

M4103C

Programmation Web – Client riche

Support de cours – Partie 1

DUT Informatique – 2^{de} année

Matthieu Le Lain, Nicolas Le Sommer, Goulven Kerbellec
IUT de Vannes

Plan global

- 1. Rappels sur les langages de programmation Web
- 2. AJAX & JQuery
- 3. AngularJS
- 4. Cordova



Plan – Chapitre 1

- Rappels – Langage de programmation Web (client)
 - 1.1 HTML5
 - 1.2 CSS3
 - 1.3 JavaScript
 - 1.4 DOM

Chapitre N°1

Programmation web – Client riche

Client riche

Client riche

Les clients dits riches permettent de mettre en œuvre une interface graphique un peu plus évoluée qu'un client léger et pouvant mettre en œuvre des fonctionnalités semblables à un client dit lourd. Par ailleurs, les traitements relativement lourds sont effectués sur le serveur. La réponse est ensuite envoyée au poste client, ce dernier pouvant la prendre en compte, effectuer d'autres traitements plus légers et la présenter. Ainsi, le traitement ne se situe pas uniquement du côté client, ou serveur...

Client lourd

avant 2000

Client riche / Riche internet application

Après 2002

Client léger

À partir de 2000

1.1 – HTML5

HTML



- **1970 : Naissance du protocole TCP/IP**, uniformisation des règles communes de transit des informations (notamment textuelles) dans un réseau.
- **1984 Naissance du protocole HTTP**. Transmission d'un ensemble de pages à travers le réseau (texte simple et liens).
- **1990 : Naissance du langage HTML** (HyperText Markup Language) et Netscape/Mosaic. Structuration des pages transmises sur le réseau grâce aux éléments de description (balise).
- **1994 : Fondation du consortium W3C**. Promouvoir la standardisation des langages associés à Internet

- **1995 : naissance de JavaScript.** Langage de programmation ajoutant de l'interactivité dans les pages web.
- **1996 : Naissance de CSS** (Cascading Style Sheet). Feuille de style permettant de mettre en forme les éléments qui composent les pages web. Séparation du fond et de la forme.
- **1997 : Naissance de XHTML.** Syntaxe plus stricte de l'écriture des pages web.
- **2007/2008 : Première ébauche d'HTML5**
- **2014 : Standardisation de l'HTML5 par le W3C**

D'une manière globale, avec l'arrivée du HTML5 on ne parle plus de pages web, mais dorénavant d'application web. Le HTML5 est une évolution du HTML4, principalement conçu par les navigateurs les plus actifs du web (Firefox, chrome, opéra, safari).

Ces applications web, de plus en plus performantes, sont généralement construites par l'association de trois langages : HTML5 + CSS3 + JavaScript.

Cela permet ainsi de faire du HTML5, une plateforme d'interface d'applications et d'intégrer des fonctionnalités complexes comme la géolocalisation, le drag & drop, etc.

HTML5

5.1



Tous les navigateurs n'ont pas encore intégré totalement le HTML5 ni le CSS3. Il est donc important lors de la conception d'application web de prévoir une compatibilité maximale aux clients qui vont devoir traiter ces dernières.

Plusieurs sites permettent de voir à quel niveau, et selon quelles versions, les navigateurs sont capables de prendre en compte les fonctionnalités du HTML5 ! En voici une liste non exhaustive

:

- <https://html5test.com>
- <http://browsershots.org/>
- <http://caniuse.com/>



Nouvelles balises d'organisation

5.1

<header></header>

Éléments du menu de navigation

<footer>/footer>

Éléments du pied de page

<aside></aside>

Éléments annexes au contenu

<section></section>

&

<article></article>

Il est également possible de diviser encore le contenu principal en d'autres parties avec les balises

En-tête

Navigation

Contenu annexe

Pied de page

Balises sémantiques

5.1

De nouvelles balises sémantiques ont également fait leur apparition :

`<hgroup>`

- Balise permettant de recevoir un groupe de balise (par exemple les titres)

`<time>`

- Définition d'une date et d'un heure (`<time datetime="2015-02-12T00:14:00+01:00"> Le 12 février à 14H</datetime>`)

`<mark>`

- Mise en évidence du texte

`<meter>`

- Définition d'une mesure

`<figure>`

- Groupement d'éléments (image, vidéos, etc..)

`<details> & <summary>`

- Détail et résumé d'éléments

Balises de saisie de texte

5.1

Dès lors que le développeur veut pouvoir récupérer de l'information venant des utilisateurs il lui faut prévoir des zones de saisies de texte. Toutefois jusqu'alors il était nécessaire d'effectuer la vérification de la saisie avec un script construit par ses soins. Le HTML5 permet de résoudre cette besogne avec l'ajout de balises de vérification de texte, bien que cette fonctionnalité ne soit pas encore supportée par tous les navigateurs...

Email : <*input type="email" name="mail"*>

URL : <*input type="url" name="web"*>

Numérique : <*input type="number" name="n" min="2" max="10" value="4"*>

Date : <*input type="date" name="in"*><*br*>

Mais aussi : week, month, time, color, search, range,

1.2 – CSS3

CSS



La séparation entre la structure d'une page web et son style est encore plus prononcée avec l'arrivée du HTML5, car ce dernier n'intègre quasiment plus aucune balise ou attribut permettant de préciser la présentation. La première ébauche de « CSS3 » voit le jour vers l'année 2009.

Tout comme le HTML5, la compatibilité des navigateurs est à prendre au sérieux concernant les diverses fonctionnalités présentes au sein du CSS3.

Rappel du fonctionnement :

Sélecteur { propriété : valeur; }

Exemple : div { background-color : green; }

Balise
de flux

- Body div p...

