

HTML5

Historique

Historique

- HTML4 en 1999, beaucoup de changements depuis...
- Coopération entre le World Wide Web Consortium (W3C) et le Web Hypertext Application Technology Working Group (WHATWG).
- WHATWG travaillait sur de nouveaux formulaires et applications, le W3C sur XHTML 2. En 2006, ils ont décidé de coopérer pour créer HTML5.
- Forte implication des industriels et fabricants de navigateurs

Historique (2)

- Des “règles” ont été établies
 - Toutes les nouvelles fonctionnalités seront basées sur HTML, CSS, DOM, et JavaScript
 - Réduire au maximum la nécessité de plugins externes comme Flash,
 - Meilleure gestion des erreurs
 - Essayer de remplacer par des tags ce qui se fait en masse avec du script (menus, etc)
 - HTML5 doit supporter plusieurs devices (mobiles...)
 - Le processus de développement doit être ouvert au public

**Les nouveautés ! Des démos
avant d'aller plus loin !**

Ressources pour démos, tutos et exemples

- Page de ressources HTML5 régulièrement mise à jour :
 - <http://tinyurl.com/7px5gpe>
- **Autres démonstrations utilisées**
 - <http://slides.html5rocks.com/>
 - <http://www.w3.org/2010/Talks/0430-www2010-plh/>
 - <http://www.htmlfivewow.com>
 - <http://io-2011-html5-games-hr.appspot.com/#1>

