Javascript - JQuery

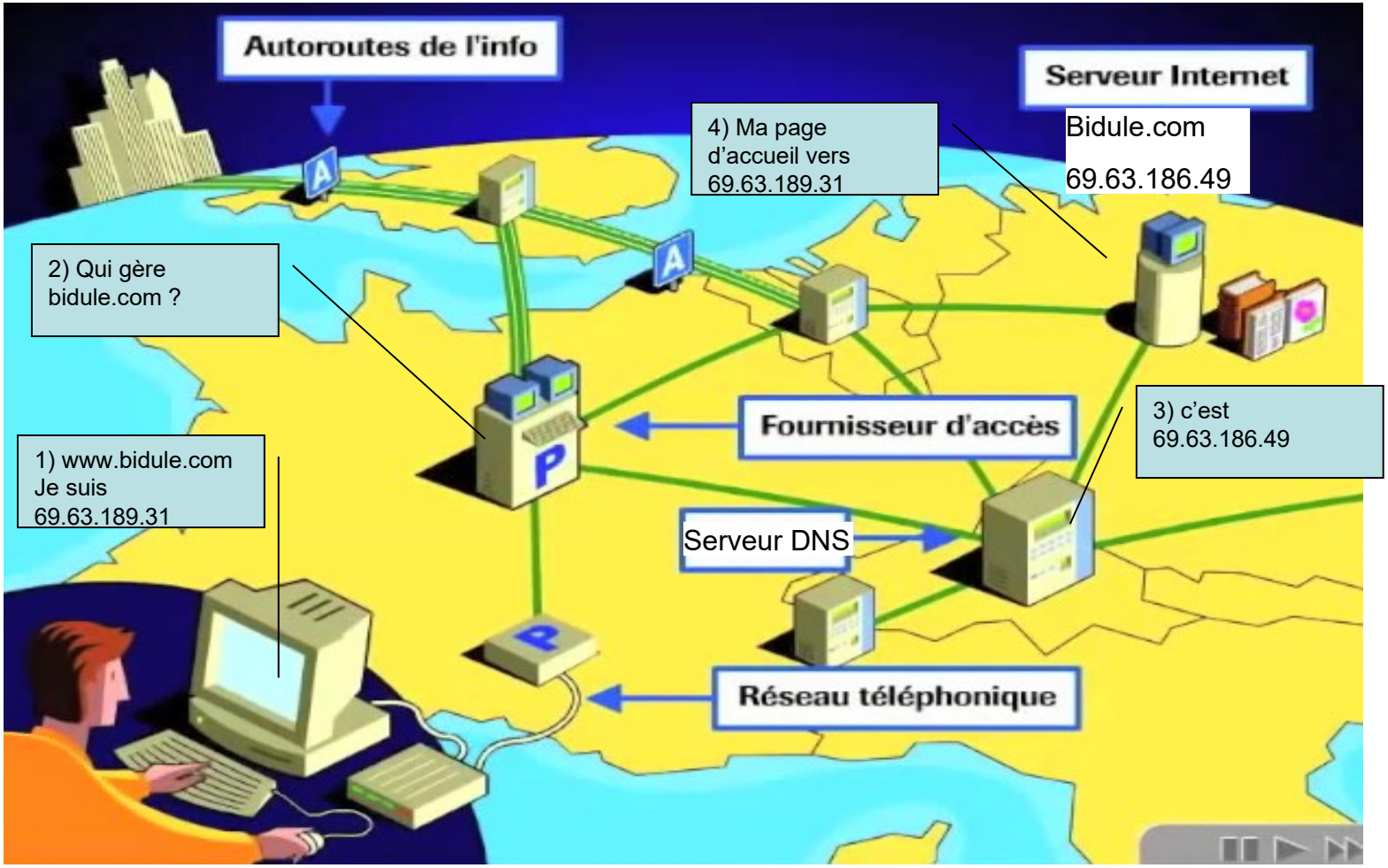
Site Web dynamique

- Introduction à cette formation
  - Votre formateur … Et Vous

  - Le matériel et logiciels
    - L'éditeur de texte Visual Studio code.
    - Navigateurs Chrome et Firefox
  - L'organisation – horaires
    - Formation de 3 jours
  - La forme :
    - Un mélange de concepts avec application directe par un exemple simple
    - Des exercices

- Les liens utiles
  - https://www.w3schools.com/jsref/default.asp
  - https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/introduction-au-javascript
  - https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference
    site de référence Javascript
  - https://openclassrooms.com/fr/courses/3504441-introduction-a-jquery

- Généralités sur le web
- Notre environnement de travail
- Javascript
- JQuery


Autoroutes de l'info
Serveur Internet
Bidule.com
69.63.186.49
4) Ma page d'accueil vers 69.63.189.31
2) Qui gère bidule.com ?
Fournisseur d'accès
3) c'est 69.63.186.49
1) www.bidule.com Je suis 69.63.189.31
Serveur DNS
Réseau téléphonique

