

Technologies du Web

Master COMASIC

Contenus dynamiques côté client

Antoine Amarilli¹

3 octobre 2013

1. Matériel de cours inspiré de notes par Pierre Senellart. Merci à Pablo Rauzy et à Pierre Senellart pour leur relecture.

Motivation

- Nécessité d'avoir un comportement dynamique **côté client**, sans interagir avec le serveur.
- **Cas simples** : validation de formulaire, ouverture de menus...
 - pour aller au-delà de ce que permettent HTML et CSS ;
 - par compatibilité avec les vieux navigateurs ;
 - ou, il y a quelques années, par nécessité.
- **Cas complexes** : écrire des applications complètes pour le navigateur pour remplacer les applications client.
- Solution **la plus répandue** : langage **JavaScript** (ECMAScript). Attention, rien à voir avec Java.
- Quelques **autres technologies** généralement plus isolées du reste de la page : Flash, Java, Silverlight, etc.

Table des matières

- 1 Introduction
- 2 Le langage JavaScript**
- 3 JavaScript et HTML
- 4 AJAX
- 5 jQuery
- 6 Autres technologies

Langage JavaScript

- **ECMAScript** : nom du standard. ECMA-262, 2011, 258 pages.
- JavaScript : la version de Mozilla. JScript : celle d'IE. Par abus de langage : JavaScript.
- JavaScript est un langage de programmation **complet** !
- Langage **interprété** (a priori), haut niveau.
- Typage **dynamique** (typage à l'exécution, typage des valeurs et non des variables... très (trop) grande flexibilité pour les conversions implicites).
- Fonction **eval** pour évaluer du code dynamiquement.

Langage JavaScript (suite)

- Langage **orienté objet**, mais **prototypes** (objets existants clonés) plutôt que des classes.
- Fonctions de **première classe** (fonctions anonymes, clôtures ; les fonctions sont aussi des objets).
- Support des **exceptions**.
- **Syntaxe objet** :
 - `foo.a` pour le membre `a` de l'objet `foo`.
 - `foo.b(arg)` pour appeler la méthode (fonction) `b` de `foo` avec l'argument `arg`.

Exemple de JavaScript

```
function factorial(n) {  
    if (n === 0) {  
        return 1;  
    }  
    return n * factorial(n - 1);  
}
```

Implémentations de JavaScript

IE. Chakra, propriétaire, depuis IE9.

Firefox. SpiderMonkey, libre, depuis 1996 (mais largement amélioré depuis).

Chrome. V8, lancé avec Chrome en 2008. Utilisé aussi par les dernières versions d'Opera.

Safari. JavaScriptCore, renommé en SquirrelFish (2008).

- ⇒ Champ actif de recherche pour améliorer la **performance** !
- ⇒ Compilation **à la volée** ; pas seulement interprété.
- ⇒ Sous-ensembles limités de JavaScript optimisés agressivement (`asm.js`) et utilisables comme cible de compilation (une sorte d'assembleur) pour d'autres langages.

Table des matières

- 1 Introduction
- 2 Le langage JavaScript
- 3 JavaScript et HTML**
- 4 AJAX
- 5 jQuery
- 6 Autres technologies

Intégrer JavaScript à HTML

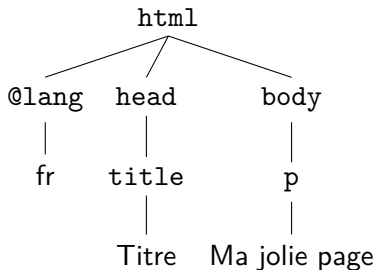
- Un peu comme **CSS** :

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

- **Dépendance externe**, et risque de **sécurité** avec codes tiers !
 - Possibilité de mettre le code **directement** dans `<script>`.
 - **Événements** comme `onclick` (voir plus tard).
 - Aussi, **liens** en javascript:.
- ⇒ Attention : tous les navigateurs ne supportent pas JavaScript !
(En particulier pas les navigateurs textuels et robots.) Il est donc préférable que le site soit **utilisable sans**, si c'est possible.

Document Object Model

```
<html lang="fr">
  <head>
    <title>Titre</title>
  </head>
  <body>
    <p>
      Ma jolie page.
    </p>
  </body>
</html>
```



Objet document

L'objet `document` représente le document `courant`.

