



Chaire E- Business & Mobilité 2017-2018

Thomas Bourdeaud'huy

Septembre 2017



Objectifs

- Homogénéiser le groupe d'étudiants
- Préparer les interventions avancées de Maxence Lefebvre et Fabien Rondeau
- Donner des éléments de culture du Web et de bonnes pratiques
- Comprendre les architectures Web modernes :
 - RIA/Ajax, REST, MVC
- Aborder les problématiques de conception des applications Web :
 - UX, Responsive Design ...



Évaluation

- Connaissance des concepts : test papier
- Compétences :
 - Remise du travail de séance
 - Mini-projet d'approfondissement



Pédagogie

- Supports **minimalistes**
- Pour aller + loin :
 - Supports IG2I
 - Supports électif Web
- Cas d'étude corrigés en live-code
 - Cf. trame de séance
- Interventions des étudiants **bienvenues**
- Travail en dehors des séances **recommandé**



Plan

- Rappels historiques & perspectives
- Bases de Frontend : (X)HTML/CSS/JS
 - Design patterns XHTML/CSS
 - Cas d'étude trylt / signin
- Bases de Backend : Php/Mysql
 - Protocole HTTP, redirections, cookies, sessions
 - Cas d'étude sécurisation
 - TinyMVC
 - Cas d'étude chat
- AJAX/ JSON
 - Cas d'étude suggest
- JQuery
 - Cas d'étude paragraphes
- Bootstrap / HTML5
- Préprocesseurs CSS : SASS / LESS
- Mini-projet...



Outils

- Firefox/Firebug ou Chrome
- Distribution WAMP / Easyphp
 - Le mieux = apache2 sous Linux
- Un bon IDE
 - WebStorm/PhpStorm



***Un peu
d'Histoire***



Once Upon A Time ...

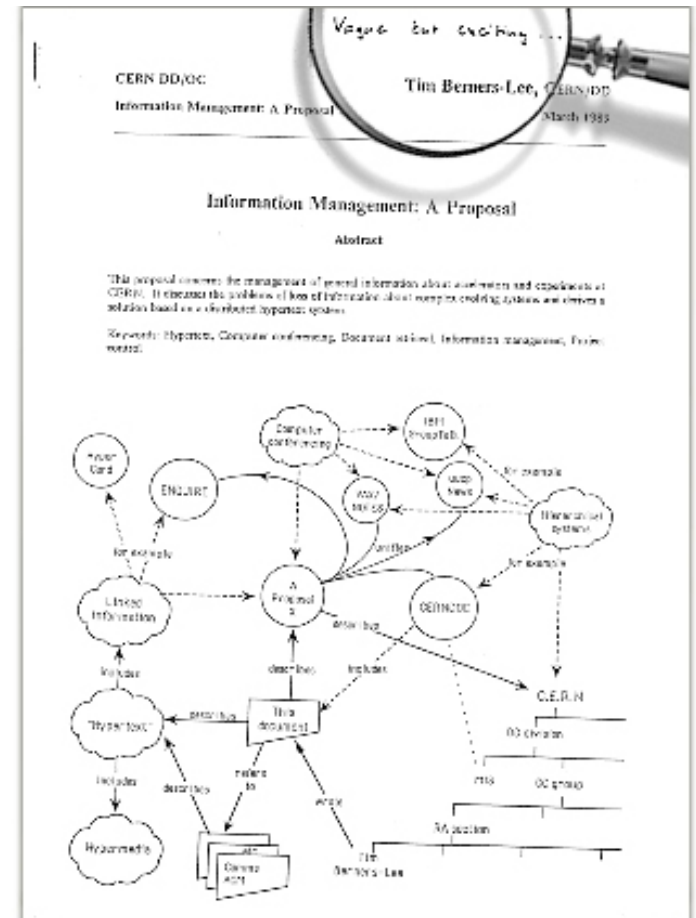
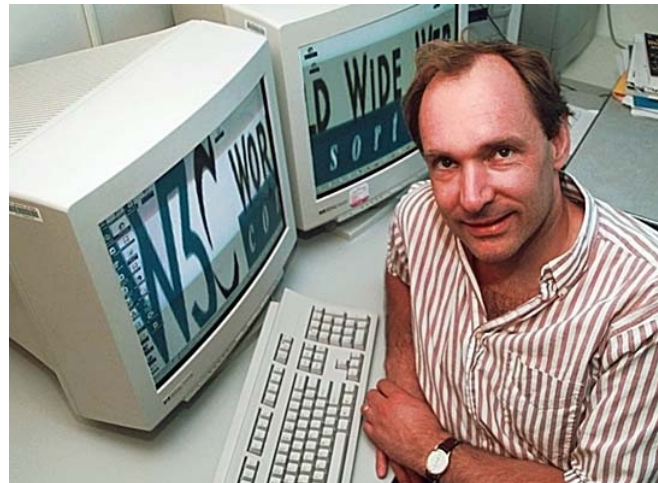
- 1980 : l'organisme ISO normalise le SGML - ISO 8879
 - Solution de portage de documents entre plateformes
- 1983 : adoption de TCP/IP
 - Arpanet (qui existe depuis les années 70) peut devenir Internet
- 1989 : Tim Berners Lee (CERN) propose HTTP & HTML
 - Une nouvelle méthode indépendante des matériels et des logiciels pour profiter des anciens travaux archivés
- HTTP = un protocole applicatif utilisant le réseau internet
- HTML = une application du SGML
 - « WorldWideWeb » est le nom de son navigateur
- 1994 : création du W3C





Vague, but exciting !

- WWW born at CERN
- <http://webfoundation.org/about/vision/history-of-the-web/>



Timothy Berners-Lee





Vocabulaire

- SGML : Standard Generalized Markup Language
- HTTP : Hypertext Transfer Protocol
- HTML : Hypertext Markup Language
- WorldWideWeb : le premier navigateur



Retour vers le futur

- 1999 : Tim Berners Lee propose le Web Sémantique
 - « donner du sens aux sites web »
 - <http://www.infotheque.info/cache/8094/www.urfist.cict.fr/archive/lettres/lettre28/lettre28-22.html>
- 2003 - 2005 : Tim O'Reilly parle du « Web 2.0 »
 - <http://oreilly.com/web2/archive/what-is-web-20.html>
 - Web 2.0 = interfaces riches; réseaux sociaux
- Depuis 2008 : HTML5
 - Standard en cours de rédaction, déjà implémenté dans les navigateurs
 - Futur concurrent du Flash; fortement soutenu par Apple
 - Nouvelles balises, APIs javascript



Un thème de CLOK ? Le Web Sémantique





RFC de l'IETF



- Internet Engineering Task Force
- Ensemble d'experts, universitaires majoritairement américains et scandinaves qui définissent les normes (protocoles) liées aux communications sur Internet (IP, TCP, UDP, HTTP, FTP, LDAP, SMTP, ...)
- Les RFC de l'IETF
 - « Request For Comments »
 - <http://www.rfc-editor.org/rfcsearch.html>
 - En français : <http://abcdrfc.free.fr/>
- Peu de RFC sont des standards, mais tous les standards d'Internet publiés par l'IETF sont des RFC



W3C



- World Wide Web Consortium
 - Consortium, installé au MIT à Boston, qui développe les technologies liées au web, dont le célèbre langage HTML (et ses variantes XHTML, XML, ...) et le protocole correspondant HTTP
 - Organisme de standardisation du Web
- Tous les standards du W3C :
 - <http://www.w3.org/TR/>
 - En français :
<http://www.w3.org/2005/11/Translations/Lists/ListLang-fr.html>



IETF / W3C

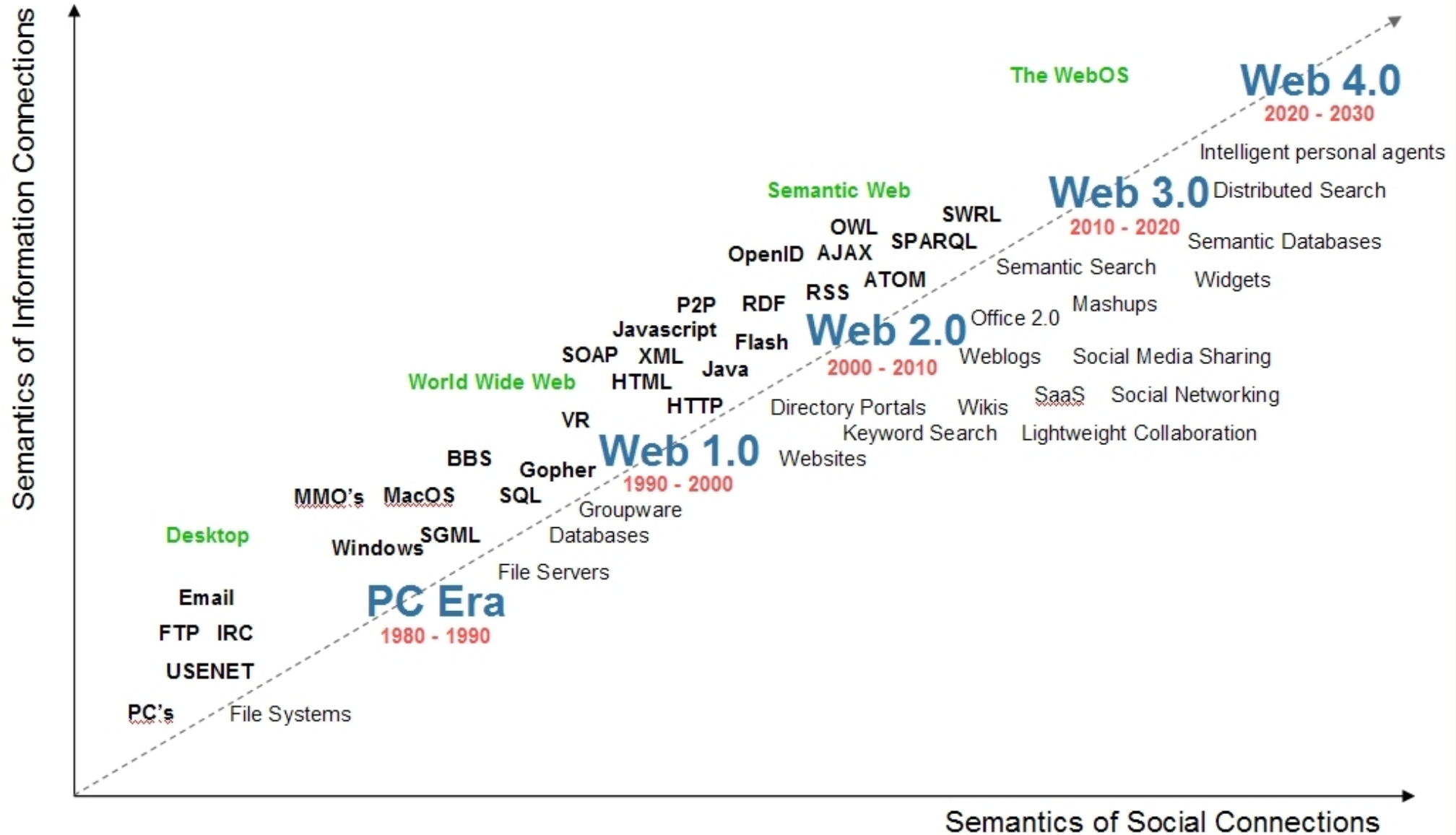
- IETF : les protocoles de l'internet
 - Documente les protocoles existants et leurs usages dans la vie courante
- W3C : l'architecture du Web
 - Propose de nouveaux standards pour une adoption future
 - Plus prospectif
- En détails
 - <http://www.w3.org/Consortium/Process/Process-19991111/appendix.html#ietf>



Où allons-nous ?

