

# Techniques de publication électronique III : Javascript

Clément Hallet

École nationale des chartes - février 2014

# Sommaire

- Historique et Introduction
- Etude du langage
- Outils et environnement applicatif

# ***Historique et Introduction***

# **Historique et Introduction**

## **Présentation**

Javascript est un language de script qui permet de programmer et manipuler des pages web pour les rendre interactives.

### **Définition de *script* :**

Programme s'exécutant dans un environnement fourni par un programme hôte, et permettant de contrôler ses fonctionnalités.

### **Exemples :**

- AppleScript : Mac OS
- VBScript : Windows et Microsoft Office
- JavaScript : dans l'environnement des navigateurs web.

# Historique et Introduction

## Création et standardisation de Javascript

- Mars 1996 : première version de Javascript avec le navigateur *Netscape Navigator 2.0*
- Août 1996 : version concurrente de Jscript avec le navigateur *Internet Explorer 3.0*
- Juin 1997 : Standardisation des deux langages par l'organisme *Ecma International*. Les rendant, en théorie, inter-compatibles.

# **Historique et Introduction**

Adoption généralisée > Compatibilité multi-navigateurs

Au début des années 2000, les différences d'implémentations entre JavaScript et Jscript empêchent encore une généralisation de JavaScript, malgré la standardisation ECMAScript.

**Et maintenant ?**

Bien que toujours officiellement nommé Jscript dans Internet Explorer, on parle de JavaScript pour tous les navigateurs. Quelques différences substantielles persistent quand même.

# **Historique et Introduction**

Adoption généralisée > librairies d'uniformisation de l'API

En 2005, des librairies Javascript sont créées pour uniformiser l'API des navigateurs.

**Exemples :** Prototype, Dojo, Mootools, jQuery

Grâce à la meilleure compatibilité qui en résulte, les usages de Javascript vont se développer.

**Et maintenant ?**

Les APIs natives des navigateurs sont assez similaires. Certaines librairies Javascript continuent d'exister pour les fonctions utilitaires ou raccourcies qu'elles apportent.

# Historique et Introduction

Adoption généralisée > librairies d'uniformisation de l'API

Définition de *librairie* :

Ensemble de classes et fonctions de bases, qui étendent le noyau du langage.

Définition de *API* (*Interface de Programmation Applicative*) :

Ensemble de classes et fonctions permettant d'interagir avec un système tierce.

Dans le cas de JavaScript, l'API est fournie par le navigateur et elle permet un accès logiciel à la page affichée.

# Historique et Introduction

Adoption généralisée > librairies d'uniformisation de l'API

## Exemple de différences pour un gestionnaire d'évènement

JavaScript

```
1 | var a = document.getElementById('link');  
2 | link.addEventListener('click', callback, false);
```

JScript

```
5 | var a = document.getElementById('link');  
6 | link.attachEvent('onclick', callback);
```

# Historique et Introduction

Adoption généralisée > librairies d'uniformisation de l'API

**Extrait de la librairie Prototype, uniformisant la gestion des évènements.**

```
770     function observeStandardEvent(element, eventName, responder) {
771         var actual(eventName = getDOMEventName(eventName));
772         if (element.addEventListener) {
773             element.addEventListener(actual(eventName, responder, false);
774         } else {
775             element.attachEvent('on' + actual(eventName, responder);
776         }
777     }
778 }
```

# **Historique et Introduction**

## Transition dans les usages de Javascript

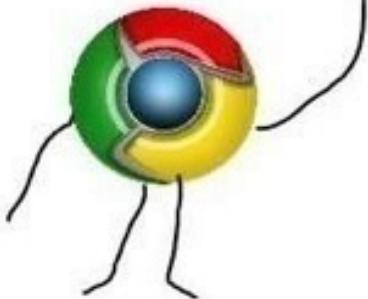
- Animations visuelles des éléments de la page
- Requêtes HTTP asynchrones
- passage de la notion de script à celle d'application

# **Historique et Introduction**

## Applications Javascript marquantes

- Gmail (2004)
- Google Maps (2004)
- Netvibes (2005)

what are we?!