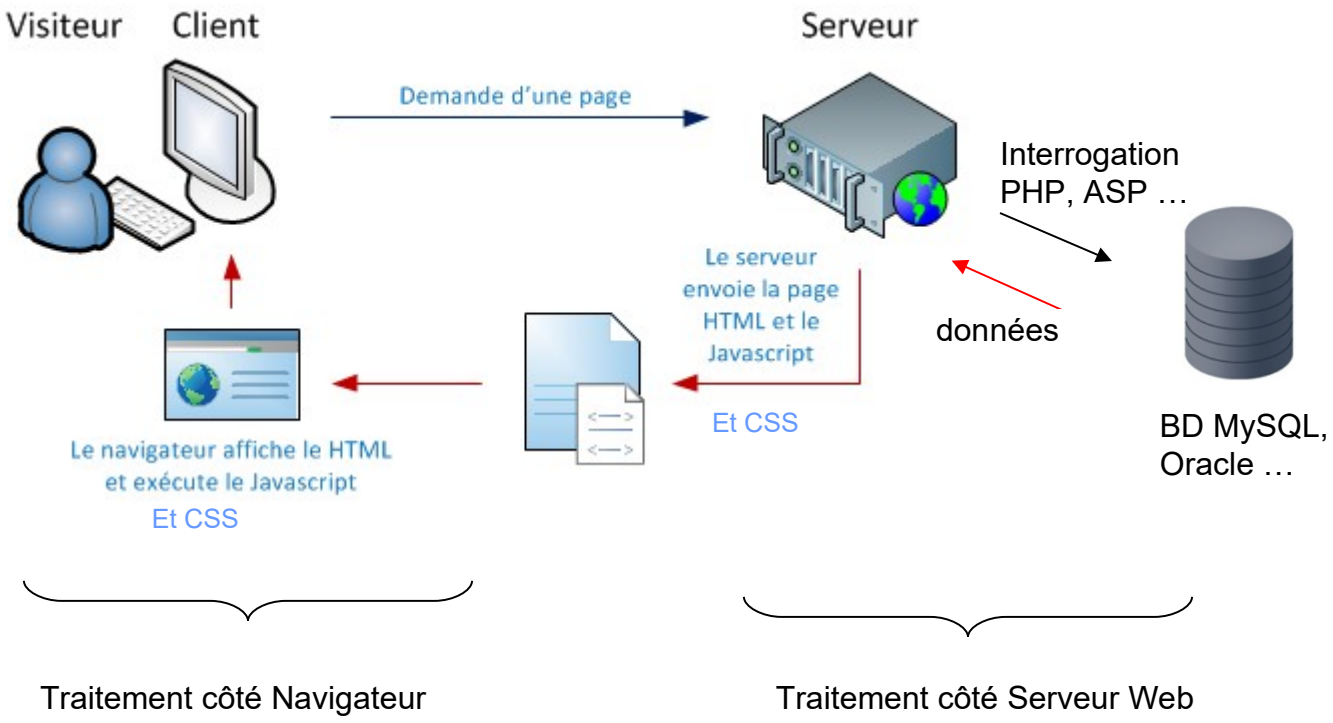
Javascript - JQuery

- La communication client/serveur (navigateur/serveur du site Web) se fait en TCP/IP
- Protocoles utilisés
  - http, HyperText Transfert Protocol
  - https, crypté
  - ftp pour télécharger de gros fichiers
  - Imap, smtp pour les e-mails
  - …
- Pour l'affichage d'une page web, deux acteurs principaux
  - Le serveur Web qui fournit les données
  - Le navigateur qui sait les interpréter et afficher




Javascript - JQuery

- Traitements serveur et client



Sommaire

- Pourquoi faire ?

- Le contexte

- Les bases

- Le DOM

- Les événements

- Les formulaires

- Les objets

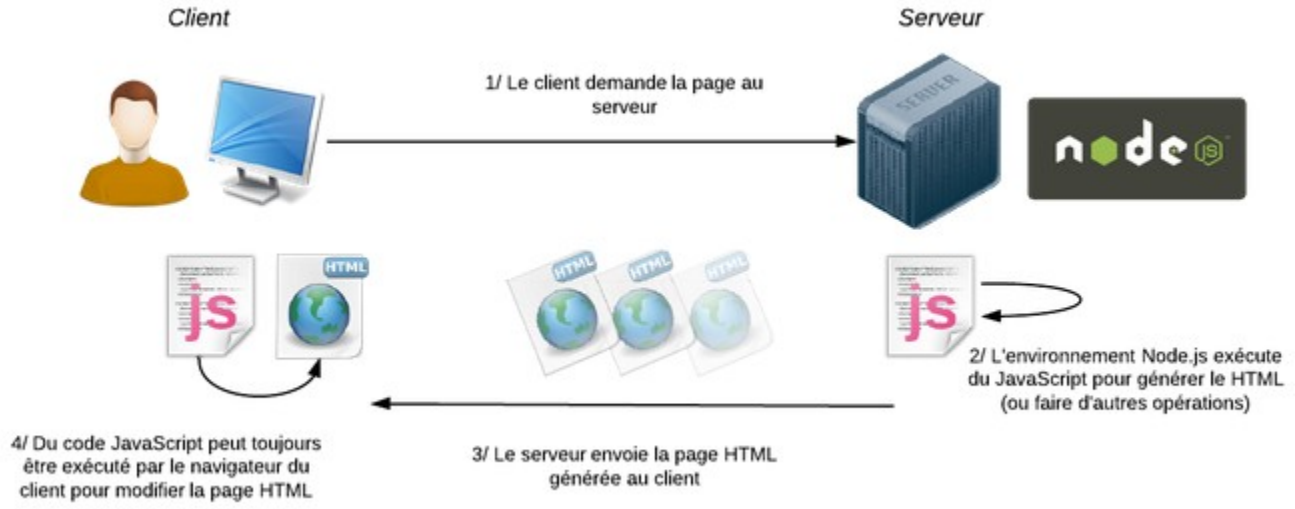
- AJAX

- Il s'insère dans le code (x)HTML d'une page web, et permet d'en augmenter le spectre des possibilités.
  Ce langage de POO *[Programmation Orientée Objet]*, faiblement typé, est exécuté côté client.

  On l'utilise pour :

  - Afficher/masquer du texte ;
  - Contrôler des champs de formulaire;
  - Faire défiler des images ;
  - Créer un diaporama avec un aperçu « en grand » des images ;
  - Créer des infobulles.
  - …….

