

INTERFACES WEB AVANCEES - 2

3C – 2015/2016

Agenda

Session 1 Les bases

- Introduction
- CSS
- HTML
- Javascript

Session 2 La productivité

- Frameworks
- Outils de developpement

Session 3 Allons plus loin

- ECMAScript 6 (Object.observe, Quick fonction, Promise)
- Introduction à d'autres langages
- Web component

<https://iwa2015.herokuapp.com/>

Slides, exemples de cours, démo, sujet du TP

RAPPEL

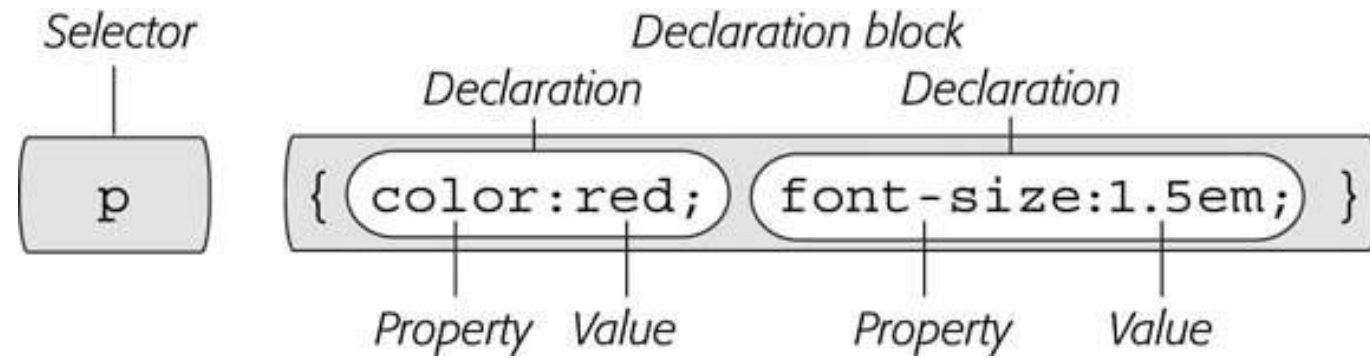
HTML – CSS – JAVASCRIPT

HTML

- **Structure** le contenu d'une page
- Langage composé de balise. <BALISE> ... </BALISE> ou <BALISE ... />
- <html> indique au navigateur que l'on va utiliser de l'HTML
- Les informations pour le navigateur se trouve entre les balises <head>, et le contenu qui sera afficher au visiteur son dans les balises <body>
- **Best Practice :**
 - Mettre les <script> en fin de page pour accélérer le chargement
 - Utiliser les balises pour leur sens sémantiques (<footer>, <header>, <section>, .)



CSS



- Pour **ajouter un style** au HTML
- Pour lier le HTML et le CSS, on importe le fichier en mettant dans les balises <head> </head> :
`<link rel="stylesheet" type="text/css" href="css/global.css" />`
- On lie un élément HTML avec le CSS grâce au **sélecteur**, #ID, .CLASSE, ou directement le tagName (« a » pour un lien, « div » pour un block, « img » pour une image)
- Le style directement appliqué gagne en cas de conflit
- Les balises peuvent **hériter** des propriétés CSS de leur parent

JavaScript

- Langage utilisé pour rendre **dynamique** les pages HTML
- On déclare une variable avec le **mot clé « var »**
- Les instructions se finissent par « ; »
- Le mot clé **« this »** fait référence à l'objet courant.
- Il utilise des **prototypes** et non des classes => Une fonction faisant office de **constructeur**, et des prototypes comme **méthodes**.
- **Remarques:**
 - Utiliser des noms de variables indiquant ce qu'elles **contiennent** (var ListDesEleves = [];)
 - Attention aux fonctions setTimeout et setInterval qui modifie la portée de « this »

LIVE CODING

CALCULATRICE – 30MIN

Frameworks

Introduction

Framework

« Un *framework* est un ensemble d'**outils** et de **composants logiciels** organisés conformément à un **plan d'architecture** et des *patterns*, l'ensemble formant ou promouvant un « squelette » de programme. Il est souvent fourni sous la forme d'une bibliothèque logicielle, et accompagné du plan de l'architecture cible du *framework* »

« Un *framework* est conçu en vue d'aider les programmeurs dans leur travail. L'organisation du *framework* vise la productivité maximale du programmeur qui va l'utiliser — gage de baisse des coûts de construction et maintenance du programme. Le contenu exact du *framework* est dicté par le type de programme et l'architecture cible pour lequel il est conçu. »

Framework Front



ETC ...

Bootstrap

FRAMEWORK CSS

Twitter Bootstrap !

Bootstrap

Sleek, intuitive, and powerful front-end framework for faster and easier web development.