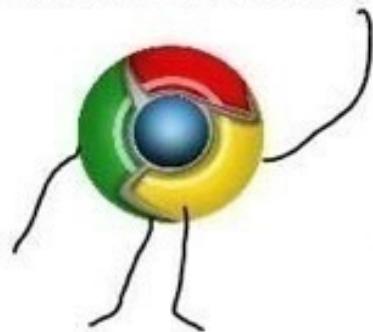
browsers!



browsers



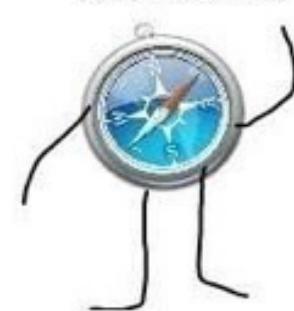
what do we want?!



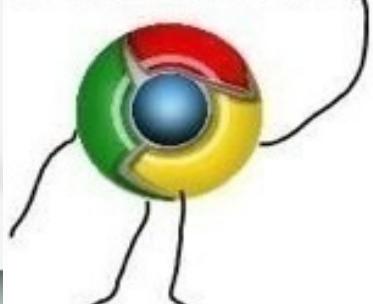
fast internet!



fast internet!



when do we want it?!



browsers!



# **Historique et Introduction**

Autres technologies associées

## **HTML5**

Format de données utilisé par les navigateurs modernes. Une API spécifique lui est associée pour, par exemple, lire des vidéos.

## **CSS3**

Feuilles de styles en cascade. La version 3 ajoute notamment la gestion native des animations visuelles.

## **Canvas**

Permet un rendu 3D dans une zone spécifique.

# **Historique et Introduction**

Autres technologies associées

## **SVG**

Images vectorielles définies en XML

## **Websockets**

Connexion persistante à un serveur distant (à opposer à HTTP où la connexion est temporaire)

## **Local storage**

Permet d'embarquer une micro base de données dans le navigateur pour y stocker des informations hors ligne.

# **Historique et Introduction**

## Autres langages dérivés

### **ActionScript**

Utilisé pour scripter les animations *Flash*. Les langages sont très semblables car reposant sur la même norme ECMAScript. L'API est cependant très différentes (page Web vs animation Flash)

### **Node.js**

Serveur web et framework côté serveur. Là aussi l'API est totalement différentes : ce n'est pas une page qui est manipulée, mais principalement des requêtes HTTP.

# Historique et Introduction

## Exemples d'application utilisant Javascript

### Trello

<https://trello.com/b/EYrXmsXi/cours-javascript>

The screenshot shows a Trello board titled "cours Javascript". The board has four main lists:

- Concepts**: Contains cards for DOM, Requête HTTP, JSON, Librairies, Prototype, and Evenements (pattern). A "Add a card..." button is at the bottom.
- Librairies**: Contains cards for jQuery, Backbone, Underscore, and a "Add a card..." button.
- Outils**: Contains cards for console de debug, GitHub, Stack Overflow, Mailing List, and a "Add a card..." button.
- Outils de représentation des données**: Contains cards for Timeline, Raphael, Google Map, Open Street Maps, Simile, and a "Add a card..." button.