- Un nouvel engouement début 2010, grâce aux développements pour Chrome, l'a replacé en bonne position côté serveur : utilisation de NodeJS



- Utilisé pour des extensions de navigateur : Chrome, Firefox

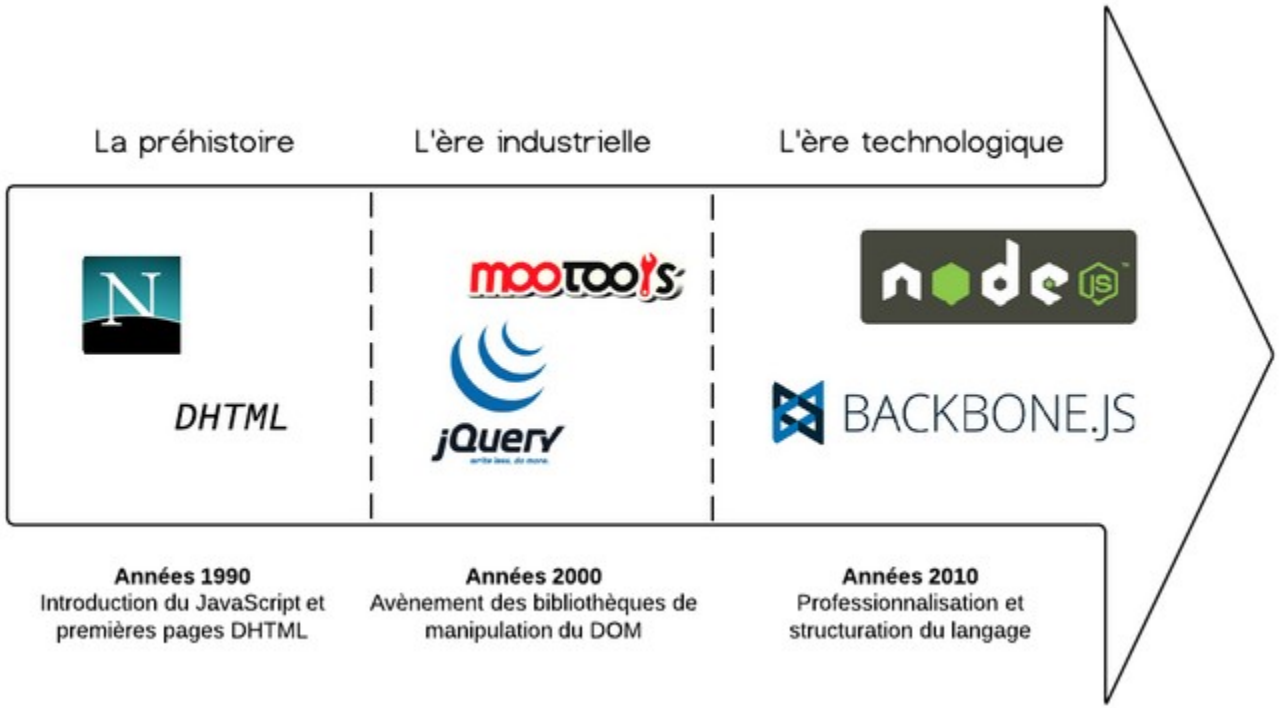
- Utilisé par d'autres applications : ex Brackets

- JavaScript développé par Netscape en 1995 sous le nom de LiveScript

- Normé ensuite par L'ECMA (European Computer Manufactures Association) pour fournir le langage ECMAScript

- JavaScript dérive de ECMAScript avec des extensions
- ActionScript (Adobe Flash) et JScript (Microsoft) implémentent aussi ECMAScript
- Javascript : Rien à voir avec JAVA
- Langage basé sur les Objets (pas de classe)
- Interprété :
  - Par tout navigateur internet. Les interpréteurs sont différents :
    - Chakra Microsoft à partir de IE9
    - JScript avant IE9
    - SpiderMonkey pour Firefox
    - V8 pour Chrome : très optimisé, open source, contient un JIT
  - Par la plateforme NodeJS : interpréteur V8

•


La préhistoire
L'ère industrielle
L'ère technologique
DHTML
mootools
jQuery
node
BACKBONE.JS
Années 1990
Introduction du JavaScript et premières pages DHTML
Années 2000
Avènement des bibliothèques de manipulation du DOM
Années 2010
Professionnalisation et structuration du langage

exemple :


```
<html>
 <head>
  <title>Page statique</title>
 </head>
 <body>
  <div>
   Nous sommes le 2/10/2008
  </div>
 </body>
</html>
```

Essayons ce code

avec Brackets

```
<html>
 <head>
  <title>Page dynamique</title>
 </head>
 <body>
  <script type = "text/javascript">

   date = new Date();
   document.writeln("Nous sommes le ", date);

  </script>
 </body>
</html>
```

Javascript Où mettre le code ?

- Déclaration de code JS :

Interne :

```
<script type="text/javascript">

    code javascript

</script>
```

Externe, dans un fichier .js local ou depuis une url

```
<script type="text/javaScript" src = "js/script.js"></script>
<script src="https://www.w3schools.com/js/myScript.js"></script>
```

- A partir de HTML5 `type="text/javaScript` n'est pas nécessaire pour JS
- Déclaration dans l'élément <head> si peu volumineux
- Déclaration avant le </body> si volumineux, pour ne pas ralentir le 1er affichage de la page

- Autres déclarations possibles :
  - Dans une URL

```
<a href="javascript:instructionJavaScript;">Texte</a>
```

  - Dans un événement

```
<balise onEvenement="instructionJavaScript">...</balise>
```

```
<!DOCTYPE HTML >
<html>
<head>
 <title>Exemple de page HTML contenant du JavaScript</title>
  <script >

  function texte() { document.body.appendChild(document.createTextNode("Texte généré")); }

  </script>
 </head>
 <body>
  <script>

   document.write("Vous pouvez mettre du code javascript dans le corps du document.");

  </script>
  <p>
   Ou bien dans une fonction appelée en cliquant
   <a href="Javascript:texte()">ici</a>,
  <p>
   ou en passant la souris au-dessus de
   <a href="" onMouseOver="texte()">cela</a>...
   </body>
</html>
```

- Les variables
  - Une variable est un boîte qui va conserver et fournir une valeur
  - Une variable a un nom qui ne commence pas par un chiffre, qui est sensible à la casse (minuscule/majuscule)

- 3 types de base :

  - Number entier : 127 (base 10), 0755 (base 8), 0xFA15 (base 16)
  - Number flottant : 0.123, -0.4e5, .67E-89
  - booleen : true, false
  - String, chaîne de caractères : "chaine" ou 'chaine'

- Déclaration des variables : on ne déclare pas le type, seulement la « portée » :
  - let age = 18 ; // la variable est visible dans le bloc de code délimité par {}
  - var age = 18 ; // age a la portée de la page, portée globale
  - age = 18 ; // var implicite

- Connaître le type d’une variable : typeof()

Exercice section 3.1

- Il est possible d'effectuer des calculs à partir du contenu des variables. Le type de calcul dépend du type des variables et des valeurs utilisées.

