

# Le web 2.0

## Olivier Caron

Polytech Lille  
Avenue Paul Langevin Cité Scientifique  
Université de Lille  
59655 Villeneuve d'Ascq cedex

<http://ocaron.polytech-lille.net>  
Olivier.Caron@polytech-lille.fr



## Clients légers versus clients riches

- Constats clients légers webs :

## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit

## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - – une interface graphique limitée

## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - – une interface graphique limitée
  - – programmation web fastidieuse (succession de pages HTML)  
peut-on parler de programmation ?

## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - – une interface graphique limitée
  - – programmation web fastidieuse (succession de pages HTML)  
peut-on parler de programmation ?
- Vers des clients webs riches :

## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - – une interface graphique limitée
  - – programmation web fastidieuse (succession de pages HTML)  
peut-on parler de programmation ?
- Vers des clients webs riches :
  - – nécessite un runtime intégré au navigateur (exemple plugin flash)

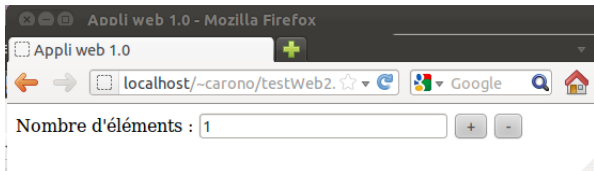
## Clients légers versus clients riches

- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - – une interface graphique limitée
  - – programmation web fastidieuse (succession de pages HTML)  
peut-on parler de programmation ?
- Vers des clients webs riches :
  - – nécessite un runtime intégré au navigateur (exemple plugin flash)
  - + objectifs : meilleure ergonomie, retour à la "vraie" programmation



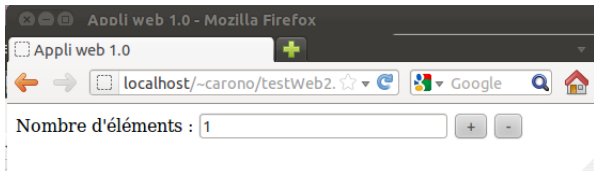
## Qu'est ce qui ne va pas dans Web 1.0 ?

- Une comparaison Web 1.0 - 2.0 par l'exemple :



## Qu'est ce qui ne va pas dans Web 1.0 ?

- Une comparaison Web 1.0 - 2.0 par l'exemple :



- Objectif : modifier la valeur du champ de texte par l'appui sur des boutons '+' et '-'  
Exemple classique de quantité d'un produit dans les applications de commerce électronique.

## Implémentation Web 1.0

- Technologies HTML et PHP, fichier web1.php

```
<html>
  <head> <title>Appli web 1.0</title> </head>
  <body>
    <?php
      extract($_POST) ;
      if (isset($_POST['nbElement'])) {
        if ($action=="+") $nbElement=$nbElement+1 ;
        else $nbElement=$nbElement-1 ;
      } else $nbElement=1 ;
    ?>
    <form action="web1.php" method="post">
      Nombre d'elements :
      <input type="text" name="nbElement"
        value="<?=$_$nbElement_ ;_?>" />
      <input type="submit" name="action" value="+" />
      <input type="submit" name="action" value="-" />
    </form> </body> </html>
```

## Implémentation Web 2.0

- Technologies HTML et Javascript, fichier `web2.html`

```
<html>
  <head> <title>Appli web 2.0</title>
  <script type="text/javascript">
    function modifierNbElement(action) {
      textfield=document.getElementById('tf_nbEl') ;
      valeur=parseInt(textfield.getAttribute('value')) ;
      if (action=='+')
        textfield.setAttribute('value',valeur+1) ;
      else
        textfield.setAttribute('value',valeur-1) ;
    }
  </script> </head> <body>
  Nombre d'elements :
  <input id="tf_nbEl" type="text" name="nbElement" value="1" />
  <button onclick="modifierNbElement('+');"></button>
  <button onclick="modifierNbElement('-');"></button>
</body> </html>
```