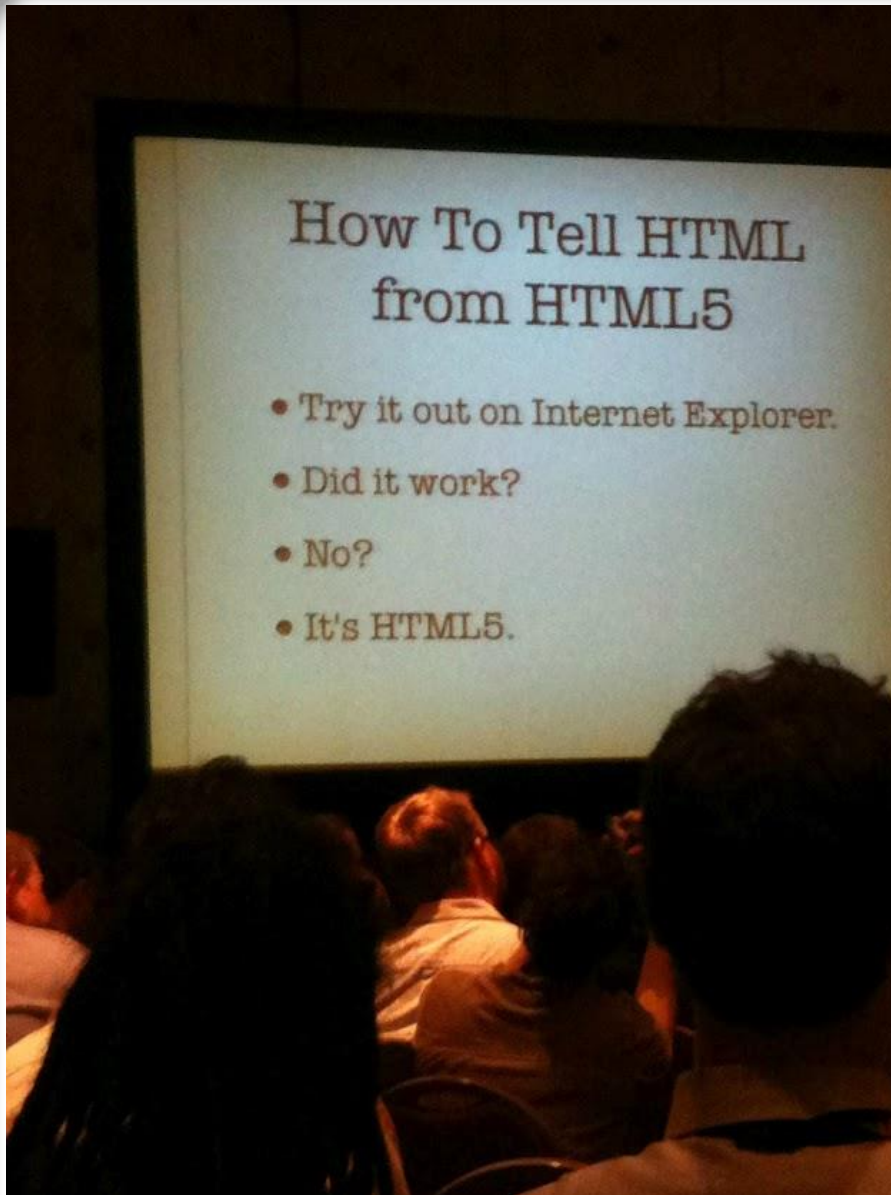
Supports de HTML5 dans les navigateurs

- Tout n'est pas encore supporté, voir :
 - <http://caniuse.com/> : tables de compatibilité régulièrement mises à jours,
 - <http://html5please.us/> : donne l'état des lieux pour chaque feature + indique des solutions de repli,
 - <http://html5test.com/> : test interactif de votre navigateur,
 - http://www.quirksmode.org/html5/inputs_mobile.html
- Pour cours nous testerons sur des versions récentes des navigateurs
- Nous verrons aussi quelques outils permettant d'assurer la compatibilité avec d'anciens navigateurs

Supports de HTML5 dans les navigateurs

- Les navigateurs en pointe : Opera (un des développeurs d'Opera est à l'origine de HTML5), ceux basés sur WebKit (Chrome, Safari, y compris sur mobiles), Firefox (un peu de retard)
- Et IE ?

Internet Explorer...



- Ca va changer : IE 10 beta promet un très bon support,
- Grosse activité HTML5 chez Microsoft,
- Bureau Windows 8 = un navigateur
- Solutions de repli pour vieux IE...

Structure générale d'un document HTML5

Un document minimal

```
<!doctype html>
```

```
<html lang="fr">
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<title>Titre de la page</title>
```

```
<link rel="stylesheet"
```

```
href="style.css">
```

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

```
</head>
```

```
<body>
```

```
... <!-- Le reste du contenu --> ...
```

```
</body>
```

```
</html>
```

Remarques

- Doctype simple à mémoriser,
- Plus de « type= » dans la balise <link>, utilisation de « rel= » à la place
 - Nombreuses valeurs possibles pour rel : stylesheet, author, icon, prefetch, next, prev, etc.
- Plus de « type= » dans la balise <script> non plus
 - <script src="script.js"></script> suffit...

Formulaires

Formulaires HTML5

- Enorme évolution, alors que les formulaires n'avaient quasiment pas changé depuis 1997
- Ajout de nombreux validateurs qui auparavant nécessitaient du JavaScript,
- Ajout de nombreux types nouveaux pour les champs de formulaire, auparavant fournis par des bibliothèques JavaScript

Les nouveaux champs <input>

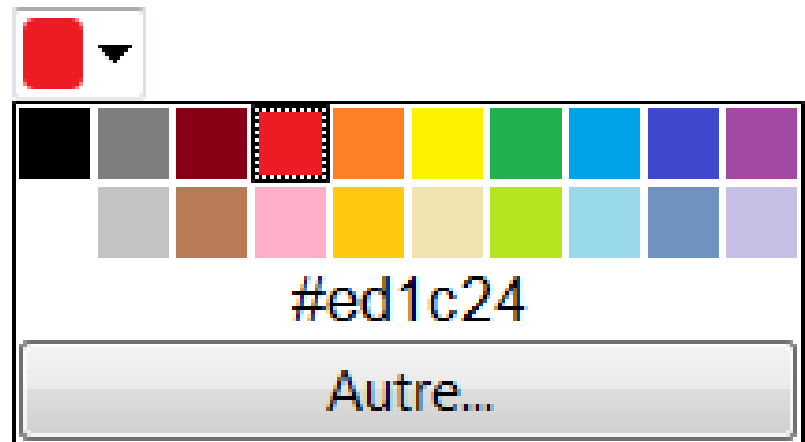
- De nombreux nouveaux types de champs <input> ont été introduits, tous ne sont pas encore supportés, même par les versions les plus récentes des navigateurs
- color, date, datetime, datetime-local, email, month, number, range, search, tel, time, url, week
- Nous allons voir des exemples de chacun de ces champs

Le champ <input type=color>

- Choisissez une couleur: <input type="color" name="favcolor" />
- Supporté aujourd'hui uniquement dans Opera

Select your favorite color:

