esprit
Se former autrement
HONORIS UNITED UNIVERSITIES

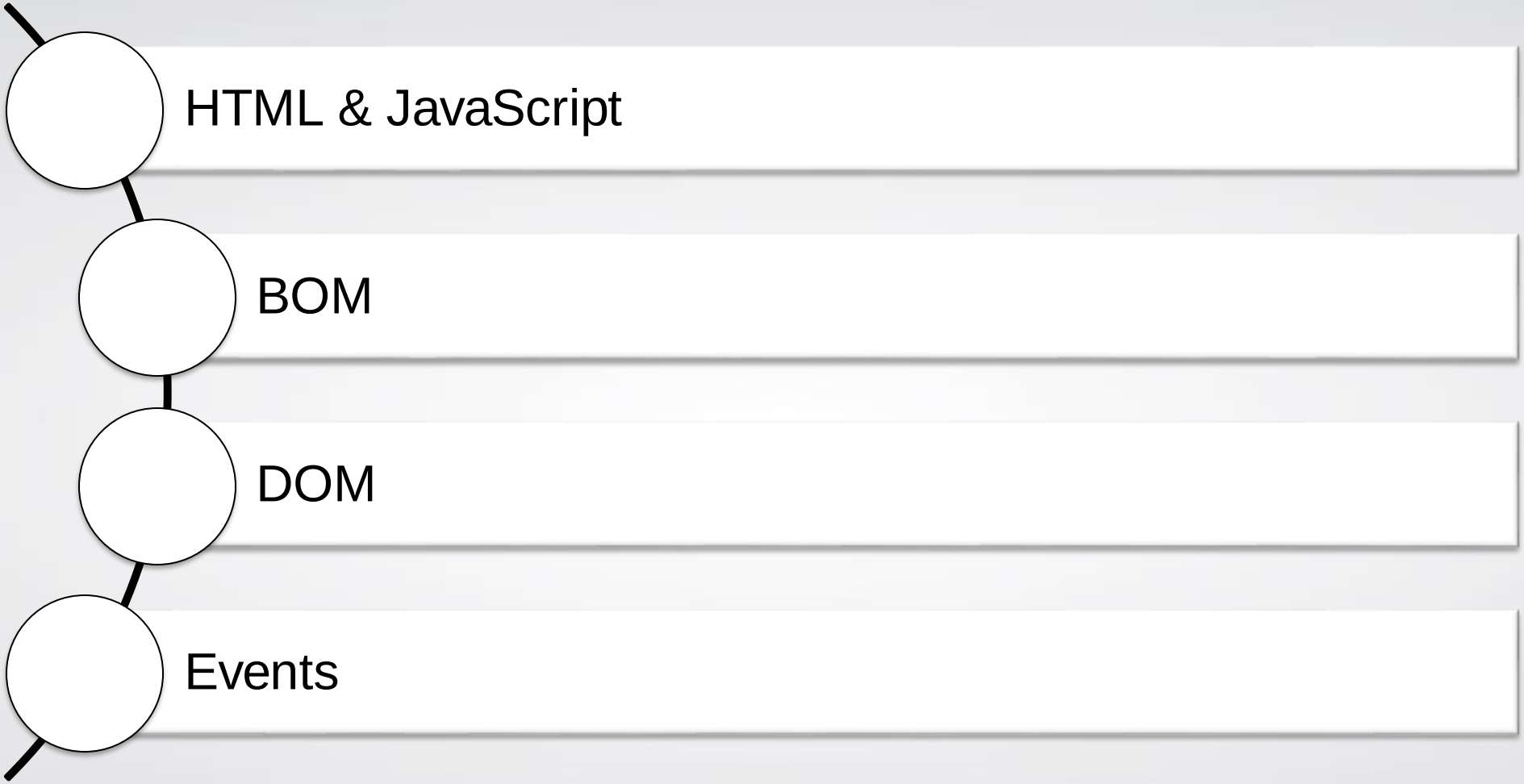
Chapitre 3: JavaScript

UP Web

AU: 2023/2024

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
EUR-ACE® Délivrée par la Commission des Titres d'Ingénieur
CONFÉRENCE DES GRANDES ÉCOLES
cdio CONCEIVE DESIGN IMPLEMENT OPERATE

Plan



Objectifs

- Manipuler le DOM.
- Différencier entre les événements.
- Ecrire un script en utilisant les fonctions prédéfinis, événement…

Prérequis

- HTML

HTML & JavaScript

Console développeur

- Firefox: Ctrl+Shift+K
- Chrome / Edge: Ctrl+Shift+I

Balise HTML

- Cliquez-moi !

