

I55

JAVASCRIPT

3^{ÈME} ANNÉE LICENCE EN INFORMATIQUE

FSM 2020-2021



Introduction

- Bien que les langages HTML/CSS soient reconnus comme étant des langages de création de contenu et de mise en forme des pages Web, ils présentent des limites :
 - Absence de structures de contrôle de flux
 - Aucune connectivité avec les serveurs de base de données
 - Manque de dynamicité (pas d'interactions avec l'utilisateur)
- Pour étendre les capacités de ces langages, la langage de programmation "JavaScript" a été mis en œuvre.

Introduction

- JavaScript est un langage de scripts incorporé aux balise HTML, permet d'améliorer la présentation et l'interactivité des pages Web:
 - Contrôler la validité du contenu saisi par l'utilisateur avant soumission du formulaire,
 - Enrichir le contenu des pages web avec un contenu interactif et/ou animé
- La balise **<script>** permet d'intégrer du Javascript au sein d'un document HTML.
- Pour écrire et tester des codes JavaScript, il faut :
 - un navigateur Web qui reconnaît JavaScript (IE, Netscape, Mozilla Firefox, ...),
 - un éditeur de traitement de texte (Notepad++, bloc note, Word pad,...)

Intégration de JavaScript dans HTML

- Le code JavaScript peut être placé:
 - dans le corps de la page HTML (`<body>...</body>`)
 - dans l'entête de la page HTML (`<head>...</head>`)

```
<body>
  <script type="text/javascript">
    // Mon code Javascript
    ...
  </script>
  ....
</body>
```

Intégration de JavaScript dans HTML

- Le code JavaScript peut être inséré **directement** au niveau de la balise ouvrante
 - Sous la forme d'un couple *attribut-valeur* HTML :
 - **attribut** = événement déclencheur
 - **valeur** = code JavaScript déclenché

```
<form method="post" ...  
      onsubmit='verifie()';>
```

```
<input type="button" value="cliquez ici"  
      onclick="alert('vous avez bien cliqué ici')">
```

Intégration de JavaScript dans HTML

- Le code JavaScript peut être défini dans un fichier externe qui porte l'extension ".js" et puis l'intégrer dans la page HTML en utilisant la balise `<script>` et en utilisant l'attribut `src`

```
<script type="text/javascript" src="fichier.js"></script>
```

- Cette méthode a pour intérêt de réutiliser ce code dans d'autres pages HTML autant de fois désirées.

161

ELÉMENTS DU LANGAGE



Syntaxe générale : caractéristiques

- Similarités avec les langages

- C, C++, Java, ...

- Commentaire

- Sur une ligne :

// ... commentaire...

- Sur plusieurs lignes :

```
/*    ...  
      commentaire  
    */
```

- Séparateur d'instructions

- Point virgule :

instruction ;

- Groupement d'instructions

- Accolades :

{ ... instructions ... }

Déclaration de variables

- En utilisant le mot clé *var*
 - soit de façon explicite, en utilisant le mot clé *var*: `var chaine= "bonjour"`
 - Portée :
 - déclaration *en dehors* de fonction => globale
 - déclaration *dans* une fonction => locale
 - soit de façon implicite, en écrivant le nom de la variable suivie du caractère `=` et de la valeur à affecter : `chaine= "bonjour"`
 - Portée : globale
- En utilisant le mot clé *let*
 - `let numberOfCats = 3;`
 - Portée : « de bloc », càd, la variable sera accessible dans le bloc dans lequel elle a été définie et dans les blocs que le bloc contient.