<http://getbootstrap.com/>

- Développé par Mark Otto and Jacob Thornton (Twitter)

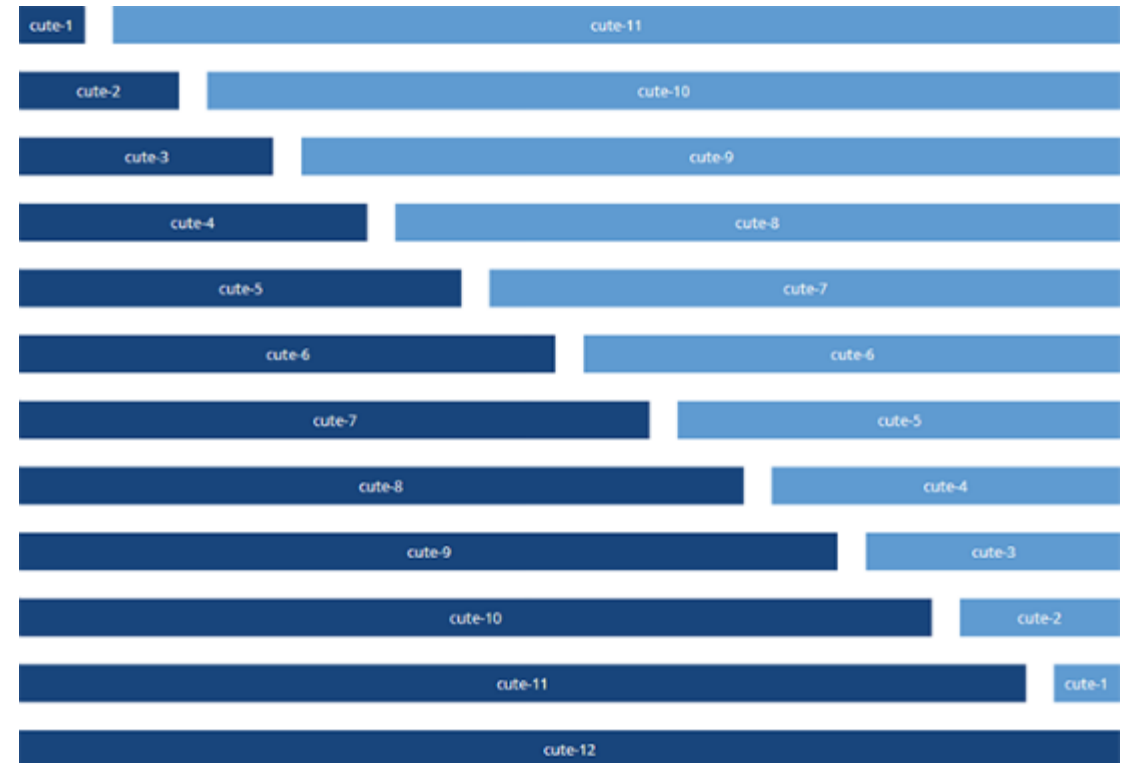
Key features :

- Collection de templates HTML et CSS (typo, boutons, formulaire, ...)
- Composant d'interface (modal, menus, header / footer, ...)
- Responsive design (browser, tablette, smartphone, ...)
- Open source (... et projet le plus populaire sur GitHub !)

Responsive design

- « Le Responsive Web design est une approche de conception Web qui vise à l'élaboration de sites offrant une expérience de lecture et de navigation optimales pour l'utilisateur quelle que soit sa gamme d'appareil (téléphones mobiles, tablettes, liseuses, moniteurs d'ordinateur de bureau). »
- Structure le CSS sous forme d'une grille qui s'adapte à la taille des écrans en cachant, déplaçant des éléments.

Bootstrap est responsive



Utiliser Bootstrap !

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <link href="css/bootstrap.min.css" rel="stylesheet" media="screen">
  <link href="css/styles.css" rel="stylesheet" media="screen">
</head>
<body>

  <!-- JavaScript placed at the end of the document so the pages load faster -->
  <script src="js/jquery-1.8.3.min.js"></script>
  <script src="js/bootstrap.min.js"></script>
  <script src="js/scripts.js"></script>
</body>
</html>
```

Link	Button	Input	Submit
------	--------	-------	--------

```
<a class="btn btn-default" href="#" role="button">Link</a>
<button class="btn btn-default" type="submit">Button</button>
<input class="btn btn-default" type="button" value="Input">
<input class="btn btn-default" type="submit" value="Submit">
```

<http://getbootstrap.com>



Introduction

- Librairie **JavaScript**
- Développé par John Resig
- **Open source** (MIT, GPL)
- Publier au BarCamp de NYC en Jan. 2006
- jQuery v1.0 en Aou. 2006
- Versions actuelles 1.11.3 et 2.1.4
- 31KB, Minifié et Gzippé
- Compatible IE 6+, Firefox, Safari 2+, Opera 9+ et Chrome

Avantages

- Améliore les **interactions** entre HTML et JavaScript
 - Permet de manipuler le DOM (selectors CSS)
 - Gère les évènements, les effets et Ajax
 - Validation de formulaire, widgets
- Code **efficace** et **peu volumineux**
- Ajout d'autres fonctionnalités via des **plugins**
- Exemple :

