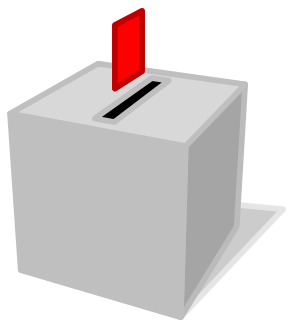
INF344 – Données du Web
Javascript et contenus dynamiques côté client

Antoine Amarilli

SONDAGE : Technologies côté client

Quel est l'intrus ?

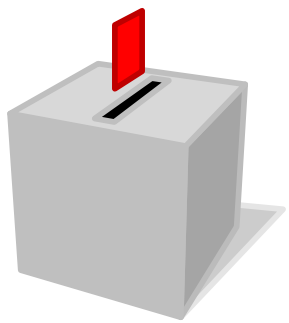
- **A**: Java
- **B**: JS
- **C**: ECMAScript
- **D**: JavaScript



SONDAGE : Technologies côté client

Quel est l'intrus ?

- **A**: **Java**
- **B**: JS
- **C**: ECMAScript
- **D**: JavaScript



Motivation

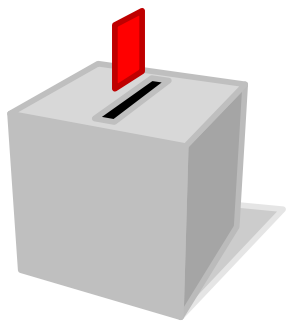
- Avoir un comportement dynamique **côté client**,
  sans interagir avec le serveur
- **Cas simples** : validation de formulaire, ouverture de menus…
  - → pour aller au-delà de ce que permettent HTML et CSS
  - → par compatibilité avec les vieux navigateurs
  - → ou, il y a quelques années, par nécessité
- **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** !
- **Autres technologies** anciennes : Flash, Java, Silverlight, etc.

Table des matières

SONDAGE : Utilisation de Javascript

Quelle proportion des 10 millions de sites Web les plus visités utilise Javascript ?

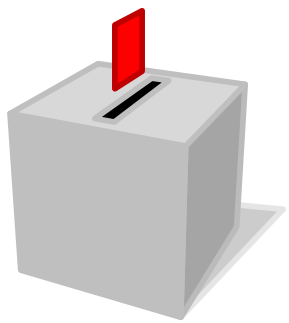
- **A**: Moins de 90%
- **B**: Entre 90% et 95%
- **C**: Entre 95% et 98%
- **D**: Plus de 98%



SONDAGE : Utilisation de Javascript

Quelle proportion des 10 millions de sites Web les plus visités utilise Javascript ?

- **A**: Moins de 90%
- **B**: **Entre 90% et 95%**
- **C**: Entre 95% et 98%
- **D**: Plus de 98%



Langage JavaScript

- **ECMAScript** : nom du standard (ECMA-262, 2011, 258 pages)
- 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

Versions de JavaScript

- 1999 : ECMAScript 3, ajout des **regexps**, **try/catch**
- 2009 : ECMAScript 5, ajout du **mode strict**
- ES2015 : ajout des **classes**, **itérateurs**, **générateurs**, **promesses**, **fonctions fléchées**
- ES2017 : ECMAScript 8, ajout des **atomics**, de **async**/**await**, etc.
- ES2018 : boucles asynchrones, extensions pour les promesses, extensions pour les regexps (e.g., groupes de capture nommés)

→ **Polyfills :** implémenter des nouvelles fonctionnalités dans d'anciennes versions du langage pour les vieux navigateurs

→ **Guides de style** pour JavaScript, e.g., Google [1], AirBNB [2]

1. `https://google.github.io/styleguide/javascriptguide.xml`
2. `https://github.com/airbnb/javascript`

Fonctionnalités de JavaScript

- 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`

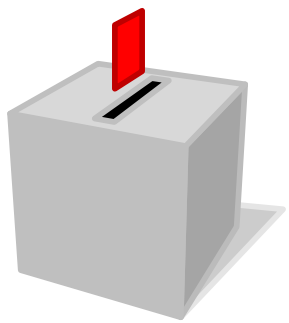
Exemple de JavaScript

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

SONDAGE : Moteur Javascript de Node.js

Le moteur Javascript utilisé par Node.js est celui de...