Déclaration de variables

- Différences entre *var* et *let*:
 - **La remontée des variables** : Lorsqu'on utilise la syntaxe avec *var*, on n'est pas obligé de déclarer la variable avant de la manipuler dans le code, on peut effectuer des manipulations en haut du code et la déclarer en fin de code
 - **La redéclaration de variables** : Avec la syntaxe *var*, on a droit à redéclarer la même variable (ce qui a pour effet la modification de sa valeur). La syntaxe *let* n'autorise pas cela.

```
//Ceci fonctionne  
prenom = "Ali";  
var prenom;  
  
//Ceci ne fonctionne pas  
nom = "Ben Salah";  
let nom;
```

```
//Ceci fonctionne  
var prenom = "Ali";  
var prenom = "Salah";  
  
//Ceci ne fonctionne pas  
let prenom = "Ali";  
let prenom = "Salah";
```

Déclaration de variables

- Pas de typage (détection automatique par l'interpréteur)
- Types primitifs:
 - **number** : entier, réel
 - **string** : chaîne de caractères
 - **boolean** : valeur logique
 - **null** : absence de données dans une variable
 - **undefined** : une variable dont le contenu n'est pas clair car elle n'a jamais stocké de valeur, pas même **null**
- Nom de variable **sensible à la casse**.

Les opérateurs

- Les opérateurs arithmétiques de base : +, -, *, /
- Les opérateurs de comparaison : ==, !=, <, <=, >, >=
- Les opérateurs logiques : &&, ||, !
- L'opérateur de concaténation : +
- L'opérateur **typeof** (retourne le type d'une variable/objet...)

```
typeof 42; // expected output: "number"  
typeof 'blubber'; // expected output: "string"  
typeof true; // expected output: "boolean"  
typeof undeclaredVariable; // expected output: "undefined"
```

Les constantes

- Pour créer ou déclarer une constante en JavaScript, nous allons utiliser le mot clé **const**.

```
const prenom = 'Ali';  
const age = 20;
```


Structures de contrôle

- **if** (condition) {.....} **else**{.....}
- **switch**(X) {**case 1** : ...; **break**; **default** ...;}
- **for** (x=A; x<B; x++){.....}
- **while** (condition) {.....}
- **do**{.....} **while**(condition)

Les fonctions

- déclarées par le mot clef **function**
- retour du résultat avec **return**

```
function calculPrixTTC(prixHT, tauxTVA){  
    if (!(typeof prixHT == "number") || !(typeof tauxTVA == "number")){  
        throw new Error("Function calculPrixTTC appelée avec paramètre incorrect.")  
    }  
    return prixHT*(1.0+tauxTVA/100.0);  
}
```

Les fonctions prédéfinies

- **isFinite**

- Détermine si le paramètre est un nombre fini ou non
- Renvoie false si le paramètre n'est pas un nombre (**NaN**) ou l'infini positif (**Infinity**) ou l'infini négatif (**-Infinity**)

```
isFinite(240) //retourne true  
isFinite("Un nombre") //retourne false
```

- **isNaN**

- détermine si le paramètre n'est pas un nombre (**NaN** : Not a Number).

```
isNaN("Un nombre")//retourne true  
isNaN(20) //retourne false
```

Les fonctions prédéfinies

- **parseFloat**

- analyse une chaîne de caractères et retourne un nombre décimal.
- Si l'argument évalué n'est pas un nombre, renvoie **NaN** (Not a Number).

```
let numero="125";  
let nombre=parseFloat(numero); //retourne le nombre 125
```

- **parseInt**

- analyse une chaîne de caractères et retourne un nombre entier de la base spécifiée.

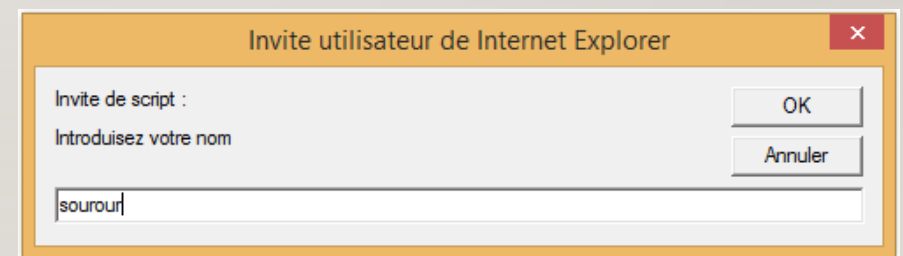
```
let prix=30.75;  
let arrondi = parseInt(prix, 10); //retourne 30
```