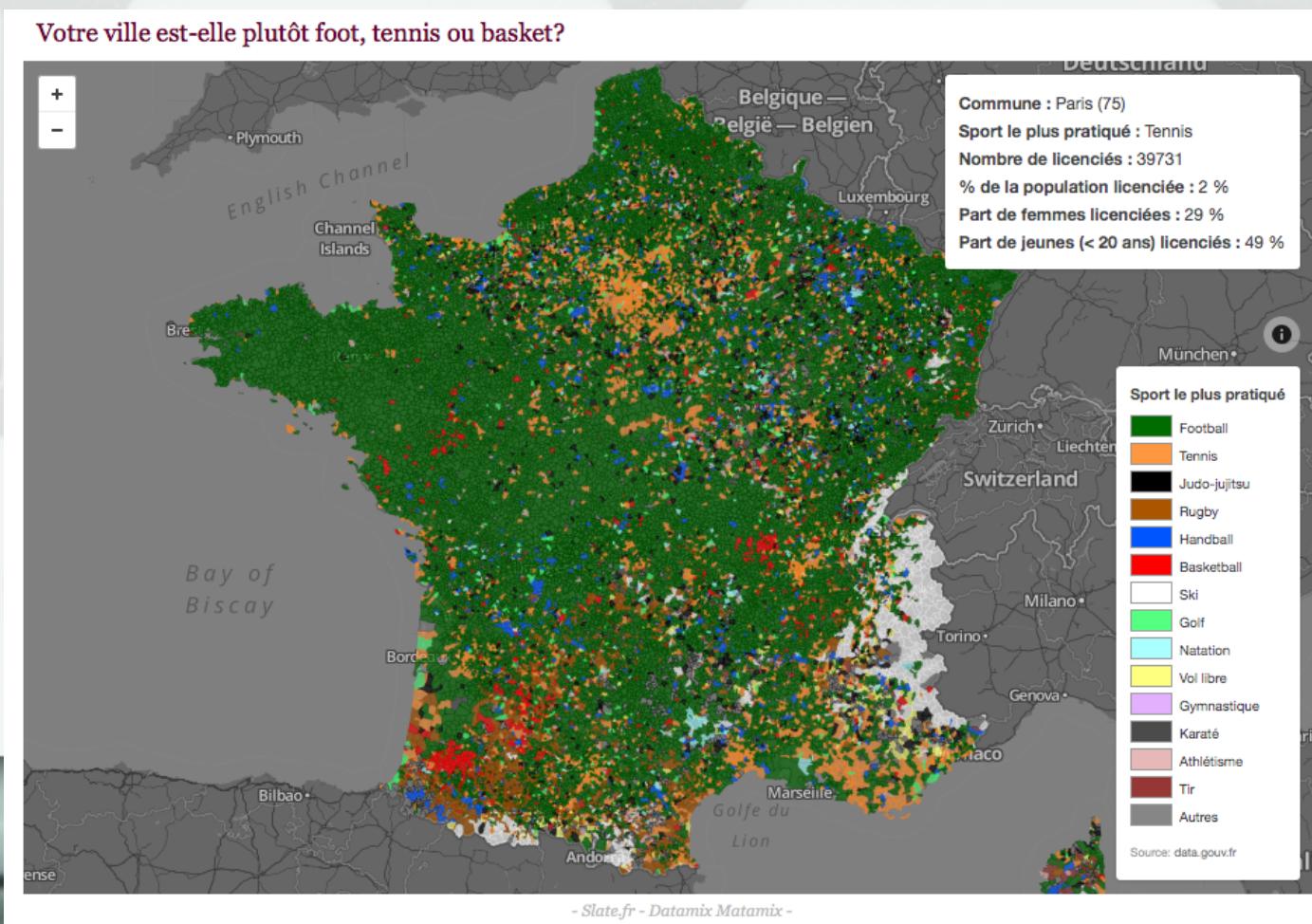
At the top left, there are buttons for "Boards" and "Private". At the top right, there are user info ("Clément Hallet") and a notification bell icon. A "Show sidebar" link is also visible.

