div>
```



```js
// On crée l'élément conteneur
var mainDiv = document.createElement('div');
    mainDiv.id = 'divTP2';

// On crée tous les nœuds textuels, pour plus de facilité
var languages = [
    document.createTextNode('JavaScript'),
    document.createTextNode('JScript'),
    document.createTextNode('ActionScript'),
    document.createTextNode('EX4')
];

// On crée le paragraphe
var paragraph     = document.createElement('p');
var paragraphText = document.createTextNode('Langages basés sur ECMAScript :');
paragraph.appendChild(paragraphText);


// On crée la liste, et on boucle pour ajouter chaque item
var uList = document.createElement('ul'),
    uItem;

for (var i = 0, c=languages.length; i < c; i++) {
    uItem = document.createElement('li');

    uItem.appendChild(languages[i]);
    uList.appendChild(uItem);
}

// On insère le tout dans mainDiv
mainDiv.appendChild(paragraph);
mainDiv.appendChild(uList);

// On insère mainDiv dans le <body>
document.body.appendChild(mainDiv);
```

Evénements

- Les événements sont des actions qui peuvent être détectés par Javascript

- Les événements sont utilisés pour exécuter une fonction (event handler) lors de la détection de l'événement.

Evénement	Les balises HTML	Les balises HTML
onfocus=""	Se produit lorsque l'utilisateur donne le focus à un élément, c'est-à-dire que cet élément est sélectionné comme étant l'élément actif	select, text, textarea
onblur=""	Se produit lorsque l'élément perd le focus	select, text, textarea
onchange=""	Se produit lorsque l'utilisateur modifie le contenu d'un champ de données	select, text, textarea
onselect=""	Se produit lorsque l'utilisateur sélectionne un texte (ou une partie d'un texte) dans un champ	text, textarea
onmouseover=""	Se produit lorsque l'utilisateur positionne le curseur de la souris au-dessus d'un élément	a
onmouseout=""	Se produit lorsque le curseur de la souris quitte un élément	a
onclick=""	Se produit lorsque l'utilisateur clique sur l'élément associé à l'événement	a, button, checkbox, radio, reset, submit
onload=""	Se produit lorsque le navigateur de l'utilisateur charge la page en cours	body, frameset
onunload=""	Se produit lorsque le navigateur de l'utilisateur quitte la page en cours	body, frameset
onsubmit=""	Se produit lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire (le bouton qui permet d'envoyer le formulaire)	form

Formulaires

- Un formulaire est simplement une zone qui peut contenir des champs de formulaire.
- Les champs de formulaire sont des objets qui permettent au visiteur de saisir des informations, qui est habituellement envoyé à un serveur.
- Exemple:

C'est ce qui arrive lorsque le formulaire est soumis:
•Le nom d'utilisateur et mot de passe sont envoyés à un programme sur le serveur.
•Le programme sur le serveur va chercher une base de données pour les entrées valides.
•Si l'entrée est valide, le visiteur est dirigé vers la page protégée. Sinon, le visiteur est envoyé à la page «échec».

La balise FORM

- Quand un formulaire est soumis, tous les champs du formulaire sont envoyés vers une url.

- La balise form indique au navigateur où la forme commence et se termine:il peut contenir des champs de formulaire, mais aussi d'autres balises HTML (e.g., div, table...). Seulement les valeurs des champs du formulaire sont envoyés.

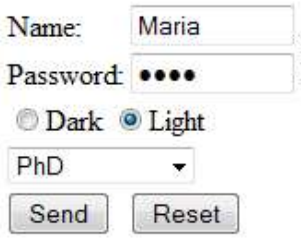
- Propriétés de la balise form :
  - action=addresse (l'URL vers laquelle le contenu doit être envoyé à)
  - method=post or method=get (modalités de présentation des données)

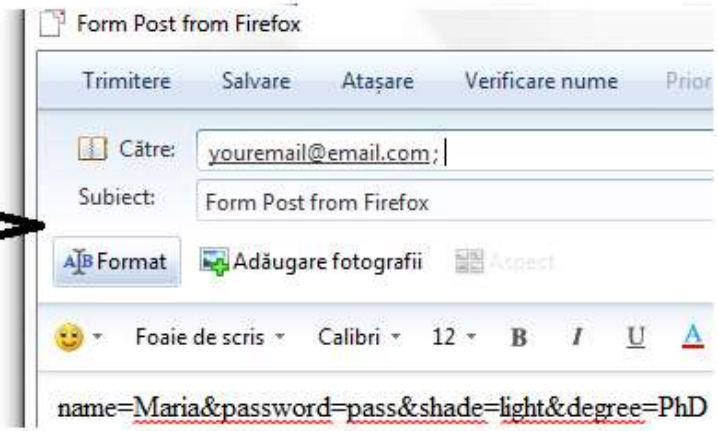
Quelques champs du formulaire

Champ (explication)	Attributs (explications: http://www.commentcamarche.net/contents/html/htmlform.php3)
textarea (une zone de saisie)	rows= (rows in the field) cols= (columns in the field) name= (name of the field) wrap=off/virtual/physical(control lines breaks)
text (champ de saisie)	size= (no. of characters shown) maxlength= (max characters allowed) name= (name of the field) value=(initial value in the field)
password (champ mot de passe)	size= (no. of characters shown) maxlength= (max characters allowed) name= (name of the field) value=(initial value in the field)
checkbox (case à cocher)	name= (name of the field) value=(initial value in the field)
radio (bouton radio)	name= (name of the field) value=(initial value in the field)
select (une liste à choix multiples)	name= (name of the field) size= (number of items in list) multiple=(allow multiple choice if yes)
option (des éléments individuels dans la liste à choix multiples)	selected=(make an item default) value=(value to send if selected)
hidden (n'apparaît pas sur la forme)	name= (name of the field) value=(value in the field)
reset (button pour réinitialiser tous les champs)	name= (name of the button) value=(text shown on the button)
submit (bouton pour soumettre le formulaire)	name= (name of the button) value=(text shown on the button)
image (image squi e comporte comme un bouton)	name= (name of the image)

Exemple