## Comparaison coût réseau des deux approches

- Approche Web 1.0 :

## Comparaison coût réseau des deux approches

- Approche Web 1.0 :
  - Premier chargement : appel puis transfert web1.php (interface graphique + données)

## Comparaison coût réseau des deux approches

- Approche Web 1.0 :
  - Premier chargement : appel puis transfert `web1.php` (interface graphique + données)
  - à chaque modification : appel puis transfert `web1.php` (interface graphique + données)

## Comparaison coût réseau des deux approches

- Approche Web 1.0 :
  - Premier chargement : appel puis transfert `web1.php` (interface graphique + données)
  - à chaque modification : appel puis transfert `web1.php` (interface graphique + données)
- Approche Web 2.0 :



## Comparaison coût réseau des deux approches

- Approche Web 1.0 :
  - Premier chargement : appel puis transfert `web1.php` (interface graphique + données)
  - à chaque modification : appel puis transfert `web1.php` (interface graphique + données)
- Approche Web 2.0 :
  - Premier chargement : appel puis transfert `web2.html` (interface graphique + données)

## Comparaison coût réseau des deux approches

- Approche Web 1.0 :
  - Premier chargement : appel puis transfert `web1.php` (interface graphique + données)
  - à chaque modification : appel puis transfert `web1.php` (interface graphique + données)
- Approche Web 2.0 :
  - Premier chargement : appel puis transfert `web2.html` (interface graphique + données)
  - à chaque modification : coût nul.

## Qu'est ce qui ne va pas dans Web 1.0 ?

- En résumé :

## Qu'est ce qui ne va pas dans Web 1.0 ?

- En résumé :
  - L'interface graphique est figée :  
changer une partie de l'interface nécessite de recharger toute la page !

## Qu'est ce qui ne va pas dans Web 1.0 ?

- En résumé :
  - L'interface graphique est figée :  
changer une partie de l'interface nécessite de recharger toute la page !
  - Problème de latence réseau (insupportable pour l'utilisateur) dues à :

## Qu'est ce qui ne va pas dans Web 1.0 ?

- En résumé :
  - L'interface graphique est figée :  
changer une partie de l'interface nécessite de recharger toute la page !
  - Problème de latence réseau (insupportable pour l'utilisateur) dues à :
    - mode synchrone d'envoi des données d'un formulaire

## Qu'est ce qui ne va pas dans Web 1.0 ?

- En résumé :
  - L'interface graphique est figée :  
changer une partie de l'interface nécessite de recharger toute la page !
  - Problème de latence réseau (insupportable pour l'utilisateur) dues à :
    - mode synchrone d'envoi des données d'un formulaire
    - les données récupérées concernent l'interface ET les données (toute la nouvelle page HTML), c'est lourd et ça prend du temps à se télécharger et s'afficher

## La technologie AJAX

- La technologie du moment, "Quoi, tu ne connais pas Ajax ?"



## La technologie AJAX

- La technologie du moment, "Quoi, tu ne connais pas Ajax ?"
- Symbole du web 2.0 (c'est du marketing)

## La technologie AJAX

- La technologie du moment, "Quoi, tu ne connais pas Ajax ?"
- Symbole du web 2.0 (c'est du marketing)
- Une bonne référence : "AJAX en pratique" (Ajax in Action)

## La technologie AJAX

- La technologie du moment, "Quoi, tu ne connais pas Ajax ?"
- Symbole du web 2.0 (c'est du marketing)
- Une bonne référence : "AJAX en pratique" (Ajax in Action)
- Rien de révolutionnaire mais la synergie de technologies existantes :

## La technologie AJAX

- La technologie du moment, "Quoi, tu ne connais pas Ajax ?"
- Symbole du web 2.0 (c'est du marketing)
- Une bonne référence : "AJAX en pratique" (Ajax in Action)
- Rien de révolutionnaire mais la synergie de technologies existantes :
- Acronyme pour "Asynchronous JavaScript + XML"