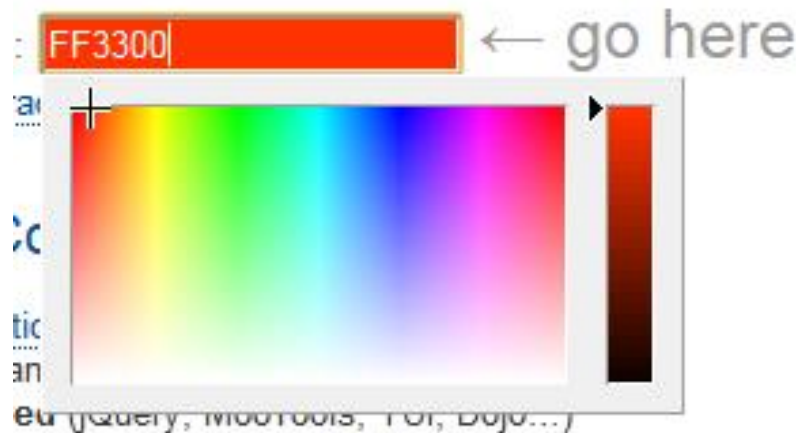
Soumettre



Démo

Le champ <input type=color>

- Solution de repli : la librairie jsColor (cf <http://jscolor.com/>, gratuit)
- <script src="jscolor/jscolor.js"></script>
- <input **class="color"**>



<input type=date>

- Anniversaire: <input type="date" name="bday" />

:

←

Février

→

2012

⬆

⬇

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	1	2	3	4
5	6	7	8	9	10	11

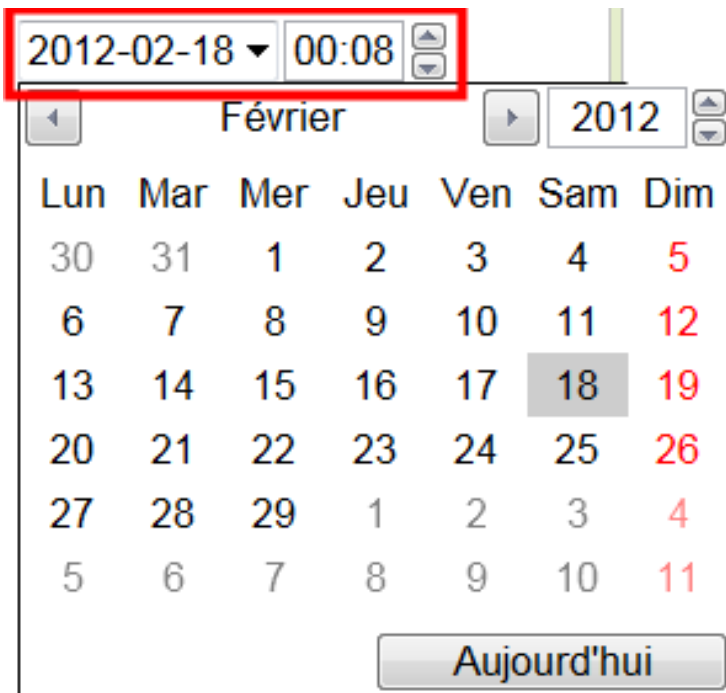
Aujourd'hui

Démonstration

- Marche bien dans Opera, support partiel Chrome, Safari, adapte le clavier sur mobiles
- Solution de repli : lib javascript jQuery UI, Dojo, etc.

<input type=datetime-local>

- Permet de choisir la date **et l'heure**
- Support idem type=date
- Anniversaire : <input **type="datetime-local"** name="bdaytime" />



The image shows a web browser's datetime-local input field. The input field is highlighted with a red rectangle and contains the text "2012-02-18" followed by a dropdown arrow and "00:08" followed by up/down arrows. Below the input field is a calendar picker for the month of February 2012. The calendar shows days of the week (Lun, Mar, Mer, Jeu, Ven, Sam, Dim) and dates. The date 18 is highlighted in grey. At the bottom of the calendar is a button labeled "Aujourd'hui".

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	1	2	3	4
5	6	7	8	9	10	11

Aujourd'hui

Démonstration

<input type=datetime>

- Permet de choisir la date, l'heure **et le fuseau horaire**
- Support idem type=date



The screenshot shows a web form with a datetime input field. The field is divided into three parts: a date selector, a time selector, and a time zone selector. The date selector shows '2012-02-18'. The time selector shows '00:03'. The time zone selector shows 'UTC'. Below the input field is a calendar for February 2012. A red arrow points from the '00:03' time selector to the calendar. The calendar shows the days of the week (Lun, Mar, Mer, Jeu, Ven, Sam, Dim) and the dates (30, 31, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11). The date '18' is highlighted. At the bottom of the calendar is a button labeled 'Aujourd'hui'.

type="datetime"
name="bdaytime" />

Démonstration

<input type=time>

- Heure du rendez-vous :
<input type="time" name="rdv_time" />

Select a time:

- Supporté par Opera pour le moment

<input type=week>

- Choisissez la semaine :

<input **type="week"** name="no_semaine" />

Select a week:

Semair	Lun	Mar	Mer	Jeu	Ven	Sam	Dim
5	30	31	1	2	3	4	5
6	6	7	8	9	10	11	12
7	13	14	15	16	17	18	19
8	20	21	22	23	24	25	26
9	27	28	29	1	2	3	4
10	5	6	7	8	9	10	11

Aujourd'hui

Select a week: 2012-W07 ▼

- Supporté par Opera pour le moment...

