



# JavaScript

RÉALISÉ PAR : M, AMINE OTMANI



## Introduction :

- Le **JavaScript** est un langage de programmation créé en 1995.
- Le JavaScript va nous permettre de créer des pages interactives et « vivantes » à l'aide de scripts.
- Un script une suite d'instructions qui vont être interprétées par un programme.
- Ainsi, pour lire du code JavaScript, il va nous falloir un interpréteur. Heureusement, tous les navigateurs (Google Chrome, Safari, etc.) possèdent leur propre interpréteur JavaScript.

## Langage client-side

- Le JavaScript est un langage de programmation dit « client-side ».
- Il existe deux grands types de langages de programmation : les langages « client-side » et les langages « server-side ».
- Les langages client-side vont s'exécuter du côté du client, c'est-à-dire sur l'ordinateur de la personne qui va demander la page web.
- Les langages server-side, au contraire, vont s'exécuter côté serveur.

## un langage orienté objet

- Finalement, le JavaScript est également un langage orienté objet à prototype.
- Cela signifie simplement que le JavaScript va utiliser des objets dans son fonctionnement, et que ces objets sont prototypés.

## Où écrire notre code JavaScript ?

- On va pouvoir **placer du code JavaScript** à trois endroits différents :
  - Dans l'élément head d'une page HTML ;
  - Dans l'élément body d'une page HTML ;
  - Dans un fichier portant l'extension « .js » séparé
- Bien que cette dernière méthode soit généralement recommandée, notamment pour des gros projets, parfois vous serez « obligé » d'écrire votre code JavaScript dans votre fichier HTML.

## Syntaxe, Commentaire du JavaScript

- Au niveau de la syntaxe générale du JavaScript, retenez bien que chaque **instruction en JavaScript** doit être terminée par un point virgule.
- En JavaScript, il existe deux types de commentaires qui vont s'écrire différemment : les commentaires mono-ligne et les commentaires multi-lignes.

## Les variables en javascript

- On déclare une variable avec le mot clef « var » suivi du nom de notre variable. Chaque nouvelle variable doit avoir un nom unique.
  - Le nom des variables doit commencer par une lettre ;
  - Le nom ne peut contenir que des lettres (non accentuées), des chiffres ou les signes « \_ » et « \$ » ;
  - Le nom des variables est sensible à la casse (« a » est différent de « A ») ;
  - Le JavaScript possède des mots « réservés » qu'on ne peut utiliser pour créer une variable.

## Déclarer une variable en JavaScript

- Il existe différentes façons de **déclarer une variable en JavaScript**. On va pouvoir :
  - Déclarer une variable et lui assigner une valeur immédiatement ;
  - Déclarer une variable sans lui assigner de valeur et lui assigner une valeur plus tard ;
  - Déclarer plusieurs variables en même temps.



## Types de variables en JavaScript:

- Les variables en JavaScript peuvent stocker bien d'autres valeurs n'étant pas de type String, Number ou Boolean.
- Parmi les autres valeurs remarquables, on peut citer la valeur « null », qui correspond à la non connaissance à priori de la valeur ainsi que la valeur « undefined » qui correspond au fait de ne pas avoir défini de valeur pour notre variable.

## Tester le type de valeur d'une variable

- Pour tester le type de la valeur que contient une variable, on utilise généralement la fonction ***typeof()***.
- Cette fonction renvoie parfois des résultats contestables sur certaines valeurs.

## Array et des tableaux

- Un tableau est un type de variable spécial qui peut contenir plusieurs valeurs.
- Les tableaux vont être gérés par l'objet **Array**, un objet natif en JavaScript qui possède ses propres propriétés et méthodes.
- Lorsque l'on crée un tableau, un indice (ou une clef) numérique unique est automatiquement affectée à chaque valeur de notre tableau. la première valeur de notre tableau va être associée à l'index 0

## Création et syntaxe d'un tableau

- on crée un nouveau tableau en utilisant :
  - La syntaxe : ***new array()***.  
***var MonTableau = new Array("donnée 1", "donnée 2");***
  - Ou on créant une variable de type tableau :  
***Var montableau=[elt1, elt2, ... eltN].***
- L'accès aux éléments du tableau se fait en écrivant le nom du tableau suivi de crochets contenant l'indice de l'élément.

# Tableaux associatifs

- Il est possible d'utiliser des indices personnalisés pour indexer des valeurs, on parle alors de tableau associatif. Un tableau associatif n'est autre qu'un objet javascript ou json.

## Affectation de valeurs

```
var tableau_associa={  
  "valeur1":10,  
  "valeur2":55,  
  "valeur3":30  
};
```

## Accès aux données

```
var tableau_associa={  
  "valeur1":10,  
  "valeur2":55,  
  "valeur3":30  
};  
  
alert(tableau_associa.valeur2);  
  
// Affichera "55"
```

# Opérations sur les variables

## Les opérateurs de calcul

Signe	Nom	Signification
+	plus	addition
-	moins	soustraction
*	multiplié par	multiplication
/	divisé par	division
%	modulo	reste de la division par
=	a la valeur	affectation

## Les opérateurs de comparaison

Signe	Nom
==	égal
<	inférieur
<=	inférieur ou égal
>	supérieur
>=	supérieur ou égal
!=	différent

## Les opérateurs associatifs

Signe	Description
+=	plus égal
-=	moins égal
*=	multiplié égal
/=	divisé égal

## Les opérateurs logiques

Signe	Nom
&&	et
	ou

## Les opérateurs d'incrémentation

++ & --

# Boucle & Conditions

If

```
if (condition vraie) {  
  une ou plusieurs instructions;  
}
```

If ... else

```
if (condition vraie) {  
  instructions1;  
}  
else {  
  instructions2;  
}
```