```
$("table tr:nth-child(even)").addClass("striped");
```

Year	Make	Model
1965	Ford	Mustang
1970	Toyota	Corolla
1979	AMC	Jeep CJ-5
1983	Ford	EXP 356x256
1985	Dodge	Daytona
1990	Chrysler	Jeep Wrangler Sahara
1995	Ford	Ranger
1997	Chrysler	Jeep Wrangler Sahara
2000	Chrysler	Jeep Wrangler Sahara
2005	Chrysler	Jeep Wrangler Unlimited
2007	Dodge	Caliber R/T

JavaScript & jQuery

- Permet d'éviter l'introduction de **comportement** (ie. JavaScript) dans la **structure** HTML
- Mauvais exemple :

```
<a href="#" onclick="showPopup('image.jpg'); return false;">see the image</a>
```

- Exemple correctement refactoré :

```
...
<head>
  <script type="text/javascript" src="jquery.js"/>
  <script type="text/javascript" src="behaviors.js"/>
  ...
</head>
<body>
  <a href="image.jpg" id="foobar">see the image</a>
  ...

// in behaviors.js
$(document).ready(function(){
  $('#foobar').click('showPopup('image.jpg')');
});
```

jQuery basics

- Récupérer des groupes d'éléments :

```
$(selector)  
// ou  
jQuery(selector)
```

- La fonction `$()` retourne un **objet** contenant un **tableau d'éléments du DOM** (ou collection) qui vérifie le sélecteur.
- Existence de méthodes permettant de modifier tous les éléments (pas besoin de boucle JavaScript !)
- Exemple :

```
$("div.notLongForThisWorld").fadeOut();
```

jQuery basics

- On peut **chainer les méthodes**
 - La plupart des fonctions jQuery retournent un objet (collection), généralement la même que celui passé en entrée

```
$("div.notLongForThisWorld")  
  .fadeOut()  
  .addClass("removed");
```

- Accès aux éléments d'une collection

```
$("#someElement")[0].innerHTML = "Blabla";
```

- jQuery utilise des sélecteurs (CSS + sélecteurs « maison »)
 - Exemple : sélection de tous les éléments <P> pairs

```
$("p:even");
```

Chainage complexe, exemple

```
$('form#login')
  // Cache tous les label du formulaire avec la class optional
  .find('label.optional').hide().end()
  // Ajoute une bordure rouge à tous les champs password
  .find('input:password').css('border', '1px solid red').end()
  // Ajoute un event handler submit
  .submit(function(){
    return confirm('Are you sure you want to submit?')
  });

// Recupere toutes les cellules qui contiennent henry
$('td:contains("Henry")')
  // Recupere son parent
  .parent()
  // Recupere la 2e cellule de ce parent
  .find('td:eq(1)')
  // Ajoute la class highlight a cette cellule
  .addClass('highlight')
  // Retour au parent de la cellule contenant Henry
  .end()
  // Recupere la 3e cellule de ce parent
  .find('td:eq(2)')
  // Ajoute la class highlight a cette cellule
  .addClass('highlight');
```

jQuery basics : sélecteurs

```
// This selector selects the first row of each table
$("tr:nth-child(1)");
// This selector selects direct <div> children of <body>
$("body > div");
// This selector selects links to PDF files
$("a[href$=pdf]");
// This selector selects direct <div> children of
// <body>-containing links
$("body > div:has(a)")
// Finds the elements that match any of these three selectors.
// Set its border with css
$("div,span,p.myClass").css("border","3px solid red");
// Finds all input checkboxes that are not checked and
// highlights the next sibling span
$("input:not(:checked) + span").css("background-color", "yellow");
// Finds all inputs that have an id attribute and whose name
// attribute ends with man and sets the value
$("input[id][name$='man']").val("only this one");
// Disable all input with class special in the form with id myForm
$("form#myForm input.special").disable();
```


jQuery basics : sélecteurs

Sélecteurs CSS

Expression	Retour
*	Toutes les balises.
elem	Les balises elem .
#id	Balise ayant l'id "id".
.class	Balises ayant la classe "class".
elem[attr]	Balises elem dont l'attribut "attr" est spécifié.
elem[attr="val"]	Balises elem dont l'attribut "attr" est à la valeur val .
elem bal	Balises bal contenues dans une balise elem .
elem > bal	Balises bal directement descendantes de balises elem .
elem + bal	Balises bal immédiatement précédées d'une balise elem .
elem ~ bal	Balises bal précédées d'une balise elem .

Sélecteurs custom

Expression	Retour
:hidden	Éléments invisibles, cachés.
:visible	Éléments visibles.
:parent	Éléments qui ont des éléments enfants.
:header	Balises de titres : h1, h2, h3, h4, h5 et h6.
:not(s)	Éléments qui ne sont pas sélectionnés par le sélecteur s .
:has(s)	Éléments qui contiennent des éléments sélectionnés par le sélecteur s .
:contains(t)	Éléments qui contiennent du texte t .
:empty	Éléments dont le contenu est vide.
:eq(n) et :nth(n)	Le n-ième élément, en partant de zéro.
:gt(n) (greater than, signifiant plus grand que)	Éléments dont le numéro (on dit l'« index ») est plus grand que n .
:lt(n) (less than, signifiant plus petit que)	Éléments dont l'index est plus petit que n .
:first	Le premier élément (équivalent à :eq(0)).
:last	Le dernier élément.
:even (pair)	Éléments dont l'index est pair.
:odd (impair)	Éléments dont l'index est impair.

Utiliser jQuery

- Importation de la librairie jQuery

- Depuis un fichier

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

- Depuis le site de jQuery

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

- Point de départ avec jQuery (fichier JS)