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML



## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur
- Technologie 2 : la technologie CSS

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur
- Technologie 2 : la technologie CSS
  - Acronyme pour "Cascading Style Sheet"

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur
- Technologie 2 : la technologie CSS
  - Acronyme pour "Cascading Style Sheet"
  - Découpage propre charte graphique / les données à afficher

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur
- Technologie 2 : la technologie CSS
  - Acronyme pour "Cascading Style Sheet"
  - Découpage propre charte graphique / les données à afficher
  - Styles visuels **réutilisables**

## Les 4 technologies associées à AJAX (1/3)

- Technologie 1 : le langage JavaScript
  - Langage de script. . .
  - Orienté objet
  - Code embarqué dans les pages HTML
  - Fortes connexions avec le cycle de vie du navigateur
- Technologie 2 : la technologie CSS
  - Acronyme pour "Cascading Style Sheet"
  - Découpage propre charte graphique / les données à afficher
  - Styles visuels **réutilisables**
  - Qualités évolutives (un seul fichier à évoluer)

## Syntaxe Générale CSS

- Syntaxe ultra simple :

```
sélecteur {  
    nomPropriété: valeur ;  
    nomPropriété: valeur ;  
    ...  
}  
sélecteur ...
```

- La complexité réside dans la richesse des propriétés et des effets de bord des propriétés entre elles (placement des objets)

## Petite illustration de règles CSS

- `h1 { color : red }` → les balises `<h1>` sont en rouge



## Petite illustration de règles CSS

- `h1 { color : red }` → les balises `<h1>` sont en rouge
- `div h1 { color : red ; }` → les balises `<h1>` contenues dans une balise `<div>` sont en rouge

## Petite illustration de règles CSS

- `h1 { color : red }` → les balises `<h1>` sont en rouge
- `div h1 { color : red ; }` → les balises `<h1>` contenues dans une balise `<div>` sont en rouge
- `.attention { border: solid blue 1px; background-color: cyan }` → style applicable à toute balise html (exemple `<div class="attention"> ...`)

## Petite illustration de règles CSS

- `h1 { color : red }` → les balises `<h1>` sont en rouge
- `div h1 { color : red ; }` → les balises `<h1>` contenues dans une balise `<div>` sont en rouge
- `.attention { border: solid blue 1px; background-color: cyan }` → style applicable à toute balise html (exemple `<div class="attention"> ...`)
- `#id121 { color:blue }` → style applicable à l'**unique** élément `<XXX id="id121"> ...`

## Les sélecteurs

- Les sélecteurs de base :  
Nom de la balise, Nom de l'id ('#'), Nom de la classe ('.').

## Les sélecteurs

- Les sélecteurs de base :  
Nom de la balise, Nom de l'id ('#'), Nom de la classe ('.').
- Les sélecteurs avancés :
  - \* : désigne toutes les balises
  - sél1 sél2 : les sélecteurs sél2 situés à l'intérieur de sél1
  - A[B] : une balise A qui possède un attribut B
  - A[B="value"] : une balise A qui possède un attribut B de valeur value

## Les sélecteurs

- Les sélecteurs de base :  
Nom de la balise, Nom de l'id ('#'), Nom de la classe ('.').
- Les sélecteurs avancés :
  - \* : désigne toutes les balises
  - sél1 sél2 : les sélecteurs sél2 situés à l'intérieur de sél1
  - A[B] : une balise A qui possède un attribut B
  - A[B="value"] : une balise A qui possède un attribut B de valeur value
- Et bien plus encore

## Comment appliquer un style ?

- Au niveau de la balise, attribut `style` :

```
<p style="color:blue;">
```

- Au niveau du fichier html :

```
<head> ...
```

```
  <style>
```

```
    p {
```

```
      color:blue;
```

```
    }
```

```
  </style>
```

```
  ...
```

- Dans une feuille de style (recommandé) :

```
<head> ...
```

```
  <link rel="stylesheet" href="./css/style.css" />
```



## Qui s'occupe des feuilles de style ?

- Absolument pas le développeur !