# Historique et Introduction

## Exemples d'application utilisant Javascript

### Carte interactive

<http://www.slate.fr/story/79066/carte-france-sports-plus-pratiques-commune>

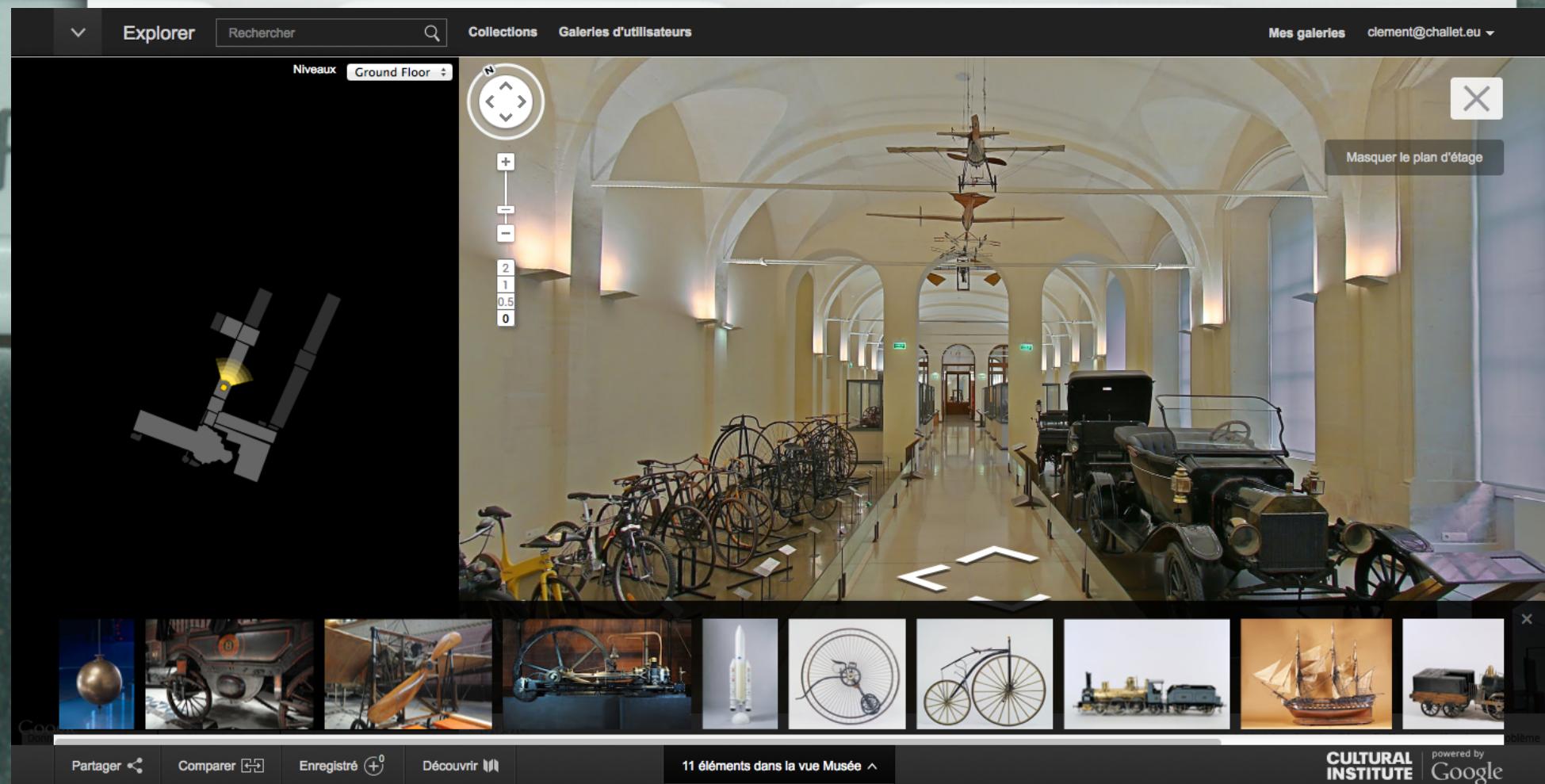


# Historique et Introduction

## Exemples d'application utilisant Javascript

### Visites virtuelles de musées

<http://www.google.com/culturalinstitute/>



# Historique et Introduction

## Exemples d'application utilisant Javascript

### Représentations temporelles et géographiques

<http://test.laderdesders.fr/>

Il y a 100 ans  
Autour de moi

Timeline SIMILE

La guerre de position

1916

Bataille de Verdun

Offensives franco-britanniques en Somme

Le général Nivelle remplace Joffre

1917

05/06/1916

Verdun

Châlons-en-Champagne

Vitry-le-François

Saint-Dizier

Bar-le-Duc

Pont-à-Mousson

LORRAINE

Toul

Nancy

Dieuze

Forêt de Rémilly

Metz

Rombas

Maizières-lès-Metz

Fameck

Briey

Jarny

Thionville

Esch-sur-Alzette

Longwy

Dillingen / Saar

Saarbrücken

Völklingen

Freyming

Saint-Avold

Merzig

A 8

A 30

A 4

A 31

**Bataille de Verdun (1916)**

La bataille de Verdun fut une bataille de la Première Guerre mondiale qui eut lieu du 21 février au 19 décembre 1916 près de Verdun en France, opposant les armées françaises et allemandes. Conçue par le général Erich von Falkenhayn, commandant en chef de l'armée allemande, d'après la version qu'il en donna dans ses Mémoires, comme une bataille d'attrition pour « saigner à blanc l'armée française » sous un déluge d'obus dans un rapport de pertes de un pour deux, elle se révélera en fait presque aussi coûteuse pour l'attaquant : elle fit plus de 714 231 morts, disparus ou blessés, 362 000 soldats français et 337 000 allemands, une moyenne de 70 000 victimes pour chacun des dix mois de la bataille. On peut noter que selon les travaux historiques récents, notamment ceux de l'historien allemand Holger Afflerbach, l'objectif allemand était plus simplement de prendre le saillant de Verdun, la version d'une bataille d'attrition étant une justification inventée après-coup par Falkenhayn pour masquer son échec.

C'est la plus longue et l'une des batailles les plus dévastatrices de la Première Guerre mondiale et de l'histoire de la guerre. Verdun

Leaflet | Data, imagery and map information provided by MapQuest, OpenStreetMap and contributors, ODbL

*Etude du langage*