- Les grandes révolutions informatiques
 - Tim O'Reilly, Oscon'04
- « **hardware lock in** » (IBM) : l'informatique était verrouillée par les constructeurs d'ordinateurs
- « **software lock in** » (Microsoft) : les éditeurs de logiciels faisaient la loi
- « **data lock in** » (Google) : le Web est devenu LA plateforme, et les données sont sur le web
 - La valeur des applications repose sur leurs données
 - Celui qui hébergera les données contrôlera tout

Le WebOS





Frontend



Architecture côté client

3 Couches / 3 langages


- Structure : (X)HTML
- Présentation : CSS
- Interaction : Javascript
- Point de vue : Séparation des responsabilités



Frontend : ressources

*Avec le Trylt
Editor*

- Structure
 - <http://www.w3schools.com/tags/default.asp>
- Présentation
 - <http://www.csszengarden.com/tr/francais/>
 - <http://www.alsacreations.com/>
 - <https://www.w3.org/Style/CSS/>
 - <http://alistapart.com/>
 - Refcard CSS2 sur moodle
- Implémentation des standards dans les navigateurs
 - <http://www.quirksmode.org/>
- Exemples en ligne
 - <https://jsfiddle.net/>
- Code Babes
 - <https://codebabes.com/>



Frontend : Couche Structure

XHTML



Markup ?

Cf. markup.html

- Utilisation d'étiquettes (tags) ou balises prédéfinies
 - Annotations encadrant le contenu d'une page pour lui donner un sens particulier
- Référence aux annotations manuscrites laissées par l'auteur à l'imprimeur :
 - L'auteur dit : « ceci est un titre de chapitre »
 - `<=>` Couche **structure**
 - L'imprimeur dit : « ceci sera centré, en gras, et numéroté »
 - `<=>` Couche **présentation**
- e.g.
 - `<h1>Titre de premier niveau</h1>`



XHTML1 \neq HTML4

Une longue histoire

- HTML 4.01(1999)
 - Elle devait être la dernière pour laisser place à XHTML
- XHTML 1.0 (2000) puis XHTML 1.1 (2008)
 - Application de XML au HTML
 - Des règles plus strictes pour garantir une bonne visualisation sur les média alternatifs : pda, tv, téléphones, consoles, audio...
 - Rend obsolète les balises de spécification de forme : ``; `<i>`; `<frame>`...
 - Plusieurs DTD pour la rétrocompatibilité
- [Hot News] - 2009
 - XHTML 2 a été annulée
 - HTML5 le remplacera



Bonnes pratiques XHTML

Cf. dtd_stricte.html

- Utiliser la DTD stricte
 - <http://www.alsacreations.com/article/lire/560-dtd-doctype-html-xhtml-comment-choisir.html>
- Règles de syntaxe :
 - Les balises propriétaires ne sont pas autorisées
 - Toutes les balises sans exceptions doivent être fermées
 - Toutes les balises et leurs attributs doivent être en minuscules
 - Les attributs ne peuvent plus être minimalisés : on ne peut plus écrire `<option selected>`. Il faut écrire uniquement `<option selected="selected">`
 - Les guillemets sont obligatoires autour de toutes les valeurs d'attributs
- Valider vos pages XHTML & CSS
 - <http://validator.w3.org/>
 - <http://jigsaw.w3.org/css-validator/>



XHTML : Balise sémantique et accessibilité

- Le XHTML n'a pas pour objet de préciser la forme du contenu, juste sa structure
 - « Logical Markup » \neq « physical markup »
- Balise Sémantique = **Accessibilité**
 - Respect de la dualité « auteur » / « éditeur »
 - Capacités de « visualisation » par des périphériques alternatifs : audio, braille ... pour les personnes atteintes de handicaps

Contentez-vous de déclarer la structure du document, vous garantirez sa pérennité




Un thème de CLOK ?

WAI - WCAG

- Web Accessibility Initiative
- Web Content Accessibility Guidelines
- Directives du W3C pour l'accessibilité
 - <http://www.w3.org/WAI/>
 - <http://www.w3.org/2008/12/wcag20-pressrelease.html.fr>





Frontend : Couche Présentation

CSS

<http://www.w3.org/Style/CSS/>



Spécifications

- Style en ligne

```
<h1 style="font-style:italic; font-family:helvetica; font-size:14pt;"> titre 1 </h1>
```

- Style interne (a.k.a. style de document)

```
<head>
```

```
  <style>
```

```
    b { color : red; font-size : 14pt }    /* Commentaires */
```

```
  </style>
```

```
</head>
```

- Style externe (Feuille de style)

```
<head>
```

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

```
</head>
```





Sélecteurs

Fondamental pour jQuery !

- Sélecteur universel : `*`
 - `* {border: 1px red solid}`
- Sélecteur d'élément : `E`
 - `h1 {color: yellow; font-weight: bold}`
- Plusieurs sélecteurs `E1, E2, E3`
 - `h1,h2 {color: yellow; font-weight: bold}`
- Pseudo-classes :
hover, :active, :link, :visited, :first-child, :first-letter ...
 - `a:hover {background-color:yellow}`
- Sélecteur d'identificateur : `#id E#id`
 - `#id1 {color: yellow; font-weight: bold}`
- Sélecteurs de classe : `.class E.class`
 - `.classe1 {background-color: black; color: white}`
- Attribut `id="id1"`
 - Identifiant unique
- Attribut `class="classe1 classe2"`
 - Ensemble de balises



Sélecteurs CSS2

Cf. refcard CSS2

- Sélecteur de descendance
 - E F (hiérarchie)
 - E > F (enfant direct)
- Sélecteur de voisinage
 - E + F (éléments adjacents)
- Sélecteur d'attribut
 - E[att] présence de l'attribut att
 - E[att="val"] L'attribut att vaut val
- Et bien d'autres ...
- Et encore davantage en CSS3 !





Cascade : règles d'application des spécifications

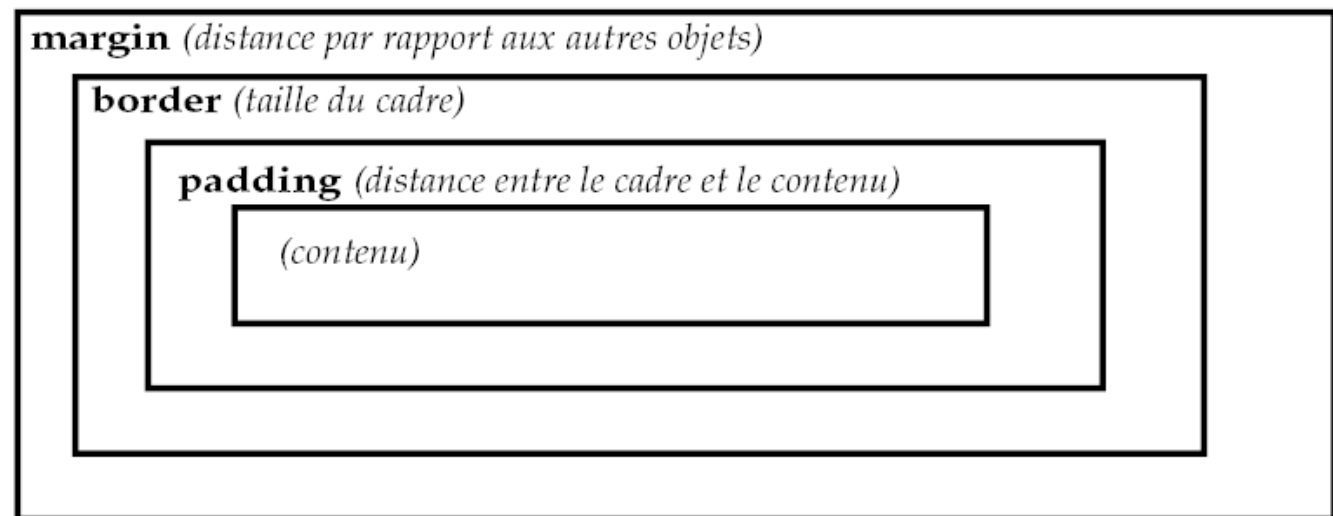
- Priorités :
 - plus une spécification est **proche**, plus elle est forte
 - plus une spécification est **précise**, plus elle est forte
- **!important**
 - Employé à la fin d'une règle, permet de forcer son application
- Héritage
 - Certaines propriétés sont automatiquement héritées du parent
 - La valeur **inherit** permet de forcer l'héritage



CSS : Boîtes

Cf. `boite.html`

- Modèle de boîte





CSS : display

Cf. display.html

- Définit la manière d'afficher une boîte
- inline
 - Plusieurs sur la même ligne
- block
 - Seul sur sa ligne, possibilité de lui donner des propriétés de taille : width, height...
- inline-block
 - Plusieurs sur la même ligne, peut être dimensionné
- none
 - Invisible, sans réservation de place
 - Pratique pour les “popins”, les styles d'impression...

Et d'autres !