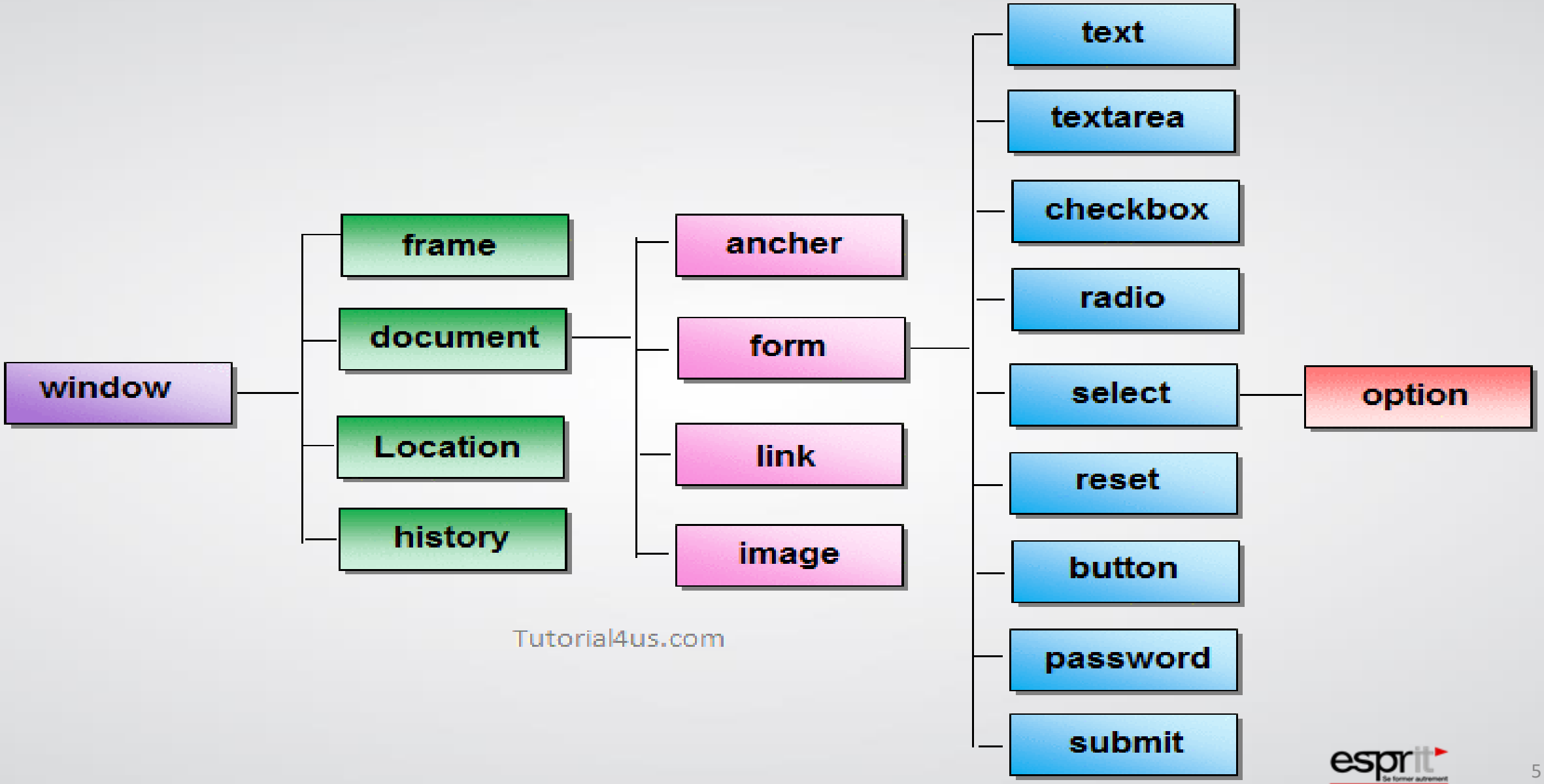
Code HTML (interne)

- <script> … </script>

Fichier séparé (externe)

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

BOM: Browser Object Model



DOM: Document Object Model

- Structure arborescente créée par le navigateur.
- Facilite l'accès à la structure HTML.
- Le navigateur utilise le DOM pour appliquer le style et corriger les éléments.
- Le développeur utilise le DOM pour manipuler la page.