Entrée et sortie de données

- La fonction **alert** sert à afficher une valeur
 - **alert** ("Hello World !");

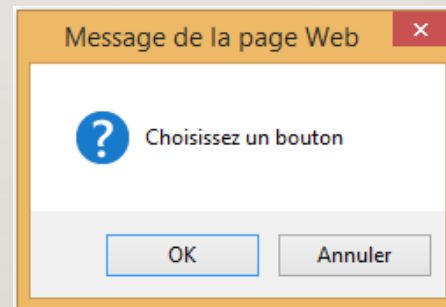


- La fonction **prompt** sert à lire une valeur
 - **x = prompt**("Introduisez votre nom");



Entrée et sortie de données

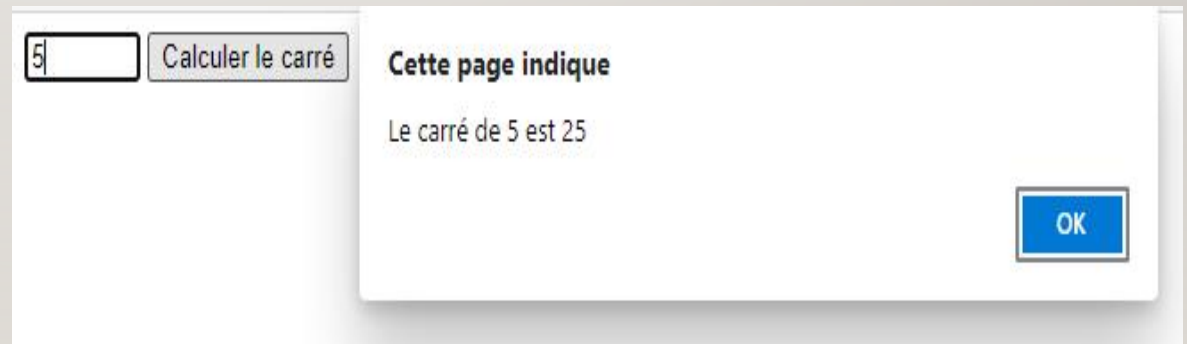
- La fonction **confirm** permet à l'utilisateur de choisir entre les boutons **OK** et **Annuler**.
 - **nom = confirm("Choisissez un bouton");**



- La méthode **document.write** permet d'écrire du code HTML dans la page WEB

Exercice 1

- Ecrire le code d'un fichier HTML permettant de calculer et d'afficher le carré d'un nombre saisi au clavier.



The image shows a web form with a text input field containing the number '5' and a button labeled 'Calculer le carré'. To the right, a message box titled 'Cette page indique' displays the result: 'Le carré de 5 est 25'. An 'OK' button is located at the bottom right of the message box.

5 Calculer le carré

Cette page indique
Le carré de 5 est 25

OK

175

NOTION D'OBJET EN JAVASCRIPT



Les objets en JavaScript

- Une **classe** est la description d'un ensemble de:
 - **propriétés** (les données) et de
 - **méthodes** (les fonctions).
- Un **objet** est une **instance de classe** (c'est à dire une variable de type classe)
- Les objets de JavaScript sont classés en deux catégories:
 - Les objets prédéfinis : String, Number, Boolean, Array, Date, Math,....
 - Les objets d'interface : navigator, window, document, ...

L'objet tableau

- Les variables tableaux sont des objets de type **Array**:

```
let tab = new Array();  
tab[0] = 123;  
tab[1] = 456;
```

```
let tab = new Array(123, 456, 789);
```

```
let tab = [123, 456, 789];
```

L'objet tableau

- Propriété :
 - **length** : retourne le nombre d'éléments du tableau;
- Méthodes :
 - **concat()** : permet de concaténer 2 tableaux;
 - **join()** : converti un tableau en chaîne de caractères;
 - **reverse()** : inverse le classement des éléments du tableau;
 - **slice()** : retourne une section du tableau;
 - **sort()** : permet le classement des éléments du tableau;
 - **splice()** : modifie le contenu d'un tableau en retirant des éléments et/ou en ajoutant des nouveaux éléments.