CSS : position

Cf. positions.html

- Définit la **position d'une boîte**
 - static
 - Dans le flux
 - Positionnement par défaut
 - relative
 - Déplacement relatif à la position qu'il devrait avoir dans le flux
 - absolute
 - Sort du flux, se place dans le **conteneur**
 - Se superpose aux éléments déjà présents sans les contraindre
 - fixed
 - Ne scrolle pas avec la page

Conteneur :
Plus récent ancêtre
positionné



CSS : float

Cf. flottants.html

- Propriété `float:right` ou `float:left`
 - La balise “flotte” et remonte à la surface côté gauche ou droit.
 - Le texte qui suit s'écoulera autour de la boîte
- On utilise `clear:left` ou `right` pour attendre d'avoir atteint le bas du flottant
- Très utile pour répartir des boites dans un contexte responsive
 - Cf. exemple “Galerie d'images”
- Utiliser le site W3schools pour tester les flottants :
 - http://www.w3schools.com/cssref/pr_class_float.asp





XHTML / CSS : Exercices

- Paragraphes
- Galerie d'images





Un thème de CLOK ?

Préprocesseurs CSS

- Sass
 - Syntactically Awesome Stylesheets
- Less
- ...



Un thème de CLOK ? CSS3 & Flexbox

- Modèle de boîte flexible
- <http://www.alsacreations.com/tuto/lire/1493-css3-flexbox-layout-module.html>



Frontend : Couche Interaction

Javascript



Javascript : `getElementById`

Cf. `display_Acorriger.html`

- `var rNoeud = document.getElementById`
 - Permet de récupérer une référence sur un noeud du DOM
 - DOM : modèle objet de document
- Le noeud peut alors être manipulé :
 - `rNoeud.value` si c'est un champ de formulaire
 - `rNoeud.innerHTML` sinon
 - `rNoeud.style` pour modifier son style
 - ...
- Attendre que le navigateur ait terminé la lecture de la page complète !



Javascript : événements

- Des gestionnaires d'événements peuvent être placés comme attributs des noeuds
 - `<div onclick="monTraitement();">` une boite réactive `</div>`
- Une **référence sur l'élément siège de l'événement** peut être passée en paramètre du gestionnaire d'événement
 - `<div onclick="monTraitement(this);">` une boite réactive `</div>`
- Bonne pratique : commencer chaque gestionnaire d'événement par une trace d'exécution
 - alert, prompt, confirm : boîtes de dialogue
 - console.log : console de développement



Contexte de l'événement

- Les propriétés de l'événement ayant déclenché l'appel peuvent être transmis dans un objet 'event'
 - `<div onclick="monTraitement(event);"> une boite réactive </div>`
- Propriétés de l'objet event e :
 - `e.keyCode` : code ASCII de la touche appuyée
 - `e.clientX` : position de la souris lors du clic
 - `e.target` : élément cible de l'événement (cas de la propagation des événements aux parents)
 - ...
- <http://www.alsacreations.com/article/lire/578-La-gestion-des-evenements-en-JavaScript.html>



Liste des événements

- Abort l'utilisateur interrompt le chargement de l'image
- Blur l'élément perd le focus
- Change l'utilisateur modifie le contenu d'un champ de données
- Click l'utilisateur clique sur l'élément associé à l'événement
- Dblclick l'utilisateur double-clique sur l'élément associé à l'événement
- Dragdrop l'utilisateur effectue un glisser-déposer sur la fenêtre du navigateur
- Error se déclenche lorsqu'une erreur apparaît durant le chargement de la page
- Focus l'utilisateur donne le focus à un élément
- Keydown l'utilisateur appuie sur une touche de son clavier
- keypress l'utilisateur maintient une touche de son clavier enfoncée
- Keyup l'utilisateur relâche une touche de son clavier préalablement enfoncée
- Load le navigateur de l'utilisateur charge la page en cours
- MouseOver l'utilisateur positionne le curseur de la souris au-dessus d'un élément
- MouseOut le curseur de la souris quitte un élément
- Reset l'utilisateur efface les données d'un formulaire à l'aide du bouton Reset
- Resize l'utilisateur redimensionne la fenêtre du navigateur
- Select l'utilisateur sélectionne un texte dans un champ de type "text" ou "textarea"
- Submit l'utilisateur clique sur le bouton de soumission d'un formulaire
- Unload le navigateur de l'utilisateur quitte la page en cours
- MouseMove, MouseUp, MouseDown ...



Propriétés de l'objet event

- altKey Returns whether or not the "ALT" key was pressed
- Button Returns which mouse button was clicked
- clientX Returns the horizontal coordinate of the mouse pointer
- clientY Returns the vertical coordinate of the mouse pointer
- ctrlKey Returns whether or not the "CTRL" key was pressed
- metaKey Returns whether or not the "meta" key was pressed
- relatedTarget Returns the element related to the element that triggered the event
- screenX Returns the horizontal coordinate of the mouse pointer
- screenY Returns the vertical coordinate of the mouse pointer
- shiftKey Returns whether or not the "SHIFT" key was pressed
- keyCode Code ascii





Modularité

*Les variables locales
non déclarées par 'var'
sont globales !*

- Définition d'une fonction

```
function bonjour(NOM) {  
  var Bnom="bonjour "+NOM;  
  return Bnom; // facultatif  
}
```

- Définition d'une librairie

```
<script src="lib.js">  
  //rien ici, le fichier lib.js ne  
  contiendra pas les balises  
</script>
```

- Passage de paramètres :
 - par valeur pour les type natifs
 - par **référence pour les objets**
- Possibilité de ne pas fournir le paramètre
 - dans ce cas il est 'null' et il faut prévoir sa valeur dans le corps de la fonction





JS : exercices

- Fichier display_ACorriger.html
- Librairie utils.js
- TryIt Editor
- Cas d'étude côté client : Signin





Frontend : Conclusion

**Quizz Yourself
Quizz W3Schools**



Le Web côté Client

- HTTP = HyperText Transfer Protocol
- HTML = HyperText Markup Language
- XHTML = HTML à la sauce XML (DTD)
 - XML = Extensible Markup Language
 - DTD = Data Type Definition
- CSS = Cascading Style Sheets
- JS = Javascript
- DOM = Document Object Model
- DHTML = Dynamic HTML = XHTML + CSS + JS + DOM
- AJAX = Asynchronous Javascript & XML(HttpRequest)
- RIA = Rich Internet Application



Pour la prochaine fois ...

- Télécharger / prendre en main un IDE
 - WebStorm : javascript
 - PhpStorm : php
- Connaître de manière efficace :
 - Les principales balises HTML : tables, liens, formulaires
 - Les principaux sélecteurs & propriétés CSS
- Traiter le cas d'étude signini





Backend

Php
Mysql



Présentation




- 1994 PHP/FI (Rasmus Lerdorf)
 - pour “Personal Home Page Tools/Form Interpreter”
 - Acronyme récursif “Php Hypertext Processor”
- PHP 4.0 : Intégration du Moteur Zend
 - Du nom de l’entreprise qui dirige le projet
- PHP 5 : Nouveau moteur Zend2
 - Support objet complet
- PHP 7 : déc. 2015
- Langage de script spécialisé dans la génération de code HTML
- Nombreuses bibliothèques spécialisées : images, BDD, PDF ...
- Hérite de spécificités syntaxiques et sémantiques du langage C



Serveur !

- (W/M)AMP / Easyphp
 - Windows/Mac Apache Mysql Php
- Apache2 sous Linux
- Cf. remise à niveau pour la configuration : php.ini
 - NB : phpinfo() affiche le chemin du fichier php.ini
 - display_errors = On
 - error_reporting = E_ALL



The logo features two overlapping circles, one pink and one blue, with a dark purple silhouette of a person in a dynamic pose jumping or running across them.

Quelques éléments de php



\$Variables en php

elements.php

- La meilleure référence : <http://php.net/manual/fr>

```
<html><body> ...
```

```
<?php
```

```
// intégré dans le html; sera interprété par le serveur
```

```
$uneVariable="sa valeur"; // non typée
```

```
echo "Valeur de cette variable : " . $uneVariable;
```

```
?>
```

```
... </body></html>
```





Tableaux en php

elements.php

```
<?php
```

```
$unTableau = array(1,2,"trois");    // hétérogène
```

```
$unTableau[] = 4; // insertion sans indice !
```

```
$tabAssociatif = array("cle"=>"valeur", "cle2"=>2);
```

```
$tabAssociatif["cle3"] = "encore";
```

- Parcours :

```
for($i=0;$i<count($unTableau);$i++) echo $unTableau[$i];
```

```
foreach ($unTableau as $nextVal) ...
```

```
foreach ($tabAssociatif as $nextCle => $nextVal) ...
```





Modularité

elements.php

Déclaration d'une fonction: `function foo() {...}`

- Valeur par défaut des arguments optionnels : “=”

```
function puissance($x,$exposant=2) {  
    return pow($x, $exposant) ;  
}
```

```
$a =puissance(2);           //4
```

```
$b =puissance(2, 3);        //8
```

Charger une librairie : `include(“chemin_de_la_librairie.php”)`





Le protocole HTTP



Notion de Resource : URL

RFC 3986

- Uniform Resource Locator « Adresse Réticulaire »
 - Ressource : ***plus petite unité d'information adressable***
- Permet au client d'identifier :
 - le **type de protocole** à utiliser
 - l'**adresse précise de la ressource**
- Format : **<protocole>://<machine.domaine>/<chemin/ressource>#<a>?<q>**
 - RFC 3986 <http://www.w3.org/Addressing/>
 - Insensible à la casse
 - Encodage hexadécimal possible , recommandé pour # % < > @ & ? : / { } () | \ ^ ~ [] ` et espace
 - Cf. fonction php **urlencode()**
 - IRI permet d'utiliser tout unicode (RFC 3987)



Chemins Absolus/Relatifs

chemins.html



- Ne pas confondre
 - URL (http://)
 - Le navigateur va effectuer une connexion réseau
 - Chemin local (c:\...)
 - Le navigateur lit un fichier présent sur le disque dur de l'utilisateur
- Ne pas confondre
 - URL ABSOLUE (http://...)
 - Permet de se connecter à un serveur différent
 - Faute de goût si on reste sur le même serveur
 - URL RELATIVE (rep/fichier) - À privilégier
 - Induit une recherche de ressource sur le même serveur

Protocole HTTP 1.1

RFC 2616 / 7230

❶ Demande de connexion Tcp (Port 80)



❷ Connexion acceptée

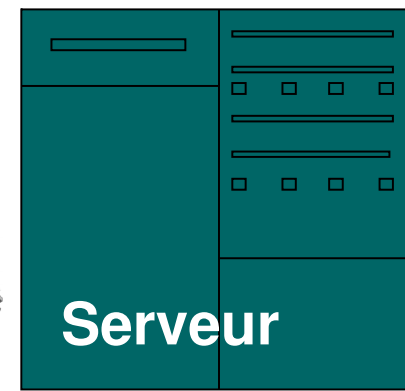
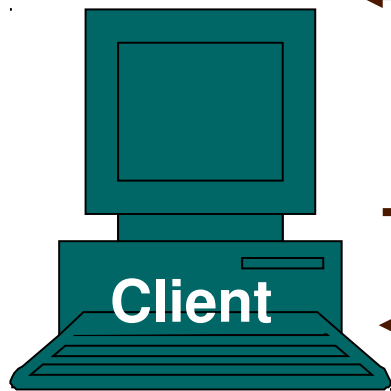


❸ Requête HTTP



❹ Envoi d'un page html

Fermeture de la connexion



NB : Il n'y a pas de session permanente entre le client et le serveur !

