LIFIHM - Interactions Homme Machine
Programmation pour l'IHM - interactions

Fabien Duchateau

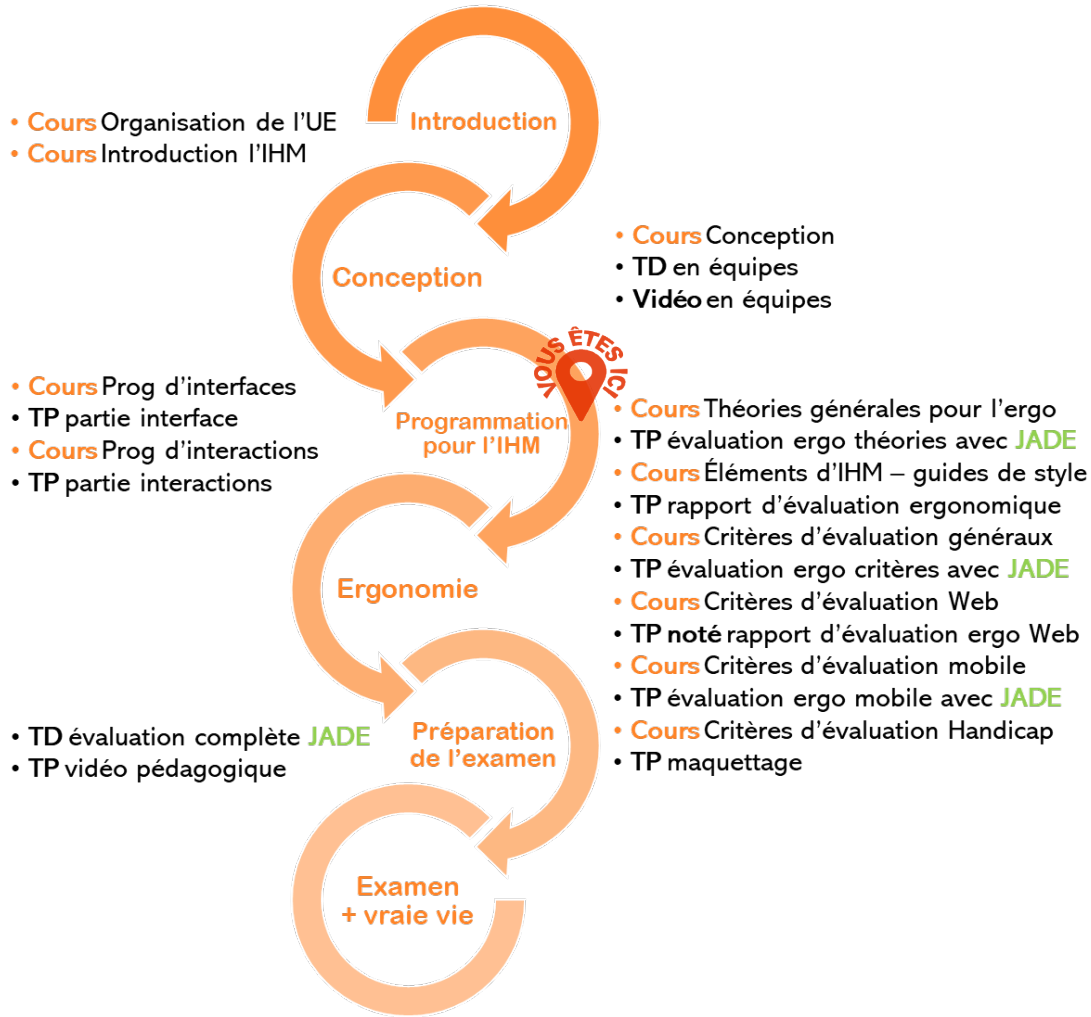
fabien.duchateau [at] univ-lyon1.fr

Université Claude Bernard Lyon 1

2021 - 2022

http://liris.cnrs.fr/stephanie.jean-daubias/enseignement/IHM/

Positionnement dans LIFIHM



Ces transparents utilisent le genre féminin (e.g., chercheuse, développeuses) plutôt que l'écriture inclusive (moins accessible, moins concise, et pas totalement inclusive)

Objectifs du cours

Rappels des cours précédents :

- Méthodes de conception pour les IHM (maquettes)
- Système de grille de Bootstrap, styles pré-définis (pour les balises HTML) et composants supplémentaires de Bootstrap

Comment programmer les interactions, notamment les événements déclenchés par des actions sur les composants ?

- Découvrir le Document Object Model (DOM), qui définit les propriétés et méthodes d'un document HTML
- Connaître les types d'événements disponibles en JavaScript et programmer les actions associées

Dans cette UE, nous nous focalisons sur des événements JavaScript

Plan

Langage JavaScript

Document Object Model et Browser Object Model

Évènements JavaScript

Types d'évènements JavaScript

Généralités

JavaScript (JS), un langage de programmation de scripts :

- Standard ECMA depuis 1997, version 8 en 2021
- Haut-niveau, dynamique, à typage faible
- Paradigme orienté objet à prototypage
- Fichiers texte avec extension **.js**, qui sont lus et interprétés par le navigateur (mais compilation possible)

---

http://www.w3schools.com/js/

http://developer.mozilla.org/fr/docs/Web/JavaScript

http://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript

http://fr.wikipedia.org/wiki/JavaScript

Généralités (2)

- JavaScript est surtout inspiré de Java (pour la syntaxe), mais aussi de Perl et Python
- Sensible à la casse (e.g., noms de variable)
- Balise `<script>...</script>` :
  - soit une référence vers un fichier JS (attribut *src*)
  - soit directement du code JS

```
11  <!-- un script dans un fichier externe -->
12  <script src="unFichier.js"></script>
13  <!-- un script directement dans la balise script -->
14  <script>
15      /* commentaire multi-lignes */
16      var uneVariable = "Hello World !"; // déclaration / affectation de variable
17      // commentaire sur une seule ligne
18      console.log(uneVariable); // affiche le texte en paramètre dans la console
19  </script>
```

Variables

- Les variables sont déclarées avec les mot-clés `var` ou `let`
- Aucune déclaration d'un type, c'est l'affectation qui détermine le type de la variable
- Types : nombre (pas d'entier !), chaîne de caractères, booléen, objet (i.e., fonction, tableau, date, expression régulière), symbole
- `null` et `undefined` quand une variable n'est pas affectée

```
22 var chaine = 'cerise';
23 var chaine2; // vaut undefined
24 console.log('chaine == null ? ' + (chaine == null)); // false
25 console.log('chaine2 == null ? ' + (chaine2 == null)); // true
26 console.log('chaine2 == undefined ? ' + (chaine2 == undefined)); // true
```

Opérateurs de comparaison

- Opérateurs traditionnels ($<$, $>$, $\leq$, $\geq$)
- Deux opérateurs d'égalité :
  - $==$ et $!=$ pour une (in-)égalité faible (conversion automatique de type)
  - $===$ et $!==$ pour une (in-)égalité stricte (sans conversion, donc les types doivent être identiques)

```
28 var nombre = 1;
29 console.log('nombre == "1" (égalité faible) ? ' + (nombre == '1')); // true
30 console.log('nombre == 1 (égalité faible) ? ' + (nombre == 1)); // true
31 console.log('nombre === "1" (égalité stricte) ? ' + (nombre === '1')); // false
32 console.log('nombre == true ? ' + (nombre == true)); // true
33 console.log('nombre === true ? ' + (nombre === true)); // false
```

http://developer.mozilla.org/fr/docs/Web/JavaScript/Les_diff%C3%A9rents_tests_d_%C3%A9galit%C3%A9