`document.documentElement` Renvoie le nœud `racine`.

`document.getElementById("foo")` Renvoie le nœud portant l'attribut `id="foo"` (ou `null` si aucun).

`document.getElementsByTagName("p")` Renvoie un tableau des nœuds qui sont des éléments `<p>`.

Objets Node : bases

Les objets **Node** représentent des **nœuds** du DOM.

node.nodeName Nom de l'élément (BODY...) ou de l'attribut.

node.nodeType Type, comme `Node.ELEMENT_NODE`, aussi `TEXT`, `ATTRIBUTE`, `COMMENT`...

node.nodeValue Valeur des nœuds texte et attribut (aussi : `node.setAttribute("nom", "valeur")` et `node.getAttribute("nom")`).

node.className La classe du nœud.

node.style Le style CSS, avec les différentes propriétés (remplacer les tirets par du camelCase). Par exemple : `node.style.borderStyle = "solid"`.

Objets Node : parcours

`node.parentNode` Le nœud parent.

`node.childNodes` Tableau des nœuds enfants (ou `null` si aucun).

`node.appendChild(child)` Ajoute un enfant (en dernier).

`node.removeChild(child)` Retire un enfant.

`node.cloneNode(true)` Clone le nœud et ses descendants :

- Le nœud cloné n'est pas encore dans le document.
- Attention aux 'id' en double !

`node.cloneNode(false)` Clone le nœud (pas ses descendants).

`node.innerHTML` Représentation HTML du contenu du nœud.

Objet window

Par défaut, les **méthodes** de window sont accessibles directement :

alert("x") Boîte de dialogue indiquant "x".

back() Navigue vers la page précédente.

a = open("URL", "name") Ouvre une fenêtre (ou onglet) sur "URL" avec le nom "name", `a.close()` pour fermer.

confirm("Confirmer ?") Dialogue de confirmation indiquant "Confirmer ?" (voir la valeur de retour).

prompt("Valeur ?", "défaut") Prompt indiquant "Valeur" et prérempli par "défaut" (voir la valeur de retour).

t = setTimeout("f()", 42000) Appeler `f()` dans 42 secondes.
Annuler : `clearTimeout(t)`. Utile pour l'exécution périodique, les animations...

Les **variables globales** sont des **membres** de window.

Événements

- **Attribut** `onfoobar="return f(this)"` sur `x` :
 - ⇒ Lorsque `foobar` survient sur l'élément `x`, appelle `f(x)`.
- **Nombreux événements**. Les plus utiles :
 - `onclick` Lorsqu'un clic survient ; retourner faux annule.
 - `onchange` Lorsque le contenu d'un champ change.
 - `onfocus` Lorsqu'un élément est sélectionné.
 - `onblur` Lorsqu'un élément est désélectionné.
 - `onsubmit` Lorsqu'un `<form>` est soumis ; annulable.
 - `onload` Sur `<body>`, lorsque la page s'est chargée.
 - `ondblclick` Lorsqu'un double-clic survient.
 - `onmouseover` Lorsque l'élément est survolé.
 - `onresize` Lorsque la fenêtre est redimensionnée.
 - `onselect` Sur `<input>`, lorsque du texte est sélectionné.

Exemple : validation de formulaire

```
function vrf() {  
    v1 = document.getElementById("mdp1").value;  
    v2 = document.getElementById("mdp2").value;  
    if (v1 != v2) {  
        window.alert("Erreur de saisie !");  
        return false;  
    }  
}
```

```
<form action="test.php" method="post" onsubmit="return vrf()">  
    <input type="text" name="nom" placeholder="Nom" required><br>  
    <input type="password" name="mdp1" id="mdp1"  
        placeholder="Mot de passe" required><br>  
    <input type="password" name="mdp2" id="mdp2"  
        placeholder="Retaper le mot de passe" required><br>  
    <input type="submit">  
</form>
```


Table des matières

- 1 Introduction
- 2 Le langage JavaScript
- 3 JavaScript et HTML
- 4 AJAX**
- 5 jQuery
- 6 Autres technologies