***.html → aucune modification**
***.php *.cgi, *.asp ...**
→ interprétation préalable





Entêtes de requête

→ Le navigateur envoie plus qu'une simple URL

- Une ligne de requête: comprend trois éléments devant être séparés par un espace :
 - La **méthode**
 - La **ressource**
 - La **version du protocole** utilisée par le client (généralement HTTP/1.0)
- Les champs d'en-tête de la requête: lignes composées du nom de l'en-tête, deux points (:) et de la valeur de l'en-tête
 - Ligne vide
 - Le corps de la requête (cas POST)



Entêtes de réponse

→ Ce qui est affiché par le navigateur n'est pas tout ce qui a été reçu

- Une ligne de statut :
 - La **version du protocole** utilisé
 - Le **code de statut**
 - La **signification du code**
- Les champs d'en-tête de la réponse : lignes composées du nom de l'en-tête, deux points (:) et la valeur de l'en-tête
- Ligne vide
- Le corps de la réponse:
 - Document demandé codé en (X)HTML
 - Autre chose



Exemples http.php

- Inspecter les interactions entre votre navigateur et des serveurs : firebug/onglet “Réseau”
 - telnet sur le port 80 sous Linux
- Le navigateur envoie une entête **Accept-Language**
 - Le serveur envoie une entête Location
- Le navigateur envoie une entête **Accept-Encoding**
- Le serveur envoie une entête **Content-Type**



Lire/Envoyer des entêtes en php : entetes.php

- Lire :
 - Tableau associatif `$_SERVER`
- Ecrire :
 - Fonction `header()`
 - A écrire AVANT tout HTML !
 - Sinon : erreur “headers already sent” ... KEZAKO?
- NB : Bufferisation possible
 - Configuration dans php.ini : `output_buffering`



Formulaires

- Un moyen d'envoyer des informations au serveur
 - 1) Ces informations sont collectées par le navigateur
 - Chaque information porte un nom : attribut **"name"**
 - Et une valeur : définie par l'utilisateur ou prédéfinie : attribut **"value"**
 - 2) Le navigateur prépare la chaîne de requête :
 - Format : **cle1=valeur1&cle2=valeur2**
 - 3) Il l'envoie au serveur à l'aide d'une requête HTTP
 - L'adresse de la page destination est précisée dans l'attribut **"action"**
 - Le type de requête est précisé dans l'attribut **"method"** : GET ou POST



Exemple : identification.html

```
<form action="identification.php" method="GET" />
<label for="pseudo">Pseudo : </label>
<input type="text" id="pseudo" name="login" />
<br />
<label for="mdp">Mdp :</label>
<input type="password" id="mdp" name="passe" />
<br />
<input type="submit" value="Envoyer" />
</form>
```

Tester POST





Tous les champs

```
<input type="" name ="nom" />
```

type= radio | checkbox | text | password | image | reset | submit | hidden | file

```
<select name = nom2>
```

```
  <option value = "val1">
```

```
  <option value = "valn">
```

```
</select>
```

```
<texarea name = nom3 rows = nb_lignes cols = nb_colonnes>
```

texte par défaut de la zone

```
</textarea>
```



Récupérer les données des formulaires en php

- Tableaux `$_GET`, `$_POST`
 - Tableaux superglobaux : disponibles dans toutes les portées
- `$_REQUEST`
 - un mix des deux avec POST qui écrase GET
- ***Never Trust User Input***
 - Ne pas afficher les erreurs => tester les paramètres

Librairie
maLibUtils.php





GET ou POST ?

- GET

- Variables visibles dans l'URL
- Limitation de la taille (dépend de l'OS)
- Simplicité d'extraction des données
- Persistance des données
- Données dans le fichier log

- POST

- les variables ne sont pas visibles dans l'URL
- les variables sont envoyées dans le corps de la requête HTTP(après les entêtes)
- Pas de limitation de la taille
- Possibilité d'upload (attribut de form : enctype=multipart/form-data)



GET ou POST ?

- Pour une meilleure performance, opter pour des champs courts transmis via GET
 - Permet aussi un débogage plus simple !
 - GET est plus simple à l'utilisation surtout dans le cas de langages non écrits pour le WEB
- Si les champs sont nombreux ou longs, opter pour POST
 - Pour la sécurité, utiliser POST (données non capturables directement dans l'URL)



Principales Méthodes HTTP

... vers REST - RFC 2616

- GET : récupérer une ressource
- HEAD : récupérer uniquement les méta-informations associées à une ressource, pas la ressource elle-même
- POST : annoter une ressource
- PUT : créer une ressource
- DELETE : demander à supprimer la ressource
- PUT ou POST in REST ?
<http://stackoverflow.com/questions/630453/put-vs-post-in-rest>



Redirections

- Entête de réponse HTTP : "Location:<url>"
- Déclenche une réaction de la part du navigateur :
 - Il va émettre une nouvelle requête vers l'url indiquée
 - Et donc afficher une autre page
 - l'entête REFERER contiendra l'adresse de la page initiale





Redirections : en php

redirection_form.php

- Fonction `header("Location:<url>")`
- Attention :
 - `header` n'arrête pas l'interprétation du script !
 - A vous de le faire avec `"die"`
- Renvoyer un message à l'occasion d'une redirection : chaîne de requête GET !
 - Utiliser `urlencode()`



Cookies - RFC 2109

“Etat persistant de données coté client”

- Fichier texte créé lors de la première visite du client
- Envoyé à l'initiative du **SERVEUR**
- Renvoyé par le client à chaque nouvelle requête vers cette destination, tant que la date d'expiration n'est pas échue
- Entête de réponse HTTP : **Set-Cookie**
 - Spécifie le domaine, un sous-chemin éventuel, la date d'expiration
- Entête de requête HTTP : **Cookie**
 - Quelques contraintes : nombre total limité à 300 ; taille maximale 4 ko ; au maximum 20 cookies par domaine



Cookies : en php

cookies.php

- Envoyer un cookie
 - fonction `setcookie(nom,valeur,[timestamp_expiration] ...)`
 - Sans date d'expiration, un cookie est supprimé dès la fermeture du navigateur
 - à écrire AVANT tout HTML :
 - Pourquoi ? Envoie un cookie d'identification !
 - Sinon ? Error : headers already sent
- Récupérer des cookies envoyés par client
 - `$_COOKIE["nomCookie"]`



Cas d'étude

"rester connecté"

- Améliorer le formulaire d'identification pour permettre à l'utilisateur de rester connecté, même plusieurs jours après son identification
- Si le client a coché la case "rester connecté", on lui envoie un cookie avec ses identifiants
 - Le navigateur renverra les identifiants du client tant qu'il possèdera le cookie
 - Le serveur s'en servira pour l'identifier
- Comment faire du *retargeting* ?





Cookies & Sessions

session.php

- Les cookies permettent au serveur de se souvenir du client, pour lui attribuer certaines ressources
- Par exemple, un espace de stockage de données spécifique : les variables de session
- Création d'une session : `session_start()`
- à écrire AVANT tout HTML
 - Pourquoi ? Envoie un cookie d'identification !
 - Sinon ? Error : headers already sent
- Une fois la session créée, on manipule le tableau `$_SESSION` en lecture/écriture



Vie des sessions

- En l'absence de requête de la part du client pendant **24 minutes** (cf. php.ini : session.gc_maxlifetime), le serveur oublie l'identifiant de session du client
 - Le client devra se reconnecter
- Les cookies d'identification de session n'ont **pas de date d'expiration**, ils sont donc stockés dans **l'environnement du navigateur** et pas sur le disque dur du client
 - Si le navigateur est fermé, le cookie disparaît, le client devra se reconnecter



Vie des sessions

- Les cookies sont partagés entre ***toutes les fenêtres du navigateur !***
 - => Pour se connecter avec deux utilisateurs différents, utiliser deux navigateurs différents
- Le serveur peut supprimer les données de session à tout moment
 - Utilisation de `session_destroy()`



Cas d'étude "sécurisation"

- Tester les identifiants d'un utilisateur
- Rediriger vers la page de menu en cas de succès
 - Après avoir sauvegardé une variable de SESSION attestant que l'utilisateur est parvenu à s'identifier
- Rediriger vers la page de connexion en cas d'échec
 - Eviter aussi les intrus qui arrivent directement dans la page de menu
- Offrir un menu de déconnexion manuelle





Php avancé

- **Templates**
- **Bases de données**



Retour sur les librairies

- `include("chemin_de_la_librairie.php")`
 - `include_once("chemin")` : ne sera inclus qu'une fois
- Le code de ces fichiers est **recopié** à l'endroit où la fonction est appelée
 - Il est aussi **exécuté**
 - Les fonctions qu'il définit deviennent disponibles !
- Très pratique pour la production de templates, l'internationalisation de sites...



