

# Javascript

Dans un navigateur web

# Inclusion du code

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>maPage</title>
    <link rel="stylesheet" href="mesStyles.css"/>
    <script type="module" src="monScript.js"></script>
  </head>
  <body>
    <!-- Contenu de ma page -->
  </body>
</html>
```

`type module` : permet de scinder le script en plusieurs fichiers à l'aide d'imports/exports. Le script est exécuté après analyse de la page.

# Document Object Model

Standard du [W3C](#) qui décrit une API indépendante du langage et de la plate-forme, permettant à des programmes d'accéder et de mettre à jour le contenu de documents HTML et XML.

# DOM Window

Rappel : `window` est l'objet global dans l'environnement du navigateur. Définir une variable globale revient à définir une propriété de l'objet global.

```
typeof window; //"object"

window.toto = "tata";

toto; //"tata"

function add(a,b) { return a+b; }

add(2,1); //3
window.add(2,1); //3
```

# DOM Window

## Propriétés usuelles

- `document` : référence du document contenu dans la fenêtre
- `innerHeight` : hauteur du contenu visible de la fenêtre
- `innerWidth` : largeur du contenu visible de la fenêtre
- `location` : lit/définit l'URL courante de la fenêtre
- `outerHeight` : hauteur de l'extérieur de la fenêtre
- `outerWidth` : largeur de l'extérieur de la fenêtre
- `scrollX` : défilement horizontal du document
- `scrollY` : défilement vertical du document

⇒ Référence complète

# DOM Window

## Méthodes usuelles

- `alert` : affiche une boîte de message d'alerte
- `back` : recule d'une page dans l'historique de la fenêtre
- `confirm` : affiche une boîte de demande de confirmation
- `prompt` : renvoie le texte saisi dans une boîte d'invite
- `scroll` : fait défiler la fenêtre à un endroit particulier
- `setInterval` : exécute une fonction à intervalles réguliers
- `setTimeout` : définit un délai avant d'exécuter une fonction

⇒ [Référence complète](#)

# DOM Document

Toute page web chargée dans un navigateur web a son propre objet document. Cet objet sert de point d'entrée au contenu de la page et apporte des fonctions générales au document.

```
<!doctype html>
<html>
  <head>
    <title>maPage</title>
  </head>
  <body>
    <div id="maDiv" class="maClasse">
      Allez donc voir le <a href="http://mpfc/">site de MPFC</a>
    </div>
  </body>
</html>
```

```
//accès à des nœuds existants
document.getElementById("maDiv"); //nœud par id
document.querySelector("div.maClasse"); //nœud par selecteur css

//création d'un nœud
let div = document.createElement("div");

//insertion de l'objet DOM dans le corps de la page
document.body.appendChild(div);
```

# DOM Document

## Propriétés usuelles

- **body** : renvoie le nœud BODY du document
- **cookie** : renvoie la liste des cookies ou définit un cookie
- **head** : renvoie le nœud HEAD du document
- **referrer** : renvoie l'URI de la page qui a amené à cette page
- **title** : renvoie ou définit le titre du document

⇒ [Référence complète](#)



# DOM Document

## Méthodes usuelles

- `createElement` : crée un nouvel élément du type spécifié
- `createTextNode` : crée un nœud de texte
- `getElementById` : renvoie l'élément d'id donné
- `querySelector` : renvoie le 1er élément correspondant au selecteur css
- `querySelectorAll` : renvoie la liste des éléments correspondant au selecteur css

⇒ [Référence complète](#)

# DOM Element

Tous les nœuds html de la page sont représentés par des objets DOM qui partagent des propriétés communes.

```
<!doctype html>
<html>
  <head>
    <title>maPage</title>
  </head>
  <body>
    <div id="maDiv" class="maClasse">
      Allez donc voir le <a href="http://mpfc/">site de MPFC</a>
    </div>
  </body>
</html>
```

```
let div = document.getElementById("maDiv");
div.parentNode === document.body; //true
div.classList.remove("maClasse");
```

# DOM Element

## Propriétés usuelles (1/2)

- `attributes` : tous les attributs associés
- `className` : définit/obtient la classe
- `classList` : gestion avancée des classes
- `id` : définit/obtient l'identifiant
- `style` : objet représentant l'attribut style
- `tagName` : nom de la balise
- `textContent` : définit/obtient le contenu textuel
- `innerHTML` : définit/obtient l'ensemble du balisage contenu
- `outerHTML` : définit/obtient l'ensemble du balisage de l'élément et son contenu

# DOM Element

## Propriétés usuelles (2/2)

- `firstElementChild` : premier élément enfant direct
- `lastElementChild` : dernier élément enfant direct
- `parentNode` : nœud parent
- `children` : tous les éléments enfants
- `previousElementSibling` : élément précédant immédiatement
- `nextElementSibling` : élément suivant immédiatement

⇒ Référence complète

# DOM Element

## Méthodes usuelles (1/2)

- `remove` : retire l'élément courant du DOM
- `appendChild` : insère un nœud comme dernier enfant
- `removeChild` : retire un nœud enfant
- `replaceChild` : remplace un nœud enfant par un autre
- `insertBefore` : insère un nœud avant le nœud enfant spécifié
- `insertAdjacentHTML` : parse en tant que HTML et insert les nœuds résultants à la position spécifiée
- `cloneNode` : clone le nœud

# DOM Element

## Méthodes usuelles (2/2)

- `addEventListener` : ajoute un écouteur d'événements
- `removeEventListener` : retire un écouteur d'événements
- `querySelector` : renvoie le 1er élément parmi les descendants correspondant au selecteur css
- `querySelectorAll` : renvoie la liste des éléments parmi les descendants correspondant au selecteur css
- `matches` : teste si l'élément correspond au selecteur css
- `getBoundingClientRect` : renvoie les coordonnées
- `getAttribute` : renvoie la valeur de l'attribut spécifié
- `setAttribute` : définit la valeur de l'attribut spécifié
- `removeAttribute` : retire l'attribut spécifié

⇒ Référence complète

# Gestionnaires d'événements

Très vieille école  $\Rightarrow$  à bannir