<input type=month>

- <input **type="month"** name="bdaymonth" />
- Idem date mais permet de choisir juste le mois
- Support Opera aujourd'hui

2012-02 ▾ Soumettre

Février 2012						
Lun	Mar	Mer	Jeu	Ven	Sam	Dim
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	1	2	3	4
5	6	7	8	9	10	11

Aujourd'hui

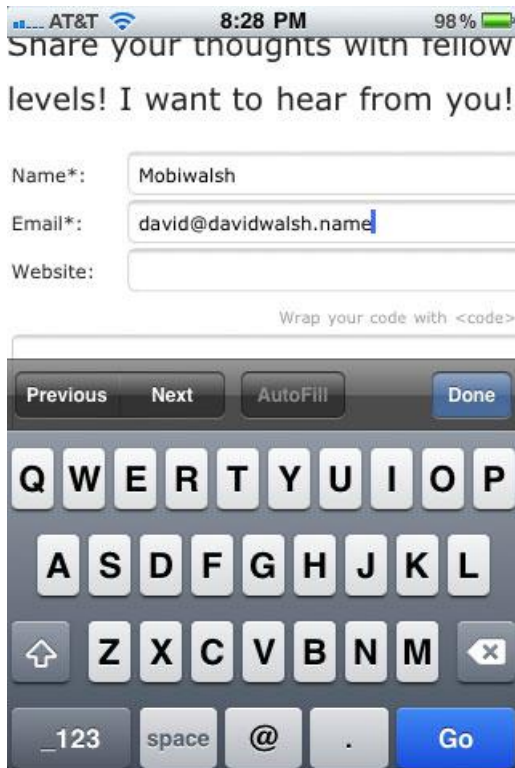


Birthday (month and year): 2012-02 ▾ Soumettre

Démonstration

<input type=email>

- E-mail: <input **type="email"** name="useremail" />
- Intérêt : poppe un clavier contextualisé sur mobile ou tablette



- Attributs implicites “required”, “invalid” etc.
- Possibilité de styler la saisie
- Vrai aussi pour les autres champs <input>

<input type=email> validation

- Validation + messages d'erreurs. Parfois tooltips lors de la saisie

E-mail: ds

Veuillez saisir une adresse courriel valide.

- Et lors de la soumission

E-mail: ds

Veuillez saisir une adresse courriel valide.

[Démonstration](#)

<input type=email>

```
<!DOCTYPE html>
<html>
<head>
<style>
  [required] {
    border-color: #88a;
    -webkit-box-shadow: 0 0 10px rgba(0, 0, 255, .8);
  }

  :invalid {
    border-color: #e88;
    -webkit-box-shadow: 0 0 10px rgba(255, 0, 0, .8);
  }
</style>
</head>
<body>
<form action="demo_form.asp">
  Email: <input type="email" name="email" />
  <input type="submit" />
</form>
</body>
</html>
```

Démonstration

<input type=email>, attribut multiple

- On peut spécifier une liste de valeurs possibles
- Prend en compte la validation

```
<input type="email" multiple value="foo@bar.org, bob@sponge.org">
```

<input type=number>

- <input **type="number"** name="quantity" min="1" max="5" />

Quantity (between 1 and 5):  

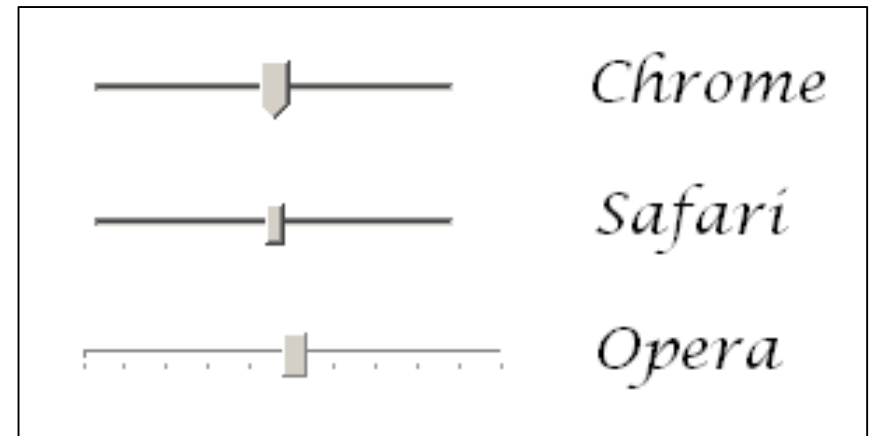
- Attributs possibles : max, min, step, value (valeur par défaut)
- Clavier contextuel sur mobiles, validations idem type=email
- Support Opera, Chrome, Safari...
- [Démonstration](#)