Exemple : “templates”

templates.php

- Un fichier contient des définitions du vocabulaire dans la langue choisie, sous forme de tableau associatif :
`dictionnaire.php`
- Des templates utilisent ces mots de vocabulaire dans une structure XHTML
 - `presentation_fr.php`
 - `presentation_en.php`
- Le fichier principal inclut le vocabulaire et la structure :
`templates.php`





Créer une base de données

- Serveur de Bdd : mysql
- Utilisation de phpMyAdmin
 - Interface Web de gestion
- Fonctionnalités :
 - Import/export
 - Test de requêtes
 - Sécurisation



Se connecter à mysql en php

Ancienne méthode...

- Connection au serveur

```
mysql_connect("localhost", "root", "");
```

- Sélection de la base de données

```
mysql_select_db("mabase");
```

- Exécution d'une requête

```
$res = mysql_query("SELECT login FROM ... ");
```

- Parcours des résultats

```
while ($enregistrement = mysql_fetch_array($res)) {  
    echo $enregistrement["login"];  
}
```





Librairie maLibSQL.pdo.php

- Fichier **config.php**
 - Le **SEUL** fichier où l'on paramètre l'accès à la base de données
- **SQLUpdate(\$sql)**
 - Renvoie le nb d'enregistrements affectés
- **SQLInsert(\$sql)**
 - Renvoie l'identifiant (clé primaire/numAuto) de l'enregistrement inséré
- **SQLGetChamp(\$sql)**
 - Renvoie directement la valeur du champ recherché
- **SQLSelect(\$sql)**
 - Renvoie faux si aucun résultat



parcoursRs(\$result)

bdd.php

- **maLibSQL.pdo.php**
 - Fonction **parcoursRs()**
 - Transformation
ressource mySQL
<=> TabAssociatif

```
$sql= "select login from... ";  
$res = SQLSelect($sql);  
if ($res)  
{  
    $tab = parcoursRs($res);  
    foreach ($tab as $enrg)  
    {  
        echo $enrg["login"];  
    }  
}
```





Quelques mots sur ...

PDO : Php Data Objects

- Extension définissant l'interface pour accéder à une base de données depuis php
- Orientée Objet
- Nombreux pilotes de SGBD pour PDO
 - Facilite la migration d'une application vers un autre SGBD
- Requêtes préparées pour éviter les injections SQL



Quelques mots sur ... Injections SQL

- **NEVER TRUST USER INPUT**
- Lutter contre les injections SQL
 - Utiliser **addslashes** : protège les caractères spéciaux à l'aide de '\'
 - `maLibUtils.php` : **proteger(\$str)**
 - Nécessite d'encadrer TOUS les champs SQL provenant de l'utilisateur par des apostrophes





Un thème de CLOK ?

Sécurité du Web

- La fonction `protéger()` est insuffisante...
- A vous de le prouver !





Quelques mots sur ...

CRUD

- Les 4 opérations de base pour la gestion d'une base de données :
- Create : créer
- Read : lire
- Update : mettre à jour
- Delete : supprimer



Vers un Framework MVC

TinyMVC



Organisation Générale

- Principe de séparation des responsabilités
- Découpage **technique** pour la réalisation des interfaces
 - Un même besoin **fonctionnel** donne lieu au développement dans **chacune** des couches
 - On **contraint** le développeur pour garantir la qualité du code
- M – Modèle
- V – Vues
- C – Contrôleur



TinyMVC

- Trois fichiers php et des templates génèrent des morceaux d'interface web (éléments de menu, formulaires, etc.)
- Utilise systématiquement des requêtes de type GET pour faciliter le débogage des pages
- Pourquoi réécrire un framework MVC ?
 - zend, cakephp, symfony2 ...
 - On ne fait pas de la magie !



Couche Modèle

- Des fonctions « métier » d'accès à la base de données, en lecture/écriture
 - Permet une abstraction de la base de données
 - Vers les ORM...





TinyMVC : couche Modèle

- Fichier modele.php
 - Inclut maLibSQL.pdo.php (qui inclut config.php)
- Les fonctions de lecture renvoient des tableaux associatifs
 - Appels à parcoursRs()



Vues

- Les vues représentent l'IHM de l'application
- Elles utilisent des appels aux fonctions de la couche modèle pour récupérer les données à afficher
- Elles invoquent le contrôleur pour réaliser les traitements



TinyMVC : Vues

- Fichier index.php
 - Inclut modele.php
- Un paramètre “**view**” devra être systématiquement passé en paramètre
 - Selon les valeurs du paramètre view, il faudra parfois également passer d'autres paramètres.
- Inclut le template correspondant au paramètre “**view**” reçu
- Tous les formulaires présents dans les templates doivent renvoyer leurs données vers le contrôleur
 - en lui fournissant un paramètre “**action**”




Contrôleur

- Page destinataire des soumissions de formulaire
 - Elle exécute des appels aux fonctions de la page `modele.php` pour mettre à jour la base de données
 - Elle choisit la vue à afficher en retour par le navigateur



TinyMVC : contrôleur

- Fichier controleur.php
 - Inclut modele.php
- Teste la valeur d'un paramètre “**action**” qui caractérise le traitement à réaliser
 - Appelle des fonctions de modele.php pour réaliser ce traitement
- Redirige vers index.php en indiquant la vue à afficher (paramètre “**view**”)



Cas d'étude : "messagerie instantanée"

- Des utilisateurs peuvent se connecter
 - Le serveur enregistre leur identité en variables de session
- Une fois connectés, ils ont accès à la liste des conversations
- Une fois la conversation choisie, ils accèdent aux messages de la conversation
 - Postés par des utilisateurs non blacklistés
- Ils peuvent poster de nouveaux messages
 - Sauf si la conversation est archivée





Faibles XSS

- Injection de javascript dans les bases de données
- Faible de sécurité :
 - Le javascript peut insérer de nouveaux éléments non désirés dans la page
 - Il peut servir à “écouter” les événements déclenchés par l'utilisateur dans la page
 - e.g. appuis sur les touches du clavier lors de la saisie d'un mot de passe
- On s'en prémunit avec `htmlspecialchars()`



Limites du chat 1.0

- Comment afficher à l'utilisateur les nouveaux messages régulièrement ?
 - Recharger la page
- Sans lui faire perdre ce qu'il était en train d'écrire ?
 - Utiliser des requêtes asynchrones : Web 2.0



AJA...J ? ***[AJAx & Json]***

JSON : Javascript Object Notation

**AJAX : Asynchronous Javascript &
XML(HttpRequest)**



JSON :

Notation Objet Javascript

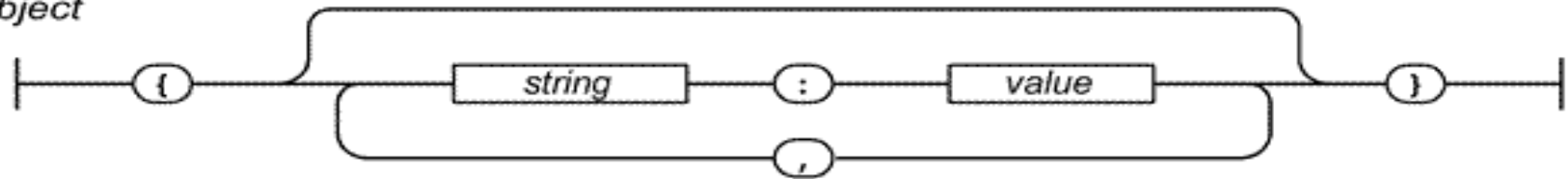
- Format textuel permettant de structurer des données, sous forme d'objet javascript
- Format :
 - <http://www.json.org/jsonfr.html>
- JSON dans du Javascript

```
var o = {"nom": "Bourdeaud'huy", "prenom": "thomas"};  
alert(o.nom);
```
- Tester si une propriété est absente :
 - `if (o.prop == undefined) ...`