## Qui s'occupe des feuilles de style ?

- Absolument pas le développeur !
- Réservé au concepteur graphique multimédia, pour s'en convaincre, voici une démo :

<http://www.csszengarden.com/tr/francais/>

## Qui s'occupe des feuilles de style ?

- Absolument pas le développeur !
- Réservé au concepteur graphique multimédia, pour s'en convaincre, voici une démo :  
`http://www.csszengarden.com/tr/francais/`
- Complexe, mais des frameworks CSS, ex :  
framework bootstrap (twitter) `http://getbootstrap.com/`

## Le framework Twitter Bootstrap

- Une charte graphique épurée
- Beaucoup de règles pour de multiples composants graphiques
- S'appuie sur la notion de classe pour utiliser la charte

```
<button>un bouton</button> <!-- pas de style appliqué -->  
<button class="btn btn-primary">un joli bouton</button>
```

- Du code javascript permet d'associer du comportement aux composants
- Le placement des composants est simplifié grâce au concept de grille
- Règles dédiées aux mobiles/smartphones (responsive design).

## Installation du framework Twitter Bootstrap

- Téléchargez bootstrap sur <http://www.getbootstrap.com>
- Téléchargez JQuery <http://www.jquery.com>
- Télécharger popper sur <https://popper.js.org/>
- Installez le code bootstrap dans votre projet :

```
cd public_html/votreProjet/  
unzip ~/Downloads/bootstrap-X.X.X-dist.zip  
cp votreChemin/jquery.min.js ./js  
cp votreChemin/popper.js ./js
```

## Configuration de bootstrap

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"> <title>mon projet web</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="./css/bootstrap.min.css" rel="stylesheet" type="text/css">
    <script type="text/javascript" src="./js/jquery.min.js"></script>
    <script type="text/javascript" src="./js/popper.js"></script>
    <script type="text/javascript" src="./js/bootstrap.min.js"></script>
  </head>
  <body> ... </body>
</html>
```

Ces quelques lignes sont disponibles dans le fichier  
~ocarou/public/peip/preamble.html

## Les Grilles CSS

- Le placement des objets dans une page se fait dans une grille de **12** colonnes
- La grille est définie à l'aide de la classe `container`
- Une ligne de la grille est définie par la classe `row`
- Chaque colonne est spécifiée à l'aide des classes `col-xx-nb` ou `col-nb`  
*xx* précise l'équipement, *nb* le nombre de colonnes.

```
<div class="container">  
  <div class="row">  
    <div class="col-6">texte sur 6 colonnes</div>  
    <div class="col-3">texte sur 3 colonnes</div>  
  </div>  
  ...  
</div>
```

## Les tailles des grilles CSS

- Plusieurs formats selon l'équipement :

Equipement	Largeur	Nom classe CSS	Taille max col
Extra small-devices	<768px	col- <i>nb</i>	auto
Small devices	>=768px	col-sm- <i>nb</i>	60px
Medium devices	>=992px	col-md- <i>nb</i>	78px
Large devices	>=1200px	col-lg- <i>nb</i>	95px

## Les tailles des grilles CSS

- Plusieurs formats selon l'équipement :

Équipement	Largeur	Nom classe CSS	Taille max col
Extra small-devices	<768px	col- <i>nb</i>	auto
Small devices	≥768px	col-sm- <i>nb</i>	60px
Medium devices	≥992px	col-md- <i>nb</i>	78px
Large devices	≥1200px	col-lg- <i>nb</i>	95px

- nb* indique le nombre de colonnes (entre 1 et 12)



## Exemples grilles (1/2)

- 2 colonnes de même taille quelque soit l'équipement :

```
<div class="row">  
  <div class="col-6">du texte</div>  
  <div class="col-6">autre texte</div>  
</div>
```

- 1ère colonne : moitié de la grille pour mobiles, 33% pour desktop :