<input type=range>

```
<input type="range" name="n" min="1" max="10" />
```

Points: 

- Attributs : idem que ceux de type=number
- Poppe un clavier contextuel sur mobile
- Support idem (Opera, Chrome, Safari)
- Démonstration



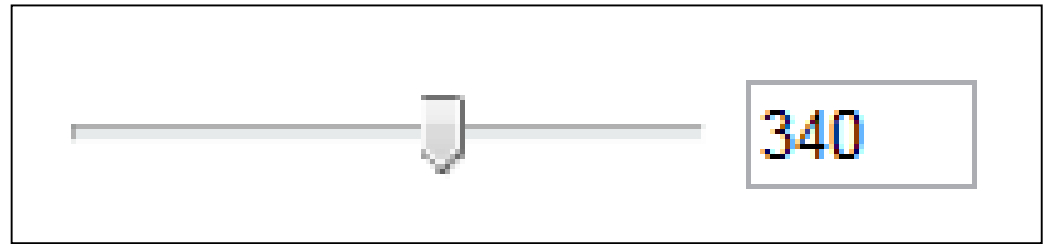
<input type=range>

■ Exemple avec feedback

```
<input id="slider1" type="range" min="100" max="500" step="10"
      onchange="printValue('slider1','rangeValue1')"/>
<input id="rangeValue1" type="text" size="2"/>
```

Et une fonction JavaScript

```
<script>
function printValue(sliderID, textbox) {
    var x = document.getElementById(textbox);
    var y = document.getElementById(sliderID);
    x.value = y.value;
}
</script>
```



Démonstration (Chrome, Opera)

<input type=tel>

- Telephone: <input **type="tel"** name="usrstel" />
- Intérêt principal : tablettes, smartphones !



<input type=tel>

- Possibilité de mettre des patterns et un placeholder

```
<input type="tel" placeholder="(555) 555-5555"  
pattern="^\(?\d{3}\)?[-\s]\d{3}[-\s]\d{4}.*?$" />
```

- Utile via les styles ou de la gestion d'événement JavaScript pour la validation

<input type=url>

Homepage: <input **type="url"** name="homepage" />

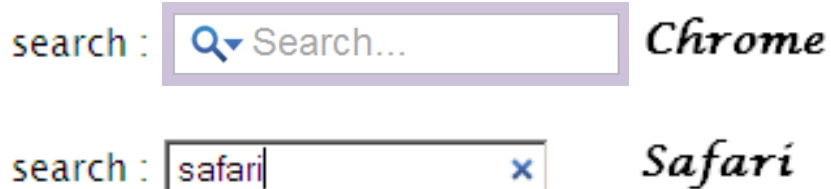


- Idem, soumis à validation par défaut, peuvent être stylés (required, invalid)

<input type=search>

```
<label for="mysearch2">Enter your  
search string here : </label>
```

```
<input id="mysearch2" type="search"  
placeholder="search">
```



Démonstration

- Meilleur affichage dans Chrome et Safari, petite croix pour vider le champs, possibilité de mettre un “placeholder” (val par défaut grisée), attribut “result”

Reconnaissance vocale (expérimental)

- Veuillez parler : `<input type="text" x-webkit-speech name="search_value " />`
- Marche dans les navigateurs à base de WebKit (Chrome, Safari)



- Démonstration

Nouveaux éléments de formulaires

- HTML5 propose de nouveaux éléments utiles aux formulaires
- `<output>`, `<datalist>` et `<keygen>`
- `<output>` évite de faire certains feedbacks qu'on faisait auparavant en JavaScript,
- `<datalist>` permet de définir des données utilisées par des formulaires (pour de la suggestion/autocomplétion par exemple),
- `<keygen>` sert à proposer un menu avec les types de cryptage supportés par le browser

Le champ <output>

```
<form oninput="o.value=a.value*b.value">  
<input name="a" value="2" type="number"> x  
<input name="b" value="3" type="number"> =  
<output name="o">6</output>  
</form>
```