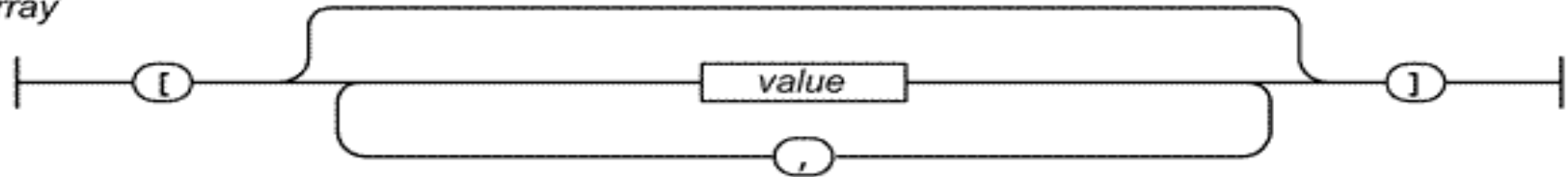


JSON : Grammaire Formelle

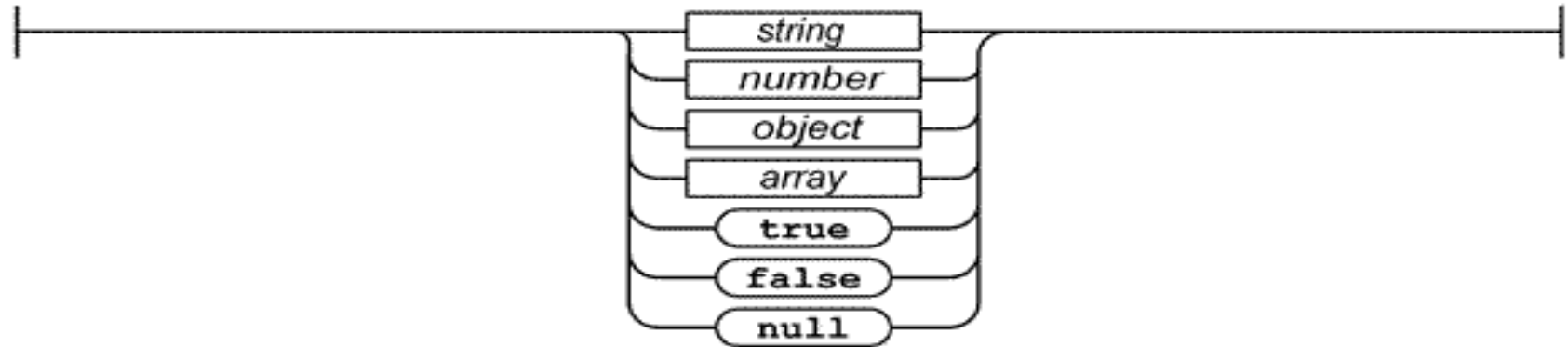
object



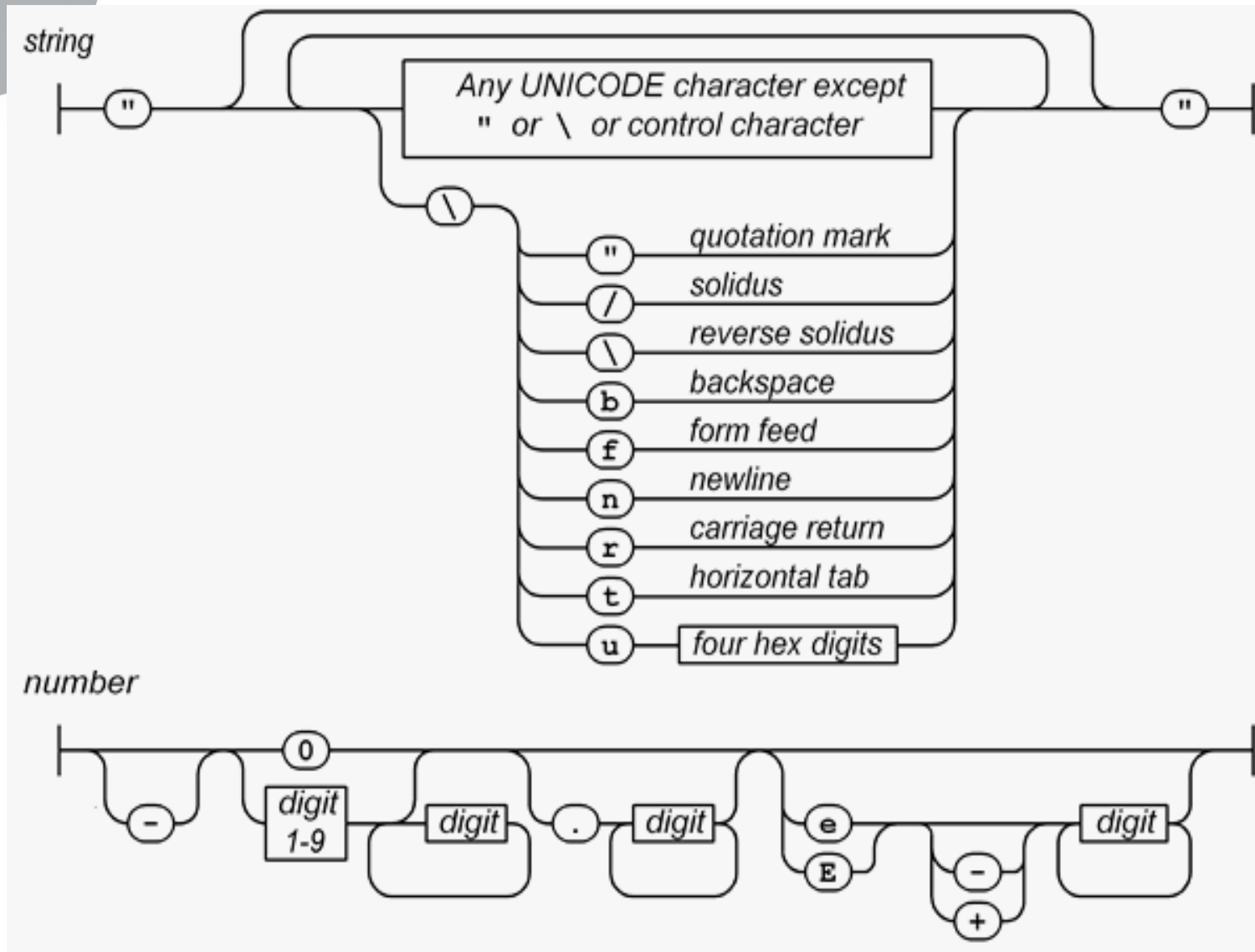
array



value



JSON : Grammaire Formelle





Intérêt de JSON

- Moins verbeux que XML
- Plus facile à interpréter
 - Des interpréteurs disponibles pour tous les langages
- Facile à afficher dans les consoles de développement des navigateurs
 - Utiliser `console.log` !
- En php :
 - `json_encode`
 - `json_decode`



Exemple : json.html

- Création/Interprétation & parcours d'objets json en javascript
 - On peut même y stocker des fonctions
- Attention !
 - Le passage d'objets aux fonctions
 - L'affectation d'objets entre eux

SE FONT PAR RÉFÉRENCE !



Exemple : json_encode.php

- Affichage de tableaux php sous la forme d'objets json
- Directement utilisable sur un résultat fourni par **parcoursRs()**
 - NB : json_encode nécessite des données encodées en utf8



Exercice : TryIt Editor

- Refactoriser `boucle(iPeriode,fnCbTraitement, fnCbContinuer)`
- Passer les arguments en JSON : `boucle({ [periode], traitement, [poursuite] })`
 - On définira une fonction `enrichir(oDefaut,oConfig)` qui renvoie un nouvel objet en ayant complété les propriétés absentes de l'objet de configuration par les valeurs de l'objet par défaut.
 - Utiliser une closure pour permettre les exécutions multiples
 - http://www.w3schools.com/js/js_function_closures.asp
- Refaire le tryIt Editor à l'aide de la fonction `boucle`
 - En utilisant des fonctions anonymes



- Terme employé depuis 2005 (Jesse James Garrett)
 - <http://www.adaptivepath.com/publications/essays/archives/000385.php>
- Utilisation conjointe d'un ensemble de technologies couramment utilisées sur le web :
 - HTML : structure sémantique des informations
 - CSS : présentation
 - Javascript / DOM : interagir dynamiquement
 - XML et l'objet XMLHttpRequest pour échanger et manipuler les données
- Permet de faire des requêtes au serveur sans changer de page
 - Les requêtes permettent de recharger seulement quelques éléments de la page
 - Navigateurs récents tous compatibles
 - Nombreuses bibliothèques disponibles pour faciliter le développement



Quelques exemples

- Gmail : <http://gmail.google.com/>
- Google suggest <http://www.google.com/>
- Portail Netvibes <http://www.netvibes.com/>
- Google maps <http://map.google.com/>
- Materiel.net <http://www.materiel.net>
 - Ajouter un article au panier
- « RIA » : Rich Internet Applications
 - Qu'est-ce qui est si riche ??

HTTP : protocole

❶ Demande de connexion Tcp (Port 80)

❷ Connexion acceptée

❸ Requête HTTP

❹ Envoi d'un page html

❺ Envoi d'un flux de données
XML ou JSON

de la connexion

**NB : Il n'y a pas de session
permanente entre le client et le
serveur !**

***.html → aucune modification
*.php *.cgi, *.asp ...
→ interprétation préalable**





Principe de fonctionnement

1) Préparation d'une fonction « callback » de traitement de la réponse

2) Préparation de l'objet de connexion javascript

3) Appel de la méthode « send » de l'objet de connexion javascript

Le navigateur émet une requête HTTP asynchrone SANS CHANGER DE PAGE

4) Le serveur traite la requête

Le navigateur reçoit la réponse et appelle la fonction callback de traitement

5) Le serveur renvoie la réponse au format HTTP

6) Exécution de la fonction « callback » de traitement de la réponse





Objet XMLHttpRequest : propriétés

- readyState (entier) :
 - 0 <=> non initialisé
 - 1 <=> ouverture
 - La méthode open() a été appelée avec succès
 - 2 <=> envoyé
 - La méthode send() a été appelée avec succès
 - 3 <=> en train de recevoir
 - On a reçu les entêtes de réponse, des données sont en train d'être transférées, le transfert n'est pas terminé
 - 4 <=> terminé
 - Les données sont chargées
- Onreadystatechange : fonction callback
- Gestionnaire d'événement invoqué à chaque fois que readystate change
- responseText (string) : Réponse en texte brut
- responseXML (objet DOM) : valeur retournée en XML
- status (entier) : code de statut de la réponse
- statusText (string) : texte associé au statut de la réponse



Objet XMLHttpRequest : méthodes

- void open(string type, string url, bool asynch)
 - Prépare la requête en précisant si elle est asynchrone ou non
 - Ouvre la connexion TCP avec le serveur
- void setHeader(string header, string valeur)
 - Assigne une valeur à un champ d'entête (utilisé après open)
- void send(string)
 - Envoie la requête
- void abort()
 - Abandonne
- string getAllResponseHeaders()
 - Retourne l'ensemble de l'entête de réponse
- string getResponseHeader(string header)
 - Retourne la valeur d'un champ d'entête spécifié



Un thème de CLOK ?

Limitations XMLHttpRequest

- Interdiction de faire des requêtes cross-domain
 - On peut le faire avec XMLHttpRequest niveau 2
 - Ou avec JSONP
 - ...





Exemple : Envoi Requête ajax.html

- Une requête asynchrone est envoyée par la méthode GET
 - Utilisation de `envoiRequete(type,url,donnees,callback)`
 - Qui instancie un objet `XMLHttpRequest`
- Lorsque le navigateur reçoit la réponse, il appelle la fonction de rappel passée en paramètre
- `EnvoiRequete` permet d'indiquer la méthode à employer pour l'envoi ("GET" ou "POST")



Exercice :

Refactoriser ajax.js

- `envoiRequete(type,url,donnees,callback)`
=> « Handler » `ajax({ url,[type],[data],[callback] })`
- Les arguments ET les données deviennent du JSON
- [Optionnel] Permettre les requêtes multiples



AJAX : architecture en ... 5 couches !

- Structure : HTML
- Présentation : CSS
- Interaction : JS
- Fourniture : PHP, côté serveur
 - Généralement, on fournit un flux en JSON
- Intégration : JS
 - On peut inverser ces deux étapes en fournissant un flux statique (une « fixture ») le temps de développer la couche intégration



Cas d'étude : EBM suggest

- V1 : couches structure, présentation
 - Importance de l'expérience utilisateur : UX
- V2 : couches interaction, intégration
 - Couche fourniture de données “factice” : **fixture**
- V3 : couche fourniture de données effective
 - Mauvaise pratique : le serveur nous renvoie un flux XHTML !