Autres opérateurs

- Arithmétiques : +, −, *, /, %, ++, −−
- Concaténation de chaines de caractères : +
- Logiques : &&, ||, !
- Type d'un objet donné : `typeof`

```
40 console.log('4 + nombre (somme) = ' + (4 + nombre)); // 5
41 console.log('chaine + " sur le gâteau" (concatenation) = ' + (chaine + ' sur le
       gâteau')); // cerise sur le gâteau
42 console.log('typeof chaine = ' + (typeof chaine)); // string
43 console.log('typeof variableInconnue = ' + (typeof variableInconnue)); // undefined
```

http://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Operators

Structures conditionnelles

- Structure traditionnelle `if … else`
- Possibilité d'ajouter des tests avec l'instruction `else if` avant le `else` final
- Bloc d'instructions entre accolades
- Aussi disponibles : `switch` et opérateur ternaire … ? … : …

```
45 if (typeof chaine === 'number')
46     console.log('chaine est de type number');
47 else if (typeof chaine === 'string')
48     console.log('chaine est de type string'); // true
49 else
50     console.log('chaine est d\'un autre type');
```

---

http://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Contr%C3%B4le_du_flux_Gestion_des_erreurs

Boucles

Boucles similaires aux autres langages :

- `while(condition d'arrêt) { actions }`
- `for(initialisation; condition d'arrêt; itération) { actions }`
- `do { actions } while(condition d'arrêt);`

```
52 var compteur = 0;
53 while (compteur < 3) {
54     console.log('Compteur du "while" vaut ' + compteur); // trois affichages
55     compteur++;
56 }
57 for (var i = 1; i <= 5; i++)
58     console.log('Itération du "for" numéro ' + i); // cinq affichages
```

http://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Boucles_et_it%C3%A9ration

Collections

- Avec index numériques (tableaux)
  - types de données différents
  - nombreuses propriétés et méthodes (e.g., `length`, `push()`, `concat()`)
  - tableaux multi-dimensionnels par imbrication de tableaux
- Avec clés (e.g., Set, Map, objets)

```
66 var unTableau = ["La vérité", 4, "Timbré", 6.5, "Sourcellerie"];
67 console.log('Taille du tableau : ' + unTableau.length); // 5
68 console.log('Troisième élément du tableau : ' + unTableau[2]); // Timbré
69 for (var i = 0; i < unTableau.length; i++)
70     console.log(unTableau[i]); // La vérité   4   Timbré   6.5   Sourcellerie
71 unTableau.forEach(function (element) {
72     console.log(element); // La vérité   4   Timbré   6.5   Sourcellerie
73 });
```

---

http://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Collections_index%C3%A9es

http://developer.mozilla.org/fr/docs/Web/JavaScript/Guide/Collections_avec_cl%C3%A9s

Fonctions

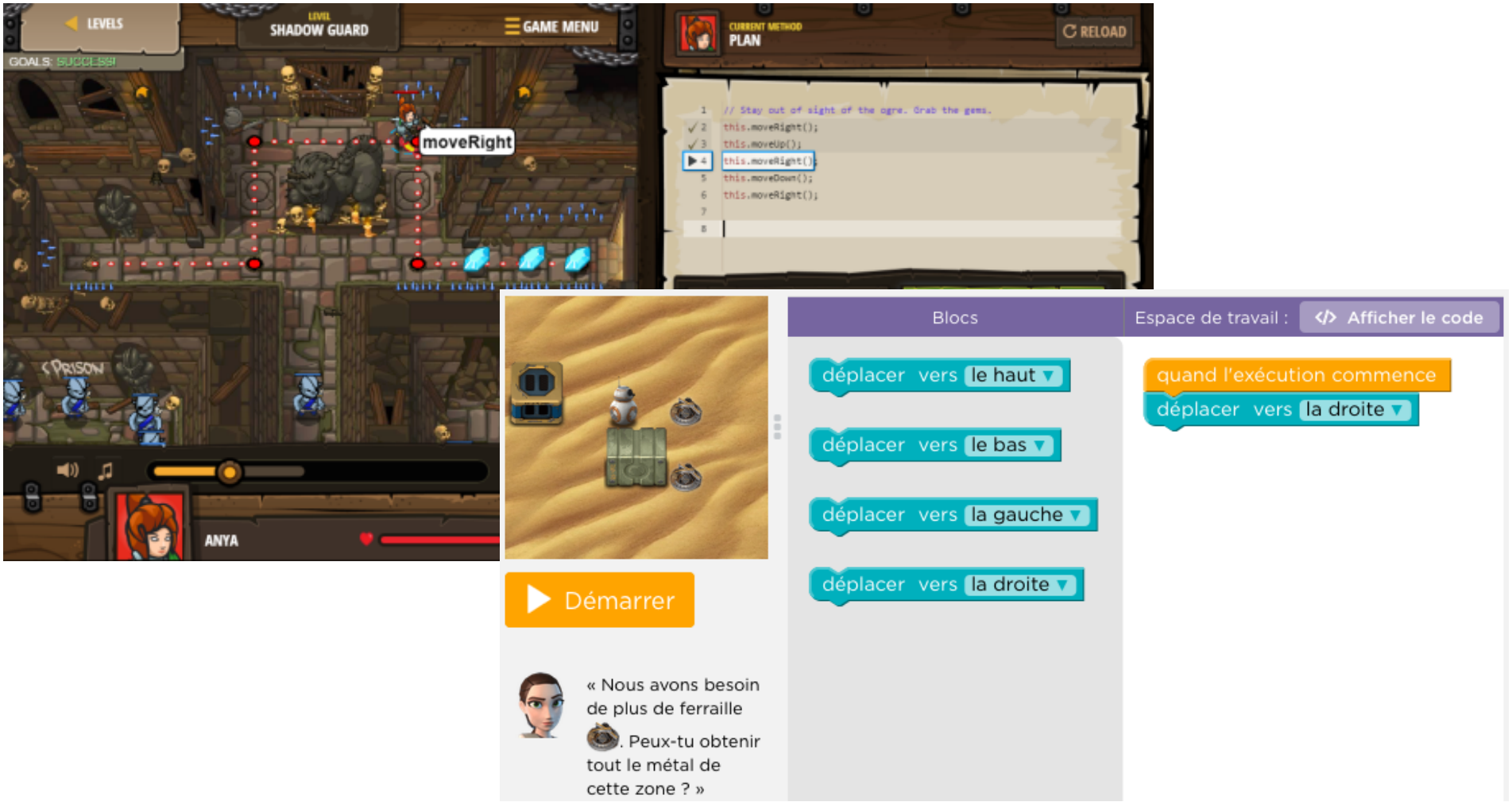
Fonctions similaires aux autres langages, mais sont considérées comme des objets :

- Mot-clé `function`, suivi du nom de la fonction et des paramètres optionnels
  `function maFonction(paramètre1, paramètre2) { … }`
- Valeurs de retour avec le mot-clé `return`

```
60 function ajoute(x, y) {
61     var total = x + y;
62     return total;
63 }
64 console.log('Somme de 143 et 682 = ' + ajoute(143, 682)); // 825
```

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Fonctions

En résumé



---

http://codecombat.com/

http://studio.code.org/

Plan

Un modèle de données

Le DOM (Document Object Model) permet aux langages de programmation de manipuler un document (i.e., une page web)