DOM: Document Object Model

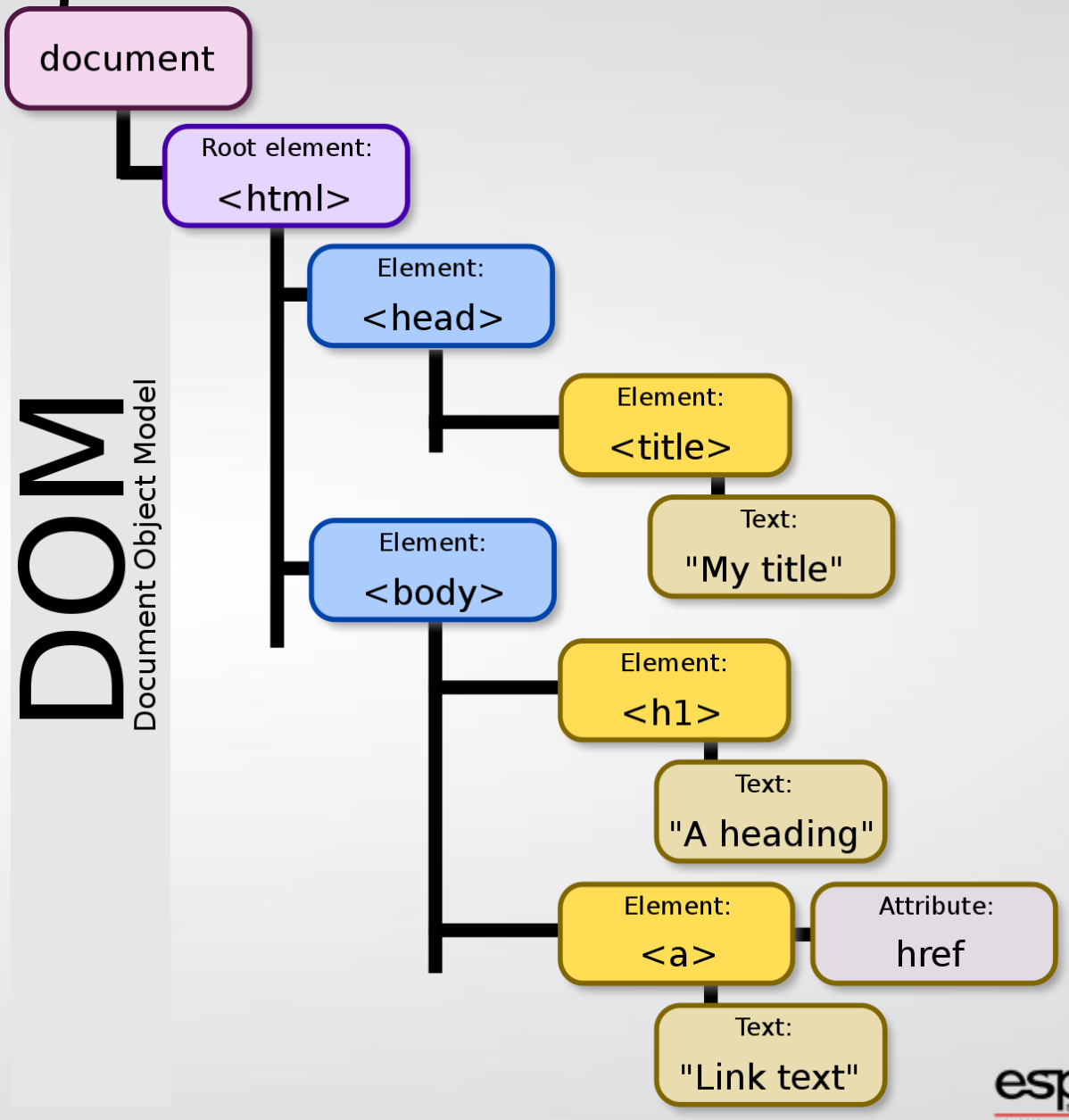
Exemple

```
<html Lang="en">
    <head>
        <title>My title</title>
    </head>
    <body>
        <h1>A heading</h1>
        <a href="">Link text</a>
    </body>
</html>
```