Id

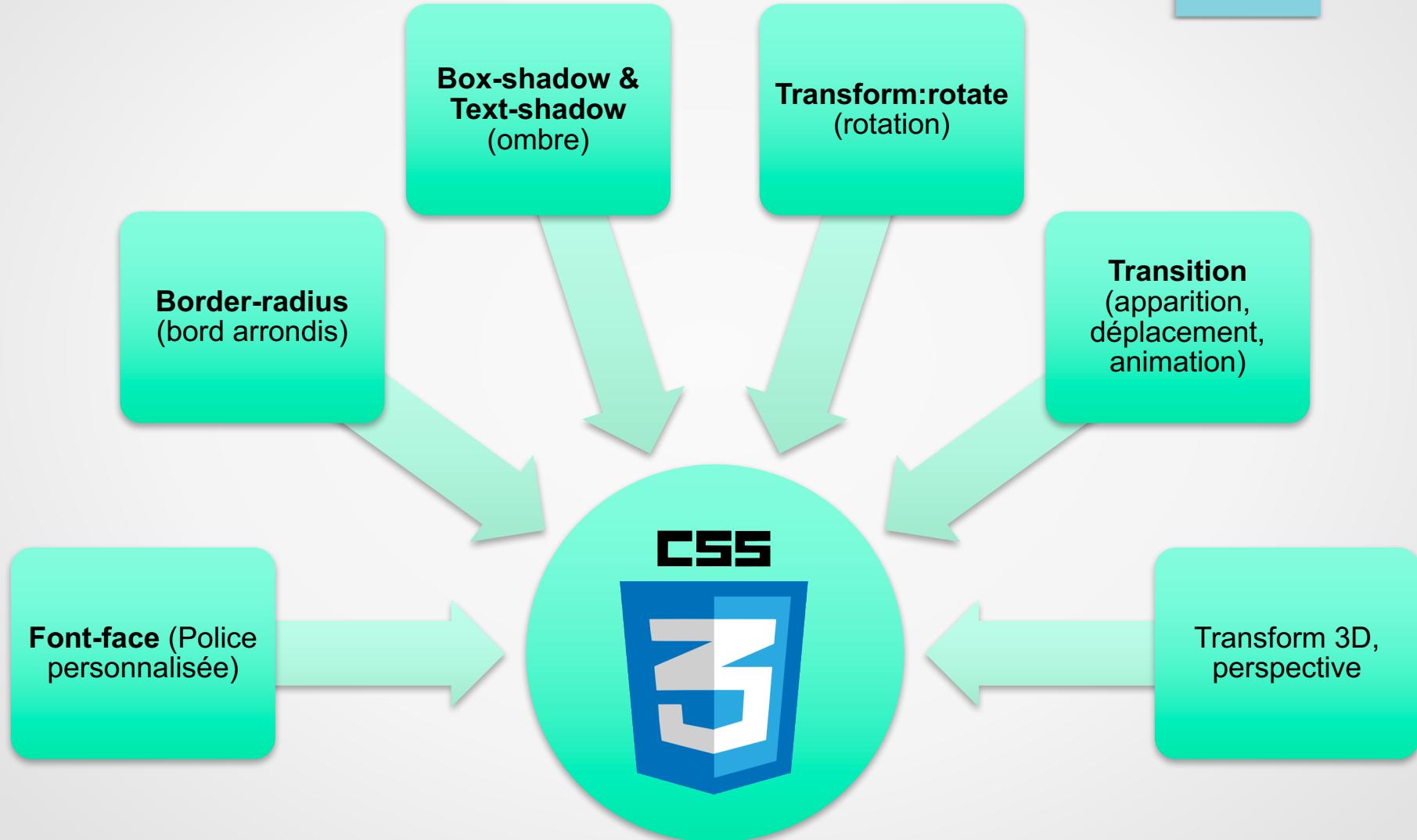
- #content

Classe

- .citation

CSS 3 - Nouveautés

5.2



Media Queries

Aujourd'hui, la multiplicité des plateformes et de leurs tailles complexifie la construction des designs. Pour répondre à cette problématique, les développeurs utilisent les **Media Queries**.

Ainsi, en fonction du type (écran de pc, mobile, TV, projecteur), de la taille de l'écran, de sa capacité à afficher les couleurs ou de son orientation, une feuille de style différente s'applique. Ces media queries font partie des techniques utilisées pour effectuer un style adaptatif, encore appelé **Responsive Design**.

Déclaration au sein de la page HTML

```
<link rel="stylesheet" media="screen and (max-width: 1280px)" href="mobil.css" />
```

Ou directement au sein de la feuille de style

```
@media screen and (max-width: 1280px){ ... }
```

Media Queries - Attributs

5.2

Color

Height

Width

Device-
Height

Device-
width

Orientation

Media

Dans la majorité des cas, il est possible d'adoindre un préfixe **min/max** devant chaque attribut pour encadrer les valeurs limites. De même pour ajouter plusieurs conditions, il est possible d'utiliser les mots clés **and, or et not**.

Exemple :

`@media tv and (min-width: 1024px) and (max-width: 1280px)`

Screen

Handheld

Print

Tv

Projection

All

À noter toutefois que l'attribut Handheld pour préciser le média « mobile » n'est quasiment pas supporté par les navigateurs à ce jour. Plusieurs thèmes responsives permettent de faciliter le travail des développeurs. (bootstrap : <http://getbootstrap.com/>)

1.3 – JavaScript

JS



Le JavaScript est un langage interprété par le navigateur conçu par Netspace Communication en 1995, standardisé en 1996. Ce langage permet de rajouter de l'interactivité et du dynamisme aux pages web en manipulant le DOM (Document Object Model), nous y reviendrons plus loin.

On peut alors, créer des balises au sein de la page, modifier leurs attributs, en supprimer sans avoir besoin de recharger la page ni aucun appel auprès d'un serveur.

Toutefois, ce langage permet aussi les échanges de données avec un serveur (AJAX), sans aucun rechargement encore une fois.