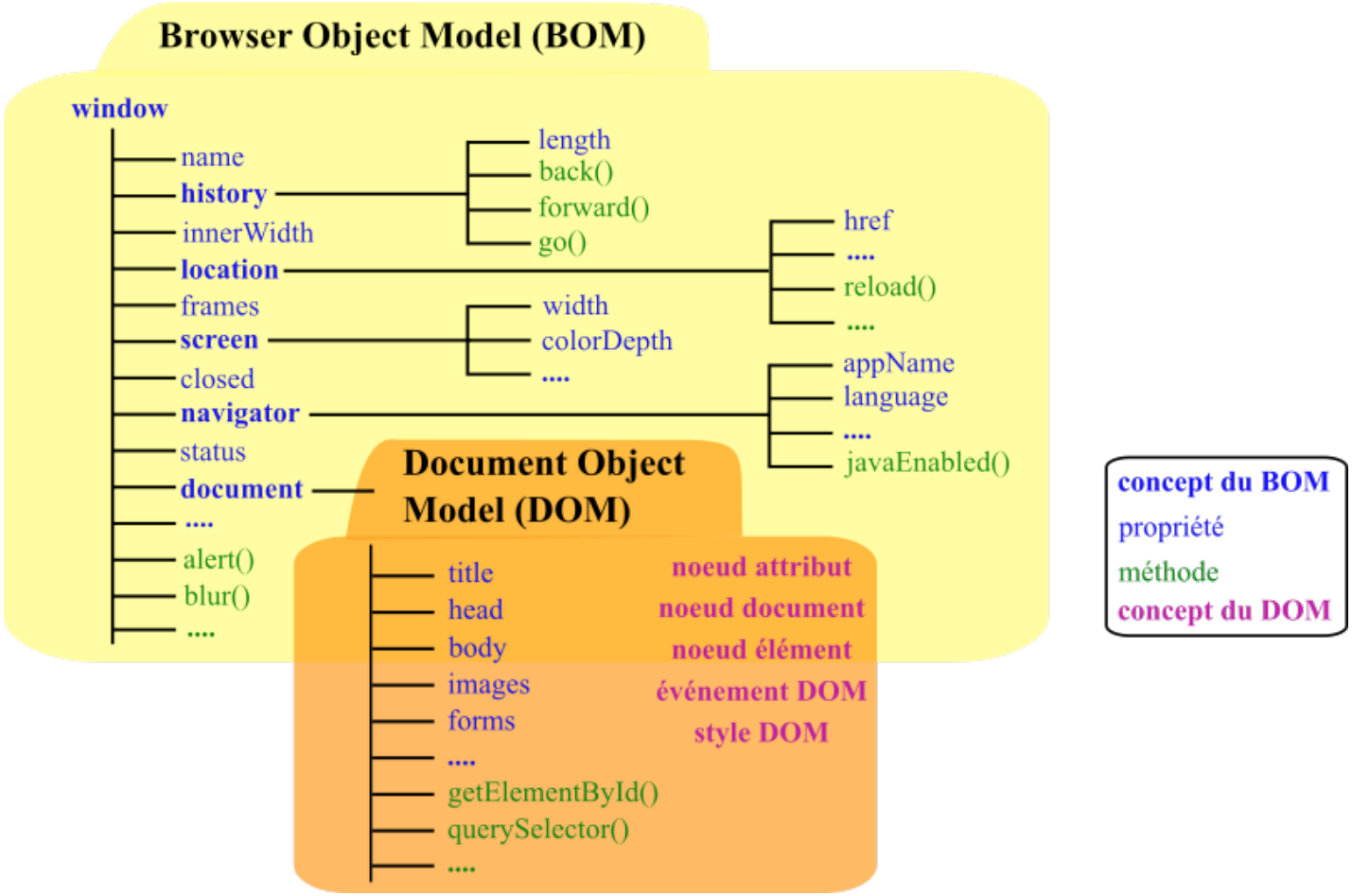
Le DOM est inclus dans un environnement, le BOM (Browser Object Model) composé de :

- **window**, la fenêtre courante
- **navigator**, le navigateur qui affiche la fenêtre
- **screen**, l'écran qui permet de visualiser la fenêtre
- **history**, l'historique de la fenêtre
- **location**, l'URL de la fenêtre
- **document**, le document HTML (DOM)

À l'exception du DOM, il n'y a pas de standard pour les objets du BOM, mais ils sont supportés par les navigateurs récents

Un modèle de données (2)

Les objets du BOM (et le DOM) incluent des **propriétés** (e.g., résolution de l'écran) et des **méthodes** (e.g., rafraichir la page)



Le DOM (objet `document`)

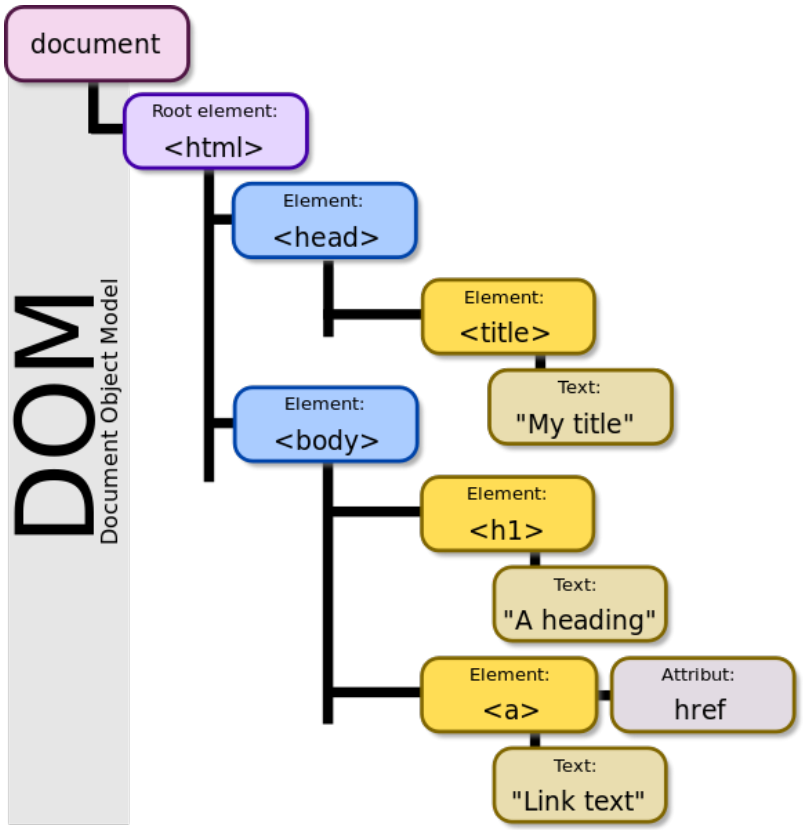
- Le DOM représente un document HTML sous forme d'arbre :
  - la racine correspond au nœud document (`<html>`)
  - une douzaine de types de nœuds :
    - *document* (le document HTML)
    - *élément* (balises HTML)
    - *attribut* (attributs HTML)
    - *textuels* (textes entre balises)
    - *commentaire*
    - ...
  - chaque type de nœud dispose de ses propres propriétés et méthodes
- Chaque nœud du DOM, représenté par un objet en JavaScript, est donc **directement manipulable**