```
<a href="#" onclick="alert('toto'); return false;">Cliquez ici</a>
```

Vieille école  $\Rightarrow$  à éviter

```
<a href="#" id="monLien">Cliquez ici</a>
```

```
let lien = document.getElementById("monLien");  
lien.onclick = function() { alert('toto'); return false; };
```

Méthode actuelle

```
<a href="#" id="monLien">Cliquez ici</a>
```

```
let lien = document.getElementById("monLien");  
lien.addEventListener("click", function(e) {  
  e.preventDefault();  
  alert('toto');  
});
```

$\Rightarrow$  permet de gérer plusieurs écouteurs sur un même type d'événement.

# L'objet Event

C'est la nature de l'objet passé en argument de la fonction enregistrée.

```
let lien = document.getElementById("monLien");

lien.addEventListener("click", function(e) {

  e instanceof Event; //true
  e.type; //type d'événement sous forme de chaîne ("click" dans ce cas)

});
```

⇒ [Référence complète](#)



# L'objet Event

## Propriété `target`

Référence à l'objet qui a envoyé l'événement.

```
<div id="maDiv">  
  Ceci est un <span>exemple</span>.  
</div>
```

```
function afficheCible(e) {  
  console.log( e.target.tagName ); //"div" ou "span"  
}  
  
let div = document.getElementById("maDiv");  
  
div.addEventListener("click",afficheCible);  
  
//les
```

# L'objet Event

## Méthode `preventDefault`

Annule le comportement par défaut.

Cherchez sur `<a id="monLien" href="http://www.google.com">google</a>`.

```
function changeLien(e) {  
  e.preventDefault();  
  window.location = "http://duckduckgo.com";  
}  
  
let lien = document.getElementById("monLien");  
lien.addEventListener("click", changeLien);
```

# L'objet MouseEvent

Dérivation de l'objet Event pour les événements de pointage (*click*, *dblclick*, *contextmenu*, *mousedown*, *mouseup*, *mousemove*, *mouseover*, *mouseout*, etc)

```
let lien = document.getElementById("monLien");

lien.addEventListener("click", function(e) {

    e instanceof Event; //true
    e instanceof MouseEvent; //true

    e.clientX; //abscisse du pointeur dans la fenêtre (et non la page)
    e.clientY; //ordonnée du pointeur dans la fenêtre (et non la page)

});
```

⇒ [Référence complète](#)

# L'objet KeyboardEvent

Dérivation de l'objet Event pour les événements liés au clavier  
(*keydown*, *keypress*, *keyup*)

```
window.addEventListener("keydown", function(e) {  
  
    e instanceof Event; //true  
    e instanceof KeyboardEvent; //true  
  
    e.key; //nom de la touche  
  
});
```

⇒ Référence complète

⇒ Nom des touches

# Compatibilité des navigateurs

Longtemps problématique, aujourd'hui  
anecdotique

Depuis la fin d'Internet Explorer, la généralisation des "**evergreen browsers**" (Firefox, Chrome, Safari, Edge, etc), et l'utilisation courante de **frameworks** (Angular, React), il est devenu rare d'avoir à s'en soucier.

# Gérer la compatibilité

Evolution de la répartition des navigateurs sur les 28 dernières années

# Gérer la compatibilité

## La mauvaise méthode : browser sniffing

```
var browser = navigator.userAgent.matches(/MSIE/) ? "IE" : "autre";

function addListener(elmt,type,fct) {

    if (browser == "IE") elmt.attachEvent('on'+type,fct);
    else elmt.addEventListener(type,fct,false);
}

var div = document.getElementById("maDiv");
addListener(div,"click",function() { alert("hello world"); });
```

# Gérer la compatibilité

## La mauvaise méthode : browser sniffing

- aucune garantie sur la chaîne userAgent

Ex : IE8    Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1;

- impossible de prendre en compte toutes les versions de tous les navigateurs
- le code peut planter avec les versions futures des navigateurs  
Ex : IE11 ne supporte plus attachEvent
- On peut toujours faire autrement



# Gérer la compatibilité

## La bonne méthode

On teste l'existence des méthodes, de la plus standard à la moins répandue.

```
function addListener(elm, type, fct) {  
    if (elm.addEventListener) elm.addEventListener(type, fct, false);  
    else if (elm.attachEvent) elm.attachEvent('on'+type, fct);  
}  
  
var div = document.getElementById("maDiv");  
addListener(div, "click", function() { alert("hello world"); });
```

# Introduction à jQuery

Aujourd'hui supplantée par des frameworks comme Angular ou React, **jQuery** est une bibliothèque javascript qui a dominé le monde du web pendant de nombreuses années et reste très présente.

# Quelques principes de jQuery

- Pas de pollution de l'espace global (Une seule variable globale jQuery et son alias \$)
- Compatibilité entre navigateurs
- Concision

```
let list = document.querySelectorAll(".classe");  
for (let item of list) { list.remove(); }  
// versus  
$('.classe').detach();
```

- Les méthodes renvoient toujours la collection jQuery, ce qui permet de chaîner les opérations

```
$('a').css("color", "red").show();
```