L'objet tableau

- **push()** : ajouter des éléments en fin de tableau et retourner la nouvelle taille du tableau
- **pop()** : supprimer le dernier élément d'un tableau et retourner l'élément supprimé.
- **unshift()** : ajouter des éléments en début de tableau et retourner la nouvelle taille du tableau
- **shift()** : supprimer le premier élément d'un tableau et retourner l'élément supprimé.
- **includes()** : déterminer si un tableau contient une valeur

L'objet tableau : Exemples

```
let a = new Array("Vent", "Pluie", "Feu");  
a.join(); // "Vent,Pluie,Feu"  
a.join(", "); // "Vent, Pluie, Feu"  
a.join(" + "); // "Vent + Pluie + Feu"  
a.join(""); // "VentPluieFeu"
```

```
let b = ["zéro", "un", "deux", "trois"];  
let tranche = b.slice(1, 3); // ["un", "deux"]
```

```
let poissons = ["truite", "saumon", "perche"];  
poissons.splice(2, 0, "thon"); // poissons vaut désormais  
// ["truite", "saumon", "thon", "perche"];
```


Parcours d'un tableau

- Avec la propriété **length** :

```
for (let i = 0; i < poissons.length; i++) {  
  console.log(poissons[i]);  
}
```

- La boucle **for...in**

```
for (let i in poissons) {  
  console.log(poissons[i]);  
}
```

- La boucle **for...of**

```
for (let poisson of poissons) {  
  console.log(poisson);  
}
```

L'objet String

- Propriété :
 - **length** : retourne la longueur de la chaîne de caractères;
- Méthodes :
 - **charAt()** : renvoie le caractère se trouvant à une certaine position;
 - **charCodeAt()** : renvoie le code du caractère se trouvant à une certaine position;
 - **concat()** : permet de concaténer 2 chaînes de caractères;
 - **fromCharCode()** : renvoie le caractère associé au code;
 - **indexOf()** : permet de trouver l'indice d'occurrence d'un caractère dans une chaîne;
 - **lastIndexOf()** : permet de trouver le dernier indice d'occurrence d'un caractère;

L'objet String

- **link()** : formate la chaîne avec la balise pour permettre de faire un lien;
- **slice()** : retourne une portion de la chaîne;
- **substr()** : retourne une portion de la chaîne;
- **substring()** : retourne une portion de la chaîne;
- **toLowerCase()** : permet de passer toute la chaîne en minuscule;
- **toUpperCase()** : permet de passer toute la chaîne en majuscules;
- **includes()** : déterminer si une chaîne de caractères est incluse dans une autre.

L'objet String

- **startsWith()** : déterminer si une chaîne commence par une certaine sous chaîne.
- **endsWith()** : déterminer si une chaîne se termine par une certaine sous chaîne
- **replace()** : remplacer une sous chaîne dans une chaîne par une autre
- **trim()** : supprimer les espaces d'une chaîne

L'objet String : Exemples

```
let myString = String.fromCharCode(74, 97, 118, 97, 83, 99, 114, 105, 112, 116); // le mot JavaScript en ASCII
alert(myString); // Affiche : « JavaScript »
```