---

https://developer.mozilla.org/fr/docs/DOM

https://fr.wikipedia.org/wiki/Document_Object_Model

Exemple de DOM - représentation

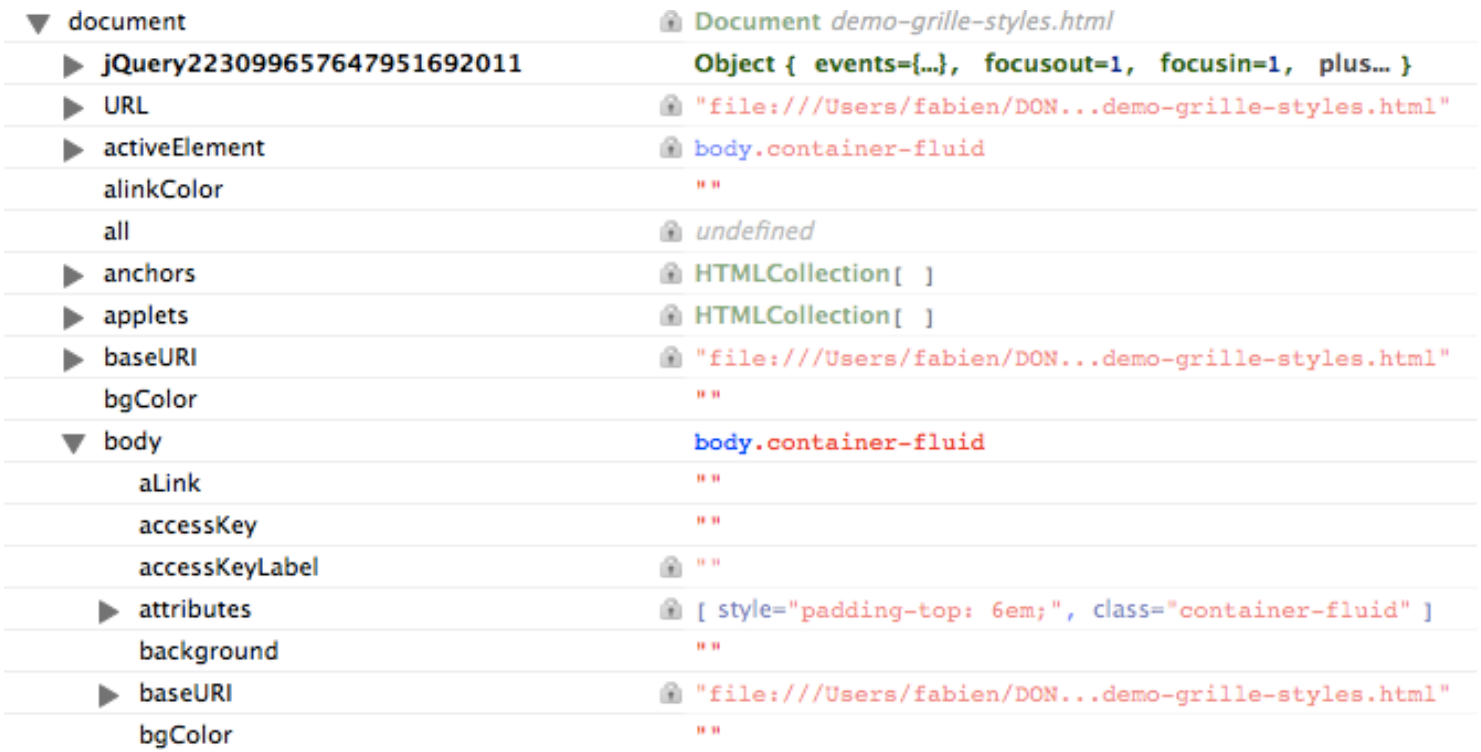


Représentation du DOM d'une page web, avec des nœuds élément, des nœuds textuels et un nœud attribut

---

https://commons.wikimedia.org/wiki/File:DOM-model.svg

Exemple de DOM - inspecteur d'objets



Affichage du DOM dans Firebug : le nœud racine `document` avec ses propriétés/méthodes, parmi lesquelles son nœud enfant `body` (lui aussi avec ses propriétés/méthodes)

Quelques propriétés/méthodes du nœud document

- **document.head** retourne le nœud <head>, **document.body** retourne le nœud <body>
- **document.images** retourne une collection contenant tous les nœuds <img>
- **document.forms** retourne une collection contenant tous les nœuds <form>
- Pour retourner des éléments selon un sélecteur CSS :
  - **document.querySelector(css)** retourne le premier élément
  - **document.querySelectorAll(css)** retourne une collection de tous les éléments qui satisfont au sélecteur *css*

```
267 var nbImages = document.images.length;
268 console.log('Nombre d\'images sur cette page : ' + nbImages); // 1
269 console.log('Nombre de formulaires sur cette page : ' + document.forms.length); // 2
```

http://www.w3schools.com/jsref/dom_obj_document.asp

Quelques propriétés/méthodes du nœud document (2)

- **`document.getElementById(i)`** retourne l'élément dont l'attribut *id* vaut *i*
- **`document.getElementsByName(n)`** retourne une collection d'éléments dont l'attribut *name* vaut *n*
- **`document.getElementsByClassName(c)`** retourne une collection d'éléments dont l'attribut *class* contient *c*
- **`document.getElementsByTagName(t)`** retourne une collection d'éléments dont le nom de balise est *t*

```
270 var body = document.querySelector("body");
271 console.log('Nombre de noeuds enfants du body : ' + body.childNodes.length); // 20
272 console.log('Nombre d\'éléments enfants du body : ' + body.children.length); // 10
273 var sections = document.getElementsByTagName('section');
274 console.log('Nombre d\'éléments section : ' + sections.length); // 8
275 var boutonVal = document.getElementById('boutonValider');
276 console.log('Nombre de classes du bouton : ' + boutonVal.classList.length); // 2
```

Quelques propriétés des nœuds élément

- **`element.id`** définit ou retourne l'identifiant de *element*
- **`element.style`** définit ou retourne un objet contenant le style CSS de *element* (voir DOM style)
- **`element.innerHTML`** et **`element.textContent`** définissent ou retournent le texte de *element* (avec et sans sous-balises)
- **`element.parentElement`** et **`element.parentNode`** retournent l'élément ou le noeud parent

```
278 var spanObj = document.querySelector("span");
279 console.log('ID de "spanObj" : ' + spanObj.id); // span1
280 console.log('Style de "spanObj" : ' + spanObj.style.fontStyle); // italic
281 console.log('Balise du parent du "spanObj" : ' + spanObj.parentElement.tagName); // P
282 console.log('Texte de "spanObj" : ' + spanObj.innerHTML); // ?
283 spanObj.innerHTML = 'Outils > développement web'; // modification du texte
284 console.log('Nouveau texte de "spanObj" : ' + spanObj.innerHTML); // Outils >
      développement web
```

---

http://www.w3schools.com/jsref/dom_obj_all.asp

http://www.w3schools.com/jsref/dom_obj_style.asp

Quelques propriétés des nœuds élément (2)

- **`element.attributes`** retourne une collection contenant les attributs de *element*
- **`element.nodeType`** retourne le type de noeud (1 à 12)
- **`element.className`** retourne la valeur de l'attribut *class*
- **`element.classList`** retourne une collection contenant les classes de *element*
- **`element.children`** et **`element.childNodes`** retournent une collection avec les éléments ou les nœuds enfant de *element*

```
285 var classes = boutonVal.classList;
286 for (var i = 0; i < classes.length; i++)
287     console.log('Classe ' + i + ' du bouton : ' + classes[i]); // btn, btn-primary
288 var classe = boutonVal.className;
289 console.log('ClassName du bouton : ' + classe); // btn btn-primary
```