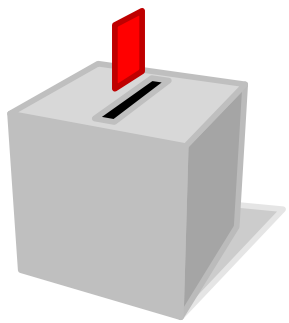
- **A**: Internet Explorer
- **B**: Chrome
- **C**: Firefox
- **D**: Safari



SONDAGE : Moteur Javascript de Node.js

Le moteur Javascript utilisé par Node.js est celui de...

- **A**: Internet Explorer
- **B**: **Chrome**
- **C**: Firefox
- **D**: Safari



Implémentations de JavaScript

IE Chakra, propriétaire, depuis IE9

Firefox SpiderMonkey, libre, depuis 1996

Chrome V8, lancé avec Chrome en 2008

→ Utilisé aussi par Opera et Node.js

Safari JavaScriptCore, renommé en Nitro (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`, WebAssembly) et utilisables comme cible de compilation

Dialectes modernes de Javascript

- **TypeScript :** Javascript avec du typage statique optionnel
  - → Le langage principal du framework **Angular**
- Également **CoffeeScript**, qui ajoute du sucre syntaxique (filtrage, etc.)
- Ces langages peuvent être **transpilés** vers Javascript

Table des matières

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:`

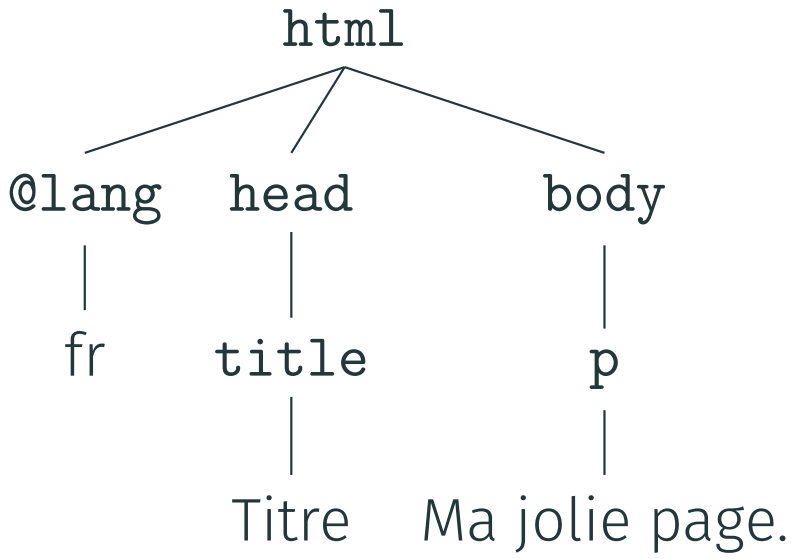
→ Attention : tous les navigateurs ne supportent pas JavaScript !

  → En particulier pas les navigateurs textuels et robots

→ 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

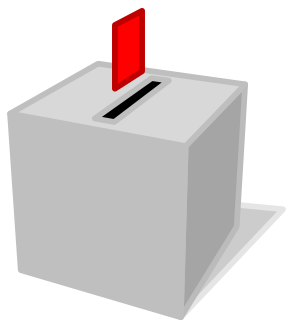
Requêtes asynchrones

- 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

SONDAGE : Introduction d'AJAX

Quel navigateur Web a introduit AJAX ?

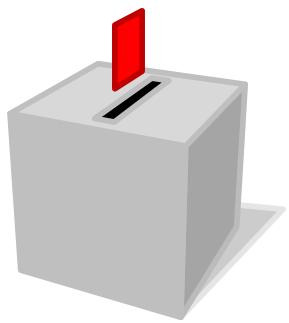
- **A**: Netscape
- **B**: Opera
- **C**: Firefox
- **D**: Internet Explorer



SONDAGE : Introduction d'AJAX

Quel navigateur Web a introduit AJAX ?