Concision

```
(expression) ? instruction a : instruction b
```

L'expression for

```
for (valeur initiale ; condition ; progression) {  
  instructions;  
}
```

While

```
while (condition vraie){  
  continuer à faire quelque chose  
}
```

Break

```
compt=1;  
while (compt<5) {  
  if (compt == 4)  
    break;  
  document.write ("ligne : " + compt + "<BR>");  
  compt++;  
}  
document.write("fin de la boucle");
```

Continue

```
compt=1;  
while (compt<5) {  
  if (compt == 3){  
    compt++  
    continue;  
  }  
  document.write ("ligne : " + compt + "<BR>");  
  compt++;  
}  
document.write("fin de la boucle");
```

## Les fonctions :

- Une fonction est un groupe de ligne(s) de code de programmation destiné à exécuter une tâche bien spécifique et que l'on pourra, si besoin est, utiliser à plusieurs reprises.

### Déclaration des fonctions

```
function nom_de_la_fonction(arguments) {  
  ... code des instructions ...  
}
```

### Retourner une valeur : return

```
function cube(nombre) {  
  var cube = nombre*nombre*nombre  
  return cube;  
}
```

- Les fonctions dans <HEAD>...<HEAD> Il est donc prudent ou judicieux de placer toutes les déclarations de fonction dans l'en-tête de votre page c-à-d dans la balise <HEAD> ... <HEAD>.



## Les événements :

- **Généralités** Avec les événements et surtout leur gestion, nous abordons le côté "**magique**" de Javascript.
- Les événements Javascript, associés aux fonctions, aux méthodes et aux formulaires, ouvrent grand la porte pour une réelle **interactivité** de vos pages.

# Les événements

Description	Événement
Lorsque l'utilisateur clique sur un bouton, un lien ou tout autre élément.	<b>Clik</b>
Lorsque la page est chargée par le browser ou le navigateur.	<b>Load</b>
Lorsque l'utilisateur quitte la page.	<b>Unload</b>
Lorsque l'utilisateur place le pointeur de la souris sur un lien ou tout autre élément.	<b>MouseOver</b>
Lorsque le pointeur de la souris quitte un lien ou tout autre élément.	<b>MouseOut</b>
Lorsque un élément de formulaire a le focus c-à-d devient la zone d'entrée active.	<b>Focus</b>
Lorsque un élément de formulaire perd le focus c-à-d que l'utilisateur clique hors du champs et que la zone d'entrée n'est plus active.	<b>Blur</b>
Lorsque la valeur d'un champ de formulaire est modifiée.	<b>Change</b>
Lorsque l'utilisateur sélectionne un champ dans un élément de formulaire.	<b>Select</b>
Lorsque l'utilisateur clique sur le bouton <i>Submit</i> pour envoyer un formulaire	<b>Submit</b>

## Les gestionnaires d'événements

- il faut qu'à ces événements soient associées les actions prévues par vous. C'est le rôle des gestionnaires d'événements.

```
onévénement="fonction()"
```

**Exemple :**

```
onClick="alert('Vous avez cliqué sur cet élément')"
```

**Exemple de référence des événements javascript (Mozilla).**

## DOM : Document Object Model

- Le Modèle Objet de Document, ou DOM, Document Object Model est un outil permettant l'accès aux documents HTML et XML. Il permet deux choses au développeur :
  - Il fournit une représentation structurée du document ;
  - Il codifie la manière dont un script peut accéder à cette structure.
- Il s'agit donc essentiellement d'un moyen de lier une page Web, par exemple, à un langage de programmation ou de script.

## DOM et JavaScript

- Le DOM n'est pas un langage de programmation, mais sans lui, le JavaScript ne pourrait avoir aucun modèle pour appréhender la page Web.