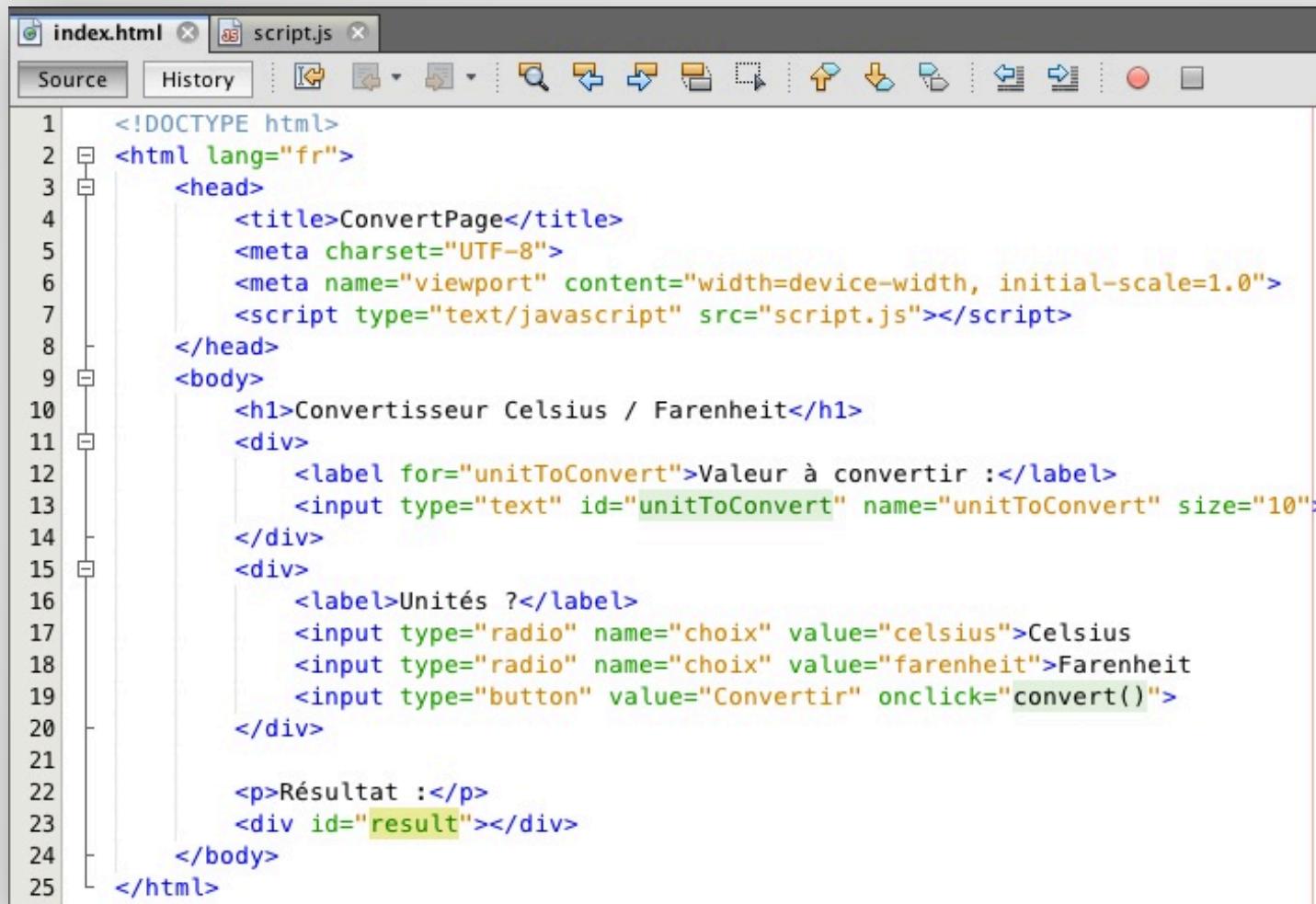
Implémentation au sein de la page HTML :

<script type="text/JavaScript" src="mon_script.js"> </script>

Ou en utilisant directement la balise <script></script>

JavaScript

5.3



The screenshot shows a code editor interface with two tabs: "index.html" and "script.js". The "index.html" tab is active, displaying the following HTML code:

```
1  <!DOCTYPE html>
2  <html lang="fr">
3      <head>
4          <title>ConvertPage</title>
5          <meta charset="UTF-8">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7          <script type="text/javascript" src="script.js"></script>
8      </head>
9      <body>
10         <h1>Convertisseur Celsius / Farenheit</h1>
11         <div>
12             <label for="unitToConvert">Valeur à convertir :</label>
13             <input type="text" id="unitToConvert" name="unitToConvert" size="10">
14         </div>
15         <div>
16             <label>Unités ?</label>
17             <input type="radio" name="choix" value="celsius">Celsius
18             <input type="radio" name="choix" value="farenheit">Farenheit
19             <input type="button" value="Convertir" onclick="convert()">
20         </div>
21
22         <p>Résultat :</p>
23         <div id="result"></div>
24     </body>
25 </html>
```

Plutôt que de parcourir le fonctionnement du langage de manière exhaustive, prenons un exemple pour en discuter.

JavaScript

5.3

D'une manière globale, on retrouve dans le JavaScript les possibilités des langages de programmation (variables, tableaux, méthodes, objets, surcharges, héritages, etc..)

Convertisseur Celsius / Farenheit

Valeur à convertir :
Unités ? Celsius Farenheit

Résultat :

68 °F

The screenshot shows a code editor with two tabs: 'index.html' and 'script.js'. The 'script.js' tab is active, displaying the following code:

```
1 //Le javascript est un langage faiblement typé.  
2 //Une variable déclarée avec le mot clé var est locale (déclaration explicite).  
3 // Une variable déclarée sans le mot clé var est globale (déclaration implicite)  
4  
5 function convert(){  
6  
    //Récupération de la valeur saisie par l'utilisateur  
8     var value = document.getElementById("unitToConvert").value;  
9  
10    //Si la checkbox celsius est cochée  
11    if(document.getElementsByName("choix")[0].checked){  
12        resultat = (value - 32) * 5/9;  
13    }  
14  
15    //Si la checkbox farenheit est cochée  
16    else if(document.getElementsByName("choix")[1].checked){  
17        resultat = (value * 9/5 + 32) + " °F";  
18    }  
19  
20    else{  
21        alert("Veuillez choisir une unité de conversion");  
22    }  
23  
24    //Écriture du résultat dans la div ayant l'id result  
25    document.getElementById("result").innerHTML = resultat;  
26  
27}
```

Asynchronous JavaScript and XML

L'AJAX est une architecture informatique qui permet d'effectuer des appels auprès d'un serveur sans avoir à recharger la page. On peut ainsi modifier le contenu de la page ce qui permet de donner plus de dynamisme et d'interactivité. On dit que l'appel est asynchrone, le code JavaScript continue son exécution sans attendre une réponse du serveur, cette dernière sera traitée dès son retour. Cela évite ainsi de geler l'interaction de l'utilisateur avec le navigateur durant l'attente de retour par exemple.

Derrière la mise en œuvre de cette technique se cache:

- **Le JavaScript et le DOM** : présente et/ou modifier la réponse du serveur
- **L'Objet XMLHttpRequest** pour effectuer les appels asynchrone
- **Le XML ou JSON** structurant les informations transmises.

JavaScript – Ajax - Exemple

5.3

```
<?xml version="1.0" encoding="UTF-8"?>
<query>
    <status>success</status>
    <country><![CDATA[COUNTRY]]></country>
    <countryCode><![CDATA[COUNTRY CODE]]></countryCode>
    <region><![CDATA[REGION CODE]]></region>
    <regionName><![CDATA[REGION NAME]]></regionName>
    <city><![CDATA[CITY]]></city>
    <zip><![CDATA[ZIP CODE]]></zip>
    <lat><![CDATA[LATITUDE]]></lat>
    <lon><![CDATA[LONGITUDE]]></lon>
    <timezone><![CDATA[TIME ZONE]]></timezone>
    <isp><![CDATA[ISP NAME]]></isp>
    <org><![CDATA[ORGANIZATION NAME]]></org>
    <as><![CDATA[AS NUMBER / NAME]]></as>
    <query><![CDATA[IP ADDRESS USED FOR QUERY]]></query>
</query>
```

Nous allons récupérer diverses informations sur une IP, grâce à une API. La restriction est bien entendu ici de récupérer et d'afficher les valeurs sans aucun rechargement de page. Voici la structure de la réponse que nous allons recevoir, en XML.