---

http://www.w3schools.com/jsref/prop_node_nodetype.asp

Quelques méthodes des nœuds élément

- `element.querySelector(css)`, `element.querySelectorAll(css)`, `element.getElementsByTagName(t)`, etc.
- `element.hasAttribute(a)` et `element.getAttribute(a)`
- `element.createElement(tag)`, `element.createTextNode(chaine)` et `element.createAttribute(nom)`
- `element.appendChild(noeud)`

```
290 var texte = document.createTextNode(" Le navigateur Firefox est recommandé.");
291 document.querySelector("p").appendChild(texte);
292 var label1 = document.querySelector("label");
293 if (label1.hasAttribute("for")) // true
294     console.log('Valeur de "for" de label1 : ' + label1.getAttribute("for")); //
          input1
```

https://developer.mozilla.org/fr/docs/Web/API/Document/createTextNode

Quelques propriétés/méthodes des nœuds `attribut`

- **`attribut.name`** retourne le nom de l'attribut
- **`attribut.value`** définit ou retourne la valeur de l'attribut
- **`attribut.isId`** retourne *true* si l'attribut est un identifiant
- **`element.attributes.setNamedItem(a)`** et **`element.attributes.removeNamedItem(a)`** permettent d'ajouter ou supprimer l'attribut *a* à l'élément *element*

```
296 var att1 = boutonVal.getAttributeNode("id");
297 console.log("Nom de l\'attribut : " + att1.name); // id
298 console.log("Valeur de l\'attribut : " + att1.value); // boutonValider
299 console.log("L\'attribut est ID ? " + att1.isId); // undefined (pb support)
300 var input1 = document.querySelector("input");
301 input1.attributes.removeNamedItem("placeholder"); // bouton sans placeholder
```

---

http://www.w3schools.com/jsref/dom_obj_attributes.asp

Exemple d'ajout de valeurs dans un tableau

```
307 <table class="table table-striped">
308  <thead>
309   <tr><th>Nom</th><th>Âge</th></tr>
310  </thead>
311  <tbody id="tableBody" style="text-transform: capitalize;">
312   <tr><td>alice</td><td>18</td></tr>
313   <tr><td>bob</td><td>21</td></tr>
314   <tr><td>chloé</td><td>24</td></tr>
315  </tbody>
316 </table>
317 <button class="btn btn-primary" onclick="addRow(event);"
        >Ajouter une ligne</button>
318
319 <script>
320  function addRow(evt) {
321   evt.preventDefault(); // annule action par défaut
322   let tbody = document.getElementById("tableBody");
323   let name = Math.random().toString(36).replace(/[^a-z]+/g, ''
         ).substr(0, 5);
324   let age = parseInt(Math.random() * 100);
325   var newRow = tbody.insertRow();
326   var nameCell = newRow.insertCell();
327   nameCell.appendChild(document.createTextNode(name));
328   var ageCell = newRow.insertCell();
329   ageCell.appendChild(document.createTextNode(age));
330  }
331 </script>
```

Nom	Âge
Alice	18
Bob	21
Chloé	24

Ajouter une ligne

https://developer.mozilla.org/fr/docs/Web/API/Event/preventDefault

Exemple d'ajout de valeurs dans un tableau

```
307 <table class="table table-striped">
308  <thead>
309   <tr><th>Nom</th><th>Âge</th></tr>
310  </thead>
311  <tbody id="tableBody" style="text-transform: capitalize;">
312   <tr><td>alice</td><td>18</td></tr>
313   <tr><td>bob</td><td>21</td></tr>
314   <tr><td>chloé</td><td>24</td></tr>
315  </tbody>
316 </table>
317 <button class="btn btn-primary" onclick="addRow(event);"
        >Ajouter une ligne</button>
318
319 <script>
320  function addRow(evt) {
321   evt.preventDefault(); // annule action par défaut
322   let tbody = document.getElementById("tableBody");
323   let name = Math.random().toString(36).replace(/[^a-z]+/g, ''
          ).substr(0, 5);
324   let age = parseInt(Math.random() * 100);
325   var newRow = tbody.insertRow();
326   var nameCell = newRow.insertCell();
327   nameCell.appendChild(document.createTextNode(name));
328   var ageCell = newRow.insertCell();
329   ageCell.appendChild(document.createTextNode(age));
330  }
331 </script>
```

Nom	Âge
Alice	18
Bob	21
Chloé	24

Ajouter une ligne

Nom	Âge
Alice	18
Bob	21
Chloé	24
Bixsc	99

Ajouter une ligne

https://developer.mozilla.org/fr/docs/Web/API/Event/preventDefault

En résumé

Nombre de propriétés et méthodes des objets du BOM :

- 4 pour `history`
- 6 pour `screen`
- 12 pour `navigator`
- 12 pour `location`
- 56 pour `window`
- Des centaines pour `document` (DOM), dont des **propriétés liées aux évènements**

---

http://www.w3schools.com/jsref/

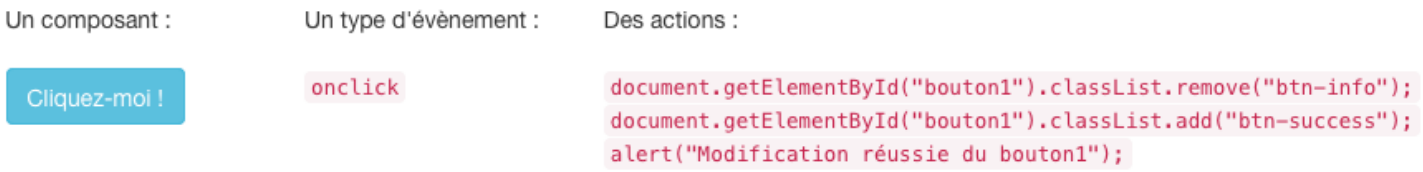
Plan

Généralités

Rappel : l'utilisatrice interagit avec une application en agissant sur ses composants (e.g., champ texte, liste déroulante, menu)

Pour créer un évènement, il faut :

- Choisir un **type d'évènement** (approprié) pour le composant
- Définir les **actions** qui seront exécutées au déclenchement de l'évènement
- Créer un **lien** entre le composant et les actions



Trois méthodes de création : inline HTML, inline JS, écouteurs

---

http://www.w3schools.com/js/js_htmldom_events.asp

Un objet évènement

Lorsqu'un évènement se déclenche, un objet (spécifique au type d'évènement) est automatiquement créé et contient des informations supplémentaires (e.g., quelle touche clavier a été pressée, quel est le composant qui a déclenché l'évènement)

Quelques propriétés :

- `target`, `currentTarget` le composant déclencheur
- `type` retourne le type d'évènement (générique)
- `button`, `buttons`, `which` ou `clientX` pour l'objet "évènement souris"
- `shiftKey`, `ctrlKey`, `key`, `keycode` pour l'objet "évènement clavier"
- ...