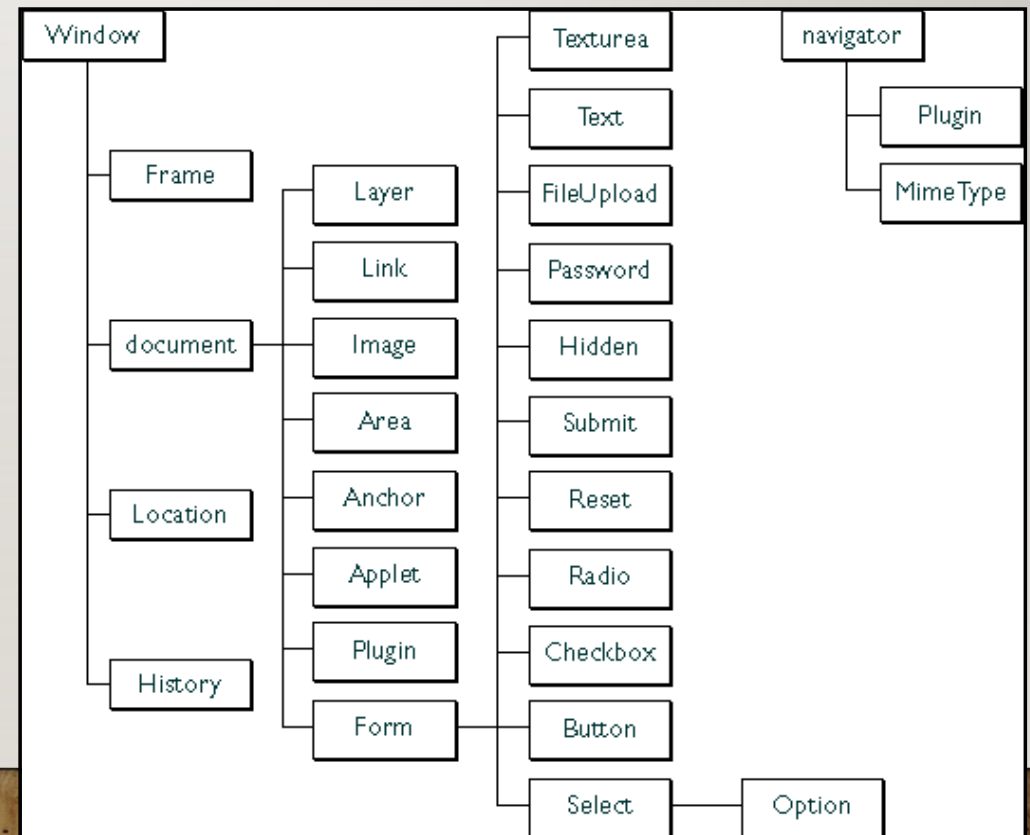
```
let myString2 = 'Le JavaScript est plutôt cool';
let result = myString2.indexOf('JavaScript');
if (result > -1) {
    alert('La chaîne contient le mot "JavaScript" qui débute à la position ' + result);
}
```

Exercice 2

- Créer une variable de type tableau dans laquelle on enregistre les jours de la semaine
- A l'aide de la méthode **join**, affichez :
Les jours de la semaine :Lundi ; Mardi ; Mercredi ; Jeudi ; Vendredi ; Samedi ; Dimanche.
- En utilisant la boucle **for**, affichez :
Jour 0 : Lundi
Jour 1 : Mardi
Jour 2 : Mercredi
Jour 3 : Jeudi
Jour 4 : Vendredi
Jour 5 : Samedi
Jour 6 : Dimanche

Les objets d'interface

- Javascript traite les éléments qui s'affichent dans votre navigateur comme des objets, c'est-à-dire des éléments classés selon :
 - une **hiérarchie** pour pouvoir les désigner précisément
 - auxquels on associe des propriétés

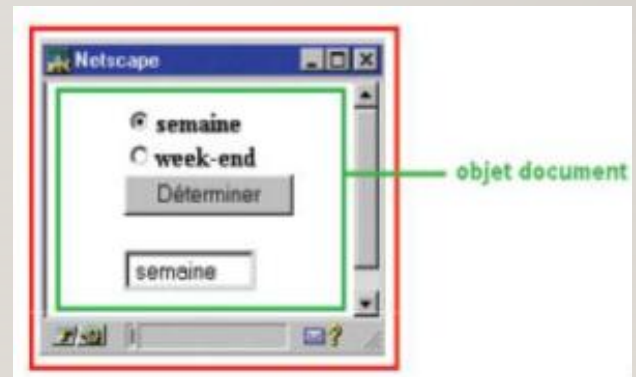
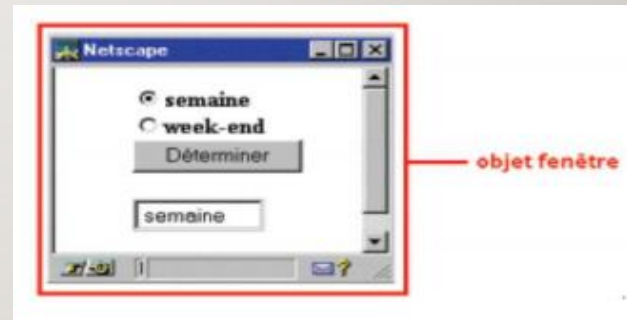