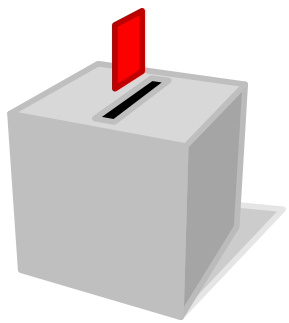
- **A**: Netscape
- **B**: Opera
- **C**: Firefox
- **D**: **Internet Explorer**



SONDAGE : Sécurité d'AJAX

Du code AJAX qui tourne sur une page peut faire des requêtes vers...

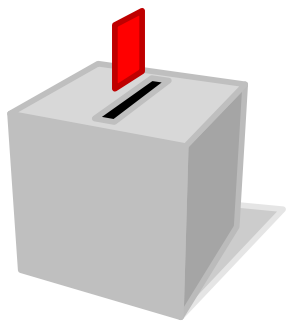
- **A**: Seulement la même page
- **B**: Seulement le même site Web
- **C**: N'importe quel site Web
- **D**: C'est plus compliqué



SONDAGE : Sécurité d'AJAX

Du code AJAX qui tourne sur une page peut faire des requêtes vers...

- **A**: Seulement la même page
- **B**: Seulement le même site Web
- **C**: N'importe quel site Web
- **D**: **C'est plus compliqué**



Historique et problèmes d'AJAX

- 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, cross-document messaging (`postMessage`)
    - → 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>
```

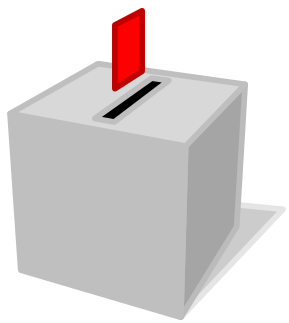
Exemple de traitement XML

```
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;
```

SONDAGE : JSON

La spécification JSON fait...

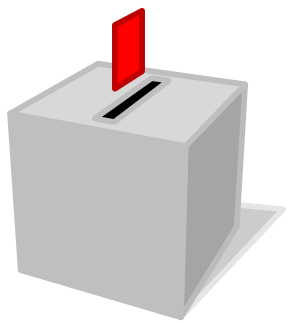
- **A**: Moins de 20 pages
- **B**: Entre 20 et 50 pages
- **C**: Entre 50 et 200 pages
- **D**: Plus de 200 pages



SONDAGE : JSON

La spécification JSON fait...

- **A**: **Moins de 20 pages**
- **B**: Entre 20 et 50 pages
- **C**: Entre 50 et 200 pages
- **D**: Plus de 200 pages



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 !

Comet Ouvrir régulièrement des requêtes, le serveur attend le prochain événement pour répondre

→ Peu efficace, un peu fragile, pas très élégant

WebSocket IETF, RFC 6455, 2011

- Upgrade la connexion HTTP en bidirectionnel
- Rétrocompatibilité complexe (proxys HTTP, caches)
- Librairie Javascript : socket.io

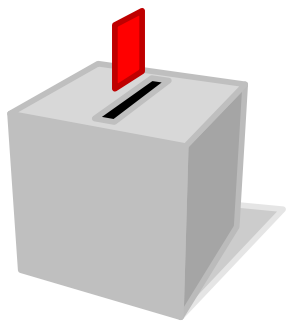
Server-Sent Events Draft W3C, 2013, toujours pas dans IE

Table des matières

SONDAGE : Parts de marché de jQuery

Quelle proportion des 10 millions de sites Web les plus visités utilise jQuery ?

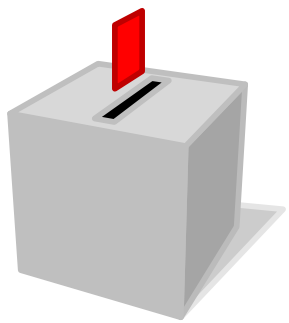
- **A**: Moins de 25%
- **B**: Entre 25% et 50%
- **C**: Entre 50% et 75%
- **D**: Plus de 75%



SONDAGE : Parts de marché de jQuery

Quelle proportion des 10 millions de sites Web les plus visités utilise jQuery ?

- **A**: Moins de 25%
- **B**: Entre 25% et 50%
- **C**: **Entre 50% et 75%**
- **D**: Plus de 75%



jQuery

- Lancé en 2006
- Environ 80 kilo-octets (version 2 ou 3)
- Gratuit et libre (MIT)
- Utilisé par $>$ 73% des 10 millions de sites les plus visités.[3]
- Simplifie l'écriture de JavaScript
- Couche d'abstraction au-dessus des spécificités des différents navigateurs

3. `https://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>
```