Requêtes asynchrones

- JavaScript peut aussi faire des **requêtes HTTP**.
- **AJAX** : Asynchronous JavaScript And XML.
- Échanger avec le **serveur** sans charger une nouvelle page.
- **Asynchrone** : les requêtes ne bloquent pas JavaScript.
- Introduit dans **Internet Explorer 5** en 1999, standardisé par le W3C (Working Draft, 2012).
- Problèmes de **sécurité** :
 - **Same origin policy** : pas possible de faire de requêtes à un autre domaine. (Sinon, danger!)
 - Mécanismes pour **autorisations** au cas par cas.
 - Entre **clients**, éditer `document.domain` (historique) ou utiliser `postMessage` (moins dangereux).
 - Avec le **serveur** : Cross-Origin Resource Sharing.

Exemple XMLHttpRequest

```
// Attention, non compatible avec vieilles versions de IE
var req = new XMLHttpRequest();
req.onreadystatechange = function() {
    if (req.readyState === 4) { // est-ce prêt ?
        if (req.status === 200) { // est-ce bon ?
            window.alert("Obtenu : " + req.responseText);
        } else {
            window.alert("Problème avec la requête");
        }
    }
}
// true pour être asynchrone, t pour éviter le cache
var t = new Date().getTime();
req.open("GET", "data.xml?t=" + t, true);
// peut fournir du contenu en POST (à encoder soi-même)
req.send(null);
```

XML

- Généralement, on récupère non pas une page Web mais du contenu **sérialisé** destiné au programme. **Web API**.
- Le code JavaScript se chargera de **traiter** la réponse.
- À l'origine, AJAX se fait avec **XML**, qui est le langage générique qui ressemble à HTML.

```
<?xml version="1.0" ?>  
<etat>  
  <date>2013-10-03</date>  
  <time>13:37</time>  
  <load>0.2</load>  
  <speed>1337 RPM</speed>  
</etat>
```

Exemple de traitement XML

```
var doc = req.responseXML;
var load = doc.getElementsByTagName(
    "load").item(0).firstChild.data;
var speed = doc.getElementsByTagName(
    "speed").item(0).firstChild.data;
document.getElementById("load").textContent = load;
document.getElementById("speed").textContent = speed;
```

JSON

- Format **alternatif** plus récent.
- Normalisé (RFC 4627, 2006, 11 pages).
- Plus **simple**, plus **léger**, plus proche de JavaScript.
- **Types de base** : nombres, booléens, chaînes, null.
- **Types complexes** : listes et dictionnaires.

Exemple JSON

```
{  
  "date": "2013-10-03",  
  "time": "13:37",  
  "load": "0.2",  
  "speed": "1337 RPM"  
}
```

```
var json = JSON.parse(req.responseText);  
document.getElementById("load").textContent = json.load;  
document.getElementById("speed").textContent = json.speed;
```

Communication push

- Si on **attend** un événement du serveur, comment faire ?

HTTP. Non supporté par le protocole.

TCP. Impossible : NAT, firewalls...

Polling. Demander régulièrement une mise à jour.

⇒ Gaspillage de ressources !

Long polling. Ouvrir régulièrement des requêtes, le serveur attend le prochain événement pour répondre.

⇒ Peu efficace, un peu fragile, pas très élégant.

Comet. Techniques historiques, divers hacks.

WebSocket. IETF, RFC 6455, 2011.

Server-Sent Events. Draft W3C, 2013, pas encore dans IE.

⇒ WebSocket supporté par **tous** sauf le navigateur Android.

- **Upgrade** la connexion HTTP vers du bidirectionnel.
- Compliqué : proxys HTTP, caches...

Table des matières

- 1 Introduction
- 2 Le langage JavaScript
- 3 JavaScript et HTML
- 4 AJAX
- 5 jQuery**
- 6 Autres technologies

jQuery

- Lancé en 2006.
- Environ 80 kilo-octets (version 2).
- Gratuit et libre (MIT).
- Utilisé par > 55% des 10 millions de sites les plus visités.²
- Simplifie l'écriture de JavaScript.
- Couche d'abstraction au-dessus des spécificités des différents navigateurs.