JavaScript – Ajax - Exemple

5.3

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>AJAX-Ip</title>
5          <meta charset="UTF-8">
6          <meta name="viewport" content="width=device-width, initial-scale=1.0">
7          <script type="text/javascript" src="script.js"></script>
8      </head>
9      <body>
10         <input type='button' value='Récupérer les infos' onclick='findIP()' />
11         <div id="result">Résultat</div>
12     </body>
13 </html>
14
```

Au sein de notre page HTML, une simple div qui va recevoir votre résultat, et un bouton appelant la méthode « findIP » de notre fichier JavaScript « script.js ».

JavaScript – Ajax - Exemple

5.3

```
1  function findIP(){
2
3      var xhr;
4      if(window.XMLHttpRequest)//Firefox, Chrome, etc..
5          xhr = new XMLHttpRequest();
6      else if(window.ActiveXObject)//Internet Explorer
7          xhr = new ActiveXObject("Microsoft.XMLHTTP");
8      else
9      { //Non supporté
10         alert("Navigateur non supporté");
11         return;
12     }
13
14     //Documentation : http://ip-api.com/docs/api:xml
15     //Ouverture de la connexion avec le serveur
16     //Type GET/POST, adresse appelée, asynchrone true/false
17     xhr.open("GET", "http://ip-api.com/xml/208.80.152.201", true);
18     xhr.send(null);
19
20     // Si on souhaite envoyer quelque chose (un post de formulaire par exemple)
21     //xhr.open("POST", "user-post.php", true);
22     //xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
23     //xhr.send(data);
24 }
```

JavaScript – Ajax - Exemple

5.3

```
25 //Dès que l'état change -> on execute cette fonction
26 //Autrement dit à chaque retour du serveur
27 xhr.onreadystatechange = function()
28 {
29     //Lorsque la réponse est renvoyée par le serveur l'état est à 4.
30     //(0 : Non initialisé - 1 : Ouverture - 2 : Envoyé - 3 : En cours - 4 : OK)
31     if(xhr.readyState == 4)
32     {
33         //Si Ok : code de status = 200 || Sinon si NOK code de status = 404...
34         if(xhr.status == 200){
35             //Récupération de la réponse XML
36             var doc = xhr.responseXML;
37
38             var resultat = "";
39             resultat += "Status : " + doc.getElementsByTagName("status").item(0).firstChild.data + "<br/>";
40             resultat += "Pays : " + doc.getElementsByTagName("country").item(0).firstChild.data + "<br/>";
41             resultat += "Code Pays : " + doc.getElementsByTagName("countryCode").item(0).firstChild.data + "<br/>";
42             resultat += "Region : " + doc.getElementsByTagName("region").item(0).firstChild.data + "<br/>";
43             resultat += "Nom région : " + doc.getElementsByTagName("regionName").item(0).firstChild.data + "<br/>";