DOM: Document Object Model

Exemple

- Le DOM correspondant au code précédent ressemble à cette arborescence ➔
- Chaque élément et donnée texte forme un nœud (element node) de l'arbre.



DOM: Document Object Model

Propriétés d'un nœud

Propriétés	Explication
childNodes	La liste des nœuds enfants
firstChild	Le premier nœud enfant
lastChild	Le dernier nœud enfant
nextSibiling	Le prochain nœud du même niveau
parentNode	Le nœud parent
previousSibiling	Le nœud précédent du même niveau
nodeName	Le nom du nœud
nodeValue	La valeur ou contenu du nœud
nodeType	Le type du nœud
innerHTML	Le contenu littéral du nœud

DOM: Document Object Model

Méthodes d'un nœud

Méthodes	Explication
createElement()	Permet de créer un nouvel élément HTML dans le document
createTextNode()	Permet de créer un nœud texte
appendChild()	Permet d'ajouter l'élément créé au document comme le dernier nœud enfant de l'élément parent.
removeChild()	Permet de supprimer un nœud.

DOM: Document Object Model

Manipuler le DOM

Méthodes	Explication	Syntaxe
querySelector()	Retourne le premier élément dans le document correspondant au sélecteur(s) spécifié(s) ou null	element = document.querySelector (sélecteur);
querySelectorAll()	Retourne uncontenant la liste des éléments du document qui correspondent au sélecteur(s) spécifié(s)	element = document.querySelectorAll (sélecteur);
getElementById()	Renvoie un objet (élément) qui représente l'élément portant l'id spécifié.	element = document.getElementById (id);

DOM: Document Object Model

Manipuler le DOM

Méthodes	Explication	Syntaxe
getElementsByTagName()	Renvoie un tableau qui contient tous les éléments ayant un type donné.	elements = document.getElementsByTagName (name);
getElementsByClassName()	Renvoie un tableau qui contient tous les éléments portant le nom de classe spécifié.	elements = document.getElementsByClassName (className);
getElementsByName()	Renvoie un tableau qui contient tous les éléments portant le « name » spécifié.	elements = document.getElementsByName (name);

Events

- Un événement est un signal généré suite à une action sur les éléments de l'interface DOM visualisés dans un navigateur.
- **Par exemple**
  - un clic de souris,
  - le curseur de la souris qui survole une zone,
  - le clic sur un bouton
  - la saisie d'un texte dans un formulaire

Events

Un événement en JavaScript est représenté dans une balise HTML en utilisant une **action** et une **fonction** que l'on nomme un callback .

L'action: une propriété qui représente le gestionnaire d'évènement de l'élément courant (onClick , OnMouseMove ...)

La fonction callback: est appelée à chaque fois que l'action sur un élément est exécutée.

Events

```
<html>
<head>
    <meta charset="utf-8">
    <title> event</title>
 <script src="script.js"></script>
</head>
<body>
    <button onclick="verification()"> Valider</button>
</body>
</html>
```

```
function verification()
{
    alert("boutton de verification");
}
```

nom

callback

Events

Gérer les événements avec la méthode JavaScript
element.addEventListener(event, function)

- La méthode addEventListener() est appelée par un élément, le document lui-même et une Window ou par n'importe quel objet prenant en charge les évènements.
- Lorsque l'événement se produit, la fonction callback est exécutée en lui passant un argument. Le code est placé au niveau de cette fonction.

Events

Gérer les événements avec la méthode JavaScript

element.addEventListener(event, function)

Chaine de caractères qui spécifie le nom de l'événement.