- Opérateurs Arithmétiques
  https://www.w3schools.com/js/js_operators.asp

- Opérateur + pour les chaînes de caractères

  resultat = " L'application" + " " + 'qui fonctionne' ;

- ` et ${ } pour afficher le contenu d'une variable

  age = 20 ;
  resultat = `Cette personne a ${age} ans` ; // Alt GR touche 7 pour apostrophe inversée

- Opérateur Logique
  result = temp == 37 ;
  h2o = (temp<100) ? "eau" : "vapeur";
  h2o = (temp>0) ? ((temp<100) ? "eau" : "vapeur") : "glace";

- Opérateurs
  - Affectation
    +=, -=, *=, /=, %=, &=, |=, <<=, >>=

- Comparaison
  ==, !=, <, <=, >, ≥, ===
  == ne s'occupe pas du type : 7 == '7' rend true
  === s'occupe du type (vu plus loin)

- Arithmétique https://www.w3schools.com/js/js_arithmetic.asp
  %, ++, --

Exercice section 3.2

  - logique
    &&, ||, !

  - Bit
    &, |, ^ (XOR), <<, >>, >>> (décalage à droite avec zéro à gauche)

Structures de contrôle

if else, else if, switch case, for, while, break, continue, do while

https://www.w3schools.com/js/js_if_else.asp

```
if (hour < 18) {
    greeting = "Good day";
}
```

https://www.w3schools.com/js/js_loop_for.asp

```
var i;
for (i = 0; i < 10; i++) {
    if (i === 3) { continue; }
    text += "The number is " + i + "<br>";
}
```

https://www.w3schools.com/js/js_switch.asp

```
switch(expression) {
    case n:
        code block
        break;
    case n:
        code block
        break;
    default:
        code block
}
```

https://www.w3schools.com/js/js_loop_while.asp

```
while (condition) {
    code block to be executed
}
```

- L'opérateur === et !== . Supposons que x =5;

Operator	Description	Comparing	Returns
==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

- Cas singulier du OU || :

```
var conditionTest1 = '', conditionTest2 = 'Une chaîne de caractères';
alert(conditionTest1 || conditionTest2); // va montrer conditionTest2
```

- Réaliser les exercices de la section 4:

- L'objet String

  Lorsqu'on définit une constante ou une variable chaîne de caractères, JavaScript crée d'une façon transparente une instance String
  https://www.w3schools.com/jsref/jsref_obj_string.asp

  – 1 seule propriété : length
  – De nombreuses méthodes , exemple toLowerCase()

  – les propriétés HTML/XHTML ont leur équivalent en méthodes JS : (extrait)
    bold(), italics(), fontcolor(), fontsize(),
    small(), big(), toUpperCase(), toLowerCase(),
    sub(), sup(), substring(), eval(), split(), replace()

- L'objet Math

  https://www.w3schools.com/jsref/jsref_obj_math.asp

```
<script>
document.getElementById("demo").innerHTML = Math.sqrt(64);
</script>
```

- Les tableaux ordinaires

```
var jours = new Array();
var jours = new Array("Lundi", "Mardi", "Mercredi", "Jeudi",
"Vendredi", "Samedi", "Dimanche");
jours[0]
jours.length
```

- Les tableaux associatifs

```
var tableau = new Array();
tableau["un"] = "La premiere chaîne";
tableau["deux"] = "La deuxieme chaîne";
tableau["tnt"] = "Plein d'autres chaînes";
tableau["un"]
tableau.length
```

Les méthodes de Array
push(), pop(), shift(), concat() …..
https://www.w3schools.com/jsref/jsref_obj_array.asp

- Des « simili » tableaux : les objets littéraux :

```
var family = {
    self: 'Sébastien',
    sister: 'Laurence',
    brother: 'Ludovic',
    cousin_1: 'Pauline',
    cousin_2: 'Guillaume'
};      // family est un objet dont sister est une propriété

// divers accès :
family.sister
family['sister']
var id = 'sister';
family[id]

// utilisation de for in
for (var id in family){
        // id contient chaque nom de propriété
        alert(family[id]);
}
```

- Fonctions

```
function maFonction(first, second) {
    // On peut maintenant utiliser les variables « first » et « second » comme on le souhaite :
    alert('Votre premier argument : ' + first);
    alert('Votre deuxième argument : ' + second);
}
```

- `return val;` si une valeur de retour est souhaitée
- Les paramètres sont facultatifs. Exemple :

```
function prompt2(text, allowCancel) {
    if (typeof allowCancel === 'undefined') { // Souvenez-vous de typeof, pour vérifier le type d'une variable
        allowCancel = false;
    }

    // Le code…
}

prompt2('Entrez quelque chose :'); // On exécute la fonction seulement avec le premier argument, pas besoin du deuxième
```

- Les fonctions anonymes : on a besoin d'un traitement immédiat sans avoir besoin d'un nom de fonction
- Une fonction anonyme est donc une fonction … sans nom

```
function (arguments) {
    // Le code de votre fonction anonyme
}
```

- La référence d'une telle fonction peut être mise dans une variable :

```
var sayHello = function() {
    alert('Bonjour !');
};

..
sayHello(); // Affiche : « Bonjour ! »
```

- L'isolement de code
  - ?? : Permet d'éviter qu'une partie de code affecte le reste
  - Explications par étapes :
    - Ce que nous venons de voir :

```
function test() {
    // Du code…
}

test(); // appel de la fonction
```

    - Auto appel de la fonction :

```
(function test() {
    // Du code…
})();                        // la fonction est appelée
```

- L'isolement de code (suite)

  En utilisant l'auto exécution, l'isolement de code s'écrit ainsi :

```
(function () {
    // Code isolé
        …
})();
```

```
var test = 'noir'; // On crée une variable « test » contenant le mot « noir »

(function() { // Début de la zone isolée

 // On crée une variable du même nom avec le contenu « blanc » dans la zone isolée
   var test = 'blanc';

   alert('Dans la zone isolée, la couleur est : ' + test);

})(); // Fin de la zone isolée. Les variables créées dans cette zone sont détruites.

alert('Dans la zone non-isolée, la couleur est : ' + test);
// Le texte final contient bien le mot « noir » vu que la «
zone isolée » n'a aucune influence sur le reste du code
```