44             resultat += "Ville : " + doc.getElementsByTagName("city").item(0).firstChild.data + "<br/>";
45             resultat += "Code Postal : " + doc.getElementsByTagName("zip").item(0).firstChild.data + "<br/>";
46             resultat += "Latitude : " + doc.getElementsByTagName("lat").item(0).firstChild.data + "<br/>";
47             resultat += "Longitude : " + doc.getElementsByTagName("lon").item(0).firstChild.data + "<br/>";
48             resultat += "Pays : " + doc.getElementsByTagName("country").item(0).firstChild.data + "<br/>";
49             resultat += "Timezone : " + doc.getElementsByTagName("timezone").item(0).firstChild.data + "<br/>";
50             resultat += "Fournisseur : " + doc.getElementsByTagName("isp").item(0).firstChild.data + "<br/>";
51             resultat += "Organisme : " + doc.getElementsByTagName("org").item(0).firstChild.data + "<br/>";
52             resultat += "Alias : " + doc.getElementsByTagName("as").item(0).firstChild.data + "<br/>";
53             resultat += "IP : " + doc.getElementsByTagName("query").item(1).firstChild.data + "<br/>";
54             //Affichage sur la page
55             document.getElementById("result").innerHTML = resultat;
56         }
57         else
58             document.getElementById("result").innerHTML = "Error code " + xhr.status;
59     }
```

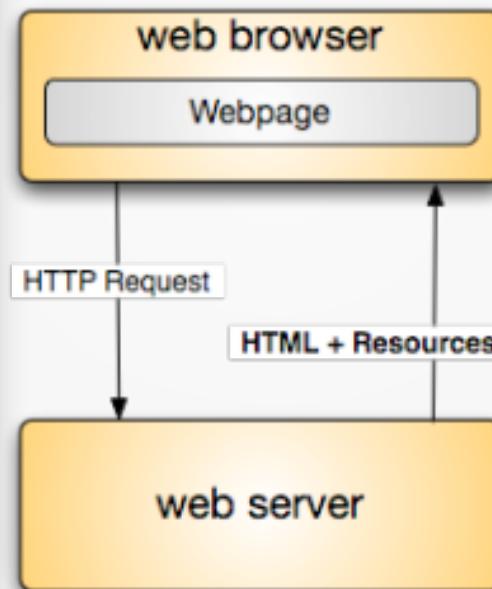
JavaScript – Ajax - Exemple

5.3

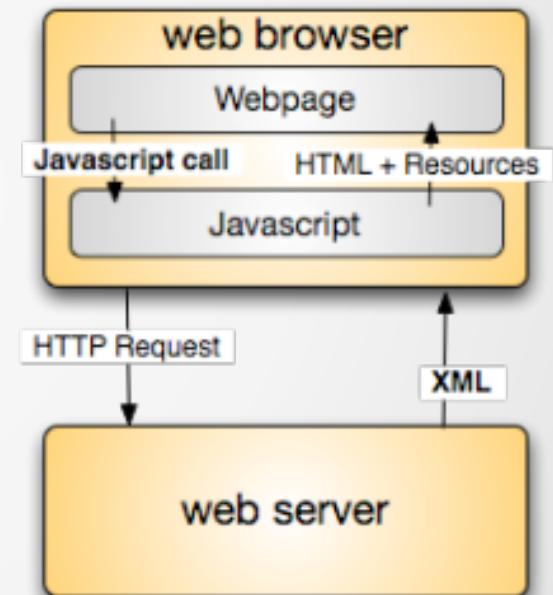
Récupérer les infos

Status : success
Pays : United States
Code Pays : US
Region : CA
Nom région : California
Ville : San Francisco
Code Postal : 94105
Latitude : 37.7898
Longitude : -122.3942
Pays : United States
Timezone : America/Los_Angeles
Fournisseur : Wikimedia Foundation
Organisme : Wikimedia Foundation
Alias : AS14907 Wikimedia Foundation Inc.
IP : 208.80.152.201

Traditional web model



AJAX web model



1.4 – DOM

Le DOM, pour Document Object Model, est une structure de données, fournie par le JavaScript, côté client, qui permet de manipuler les éléments du document HTML. C'est une arborescence correspondant aux imbrications des balises HTML d'un document.

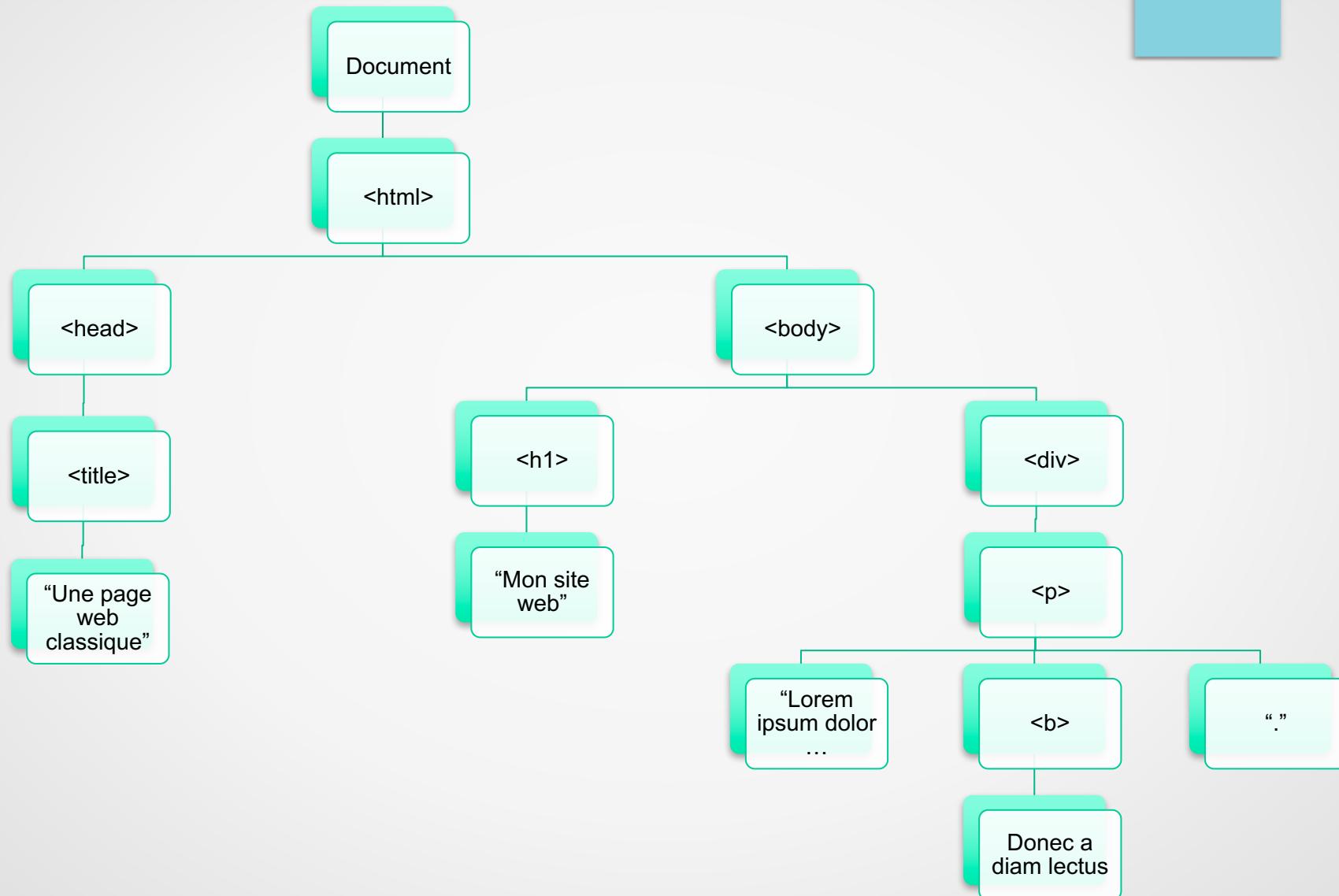
La possibilité de manipuler cette structure sans recharger la page permet de fournir une plus grande interactivité aux utilisateurs, mais aussi de moins solliciter le serveur.

Prenons
une page
web
standard et
regardons le
DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Une page web classique</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>Mon site web</h1>
    <div>
      <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. <b>Donec a diam lectus</b>.</p>
    </div>
  </body>
</html>
```