Spécifie la fonction à exécuter lorsque l'événement se produit.

```
<body>
<p> Example CLICK </p>
<button id="myBtn">Tester le bouton</button>
</body>
<script>
var b =document.querySelector('button');

b.addEventListener('click',function() { window.alert('Clic sur le paragraphe') });
</script>
</html>
```

nom de l'événement

la fonction à exécuter

Events

Exemples d'événements

Nom de l'événement	Description
load	Fin de chargement de la page web
click	Clic sur un element
dbclick	Double clic sur un element
keydown	Une touche est appuyée
keypress	Une touche est maintenue enfoncée
keyup	Une touche est relachee
mouseenter	Le curseur entre au dessus d'un élément
mouseleave	Le curseur quitte l'element
select	Selection d'une option dans un select
change	Changement de valeur sur un select, un checkbox
Submit	Soumission d'un formulaire
focus	L'élément reçoit le focus
blur	L'élément perd le focus

Events

Load

L'événement **load** se produit lorsque le navigateur a terminé le chargement de la page.

```
<html>
<head>
    <meta charset="utf-8">
    <title> event</title>
 <script src="script.js"></script>
/head>
body onload="loadB()">

   <button> Valider</button>
    <p>Qu'est-ce qu'un événement ?</p>

/body>
/html>
```

```
function loadB(){
    console.log("loading....")
}
```

Events

Click

L'événement **click** se produit lorsque l'utilisateur clique sur un élément ,un bouton, un formulaire ou sur le document .

```
<body>
<p> Example CLICK </p>
<button id="myBtn">Tester le bouton</button>
</body>
<script>
    // avec getElementById
var b =document.getElementById("myBtn")
b.addEventListener("click", function()  { window.alert('Hello world') });
    // avec querySelector
var b =document.querySelector('button');
b.addEventListener('click',function() { window.alert('Clic sur le paragraphe') });
</script>
</html>
```

Events

KeyUp / KeyDown

L'événement **KeyUp** se produit au moment où l'utilisateur relâche une touche du clavier.

L'événement **KeyDown** se produit au moment où l'utilisateur enfonce ou clique une touche du clavier.

```
<input type="text" onkeyup="showCount()" onkeydown="addCount()">
```

```
var i=0;
function addCount(){
    i=i+1;
}
function showCount(){
    console.log(i);
}
```

Events

MouseLeave

L'événement **MouseLeave** ne sera exécuté que lorsque le curseur se déplace vers un élément.

```
<body>
    <input type="text" id="name" onmouseleave="ChangeValue()" >
</body>
<script>
    function ChangeValue(){
    document.getElementById("name").value= "Have a nice day!";

    }
</script>
```

Events

Change

L'événement **Change** se produit lorsque l'utilisateur modifie la valeur d'un élément (Select / Text / TextArea).

```
<p> Selectionner une Classe </p>
<select id="Select" >
  <option value="2A1">2A1</option>
  <option value="2A2">2A2</option>
  <option value="2A3">2A3</option>
  <option value="2A4">2A4</option>
</select>
<p id="classe"></p>
<script>
  var x = document.getElementById("Select");
x.addEventListener("change", function()  {
var value = document.getElementById("Select").value;
  document.getElementById("classe").innerHTML = "Tu as selectionné: " + value;});
</script>
```

Events

Submit

L’événement **Submit** se produit lorsque l’utilisateur envoie des données à partir d’un formulaire.

Remarque: on utilise **e.preventDefault()** pour empêcher le formulaire de s'envoyer au serveur.

Events

Submit

L'événement **Submit** se produit lorsque l'utilisateur envoie des données à partir d'un formulaire.

```
<form id="form" >
  Entrez votre nom:
  <input id="nom" type="text" name="fname">
  <input type="submit" value="Valider">
</form>
<p id="paragraphe"></p>
<script>
document.getElementById("form").addEventListener("submit", function(e){
e.preventDefault();
  var nom = document.getElementById('nom').value;
  document.getElementById("paragraphe").innerHTML = "Bienvenue: " + nom;
} );
</script>
```

Events

Blur

L’événement **Blur** se déclenche lors de la perte de focus d'un élément.

```
<input type="text" id="name" onblur="disableInput()" >
```

```
function disableInput(){
    document.getElementById("name").disabled = true;
}
```

Merci de votre attention