# Etude du langage

## Structures du langage > Les tests conditionnels

### If / else / else if

En fonction de l'expression testée, et de la valeur booléenne renvoyée, une partie spécifique du code sera exécutée.

Pour aller plus loin :

<http://goo.gl/VXWFxV>

```
1 var date = new Date();  
2 var heure = date.getHours();  
3  
4 if(heure <= 12) {  
5   alert('Bonjour');  
6 } else if (heure < 18) {  
7   alert('Bon après-midi');  
8 } else {  
9   alert('Bonsoir');  
10 }
```

# Etude du langage

## Structures du langage > Les boucles conditionnelles

Une boucle conditionnelle permet de répéter l'exécution des instructions, toujours selon une condition booléenne.

### Boucle **while** (tant que)

```
1 var ok = false;-
2 -
3 while(!ok) {-
4   ok=confirm('Vous devez valider');-
5 }-
6 alert('Vous avez validé');
```

# Etude du langage

## Structures du langage > Les boucles conditionnelles

### Boucle *do-while*

La boucle *do-while* est une variante de la boucle *while*, où la condition est évaluée à la fin. Cela assure donc au moins une exécution des instructions.

```
1 do {-
2   var ok = confirm('Vous devez valider');-
3 } while(!ok);-
4 -
5 alert('Vous avez validé');
```

# Etude du langage

## Structures du langage > Les boucles conditionnelles

### Boucle *for*

La boucle *for* est plus complexe et permet une gestion plus fine des instructions. Elle est composée de :

- Une instruction d'initialisation
- Une condition de continuation
- Une instruction d'itération

```
1 var names = ['Roger', 'Martine', 'Jean', 'Josianne'];
2 for(var i = 0; i < names.length; i++) {
3   console.log(names[i]);
4 }
```

# Etude du langage

## Structures du langage > Les fonctions

### Appel de fonctions

C'est unité d'exécution, conçue pour une tâche précise.

Elle se déclenche lorsqu'elle est *appelée* depuis une autre unité d'exécution.

Elle peut recevoir des *paramètres* en entrée, et *renvoyer* un résultat en sortie.

Une fonction peut être *native* ou déclarée dans un script.

# Etude du langage

## Structures du langage > Les fonctions

### Appel de fonctions

```
1 var guerre_balkans = new Date(1912, 9, 8);-
2 var invasion_belgique = new Date(1914, 2, 8);-
3 -
4 console.log( pendantLaGuerre( guerre_balkans ) );-
5 console.log( pendantLaGuerre( invasion_belgique ) );-
```

- Comment est nommée la fonction ?
- Quels sont les paramètres passés à la fonction?
- Quel résultat peut on attendre en retour ?