DOM

5.4



Ainsi, on peut accéder au contenu de la page HTML, ou autre document tel que XML grâce à **l'objet Document**.

Différentes méthodes peuvent lui être appliquées pour récupérer certains nœuds, comme **getElementById** ou **getElementsByTagname**, ou encore écrire dans la page avec la méthode **write**. Parmi les attributs du document que l'on peut consulter, on notera le **doctype** et **documentElement**. S'il s'agit d'un **HTMLDocument**, on aura alors d'autres attributs accessibles, dont certains sont cités en exemples ci-dessous.

Attributs	Description
Title	Balise title du Head
Domain	Nom de domaine du site
URL	URL de la page
body	Balise body du document
images	Liste des balises images
forms	Liste des formulaires de la page

DOM - Node

5.4

On nomme node tout élément quelconque de l'arborescence du document. Ce dernier peut lui-même être contenu dans un autre node qui peut contenir d'autres éléments.. L'interface Node est définie dans le DOM 2.

Interface	Constante	Objet
Document	Node.DOCUMENT_NODE	Document (racine)
Element	Node.ELEMENT_NODE	Balise HTML
Attribut	Node.ATTRIBUTE_NODE	Attribut de l'élément
Text	Node.TEXT_NODE	Texte contenu (balise)
Comment	Node.COMMENT_NODE	Commentaire HTML

Méthode	Description	Méthode	Description
appendChild	Ajoute un noeud	insertBefore	Insère un noeud
cloneNode	Clone le nœud...	removeChild	Supprime un noeud
hasAttribute	Attribut ? (boolean)	replaceChild	Remplace un nœud par un autre
hasChildNodes	Contient nœuds?(b)		

Propriétés

Propriétés	Description
nodeName	Nom de la balise
nodeType	Type de nœud (commentaire, élément...)
nodeValue	Valeur du nœud
attributes	Liste des attributs du nœud

Attributs des nœuds adjoints

Attribut	Description
parentNode	Parent du nœud
childNodes	Liste des nœuds contenus (non récursif)
firstChild	Premier élément du nœud
lastChild	Dernier élément direct du nœud
previousSibling	Nœud précédent (même niveau)
nextSibling	Nœud suivant (même niveau)

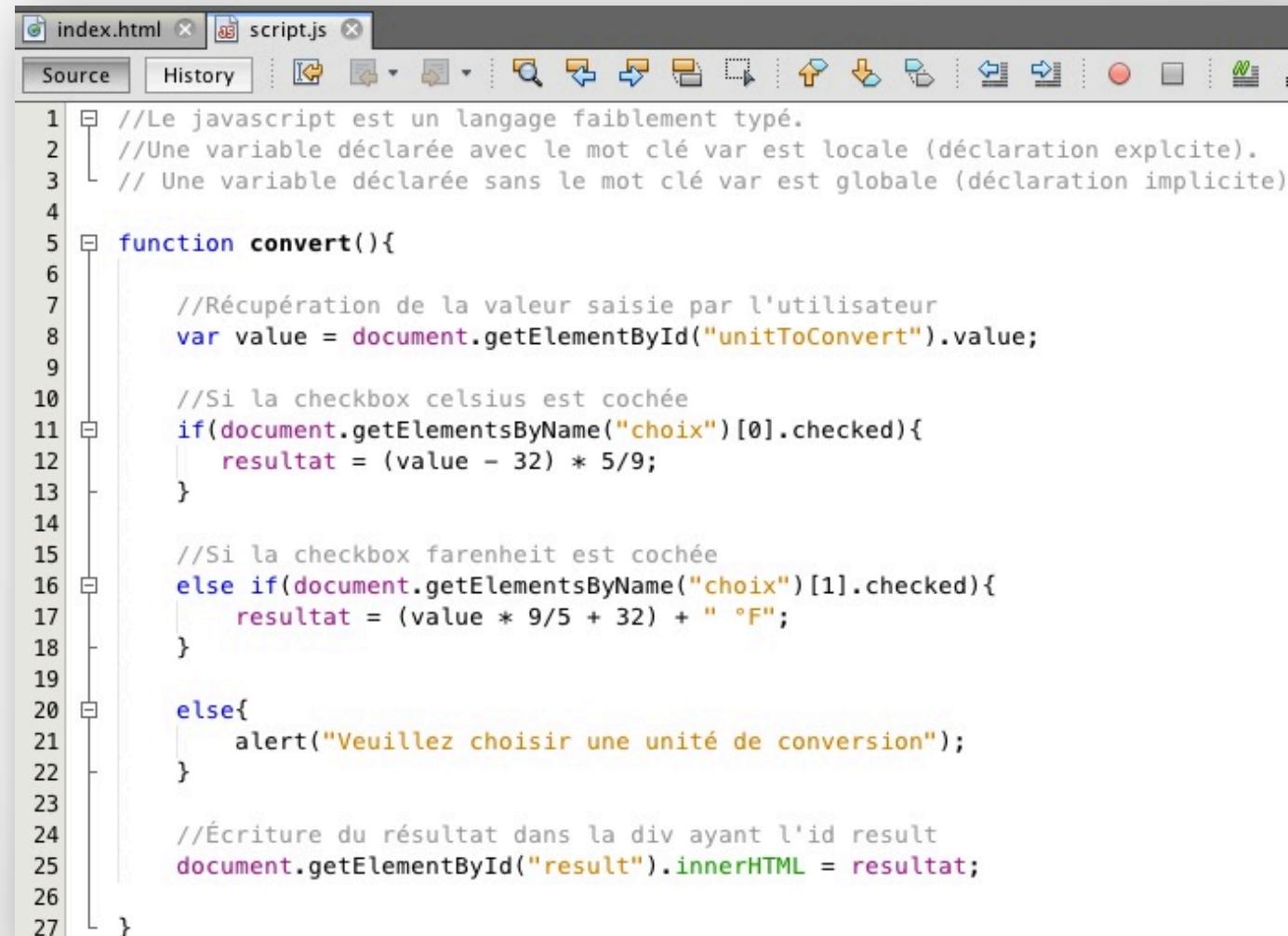
DOM - Exemple

5.4

Nous avons déjà utilisé les possibilités qu'offre le DOM dans les exemples précédents. Attention il s'agit ici de manipuler des éléments unitairement.

Pour parcourir l'arborescence du document, les algorithmes devront être récursifs..

Exemple de manipulation du DOM



The screenshot shows a code editor window with two tabs: "index.html" and "script.js". The "script.js" tab is active and displays the following JavaScript code:

```
1 //Le javascript est un langage faiblement typé.  
2 //Une variable déclarée avec le mot clé var est locale (déclaration explicite).  
3 // Une variable déclarée sans le mot clé var est globale (déclaration implicite)  
4  
5 function convert(){  
6  
    //Récupération de la valeur saisie par l'utilisateur  
    var value = document.getElementById("unitToConvert").value;  
8  
    //Si la checkbox celsius est cochée  
11 if(document.getElementsByName("choix")[0].checked){  
12     résultat = (value - 32) * 5/9;  
13 }  
14  
    //Si la checkbox farenheit est cochée  
16 else if(document.getElementsByName("choix")[1].checked){  
17     résultat = (value * 9/5 + 32) + " °F";  
18 }  
19  
    else{  
21         alert("Veuillez choisir une unité de conversion");  
22     }  
23  
    //Écriture du résultat dans la div ayant l'id result  
25 document.getElementById("result").innerHTML = résultat;  
26 }  
27 }
```