Cas d'étude : EBM suggest

- V4 : le serveur fournit un flux JSON
- Il y a maintenant indépendance entre le client et le serveur
 - Le flux fourni par le serveur pourrait être consommé par divers types de clients : smartphones, objets connectés...
- V5 : Le flux peut être fourni par une lecture de la base de données
- V6 : On peut le mettre en cache côté client
 - => Limitation de la charge du réseau





Cas d'étude "messagerie instantanée 2.0"

- La page d'affichage des messages envoie régulièrement une requête asynchrone au serveur pour recevoir les nouveaux messages
- Lorsque le serveur a répondu, les nouveaux messages sont affichés sous les précédents
- La page de fourniture des données utilise la couche modèle
 - Elle renvoie les nouveaux messages dans un flux JSON



Page rest.php / data.php

- Comme controleur.php
 - Inclut modele.php
 - Réagit au paramètre “action”
 - Mais
 - Fournit exclusivement des flux JSON
 - Ne redirige pas vers index.php
- => Fait pour les requêtes asynchrones



JQuery

Le framework javascript du Web 2.0





Kezako ?

- Framework Javascript libre et open source
 - Simplifie les interactions DOM/HTML/CSS/Javascript
 - « Write Less, Do More »
- Facilite la production d'interfaces riches (RIA)
- Développé par John Resig
 - Également auteur du plugin CCK de drupal
- Version 3.1.1 (18/10/2016) à télécharger sur internet :
<http://jquery.com/>





Ressources

- <http://docs.jquery.com/Tutorials>
- <http://api.jquery.com/>
- <http://speckyboy.com/2010/03/31/20-jquery-cheatsheets-docs-and-references-for-every-occasion/>
- Refcard, Livres et supports sur moodle



Installation

- Télécharger la dernière version du fichier js « compressé »
 - Le placer dans votre projet
- Commencer à coder !
 - Charger la librairie dans votre code source

```
<script type="text/javascript" src="<chemin>"></script>
```
 - Exercice :
 - Définir une classe “allume”
 - Définir un ou plusieurs paragraphes de la classe “exemple” cachés
 - Écrire une fonction “decollage”, appelée lors du clic sur un bouton

```
$("#p.exemple").addClass("allume").show("slow");
```



Sélecteurs

- \$ représente la fonction *jquery(selector, [context])*
- selector est une chaîne de caractère dénotant **un ou plusieurs éléments du DOM**
 - **[ou du contexte spécifié]**
- Accepte les valeurs classiquement admises en CSS
 - `$("p"); $("#id") ; $(".class")`
 - `$("div > span"); $("div + span");`
- Filtres de formulaires; de visibilité; d'attributs; de contenu; de parité
 - `$("#controle:checked"); $("form:input")`
 - `$("div:hidden");`
 - `$("input[value=toto]")`
 - `$("div:contains(coucou)")`
 - `$("tr:odd")`



Accès au contenu

- Balise quelconque :
 - `$(“balise”).html()`
 - `$(“balise”).text()`
- Champ de formulaire :
 - `$(“champ”).val()`
- Edition :
 - `$(“balise”).html(“nouveau contenu”)`
 - `$(“champ”).val(“nouveau contenu”)`
- Insertion / Suppression
 - `.remove();`
 - `.empty()`
 - `.replaceWith(“nouveau”);`
 - `.append(“nouveau”),`
 - `.prepend(“nouveau”),`
 - `.after(“nouveau”);`
 - `.before(“nouveau”)`



Insérer un contenu à la mode JQuery

- `var jP = $("<p>/<p>")`
 - Crée un élément JQuery (non affiché)
- `jP.html("contenu").addClass("allume")`
 - Modifie ses propriétés (contenu, style...)
- `$("somewhere").append(jP)`
 - Insère l'élément dans le DOM : il est affiché !
- `jP.html("modif")`
 - On peut **encore** le modifier : cela modifie l'élément affiché !
- `$("elsewhere").append(jP)`
 - Si on l'insère ailleurs, il se déplace : on manipule une référence !
- `$("elsewhere").append(jP.clone())`
 - Il faut le cloner pour le placer à plusieurs endroits



Exercice

- Modifier le label et le comportement du bouton pour qu'il permette de cacher les éléments qui avaient été affichés
- Créer plusieurs boutons synchronisés entre eux pour cacher/afficher
 - Le premier en haut, l'autre en bas
- Insérer des boutons permettant d'ajouter de nouveaux paragraphes à la page, placés avant ou après les paragraphes existants
 - Développer deux versions, selon que les paragraphes à saisir soient placés dans une balise div qui les contient tous ou directement entre les boutons





Accès au Style

- La fonction jquery renvoie un objet qui permet de manipuler les propriétés css des éléments considérés
 - `$("#bloc").position()`
 - `$("#bloc").css("background-color")`
 - `$("#bloc").css("border","1px solid red")`
 - `$("#bloc").css({ "background-color" : "yellow",
"border":"1px solid red"})`
 - `$("#bloc").addClass(classe); removeClass, toggleClass ...`



Effets

- Cacher / rendre visible
 - `show(speed,callback)`
 - `hide(speed,callback)`
 - `toggle(speed,callback)`
- Animer
 - `animate(params,options)`
 - `scrollTop(val) ...`





Evénements

- Il est préférable d'attendre que le DOM soit complètement chargé dans la mémoire du navigateur pour commencer à exécuter des traitements
 - `$(document).ready(callback)`
- JQuery permet de définir un gestionnaire d'événement
 - `$("p").click(callback);`
 - Agit juste sur les balises p du DOM actuel
 - `$(document).on("click", "p", callback);`
 - Agit sur les balises p actuelles et les futures p insérées dans le DOM
- Mais aussi de déclencher un événement
 - `$("#bloc").focus();`



Quelques astuces

- Une fois en production, ne plus nommer les fonctions :
 - Optimise le temps d'interprétation par le navigateur
 - Améliore la lisibilité du code (« unité de lieu » entre l'événement géré et le code de gestion)

```
$("#controle").click(function() {  
    // code de la fonction  
});
```

- Il existe des outils qui font ce travail pour vous...



Exercice

- Enlever le onclick, le onload
 - Le code HTML est “purifié”
- Définir les gestionnaires par du code au moment où le navigateur a terminé la lecture de la page





Quelques astuces

- `$(this)` représente l'**élément cible de l'événement** dans la fonction callback de traitement de cet événement
- Un **objet contexte** est toujours passé aux fonctions callback de traitement d'événement
 - Il suffit de **nommer** cet argument pour le récupérer !
- Exemple : pour récupérer le code de la touche appuyée, utiliser l'événement 'keyup' et la propriété `contexte.which`

```
$(document).keyup(function(contexte) {  
    alert(contexte.which);  
});
```



Quelques astuces : \$(ref)

- Si on dispose d'une référence javascript vers une balise, `$(laReference)` permet de récupérer l'objet jQuery correspondant
- `var refToto = document.getElementById("toto");`
- `$(refToto)` est la même chose que `$("#toto")`



Quelques astuces : Itérateur `.each()`

- `$("selecteur")` renvoie souvent un ensemble d'éléments
- On peut exécuter une fonction pour chacun d'eux à l'aide de `.each(callback)`
- A chaque exécution de la fonction de rappel, `$(this)` dénotera l'élément courant
- Exemple :

```
$("textarea").each(function () {  
    console.log("traitement de" + $(this).val());  
});
```



Quelques astuces : `.data(...)`

- Affectation de méta-données à un ensemble de balises :
 - `.data(cle,valeur)`
 - `.data(objet)`
- Relecture des méta-données :
 - `.data(cle)`
 - `.data()`





Quelques astuces :

Contexte

- La fonction `jQuery(selector, context)` prend deux arguments :
 - Sélecteur “CSS”
 - Contexte dans lequel rechercher les éléments à sélectionner
- Exemple : recherche d'un textarea dans le contexte d'un paragraphe (ie à l'intérieur)

```
$("p").click(function(){  
    if ($("#textarea",$(this)).length != 0) return;  
})
```



Quelques Astuces : Passer des données au gestionnaire d'evnt

- Un objet « contexte » est passé à chaque gestionnaire d'événement jQuery.
 - On peut lui associer des données supplémentaires qui pourront être récupérées quand le gestionnaire d'événement sera exécuté

```
function greet( event ) {  
    alert( "Hello " + event.data.name );  
}  
  
$("button").on( "click", {name: "Karl"}, greet);  
$("button").click({name: "Karl"}, greet);
```




Formulaires

- Filtres de formulaires
 - `$("form:input") ...`
- Accès aux attributs et valeurs des champs
 - `$("#choix").attr("name","nom")`
 - On ne peut changer l'attribut "type" :
<http://stackoverflow.com/questions/1544317/jquery-change-type-of-input-field>
 - `$("#champ").val("value")`
- Déclenchement d'actions
 - `$("#form1").submit()`
 - `$("#nom").attr("checked",true)`
- Gestionnaires d'événements
 - `$("#form1").submit(callback)`





Ajax en JQuery

- Déclenchement d'une requête
 - `$.getJSON(url,data,callback)`
 - Handler pour la méthode `ajax({...})` :

`$.getJSON(url,data,callback)`

`<=> $.ajax({
 dataType: "json",
 url: ... ,
 data: ... ,
 success: ...})`



Quelques astuces : \$.ajax ({context :...})

- Réutiliser une information connue au moment du lancement de la requête ajax, dans la fonction de rappel au retour de la réponse du serveur :
 - V1 : closure !
 - V2 : context dans la méthode ajax

```
$.ajax({  
  url: "test.html",  
  context: document.body // deviendra le contexte de la callback  
}).done(function() {  
  $(this).addClass( "done" );  
  // $(this) dénote body !  
});
```





Exercice : paragraphes éditables

- Des boutons '+' seront affichés par jQ de part et d'autre d'un div "contenu"
 - Ils permettront de créer de nouveaux paragraphes lors d'un click
- Les paragraphes créés seront éditables
 - Un clic sur un paragraphe éditable permettra d'afficher un champ de saisie permettant la modification du paragraphe.
 - Plusieurs paragraphes doivent pouvoir être éditables en même temps
 - Si on appuie sur Entrée depuis ce champ de saisie, on valide la saisie
 - Si on clique sur la touche 'ESC' depuis n'importe où dans la page, on annule l'édition en cours en réaffichant l'ancien contenu.
 - On utilisera `$(...).data` pour stocker les valeurs des anciens contenus à restaurer lors de l'appui sur 'ESC'





Cas d'étude "Tiny CMS"

- Récupérer des paragraphes depuis une base de données
- Ajouter à la page un bouton « Mode Edition » permettant d'activer les fonctionnalités d'édition
 - A chaque modification, envoyer la valeur du paragraphe édité au serveur pour le sauvegarder en base de données
 - Interdire l'envoi d'un paragraphe vide
- Ajouter des fonctionnalités de création de nouveaux paragraphes
- Utiliser JQuery UI pour développer les fonctionnalités suivantes :
 - Déplacement de nouveaux paragraphes pour en changer l'ordre
 - Suppression d'un paragraphe



Jquery UI



jquery

user interface



Kezako ?