```
$(document).ready(function(){  
    $("table tr:nth-child(even)").addClass("even");  
    // your stuff goes here  
}); // shortcut $(function(){});
```

Ce code est exécuté après le chargement du DOM, mais avant le chargement de ressources comme les images.

La fonction `$()` ou `jQuery()`

- jQuery core, une factory pour des objets jQuery
- Fourni une instance jQuery
- Toutes les opérations sont faites à partir de cette fonction
- Exemple : ce qu'on faisait avant...

```
<a href="#" onclick="toggle_visibility('foo');">Click here to toggle visibility of #foo</a>
```

```
function toggle_visibility(id) {  
    var e = document.getElementById(id);  
    if (e.style.display == 'block') e.style.display = 'none';  
    else e.style.display = 'block';  
}
```

- Avec jQuery

```
$(document).ready(function(){  
    $("a").click(function(){  
        $("#more").toggle("slow");  
        return false;  
    });  
});
```

La fonction `$()` ou `jQuery()`

- Créer des éléments DOM à la volée, exemples :

```
<script type="text/javascript">
$(function(){
    $("<p>Hi there!</p>").insertAfter("#followMe");
});
</script>
...
<body>
    <p id="followMe">Follow me!</p>
</body>
```

```
$('#someDiv').html('There are '+$('a').size()+' link(s) on this page.');
```

```
$("<div class='foo'>I have foo!</div><div>I don't</div>")
    .filter(".foo").click(function(){
        alert("I'm foo!");
    }).end().appendTo("#someParentDiv");
```

- On crée deux éléments `<DIV>`, un des deux ayant pour class `foo`.
- On filtre sur l'élément ayant la class `foo` puis on lui associe un event handler qui affichera une alerte au click
- Finalement, on utilise la méthode `end()` qui retire le filtre puis on injecte les éléments dans le DOM

jQuery : quelques exemples

```
$('#main div').addClass('test');
```

```
$('#main').slideDown('slow');
```

```
$('ul > li').click(function(){  
    $(this).find('ul').toggle();  
});
```

```
$('#contents').load('doc.html');
```

```
$('ul#some-ul > li')
```

```
// Nb d'élément retourné par le sélecteur  
$('div.section').size()
```

```
$('div.section').each(function(div) {  
    // On peut manipuler le div ici...  
});
```

Manipuler les collections

- La plupart des méthodes jQuery **opèrent sur tous les éléments** présent dans la collection (récupérés via un sélecteur)

```
// Ajouter une class à div.section
$('div.section').addClass('highlighted')
// Ajouter un attribut à img.photo
$('img.photo').attr('src', '/default.png');
// Affecter au conteneur a.foo un élément HTML
$('a.foo').html('<em>Click me now!</em>');
// Affecter un style à tous les p pairs
$('p:odd').css('background-color', '#ccc');
```

- Certaines méthodes n'opèrent que sur le **premier élément** d'une collection

```
var height = $('div#intro').height();
var src = $('img.photo').attr('src');
var lastP = $('p:last').html();
```

Evènements

- On peut également utiliser une multitude de méthodes permettant de **gérer des interactions** avec l'utilisateur (click, mouseover, focus, etc.)
- Voir le sujet Events dans la documentation jQuery

```
$('#foo').bind('click', function() {  
    alert($(this).text());  
});  
  
$(document).ready(function() {  
    $('#foo').bind('click', function(event) {  
        alert('The mouse cursor is at ('  
            + event.pageX + ', ' + event.pageY + ')');  
    });  
});  
  