- [Démonstration1](#) et [Démonstration2](#)
- Sitôt qu'on a saisi les deux premières valeurs le champ <output> est mis à jour, pas besoin de JavaScript !

x = 6

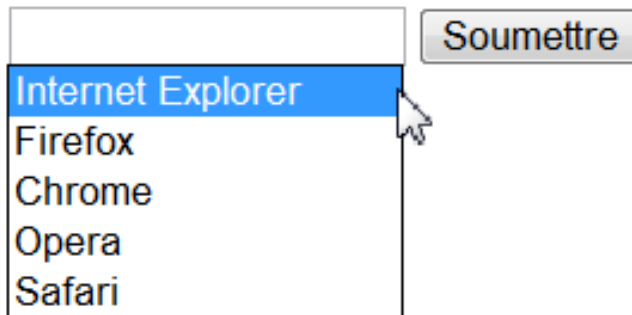
Le champ <datalist>

- Cette balise sert à créer une liste de suggestions associée à un champ (comme dans Google Suggest).
- L'id du tag <datalist> doit être identique à la valeur de l'attribut "list " du champ <input> associé.

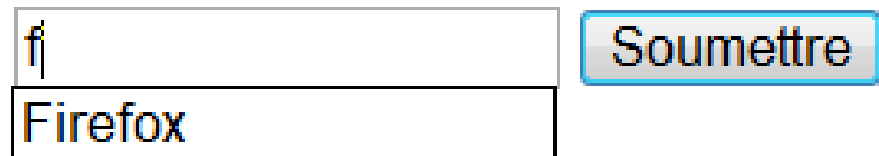
Le champs <datalist>, exemple

```
<form action="demo_form.asp" method="get">
  <input list="browsers" name="browser" />
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
  <input type="submit" />
</form>
```

- Le champs de saisie va proposer une liste avec de l'auto-complétion :



A screenshot of a web form. On the left, there is a text input field with a dropdown menu open, displaying a list of browser names: Internet Explorer, Firefox, Chrome, Opera, and Safari. A mouse cursor is pointing at the 'Internet Explorer' option. To the right of the dropdown is a 'Soumettre' (Submit) button.



A screenshot of a web form. On the left, there is a text input field containing the text 'f'. Below the input field, a dropdown menu is open, showing the word 'Firefox' in a larger font. To the right of the dropdown is a 'Soumettre' (Submit) button.

<datalist>

- Supporté par : Firefox, Opera, IE10
- [Démonstration1](#) et [Démonstration2](#)
- On peut utiliser une même liste avec plusieurs champs de plusieurs formulaires,
- Pour les navigateurs qui ne le supportent pas, on peut utiliser un <select> dans le <datalist> + du JavaScript

L'élément <keygen>

- <keygen> sert à la génération de clés publiques-privées, en particulier pour que le navigateur puisse générer ces clés parfois nécessaires à certains serveurs ou certificats.
- Pose problème à Microsoft (voir cette [discussion](#)), supporté par les autres...

<keygen> exemple

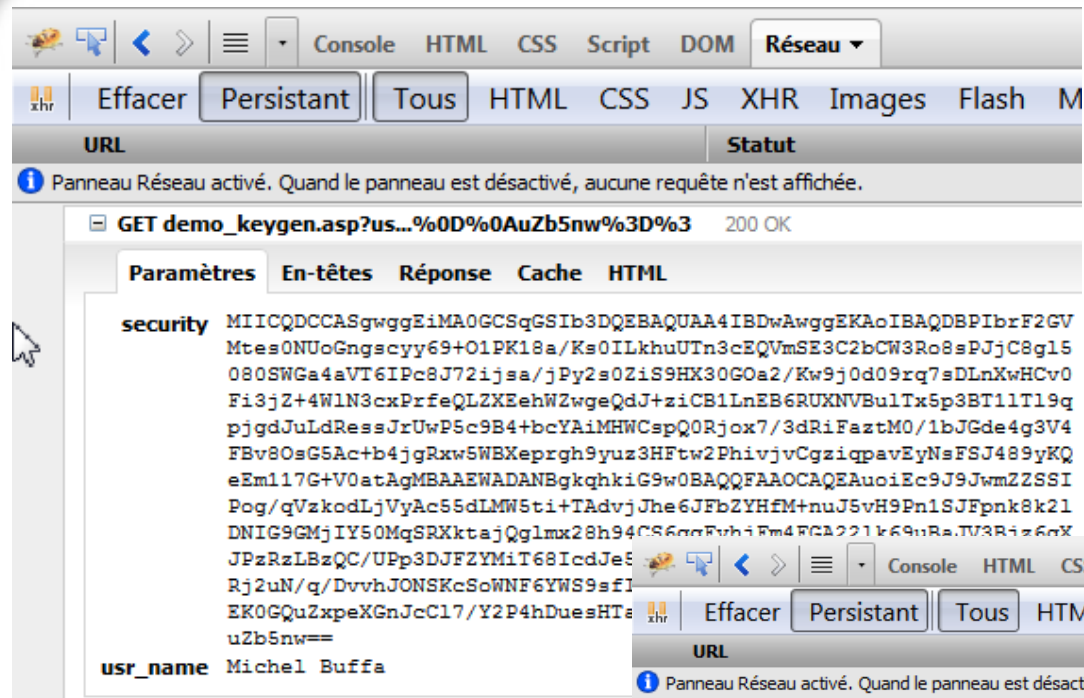
```
<form action="demo_keygen.asp" method="get">  
Username: <input type="text" name="usr_name" />  
Encryption: <keygen name="security" />  
<input type="submit" />  
</form>
```