```
<form method="post" action="mailto:youremail@email.com">
    <table>
        <tr><td> Name: </td>
            <td> <input type="text" size="10" maxlength="40" name="name"></td>
        </tr>
        <tr><td> Password: </td>
            <td><input type="password" size="10" maxlength="10" name="password"> </td>
        </tr>
        <tr><td colspan="2">
            <input type="radio" name="shade" value="dark">Dark
            <input type="radio" name="shade" value="light">Light</td>
        </tr>
        <tr><td colspan="2">
            <select name="degree">
                <option>Choose One</option>
                <option>Bachelor</option>
                <option>Master</option>
                <option>PhD</option>
            </select>
        </td></tr>
        <tr><td> <input type="submit" value="Send"> </td>
            <td> <input type="reset" value="Reset"> </td>
        </tr>
    </table>
</form>
```





Validation de formulaire - Pré-requis

- Vérifiez quelque chose avant de soumettre un formulaire (par exemple vérifiez si une valeur à la hauteur de la syntaxe générale d'un e-mail, vérifiez pour voir si un nombre se situe dans un certain intervalle, vérifiez pour voir si un consits valeur d'un certain nombre de chiffres, vérifiez pour voir si un champ est vide ou non, ...)

- Problème: comment voulez-vous récupérer les valeurs de vos éléments?

- Solutions possibles:
  - mettez identifiants uniques à vos éléments et puis, utilisez *document.getElementById* fonction
  - utilisez *document.getElementsByTagName* fonction
  - utilisez le mot réservé *this*

```
<html>
<head>
    <title>My Javascript Page</title>
        <script type="text/javascript">
        function checkIfEmpty(){
            var myTextField = document.getElementById("myText");
            var result = document.getElementById("myDiv");
            if(myTextField.value != "")
                result.innerHTML = "The value of the text is <br/>"+ myTextField.value;
            else
                result.innerHTML = "Your text is empty!";
        }
        </script>
</head>
<body>
<form method="post" action="mailto:youremail@email.com">
    <input type="text" id="myText" />
    <input type="button" onclick="checkIfEmpty()" value="Form Checker" />
</form>
<div id="myDiv">Here the result will be displayed!!!</div>
</body>
</html>
```

Before pressing the button:

Form Checker

Here the result will be displayed!!!

After pressing the button:

ggggg Form Checker

The value of the text is
ggggg

this mot réservé: se réfère à la «propriétaire» de la fonction que nous exécutons

```
<html>
<head>
    <title>My Javascript Page</title>
        <script type="text/javascript">
        function checkIfEmpty(myForm){
            var inputFieldValue = myForm.myText.value;
            var result = document.getElementById("myDiv");
            if(inputFieldValue != "")
                result.innerHTML = "The value of the text is <br/>"+ inputFieldValue;
            else
                result.innerHTML = "Your text is empty!";
        }
        </script>
</head>
<body>
<form method="post" action="mailto:youremail@email.com">
    <input type="text" id="myText" />
    <input type="button" onclick="checkIfEmpty(this.form)" value="Form Checker" />
</form>
<div id="myDiv">Here the result will be displayed!!!</div>
</body>
</html>
```

- Remarque: Il existe plusieurs solutions pour le même problème!
- Pour savoir plus: http://jlcastellani.free.fr/jscript.htm

Exemple

- Site développé avec JS:
  - http://www.pm.org.ro/certexam/