$("td").hover(  
    // handlerIn, la methode mouseenter() est equivalente  
    function () { $(this).addClass("hover"); },  
    // handlerOut, la methode mouseleave() est equivalente  
    function () { $(this).removeClass("hover"); }  
);  
$("td").unbind('mouseenter mouseleave');
```

Parcours du DOM

- jQuery offre des **méthodes avancées** pour parcourir le **DOM** d'un document HTML

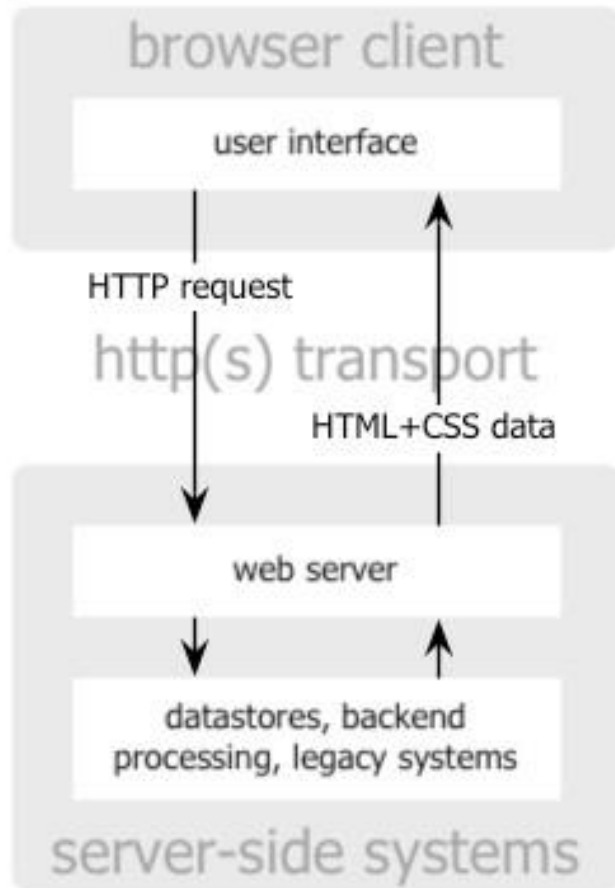
```
$('#div.section').next()
$('#div.section').prev()
$('#div.section').prev('a')
$('#div.section').parent()
$('#div.section').parents()

$('tr:odd') //all odd rows
$('tr').filter(':odd') //same effect
$('tr:even') //all even rows
$('tr').filter(':even') //same effect