Les objets d'interface

- La **fenêtre** du navigateur se nomme "**window**".
- La **page HTML** est un autre objet, que l'on appelle "**document**".
- Un **formulaire** à l'intérieur d'un "**document**", est aussi un objet.
- Les éléments d'un formulaire sont accessibles via leur noms : **nom**, **prenom**, **age**, ...
- Un **lien hypertexte** dans une page HTML, est encore un autre objet. Il s'appelle "**link**".
- etc...
- Les objets javascript peuvent réagir à des "**Evénements**".
- Tous les navigateurs ne supportent pas les mêmes objets

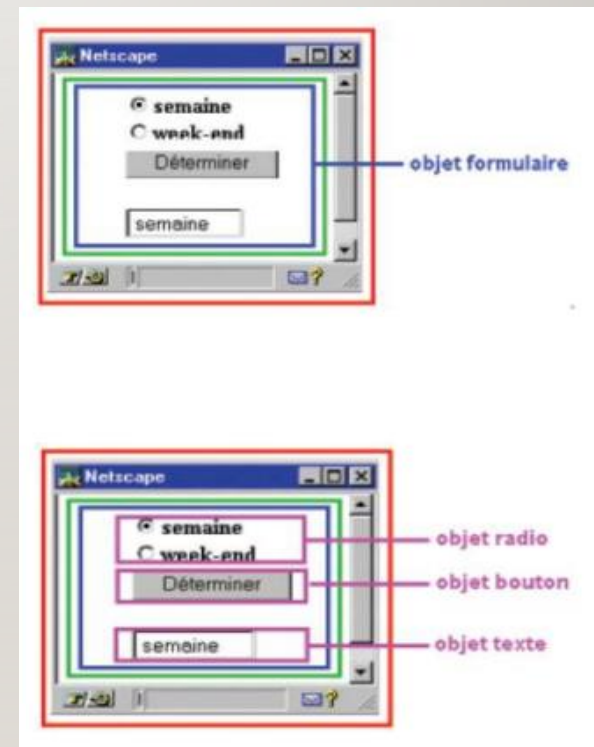
Hiérarchie des objets

- L'objet le plus **haut** dans la hiérarchie est l'objet de la classe **window** (fenêtre)
- Dans cette fenêtre, il y a un document HTML : c'est l'objet **document**. Donc, L'objet fenêtre contient l'objet document



Hiérarchie des objets

- Dans ce document, on trouve un **formulaire** au sens HTML. Donc, l'objet fenêtre contient un objet document qui lui contient un objet formulaire
- Dans ce formulaire, on trouve trois objets :
 - l'objet radio,
 - l'objet bouton,
 - l'objet texte.



L'accès aux objets

- L'accès aux objets se fait par une notation par points.
- comme il peut y avoir plusieurs formulaires et plusieurs boutons dans chaque formulaire, certains objets sont automatiquement numérotés.
 - accéder au premier bouton du premier formulaire d'une page web se fait par la notation suivante : `window.document.forms[0].checkbox[0]`
`document.forms[0].checkbox[0]` //correcte sans window
 - Il est aussi possible d'utiliser le nom de l'objet directement. Ainsi, si le formulaire s'appelle `enquete` et le bouton s'appelle `courrier`, il sera possible d'y accéder de la manière suivante : `window.document.enquete.courrier`