# Etude du langage

## Structures du langage > Les fonctions

### Déclaration de fonctions

Les arguments passés en paramètres lors de l'appel (extérieur) sont nommés pour être utilisés dans la fonction.

Une valeur est renvoyée (*return*) qui correspond au résultat de l'opération, à partir des paramètres reçus.

```
1 |function pendantLaGuerre(date) {  
2 |  
3 |    // /\ la numérotation des mois commence à 0  
4 |    var debut = new Date(1914, 6, 28);  
5 |    var fin = new Date(1918, 10, 11);  
6 |  
7 |    return (date >= debut) && (date <= fin);  
8 |  
9 |}
```

# Etude du langage

Structures du langage > Les fonctions

Vu dans l'exemple précédent

## Les commentaires

```
18 // commentaire sur une ligne
19 console.log("code exécuté");
20
21 /*
22  * commentaires sur plusieurs lignes
23  */
24
25
26 // console.log("lui non plus");
```

## Utilisation de l'objet Date

- Les paramètres pour instancier une date sont :

new Date(annee, mois, jour, heure, minute,  
... )

- Aucun paramètre signifie la date et l'heure actuelle
- La numérotation des mois commence à 0
- Les dates peuvent être comparées entre elles

# Etude de la librairie jQuery

## Introduction

L'utilisation de jQuery commence avec la fonction \$ qui permet de sélectionner des éléments dans le document.

Une fonction est ensuite appliquée sur ce(s) élément(s) sélectionnés

```
1 $(document).ready(function () {  
2     //  
3     $('#debutguerredecentans').css('color', 'red');  
4     $('#finguerredecentans').css('color', 'green');  
5     //  
6 });
```

# Etude de la librairie jQuery

## Introduction

### Vu dans l'exemple précédent

La fonction `.ready()` appliquée sur le document permet de déclencher l'exécution quand on est assuré que toutes les ressources externes sont chargées.

La fonction `.css()` permet d'appliquer un style CSS sur les éléments sélectionnés.

# Etude de la librairie jQuery

## Les sélecteurs CSS

La sélection se fait par des expressions appelées sélecteurs CSS : ils ont le même fonctionnement que les règles du même nom utilisées dans les feuilles de styles.

exemple	définition	éléments sélectionnés
#element	sélection par id	<p id="element">du texte</p>
.lieu	sélection par classe	<p class="lieu">Saint Martin</lieu>
p	sélection par type d'élément	<p class="lieu"></p> <p class="date"></p>
p.date	combinaison type et classe	<p class="lieu"></p> <p class="date"></p>
div p	sélection par hiérarchie	<div class="date"> <p class="lieu"></p> </div>
div p.selected	combinaison hiérarchie et classe	<div> <p></p> <p class="selected"></p> </div>

# **Etude de la librairie jQuery**

Notion importante : définition du DOM

Le Document Object Model est le nom donné à l'interface exposé par le navigateur pour accéder ou modifier le contenu de la page.

Il s'agit d'une structure en arbre où les éléments de la page et leurs attributs sont hiérarchisés.

**Voir :** panneau DOM de la console de debug

# Etude de la librairie jQuery

## Principales fonctions jQuery

### *Traversing : Parcourir le DOM*

Fonctions qui permettent de sélectionner des éléments adjacents.