```
<div class="row">  
  <div class="col-6_col-md-4">.col-6 .col-md-4</div>  
  <div class="col-6_col-md-4">.col-6 .col-md-4</div>  
  <div class="col-6_col-md-4">.col-6 .col-md-4</div>  
</div>
```

## Exemples grilles (2/2)

- colonnes inutilisées, notion d'offset :

```
<div class="row">  
  <div class="col-md-4">col-md-4</div>  
  <div class="col-md-4_offset-md-4">col-md-4 offset-md-4</div>  
</div>
```

- Autre exemple :

```
<div class="row">  
  <div class="col-md-3_offset-md-3">col-md-3 offset-md-3</div>  
  <div class="col-md-3_offset-md-3">col-md-3 offset-md-3</div>  
</div>
```

## Les 4 technologies associées à AJAX (2/3)

- Technologie 3 : la technologie XML

## Les 4 technologies associées à AJAX (2/3)

- Technologie 3 : la technologie XML
  - Les pages HTML sont en fait des pages XML

## Les 4 technologies associées à AJAX (2/3)

- Technologie 3 : la technologie XML
  - Les pages HTML sont en fait des pages XML
  - On peut parcourir le contenu de cette page grâce à l'API DOM avec JavaScript

## Les 4 technologies associées à AJAX (2/3)

- Technologie 3 : la technologie XML
  - Les pages HTML sont en fait des pages XML
  - On peut parcourir le contenu de cette page grâce à l'API DOM avec JavaScript
  - On peut modifier la structure de la page (et pas la totalité) !!

## Petite illustration du DOM avec JavaScript/JQuery

- Soit le code HTML suivant :