L'accès aux objets

- Enfin, il est aussi possible d'utiliser la notation :
`window.document.form["enquête"].button["courrier"]`
- On peut également retrouver un élément en utilisant la méthode `document.getElementById()` pour accéder à retrouver un **élément précis** par son id
- `innerHTML` est une propriété de tout élément HTML qui désigne le contenu qui se trouve entre la balise entrante et la balise fermante.
 - `let contents = myElement.innerHTML`; Cela vous permet de regarder le balisage HTML des nœuds de contenu de l'élément.
 - On peut également changer le contenu de cet élément en attribuant une valeur à cette propriété: `myElement.innerHTML=""`;

193

GESTION DES ÉVÉNEMENTS



Les événements

- Les évènements sont des actions qui se produisent et auxquelles on va pouvoir répondre en exécutant un code.
- Par exemple, afficher ou cacher du texte suite à un clic d'un utilisateur sur un élément, changer la taille d'un texte lors du passage de la souris d'un utilisateur sur un élément.
- Les évènements et leur prise en charge sont l'un des mécanismes principaux du JavaScript qui vont nous permettre d'ajouter un vrai dynamisme à nos pages Web.

Les événements

- En JavaScript, un évènement est une action qui se produit et qui possède deux caractéristiques essentielles :
- C'est une action qu'on peut « écouter », c'est-à-dire une action qu'on peut détecter car le système va nous informer qu'elle se produit ;
- C'est une action à laquelle on peut « répondre », c'est-à-dire qu'on va pouvoir attacher un code à cette action qui va s'exécuter dès qu'elle va se produire
- Un gestionnaire d'évènements est défini pour écouter et répondre à un évènement

Les événements

- En JavaScript, il existe trois grandes façons d'implémenter un gestionnaire d'évènements :
 - On peut utiliser des attributs HTML de type événement ;
 - On peut utiliser des propriétés JavaScript liées aux événements ;
 - On peut utiliser la méthode **addEventListener()**

Les événements

- Utiliser des attributs HTML de type évènement : attributs **onXXX** en HTML
- Ces attributs HTML de « type évènement » possèdent souvent le nom de l'évènement qu'ils doivent écouter et gérer précédé par « on » comme par exemple :
 - L'attribut **onclick** pour l'évènement « clic sur un élément » ;
 - L'attribut **onmouseover** pour l'évènement « passage de la souris sur un élément » ;
 - L'attribut **onmouseout** pour l'évènement « sortie de la souris d'élément » ;
 - Etc.
- Exemple :

```
<input type="text" size="30" id="email" onchange="checkEmail( )" >  
<a href="http://www.fsm.rnu.tn" onmouseover="window.status='site  
de FSM'; ">FSM  
</a>
```


Les événements

- Utiliser les propriétés JavaScript : propriétés **onXXX** des éléments
- Le code à exécuter (généralement sous forme de fonction anonyme) en cas de déclenchement de l'évènement est défini en valeur de la propriété relative à l'évènement.
 - **element.onclick=maFonction;**//sans parenthèses
 - maFonction a accès à **this**

```
element.onclick = maFonction;  
another_element.onclick = maFonction;  
function maFonction()  
{this.style.backgroundColor = grey;}
```


Les événements

- La méthode **addEventListener()** est une méthode utilisée avec des éléments
- La méthode possède deux arguments : le nom d'un événement et le code à exécuter (qui prendra souvent la forme d'une fonction) en cas de déclenchement de cet événement.

```
document.addEventListener("click" , function(){ ..... });
```

Les événements

- Les événements sur la souris

onclick	Click sur l'élément
ondblclick	Double-click sur l'élément
onmousedown	Appui sur un bouton de la souris
onmousemove	Déplacement de la souris au-dessus de l'élément
onmouseover	Arrivée de la souris sur l'élément
onmouseout	Sortie de la souris de l'élément
onmouseup	Relachement du bouton de la souris

Les événements

- Les événements sur le clavier

onkeydown	Appui sur une touche
onkeypress	Maintien de la touche enfoncée
onkeyup	relachement de la touche