#### Exemples :

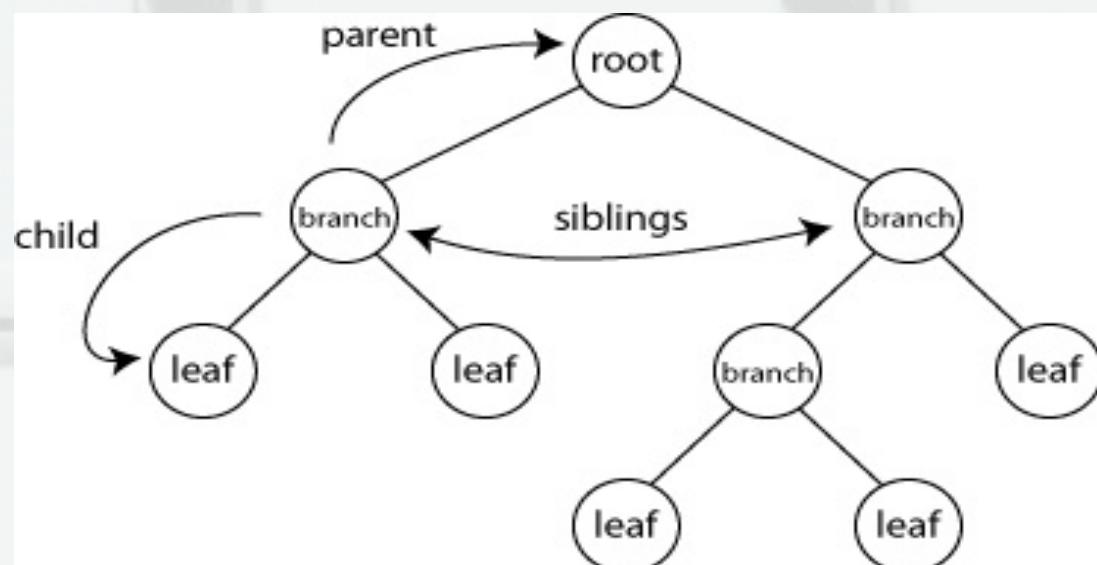
`$('#element').next()`

`$('#element').previous()`

`$('#element').parent()`

`$('#element').children()`

`$('#element').find()`



# **Etude de la librairie jQuery**

## **Principales fonctions jQuery**

### **CSS : Modifier les styles d'un élément**

Fonctions qui interagissent avec les styles CSS d'un élément

#### **Exemples :**

`$(().hasClass('class'))` : teste si la classe CSS est appliquée sur l'élément

`$(().addClass('class'))` : ajoute la classe CSS à l'élément

`$(().removeClass('class'))` : enlève la classe CSS de l'élément

Voir : TP transitions CSS en basculant la classe.

# **Etude de la librairie jQuery**

## **Principales fonctions jQuery**

### **Ajax : Requêtes de chargement de données**

Déclenche des requêtes HTTP vers un serveur pour envoyer des données (provenant d'un formulaire par exemple).

Ou pour charger des données en provenance de ce même serveur.  
Par exemple aux formats XML ou JSON.

**Voir :** TP chargement et manipulations de fichier JSON

# Etude de la librairie jQuery

## Principales fonctions jQuery

### ***Events : Réagir aux évènements de l'utilisateur***

Les principaux évènements qui peuvent activer un gestionnaire sont liés à l'interface avec l'utilisateur :

- Cliques et déplacements de souris
- Modification d'un élément de formulaire
- Navigation dans la page

# Etude de la librairie jQuery

## Principales fonctions jQuery

### Events : Réagir aux évènements de l'utilisateur

Pour réagir à ces évènements on met en place des *gestionnaires d'évènements* via la fonction **on** qui prend en paramètres :

- Le nom de l'évènement auquel réagir
- La fonction (gestionnaire d'évènement) qui sera exécutée

```
1 $(document).ready(function () {  
2     //  
3     $('header a').on('click', function(e) {  
4         e.preventDefault();  
5         alert("vous avez cliqué sur '" + $(this).text() + "'");  
6     });  
7     //  
8});
```

# Etude de la librairie jQuery

## Principales fonctions jQuery

### **Events : Réagir aux évènements de l'utilisateur**

Quelques événements importants :

- click : un clique de la souris sur un élément
- mouseover / mouseout / mousemove : les mouvements du pointeur de la souris au dessus d'un élément
- focus : un contrôle de formulaire est sélectionné par l'utilisateur
- change : un contrôle de formulaire a changé (cases à cocher et boutons radios)
- submit : un formulaire est envoyé (clique sur le bouton *submit*)

**Voir : TP events.js**

# Bibliographie

- Dynamisez vos sites web avec Javascript
- Mozilla Developer Network
- jQuery : écrivez moins pour faire plus !
- jQuery API Documentation
- Stack Overflow frequent Javascript questions
- JSON: What It Is, How It Works, & How to Use It