- Javascript Closure

  https://www.w3schools.com/js/js_function_closures.asp

```
var add = (function () {
    var counter = 0;   // exécuté une seule fois
    return function () {counter += 1; return counter}
})();

add();
add();
add();

// counter vaut 3
```

Site Openclassroom :

https://openclassrooms.com/fr/courses/5543061-ecrivez-du-javascript-pour-le-web/5543068-comprenez-ce-quest-le-dom

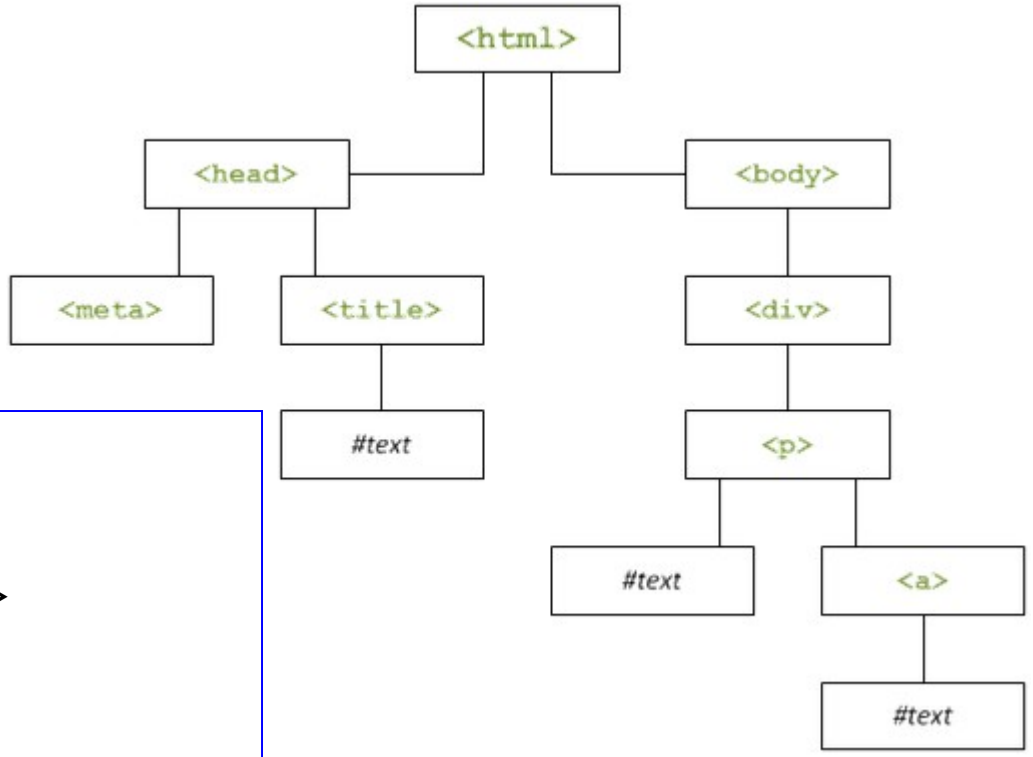
- Le Document Object Model, DOM, est une interface de programmation (API) pour les documents XML et HTML.

- Les balises du cours HTML sont nommées ici des éléments HTML en JS (se sont des objets)

- Toute page HTML gérée par un navigateur présente des données arborescentes. Le nœud le plus haut est « window », objet global qui représente la fenêtre du navigateur.
  Tout code JS est exécuté à partir de window

- document est sous le nœud windows, représente la page Web correspondant à la balise <html>

- Exemple




```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
</head>

<body>
    <div>
        <p>Un peu de texte <a>et un lien</a></p>
    </div>
</body>
</html>
```

- Pour accéder aux éléments du DOM : 5 méthodes de l'objet document :
  - getElementById('idName') : fournir l'unique id

```
<div id="myDiv">
    <p>Un peu de texte <a>et un lien</a></p>
</div>

<script>
    var div = document.getElementById('myDiv');
    …
</script>
```

  - getElementsByTagName() : fournir la balise html

```
var divs = document.getElementsByTagName('div');

for (var i = 0, c = divs.length ; i < c ; i++) {
    alert('Element n° ' + (i + 1) + ' : ' + divs[i]);
}
```

- getElementsByClassName() : fournir le nom de la classe au sens html

- querySelector() et querySelectorAll()

  Rappel : le sélecteur CSS suivant
  #menu .item span
  sélectionne les balises de type contenues dans les classes
  class=« item » elles-mêmes contenu dans un tag identifié par id=« menu »

  - querySelector() retourne le 1er élément trouvé correspondant au sélecteur CSS
  - querySelectorAll() retourne tous les éléments trouvés correspondant au sélecteur CSS

- Exemple

```
<div id="menu">

    <div class="item">
        <span>Élément 1</span>
        <span>Élément 2</span>
    </div>

    <div class="publicite">
        <span>Élément 3</span>
        <span>Élément 4</span>
    </div>

</div>

<script>
    var query = document.querySelector('#menu .item span'),
    queryAll = document.querySelectorAll('#menu .item span');
</script>
```

- getAttribute() et setAttribute()

```
<body>
 <a id="myLink" href="http://www.un_lien_quelconque.com">Un lien modifié dynamiquement</a>

 <script>
     var link = document.getElementById('myLink');
     var href = link.getAttribute('href'); // On récupère l'attribut « href »

     alert(href);

     link.setAttribute('href', 'http://http://www.meteo-paris.com/'); // On édite
l'attribut «href»
     // ecriture equivalente et directe :
     link.href = 'http://http://www.meteo-paris.com/';

 </script>
</body>
```

- Pour accéder à l'attribut class au sens CSS d'une balise HTML, utiliser

  className (puisque class est un mot réservé de JS)

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Le titre de la page</title>
    <style>
        .blue {
            background: blue;
            color: white;
        }
    </style>
</head>

<body>
    <div id="myColoredDiv">
        <p>Un peu de texte <a>et un lien</a></p>
    </div>