```
<html>
  <head>
    <script type="text/javascript" src="./js/jquery.min.js"></script>
    <script type="text/javascript">
      function ajout() {
        nb=$("#zone_p").length+3 ; // analyse du DOM
        if (nb==6) return ;
        $("#zone").append("<p>IS_et_IS_2A_+nb+</p>") ;// modification DOM
      }
    </script>
  </head>
  <body>
    <h1 id="titre">bonjour les IS
      <button onclick="ajout();">...</button>
    </h1>
    <div id="zone"></div>
  </body>
</html>
```

## Les 4 technologies associées à AJAX (3/3)

- Technologie 4 : l'asynchronisme



## Les 4 technologies associées à AJAX (3/3)

- Technologie 4 : l'asynchronisme
  - invocation d'un traitement sans bloquer l'utilisateur

## Les 4 technologies associées à AJAX (3/3)

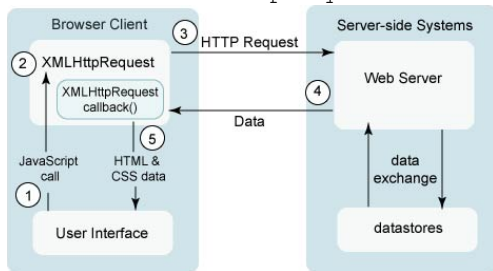
- Technologie 4 : l'asynchronisme
  - invocation d'un traitement sans bloquer l'utilisateur
  - Diminue notablement les problèmes de latence réseau

## Les 4 technologies associées à AJAX (3/3)

- Technologie 4 : l'asynchronisme
  - invocation d'un traitement sans bloquer l'utilisateur
  - Diminue notablement les problèmes de latence réseau
  - Charge uniquement des données (pas l'interface) de manière asynchrone

## Les 4 technologies associées à AJAX (3/3)

- Technologie 4 : l'asynchronisme
  - invocation d'un traitement sans bloquer l'utilisateur
  - Diminue notablement les problèmes de latence réseau
  - Charge uniquement des données (pas l'interface) de manière asynchrone
  - "The" méthode : XMLHttpRequest et ses multiples extensions



source : <https://marcautran.developpez.com>

## Architecture Logicielle Web 2.0 (1/2)

- Structuration de l'application en niveaux :

## Architecture Logicielle Web 2.0 (1/2)

- Structuration de l'application en niveaux :
  - Niveau données (exemple SGBD)

## Architecture Logicielle Web 2.0 (1/2)

- Structuration de l'application en niveaux :
  - Niveau données (exemple SGBD)
  - Niveau code métier (exemple PHP)

## Architecture Logicielle Web 2.0 (1/2)

- Structuration de l'application en niveaux :
  - Niveau données (exemple SGBD)
  - Niveau code métier (exemple PHP)
  - Niveau Interface Homme Machine (exemples : HTML et PHP, HTML et javascript)



## Architecture Logicielle Web 2.0 (2/2)

- Interactions entre les niveaux :

## Architecture Logicielle Web 2.0 (2/2)

- Interactions entre les niveaux :
  - Données - code métier : API bases de données (ex : PDO)

## Architecture Logicielle Web 2.0 (2/2)

- Interactions entre les niveaux :
  - Données - code métier : API bases de données (ex : PDO)
  - Code métier - IHM : échanges de données (ex : XML ou JSON)

## Echange de données

- Données XML :

## Echange de données

- Données XML :
  - Utilisé pour services webs

## Echange de données

- Données XML :
  - Utilisé pour services webs
  - inconvénient : verbeux, lourd à décoder



## Echange de données

- Données XML :
  - Utilisé pour services webs
  - inconvénient : verbeux, lourd à décoder
- Données JSON (JavaScript Object Notation) :

## Echange de données

- Données XML :
  - Utilisé pour services webs
  - inconvénient : verbeux, lourd à décoder
- Données JSON (JavaScript Object Notation) :
  - format de données textuel, générique



## Echange de données

- Données XML :
  - Utilisé pour services webs
  - inconvénient : verbeux, lourd à décoder
- Données JSON (JavaScript Object Notation) :
  - format de données textuel, générique
  - Avantages : simplicité, décodage rapide, typage des données

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur
  - des listes ordonnées de valeurs

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur
  - des listes ordonnées de valeurs
- Ces mêmes éléments représentent 3 types de données :

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur
  - des listes ordonnées de valeurs
- Ces mêmes éléments représentent 3 types de données :
  - des objets ('{' ... '}')

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur
  - des listes ordonnées de valeurs
- Ces mêmes éléments représentent 3 types de données :
  - des objets ('{' ... '}')
  - des tableaux ('[' ... ']')

## Structure de documents JSON

- Un document JSON ne comprend que deux éléments structurels :
  - des ensembles de paires nom / valeur
  - des listes ordonnées de valeurs
- Ces mêmes éléments représentent 3 types de données :
  - des objets ('{ ... }')
  - des tableaux ('[ ... ]')
  - des valeurs génériques de type tableau, objet, booléen (true, false) , nombre, chaîne ou null.



## Un exemple JSON

```
{  
  "name": "Frank",  
  "age": 24,  
  "engaged": true,  
  "favorite_tv_shows": [  
    "Lost", "Dirty_Jobs",  
    "Deadliest_Catch", "Man_vs_Wild"  
  ]  
}
```

# JavaScript et JSON

- JSON sous-ensemble de la notation objet de JavaScript

```
var objetJSON = {
  "name": "Franck", "age":24, "engaged" : true ,
  "favorite_tv_shows" : [
    "Lost", "Dirty_Jobs", "Deadliest_Catch", "Man_vs_Wild"
  ]
};
alert(objetJSON.name) // "Franck"
alert(objetJSON.favorite_tv_shows[1]) ; // "Dirty Jobs"
var chaineJSON = JSON.stringify(objetJSON) ; alert(chaineJSON) ;
var secondObjetJSON = eval( '(' + chaineJSON + ')' + ) ; // String to Json
```

## PHP et JSON

- Utilisation des tableaux associatifs PHP
- Deux fonctions : `json_encode($tab)` et `json_decode($chaine)`

```
$tab=array (  
    "name" => "Franck", "age" => 24, "engaged" => true ,  
    "favorite_tv_shows" => array ("Lost", "Dirty_Jobs" ,  
                                "Deadliest_catch", "Man_vs_Wild")  
);  
$chaineJSON=json_encode($tab);
```

## PHP, bases de données et JSON