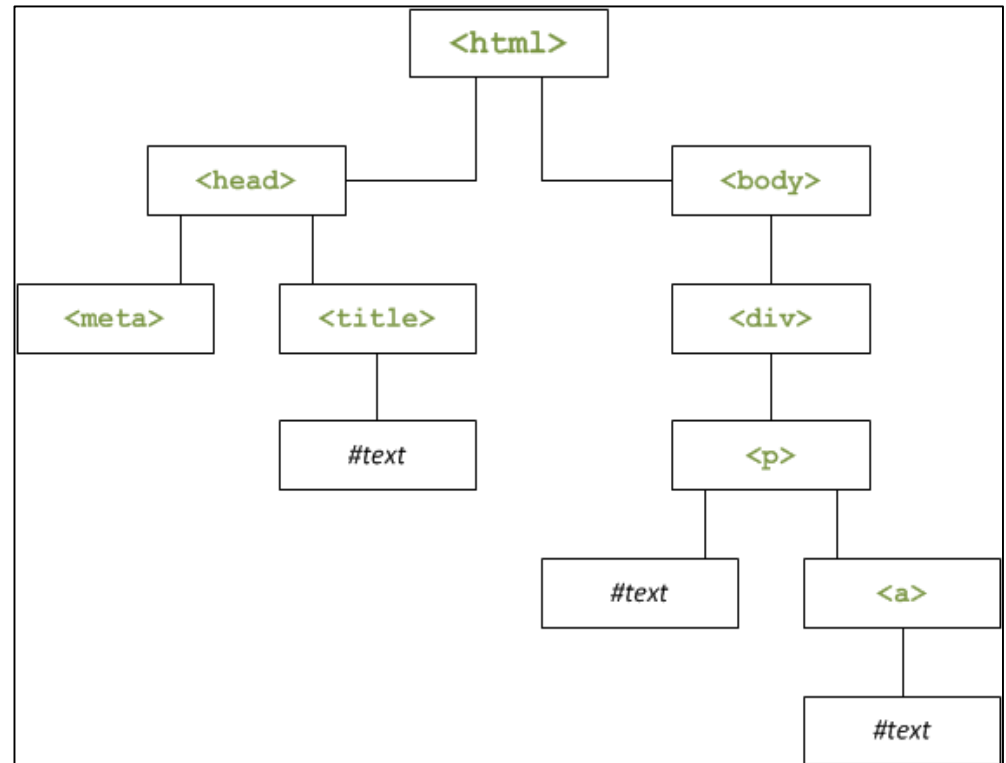
## Le document

- L'objet document est un sous-objet de window, l'un des plus utilisés. Et pour cause, il représente la *page Web* et plus précisément la balise `<html>`. C'est grâce à cet élément-là que nous allons pouvoir accéder aux éléments HTML et les modifier. Voyons donc, dans la sous-partie suivante, comment naviguer dans le document.

# Naviguer dans le document

## La structure DOM

Le DOM pose comme concept que la page Web est vue comme un arbre, comme une hiérarchie d'éléments. On peut donc schématiser une page Web simple comme ceci :



## Accéder aux éléments

- L'accès aux éléments HTML via le DOM est assez simple.
- L'objet document possède les méthodes principales :
  - getElementById(),
  - getElementsByTagName()
  - getElementsByName().
  - getElementsByClassName()
  - La méthode querySelector()
  - La méthode querySelectorAll().



## L'utilisation du mot clef *this* dans la gestion d'évènements

- En JavaScript, le mot clef *this* sert de référence à différents objets selon le contexte.
- Dans le contexte de la gestion d'évènements, *this* va faire référence à l'objet (représentant l'élément HTML) qui est le sujet de l'événement.

# Les évènements et le DOM

- La gestion des évènements va devenir véritablement intéressante lorsqu'on va réagir aux évènements via le DOM HTML.
- Le DOM HTML nous permet d'assigner des gestionnaires d'évènements spécifiques à des éléments HTML en utilisant le JavaScript.
  - Plus utiliser des attributs HTML mais du code JavaScript à proprement parler pour construire nos évènements.
- Nous avons deux manières de réagir aux évènements avec le JavaScript :
  - Soit utiliser des propriétés qui vont assigner un gestionnaire d'événement à un élément spécifique en HTML,
  - Soit utiliser la méthode ***addEventListener()***.

- La première méthode, à savoir l'utilisation de propriétés est une ancienne méthode (éviter de l'utiliser car elle possède des limitations).
  - on ne peut pas assigner plusieurs fois le même gestionnaire d'événement à un élément HTML.
- Notez que les propriétés possèdent souvent des noms analogues aux attributs HTML
  - L'attribut HTML onclick devient la propriété JavaScript onclick par exemple.

# La méthode JavaScript addEventListener()

- La méthode *addEventListener()* nous permet de lier du code à un évènement. On parlera alors de gestionnaire d'évènements.
- Le code sera alors exécuté dès le déclenchement de l'évènement.
- Cette méthode appartient à l'objet *Element* et va avoir besoin de deux arguments pour fonctionner : le nom de l'évènement déclencheur de l'action et le code relatif à l'action à effectuer.
- Les événements ont des noms similaires aux attributs HTML mais ne seront plus précédés du « on » (par exemple, onclick devient click).

# jQuery

- jQuery est une librairie JavaScript. Le rôle d'une librairie, en informatique, est de simplifier l'utilisation d'un certain langage de programmation en fournissant un ensemble de codes déjà prêts à l'emploi.
- La librairie jQuery consiste en un ensemble de scripts JavaScripts déjà pré-écrits et enfermés dans des variables ou dans des méthodes.
  - On n'a qu'à appeler la bonne méthode pour exécuter tout le code qu'elle contient, d'où un gain de temps substantiel.
- Les développeurs qui ont créé jQuery se sont également assuré de la validité parfaite de leur code pour l'ensemble des navigateurs les plus utilisés.

# UTILISER JQUERY

## 1- Télécharger jQuery

- Pour pouvoir utiliser la bibliothèque jQuery, on devrait d'abord la télécharger sur le site [jQuery.com](http://jquery.com).
- Deux fichiers jQuery :
  - Un fichier compressé .
  - Un fichier non compressé.
- La version compressée est une version ne contenant aucun espace entre les codes. Le but est de réduire le poids total de la librairie afin d'améliorer les performances de votre site.

## 2- Depuis un CDN de type Google

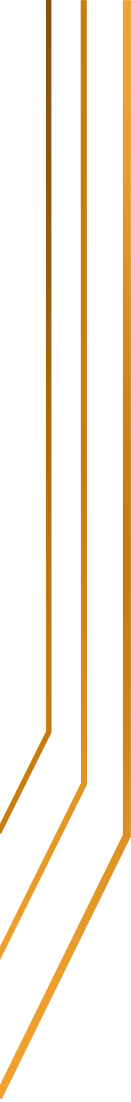
- Google propose un CDN (Content Delivery Network) dédié au chargement de différentes librairies.
- L'avantage étant de bénéficier d'un hébergement externe et rapide (mais soumis la plupart du temps à une requête HTTP+DNS supplémentaire) avec un fichier délivré en gzip sans avoir à configurer votre serveur.

```
<script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
```

# Syntaxe de base de jQuery

- Le jQuery possède une syntaxe de base assez simple, logique et intuitive.
- Le jQuery base tout son fonctionnement sur la fonction ***jQuery()***. Cette fonction est généralement écrite de manière abrégée ***\$()***.
- Cette fonction jQuery va pouvoir accepter différents arguments et va toujours retourner un objet jQuery.
- Les arguments peuvent être des sélecteurs CSS ou des fonctions.