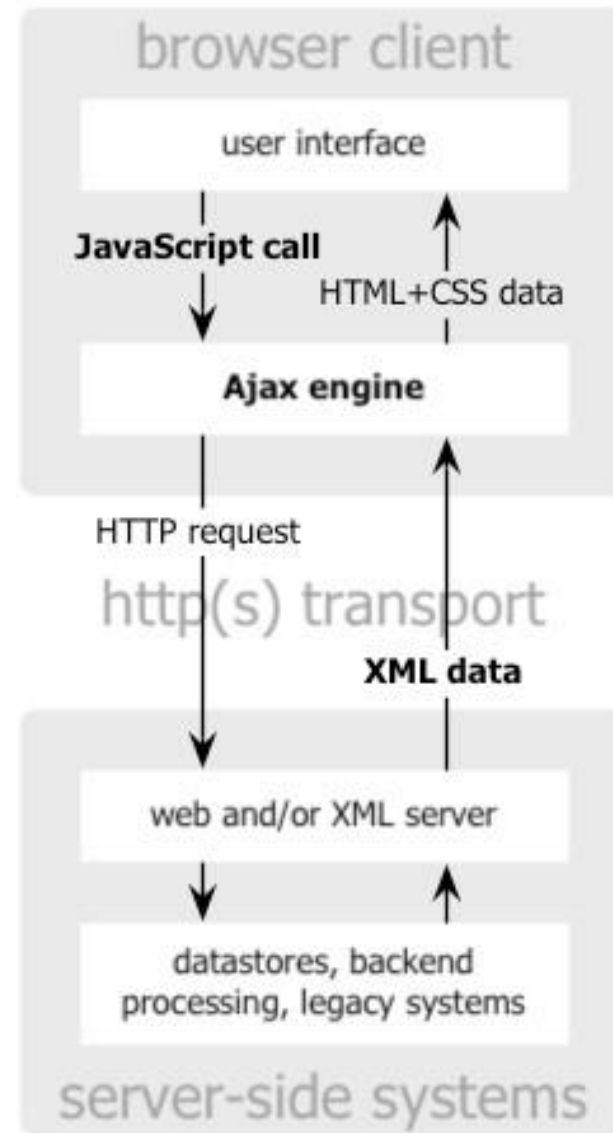
//select all even 'tr' but not that contain a 'th'
$('tr:not([th]):even')
//select all odd 'tr' but not that contain a 'th'
$('tr:not([th]):odd')
```


jQuery & AJAX

- Asynchronous Javascript and XML
- Méthodologie de conception de sites web dynamiques
- Combine plusieurs technologies (Javascript, CSS, XML, DOM)
- + XMLHttpRequest (objet JavaScript standardisé)
- Format d'échange XML ou JSON (le plus souvent)
- Permet de réaliser des requêtes asynchrones
- jQuery supporte pleinement AJAX :
 - jQuery.ajax(), jQuery.get(), jQuery.post(), ...

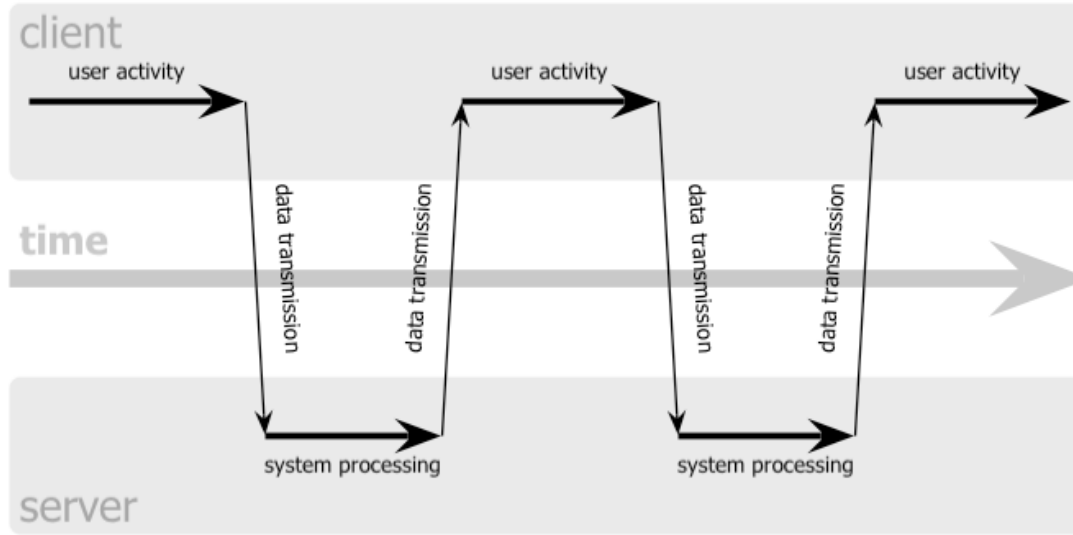


classic
web application model

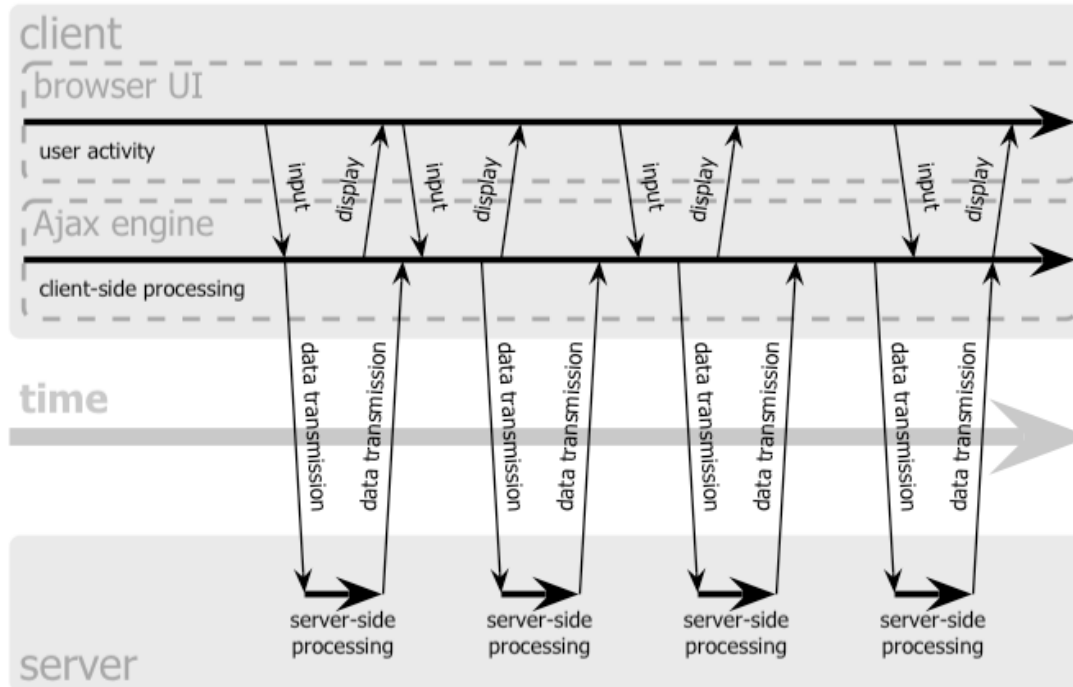


Ajax
web application model

classic web application model (synchronous)



Ajax web application model (asynchronous)



Examples AJAX