    <script>
        document.getElementById('myColoredDiv').className = 'blue';
    </script>
</body>
</html>
```

- Attention, className peut désigner plusieurs classes CSS

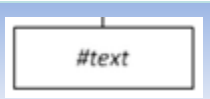
```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Partie II - Chapitre 1 - Exemple 4</title>
 </head>
 <body>
  <a id="myLink" class="external red u" href="http://www.un_lien_quelconque.com"> Un lien
avec plusieurs classes</a>
  <script>
   var classes = document.getElementById('myLink').className;
   var classesNew = [];
   classes = classes.split(' ');
   for (var i = 0, c = classes.length; i < c; i++) {
     if (classes[i]) {
       classesNew.push(classes[i]);
     }
   }
   alert(classesNew);
  </script>
 </body>
</html>
```

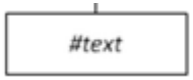
- Exercice exoJSClasse.html

- Autre méthode plus récente (version >= IE10)

```
var div = document.querySelector('div');

// Ajoute une nouvelle classe
div.classList.add('new-class');

// Retire une classe
div.classList.remove('new-class');

// Retire une classe si elle est présente ou bien l'ajoute si elle est absente
div.classList.toggle('toggled-class');

// Indique si une classe est présente ou non
if (div.classList.contains('old-class')) {
    alert('La classe .old-class est présente !');
}

// Parcourt et affiche les classes CSS
var result = '';

for (var i = 0; i < div.classList.length; i++) {
    result += '.' + div.classList[i] + '\n';
}
alert(result);
```

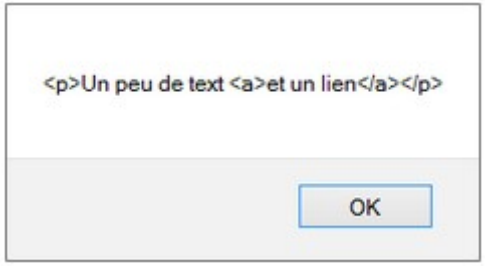
Le DOM : texte



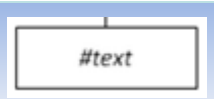
- Pour agir sur 

de l'arborescence DOM :
- propriété innerHTML : le texte et balises HTML, y compris les éléments enfants

```
<body>
    <div id="myDiv">
        <p>Un peu de texte <a>et un lien</a></p>
    </div>

    <script>
        var div = document.getElementById('myDiv');

        alert(div.innerHTML);
    </script>
</body>
```



Le DOM : texte



- Propriété textContent et innerText : le texte sans les balises

  - textContent retourne le texte de tous les éléments, y compris le texte des éléments enfants (> IE 8), avec tous les espaces

  - innerText idem textContent excepté :
    - Ne retourne pas le contenu de <script> et <style>
    - Ne retourne pas le contenu d'éléments cachés par une règle CSS

Exemple : https://www.w3schools.com/Jsref/prop_node_innertext.asp

- À partir d'un élément du DOM, la navigation est possible dans l'arbre :
  - parentNode, firstChild, lastChild, firstElementChild, lastElementChild …
  - nodeType et nodeName permettent de vérifier le nœud

- Pour créer un élément DOM : createElement()

```
var newLink = document.createElement('a');
```

- Affecter ensuite les attributs

```
newLink.id    = 'sdz_link';
newLink.href  = 'http://www.siteduzero.com';
newLink.title = 'Découvrez le Site du Zéro !';
newLink.setAttribute('tabindex', '10'); // autre façon
```

- Insérer l'élément créé dans l'arbre

```
document.getElementById('myP').appendChild(newLink);
```

- Exercice re-créer une structure de document

- Exercice construction d'une liste

- Un événement JS permet de déclencher une fonction JS suite à une action dans la page

`click`	Cliquer (appuyer puis relâcher) sur l'élément
`dblclick`	Double-cliquer sur l'élément
`mouseover`	Faire entrer le curseur sur l'élément
`mouseout`	Faire sortir le curseur de l'élément
`mousedown`	Appuyer (sans relâcher) sur le bouton gauche de la souris sur l'élément
`mouseup`	Relâcher le bouton gauche de la souris sur l'élément
`mousemove`	Faire déplacer le curseur sur l'élément

`keydown`	Appuyer (sans relâcher) sur une touche de clavier sur l'élément
`keyup`	Relâcher une touche de clavier sur l'élément
`keypress`	Frapper (appuyer puis relâcher) une touche de clavier sur l'élément
`focus`	« Cibler » l'élément
`blur`	Annuler le « ciblage » de l'élément
`change`	Changer la valeur d'un élément spécifique aux formulaires (`input` , `checkbox` , etc.)
`input`	Taper un caractère dans un champ de texte (son support n'est pas complet sur tous les navigateurs)
`select`	Sélectionner le contenu d'un champ de texte (`input` , `textarea` , etc.)

- Evénements spécifiques aux formulaires, balise HTML <form>

Nom de l'événement	Action pour le déclencher
`submit`	Envoyer le formulaire
`reset`	Réinitialiser le formulaire

- Codage des événements
  - En direct (notez le this dans l'exemple)

```
<span onclick="alert('Voici le contenu de l\'élément que vous avez
cliqué :\n\n' + this.innerHTML);">Cliquez-moi !</span>
```

  - En direct (notez le return false)

```
<a href="#" onclick="alert('Vous avez cliqué !'); return false;" >
Cliquez-moi !</a>
```

  - En utilisant le DOM

```
<span id="clickme">Cliquez-moi !</span>

<script>

    var element = document.getElementById('clickme');

    element.onclick = function() { // fonction anonyme
        alert("Vous m'avez cliqué !");
    };

</script>
```

- Codage des événements – suite
  - En utilisant DOM-2

```
<span id="clickme">Cliquez-moi !</span>

<script>
    var element = document.getElementById('clickme');

    element.addEventListener('click', function() {
        alert("Vous m'avez cliqué !");
    });
</script>
```

- L'objet Event contient des informations utiles sur l'événement

```
<div id="position"></div>

<script>
    var position = document.getElementById('position');

    document.addEventListener('mousemove', function(e) {
        position.innerHTML = 'Position X : ' + e.clientX + 'px<br
/>Position Y : ' + e.clientY + 'px';
    });
</script>
```

- Exercice Position souris

- JS permet de gérer les propriétés spécifiques aux contrôles de formulaire : value, disabled, checked …
  Exemple : liste déroulante

