

## Plan

- **jQuery**
- **jQuery UI**
- **AngularJS**

### Les premiers « Frameworks »

- Comme il était devenu difficile de coder du javascript pour tous les navigateurs, sont apparus des « Frameworks » permettant une spécification unique, indépendante du navigateur
  - PrototypeJS - [www.prototypejs.org](http://www.prototypejs.org)
    - [script.aculo.us](http://script.aculo.us)
  - Mootools - [mootools.net](http://mootools.net)
  - DoJo Toolkit - [www.dojotoolkit.org](http://www.dojotoolkit.org)
  - Yahoo UI - [developer.yahoo.com/yui/](http://developer.yahoo.com/yui/)
  - ExtJS - [www.extjs.com](http://www.extjs.com)
  - UIZE - [www.uize.com](http://www.uize.com)

### Avant-propos

- **jQuery**
  - Une bibliothèque javascript open-source et cross-browser
  - Elle permet de traverser et manipuler très facilement l'arbre DOM des pages web à l'aide d'une syntaxe fortement similaire à celle d'XPath.
  - JQuery permet par exemple de changer/ajouter une classe CSS, créer des animations, modifier des attributs, etc.
  - Gérer les événements javascript
  - Faire des requêtes AJAX simplement

### Avant-propos

- jQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant Ajax) et HTML, et a pour but de simplifier des commandes communes de JavaScript.
  - La première version date de janvier 2006.
- jQuery est "DOM centrique" et a principalement été créé dans le but de :
  - manipuler des pages (DOM + CSS) dynamiquement
  - prendre en compte des données externes (AJAX)
- jQuery 1.8, par exemple, se présente comme un unique fichier JavaScript de 91.2KB en version minimifié et 32.64KB (min + gzip) contenant toutes les fonctions de base.
  - Il peut être inclus dans toute page web en utilisant le code suivant :

```
<script src="http://code.jquery.com/jquery-1.9.1.min.js" type="text/javascript"> </script>
```

### Ses fonctions principales

- Sélecteurs : Parcourir le DOM avec les sélecteurs CSS 1 à 3 et un support basique de XPath. Filtrer les résultats
- Eléments et attributs : Changer un éléments ou avoir des informations sur lui
- Evènements : Actions utilisateurs ou de modules JS
- CSS : Manipuler les styles
- Animations / Effets : Changer l'état d'un élément dans l'instant ou sur la durée
- AJAX : Requêtes externes à la page
- Plugins : Extensions spécifiques au cas par cas
- Fonctions pratiques : Détection navigateur, manipulation d'objets JS, *deferred*, etc

### jQuery versus Javascript

```
var element = document.getElementById('elementId'); // Natif  
var $element = $('#elementId'); // jQuery  
var elements = document.getElementsByClassName('elementClass'); // Natif  
var $elements = $('.elementClass'); // jQuery
```

# Requête Ajax en JS natif

```
1 ▼ function load(url, callback) {  
2     var xhr;  
3     if (typeof XMLHttpRequest !== 'undefined') xhr = new XMLHttpRequest();  
4 ▼     else {  
5         var versions = ["MSXML2.XmlHttp.5.0", "MSXML2.XmlHttp.4.0", "MSXML2.XmlHttp.3.0", "MSXML2.XmlHttp.2.0", "Microsoft.XmlHttp"]  
6 ▼         for (var i = 0, len = versions.length; i < len; i++) {  
7 ▼             try {  
8                 xhr = new ActiveXObject(versions[i]);  
9                 break;  
10                } catch (e) {}  
11            }  
12        }  
13        xhr.onreadystatechange = ensureReadiness;  
14 ▼        function ensureReadiness() {  
15            if (xhr.readyState < 4) return;  
16            if (xhr.status !== 200) return;  
17            // all is well  
18            if (xhr.readyState === 4) callback(xhr);  
19        }  
20        xhr.open('GET', url, true);  
21        xhr.send('');  
22    }  
23  
24    //Our simplified "load" function accepts a URL and CALLBACK parameter.  
25 ▼ load('http://www.monsite.com/serveur.php', function (xhr) {  
26     document.getElementById('container').innerHTML = xhr.responseText;  
27 });  
28
```

### Requête Ajax avec JQuery

```
1 ▼ $.ajax({  
2     url: 'http://www.monsite.com/serveur.php',  
3 ▼     success: function (data) {  
4         $('#container').html(data);  
5     },  
6 ▼     error: function (jqXHR, textStatus, errorThrown) {  
7         console.log(textStatus);  
8     }  
9 });
```

```
10 // ou encore...  
11 ▼ $.get('http://www.monsite.com/serveur.php').done(function () {  
12     alert('OK');  
13 });
```



### Une simple bibliothèque à importer

- Disponible sur le site de JQuery  
<http://jquery.com/>

```
<script type="text/javascript"
  src="jquery.js"></script>
```

- Ou directement sur Google code

```
<script type="text/javascript"
  src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
</script>
```

### Autres Frameworks de la famille jQuery

- JQUERY UI
  - <http://jqueryui.com/>
  - <http://jqueryui.com/demos/>
  - Alternative : <http://jquerytools.org/demos/>
- JQUERY MOBILE
  - <http://jquerymobile.com/>
  - <http://jquerymobile.com/demos/1.1.1/>



### Le signe \$

- C'est un identifiant. Vous pouvez le considérer comme étant le nom d'une fonction

<script>

```
jQuery("#divTest1").text("Hello, world!");
```

```
$("#divTest1").text("Hello, world!");
```

</script>

- Vous pouvez utiliser jQuery à la place du signe \$

### Un premier exemple de fichier utilisant jQuery

```
1  <!DOCTYPE html>
2  <html lang="fr">
3
4  <head>
5      <meta charset="utf-8">
6      <title>jQuery : Demo</title>
7  </head>
8
9  <body>
10     <h1>Démo</h1>
11     <p><a href="http://jquery.com/">jQuery</a></p>
12     <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.2/jquery.min.js" type="text/javascript"></script>
13     <script type="text/javascript">
14         $(document).ready(function() {
15             $("a").click(function(event) {
16                 event.preventDefault();
17                 alert("As you can see, the link no longer took you to jquery.com");
18             });
19         });
20
21     </script>
22 </body>
23
24 </html>
25
```

### jQuery est utilisé pour manipuler le DOM

- Pour attendre que le DOM soit entièrement chargé et ainsi éviter des erreurs:

#### Version JavaScript

```
document.addEventListener('DOMContentLoaded', function() { // Ici le DOM est prêt });
```

#### Version jQuery

Avec jQuery, nous utiliserons la méthode `.ready()` prévue à cet effet.

```
jQuery(document).ready(function() { // Ici le DOM est prêt });
```

Cette écriture jQuery peut être simplifiée par l'alias JavaScript `$`:

```
$(document).ready(function() { // Ici le DOM est prêt });
```

Afin d'être encore plus concis le `$(document).ready` peut être omis :

```
$(function() { // Ici le DOM est prêt });
```

### Quel est la logique d'utilisation de jQuery

Une fois l'élément HTML sélectionné `$()` - il se «transforme» en objet jquery et vous pouvez lui appliquer toutes les .methodes (fonctions) incluant des paramètres d'exécutions. Sur le principe :

```
$(selecteur).methode(parametres);
```

Exemple :

```
<p id='demo1'>Ceci est un texte ciblé par jQuery</p>
```

```
$(function() {  
  $('#demo1').css('color', 'red');  
});
```

Passe la couleur du texte en rouge (propriété CSS)

### Sélection d'éléments

- Possibilité de sélectionner :
- par type de bloc
- par identifiant
- par classe
- en combinant les critères
- en filtrant sur les noms d'attributs
- en faisant référence aux positions relatives dans le DOM
- en ne récupérant qu'un seul élément parmi les objets sélectionnés
- en filtrant parmi les objets sélectionnés

### Sélection d'éléments par type de bloc, identifiant, classe

Pour renvoyer toutes les balises :

```
$("*")
```

Pour renvoyer tous les <div> de la page :

```
$("div")
```

```
// <span id="test">JL</span>
```

```
$("#test")
```

```
// <ul class="test">JL</ul>
```

```
$(".test")
```

**\$("div").length** donne le nombre de div dans la page



### Sélection d'éléments par combinaison de critères

```
// tous les divs de classe main  
$("div.main")
```

```
// tous les tableaux d'identifiant data  
$("table#data")
```

```
// objets d'id "content" ou de classe "menu"  
// attention à la position des guillemets  
$("#content, .menu")
```

### Sélection d'éléments filtrée

// Recherche de p contenant des objets avec classe header

// rendre visible ces objets

```
$("#p").find(".header").show();
```

// similaire à :

// \$(selecteur, contexte)

```
$(".header", $("#p")).show();
```

### Sélection d'éléments filtrée par numéro d'élément

- // récupération de tous les éléments + extraction  
\$("div").get(2)  
\$("div")[2] // équivalent  
// récupération d'un seul élément  
\$("div").eq(2)  
// en partant de la fin  
\$("div").eq(-2)

### Sélection d'éléments fondée sur la structure du DOM

- Possibilité d'atteindre :
  - les fils (>) ;
  - tous les descendants (espace) ;
  - le (+) ou les (~) frères suivants

```
<ul>
  <li>item 1</li>
  <li>item 2</li>
  <li class="trois">item 3
    <ol>
      <li>3.1</li>
    </ol>
  </li>
  <li>item 4
    <ol>
      <li>4.1</li>
    </ol>
  </li>
  <li>item 5</li>
</ul>
```

```
// cache 4 et 5
$('li.trois ~ li').hide();
```

```
// cache 4
$('li.trois + li').hide();
```

```
// cache les <ol>
$('ul ol').hide();
```

```
// ne cache rien
$('ul > ol').hide();
```

### Sélection d'éléments fondée sur la structure du DOM

- Sélection d'éléments fondée sur la structure du DOM
  - frère, enfants, parents
  - utilisation de fonctions

**// frère suivant**

**.next(expr)**

**// frère précédent**

**.prev(expr)**

**// frères**

**.siblings(expr)**

**// enfants**

**.children(expr)**

**// père**

**.parent(expr)**

### Autres sélecteurs

**// premier paragraphe**  
**p:first**

**// si l'élément est visible**

**\$("div:visible")**

**// dernier élément de liste**  
**li:last**

**// sinon**

**\$("div:hidden")**

**// quatrième lien**  
**a:nth(3) ou a:eq(3)**

**// paragraphes pairs ou impairs**  
**p:even or p:odd every**

**// Tous les liens à partir (greater than) du quatrième ou avant (lower than)**  
**a:gt(3) or a:lt(4)**

**// Liens qui contiennent le mot click**  
**a:contains('click')**

### Sélecteurs de formulaire

```
// sélectionner les cases à cocher  
$("input:checkbox")
```

```
// sélectionner les boutons radio  
$("input:radio")
```

```
// sélectionner les boutons  
$(":button")
```

```
// sélectionner les champs texte  
$(":text")
```

```
$("input:checked")
```

```
$("input:selected")
```

```
$("input:enabled")
```

```
$("input:disabled")
```

```
<select name="valeur">  
  <option value="1">1</option>  
  <option value="2" selected="selected">2</option>  
  <option value="3">3</option>  
</select>  
$("select option:selected").val()
```

### Fonction foreach

- Appelle une fonction pour chaque élément sélectionné :
  - `$(this)` : élément courant
  - `i` : index de l'élément courant

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



### Modifier le contenu HTML

- `.html('[contenu]')` : remplacement du contenu d'un élément (les balises sont considérées comme des balises)
- `.text('[contenu]')` : remplacement du contenu d'un élément en considérant le tout comme du texte (les caractères `<` et `>` des balises sont remplacés par les entités XML (`>` et `<`))
- `.after('[contenu]')` : insertion du contenu après l'élément sélectionné
- `.before('[contenu]')` : insertion du contenu avant l'élément sélectionné
- `.append('[contenu]')` : insertion du contenu dans l'élément sélectionné à la suite des éléments existants
- `.prepend('[contenu]')` : insertion du contenu dans l'élément sélectionné avant les éléments existants

### Modifier le contenu HTML

`.wrap('<balise></balise>')` : insertion des balises passées en paramètre de part et d'autre de l'élément

`.wrapInner('<balise></balise>')` : insertion des balises passées en paramètre de part et d'autre des enfants de l'élément

`.unwrap()` : suppression de la balise parent

Possibilité de combiner les modifications les unes à la suite des autres :  
`$("div").html("Hello jQuery").wrapInner('<b></b>')`

Attention à la lisibilité !