```
$('#div#intro').load('/some/file.html');  
$.get(url, params, callback)  
$.post(url, params, callback)  
$.getJSON(url, params, callback)  
$.getScript(url, callback)  
  
$.ajax({  
  url: "test.html",  
  type: "POST",  
  data: {id : menuId},  
  dataType: "json"  
  success: function(data){  
    $("#results").append(data);  
  }  
  error: function(data){  
    alert("Something was wrong");  
  }  
});
```

Pour aller plus loin

- jquery.com
 - documentation : docs.jquery.com
 - Tutoriaux : docs.jquery.com/Tutorials
 - Plugins : plugins.jquery.com
- jqueryui.com / jquerymobile.com
- A googler :
 - 15 days of jQuery (blog)
 - Learning jQuery (blog)

Fil rouge : conception d'une « TODO list » - Quatrième étape

- Utiliser AJAX et jQuery afin de persister les TODO sur un serveur



Ma TODO list !

Menu

- Afficher la TODO list d'une personne
 - Ajouter un TODO

Mes TODO

☐ Faire les courses
☒ Nourrir le chat
☐ Boire un verre chez Farid



<http://iwa2015.herokuapp.com/api/todo/>

Fil rouge : conception d'une « TODO list » - Quatrième étape


Description de l'API

- **GET – `http://DOMAINE/api/todo/{person}`**
 - *Récupère toutes les todo de la {person}*
- **POST – `http://DOMAINE/api/todo/{person}`**
 - *Ajoute une todo à la {person}*
- **POST – `http://DOMAINE/api/todo/{person}/delete`**
 - *Supprime toutes les todo de la {person}*

Cinquième étape

- Utiliser Bootstrap

Ma ToDo list !



Afficher

-


Faire les courses

-

Sortir le chien

-

Passer chez Farid prendre une bière



Ajouter

PAUSE – 15min

AngularJS

ANOTHER (AWESOME ?) FRAMEWORK JAVASCRIPT

Introduction

www.angularjs.org

- Ce n'est pas une Librairie **JavaScript** comme jQuery mais un framework
- Développé par Google en 2009
- **Open source** (MIT)
- Fondé sur l'extension de l'HTML
- Version actuelle 1.4.6 ou 2.0-Alpha
- Se **base** sur une version allégé de **jQuery**
- **MVC pattern, SPA**
- Changement de philosophie dans la communauté, passage de « Javascript c'est de la bidouille, à Javascript est industrialisable »



Avantages

- Découple les manipulations du DOM de la logique métier.
- Améliore les **interactions** entre HTML et JavaScript (HTML Extension)
- Code **efficace** et **peu volumineux**
- Ajout d'autres fonctionnalités via des **plugins**
- Facilite les mises en place de tests
- Industrilisation
- etc ...

Key Features

- Approche déclarative
- Two way **Data-Binding**
- Composant **réutilisable** (Object/Service)
- MVC/MVVM Design Pattern
- Injection de dépendance
- Test unitaire et d'intégration
- Directives / Modules / Services / Filters
- Templating
- Routing
- etc...

Structure

- **Controller** : Code métier de l'application, il est en charge de gérer l'application. Un controller peut être lié à un ou plusieurs Scope.
- **Scope** : C'est une zone du code HTML où les fonctions définies dans un contrôleur sont accessibles. Ceci permettant de découper les logiques métiers. Il est accessible en JavaScript par l'objet « \$scope »
- **Service** : En complément du Controller qui manipule les données, le service permet la constante et transparente synchronisation des variables entre le modèle et la vue. C'est un code qui permet de créer un objet qui sera commun à tous les Controller.
- **Vue** : « C'est ce que voit l'utilisateur final. Elle est **générée** à partir du **template**. Le template est le fichier HTML **enrichi** de certains attributs et balises propres à AngularJS, les **directives**. »
- **Directive** : C'est un attribut que l'on peut rajouter à une balise HTML qui est lié à une ou plusieurs fonctionnalités AngularJS. Elle commence par « ng- ». Exemple : « ng-repeat », « ng-show »

Utilisation

- Ajouter les scripts en bas de page (comme jQuery) dans l'ordre suivant :
 - jQuery
 - AngularJS
 - Plugins angularJS
 - Votre fichier d'initialisation
 - Vos Controllers / Directives / Services / Etc.

- Initialisation :

```
// js/todoList.js
'use strict';