```
<select id="list">
    <option>Sélectionnez le jour</option>
    <option>Lundi</option>
    <option>Mardi</option>
</select>

<script>
    var list = document.getElementById('list');

    list.addEventListener('change', function() {

        // On affiche le contenu de l'élément <option> ciblé par la
          propriété selectedIndex
        alert(list.options[list.selectedIndex].innerHTML);

    });
</script>
```

- Les boutons submit et reset

```
var element = document.getElementById('un_id_de_formulaire');

element.submit(); // Le formulaire est expédié
element.reset();  // Le formulaire est réinitialisé
```

- Pas de définition de Classe, directement définition d'un Objet via son constructeur
- L'opérateur new crée une copie de l'Objet de base, non utilisable

```
function Chien(nom, race, maitre) {   // constructeur
  this.nom = nom;                     // des propriétés
  this.race = race;
  this.maitre = maitre;
  this.print = printChien;            // une méthode
}

function printChien() {
  document.write("Chien ", this.nom, " de race ", this.race,
  " appartenant &agrave; ", this.maitre, "<br/>");
}
```

```
rantanplan = new Chien("Rantanplan", "indéfinie", "Lucky Luke");
rantanplan.print();
```

- Autre exemple

```
function Person(nick, age, sex, parent, work, friends) {
    this.nick = nick;
    this.age = age;
    this.sex = sex;
    this.parent = parent;
    this.work = work;
    this.friends = friends;

    this.addFriend = function(nick, age, sex, parent, work, friends) {
        this.friends.push(new Person(nick, age, sex, parent, work,
friends));
    };
}
```

- La fonction call() : permet d'exécuter une méthode d'un objet à un autre objet
  https://www.w3schools.com/js/js_function_call.asp

```
var person = {
    fullName: function() {
        return this.firstName + " " + this.lastName;
    }
}
var person1 = {
    firstName:"John",
    lastName: "Doe",
}
var person2 = {
    firstName:"Mary",
    lastName: "Doe",
}
person.fullName.call(person1);  // Will return "John Doe"
```

- La fonction apply() est identique, le passage de paramètres éventuels se fait dans un array

- Une autre déclaration de classe est apparue dans JS avec ECMAScript 2015.
  Non abordée dans ce cours

```
1  class Rectangle {
2    constructor(hauteur, largeur) {
3      this.hauteur = hauteur;
4      this.largeur = largeur;
5    }
6
7    get area() {
8      return this.calcArea();
9    }
10
11   calcArea() {
12     return this.largeur * this.hauteur;
13   }
14 }
15
16 const carré = new Rectangle(10, 10);
17
18 console.log(carré.area);
```

- Le mot clé prototype permet d'ajouter une propriété en dehors de la définition de classe

```
<script>

function employee(name, jobtitle, born)
  {
  this.name = name;
  this.jobtitle = jobtitle;
  this.born = born;
  }

var fred = new employee("Fred Flintstone", "Caveman", 1970);
employee.prototype.salary = null;
fred.salary = 20000;

document.write(fred.salary);

</script>
```

- Le mot clé prototype permet d'ajouter une méthode en dehors de la définition de classe

```
function Person(first, last, age, eyecolor) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eyecolor;
}
Person.prototype.name = function() {
    return this.firstName + " " + this.lastName;
};
```

- AJAX : Asynchronous JavaScript And XML
  Le transfert de données est géré *exclusivement* par le JavaScript, et utilise certaines technologies de formatage de données, comme le XML ou le JSON
- Permet de mettre à jour partiellement une page sans la ré afficher.
- Les formats possibles :
  - Classiques : texte ou texte avec tags HTML
  - Le XML, des méthodes du DOM peuvent être employées
  - LE JSON. L'objet JS JSON possède les méthodes parse() et stringify()

```
var obj = {
        index: 'contenu'
    },    chaine;

chaine = JSON.stringify(obj);
alert(typeof chaine + ' : ' + chaine ); // Affiche : « string :
{"index":"contenu"} »

obj = JSON.parse(chaine);
alert(typeof obj + ' : ' + obj); // Affiche : « object : [object Object] »
```

- Exercice : Un formulaire interactif

JQuery

- Liens :
  - https://openclassrooms.com/courses/simplifiez-vos-developpements-javascript-avec-jquery
  - http://learn.jquery.com/
  - https://api.jquery.com/

Sommaire

JQuery Introduction

- JQuery est une bibliothèque Javascript pour simplifier le dev web
- Téléchargeable sur http://jquery.com/download
  - Version uncompressed jquery*.js pour le développement
  - Version compresses jquery*.min.js pour le déploiement
- Multiplateforme : évite les problèmes JS de compatibilité entre navigateurs
- Gratuite et open source
- Facilite la sélection d'éléments d'une page web
- Facilite l'AJAX

- JQuery pour sélectionner des éléments/nœuds du DOM et agir dessus
- Objet JQuery : ensemble de nœuds du DOM
- Objet JQuery : jQuery() abrégé en $()
- $('selecteur') → objet jQuery
- Exemple :
  - **$(**"div"**)** renvoie un objet contenant tous les "div" du document
    en JS : getElementsByTagName()
  - **$(**"div"**)**.hide() cache tous les "div" du document.

- Insérer le lien vers la bibliothèque jQuery pour la charger (vers </body>)
  pour l'exploitation :

```
<SCRIPT
SRC="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</SCRIPT>
```

  pour le développement :

```
<SCRIPT SRC="jquery.js"> </SCRIPT>
```

- Attendre le chargement de la page :