Possibilité de récupérer du contenu d'un autre objet pour le passer en paramètre :

`$("div.a").html($("div.c").html());`

→ met le contenu du div.c dans le div.a

### Récupérer ou modifier les propriétés CSS

Récupération de la valeur de l'attribut CSS d'un élément :

`.css('color')` renvoie la couleur de l'élément

Attribution d'une valeur à l'attribut CSS d'un élément :

`.css('color','red')` attribue la couleur rouge à l'élément

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»

Attribution d'un ensemble de valeurs à un ensemble d'attributs CSS d'un élément :

`.css({'border' : '1px solid black', 'color' : 'red'})` attribue la couleur rouge à l'élément et lui ajoute une bordure noire

### Récupérer ou modifier la classe CSS

`.addClass('laClasse')`

Ajoute la classe CSS laClasse à l'élément.

`.removeClass('laClasse')`

Retire la classe CSS laClasse à l'élément.

`.toggleClass('laClasse')`

Ajoute la classe CSS laClasse à l'élément s'il ne l'a pas, la lui retire sinon.

`.hasClass('laClasse')`

Renvoie true si l'élément a la classeCSS laClasse, false sinon

### Principe de contexte

- un sélecteur reste une opération de recherche à réaliser à chaque appel. Et cela coûte en ressources et en temps. Il est donc important de mettre en cache les sélecteurs qui vous servent plusieurs fois. La méthode est très simple

```
var monId = $("#monId");
```

L'avantage réside dans le fait que le navigateur n'aura plus à chercher dans le DOM

```
Var sideBar = $("#sideBar");  
sideBar.find("input");
```

Retourne tous les éléments <input> compris dans l'élément ayant pour id "sideBar"

```
$("#sideBar").find("input"); // 16% moins efficace  
$("input", $("#sideBar")); // 23% moins efficace  
$("#sideBar").children("input"); // 50% moins efficace  
$("#sideBar > input"); // 70% moins efficace  
$("#sideBar input"); // 77% moins efficace  
// Utilisation du cache  
var sideBar = $("#sideBar");  
$("input", sideBar); // 5 à 10% moins efficace
```

### N'utilisez jQuery que si nécessaire

- Normalement, vous ne devriez pas avoir ce cas en production mais il illustrera très bien le propos :

```
jQuery(document).ready(function() {  
    $("a").click(function () {  
        console.log("Id de l'élément cliqué : " + $(this).attr("id"));  
    });  
});
```

- Ici vous cherchez à afficher en console l'id de l'élément sur lequel on viendrait de cliquer. En soit, ce code n'est pas erroné. Mais vous n'avez pas besoin d'un objet jQuery pour récupérer l'id de this :

```
jQuery(document).ready(function() {  
    $("a").click(function () {  
        console.log("Id de l'élément cliqué : " + this.id);  
    });  
});
```

### Ajout d'un gestionnaire d'événements

- Ajout d'un gestionnaire d'événements sur les éléments du DOM déjà existants
- jQuery propose une fonction puissante pour ajouter facilement un gestionnaire d'événements sur des éléments déjà existants. Admettons qu'on veuille afficher une alerte quand on clique sur un paragraphe:

```
jQuery(document).ready(function() {  
    $("p").click(function(event) {  
        alert(this.innerHTML); //$ (this).html() pour les puristes  
    });  
});
```

### Ajout d'un gestionnaire d'événements

- Ajout d'un gestionnaire d'événements sur les futurs éléments du DOM
- Le souci de la méthode précédente est qu'à chaque fois que vous allez modifier le DOM, vous allez devoir rappeler tous vos ajouts de gestionnaire d'événements. Un peu lourd!

```
jQuery(document).ready(function() {  
    $("p").click(function(event) {  
        $("body").append('<p>Hello world!</p>');  
    });  
});
```

Premier test, ça ne marche que si vous cliquez sur le premier paragraphe. Pas avec les nouveaux paragraphes ajoutés.



### Ajout d'un gestionnaire d'événements

- Ajout d'un gestionnaire d'événements sur les futurs éléments du DOM

```
var ajouterP = function () {  
    $("body").append("<p>Hello world!</p>");  
    $("p").unbind("click").click(ajouterP);  
}
```

```
jQuery(document).ready(function(){  
    $("p").click(ajouterP);  
});
```

Avec la fonction `.live()` maintenant :

```
jQuery(document).ready(function(){  
    $("p").live('click', function () {  
        $("body").append("<p>Hello world!</p>");  
    });  
});
```

### **\$.each()**

- jQuery.each() est une fonction itérative, elle peut être utilisée pour parcourir les objets et les tableaux qui sont itérés par un index numérique, de 0 à taille-1. Les autres objets sont itérés à travers leurs propriétés.

```
var array = [ 'un', 'deux', 'trois', 'quatre', 'cinq' ];
```

```
jQuery.each(array, function(index, value) {  
    // faire quelque chose avec `value` (ou `this` qui est `value` )  
    alert(index+": "+ value);  
});
```

0: un

1: deux

2: trois

3: quatre

4: cinq

### \$.each()

```
var json = [  
  { 'red': '#f00' },  
  { 'green': '#0f0' },  
  { 'blue': '#00f' }  
];  
  
$.each(json, function () {  
  $.each(this, function (name, value) {  
    console.log(name + '=' + value);  
  });  
});
```

red=#f00, green=#0f0, blue=#00f

### \$.each()

```
<div class="productDescription">Red</div>
<div>Pink</div>
<div class="productDescription">Orange</div>
<div class="generalDescription">Teal</div>
<div class="productDescription">Green</div>
```

```
$.each($('.productDescription'), function (index, value) {
  console.log(index + ':' + $(value).text());
});
```

0:Red, 1:Orange, 2:Green.

```
$('.productDescription').each(function () {
  console.log($(this).text());
});
```

Red  
Orange  
Green

### Envoi d'un formulaire via jQuery

```

1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <style>
6          p {
7              margin: 0;
8          }
9
10         div,
11         p {
12             margin-left: 10px;
13         }
14
15         span {
16             color: red;
17         }
18     </style>
19 </head>

```

192.168.0.131/mai/ex2.html

Type 'correct' to validate.

Validated...



```

20
21 <body>
22     <p>Type 'correct' to validate.</p>
23     <p><span></span></p>
24     <form action="update.php" method="get">
25         <div>
26             <input type="text" />
27             <input type="submit" /> </div>
28     </form>
29     <script src="jquery-1.2.6.js" type="text/javascript"></script>
30     <script>
31         alert("hello0");
32         $(document).ready(function () {
33             $('form').submit(function (e) {
34                 e.preventDefault();
35                 if ($('#input:first').val() == "correct") {
36                     $('#span').text('Validated...').show();
37                     return true;
38                 }
39                 $("#span").text("Not valid!").show().fadeOut(1000);
40                 return false;
41             });
42         });
43     </script>
44 </body>
45
46 </html>

```

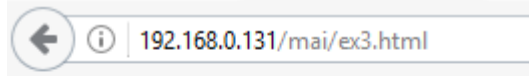
### Pour obtenir la valeur d'un champs de formulaire

- <http://api.jquery.com/val/>

```
var name  = $("input#name").val();  
var price = $("select#priceList").val();  
var infos = $("textarea#infos").val();  
// ...
```

# La fonction serialize

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
6   <script>
7     $(document).ready(function() {
8       $("button").click(function() {
9         $("div").text($("#form").serialize());
10      });
11    });
12  </script>
13 </head>
14
15 <body>
16
17   <form action="">
18     First name: <input type="text" name="FirstName" value="Mickey"><br> Last name: <input type="text" name="LastName"
19     value="Mouse"><br>
20   </form>
21
22   <button>Serialize form values</button>
23
24   <div></div>
25
26 </body>
27
28 </html>
```



First name:

Last name:

FirstName=Mickey&LastName=Mouse

# La fonction serialize

```
19 <body>
20   <p>cliquer sur le button</p>
21   <div id="stage1" style="background-color:yellow;"> STAGE - 1 </div>
22   <br />
23   <div id="stage2" style="background-color:yellow;"> STAGE - 2 </div>
24   <form id="testform">
25     <table>
26       <tr>
27         <td>
28           <p>Nom:</p>
29         </td>
30         <td>
31           <input type="text" name="nom" size="40" />
32         </td>
33       </tr>
34       <tr>
35         <td>
36           <p>Age:</p>
37         </td>
38         <td>
39           <input type="text" name="age" size="40" />
40         </td>
41       </tr>
42       <tr>
43         <td>
44           <p>Genre:</p>
45         </td>
46         <td>
47           <select name="genre">
48             <option value="Homme" selected>Homme</option>
49             <option value="Femme" selected>Femme</option>
50           </select>
51         </td>
52       </tr>
53       <tr>
54         <td colspan="2">
55           <input type="button" id="driver" value="Load Data" /> </td>
56         </tr>
57     </table>
58   </form>
59 </body>
```

← ⓘ 192.168.0.131/mai/ex4.html

cliquer sur le button

STAGE - 1

STAGE - 2

Nom:

Age:

Genre: Femme ▼

Load Data



### La fonction serialize

```
3 <head>
4 <title>The jQuery Example</title>
5 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
6 <script type="text/javascript" language="javascript">
7     $(document).ready(function() {
8         $("#driver").click(function(event) {
9             $.post("ex4.php", $("#testform").serialize(), function(data) {
10                 $('#stage1').html(data);
11             });
12             var str = $("#testform").serialize();
13             $("#stage2").text(str);
14         });
15     });
16
17 </script>
18 </head>
```

```
1 <?php
2 if( $_REQUEST["nom"] ) {
3
4     $nom = $_REQUEST['nom'];
5     echo "Bonjour ". $nom;
6     $age = $_REQUEST['age'];
7     echo "<br /> age : ". $age;
8     $genre = $_REQUEST['genre'];
9     echo "<br />genre : ". $genre;
10 }
11 ?>
12
```

### find(selector)

- Localise tous les descendants d'un type d'élément

```
1 <html>
2
3 <head>
4   <title>The JQuery Example</title>
5   <script type = "text/javascript"
6     src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
7
8   <script type = "text/javascript" language = "javascript">
9     $(document).ready(function() {
10       $("p").find("span").addClass("selected");
11     });
12   </script>
13
14   <style>
15     .selected { color:red; }
16   </style>
17 </head>
18
19 <body>
20   <p>This is 1st paragraph and <span>THIS IS RED</span></p>
21   <p>This is 2nd paragraph and <span>THIS IS ALSO RED</span></p>
22 </body>
23
24 </html>
```

This is 1st paragraph and THIS IS RED

This is 2nd paragraph and THIS IS ALSO RED

### width & length