/**
 * Déclaration de l'application demoApp
 */
var demoApp = angular.module('demoApp', [
    // Dépendances du "module"
    'todoList'
]);
```

JS

Utilisation

Lien HTML <=> Controller

■ Directive :

```
<section ng-controller="todoCtrl">
  <form id="todo-form" ng-submit="addTodo()">
    <input id="new-todo" placeholder="Que devez-vous faire ?" ng-model="newTodo" />
  </form>
</section>
```

HTML

■ Controller :

```
var todoList = angular.module('todoList', []); //Création du module todoList

/**
 * Contrôleur de l'application "Todo List" décrite dans le chapitre "La logique d'AngularJS".
 */
todoList.controller('todoCtrl', ['$scope', function ($scope) {
  var todos = $scope.todos = [];
  var newTodo = $scope.newTodo;
  $scope.addTodo = function() { return false; };
}
]);
```

JS

LIVE CODING² - AngularJS

CALCULATRICE – 30MIN

Outils de développement

Introduction

- Productivité First
- Versionning (SVN, Mercurial, Git, ...)
- IDE (Webstorm, Visual studio, Xcode, ...)
- Task runner (Grunt, Gulp)
- Deploiement continu (Jenkins)
- Test (Mocha, Karma, CasperJS)



simple, flexible, fun



Versionning

« Un logiciel de gestion de versions agit sur une **arborescence de fichiers** afin de conserver toutes les versions des fichiers, ainsi que les **différences** entre les fichiers. »

- SVN : Système centralisé. Un repo « master » distant, et des repo « slave » sur les clients.
 - Mercurial : Système décentralisé mettant en avant la simplicité d'utilisation. Ecrit en python, il est utilisé par Facebook en interne. Hébergeur: bitbucket, etc.
 - Git : Système le plus en vogue de versionning décentralisé, proche de Mercurial mais semble plus dur à l'apprentissage. Hébergeur : Github*, Gitlab, bitbucket, etc.
-
- Pack étudiant : <https://education.github.com/pack> »

Git

3 niveaux d'abstractions : le répertoire de travail, le répertoire local et le répertoire distant

- Git clone ADDRESSE
- Git status
- Git commit -m « MESSAGE »
- Git add .
- Git push
- HOOK : Fonctions disponible en lien avec un évènement git (push, pre-push, pull, etc.)

<http://www.git-tower.com/blog/git-cheat-sheet/>

Task runner

Logiciel utilisé pour automatiser des tâches récurrentes dans le développement (compilation, minification, optimisation, etc.)

- **Grunt : Création de tâche qui s'exécute les une après les autres**
- **Installation : NodeJS + NPM**

```
$ npm install -g grunt-cli  
$ npm install grunt --save-dev  
  
$ npm install grunt-PACKAGE
```

```
module.exports = function(grunt) {  
  
  grunt.initConfig({  
    sass: {                                // Nom de la tâche  
      dist: {                             // Nom de la sous-tâche  
        options: {                       // Options  
          style: 'expanded'  
        },  
        files: {                         // Liste des fichiers  
          'main.css': 'main.scss',      // 'destination': 'source'  
          'widgets.css': 'widgets.scss'  
        }  
      }  
    }  
  })  
  
  // Import du package  
  grunt.loadNpmTasks('grunt-contrib-sass')  
  
  // Redéfinition de la tâche `default` qui est la tâche lancée dès que vous lancez Grunt  
  // Note : ici, nous définissons sass comme une tâche à lancer si on lance la tâche `default`  
  grunt.registerTask('default', ['sass:dist'])  
}
```

Task runner²

- **Gulp: Création de pipe d'action qui s'exécute en flux. Plus simple et fluide que grunt**
- **Installation : NodeJS + NPM**

```
$ npm install -g gulp  
$ npm install gulp-PACKAGE
```

```
var gulp = require("gulp")  
var gutil = require("gulp-util")  
var plumber = require("gulp-plumber")  
var cssnext = require("gulp-cssnext")  
var csso = require("gulp-cssso")  
var options = require("minimist")(process.argv.slice(2))  
  
gulp.task("styles", function() {  
  gulp.src("./src/css/*.css")  
    .pipe(!options.production ? plumber() : gutil.noop())  
    .pipe(cssnext({sourcemap: !options.production}))  
    .pipe(options.production ? csso() : gutil.noop())  
    .pipe(gulp.dest("./dist/css/"))  
})  
  
gulp.task("default", ["styles"], function() {  
  gulp.watch("./src/css/**/*.css", ["styles"])  
})
```

Déploiement continu

il assure la bonne **compilation** du code, le jeu des **tests unitaires**, le **packaging**, le **déploiement** et l'exécution des tests dans un environnement d'intégration. Il est en outre indépendant des configurations spécifiques aux postes de développeurs. Ensuite il va déployer l'application sur l'environnement de production.