- 
- L'une des grandes forces du jQuery est que l'on va pouvoir avoir accès aux éléments HTML en utilisant des sélecteurs CSS. Cela simplifie grandement l'accès au contenu HTML.
  - La syntaxe de base du jQuery pour accéder aux éléments HTML va donc être de la forme suivante : ***\$(selecteur)***.
  - Par exemple, pour accéder à tous les éléments p de notre page HTML, nous écrirons en jQuery : ***\$('p')***.

# Sélecteurs CSS, objets et méthodes jQuery

- On peut utiliser n'importe quel sélecteur CSS pour accéder à nos éléments HTML en jQuery.
  - Par exemple, on va pouvoir sélectionner tous les éléments portant la class, en écrivant `$('.class')` en jQuery.
- Une fois un élément sélectionné, le jQuery va retourner un objet jQuery faisant référence à l'élément en question.
- Si plusieurs éléments ont été sélectionnés d'un coup.
- Comme la fonction jQuery renvoi toujours un objet jQuery, nous allons pouvoir pratiquer ce qu'on appelle le « chaînage » de méthodes. Comme suit :

***`$(selecteur).methode1().methode2().methode3().`***

# Les sélecteurs

Sélecteur CSS	Explication
<code>\$('*');</code>	Cible tous les éléments (inclus <code>&lt;html&gt;</code> , <code>&lt;head&gt;</code> et <code>&lt;body&gt;</code> ).
<code>\$('element');</code>	Cible tous les éléments étant de type correspondant.
<code>\$('#element');</code>	Cible l'élément ayant l'identifiant donné (obligatoirement unique).
<code>\$('.element');</code>	Cible tous les éléments ayant la classe donnée.
<code>\$('parent enfant');</code>	Cibles tous les enfants directs ou indirects de l'élément parent.
<code>\$('element, element2');</code>	Cible d'abord les premiers élément trouvés, puis les seconds, etc.
<code>\$('parent &gt; enfant');</code>	Cible tous les enfants directs de l'élément parent.
<code>\$('frere + element');</code>	Cible tous les éléments précédés directement d'un frère.
<code>\$('frere ~ element');</code>	Cible tous les éléments précédés directement ou indirectement d'un frère.

Sélecteur par attribut	Explication
<code>\$('element[attribut]');</code>	Cible tous les éléments possédant l'attribut donné.
<code>\$('element[attribut="valeur"]');</code>	Cible tous les éléments possédant l'attribut donné et dont la valeur est égale à celle spécifiée.
<code>\$('element[attribut!="valeur"]');</code>	Cible tous les éléments possédant l'attribut donné et dont la valeur est différente de celle spécifiée.
<code>\$('element[attribut*="valeur"]');</code>	Cible tous les éléments possédant l'attribut donné et dont la valeur contient, entre autres, la chaîne spécifiée.
<code>\$('element[attribut1][attribut2][attributN]');</code>	Cible tous les éléments dont les attributs correspondent à ceux donnés.

Filtres	Explication
<code>\$(':animated');</code>	Cible tous les éléments actuellement animés.
<code>\$(':eq(N)');</code>	Cible tous les éléments dont l'index est égal à N.
<code>\$(':even');</code>	Cible tous les éléments dont l'index est pair.
<code>\$(':odd');</code>	Cible tous les éléments dont l'index est impair.
<code>\$(':first');</code>	Cible le premier élément trouvé.
<code>\$(':last');</code>	Cible le dernier élément trouvé.
<code>\$(':focus');</code>	Cible tous les éléments ayant actuellement le focus.
<code>\$(':header');</code>	Cible tous les éléments de type <i>header</i> ( <code>&lt;h1&gt;</code> , <code>&lt;h2&gt;</code> , <code>&lt;h3&gt;</code> , etc).
<code>\$(':not(selecteur)');</code>	Cible tous les éléments ne correspondant pas au sélecteur donné.

Sélecteurs pour formulaire	Explication
<code>\$(':input');</code>	Cible tous les éléments de formulaire.
<code>\$(':button');</code>	Cible tous les éléments de formulaire de type <code>button</code> .
<code>\$(':checkbox');</code>	Cible tous les éléments de formulaire de type <code>checkbox</code> .
<code>\$(':checked');</code>	Cible toutes les boîtes cochées.
<code>\$(':file');</code>	Cible tous les éléments de formulaire de type <code>file</code> .
<code>\$(':password');</code>	Cible tous les éléments de formulaire de type <code>password</code> .
<code>\$(':radio');</code>	Cible tous les éléments de formulaire de type <code>radio</code> .
<code>\$(':submit');</code>	Cible tous les éléments de formulaire de type <code>submit</code> .
<code>\$(':text');</code>	Cible tous les éléments de formulaire de type <code>text</code> .

# Attendre le chargement de la page pour exécuter le jQuery

- Généralement, lorsque l'on code en JavaScript, on attend la fin du chargement de la page pour exécuter nos scripts.
- Pour faire cela, on utilise généralement un gestionnaire d'évènement **load** que l'on applique à notre objet **Window** en JavaScript.
- Le jQuery dispose de sa propre solution et met à notre disposition un évènement appelé **ready** que nous allons utiliser avec l'objet Document.