```
1 <html>
2
3 <head>
4   <title>The jQuery Example</title>
5   <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
6
7   <script type="text/javascript" language="javascript">
8     $(document).ready(function() {
9       $("div:first").width(100);
10      $("div:first").css("background-color", "blue");
11    });
12
13  </script>
14
15  <style>
16    div {
17      width: 70px;
18      height: 50px;
19      float: left;
20      margin: 5px;
21      background: red;
22      cursor: pointer;
23    }
24
25  </style>
26 </head>
27
28 <body>
29
30   <div></div>
31   <div>d</div>
32   <div>d</div>
33   <div>d</div>
34   <div>d</div>
35
36 </body>
37
38 </html>
39
```



- jQuery propose par défaut une série de fonctions d'animation et de transition qui permettent d'animer des éléments du DOM.

### hide, show

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="http://code.jquery.com/jquery-latest.min.js">
<title>animation1</title>
</head>
<body>
<div id="exemple-show">
  <a href="#">show</a>
  <span>Du texte qui apparaît brutalement</span>
</div>
<div id="exemple-hide">
  <a href="#">hide</a>
  <span>Du texte qui disparaît brutalement</span>
</div>
<div id="exemple-fadetoggle">
  <a href="#">fadeToggle</a>
  <span>Du texte qui apparaît/disparaît en fondu à chaque clic</span>
</div>
</body>
</html>
```

```
jQuery(document).ready(function($){
  // SHOW/HIDE
  $('#exemple-show span').hide();
  $('#exemple-show a').click(function(e){
    $('#exemple-show span').show();
  });

  $('#exemple-hide a').click(function(e){
    $('#exemple-hide span').hide();
  });

  // FADE
  $('#exemple-fadetoggle a').click(function(e){
    $('#exemple-fadetoggle span').fadeToggle();
  });
});
```

show

hide Du texte qui disparaît brutalement

fadeToggle Du texte qui apparaît/disparaît en fondu à chaque clic

### Les dessous techniques

- **AngularJS** est un framework JavaScript open source, développé par **Google** :
  - Facilitant la création de SPA (site web monopage ou single-page application en anglais).
  - Fourniture de tous les mécanismes techniques nécessaires à la création de ce type d'application de nouvelle génération et d'apporter une structure permettant de développer une application robuste et organisée.



## Composition

```
$( "li:odd" ).prepend( '<span>Changed</span>' ).css( {background:"red"} );
```

```
<ul>
  <li>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    Third item
  </li>
</ul>
```

```
<ul>
  <li>
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    <span>Changed</span>
    Third item
  </li>
</ul>
```

```
<ul>
  <li style="background:red;">
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li style="background:red;">
    <span>Changed</span>
    Third item
  </li>
</ul>
```

# Composition

```
$("div:hidden").find(".foo").empty().text("Changed").end().show();
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none" >
  <span>
    More text
  </span>
  <span class="foo">
    Goodbye cruel world.
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Goodbye cruel world.
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Goodbye cruel world.
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:block">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```



### Determiner si une checkbox est cochée

```
If ($('#total').attr('checked')) {  
    //Traitement si cochée  
}  
else {  
    //Traitement si non cochée  
}
```

### Intercepter le bouton submit d'un formulaire

```
$(document).ready(function() {  
    $('#ok').submit(function() {  
        if ($('#login').val() == '') {  
            alert ('Entrer un login');  
            return false;  
        }  
    })  
});
```

### Effacer le contenu d'un champs de texte lorsqu'il a le focus

```
<input name="nom" type="text" id="nom" value="Entrez votre nom">
```

```
$('#nom').focus(function() {  
    var field = $(this);  
    field.val('');  
});
```

### Tester le clic sur n'importe quel bouton radio

```
$(':radio').click(function() {  
    //do stuff  
});
```

## siblings

```
$("#span.none").click(
  function() {
    $(this).siblings(":checkbox").removeAttr("checked");
  }
);
```

```
$("#span.all").click(
  function() {
    $(this).siblings(":checkbox").attr("checked", "checked");
  }
);
```

or

```
$("#span").click(
  function() {
    if ($(this).text() == "Select All")
      $(this).siblings(":checkbox").attr("checked", "checked");
    else if ($(this).attr("class") == "none")
      $(this).siblings(":checkbox").removeAttr("checked");
  }
);
```

```
<div>
  <span class="all">Select All</span>
  <span class="none">Select None</span>
  <input name="chk1" type="checkbox"/>
  <input name="chk2" type="checkbox"/>
  <input name="chk3" type="checkbox"/>
</div>

<div>
  <span class="all">Select All</span>
  <span class="none">Select None</span>
  <input name="chk4" type="checkbox"/>
  <input name="chk5" type="checkbox"/>
  <input name="chk6" type="checkbox"/>
</div>
```

### Divers

*// récupère le contenu du dernier paragraphe trouvé et stocke dans la variable "contenu"*

```
var contenu = $('p:last').html();
```

*// récupération du contenu du champ de texte*

```
var inputText = $('input:text').val();
```

*// modification du contenu du champ de text*

```
$('input:text').val('Nouvelle valeur');
```

```
var contenu = $('p').text();  
alert(contenu);
```

### Ajax

- JQuery possède toute une panoplie de fonctions permettant de simplifier les requêtes Ajax
- La plus simple :  
`$('#maDiv').load('page.html');`
- Plus complexe :  
`$.get('test.html',  
function(data) {faire quelque chose});`
- Générale : `$.ajax({  
url: 'document.xml',  
type: 'GET',  
dataType: 'xml',  
timeout: 1000,  
error: function(){alert('Erreur chargement'); },  
success: function(xml){faire quelque chose}  
});`

### Ajax/Json

```
<html>
<head>
<title>AJAX Demo</title>
<script type="text/javascript" src="jquery.js">
</script>
<script type="text/javascript">
var cnt = 0;
$(function(){
    $.ajaxSettings({
        error:function(){alert("Communication error!");}
    });
    $("button").click(function(){
        var input = {in:$(":textbox").val(),count:cnt};
        $.getJSON("ajax.php",input,function(json){
            cnt = json.cnt;
            $(".cnt").text(cnt)
            $(".msg").text(json.out);
        });
    });
});
</script>
</head>
<body>
<p>
    Input:
    <input type="textbox"/>
    <input type="button" value="Send"/>
    Output #
    <span class="cnt"></span>:
    <span class="msg"></span>
</p>
</body>
</html>
```

```
<?php
$output = '';

switch($_REQUEST['in']) {
    case 'hello':
        $output = 'Hello back.';
        break;
    case 'foo':
        $output = 'Foo you, too.';
        break;
    case 'bar':
        $output = 'Where Andy Capp can be found.';
        break;
    case 'foobar':
        $output = 'This is German, right?';
        break;
    default:
        $output = 'Unrecognized string.';
}

$count = $_REQUEST['count']+1;

echo json_encode(
    array(
        'out' => $output,
        'cnt' => $count
    )
);

exit;
?>
```



### JSON

- format de données textuel, générique, dérivé de la notation des objets de JavaScript
- permet de représenter de l'information structurée.
- décrit par la RFC 4627 de l'IETF.
- Le type MIME **application/json** est utilisé pour le transmettre par le protocole HTTP (notamment en Ajax)
  - Standard dans les web services .Net, Java EE, etc.

- Exemple :

```
{
  "Image": {
    "Width": 800,
    "Height": 600,
    "Title": "Vue du 15ème étage",
    "Thumbnail": {
      "Url": "http://www.example.com/481989943",
      "Height": 125,
      "Width": "100"
    },
    "IDs": [116, 943, 234, 38793]
  }
}
```

### jQuery et JSON

```
jQuery.getJSON( url, [ data ],  
               [ callback(data, textStatus) ] )
```

#### Exemple :

```
<html><head>  
  <script src="jquery.min.js"></script>  
</head>  
<body>  
  <div id="images" style="height: 300px"></div>  
  <script>  
    $.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?tags=b  
    esancon&tagmode=any&format=json&jsoncallback=?",  
      function(data) {  
        $.each(data.items, function(i,item){  
          $("<img/>").attr("src", item.media.m).appendTo("#images");  
          if ( i == 3 ) return false;  
        });  
      });  
  </script>  
</body></html>
```

- <http://jqueryui.com/>
- Un ensemble de composants graphiques téléchargeable à l'adresse <http://jqueryui.com/download>.
  - un noyau (Core)
  - des « comportements » (interactions)
    - draggable : pour glisser-déplacer un élément  
<http://jqueryui.com/demos/draggable/>
    - droppable : pour « déposer » un élément  
<http://jqueryui.com/demos/droppable/>
    - resizable : pour redimensionner un élément  
<http://jqueryui.com/demos/resizable/>
    - selectable : pour sélectionner des éléments à la souris  
<http://jqueryui.com/demos/selectable/>
    - sortable : pour trier des éléments  
<http://jqueryui.com/demos/selectable/>

### Exemple !

```
1 <html>
2
3 <head>
4   <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jquery-ui.css" rel="stylesheet">
5   <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
6   <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
7   <script type="text/javascript">
8     $(function() {
9       $('#dialogMsg').dialog();
10    });
11
12  </script>
13
14 </head>
15
16 <body>
17   <form id="form1" runat="server">
18     <div id="dialogMsg" title="First JQueryUI Example">
19       Hello this is my first JQueryUI example.
20     </div>
21   </form>
22 </body>
23
24 </html>
```

#### First JQueryUI Example

Hello this is my first JQueryUI example.

### Le plugin Draggable

- Dans la librairie jQuery UI, le système **de Drag & Drop** ne forme pas un seul et même plugin. En effet, il est décomposé en deux parties distinctes, à savoir le plugin Draggable et le plugin Droppable

#### Syntaxe:

```
$(selector, context).draggable (options)  
$(selector, context).draggable ("action", [params])
```


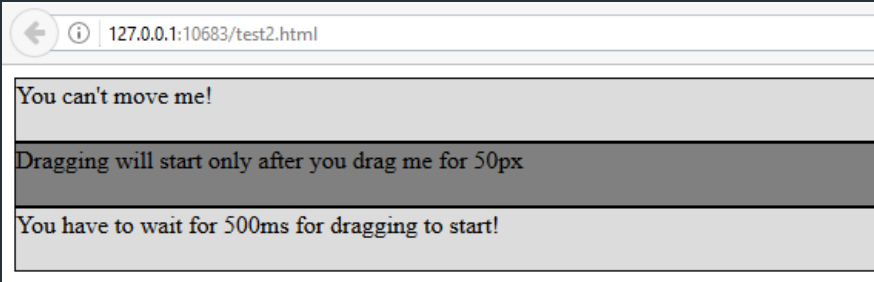
### Le plugin Draggable

```
1 <!DOCTYPE html>
2
3 <head>
4   <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jqueryui.css" rel="stylesheet">
5   <script src="http://code.jquery.com/jquery-1.10.2.js"></script>
6   <script src="http://code.jquery.com/ui/1.10.4/jquery-ui.js"></script>
7   <style>
8     #draggable {
9       width: 150px;
10      height: 150px;
11      padding: 0.5em;
12      background: #ee0;
13    }
14
15  </style>
16  <script>
17    $(function() {
18      $("#draggable").draggable();
19    });
20
21  </script>
22 </head>
23
24 <body>
25   <div id="draggable" class="ui-widget-content">
26     <p>Drag me !!!</p>
27   </div>
28 </body>
29
30 </html>
```



### Le plugin Draggable