---

https://developer.mozilla.org/fr/docs/Web/API/Event

http://www.w3schools.com/jsref/dom_obj_event.asp

Évènement - définition cachée (HTML)

Certains frameworks (e.g., Bootstrap) facilitent la création de quelques évènements (logique cachée)

```
144 <button class="btn btn-info" data-bs-toggle="collapse" href="#coll1">Cliquez !</b
        utton>
145 <div class="collapse" id="coll1">
146     Un panel replié (logique JavaScript cachée).
147 </div>
```

Évènement repli "caché", via des attributs HTML

```
149 <button class="btn btn-info" onclick="collapsePanel();">Cliquez !</button>
150 <div class="collapse" id="coll2">
151     Un second panel replié (animation codée en JavaScript).
152 </div>
153 <script>
154     function collapsePanel() {
155         if (document.getElementById("coll2").classList.contains("show"))
156             $('#coll2').collapse('hide');
157         else // panel caché
158             $('#coll2').collapse('show');
159     }
160 </script>
```

Même évènement repli mais programmé en JS

http://getbootstrap.com/docs/5.0/getting-started/javascript/

Évènement - inline HTML

Lien et actions dans le code HTML (mais codées en JavaScript) :

- Code difficilement maintenable et non réutilisable...
- Un seul évènement de type "onclick" pour ce composant

```
<balise typeEvent = "actions;">
```

```
102 <button id="bouton0" class="btn btn-info" onclick="this.classList.remove('btn-info');
        this.classList.add('btn-success'); alert('Modification réussie du bouton0');"
      >Cliquez-moi !</button>
```

Évènement - inline HTML

Lien et actions dans le code HTML (mais codées en JavaScript) :

- Code difficilement maintenable et non réutilisable...
- Un seul évènement de type "onclick" pour ce composant

```
<balise typeEvent = "actions;">
```

```
102 <button id="bouton0" class="btn btn-info" onclick="this.classList.remove('btn-info');
        this.classList.add('btn-success'); alert('Modification réussie du bouton0');"
      >Cliquez-moi !</button>
```

Évènement - inline HTML (fonction nommée)

Lien via un attribut HTML, et actions dans une fonction JS :

- Fonction nommée réutilisable
- Un seul évènement de type "onclick" pour ce composant

```
<balise typeEvent="nomFonction()">
function nomFonction() { … }
```

```
104 <button class="btn btn-info" id="bouton1" onclick="modifierBouton1();">Cliquez-moi !
        </button>
105 <script>
106     function modifierBouton1() {
107         document.getElementById("bouton1").classList.remove("btn-info");
108         document.getElementById("bouton1").classList.add("btn-success");
109         alert("Modification réussie du bouton1");
110     }
111 </script>
```

Évènement - inline JS (fonction nommée)

Lien et actions dans un script JS :

- Fonction nommée réutilisable (avec généricité des actions)
- Méthode obsolète, mais encore largement utilisée
- Un seul évènement de type "onclick" pour ce composant

```
document.querySelector(css).typeEvent = nomFonction;
function nomFonction() { … }
```

```
113 <button class="btn btn-info" id="bouton2">Cliquez-moi !</button>
114 <script>
115     var bouton = document.getElementById("bouton2");
116     bouton.onclick = modifierBouton;
117
118     function modifierBouton(event) {
119         var composant = event.target;
120         composant.classList.remove("btn-info");
121         composant.classList.add("btn-success");
122         alert("Modification réussie du " + composant.id);
123     }
124 </script>
```

Évènement - inline JS (fonction anonyme)

Lien et actions dans un script JS :

- Fonction anonyme non réutilisable
- Méthode obsolète, mais encore largement utilisée
- Un seul évènement de type "onclick" pour ce composant

```
document.querySelector(css).typeEvent = function() { … };
```

```
126 <button class="btn btn-info" id="bouton3">Cliquez-moi !</button>
127 <script>
128     var bouton = document.getElementById("bouton3");
129     bouton.onclick = function (event) {
130         var composant = event.target;
131         composant.classList.remove("btn-info");
132         composant.classList.add("btn-success");
133         alert("Modification réussie du " + composant.id);
134     };
135 </script>
```

Évènement - écouteurs d'évènements

Lien et action dans un script JS via l'interface EventTarget :

- Plusieurs évènements de même type sur le même composant
- Méthode standard W3C
- Contrôle fin sur l'activation (booléen *useCapture*)

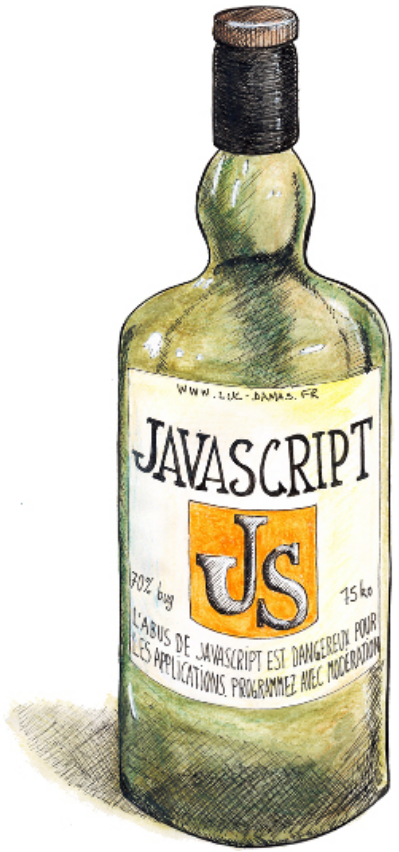
```
document.querySelector(css).addEventListener(typeEvent,
listener, useCapture);
```

```
137 <button class="btn btn-info" id="bouton4">Cliquez-moi !</button>
138 <script>
139     var bouton = document.getElementById("bouton4");
140     // réutilisation de la fonction modifierBouton() définie précedemment
141     bouton.addEventListener("click", modifierBouton, false);
142 </script>
```

https://developer.mozilla.org/fr/docs/Web/API/EventTarget/

En résumé

- Composant (source de l'évènement), type d'évènement, actions
- Un objet qui décrit l'évènement (informations supplémentaires)
- Trois méthodes pour créer un évènement :
  - privilégier celle basée sur les écouteurs ou la méthode inline HTML avec fonction nommée
  - d'autres méthodes possibles, par exemple avec des API (JQuery)



https://api.jquery.com/

http://www.luc-damas.fr/humeurs/

Plan

Événements

Types d'évènements en JS similaires à ceux des autres langages :

- Une cinquantaine, répartis en catégories (e.g., souris, formulaire, impression)
- Objets évènement selon ces catégories (e.g., objet "MouseEvent" pour les événements "Mouse")
- Attention de bien choisir l'évènement le mieux adapté (erreurs, performances)

---

http://www.w3schools.com/jsref/dom_obj_event.asp

https://w3c.github.io/uievents/

Évènements souris

Évènement	Description
onclick	L'utilisatrice clique sur un élément
ondblclick	L'utilisatrice double-clique sur un élément
onmousedown	L'utilisatrice presse un bouton de la souris
onmouseup	L'utilisatrice relâche un bouton de la souris
onmouseenter	La souris arrive sur l'élément
onmouseover	La souris arrive sur l'élément ou sur l'un de ses enfants
onmouseleave	La souris sort de l'élément
onmouseout	La souris sort de l'élément ou de l'un de ses enfants
onmousemove	La souris se déplace sur l'élément
oncontextmenu	L'utilisatrice clique avec le bouton droit (menu contextuel)