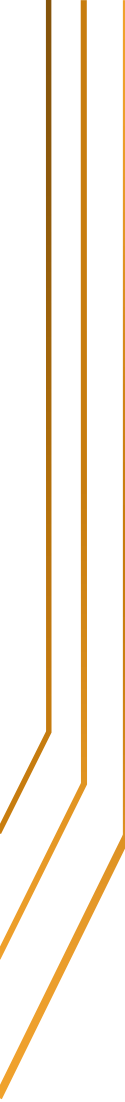
```
$(document).ready(function(){
```

Ou d'une façon implicite par : `$(function()){`

# LA GESTION DES EVENEMENTS EN JQUERY

- Un gestionnaire d'évènements est constitué de deux choses : du nom de l'évènement qu'il doit gérer et généralement d'une fonction à exécuter lors du déclenchement de l'évènement.
- Certains évènements se propagent.
  - ils partent tous de la racine du DOM, c'est-à-dire de l'élément HTML pour descendre dans le DOM en traversant les différents éléments.
  - Une fois que l'évènement a touché l'élément le plus profond du DOM, il va faire le chemin inverse et remonter jusqu'à la racine et etc.
- Ces phases de descente et de remontée des évènements s'appellent les phases de capture et de bouillonnement.
- En JavaScript, grâce à la méthode ***addEventListener()***, on peut choisir « d'écouter » ou de réagir à un évènement durant une phase ou une autre.



- 
- Le jQuery simplifie et rend encore plus puissante la gestion d'évènements en mettant à notre disposition différentes méthodes prêtes à l'emploi.
  - Ces méthodes vont tout simplement généralement porter le nom des évènements auxquels on souhaite réagir, comme par exemple la méthode `click()` qui va nous permettre de gérer l'évènement `click`.
  - Cependant, l'intérêt d'utiliser `click()` est de pouvoir exécuter un code lors d'un clic.
  - Pour cela, nous allons passer une fonction anonyme en argument à notre méthode `click()`, tout comme on le faisait en JavaScript avec `addEventListener()`.
  - Cette fonction anonyme contiendra le code à exécuter lors du déclenchement de l'évènement.

Gestionnaires d'évènements	Explication
<code>on()</code>	Initialise un gestionnaire d'évènements.
<code>off()</code>	Supprimer un gestionnaire d'évènements.
<code>trigger()</code>	Simule un évènement.

Évènement du clavier	Explication
<code>keydown()</code>	Se déclenche à l'enfoncement d'une touche.
<code>keyup()</code>	Se déclenche au relâchement d'une touche.
<code>keypress()</code>	Se déclenche à l'enfoncement et au maintien d'une touche.

Évènement de la souris	Explication
<code>click()</code>	Se déclenche au clic.
<code>dblclick()</code>	Se déclenche au double-clic.
<code>hover()</code>	Se déclenche au passage sur l'élément ciblé.
<code>mousedown()</code>	Se déclenche à l'enfoncement du bouton de la souris.
<code>mouseup()</code>	Se déclenche au relâchement du bouton de la souris.
<code>mouseenter()</code>	Se déclenche à l'entrée du curseur sur l'élément ciblé.
<code>mouseleave()</code>	Se déclenche à la sortie du curseur de l'élément ciblé.
<code>mousemove()</code>	Se déclenche au mouvement de la souris.
<code>toggle()</code>	Se déclenche à chaque clic, et alterne deux fonctions.

# Manipulation du CSS

Méthode	Explication
<code>css()</code>	Récupère ou modifie une ou plusieurs propriété(s) CSS.
<code>height()</code>	Récupère la hauteur de l'élément ciblé.
<code>width()</code>	Récupère la largeur de l'élément ciblé.
<code>innerHeight()</code>	Récupère la hauteur de l'élément ciblé en prenant en compte ses marges intérieures mais pas les bordures.
<code>innerWidth()</code>	Récupère la largeur de l'élément ciblé en prenant en compte ses marges intérieures mais pas les bordures.
<code>outerHeight()</code>	Récupère la hauteur de l'élément ciblé en prenant en compte ses marges intérieures, extérieures, et ses bordures.
<code>outerWidth()</code>	Récupère la largeur de l'élément ciblé en prenant en compte ses marges intérieures, extérieures, et ses bordures.
<code>offset()</code>	Récupère les coordonnées absolues de l'élément ciblé.
<code>position()</code>	Récupère les coordonnées relatives de l'élément ciblé.
<code>scrollTop()</code>	Récupère la position verticale de la barre de défilement par rapport à la page.
<code>scrollLeft()</code>	Récupère la position horizontale de la barre de défilement par rapport à la page.

## Animation jquery

- La méthode ***toggle()*** permet d'inverser l'état de visibilité d'un élément HTML, c'est à dire de l'afficher si il est caché ou de le cacher si il est affiché.

# Les effets de fondu en jQuery

- Le jQuery permet de créer des effets de fondu, c'est-à-dire de faire disparaître ou apparaître progressivement un élément HTML, en le faisant passer de totalement transparent à totalement opaque ou inversement sur une certaine durée.
- Le jQuery dispose de quatre méthodes pour créer des effets de fondu :
  - La méthode `fadeIn()` ;
  - La méthode `fadeOut()` ;
  - La méthode `fadeTo()` ;
  - La méthode `fadeToggle()`.
- Les méthodes `fadeOut()` et `fadeIn()`, tout d'abord, vont respectivement nous permettre de faire disparaître ou de faire apparaître progressivement des éléments HTML.