```
10 <body>
11   <div id="div1" style="border:solid 1px;background-color:gainsboro;"> <span>You can't move me!</span>
12     <br />
13     <br /> </div>
14   <div id="div2" style="border:solid 1px;background-color:grey;"> <span>
15     Dragging will start only after you drag me for 50px
16   </span>
17   <br />
18   <br /> </div>
19   <div id="div3" style="border:solid 1px;background-color:gainsboro;"> <span>
20     You have to wait for 500ms for dragging to start!
21   </span>
22   <br />
23   <br /> </div>
24
25   <script>
26     $(function() {
27       $("#div1 span").draggable({
28         disabled: true
29       });
30       $("#div2 span").draggable({
31         distance: 50
32       });
33       $("#div3 span").draggable({
34         delay: 500
35       });
36     });
37
38   </script>
39 </body>
```



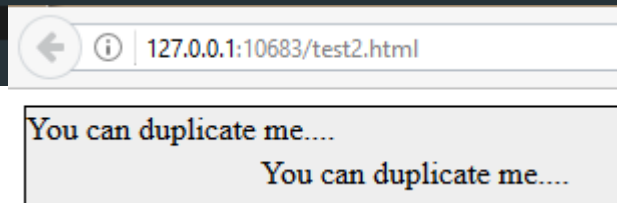
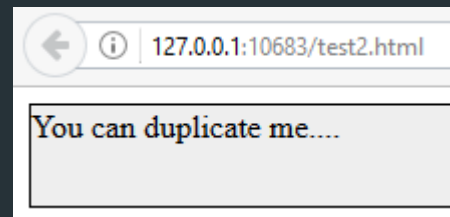
### Le plugin Draggable

```
10 ▼ <body>
11 ▼   <div id="div4" style="border:solid 1px;background-color:gainsboro;">
12     <span>You can drag me only within this div.</span><br /><br />
13   </div>
14 ▼   <div id="div5" style="border:solid 1px;background-color:grey;">
15     <span>You can drag me only along x axis.</span><br /><br />
16   </div>
17   <script>
18 ▼     $("#div4 span").draggable({
19       containment: "#div4"
20     });
21 ▼     $("#div5 span").draggable({
22       axis: "x"
23     });
24   </script>
25 </body>
26
```



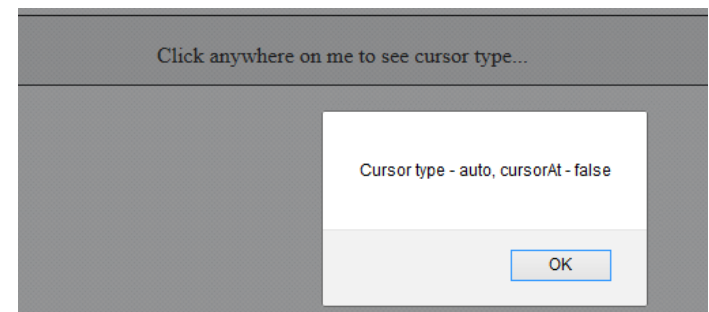
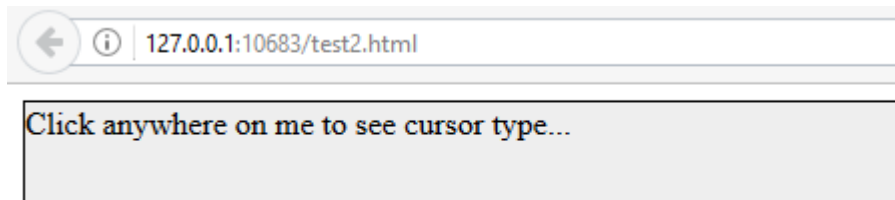
### Le plugin Draggable

```
1 <html>
2
3 <head>
4   <link href="http://code.jquery.com/ui/1.10.4/themes/ui-lightness/jqueryui.css" rel="stylesheet">
5   <script src="http://code.jquery.com/jquery-3.2.1.min.js"></script>
6   <script src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"></script>
7
8 </head>
9
10 <body>
11   <div id="div6" style="border:solid 1px;background:#eee; height:50px;">
12     <span>You can duplicate me....</span>
13   </div>
14   <script>
15     $("#div6 span").draggable({
16       helper: "clone"
17     });
18   </script>
19 </body>
20
21
22 </html>
```



### Le plugin Draggable

```
10 ▾ <body>
11 ▾   <div id="divX" style="border:solid 1px;background:#eee; height:50px;">
12     <span>Click anywhere on me to see cursor type...</span>
13   </div>
14   <script>
15     /* First make the item draggable */
16     $("#divX span").draggable();
17 ▾     $("#divX span").bind('click', function(event) {
18         var cursor = $("#divX span").draggable("option", "cursor");
19         var cursorAt = $("#divX span").draggable("option", "cursorAt");
20         alert("Cursor type - " + cursor + ", cursorAt - " + cursorAt);
21     });
22   </script>
23 </body>
24
```



### Le plugin Draggable

```
10 <body>
11   <div id="div9" style="border:solid 1px;background-color:gainsboro;">
12     <span>Drag me to check the event method firing</span><br /><br />
13   </div>
14   <script>
15     $("#div9 span").draggable({
16       cursor: "move",
17       axis: "x",
18       drag: function(event, ui) {
19         alert("hi..");
20       }
21     });
22   </script>
23 </body>
```

### Le plugin Droppable

- couplé avec le plugin Draggable, il va nous permettre de réaliser un système de Drag & Drop (glisser-déposer)

```
$('#drag').draggable(); // ce bloc sera déplaçable  
$('#drop').droppable(); // ce bloc servira de zone de dépôt
```

### Le plugin Droppable

```
11 <script>
12 $(function() {
13     $("#draggable-5").draggable();
14     $("#droppable-8").droppable({
15         drop: function(event, ui) {
16             $(this).addClass("ui-state-highlight").find("p").html("Dropped!");
17         },
18         over: function(event, ui) {
19             $(this).addClass("ui-state-highlight").find("p").html("moving in!");
20         },
21         out: function(event, ui) {
22             $(this).addClass("ui-state-highlight").find("p").html("moving out!");
23         }
24     });
25 });
26
27 </script>
28 </head>
29
30 <body>
31 <div id="draggable-5" class="ui-widget-content">
32     <p>Drag me to my target</p>
33 </div>
34 <div id="droppable-8" class="ui-widget-header">
35     <p>Drop here</p>
36 </div>
37 </body>
```

Drag me to my target

Drop here

moving in!

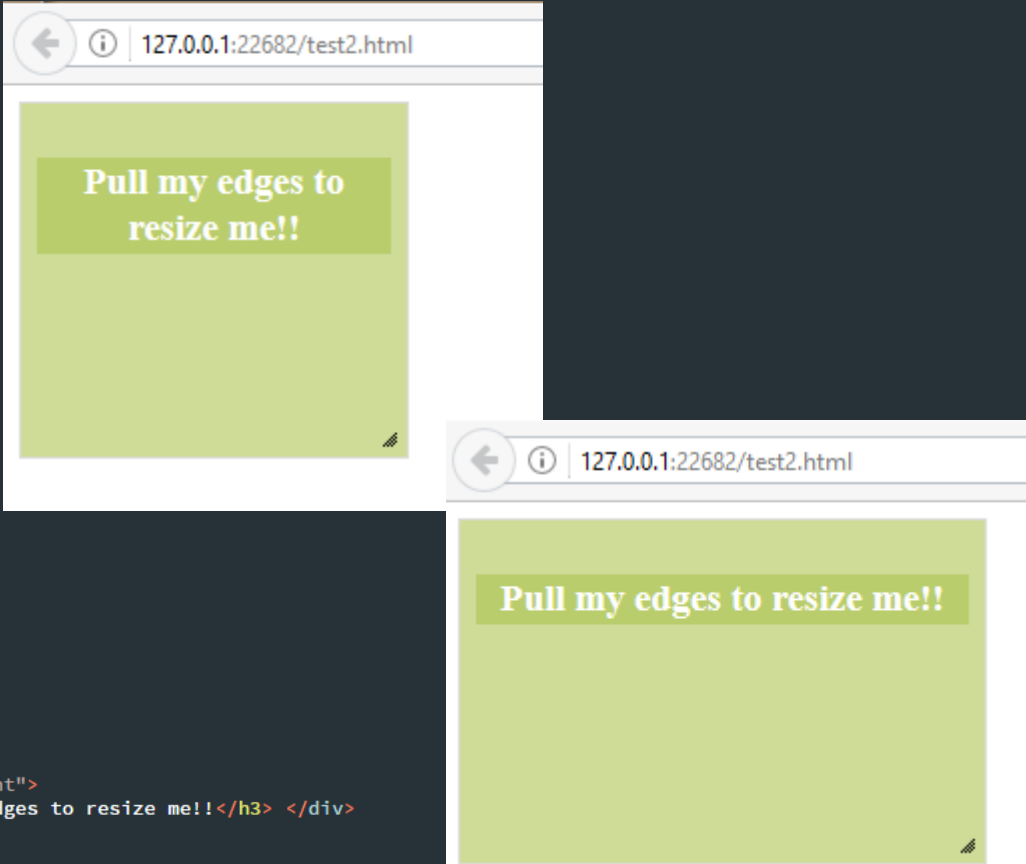
Drag me to my target

### Les plugins **Resizable** et **Selectable**

- **Resizable** permet de redimensionner à la volée un élément du DOM. **Selectable**, quant à lui, permet de sélectionner des éléments, et d'en récupérer les données, par exemple.

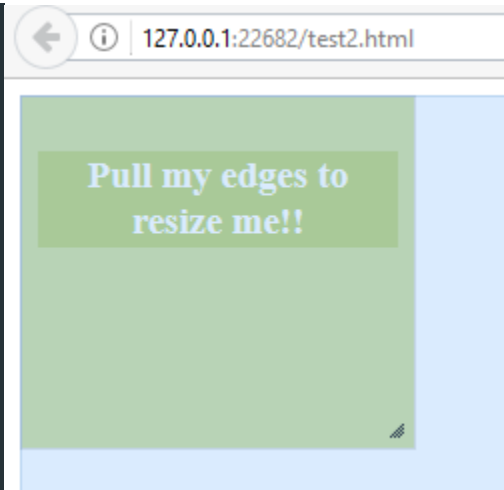
### Le plugin Resizable

```
6 <link type="text/css" rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1/themes/smoothness/jquery-ui.css" />
7 <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
8 <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jqueryui/1/jquery-ui.min.js"></script>
9 <!-- CSS -->
10 <style>
11   .ui-widget-header {
12     background: #b9cd6d;
13     border: 1px solid #b9cd6d;
14     color: #FFFFFF;
15     font-weight: bold;
16   }
17
18   .ui-widget-content {
19     background: #cedc98;
20     border: 1px solid #DDDDDD;
21     color: #333333;
22   }
23
24   #resizable {
25     width: 150px;
26     height: 150px;
27     padding: 0.5em;
28     text-align: center;
29     margin: 0;
30   }
31 </style>
32 <!-- Javascript -->
33 <script>
34   $(function () {
35     $("#resizable").resizable();
36   });
37 </script>
38 </head>
39
40 <body>
41   <!-- HTML -->
42   <div id="resizable" class="ui-widget-content">
43     <h3 class="ui-widget-header">Pull my edges to resize me!!</h3> </div>
44 </body>
45 </html>
```



### Le plugin Resizable