- Ensemble de composants d'interaction pour faciliter la production d'*Interfaces Utilisateur* riches
 - « RIA » : Rich Internet Applications
- « *jQuery UI is a curated set of user interface interactions, effects, widgets, and themes built on top of the jQuery JavaScript Library. Whether you're building highly interactive web applications or you just need to add a date picker to a form control, jQuery UI is the perfect choice.* »



Téléchargement =f(librairie JQuery utilisée)

- <https://jqueryui.com/>
 - Librairie
 - Démonos
 - Documentation
- Sélectionner les composants qui vous intéressent : « custom download »
 - Core
 - Interactions : draggable, droppable, sortable...
 - Widgets : autocomplete, datepicker, tabs ...
 - Effets : animations configurables
- Utiliser le « ThemeRoller » pour choisir le thème désiré



Principe

Cf. exemple « index.html »

```
<head>
```

```
  <link href="jquery-ui.css"
rel="stylesheet">
```

```
</head>
```

```
<script src="external/jquery/jquery.js">
</script>
```

```
<script src="jquery-ui.js"></script>
```

```
<script>
```

```
$( "#accordion" ).accordion();
```

```
var availableTags = [
```

```
  "ActionScript",
```

```
  "AppleScript", ...
```

```
];
```

```
$( "#autocomplete" ).autocomplete({
  source: availableTags,
```

```
  ...
```

```
});
```

```
...
```





Principe

- La plupart des méthodes de traitement d'événement invoquées par les contrôleurs jQuery UI sont destinataires de deux arguments :
 - event : le contexte de l'événement qui intervient
 - ui : un objet donnant accès aux propriétés des éléments manipulés : références sur item/helper/positions de départ...
- Exemple : <http://api.jqueryui.com/sortable/#event-start>



Exercice

- Sélectionner/Organiser les fichiers de la librairie pour minimiser les requêtes à réaliser

```
<html>
<head>
  <link href="js/jquery-ui.min.css" rel="stylesheet">
  <script src="js/jquery.min.js"></script>
  <script src="js/jquery-ui.min.js"></script>
</head>
<body>
...
</body>
</html>
```

*Cf. grunt ou gulp
pour automatiser l'organisation
des librairies côté client*





Exercice

- Ajouter un bouton permettant de supprimer un paragraphe à l'aide d'une croix de la forme « `.ui-icon-close-thick` »
 - Il faudra revoir la structure des paragraphes
- Utiliser un composant « sortable » pour mettre en oeuvre le déplacement des paragraphes
 - Un « handle » de la forme « `.ui-icon-arrow-thick-2-n-s` » devra être utilisé
 - Les endroits où il sera possible de les déplacer devront apparaître
- Empêcher l'activation du mode édition lors d'un clic sur les icônes de déplacement ou de suppression
- Ajouter un bouton au cas d'étude permettant d'activer les déplacements / suppressions





Exercice

- Faire de vos paragraphes éditables un composant
- Le composant ne nécessite pas forcément de base de données pour fonctionner
 - Chaque bloc dans une page peut être rendue « éditable »
 - Les blocs éditables devront pouvoir être déplacés
- Pour configurer une base de données, les chemins d'une API « rest » qui fournit les interactions avec les données doivent être précisés





HTML5 & CSS3



A suivre...

- <http://www.htmlgoodies.com/html5/javascript/a-5-minute-overview-of-all-new-javascript-apis-in-html5.html#fbid=iVPAylzxrmM>
- <http://html5index.org/>
- Objectif clair : RIA & Mobile !
 - Cache
 - Web Sockets
 - Canvas
 - Géoloc...





***Quelques mots
sur ...***



Encodage des caractères

- <http://www.alsacreations.com/astuce/lire/34-charset-iso-8859-1-iso-8859-15-utf-8-lequel-choisir.html>
- Architecture : cf. header.php
 - `header('Content-Type: text/html;charset=iso-8859-15');`
 - `<?xml version="1.0" encoding="iso-8859-15" ?>`
 - `<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-15" />`



POC & Mockups

- POC : Proof Of Concept
 - Réalisation courte incomplète d'un projet pour démontrer sa faisabilité, tester son adhésion
- Mockups / Wireframes / Storyboard
 - Maquette d'une interface utilisateur permettant de présenter les idées sur l'utilisation d'un logiciel
 - Adresse surtout la problématique de l'ergonomie
 - Souvent moins évolué qu'un POC
 - Outils : Balsamiq Mockups, Axure, inVision ...



API Rest

- Ensemble de pages Web côté serveur fournissant un accès à des données en lecture/écriture
- Se base sur les verbes du protocole HTTP
 - POST, GET, PUT, DELETE
 - Pour implémenter les opérations CRUD
- Emploie généralement du JSON pour les échanges de données



Routes & réécriture d'adresses

- Utilisation d'URLs pour passer des paramètres à la place d'une chaîne de requête
 - Améliore le référencement
- Par exemple :
 - `monsite/conversations` \Leftrightarrow `monsite/rest.php?action=getConversations`
 - `monsite/conversation/1/` \Leftrightarrow `monsite/rest.php?action=getConversation&id=1`
- Le serveur apache peut être configuré pour réaliser automatiquement les transformations
 - Le programmeur peut continuer à utiliser les chaînes de requêtes
- Les frameworks backend se basent sur le routage pour organiser les développements
 - À la manière de l'organisation de la page contrôleur avec le paramètre “action”



Push / Pull

- Pull : mécanisme qui fait que le client va régulièrement interroger le serveur pour savoir si des événements se sont produits
 - Ex: nouveaux messages disponibles ...
 - Synonyme de polling : c'est MAL !
 - On consomme des ressources pour rien : HTTP n'est pas fait pour ça !
- Push : mécanisme permettant au serveur d'informer le client qu'un événement s'est produit
 - Google Cloud Messaging sur android
 - WebSockets en HTML5



Applications Single Page

- Application Web ne comprenant qu'une page HTML
 - Consomme des flux de données obtenus à l'aide de requêtes asynchrones pour se mettre à jour
- Limites :
 - Expérience utilisateur % historique de navigation
 - Comment garder une page en favoris ?
 - Comment référencer ?



Les APIs HTML5

- Un langage pour les RIA
 - Objectif = Flash Killer
- Nouveaux objets du navigateur pour s'adapter aux nouveaux usages :
 - Négociation de cache côté client
 - Upload par Drag&Drop
 - WebSocket
 - Canvas...



MVP / Mobile First

- MVP : minimal viable product
 - Méthodologie Lean Startup
 - Se concentrer sur ce qui fait la valeur ajoutée du produit/service que l'on lance, pour tester le plus vite possible auprès de la cible
 - “On a fini quand il n'y a plus rien à ... enlever !”
- Mobile First : commencer par réfléchir aux interfaces pour smartphones/tablettes (plus contraintes) permet d'identifier les fonctionnalités essentielles



RIA / UX / Call To Action

- RIA : Rich Internet Applications
 - Qu'est-ce qui est si riche ? L'ergonomie !
- UX : expérience utilisateur
 - Il y a tellement d'offres que les utilisateurs ne reviendront pas sur votre site s'ils ont eu une mauvaise expérience
 - Latences, non respect des conventions...
 - Ne pas être original ! La meilleure ergonomie est celle qu'on ne voit pas...
- Call To Action : composants d'interaction bien marqués qui incitent à l'action



Responsive design

- Design d'une interface qui s'adapte à différentes résolutions d'écran
 - Un site Web responsive pourra être utilisé sur PC, tablette, smartphone, montre ...
 - Certains composants d'interaction changeront d'apparence ou disparaîtront si la place manque
 - e.g. phpmyadmin
- Vue adaptative : « Ctrl+ Maj+ M » sous Firefox
- Solution technique : css + javascript
 - Twitter Bootstrap



TinyMVC-bootstrap

- Twitter Bootstrap
- Feuilles de style + JS pour rendre un site responsive
- Composants d'interaction
 - <http://www.getbootstrap.com/components/>



Jquery / Jquery UI / Jquery Mobile

- Framework de développement FrontEnd javascript
 - “Write Less, Do More”
 - Basé sur les sélecteurs CSS
- Utilisé comme bibliothèque de base de nombreux composants d'interaction
- Limite : ne contraint pas suffisamment le développeur...
 - On peut vite faire n'importe quoi...
=> Modèles MVC / MV* Frontend : ember.js, angular.js, react.js...



Full Stack Javascript

- Du javascript côté serveur
- Serveur node.js
 - Paradigme de traitement des requêtes “événementiel”
 - Contrairement à apache : nuée de fils
- Effet de mode ?
 - A suivre...



Native / Cross-platform

- Dans le cadre des développements d'applications mobiles
- Native : développement android / IOS / Windows Phone
 - Capacité d'utiliser tous les périphériques du téléphone
 - Vitesse d'exécution
 - MAIS coûts de développement...
- Cross-platform (e.g. phonegap)
 - Un développement unique utilisant du javascript
 - Qui sera intégré dans une application native standardisée
 - Déploiement aisé sur toutes les plateformes



Expressions régulières/rationnelles

- Présent dans tous les langages de programmation
- Motif qui décrit un ensemble de chaînes de caractères possibles selon une syntaxe précise
 - `^ $ [a-z] [^A-Z] . + * {min,max}`
- Javascript :
 - `var oRegExp = /pattern/;`
 - `oRegExp.test(chaine)`
- Php :
 - `preg_match ($pattern , $chaine)`



Un thème de CLOCK ?

Bibliographie

- Bonnes pratiques du Web [SMILE]
 - <http://www.smile.fr/Ressources/Livres-blancs/Culture-du-web/Bonnes-pratiques-du-web>
- Les géants du Web [OctoTechnology]
 - <http://www.geantsduweb.com/>
 - Cf. Mathieu Hausherr



Culturel/Approfondissement



Exercice



Clock