```
<SCRIPT TYPE="text/javascript">
$(document).ready(function(){
// vos instructions Javascript/jQuery ici
})
</SCRIPT>
```

- **$(document).ready(**function(){…}**)** s'abrege en **$(**function(){…}**)**

JQuery Sélection

- Possibilité de sélectionner :

1. par type de bloc
2. par identifiant
3. par classe
4. en combinant les critères
5. en filtrant sur les noms d'attributs
6. en faisant référence aux positions relatives dans le DOM
7. en ne récupérant qu'un seul élément parmi les objets sélectionnés
8. en filtrant parmi les objets sélectionnés

Les sélecteurs sont ceux utilisés pour la sélection CSS

Utiliser le lien https://www.w3schools.com/jquery/trysel.asp

Regarder le contenu de ResumeJQ.pdf

JQuery Modifier le contenu HTML

- $(‘selecteur’).action(‘[contenu]’) avec action :
  - html remplacement du contenu d’un élément (les balises sont considérées comme des balises)
  - text remplacement du contenu d’un élément en considérant le tout comme du texte
  - after/before insertion du contenu après/avant l’élément sélectionné
  - append insertion du contenu dans l’élément sélectionné à la suite des éléments existants
  - prepend
  - wrap insertion des balises passées en paramètre **de part et d’autre** de l’élément
  - wrapInner, unwrap

- Exemple :
  - $("div.a").html($("div.c").html()); → met le contenu du div.c dans le div.a

Modifier les attributs d'une balise

- $('selecteur').attr('attrib')
  - permet de récupérer la valeur de l'attribut attrib du premier élément sélectionné par le sélecteur, récupère **undefined** si l'attribut attrib n'est pas défini pour cet élément.

- $('selecteur').attr('attrib', 'val')
  - Permet d'affecter la valeur val à attrib du premier élément sélectionné

- Exemple
  - Code HTML
    ```
    <img src="logoUPEM.png" alt="logo.png">
    ```
  - Code jQuery
    ```
    // alerte qui affiche : logo
    alert($("img").attr('alt'));
     // changement de l'attribut alt : logo UPEM
    $("img").attr('alt','logo UPEM');
    ```

Propriétés CSS

- $(‘selecteur’).css(‘prop’)
  - Retourne la propriété CSS
  - Ex : $(‘p’).css(‘color’)

- $(‘selecteur’).css(‘prop’, ‘val’)
  - Affecte val à la propriété CSS prop
  - Ex : $(‘p’).css(‘color’, ‘red’)

- ‘val’ peut être une fonction :

Attribution d'une valeur à l'attribut CSS des éléments de classe CSS «id» en fonction de leur valeur actuelle à l'aide d'une fonction :

```
var tailleActuelle =
   parseInt($('.id').css("font-size"));
$('.id').css("font-size",function(){
   return tailleActuelle+10;
});
```

augmente de 10 points la taille de police de caractères des éléments de classe «id»

- Si le sélecteur retourne plusieurs éléments et l'on souhaite effectuer un traitement ciblé : each
  - $(this) : élément courant
  - i : index de l'élément courant (i ou tout autre mot)

```
$("table tr").each(function(i){
  if (i % 2)
  $(this).addClass("odd");
});
```

  - Exercice exoJQ1 modification d'un document

- jQuery est aussi du Javascript, mais ne contient que des Objets jQuery.
  Pour passer d'une var Javascript vers/depuis l'objet jQuery :

  - var variableJS = 'un simple texte';
    var variableJQ = $(variableJS); // JS en JQ

  - var spans = $('span').get(); // JQ en JS

- Nombreux :
  blur, focus, load, resize, scroll, unload, beforeunload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error
- https://www.w3schools.com/jquery/jquery_events.asp

- Associer une fonction à un événement :

```
// associer une fonction à un événement
$("div").on("click", function() {
   $(this).text("code HTML : "+$(this).html())
});
// arrêter d'exécuter l'événement
$("div").on("click", function() {
   $(this).text("code HTML : "+$(this).html());
   $("div").off("click")
});
// exécuter une seule fois (pour chaque objet)
$("div").one("click", function() {
   $(this).text("code HTML : "+$(this).html())
});
```

- On peut mettre plusieurs événements différents sur un même élément

```
$("p").on({
    mouseenter: function(){
        $(this).css("background-color", "lightgray");
    },
    mouseleave: function(){
        $(this).css("background-color", "lightblue");
    },
    click: function(){
        $(this).css("background-color", "yellow");
    }
});
```

- On peut mettre plusieurs événements identiques sur un même élément

```
<a href="">clic</a>
<script>
  $("a").click(function(event) {
    alert(event.type);
  });
  $("a").click(function(event) {
    alert(event.pageX + ", " + event.pageY);
  });
</script>
```

- Comme en Javascript permet de retrouver des infos de l’événement

```
<div id="log"></div>
<script>
  $(document).on('mousemove',function(e){
  $("#log").text(e.pageX + ", " + e.pageY);
});
</script>
```

- Exemple : menu déroulant multi-niveaux.

```
<ul>
 <li> Niveau 1 : item 1</li>
 <li> Niveau 1 : item 2
  <ul><li> Niveau 2 : item 1</li>
  <li> Niveau 2 : item 2
  <ul><li> Niveau 3 : item 1</li>
    <li> Niveau 3 : item 2</li></ul>
  </li></ul>
 </li>
</ul>
```

```
$(document).ready(function() {
$("li").click(function () {
alert($(this).html());
});
});
```

- si on clique sur `<li> Niveau 3 : item 2</li>` alors on clique aussi sur le <li> du niveau 2 et celui du niveau 1.
- on a donc trois alertes.
- la propagation est ascendante.
- Si souhaité, on peut stopper la propagation des événements par e.stopPropagation();

- Show, hide, fade, slide, animate

```
$("button").click(function(){
    $("div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
    });
});
```

```
$("button").click(function(){
    var div = $("div");
    div.animate({height: '300px', opacity: '0.4'}, "slow");
    div.animate({width: '300px', opacity: '0.8'}, "slow");
    div.animate({height: '100px', opacity: '0.4'}, "slow");
    div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

Chargement de fichier

- Méthode alternative à AJAX : modifier le document avec une partie de fichier externe :
  - `$("div").load("fichier.html");`
    Déverse le contenu de fichier.html dans la balise div
  - `$("div#content").load("fichier.php",{"nom":"philippe"});`
    Envoie une requête en POST au serveur en demandant l'exécution de fichier.php avec comme paramètre nom = 'philippe'
  - `$("div").load('test.html #monid');`
    Dans le fichier test.html, extrait l'élément id='monid' et le place dans le div