Bibliothèques de fonctions utilitaires générales

- Backbone.js, framework frontend léger
- Underscore.js, bibliothèque d'utilitaires pour Backbone.js
- Lodash, successeur de Underscore.js
- Modernizr, teste quelles fonctionnalités sont supportées par le navigateur
- etc.

Angular

- **AngularJS** (version 1) : librairie Javascript
- **Angular** (version 2) : plateforme basée sur **TypeScript**
- **Idée de base :** mettre à jour la vue de façon **déclarative** en spécifiant le lien entre modèle et vue :
  - $\rightarrow$ quand une valeur change, mettre à jour la vue
  - $\rightarrow$ quand l'utilisateur change la vue, mettre à jour la valeur

Templates

- **Idée :** instancier des morceaux de HTML avec des valeurs de variables fournis par Javascript
- Divers langages, e.g., **Mustache**, **Handlebars**,

```
<script id="entry-template" type="text/x-handlebars-template">
  <div class="entry">
    <h1>{{title}}</h1>
    <div class="body">
      {{body}}
    </div>
  </div>
</script>
```

React.js (Facebook)

- Permet de créer du HTML directement dans Javascript avec JSX

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

- DOM virtuel : ne pas manipuler le vrai DOM mais une copie (plus rapide), et mettre à jour le vrai DOM avec un diff d'arbres

→ Autres alternatives : Redux, Vue.js, Ember.js

Librairies de composants d'interface graphique (*widgets*)

- Collections : jQuery UI, jQWidgets, Polymer, Angular Material, etc.
- Édition riche de texte : TinyMCE
- Tooltips : Popper
- Visualisation de données : D3.js
  - → Lier des données à une visualisation
  - → Définir des règles pour mettre à jour la vue quand des éléments sont ajoutés / modifiés / supprimés
- Également : Web Components (spécification en cours de finalisation)
  - Support des templates et des includes
  - Shadow DOM pour limiter la portée des éléments inclus

Outils de développement

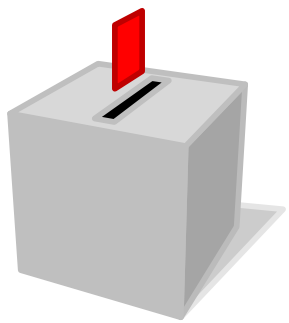
- Outils Web Developer dans les navigateurs Web
- JSFiddle : partager et tester des morceaux de Javascript
- Github gists

Table des matières

SONDAGE : Utilisation de Flash

Quelle proportion des 10 millions de sites Web les plus visités utilise encore Flash ?

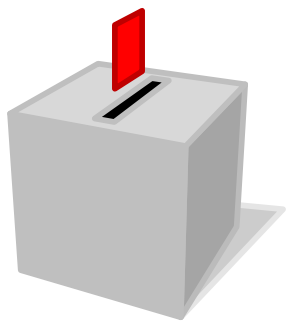
- **A**: Moins de 5%
- **B**: Entre 5% et 10%
- **C**: Entre 10% et 20%
- **D**: Plus de 20%



SONDAGE : Utilisation de Flash

Quelle proportion des 10 millions de sites Web les plus visités utilise encore Flash ?

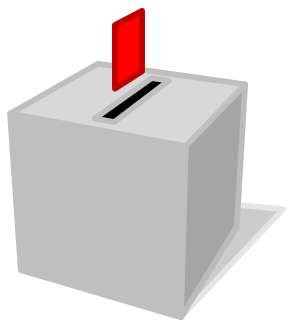
- **A**: **Moins de 5%**
- **B**: Entre 5% et 10%
- **C**: Entre 10% et 20%
- **D**: Plus de 20%



SONDAGE : Support de Flash

Quelle proportion des navigateurs supporte encore Flash ?