```
32 ▼      .help {
33          border: 1px solid #1877D5;
34          background: #84BEFD;
35          opacity: 0.3;
36      }
37
38  </style>
39  <!-- Javascript -->
40  <script>
41 ▼      $(function() {
42 ▼          $("#resizable").resizable({
43              helper: 'help', // je donne la classe stylisant mon helper
44              animate: true
45          });
46      });
47
48  </script>
49  </head>
50
51 ▼ <body>
52     <!-- HTML -->
53 ▼     <div id="resizable" class="ui-widget-content">
54         <h3 class="ui-widget-header">Pull my edges to resize me!!</h3>
55     </div>
56 </body>
57 </html>
```





### Le plugin Resizable

```
44 ▼ #resizable-4 {
45     background-position: top left;
46     width: 150px;
47     height: 150px;
48 }
49
50 #resizable-4,
51 ▼ #container {
52     padding: 0.5em;
53 }
54
55 </style>
56 <!-- Javascript -->
57 <script>
58 ▼ $(function() {
59 ▼     $("#resizable-4").resizable({
60         containment: "#container",
61         minHeight: 70,
62         minWidth: 100
63     });
64 });
65
66 </script>
67 </head>
68
69 ▼ <body>
70     <!-- HTML -->
71 ▼ <div id="container" class="ui-widget-content">
72 ▼     <div id="resizable-4" class="ui-state-active">
73 ▼         <h3 class="ui-widget-header">
74             Resize contained to this container
75         </h3>
76     </div>
77 </div>
78 </body>
```



### Le plugin Resizable

```
24 .square {
25     width: 150px;
26     height: 150px;
27     border: 1px solid black;
28     text-align: center;
29     float: left;
30     margin-left: 20px;
31     right: 20px;
32 }
33
34 </style>
35 <!-- Javascript -->
36 <script>
37     $(function() {
38         $("#resizable-5").resizable({
39             delay: 1000
40         });
41         $("#resizable-6").resizable({
42             distance: 40
43         });
44         $("#resizable-7").resizable({
45             autoHide: true
46         });
47     });
48 </script>
49 </head>
50
51 <body>
52 <div id="resizable-5" class="square ui-widget-content">
53     <h3 class="ui-widget-header">
54         Resize starts after delay of 1000ms
55     </h3>
56 </div>
57 <br>
58 <div id="resizable-6" class="square ui-widget-content">
59     <h3 class="ui-widget-header">
60         Resize starts at distance of 40px
61     </h3>
62 </div>
63 <br>
64 <div id="resizable-7" class="square ui-widget-content">
65     <h3 class="ui-widget-header">
```

127.0.0.1:22682/test2.html

Resize starts after  
delay of 1000ms

Resize starts at  
distance of 40px

Hover over me to  
see the  
magnification  
icon!

127.0.0.1:22682/test2.html

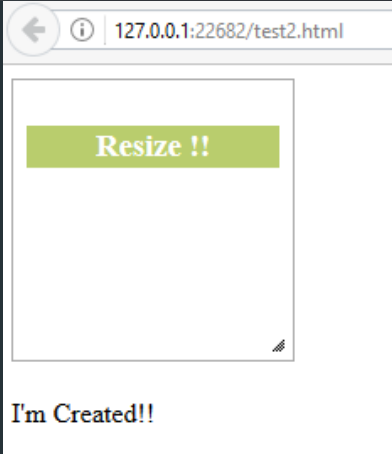
Resize starts after  
delay of 1000ms

Resize starts at  
distance of 40px

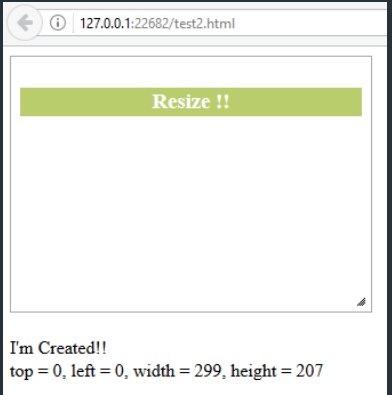
Hover over me to see the magnification  
icon!

### Le plugin Resizable

```
24 ▼ #resizable-14 {
25     width: 150px;
26     height: 150px;
27     padding: 0.5em;
28     text-align: center;
29     margin: 0;
30 }
31
32 </style>
33 <!-- Javascript -->
34 <script>
35     $(function() {
36         $("#resizable-14").resizable({
37             create: function(event, ui) {
38                 $("#resizable-15").text("I'm Created!!");
39             },
40             resize: function(event, ui) {
41                 $("#resizable-16").text("top = " + ui.position.top + ", left = " + ui.position.left + ", width = " +
42                     ui.size.width + ", height = " + ui.size.height);
43             });
44         });
45     </script>
46 </head>
47
48 <body>
49     <!-- HTML -->
50     <div id="resizable-14" class="ui-widget-content">
51         <h3 class="ui-widget-header">Resize !!</h3>
52     </div>
53     <br> <span id="resizable-15"></span>
54     <br> <span id="resizable-16"></span> </body>
55 </html>
```



I'm Created!!



I'm Created!!  
top = 0, left = 0, width = 299, height = 207

### Le plugin Selectable

- La sélection est une action utilisée pratiquement partout : sur votre système d'exploitation, dans les logiciels de retouche graphique, dans certaines applications, et maintenant sur les sites web.

```
12 <style>
13   .ui-selecting {
14     background: magenta;
15   }
16
17   .ui-selected {
18     background: red;
19     color: #fff;
20   }
21
22 </style>
23 <script>
24   $(function() {
25     $('#selection').selectable();
26   });
27
28 </script>
29 </head>
30
31 <body>
32   <!-- HTML -->
33   <ul id="selection">
34     <li>1er élément</li>
35     <li>2ème élément</li>
36     <li>3ème élément</li>
37   </ul>
38 </body>
```

- 
- 1er élément
  - 2ème élément
  - 3ème élément

### Le plugin Selectable