# Les effets de slide en jQuery

- Le jQuery va également nous permettre de créer des effets de slide qui vont pouvoir être utilisés conjointement avec des menus déroulants par exemple.
- Pour cela, nous allons pouvoir choisir parmi trois méthodes :
  - La méthode `slideUp()` ;
  - La méthode `slideDown()` ;
  - La méthode `slideToggle()`.

# la méthode jQuery animate()

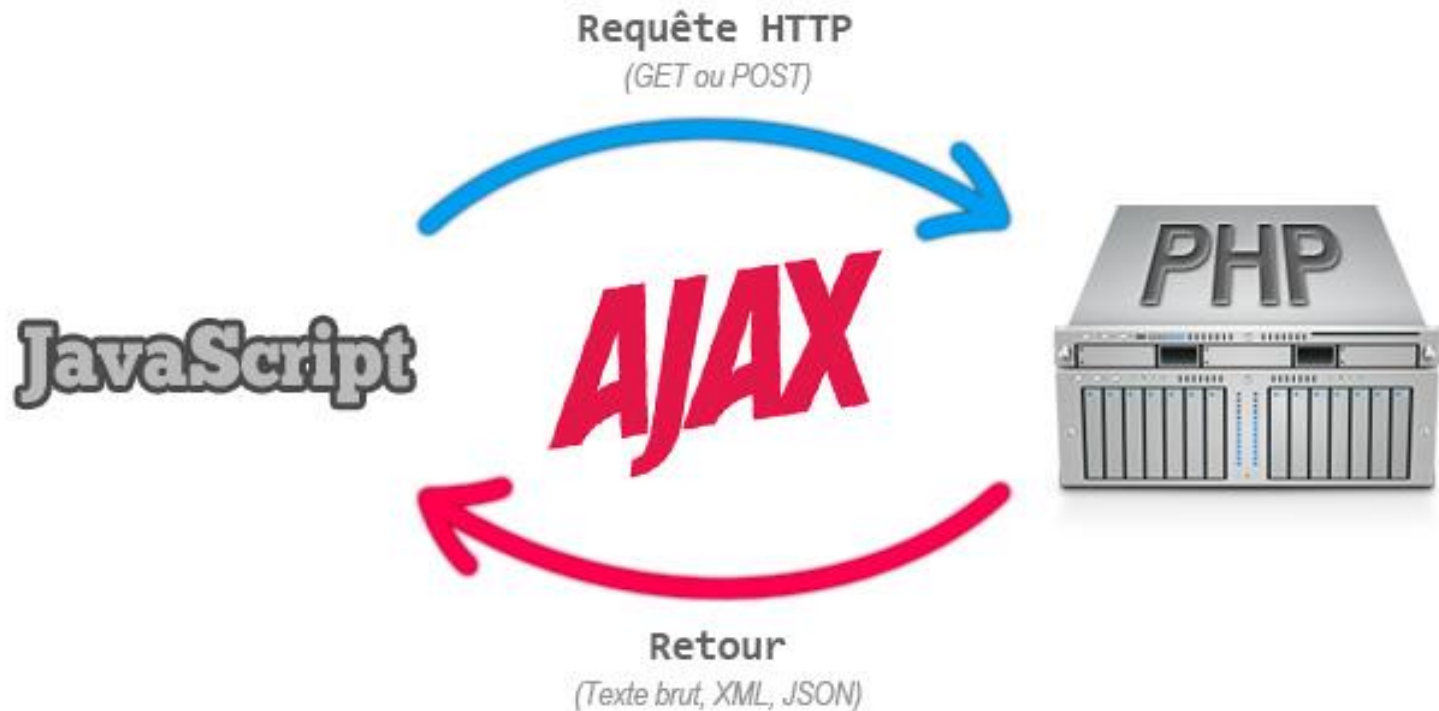
- On peut créer des effets ou animations personnalisées grâce à la méthode jQuery **animate()**.
- Pour cela, on doit fournir une ou plusieurs propriétés CSS dont on souhaite modifier le comportement sur la durée en argument de animate().
- Attention : la méthode animate() va fonctionner avec toutes les propriétés CSS de type numérique, c'est-à-dire celles dont les valeurs peuvent être numériques (comme les propriétés CSS width et height par exemple), mais nombre de propriétés CSS non numériques ne vont pas pouvoir être animées avec le jQuery « de base » (c'est-à-dire sans plugin).
- On peut animer plusieurs propriétés CSS d'un coup en utilisant une seule fois la méthode animate().



```
<script>
    $(document).ready(function(){
        $('#b1').click(function(){
            $('#h1').animate({
                width: '-=10%',
                fontSize: '20px',
                left: '100px'},
                1000);
        });
    });
</script>
```

# Ajax : Asynchronous Javascript + XML

- Ajax permet de modifier partiellement la page affichée par le navigateur pour la mettre à jour sans avoir à recharger la page entière.
- Ajax est une technique qui fait usage des éléments suivants:
  - HTML pour l'interface.
  - CSS (Cascading Style-Sheet) pour la présentation de la page.
  - JavaScript (EcmaScript) pour les traitements locaux, et DOM (Document Object Model) qui accède aux éléments de la page ou du formulaire ou aux éléments d'un fichier XML chargé sur le serveur.
  - L'objet XMLHttpRequest lit des données ou fichiers sur le serveur de façon asynchrone.
  - PHP ou un autre langage de scripts peut être utilisé coté serveur.



asynchrone

on n'attend pas le résultat des requêtes précédentes pour lancer les requêtes suivantes et la suite de l'exécution du code Javascript

## Fonctionnement :

- Ajax utilise un modèle de programmation comprenant d'une part la présentation, d'autre part les évènements.
- Les évènements sont les actions de l'utilisateur, qui provoquent l'appel de fonctions associées aux éléments de la page.
- L'interaction avec l'utilisateur se fait à partir des formulaires, des boutons ou des liens hypertextes.
- Ces fonctions JavaScript identifient les éléments de la page grâce au DOM et communiquent avec le serveur par l'objet XMLHttpRequest.