Username: Encryption:

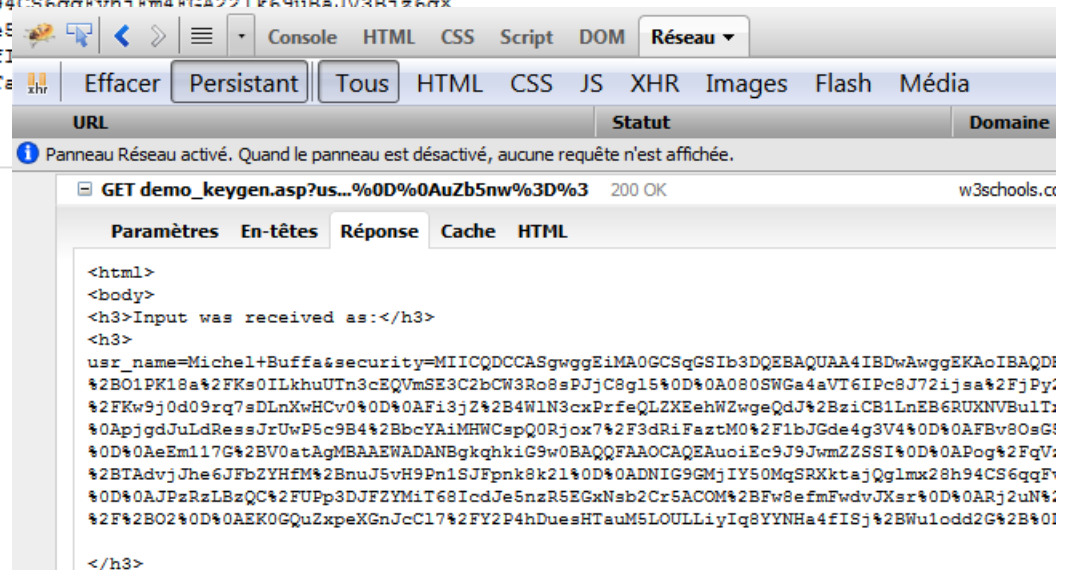
Haut niveau	▼
Haut niveau	
Niveau moyen	

Le navigateur génère une clé, il affiche un dialogue « génération de la clé, cette opération peut prendre plusieurs minutes », puis il soumet le formulaire avec la clé (il s'agit du champ usr_name crypté). [Démonstration](#)

<keygen> exemple (suite)



Requête envoyée et
réponse retournée
(traces dans
Firebug/Firefox)



<keygen> exemple (suite)

- Firefox

Username: Encryption:

- Opera

Username: Encryption:

- Chrome

Username: Encryption:

- Soucis d'ergonomie, entre autres...

Nouveaux attributs de <form> ou des champs de formulaires

- On a en a déjà vu plusieurs, notamment les attributs spécifiques à certains types (min, max, step, etc.)
- HTML5 a introduit de nouveaux attributs qui s'appliquent à tous les champs, et permet aussi de « sortir » des champs du formulaire lui-même tout en y étant rattachés
 - Permet plus de souplesse pour la création de l'interface utilisateur

Sortir des champs du formulaire, exemple

```
<input type="search" name="search_field"
                                form="search_form"/>
<form id="search_form" action="search.php"
      method="post">
  <fieldset>
    <legend>Options</legend>
    <input type="checkbox"/>Option 1
    <input type="checkbox"/>Option 2
  </fieldset>
</form>
```