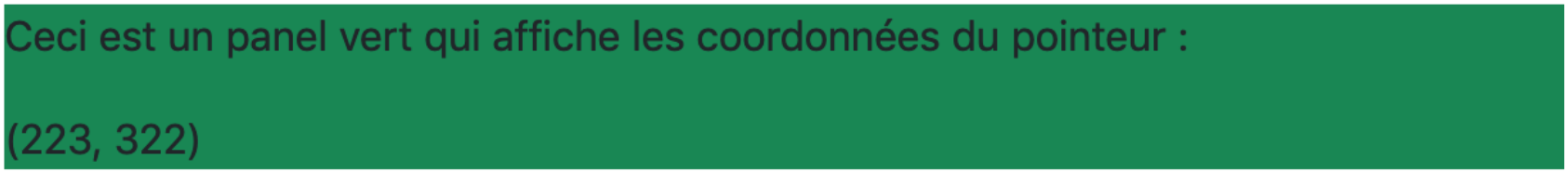
Exemple d'évènement souris

```
164 <div class="bg-success" onmousemove="afficheCoords(event);">
165     <p>Ceci est un panel vert qui affiche les coordonnées du pointeur :</p>
166     <p id="coordsParag"></p>
167 </div>
168 <script>
169     function afficheCoords(event) {
170         var x = event.clientX;
171         var y = event.clientY;
172         document.getElementById("coordsParag").innerHTML = "(" + x + ", " + y + ")";
173     }
174 </script>
```

Au chargement de la page, aucune coordonnée (second paragraphe vide et donc non affiché)

Exemple d'évènement souris

```
164 <div class="bg-success" onmousemove="afficheCoords(event);">
165     <p>Ceci est un panel vert qui affiche les coordonnées du pointeur :</p>
166     <p id="coordsParag"></p>
167 </div>
168 <script>
169     function afficheCoords(event) {
170         var x = event.clientX;
171         var y = event.clientY;
172         document.getElementById("coordsParag").innerHTML = "(" + x + ", " + y + ")";
173     }
174 </script>
```



Quand la souris survole le panel vert, les coordonnées se mettent à jour dans le second paragraphe

Évènements clavier

Évènement	Description
onkeydown	L'utilisatrice est en train d'appuyer sur une touche du clavier
onkeypress	L'utilisatrice appuie sur une touche du clavier
onkeyup	L'utilisatrice relâche une touche du clavier

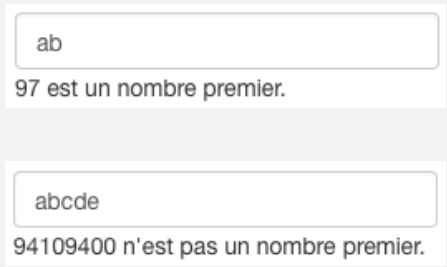
http://xkcd.com/

Exemple d'évènement clavier

```
178 <input type="text" class="form-control" id="input3" placeholder="texte libre">
179 <span id="resCalcul"></span>
180 <script>
181     var input3 = document.getElementById("input3");
182     input3.addEventListener("keydown", noCtrlkey, false);
183     input3.addEventListener("keydown", calculSaisie, false);
184 
185     function calculSaisie(event) {
186         var chaine = input3.value;
187         var resPanel = document.getElementById("resCalcul");
188         var res = 1;
189         for (var i = 0; i < chaine.length; i++)
190             res *= chaine.charCodeAt(i);
191         if (isPrime(res))
192             resPanel.innerHTML = res + " est un nombre premier.";
193         else
194             resPanel.innerHTML = res + " n'est pas un nombre premier.";
195     }
196 
197     function noCtrlkey(event) {
198         if (event.ctrlKey)
199             alert('Merci de ne pas appuyer sur la touche Ctrl');
200     }
```

Exemple d'évènement clavier

```
178 <input type="text" class="form-control" id="input3" placeholder="texte libre">
179 <span id="resCalcul"></span>
180 <script>
181     var input3 = document.getElementById("input3");
182     input3.addEventListener("keydown", noCtrlkey, false);
183     input3.addEventListener("keydown", calculSaisie, false);
184
185     function calculSaisie(event) {
186         var chaine = input3.value;
187         var resPanel = document.getElementById("resCalcul");
188         var res = 1;
189         for (var i = 0; i < chaine.length; i++)
190             res *= chaine.charCodeAt(i);
191         if (isPrime(res))
192             resPanel.innerHTML = res + " est un nombre premier.";
193         else
194             resPanel.innerHTML = res + " n'est pas un nombre premier.";
195     }
196
197     function noCtrlkey(event) {
198         if (event.ctrlKey)
199             alert('Merci de ne pas appuyer sur la touche Ctrl');
200     }
```



Exemple d'évènement clavier

```
178 <input type="text" class="form-control" id="input3" placeholder="texte libre">
179 <span id="resCalcul"></span>
180 <script>
181     var input3 = document.getElementById("input3");
182     input3.addEventListener("keydown", noCtrlkey, false);
183     input3.addEventListener("keydown", calculSaisie, false);
184
185     function calculSaisie(event) {
186         var chaine = input3.value;
187         var resPanel = document.getElementById("resCalcul");
188         var res = 1;
189         for (var i = 0; i < chaine.length; i++)
190             res *= chaine.charCodeAt(i);
191         if (isPrime(res))
192             resPanel.innerHTML = res + " est un nombre premier.";
193         else
194             resPanel.innerHTML = res + " n'est pas un nombre premier.";
195     }
196
197     function noCtrlkey(event) {
198         if (event.ctrlKey)
199             alert('Merci de ne pas appuyer sur la touche Ctrl');
200     }
```

ab

97 est un nombre premier.

abcde

94109400 n'est pas un nombre premier.

Un événement de type *"onkeypress"* est-il judicieux ici ?

Évènements objet/fenêtre

Évènement	Description
onload	Un objet a fini de charger (souvent <body>)
onpageshow	Un objet a fini de charger (déclenché aussi avec le cache)
onunload	L'utilisatrice quitte la page (pas de cache)
onpagehide	L'utilisatrice quitte la page (mise en cache)
onhashchange	L'ancre de l'URL a été modifiée
onresize	L'objet est redimensionné
onscroll	Défilement via l'ascenseur de l'objet
onabort	Annulation du chargement d'un objet
onbeforeunload	Déclenché juste avant de quitter la page (boîte de dialogue)
onerror	Erreur au chargement d'un fichier externe (e.g., une image)

Exemple d'évènement objet/fenêtre

```
216 <img id="img0" src="image-imaginaire" class="img-fluid" alt="Une image
        inexistante..." />
217 <script>
218     document.getElementById('img0').onerror = function (event) {
219         alert("Erreur lors du chargement de l'image : " + event.target.id);
220         console.log("Erreur lors du chargement de l'image : " + event.target.id);
221         event.target.getAttributeNode('src').value = "img/tigre.png";
222         console.log("Remplacement de l'image manquante");
223     };
224 </script>
```

Exemple d'évènement objet/fenêtre

```
216 <img id="img0" src="image-imaginaire" class="img-fluid" alt="Une image
        inexistante..." />
217 <script>
218     document.getElementById('img0').onerror = function (event) {
219         alert("Erreur lors du chargement de l'image : " + event.target.id);
220         console.log("Erreur lors du chargement de l'image : " + event.target.id);
221         event.target.getAttributeNode('src').value = "img/tigre.png";
222         console.log("Remplacement de l'image manquante");
223     };
224 </script>
```

Au chargement de la page, erreur sur une image (son attribut "alt" s'affiche)

Exemple d'évènement objet/fenêtre

```
216 <img id="img0" src="image-imaginaire" class="img-fluid" alt="Une image
        inexistante..." />
217 <script>
218     document.getElementById('img0').onerror = function (event) {
219         alert("Erreur lors du chargement de l'image : " + event.target.id);
220         console.log("Erreur lors du chargement de l'image : " + event.target.id);
221         event.target.getAttributeNode('src').value = "img/tigre.png";
222         console.log("Remplacement de l'image manquante");
223     };
224 </script>
```

