Le web 2.0

Olivier Caron

Polytech Lille
Avenue Paul Langevin
Cité Scientifique, Univ. 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

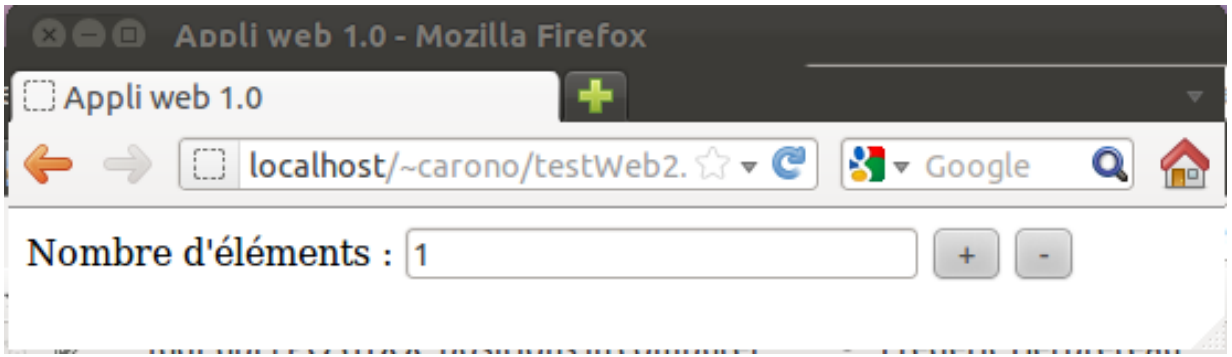
- Constats clients légers webs :
  - + déploiement quasi instantané : un navigateur web suffit
  - − une interface graphique limitée
  - − programmation web fastidieuse (succession de pages HTML) peut-on parler de programmation ?
- Vers des clients webs riches :
  - − nécessite un runtime intégré au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

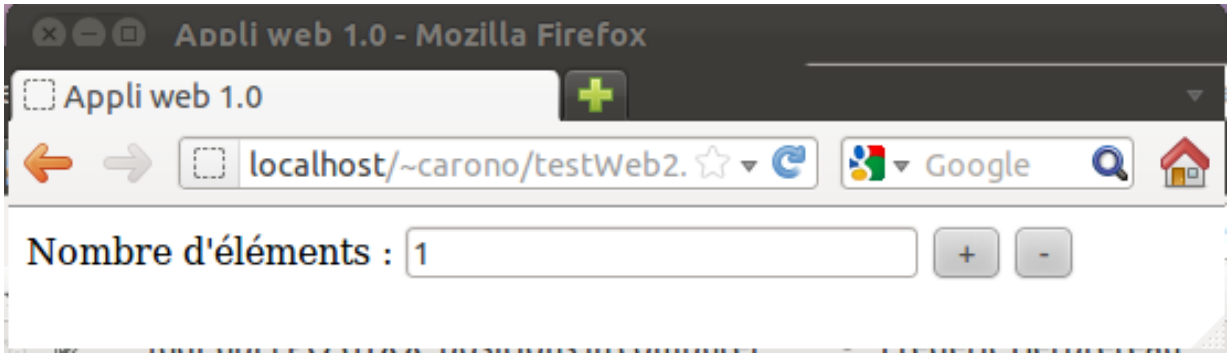
Qu'est ce qui ne va pas dans Web 1.0 ?

- 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="<? echo $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 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 sont en rouge
- `.attention { border: solid blue 1px; background-color: cyan}` → classe de 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 sont en rouge
- `.attention { border: solid blue 1px; background-color: cyan}` → classe de style applicable à toute balise html (exemple `<div class="attention">...`
- Un tutoriel : `http://www.w3.org/Style/Examples/011/firstcss.fr.html`

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 sont en rouge
- `.attention { border: solid blue 1px; background-color: cyan}` → classe de style applicable à toute balise html (exemple `<div class="attention">...`
- Un tutoriel : `http://www.w3.org/Style/Examples/011/firstcss.fr.html`
- et pour finir une démo : `http://www.csszengarden.com/tr/francais/`

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 le 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

Qui s'occupe des feuilles de style ?