```

48 <script>
49   $(function() {
50     $("#selectable-7").selectable({
51       selected: function() {
52         var result = $("#result").empty();
53         $(".ui-selected", this).each(function() {
54           var index = $("#selectable-7 li").index(this);
55           result.append(" #" + (index + 1));
56         });
57       }
58     });
59   });
60
61 </script>
62 </head>
63
64 <body>
65   <h3>Events</h3>
66   <ol id="selectable-7">
67     <li class="ui-widget-content">Product 1</li>
68     <li class="ui-widget-content">Product 2</li>
69     <li class="ui-widget-content">Product 3</li>
70     <li class="ui-widget-content">Product 4</li>
71     <li class="ui-widget-content">Product 5</li>
72     <li class="ui-widget-content">Product 6</li>
73     <li class="ui-widget-content">Product 7</li>
74   </ol>
75   <span class="resultarea">Selected Product</span>
76   <span id="result" class="resultarea"></span>
77
78 </body>

```

#### Events

Product 1

Product 2

Product 3

Product 4

Product 5

Product 6

Product 7

Selected Product

#### Events

Product 1

Product 2

Product 3

Product 4

Product 5

Product 6

Product 7

Selected Product #2

### Les dessous techniques

- AngularJS s'appuyant sur le pattern **MVC**, Model-View-Controller.
  - le **modèle** est composé des données à afficher mais ne possède aucune logique. Il s'agit simplement d'un conteneur
  - La **vue** est, quant à elle, chargée d'afficher les données du modèle pour générer la page qui sera visible par les utilisateurs. Elle a aussi pour rôle de transmettre les actions effectuées au contrôleur, comme le clic sur un bouton ou la sélection d'un élément d'une liste.
  - Le **contrôleur** est l'élément central de ce pattern puisqu'il est responsable de créer le modèle, en allant contacter une API Web ou en allant requêter une base de données par exemple, puis de le transmettre à la vue. Son rôle est également de répondre aux actions utilisations, transmises par la vue, pour effectuer le traitement attendu.

### Les dessous techniques

- AngularJS s'inspire également du pattern **MVVM**, Model-View-ViewModel
  - qui définit une organisation assez similaire au pattern MVC, pour sa notion de **binding**.
  - Le **binding** permet de synchroniser une donnée du modèle et son affichage dans la vue, de manière à ce qu'une modification du modèle déclenche automatiquement une mise à jour de la vue, et inversement.

### Les dessous techniques

- Comme l'utilisation des patterns MVC et MVVM n'est pas suffisante pour organiser une application entière,
  - ces patterns ne visant généralement que les couches UI,
  - AngularJS introduit d'autres notions,
    - comme les **services**, dont le rôle est **d'encapsuler** des fonctionnalités métiers et techniques,
    - ou comme les **modules**, permettant de regrouper des éléments pouvant être réutilisés dans plusieurs applications AngularJS.



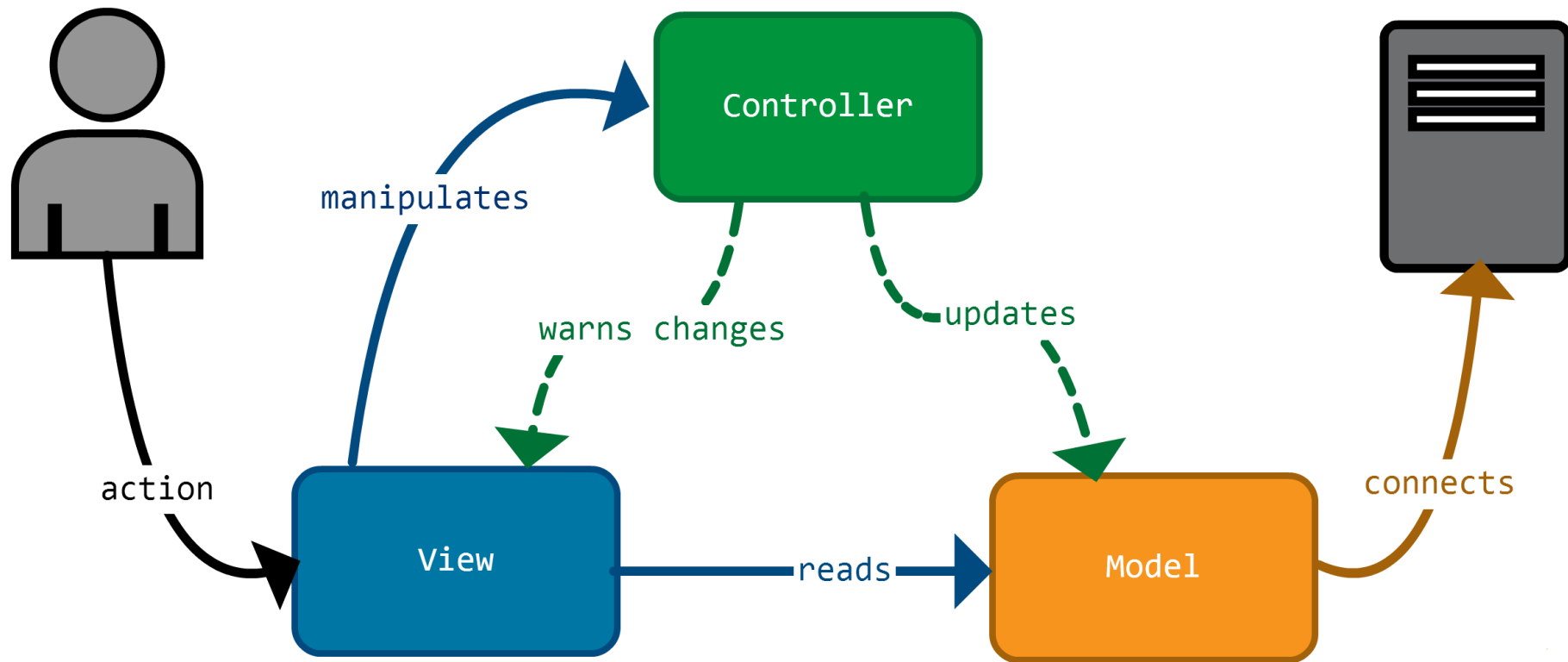
### Les dessous techniques :

- **IoC inverse of Control**: afin de décharger le développeur de la gestion du cycle de vie des différents objets AngularJS, le pattern IoC est utilisé:
  - Dans une application AngularJS, IoC permet de décharger le développeur de la gestion du cycle de vie des différents objets AngularJS
  - AngularJS se chargera automatiquement de gérer le cycle de vie des différents éléments ainsi que leurs dépendances.