DOM - Exemple

Autre exemple de manipulation du DOM

```

25 //Dès que l'état change -> on execute cette fonction
26 //Autrement dit à chaque retour du serveur
27 xhr.onreadystatechange = function()
28 {
29     //Lorsque la réponse est renvoyée par le serveur l'état est à 4.
30     //(0 : Non initialisé - 1 : Ouverture - 2 : Envoyé - 3 : En cours - 4 : OK)
31     if(xhr.readyState == 4)
32     {
33         //Si Ok : code de status = 200 || Sinon si NOK code de status = 404...
34         if(xhr.status == 200){
35             //Récupération de la réponse XML
36             var doc = xhr.responseXML;
37
38             var resultat = "";
39             resultat += "Status : " + doc.getElementsByTagName("status").item(0).firstChild.data + "<br/>";
40             resultat += "Pays : " + doc.getElementsByTagName("country").item(0).firstChild.data + "<br/>";
41             resultat += "Code Pays : " + doc.getElementsByTagName("countryCode").item(0).firstChild.data + "<br/>";
42             resultat += "Region : " + doc.getElementsByTagName("region").item(0).firstChild.data + "<br/>";
43             resultat += "Nom région : " + doc.getElementsByTagName("regionName").item(0).firstChild.data + "<br/>";
44             resultat += "Ville : " + doc.getElementsByTagName("city").item(0).firstChild.data + "<br/>";
45             resultat += "Code Postal : " + doc.getElementsByTagName("zip").item(0).firstChild.data + "<br/>";
46             resultat += "Latitude : " + doc.getElementsByTagName("lat").item(0).firstChild.data + "<br/>";
47             resultat += "Longitude : " + doc.getElementsByTagName("lon").item(0).firstChild.data + "<br/>";
48             resultat += "Pays : " + doc.getElementsByTagName("country").item(0).firstChild.data + "<br/>";
49             resultat += "Timezone : " + doc.getElementsByTagName("timezone").item(0).firstChild.data + "<br/>";
50             resultat += "Fournisseur : " + doc.getElementsByTagName("isp").item(0).firstChild.data + "<br/>";
51             resultat += "Organisme : " + doc.getElementsByTagName("org").item(0).firstChild.data + "<br/>";
52             resultat += "Alias : " + doc.getElementsByTagName("as").item(0).firstChild.data + "<br/>";
53             resultat += "IP : " + doc.getElementsByTagName("query").item(1).firstChild.data + "<br/>";
54             //Affichage sur la page
55             document.getElementById("result").innerHTML = resultat;
56         }
57     } else
58         document.getElementById("result").innerHTML = "Error code " + xhr.status;
59 }
```

Il existe une interface qui permet de manipuler le conteneur d'un document HTML, autrement dit la fenêtre. Cependant celle-ci n'est pas standardisée par le W3C. On retrouve souvent l'utilisation de cette interface pour afficher des messages popup par exemple.

```
window.alert("Hello world");
```

Outils, références

- **W3C**: <http://www.w3.org/standards/>
- **Alsacréations** : <http://www.alsacreations.com/>
- **DOM** : https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

Plan – Chapitre 2

- Client riche – Framework
 - 2. Jquery
 - 3. AngularJS - Introduction

Chapitre N°2

Programmation web – Client riche

2.1 – JQuery



jQuery est un framework JavaScript multiplateforme, permettant de simplifier, voire d'automatiser de nombreuses tâches complexes en JavaScript. Cela allant de la manipulation du DOM, aux effets de styles, en passant par la mise en œuvre de l'ajax simplement.

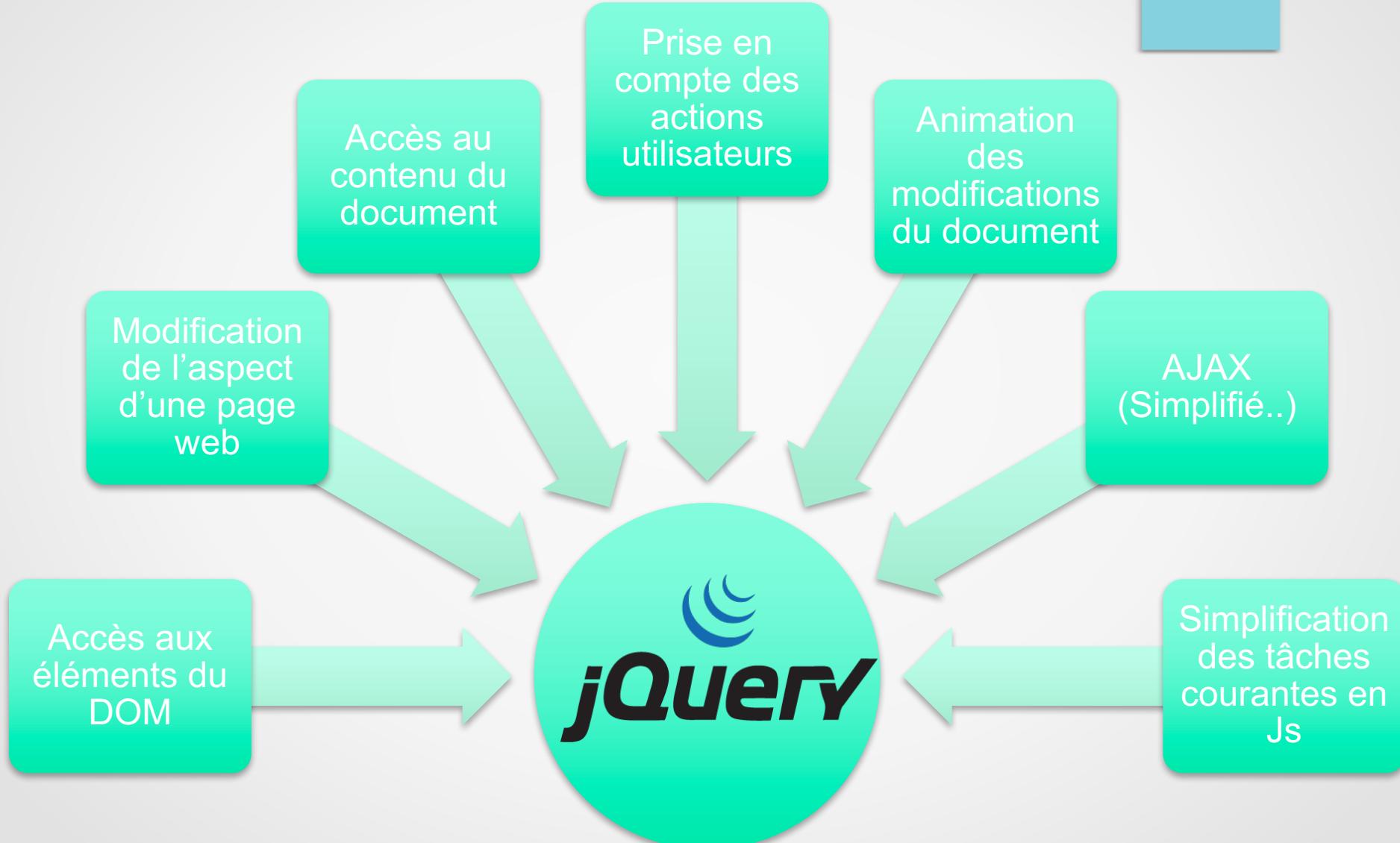
La première ébauche a vu le jour en 2005, donnant le jour à la version 1.0 en 2006. Aujourd'hui, le framework est à la version 3.1.1.