- Les évènements sur les objets/fenêtre

onabort	Chargement d'une image interrompu
onerror	Chargement d'une image qui échoue
onload	Chargement d'un élément, d'un document
onresize	Redimensionnement du document
onscroll	Scroll sur le document
onunload	Fermeture de la page

Les événements

- Les événements sur les formulaires

onblur	Un élément de formulaire perd le focus
onchange	Le contenu d'un élément change
onfocus	Un élément du formulaire prend le focus
onreset	Le formulaire est réinitialisé
onselect	Selection de texte dans un formulaire
onsubmit	Soumission de formulaire

Exercice 3

- Ecrire un fichier **tva.html** permettant de calculer la TVA sur un produit : La tva sur un produit agricole est 0%, et sur un produit non agricole est 9%.

Prix HT du produit :

Produit agricole ☐ Autre ☐

```
<body>
  Prix HT du produit :
  <form name = "FormulaireTVA" >
    <input type = "text" name = "prixht" > <br >
    Produit agricole <input type = "radio" name = "nature" value = "agricole" >
    Autre <input type = "radio" name = "nature" value = "autre" >
    <input type = "button" value = "Valider" >
  </form>
</body>
```

Exercice 4

- Ecrire un fichier **multip.html** permettant de réviser quelques multiplications :
- Lorsque l'utilisateur clique sur « **valider** », une boîte de dialogue affiche le nombre de réponses correctes.

3*2 : 6
6*7 : 44
4*4 : 16
3*5 : 12

Valider

Cette page indique

Vous avez 2 bonnes réponses, correction ?

OK Annuler

```
<body>
  <form name = "formMult" >
    3*2 : <input type = "text" name = "resultat1" id="res1"> <br >
    6*7 : <input type = "text" name = "resultat2" id= "res2" > <br >
    4*4 : <input type = "text" name = "resultat3" > <br>
    3*5 : <input type = "text" name = "resultat4" > <br>
    <input type = "button" value = "Valider" >
  </form >
</body>
```


Exercice 5

- Ecrire une page **multiplication.html** permettant d'afficher la table de multiplication pour un nombre donné par l'utilisateur.

```
<body>
  <h3>Table de multiplication par N</h3>
  <form name="TabMul" method="post" action="traitement.php">
    <input type="text" name="nombre">
    <input type="button" value="calculer"><br><br>
    <textarea cols="15" rows="12" name="table" id="t"></textarea>
  </form>
</body>
```

Table de multiplication par N

7*1=7
7*2=14
7*3=21
7*4=28
7*5=35
7*6=42
7*7=49
7*8=56
7*9=63

Exercice 6

- Ecrire une page **premiers.html** qui permet de sélectionner les nombres premiers parmi un ensemble d'entiers

The image shows two overlapping screenshots of a web form titled "Quels sont les nombres premiers". The form contains five checkboxes for the numbers 209, 223, 517, 647, and 751, and a "vérifier" button. In the top screenshot, the checkboxes for 209, 223, and 647 are checked, and a message box says "Cette page indique Désolé, continuez à chercher". In the bottom screenshot, the checkboxes for 223, 647, and 751 are checked, and a message box says "Cette page indique C'est la bonne réponse".

Quels sont les nombres premiers

☒ 209
☒ 223
☐ 517
☒ 647
☐ 751
vérifier

Cette page indique
Désolé, continuez à chercher

Quels sont les nombres premiers

☐ 209
☒ 223
☐ 517
☒ 647
☒ 751
vérifier

Cette page indique
C'est la bonne réponse

OK

```
<body>  
<h3>Quels sont les nombres premiers parmi ces cinq entiers</h3>  
<form name = "f" >  
  <input type = "checkbox" name = "c1" value = "1" >209<br>  
  <input type = "checkbox" name = "c2" value = "2" >223<br>  
  <input type = "checkbox" name = "c3" value = "3" >517<br>  
  <input type = "checkbox" name = "c4" value = "4" >647<br>  
  <input type = "checkbox" name = "c5" value = "5" >751<br>  
  <input type="button" name="bouton" value="vérifier">  
</form>  
</body>
```