# L'objet XMLHttpRequest

- AJAX se base sur un composant embarqué dans presque tous les navigateurs récents : XMLHttpRequest
- Cet objet a d'abord été développé par Microsoft, en tant qu'objet ActiveX, pour Internet Explorer 5. Il a ensuite été repris et implémenté sous Mozilla 1 Safari 1.2, Konqueror 3.4 et Opera 8.
- Il n'est pas supporté par les navigateurs dits de « vieille génération ».
- En avril 2006, il a été proposé pour devenir une recommandation du W3C.

# Attributs de XMLHttpRequest

- **Status** : statut sous forme numérique (ex 200 OK, 404 Not Found)
- **statusText** : statut sous format texte
- **readyState** : état de l'objet (0=uninitialized, 1=open, 2=sent, 3=receiving, 4=loaded)
- **Onreadystatechange**: permet d'attacher un gestionnaire d'évènement
- **responseText**: réponse du serveur sous forme texte(HTML ou JSON)
- **responseXML**: réponse du serveur sous forme XML

lorsque l'on a reçu les données du serveur et qu'elles sont prêtes à être traitées :

- `readyState = 4`
- `Status = 200`

## méthodes de XMLHttpRequest

- Ouverture d'une connexion pour envoi ou réception de données (GET, POST) :

***open( method , url , async )***

- Envoi d'une requête :

***send(content)***

- Mettre fin a la requête :

***abort()***

# L'objet XMLHttpRequest

```
function getXMLHttpRequest() {  
    try {  
        req=new XMLHttpRequest();  
    } catch(exc1) {  
        try {  
            req=new ActiveXObject("Msxml2.XMLHTTP");  
        } catch(exc2) {  
            try {  
                req=new ActiveXObject("Microsoft.XMLHTTP");  
            } catch(exc3) {  
                req=false;  
            }  
        }  
    }  
    return req;  
}
```

```
var myRequest = new getXMLHttpRequest();
```

```
var myrequest= new XMLHttpRequest();
```



## La méthode .ajax()

- La méthode .ajax() permet de maîtriser l'ensemble des paramètres de requête. Cette méthode peut s'écrire de deux façons:

***\$.ajax(url, {options});***

ou

***\$.ajax({options});***

## Les paramètres :

- Voici les options les plus utilisées dans `.ajax()`:

Option	Description
<b>URL</b>	Adresse à laquelle la requête est envoyée
<b>type</b>	Le type de requête <i>GET</i> (par défaut) ou <i>POST</i>
<b>data</b>	Les données à envoyer au serveur
<b>datatype</b>	Le type de données pouvant être transmises au serveur : <i>php</i> , <i>html</i> , <i>script</i> , <i>json</i> et <i>xml</i>
<b>success</b>	La fonction à appeler si la requête a abouti
<b>error</b>	La fonction à appeler si la requête n'a pas abouti
<b>timeout</b>	Le délai maximum en millisecondes de traitement de la demande. Passé ce délai, elle retourne le paramètre <b>error</b> .

## Exemple

## La méthode .load()

- La méthode **.load()** permet de récupérer du contenu à insérer dans notre page. Elle peut charger des fichiers du type : HTML, PHP, TXT, XML et JSON.
- Elle est un raccourci de la fonction .ajax().

### *Syntaxe*

```
$(selecteur).load(url,data,function(reponse,status,xhr));
```

# Les paramètres :

Paramètre	Description
<b>url</b>	<b>Requis</b> Une chaîne contenant l'URL vers laquelle la demande est envoyée
<b>data</b>	<b>Optionnel</b> Spécifie les données à envoyer vers le serveur en même temps que la demande
<b>function(reponse,status,xhr)</b>	<b>Optionnel</b> Indique une fonction à exécuter quand la méthode est terminée Paramètres supplémentaires : <ul style="list-style-type: none"><li>- <b>data</b> - contient les données résultant de la demande</li><li>- <b>status</b> - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror")</li><li>- <b>xhr</b> - contient l'objet XMLHttpRequest</li></ul>

## Exemple

## La méthode .get()

- La méthode **.get()** permet de recevoir des données.
- Contrairement à la méthode **.load()**, elle permet non pas de recevoir du contenu directement dans l'élément ciblé mais de nous laisser le choix sur les actions à faire.

### *Syntaxe*

```
$.get( url, data, function(data,status,xhr), dataType );
```

# Les paramètres :

Paramètre	Description
<b>url</b>	<b>Requis</b> Une chaîne contenant l'URL vers laquelle la demande est envoyée
<b>data</b>	<b>Optionnel</b> Spécifie les données à envoyer vers le serveur en même temps que la demande
<b>function(data,status,xhr)</b>	<b>Optionnel</b> Indique une fonction à exécuter lorsque la méthode est terminée Paramètres supplémentaires : <ul style="list-style-type: none"><li>- <b>data</b> - contient les données résultant de la demande</li><li>- <b>status</b> - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror")</li><li>- <b>xhr</b> - contient l'objet XMLHttpRequest</li></ul>
<b>dataType</b>	<b>Optionnel</b> Spécifie le type de données attendu. Par défaut jQuery effectue une estimation automatique (HTML, PHP, TXT, XML et JSON)

## Exemple

## La méthode .post()

- La méthode .post() permet d'envoyer des données. Par exemple, vous pouvez l'utiliser pour envoyer des données saisies dans un formulaire.
- La méthode consiste à envoyer des données vers un fichier Php qui se chargera de les transmettre au serveur.
- Cette méthode peut également retourner des informations en ***callback*** dans la page.