```
<?php
    header( 'Access-Control-Allow-Origin : _*' );
    header( "Content-type : _application / json" );

    ... // connexion à une base
    $resultatRequete=pg_query( $connect , $requeteSQL ) ;
    $data=pg_fetch_all( $resultatRequete );
    $chaineJSON=json_encode( $data );
    echo $chaineJSON ;
?>
```

## Un exemple complet AJAX - le service Web (JSON/PHP) partie : I

```
<?php
/**
 * service Web au format JSON écrit en PHP qui retourne le responsable de
 * la promo
 * l'acronyme "promo" est passé en paramètre de type get
 */
header('Access-Control-Allow-Origin :_*');
header("Content-type :_application/json");
if (! isset($_GET["promo"]))
    $result=array("code" => -1, "message" => "erreur_paramètre_incorrect") ;
else {
    $server="localhost"; $base="promos"; $user="demo"; $password="postgres";
    $base=pg_connect("host=$server_dbname=$base_user=$user_password=$password
    ");
    if (! $base) // échec connexion
        $result=array("code" => -2, "message" => "erreur_connexion_BD:_:_".
        pg_last_error());
    else {
```

## Un exemple complet AJAX - le service Web (JSON/PHP) partie : II

```
$requeteSQL="select _responsable _from _promo _where _acronyme='".$_GET["
    promo"]."'" ;
$resultat=pg_query($base,$requeteSQL) ;
if (! $resultat) // échec requête
    $result=array("code" => -3, "message" => "Erreur_SQL:_:_".
        pg_last_error()) ;
elseif (pg_num_rows($resultat)!=1) // aucune ligne trouvée
    $result=array("code"=> -4, "message"=> $_GET["promo"]." _inconnue") ;
else
    $result=array(
        "code" => 0, "message" => pg_fetch_assoc($resultat)["responsable"]
    ) ;
}
}
echo json_encode($result) ; // résultat du service :
?>
```

## Un exemple complet AJAX - la vue partie : I

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>D mo AJAX</title>
  <script type="text/javascript" src="./js/jquery-3.2.1.min.js" >
  </script>
</head>
<body >
  <h1> Les responsables p dagogiques IS 2A</h1>
  <p>
    <input type="radio" name="promo" onclick="affResp('is2a3');" />IS 2A 3
    <input type="radio" name="promo" onclick="affResp('is2a4');" />IS 2A 4
    <input type="radio" name="promo" onclick="affResp('is2a5');" />IS 2A 5
  </p>
  <p>Responsable : <span id="resp"></span></p>
```

## Un exemple complet AJAX - la vue partie : II

```
<script type="text/javascript">
  function affResp(promo) {
    $("#resp").html("patientez ...") ;
    $.ajax ( { // fonction asynchrone JQuery
      url: './serviceWebPromo.php',
      type: 'GET',
      data: "promo="+promo,
      success: function(result){
        if (result.code ==0) $("#resp").html(result.message) ;
        else $("#resp").html("Echec:_" +result.message) ;
      }
    } );
  }
</script>
</body>
</html>
```



## L'industrie AJAX

- Essor très rapide

## L'industrie AJAX

- Essor très rapide
- Des applications vedettes : Google Calendar, Google maps, Google Office, Google mail, . . .

## L'industrie AJAX

- Essor très rapide
- Des applications vedettes : Google Calendar, Google maps, Google Office, Google mail, . . .
- Une contrainte : JavaScript, langage qui ne suscite pas l'unanimité.

## L'industrie AJAX

- Essor très rapide
- Des applications vedettes : Google Calendar, Google maps, Google Office, Google mail, . . .
- Une contrainte : JavaScript, langage qui ne suscite pas l'unanimité.
- Solution proposée par Google : Google Web Toolkit (GWT) :  
Compilateur Java vers code JavaScript