Au chargement de la page, erreur sur une image (son attribut "alt" s'affiche)

L'évènement "onerror" de l'image se déclenche et affiche une image par défaut

Évènements formulaire

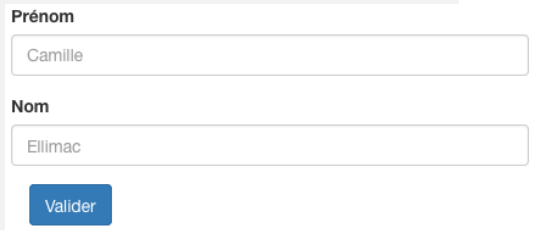
Évènement	Description
onblur	Un élément perd le focus
onchange	L'état d'un <*input*>, <*select*>, ou <*textarea*> est modifié
onfocus	Un élément gagne le focus
onfocusin	Un élément est sur le point de gagner le focus
onfocusout	Un élément est sur le point de perdre le focus
oninput	La valeur d'un <*input*> ou <*textarea*> est modifiée
oninvalid	Un composant <*input*> est invalide (e.g., vide) avant soumission
onreset	Le formulaire est réinitialisé
onsearch	Validation ou effacement d'une saisie dans un <*input="search"*>
onselect	Sélection de texte dans un <*input*> ou <*textarea*>
onsubmit	Le formulaire est soumis

Exemple d'évènement formulaire

```
228 <form class="row">
229     <div class="col-4">
230         <fieldset class="form-group">
231             <label for="inputNom">Prénom</label>
232             <input type="text" class="form-control" id="inputNom" placeholder="
                    Camille" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
233         </fieldset>
234         <fieldset class="form-group">
235             <label for="inputNom">Nom</label>
236             <input type="text" class="form-control" id="inputNom" placeholder="
                    Ellimac" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
237         </fieldset>
238         <div class="col-12">
239             <button type="submit" class="btn btn-primary">Valider</button>
240         </div>
241     </div>
242 </form>
243 <script>
244     function verifChamp(event) {
245         var source = event.target;
246         if (source.value == null || source.value == '') {
247             source.parentElement.classList.remove("has-success");
248             source.parentElement.classList.add("has-error");
249         } else { // champs non vide
250             source.parentElement.classList.remove("has-error");
251             source.parentElement.classList.add("has-success");
252         }
253     }
254 </script>
```

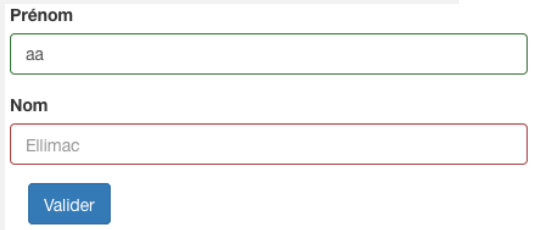
Exemple d'évènement formulaire

```
228 <form class="row">
229     <div class="col-4">
230         <fieldset class="form-group">
231             <label for="inputNom">Prénom</label>
232             <input type="text" class="form-control" id="inputNom" placeholder="
                    Camille" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
233         </fieldset>
234         <fieldset class="form-group">
235             <label for="inputNom">Nom</label>
236             <input type="text" class="form-control" id="inputNom" placeholder="
                    Ellimac" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
237         </fieldset>
238         <div class="col-12">
239             <button type="submit" class="btn btn-primary">Valider</button>
240         </div>
241     </div>
242 </form>
243 <script>
244     function verifChamp(event) {
245         var source = event.target;
246         if (source.value == null || source.value == '') {
247             source.parentElement.classList.remove("has-success");
248             source.parentElement.classList.add("has-error");
249         } else { // champs non vide
250             source.parentElement.classList.remove("has-error");
251             source.parentElement.classList.add("has-success");
252         }
253     }
254 </script>
```



Exemple d'évènement formulaire

```
228 <form class="row">
229     <div class="col-4">
230         <fieldset class="form-group">
231             <label for="inputNom">Prénom</label>
232             <input type="text" class="form-control" id="inputNom" placeholder="
                    Camille" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
233         </fieldset>
234         <fieldset class="form-group">
235             <label for="inputNom">Nom</label>
236             <input type="text" class="form-control" id="inputNom" placeholder="
                    Ellimac" required oninvalid="verifChamp(event);" oninput="verifChamp
                    (event);">
237         </fieldset>
238         <div class="col-12">
239             <button type="submit" class="btn btn-primary">Valider</button>
240         </div>
241     </div>
242 </form>
243 <script>
244     function verifChamp(event) {
245         var source = event.target;
246         if (source.value == null || source.value == '') {
247             source.parentElement.classList.remove("has-success");
248             source.parentElement.classList.add("has-error");
249         } else { // champs non vide
250             source.parentElement.classList.remove("has-error");
251             source.parentElement.classList.add("has-success");
252         }
253     }
254 </script>
```



Évènements divers

Évènement	Description
onshow	Apparition du menu contextuel
onwheel	Utilisation de la molette de la souris sur un élément
ontoggle	Ouverture / fermeture d'un élément *<details>*
...	...

```
258 <div onwheel="this.style.fontWeight = 'bold'; this.classList.remove('bg-warning');
        this.classList.add('bg-success');" class="bg-warning">
259     <p>Ceci est un panel jaune.</p>
260     <p>Il contient deux paragraphes.</p>
261 </div>
```

Ceci est un panel jaune.

Il contient deux paragraphes.

Au chargement de la page, le panel est jaune

Évènements divers

Évènement	Description
onshow	Apparition du menu contextuel
onwheel	Utilisation de la molette de la souris sur un élément
ontoggle	Ouverture / fermeture d'un élément *<details>*
...	**...**

```
258 <div onwheel="this.style.fontWeight = 'bold'; this.classList.remove('bg-warning');
        this.classList.add('bg-success');" class="bg-warning">
259     <p>Ceci est un panel jaune.</p>
260     <p>Il contient deux paragraphes.</p>
261 </div>
```

Ceci est un panel jaune.

Il contient deux paragraphes.

Après avoir utilisé la molette de la souris au-dessus du panel, ce dernier devient vert et son texte passe en gras

Autres types d'événements

- Media (e.g., `onpause`, `ondurationchanged`)
- Animation CSS (e.g., `animationstart`)
- Transition CSS (`transitionend`)
- Serveur (e.g., `onmessage`, `onerror`)
- Écrans tactiles (e.g., `ontouchmove`)
- Impression (e.g., `onbeforeprint` et `onafterprint`)
- Presse-papier (`oncopy`, `oncut` et `onpaste`)
- Glisser-déposer (e.g., `ondragstart`, `ondrop`)

Bilan

- Création d'évènements pour le web : un composant (source de l'évènement), un type d'évènement et des actions associées
  - un large choix de types d'évènements
  - des actions écrites en exploitant le DOM/BOM
  - trois méthodes pour lier un évènement à un composant (inline HTML, inline JS et écouteurs)
- Utiliser les outils de développement web du navigateur (inspecteur, console, etc.)
- Ne pas abuser du JavaScript (non référencé par les moteurs de recherche)
- Optimiser le code (e.g., outils de développement web, Pingdom, SonarQube)

---

https://tools.pingdom.com/
https://www.sonarqube.org/