### ***Syntaxe***

```
$.get( url, data, function(data,status,xhr), dataType );
```

# Les paramètres :

Paramètre	Description
<b>url</b>	<b>Requis</b> Une chaîne contenant l'URL vers laquelle la demande est envoyée
<b>data</b>	<b>Optionnel</b> Spécifie les données à envoyer vers le serveur en même temps que la demande
<b>function(data,status,xhr)</b>	<b>Optionnel</b> Indique une fonction à exécuter lorsque la méthode est terminée Paramètres supplémentaires : <ul style="list-style-type: none"><li>- <b>data</b> - contient les données résultant de la demande</li><li>- <b>status</b> - contient l'état de la demande ("success", "notmodified", "error", "timeout", ou "parsererror")</li><li>- <b>xhr</b> - contient l'objet XMLHttpRequest</li></ul>
<b>dataType</b>	<b>Optionnel</b> Spécifie le type de données attendu. Par défaut jQuery effectue une estimation automatique ( <i>xml</i> , <i>json</i> , <i>script</i> , ou <i>html</i> )

## Exemple



## TP: Ajax

- Réaliser un espace de commentaires instantané.
- Élément du dossier de travail.
  - Page html contenant le formulaire des commentaire:
    - Zone nom,
    - Zone message.
    - Zone d'affichage des messages postés,
  - Un script (**jquery**) d'envoi et de réception des commentaires.
  - Un script PHP (**ajouter.php**) pour enregistrer les message reçu dans la base de donnée.
  - Un script PHP (**envoyer.php**) pour lire les message dans la base de donnée et les envoyer à la page des commentaires.
  - Une base de données contenant la table commentaire(**id,nom,message**)

# JSON

- JavaScript Object Notation (Notation Objet issue de JavaScript) est un format léger d'échange de données.
- Il est facile à lire ou à écrire pour des humains. Il est aisément analysable ou gérable par des machines.
- JSON est un format texte complètement indépendant de tout langage, mais les conventions qu'il utilise seront familières à tout programmeur.
- Ces propriétés font de JSON un langage d'échange de données idéal.

# La syntaxe de JSON

## Les éléments de JSON:

- Un objet: contient des objets ou des variables.
- Une variable scalaire: Number, String, Boolean. Un tableau.
  - Les valeurs littérales: null, true, false, "chaîne de caractères", et les valeurs numériques.
- L'objet contient un membre ou une liste de membres, chaque membre étant de la forme:

*"nom" : "valeur"*

## La syntaxe de l'objet est:

*{ membre, membre, .... }*

**Le tableau** contient une ou plusieurs valeurs séparées par des virgules.

*[valeur, valeur .... ]*

- Les valeurs peuvent être: un objet, un tableau, un littéral (chaîne, nombre, true, false, null)

# Exemple d'un fichier JSON Vs XML

```
{
  "menu": "Fichier",
  "commandes": [
    {
      "title": "Nouveau",
      "action": "CreateDoc"
    },
    {
      "title": "Ouvrir",
      "action": "OpenDoc"
    },
    {
      "title": "Fermer",
      "action": "CloseDoc"
    }
  ]
}
```

```
<?xml version="1.0" ?>
<root>
  <menu>Fichier</menu>
  <commands>
    <item>
      <title>Nouveau</value>
      <action>CreateDoc</action>
    </item>
    <item>
      <title>Ouvrir</value>
      <action>OpenDoc</action>
    </item>
    <item>
      <title>Fermer</value>
      <action>CloseDoc</action>
    </item>
  </commands>
</root>
```

# Utilisation du format JSON

**Un fichier utilisant ce format permet de :**

- Charger de l'information à partir du serveur
- Transmettre au serveur de l'information (contenu d'un formulaire).
- Il y a donc trois aspects:
  - Le traitement par le navigateur,
  - Le traitement par serveur,
  - La transmission des données entre les deux.

## ***Coté client :***

- JSON fait partie de la norme JavaScript. Son contenu est assigné à une variable devenant un objet du programme.

## ***Coté serveur :***

- Utilisation de parseurs (*json.org*)

## ***La récupération de données :***

- la récupération d'un fichier peut se faire à partir de JavaScript de plusieurs façons:
  - L'échange de données inclusion directe du fichier dans la page HTML au même titre qu'un fichier .js de JavaScript.
  - Chargement par une commande JavaScript.
  - Emploi de ***XMLHttpRequest***.
- Le fichier JSON est parsé par la fonction JavaScript `eval()`.



## Exemple :

Afficher le contenu d'un fichier json avec XMLHttpRequest.

## JqueryUI :

- Si jQuery offre de très nombreuses méthodes pour gérer le DOM, les propriétés CSS, AJAX et la gestion événementielle, jQuery UI le complète parfaitement en offrant des méthodes additionnelles appliquées à la réalisation de l'interface utilisateur.
- JQuery UI est en quelque sorte un vaste assemblage de plugins accessibles à travers un seul fichier JavaScript.