Entre temps, certains plug-ins sont également apparus, comme jQueryUI, qui a remplacé le plug-in Interface. Ce dernier intégrant une collection de widgets préfabriqués, mais aussi un ensemble d'outils pour construire des interfaces complexes.



jQuery - Intérêts

4.1.1



jQuery - Intérêts

4.1.1

Pour permettre son utilisation, il suffit d'importer le script du framework, téléchargeable su le site officiel (www.jquery.com) comme n'importe quel script vu précédemment dans le cours. Prenons ensuite un exemple classique..

```
<!DOCTYPE html>
<html>
  <head>
    <title>JQuery</title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="jquery-2.1.3.min.js"></script>
    <script type="text/javascript" src="script.js"></script>
    <link rel="stylesheet" media="screen" href="style.css" type="text/css">
  </head>

  <body>

    <div id="content">Lorem ipsum dolor sit amet, consectetur adipiscing
      elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas
      congue ligula ac quam viverra nec consectetur ante hendrerit.</div>

  </body>
</html>
```

```
//Lorsque la page est totalement chargée on execute la fonction..
$(document).ready(function(){
  //Ajout d'un classe à l'élément ayant comme id 'content'
  $('#content').addClass('souligne');
});
```

```
body{
  font-size: 1.5em;
}

.souligne{
  background-color: green;
}
```

jQuery nous permet de faciliter l'accès aux éléments du DOM.

Chaque type de sélecteur, balise, id ou classe, peut ainsi être appelé. **Pour éviter tout conflit avec d'autres bibliothèques JavaScript, le symbole \$ est utilisé en début de chaque instruction, c'est un alias de 'jquery'.**

En réalité, de nombreux sélecteurs sont pris en charge au sein de jQuery :

Sélecteur CSS

Sélecteur d'attributs

`$('img[alt]')`

`$'a[href^=mailto:]'`

Sélecteur personnalisé

Sélecteurs de formulaires

`:text :checkbox :radio :submit :input..`

jQuery - DOM

4.1.1

Nous avons vu précédemment comment accéder aux éléments du DOM directement. jQuery nous permet aussi de parcourir le DOM efficacement et d'accéder à leurs valeurs si nécessaires, rien n'empêche d'utiliser plusieurs à la suite... En voici une liste non exhaustive :

Méthode	Objet retourné
.children([selecteur])	Nœuds enfant du sélecteur indiqué
.next([selecteur])	Frère immédiat des éléments sélectionnés
.nextAll([selecteur])	Tous les frères immédiats des él. sélect.
.siblings([selecteur])	Tous les frères du sélecteur indiqué
.parents([selecteur])	Tous les parents du sélecteur indiqué
.filter(selecteur)	Éléments sélectionnés par sélecteur
.not(selecteur)	Éléments qui ne correspondent pas au sel.

jQuery - Évènements

4.1.1

De même, plusieurs méthodes existent pour permettre de prendre en compte les divers évènements qui peuvent survenir (clic, focus, chargement de la page...).

Nous avons d'ailleurs vu dans l'exemple précédent l'événement de fin de chargement de la page `$(document).ready(function..`

Méthode	Description
<code>.ready(gestionnaire)</code>	Gestionnaire invoqué lorsque le DOM & CSS sont totalement chargés
<code>.click(gestionnaire)</code>	Gestionnaire invoqué lorsqu'il y a un clic sur l'élément
<code>.focus(gestionnaire)</code>	Gestionnaire sera invoqué dès lors que l'élément recevra le focus du clavier
<code>.blur(gestionnaire)</code>	Gestionnaire invoqué dès lors que l'élément perdra le focus clavier

Tout comme en JavaScript natif il est possible d'une manière plus simplifiée, particulièrement pour la compatibilité des navigateurs, de faire des appels AJAX avec jQuery. Reprenons notre exemple du cours précédent...

```
<!DOCTYPE html>
<html>
  <head>
    <title>JQuery</title>
    <meta charset="UTF-8">
    <script type="text/javascript" src="jquery-2.1.3.min.js"></script>
    <script type="text/javascript" src="script.js"></script>
    <link rel="stylesheet" media="screen" href="style.css" type="text/css">
  </head>

  <body>
    <div id="content">Lorem ipsum dolor sit amet, consectetur adipiscing
      elit. Donec a diam lectus. Sed sit amet ipsum mauris. Maecenas
      congue ligula ac quam viverra nec consectetur ante hendrerit.</div>
  </body>
</html>
```

jQuery - AJAX

4.1.1

```
function findIP(){

    var resultat = "";

    $.ajax({
        url: "http://ip-api.com/xml/208.80.152.201",
        dataType: "xml",
        success: function(xml){
            //Parcours des enfants
            $(xml).find('query').children().each(function(){
                //Enregistrement du contenu
                resultat += $(this).text() + "<br>";
            })
            $('#result').append(" : <br/> "+resultat);
        }
    });
}
```

Récupérer les infos
Résultat :
success
United States
US
CA
California
San Francisco
94105
37.7898
-122.3942
America/Los_Angeles
Wikimedia Foundation
Wikimedia Foundation
AS14907 Wikimedia Foundation Inc.
208.80.152.201

API Documentation : <https://api.jquery.com/>

2.2 – Angular JS Introduction



Angular JS - Introduction

4.1.1

Angular JS est un framework JavaScript libre, open source développé par Google.



Ses principes sont :

- Découpler les manipulations du DOM et la logique métier
- Découpler le côté serveur et client d'une application.
- Considérer que le test d'une application est aussi important que l'écriture de l'application elle-même.
- Favoriser le couplage faible entre la présentation, la logique métier et les données (MVC/MVVM)

Regardons comment l'utiliser pour construire notre application précédente de TodoList.

Tutoriel AngularJS : Découverte du Framework, création d'une Todo

<https://www.youtube.com/watch?v=E1NIJjTYq6U>

Outils, références

- **W3C:** <http://www.w3.org/standards/>
- **Alsacreations :** <http://www.alsacreations.com/>
- **jQuery:** <https://jquery.com/>
- **API jQuery :** <https://api.jquery.com>
- **Angular JS:** <https://angularjs.org/>

Bibliographie

Une liste, non exhaustive, d'ouvrage vous est présentée ici. Il est fortement conseillé de se tenir à jour des évolutions des divers langages présentés dans ce cours. Soit par l'intermédiaire d'ouvrages comme ci-après et/ou grâce à la documentation officielle en ligne, mentionnée sur la diapositive (outils & références).

- **HTML5 et CSS3 – Maîtrisez les standards des applications web.** Luc Van Lancker– Ed. Eni. ISBN-13 : 978-2746079700
- **jQuery – Simplifiez et enrichissez vos développements JavaScript.** J. Chaffer et K. Swedberg– Ed. Pearson. ISBN : 978-2-7440-2381-1