### MVC

- **Modèle-Vue-Contrôleur** (abr. **MVC**) est un motif d'[architecture logicielle](#) destiné aux [interfaces graphiques](#) lancé en 1978 et très populaire pour les [applications web](#).
- Le motif est composé de trois types de modules ayant trois responsabilités différentes: les modèles, les vues et les contrôleurs.
  - Un modèle contient les données à afficher.
  - Une vue contient la présentation de l'interface graphique.
  - Un contrôleur contient la logique concernant les actions effectuées par l'utilisateur.
- Ce motif est utilisé par de nombreux [frameworks](#) pour [applications web](#) tels que [Ruby on Rails](#), [Django](#), [ASP.NET MVC](#), [Spring](#), [Struts](#), [Symfony](#), [Apache Tapestry](#) ou [Angular Js](#)

### MVC

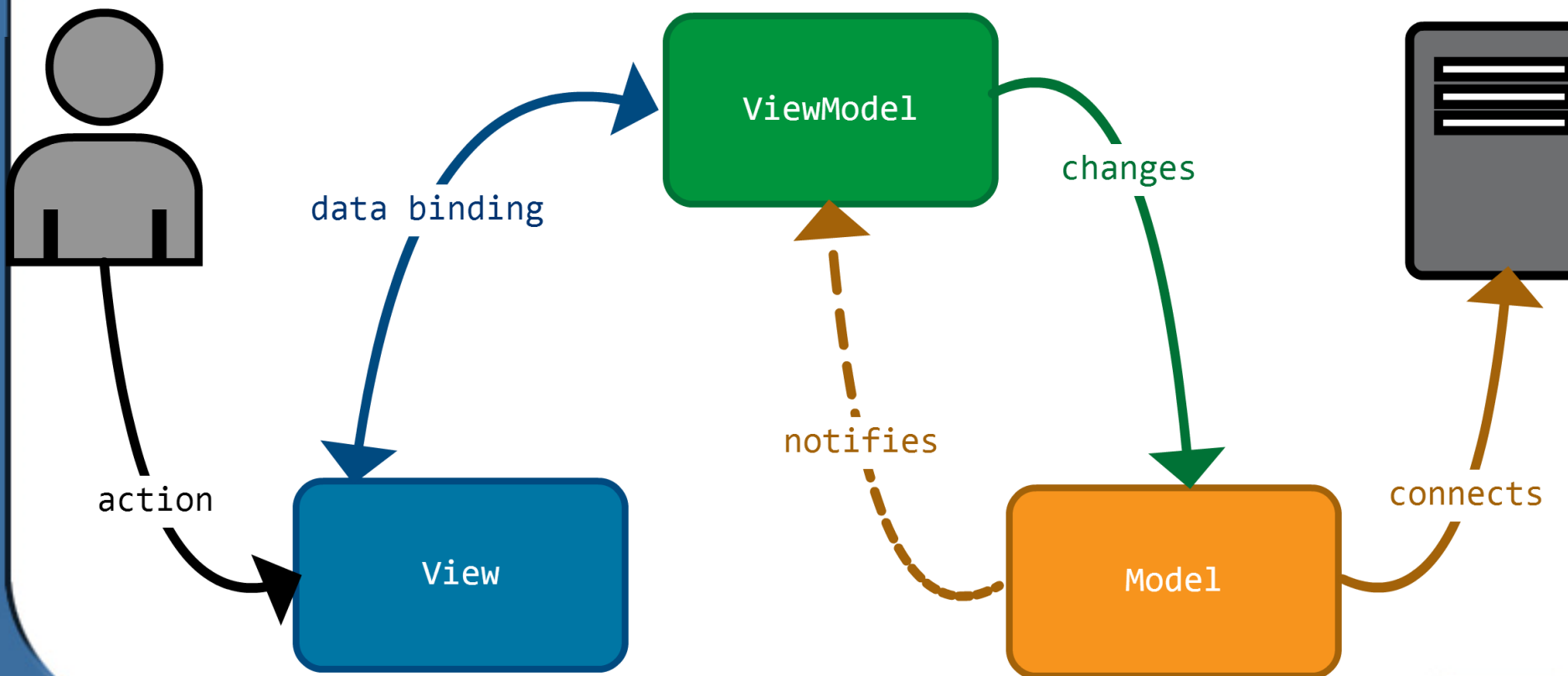


### MVC

- Le **Modèle** définit la structure des données et communique avec le serveur.
- La **Vue** affiche les informations du Modèle et reçoit les actions de l'utilisateur.
- Le **Contrôleur** gère les événements et la mise à jour de la Vue et du Modèle.

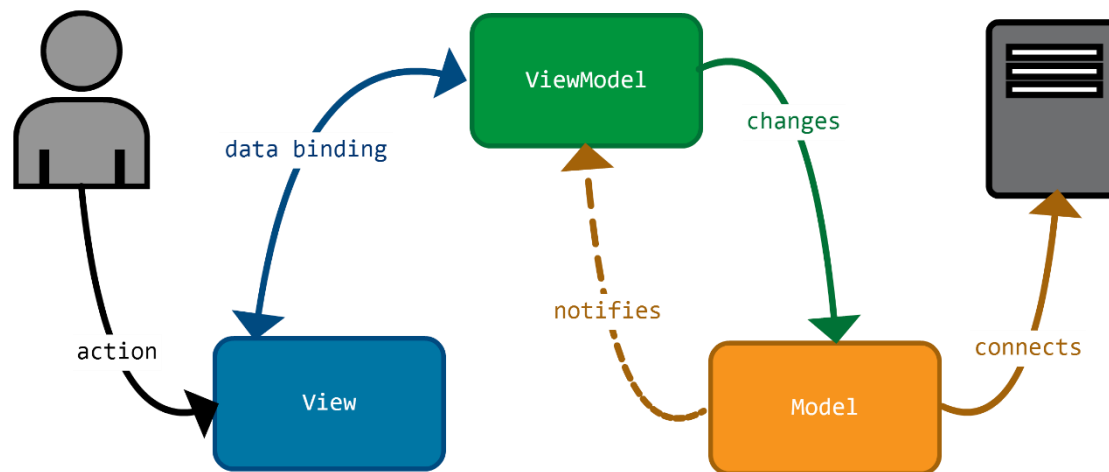
### MVVM

- MVVM est l'acronyme de : Model-View-ViewModel.

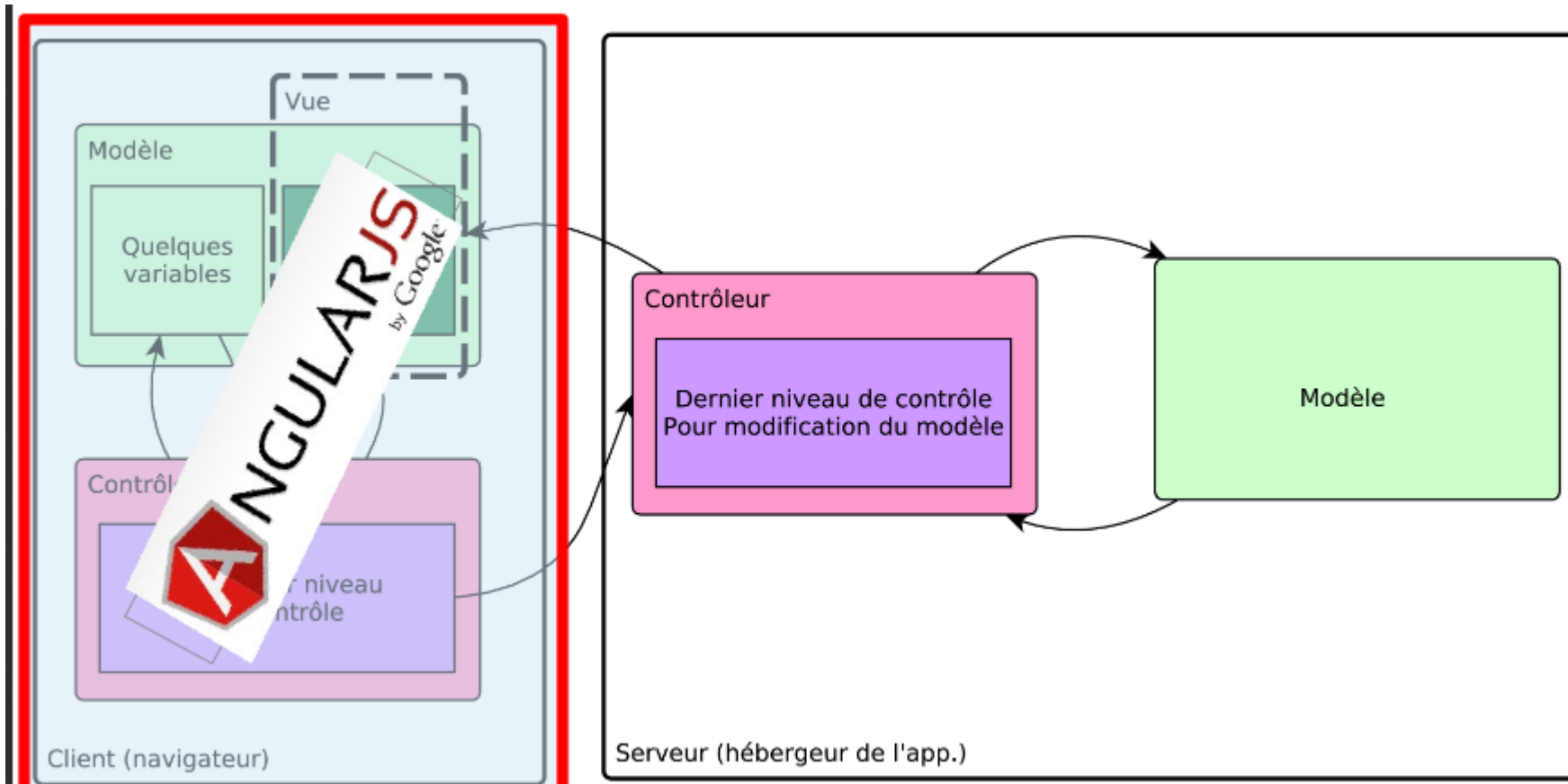


### MVVM

- La **Vue** reçoit toujours les actions de l'utilisateur et interagit seulement avec le **ViewModel**.
- Le **Modèle** communique avec le serveur et notifie le **ViewModel** de son changement.
- Le **ViewModel** s'occupe de :
  - présenter les données du Model à la Vue,
  - recevoir les changements de données de la Vue,
  - demander au Model de se modifier.

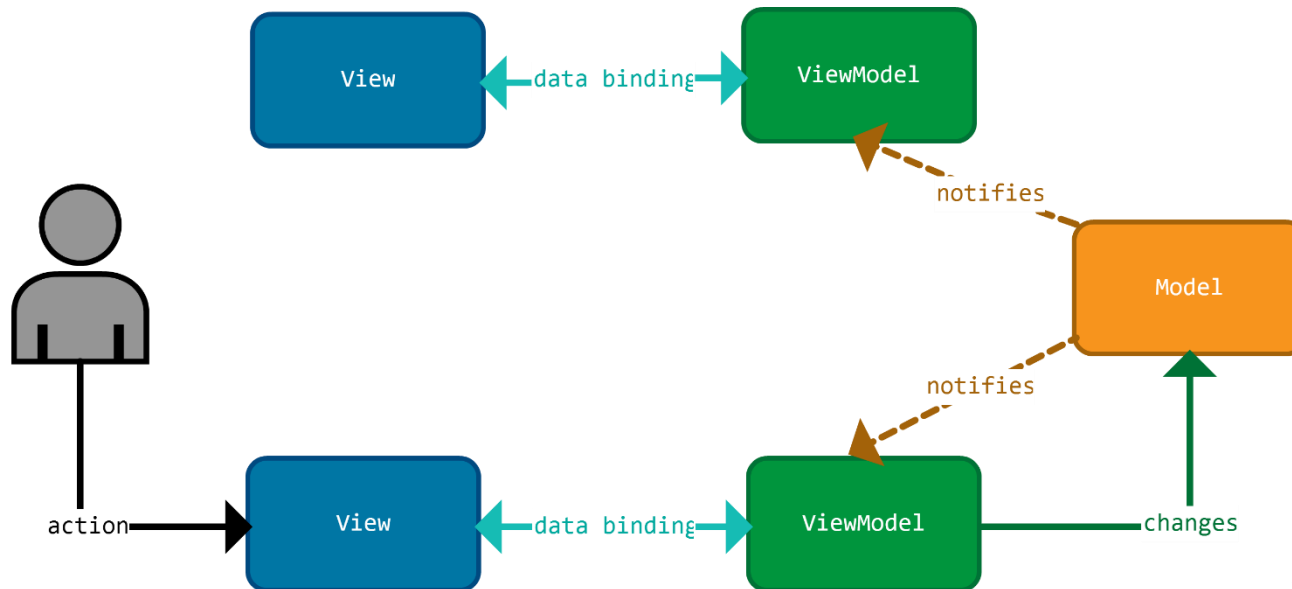


### Où intervient Angular ?



### Data Binding

- La Vue n'a donc plus aucun lien avec le Model.
- Le ViewModel s'occupe entièrement du cycle de modification de ce dernier.
  - Il réalise à la fois la réception et l'envoi des données à la Vue. On parle alors de « data binding ». Les informations affichées sont liées entre deux entités et mises à jour en temps réel.



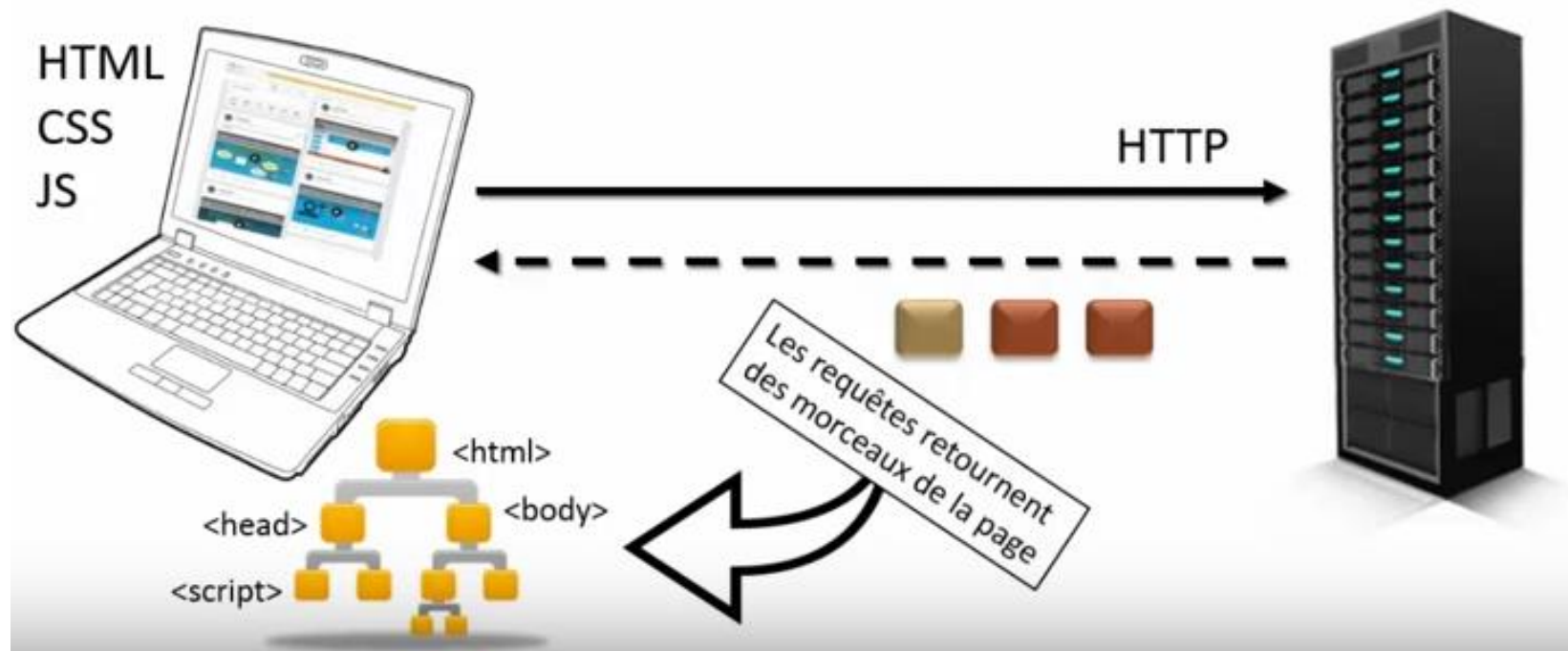


### Mettre en place un projet AngularJS (l'essentiel)

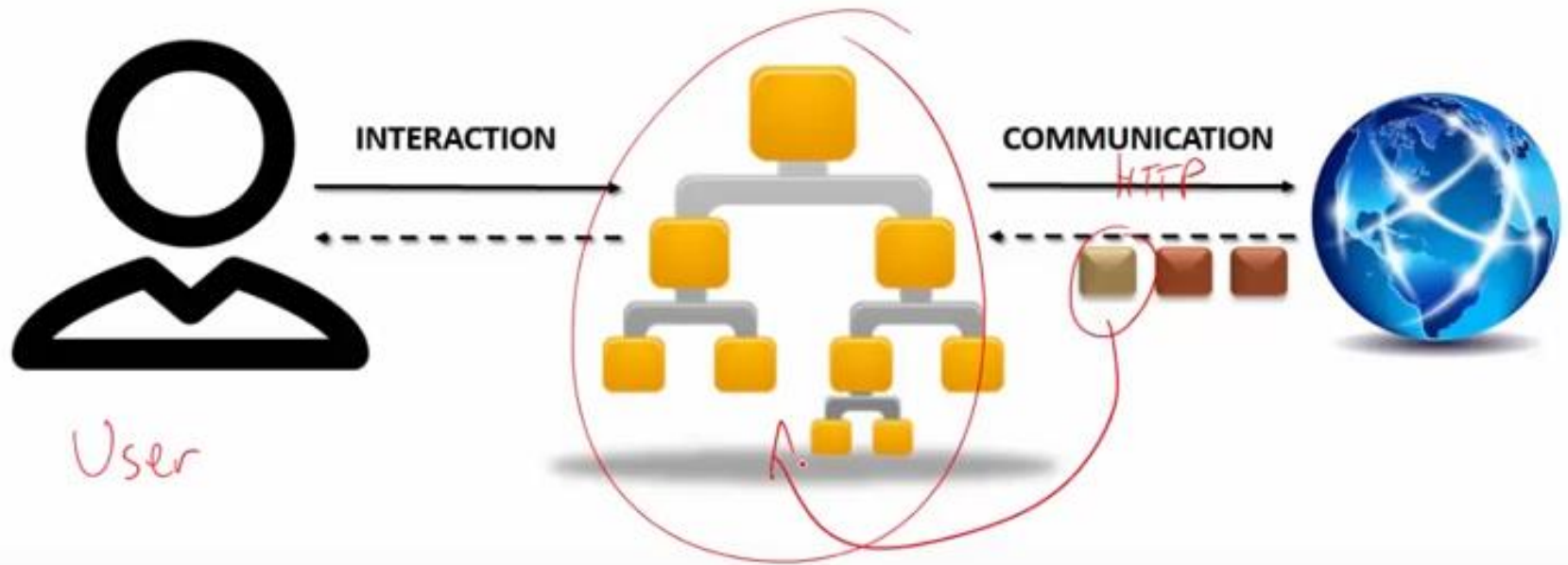
Tout d'abord, nous devons mettre en place le minimum vital d'un projet Angular. Nous devons mettre en place certaines choses avant de commencer. Cela revient, en général

- à ajouter une déclaration **ng-app**,
- écrire un **contrôleur** pour parler à la vue
- puis l'inclusion d'Angular et un attachement au DOM.

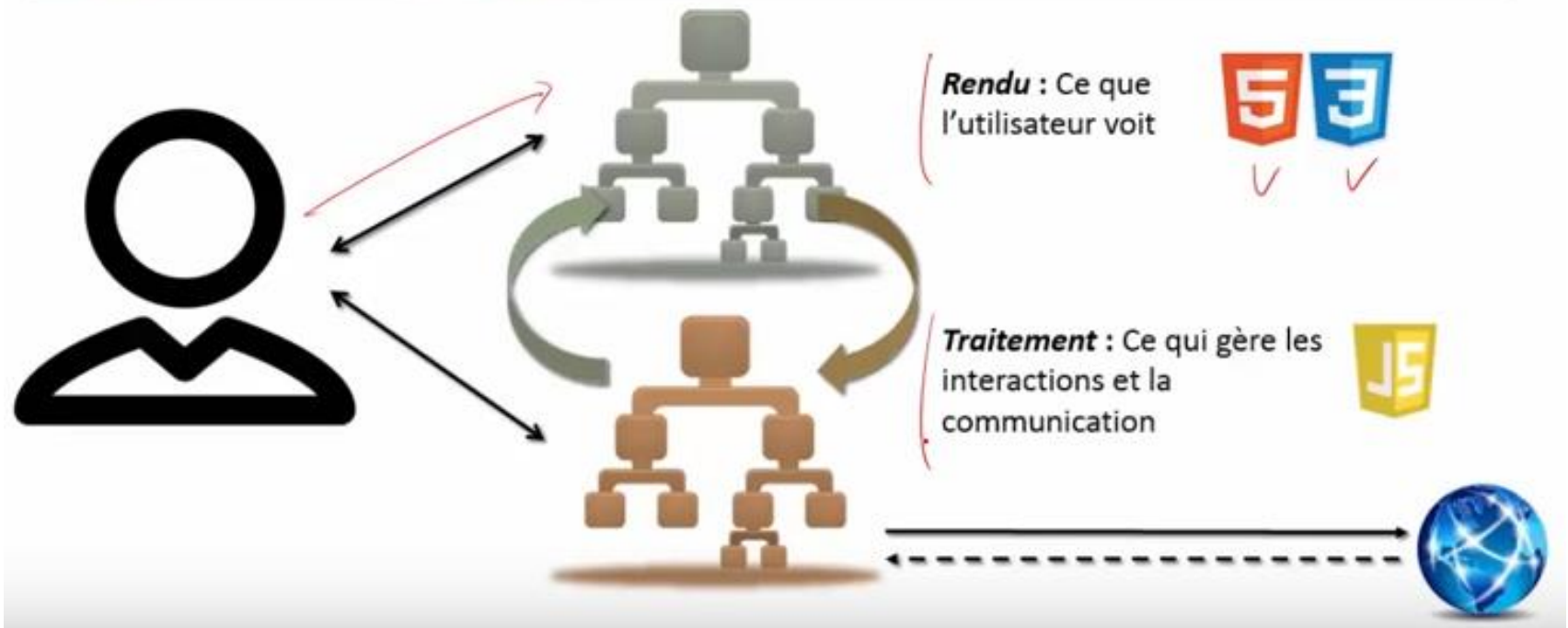
### Architecture mono-page



### A quoi sert un framework



### Rendu et Traitement



### Concepts

- Le rendu est écrit en HTML (**View**)
- Le rendu contient les templates qui sont écrits avec des **Expressions** ou des **Directives**.
- Les **templates** sont compilés après le chargement de la View. C'est là que les liens sont faits avec le traitement
- Le traitement est écrit en JS
- Le traitement est décomposé en plusieurs entités (**Controller**, **Service**).
- Ces entités sont regroupées en **modules** et sont chargées dynamiquement lors de la compilation de la **View** (accessibles).

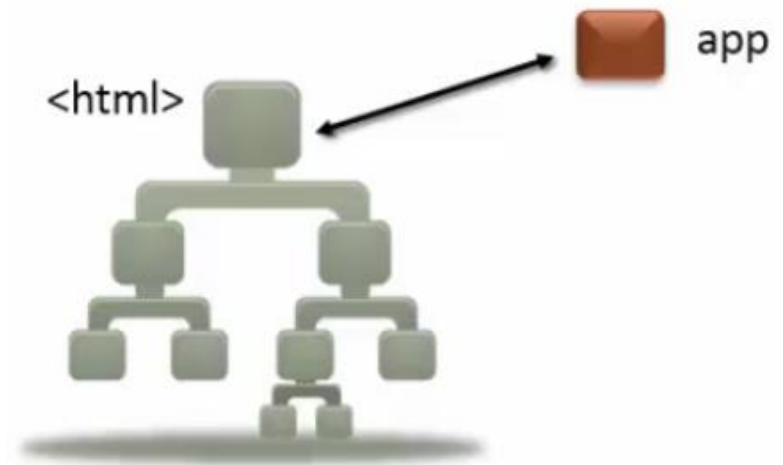


### Concepts

- **Le minimum**
- Une **view** sans *template* qui
  - Qui charge angular.js et le script de l'application
  - Se branche au module principal (ng-app)
- Un **module** principal qui se déclare à Angular

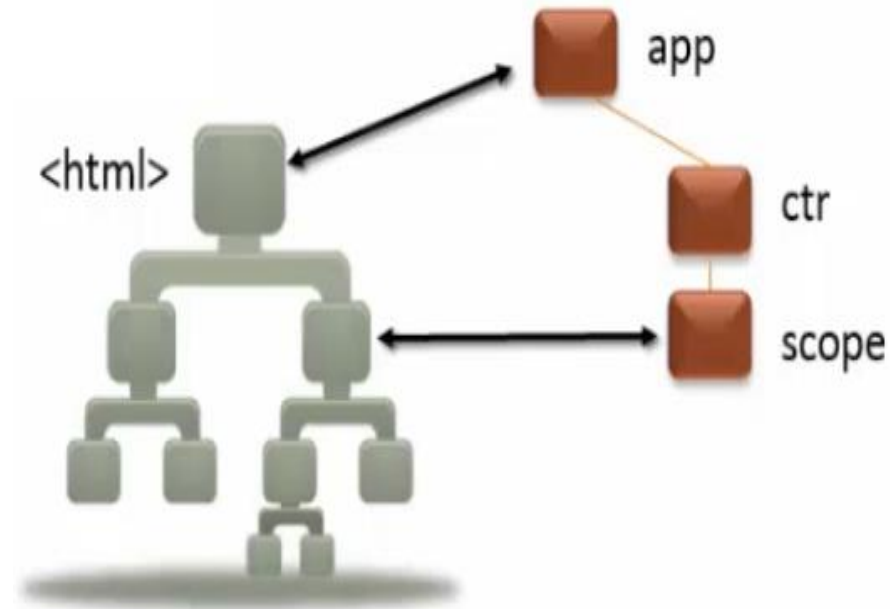
```
1 var app = angular.module('monApp', [])  
2
```