- Absolument pas le développeur !
- Réservé au concepteur graphique multimédia
- Cependant, des assistants :
  `http://css-tricks.com/examples/ButtonMaker/`
  `http://cssmenumaker.com/`

Qui s'occupe des feuilles de style ?

- Absolument pas le développeur !
- Réservé au concepteur graphique multimédia
- Cependant, des assistants :
  `http://css-tricks.com/examples/ButtonMaker/`
  `http://cssmenumaker.com/`
- ou bien 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>
```

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`
  xx précise l'équipement, *nb* le nombre de colonnes.

```
<div class="container">
  <div class="row">
    <div class="col-xs-6">texte sur 6 colonnes</div>
    <div class="col-xs-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 **s**mall-devices	$<$768px	col-*nb*	auto
Small devices	$>=$768px	col-sm-*nb*	60px
Medium **d**evices	$>=$992px	col-md-*nb*	78px
Large devices	$>=$1200px	col-lg-*nb*	95px

Les tailles des grilles CSS

- Plusieurs formats selon l'équipement :

Equipement	**Largeur**	**Nom classe CSS**	**Taille max col**
Extra **s**mall-devices	$<$768px	col-*nb*	auto
Small devices	$>=$768px	col-sm-*nb*	60px
Medium **d**evices	$>=$992px	col-md-*nb*	78px
Large devices	$>=$1200px	col-lg-*nb*	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 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

- Soit le code HTML suivant :

```
<html> <head> </head>
<body>
  <h1 id="titre">bonjour les GIS</h1>
  <p id="p1"> ...</p> ...
</body></html>
```

- Retrouver un nœud du DOM :

```
var hello=document.getElementById("titre")
;
```

- Créer un nœud du DOM :

```
var elem=document.createElement("div") ; // création
d'une balise
var txt=document.createTextNode("texte") ;
// création d'un fragment de texte
pereElem.appendChild(elem) // attachement d'un noeud
à l'arbre xhtml
```

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`

Petite illustration de `XMLHttpRequest` (1/3)

```
<html>
  <head>
    <script type="text/javascript">
      var READY_STATE_UNINITIALIZED=0;
      var READY_STATE_LOADING=1;
      var READY_STATE_LOADED=2;
      var READY_STATE_INTERACTIVE=3;
      var READY_STATE_COMPLETE=4;
```

Petite illustration de XMLHttpRequest (2/3)

```
function chargerDoc(url) {
  var req = new XMLHttpRequest();
  req.open('GET', url, true);
  req.onreadystatechange = function callback(aEvt) {
    if (req.readyState == READY_STATE_COMPLETE)
      if(req.status == 200)
        afficherConsole(req.responseText);
      else
        afficherConsole("Erreur chargement documentation");
    else
      afficherConsole("chargement en cours...["+req.readyState+"]";)
  }
  req.send(null);
}
```

Petite illustration de XMLHttpRequest (3/3)

```
function afficherConsole(data){    // modifie la page HTML !
  console=document.getElementById("console");
  if (console!=null){
    var newline=document.createElement("div");
    console.appendChild(newline);
    var txt=document.createTextNode(data);
    newline.appendChild(txt);
  }
}

</script></head>
<body>
<button onclick="chargerDoc('data.txt');">Charger documentation</button>
<div id="console"></div>
</body></html>
```

Résultat d'exécution du script

```
Charger documentation
chargement en cours...[1]
chargement en cours...[1]
chargement en cours...[2]
chargement en cours...[3]
hello World !
```

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

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 :
  - 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($tab)` et `json_decode($tab)`

```
$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
<?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 ;
?>
```

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