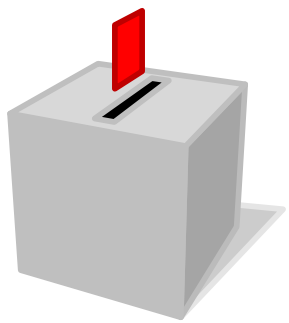
- **A**: Moins de 25%
- **B**: Entre 25% et 50%
- **C**: Entre 50% et 75%
- **D**: Plus de 75%



SONDAGE : Support de Flash

Quelle proportion des navigateurs supporte encore Flash ?

- **A**: **Moins de 25%**
- **B**: Entre 25% et 50%
- **C**: Entre 50% et 75%
- **D**: Plus de 75%



Vieilles technologies (non supportées sur mobile)

- **Flash :** sorti en **1996** par Macromedia (racheté par Adobe), pour les vidéos, jeux vidéos, etc.
  - Encore utilisé sur **3%** des sites les plus populaires [4]
  - En 2019, n'est plus supporté que par **21%** des navigateurs [5]
  - Pas de support sur **mobile**, officiellement déprécié en **2017**
- **Applets Java :** sorti en **1995**, applications riches mais qui n'interagissent pas avec le reste de la page : seulement sur $<$ **0.1%** des sites les plus populaires [6]
- Vieilles **technologies Microsoft** : ActiveX, Silverlight, etc.

---

4. `https://w3techs.com/technologies/overview/client_side_language/all`
5. `https://www.zdnet.com/pictures/2019s-tech-security-and-authentication-trends/2/`
6. `https://w3techs.com/technologies/details/cp-javaruntime/all/all`

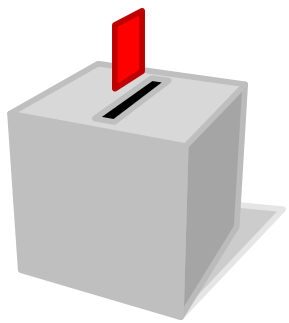
Nouveautés

`<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
WebGL	API JavaScript pour faire de la 3D accélérée par la carte graphique. Expérimental… Risques de sécurité ?
WebRTC	Communications vocales et vidéo entre navigateurs
Hors-ligne	Applications Web utilisables hors ligne

SONDAGE : WebVR

À quoi sert l'API WebVR ?

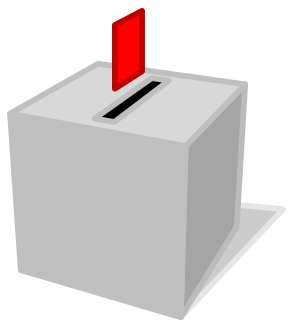
- **A**: À permettre une interaction entre Vue.js et React.js
- **B**: À définir des scènes de réalité virtuelle
- **C**: À piloter des drones depuis des pages Web
- **D**: À spécifier le rendu visuel des pages



SONDAGE : WebVR

À quoi sert l'API WebVR ?

- **A**: À permettre une interaction entre Vue.js et React.js
- **B**: **À définir des scènes de réalité virtuelle**
- **C**: À piloter des drones depuis des pages Web
- **D**: À spécifier le rendu visuel des pages



Développement Web hors du Web

- Créer des **applications Web** et les porter vers d'autres **platformes**
  - **Apache Cordova** pour le mobile : mettre l'app dans une **Web View**
  - Aussi : **React Native**
  - **Firefox OS** (abandonné)
    - Fork **KaiOS** : deuxième plateforme mobile la plus populaire en Inde (derrière Android, devant iOS)[7]
  - **Electron :** fournir l'app avec Node.js et Chromium

→ Advantage : portabilité (pas besoin de développer plusieurs apps)

→ Inconvénients : performances

7. `https://www.androidauthority.com/kaios-usa-india-958519/`

Crédits

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