```
1 <!DOCTYPE html>  
2 <html ng-app="monApp">  
3 <head>  
4   <title>Ma première application AngularJS</title>  
5   <meta charset="UTF-8">  
6 </head>  
7 <body>  
8   Pour l'instant rien mais tout est là !  
9 </body>  
10 <script src="https://ajax.googleapis.com/ajax/libs/  
    angularjs/1.3.14/angular.min.js"></script>  
11 <script src="./monAngular.js"></script>  
12 </html>  
13
```



### Concepts

- Les **controllers**
  - Éléments de base de la couche de traitement de Angular
  - Un controller est un objet JS qui sera associé à la vue (DOM)
- En fait c'est le scope (objet JS qui contient des données et traitements) qui est lié à la View
- Les templates de la **View** pourront exploiter ce scope.





### Concepts

- La directive **ng-controller** permet de définir un lien entre la **View** et le controller.
- Grâce à cette **directive**, un controller sera instancié lors de la compilation de la View.
- Le controller aura alors son **scope**, ce scope sera lié à la balise de **View**.

```
6 </head>
7 <body>
8   <div ng-controller='monController'>
9   </div>
10 </body>
```



### Concepts

- Les **expressions** sont des templates les plus basiques `{{expr}}`.
- Elles contiennent du code JS qui peut directement utilisé le scope du controller.
- Ce code sera exécuté après la compilation, et à tout moment dès que le **scope changera**.

```
7 <body>
8   <div ng-controller='monController'>
9     La valeur stockée dans le scope est : {{maData}}
10  </div>
```

### Concepts

- La directive **ng-model**
  - C'est template qui se met sur les éléments input, textarea ou select
  - Elle effectue le lien entre un élément de la View et une propriété du scope
  - Si l'élément change, le scope change et si le scope change l'élément change.

```
8 <div ng-controller='monController'>
9   La valeur stockée dans le scope est : {{maData}}
10  <input ng-model="maData">
11 </div>
```



### Mettre en place un projet AngularJS (l'essentiel)

Un peu de HTML avec les déclarations ng-\*

```
<div ng-app="myApp">  
  <div ng-controller="MainCtrl">  
    <!-- logique du contrôleur -->  
  </div>  
</div>
```

Un module Angular et un contrôleur :

```
var myApp = angular.module('myApp', []);  
  
myApp.controller('MainCtrl', ['$scope', function ($scope) {  
  // Magie du contrôleur  
}]);
```

### Mettre en place un projet AngularJS (l'essentiel)

- nous devons créer un module Angular dans lequel nous allons placer toute notre logique.

```
var myApp = angular.module('myApp', []);  
myApp.controller('MainCtrl', ['$scope', function ($scope) {...}]);  
myApp.controller('NavCtrl', ['$scope', function ($scope) {...}]);  
myApp.controller('UserCtrl', ['$scope', function ($scope) {...}]);
```

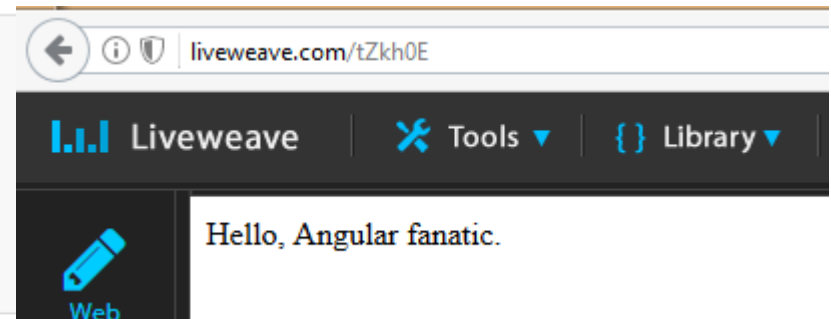
### Mettre en place un projet AngularJS (l'essentiel)

- Comment pousser des données dans le DOM depuis un contrôleur ?
  - **Angular** utilise un système de **template** qui ressemble à ceci pour parler à HTML : `{{ handlebars }}`.
  - Idéalement, la page HTML ne contient aucun texte ou valeur en dur, cela permet de tirer un maximum d'Angular.

### Mettre en place un projet AngularJS (l'essentiel)

- Voici un exemple dans lequel nous poussons une chaîne de caractères dans le DOM :

```
<div ng-app="myApp">
  <div ng-controller="MainCtrl">
    {{ text }}
  </div>
</div>
```



```
var myApp = angular.module('myApp', []);

myApp.controller('MainCtrl', ['$scope', function ($scope) {

    $scope.text = 'Hello, Angular fanatic.';

}]);
```

### \$scope

- Le concept clé ici est **\$scope** que vous passez à toutes vos fonctions au sein d'un contrôleur.
  - \$scope fait référence à l'élément courant (ou zone courante) dans le DOM (ce qui est différent de this).
  - c'est le canal de communication avec le DOM depuis le serveur (ou depuis les données statiques si vous en avez).
  - La démo donne une petite idée de comment "pousser" des données dans le DOM.

### Mettre en place un projet AngularJS (l'essentiel)

- Regardons maintenant une structure de données plus représentative:
  - que nous avons *hypothétiquement* récupérée depuis le serveur pour afficher les détails de l'utilisateur.

```
var myApp = angular.module('myApp', []);

myApp.controller('UserCtrl', ['$scope', function ($scope) {

    // Créons un namespace pour les détails de l'utilisateur
    // Également utile pour une aide visuelle dans le DOM
    $scope.user = {};
    $scope.user.details = {
        "username": "Todd Motto",
        "id": "89101112"
    };

}]);
```



### Mettre en place un projet AngularJS (l'essentiel)

- Poussons ces données vers le DOM pour les afficher :

```
<div ng-app="myApp">
  <div ng-controller="UserCtrl">
    <p class="username">Welcome, {{ user.details.username }}</p>
    <p class="id">User ID: {{ user.details.id }}</p>
  </div>
</div>
```

Welcome, Todd Motto

User ID: 89101112

### Les directives

- Une directive, dans sa forme la plus simple, est un petit morceau de template HTML, utilisé de préférence à plusieurs endroits de l'application.
  - C'est un moyen facile d'injecter sans effort du DOM dans votre application ou d'effectuer des interactions particulières avec le DOM

### Philosophie générale : faciliter la vie du programmeur

- Simplifie au maximum la programmation de la vue
  - Éviter la répétition de code
    - Angular vous force à modulariser le code
- Éviter la maîtrise du DOM ou des technos comme Ajax
  - Angular le fait pour vous
- Éviter l'aspect verbeux et illisible de HTML
  - Angular vous permet de définir et nommer des comportements HTML
- **Bref, rendre le code concis et lisible**