2. http://w3techs.com/technologies/overview/javascript_library/all

Exemple jQuery

```
<script type="text/javascript" src="jquery-1.10.2.min.js">
</script>
<script type="text/javascript">
    $.ajax({
        url: "data.json",
        cache: false,
        success: function (data) {
            $( "#load" ).html( data.load );
            $( "#speed" ).html( data.speed );
        }
    });
</script>
```

Table des matières

- 1 Introduction
- 2 Le langage JavaScript
- 3 JavaScript et HTML
- 4 AJAX
- 5 jQuery
- 6 Autres technologies**

Flash

- Lancé en 1996 par Macromedia.
- Longtemps le **seul moyen** (avec Java, cf. après) d'avoir des sons, vidéos, jeux interactifs, etc., dans le navigateur.
- Certains sites **uniquement** en Flash (heureusement rare) :
accessibilité, indexation ?
- Aussi utilisé pour des **dessins animés**, voire des **jeux** sur PC.
- Utilisé par **17%** des 10 millions de sites les plus populaires.³
(5% perdus de septembre 2012 à septembre 2013).
- Support estimé : **95 à 99%** des clients PC.⁴
- En pratique, important pour la compatibilité de HTML5
(<video>...) vers les vieux navigateurs.

3. http://w3techs.com/technologies/overview/client_side_language/all

4. Source : *The Tangled Web*, chapitre 8, 2012.

Inconvénients de Flash

- Environnement de développement **payant**. (Aussi : **Haxe**.)
- Client officiel **gratuit mais propriétaire**, Windows et MacOS.
- Fourni **avec Chrome** sous GNU/Linux.
- Autres navigateurs sous GNU/Linux : plug-in **plus à jour**.
- Support GNU/Linux toujours de **mauvaise qualité**.
- Clients **libres** en développement (format partiellement documenté) mais aucun n'est encore fonctionnel.
- Historiquement, nombreuses **failles** de sécurité du client...
- Nettement **moins crucial** depuis HTML5 et consorts.
- Essentiellement **pas de support** sur mobile. Avenir ?

Java

- Java : Langage de programmation **orienté objet**, compilé vers une machine virtuelle, utilisée pour des **applications complètes**. Très répandu, assez lourd.
- Possibilité de faire tourner des **applications Java** dans une zone de taille fixe dans une page HTML, depuis 1995.
- **Bac à sable** : par défaut, l'application ne peut pas faire n'importe quoi (contenu du disque dur, micro...).
- **Historiquement** populaire sur les mobiles.
- **Pas de support** des applets sur smartphone.
- Utilisé par **0.1%** des 10 millions de sites les plus populaires.⁵
- Support estimé : **80%** des utilisateurs.⁶

5. <http://w3techs.com/technologies/details/cp-javaruntime/all/all>

6. Source : *The Tangled Web*, chapitre 8, 2012.

Microsoft

VBScript Antiquité, concurrent malheureux de JavaScript.

ActiveX Lancé en 1996. **Windows** et **IE** seulement (en gros).
Failles. **Désactivé par défaut** depuis IE 9.

XBAP Obscur équivalent Microsoft .NET des applications
Java. Échec, **désactivé par défaut** depuis IE 9.

Silverlight

- Lancé en **2006**, similaire à **Flash**.
- Implémentation officielle **propriétaire** pour Windows et Mac OS (certains navigateurs).
- **Pas de support** sur mobile.
- Implémentation libre **Moonlight** abandonnée.
- 0.2% des 10 millions de sites plus populaires.⁷
- Support estimé : 65% des utilisateurs.⁸

7. <http://w3techs.com/technologies/details/cp-silverlight/all/all>

8. <https://en.wikipedia.org/wiki/Silverlight#Adoption>

Futur

<canvas> Région de la page dans où on peut **dessiner** avec JavaScript. Pas vectoriel. Utile pour les **jeux vidéo** (à la place de Flash).

SVG+JS Manipulation de graphiques **vectoriels** en JavaScript.

⇒ **Exemple** :

`http://isthis4real.com/orbit.xml`

WebGL API JavaScript pour faire de la **3D** accélérée par la carte graphique. Expérimental... Risques de sécurité ?

NaCl Google Native Client, exécution de **code natif** dans le navigateur avec bac à sable. Peu répandu, Chrome uniquement.

Hors-ligne Applications Web utilisables hors ligne.