Attributs formaction et formmethod

- Permettent d'avoir plusieurs actions possibles dans un formulaire. L'attribut "formaction" remplace dans ce cas l'attribut "action" du formulaire

```
<form action="post.php" method="post">  
  <input type="submit"  
    formaction="preview.php"  
    formmethod="get"  
    value="Prévisualiser">  
  <input type="submit" value="Envoyer">  
</form>
```

Attribut multiple

- Déjà vu avec type=email
- Peut-être utilisé aussi avec type=file

```
<form action="demo_form.asp">  
  Select images: <input type="file"  
                    name="img" multiple="multiple" />  
  <input type="submit" />  
</form>
```

- [Démonstration](#)

Attribut autofocus

- Permet de mettre le focus sur un champ particulier d'un formulaire, sitôt la page chargée entièrement dans le navigateur.

- Démonstration

```
<form action="demo_form.asp">
```

```
Nom:<input type="text" name="nom"
```

```
      autofocus="autofocus" /><br />
```

```
Prénom: <input type="text" name="pre"/><br />
```

```
<input type="submit"/>
```

```
</form>
```

Attributs height et width

- `<input type="image" src="img_submit.gif" alt="Submit" width="48" height="48"/>`
- Il s'agit d'un bouton qui est une image, et qui provoque la soumission du formulaire !
- Envoie deux paramètres supplémentaires : les coordonnées du point cliqué dans l'image

```
<form action="demo_form.asp">  
  First name: <input type="text" name="fname" /><br />  
  Last name: <input type="text" name="lname" /><br />  
  <input type="image" src="img_submit.gif" alt="Submit"  
    width="48" height="48"/>  
</form>
```

First name:
Last name:



Attributs width et height

- Lorsqu'on soumet, envoie :
- `fname=Michel&lname=Buffa&x=34&y=35`
- Démonstration

Pseudo classes CSS

```
<input type="text" value="optional"><br/>
<br/>
<input type="text" value="required" required>
<input type="time" value="06:06" required><br/>
<br/>
<input type="text" value="lettres9" pattern="[ a-zA-Z] *">
Effacez le chiffre, pour rendre ce champ valide.
```

■ Avec comme CSS :

[Démonstration](#)

```
<style>
  input:optional {background-color: silver}
  input:required {background-color: fuchsia}
  input:invalid {background-color: red}
</style>
```

optional

required

06:06

lettres9

Effacez le chiffre, pour rendre ce champ valide.

Attribut required

- Cet attribut indique qu'un champ input ne peut demeurer vide
- S'applique aux types text, search, url, tel, email, password, date pickers, number, checkbox, radio, et file
- `<input type="text" required>`
- Utile car peut être stylé via CSS

```
input:required {  
    background-color: green;  
}
```

Attribut required (1)

- Avec l'exemple précédent le champs input sera encadré de rouge par défaut, vert dès qu'un caractère sera saisi (cf css)

Attribut novalidate

- `<input type="submit"`
`formnovalidate="formnovalidate"`
`value="Soumettre sans valider" />`
- Permet de soumettre un formulaire même s'il contient des champs qui doivent être validés et qui sont incorrects
- Ex : un formulaire qui a un champ email ou un champ required et qui ne sont pas remplis
- En général, deux boutons de soumission
- [Démonstration](#)

Attribut pattern

- Permet de faire des validations complexes (à la volée ou à la soumission)
- Exemple :
- Code sur trois lettres : `<input type="text" name="country_code" pattern="[A-Za-z]{3}" title="Code 3 lettres" />`
- Démonstration
- Utilise la syntaxe des expressions régulières JavaScript

API de validité

- On peut aussi implémenter son propre gestionnaire de validité, grâce à l'API de validité
- La fonction JavaScript **setCustomValidity()** permet de personnaliser les messages d'erreur. Elle prend en paramètre une chaîne de caractère. Lorsque cette chaîne est vide, l'élément est considéré comme étant valide.
- Supporté par Firefox pour le moment...

API de validité exemple

```
<script>
function checkPasswords() {
var password1 = document.getElementById('password1');
var password2 = document.getElementById('password2');

    if (password1.value != password2.value) {
        password2.setCustomValidity('Les mots de passe ne
correspondent pas. ');
    } else {
        password2.setCustomValidity('');
    }
}
</script>
...

<body>
Mot de passe: <input type="password" id="password1"
oninput="checkPasswords()">
Répétez mot de passe: <input type="password" id="password2

oninput="checkPasswords()">

</body>
```

Démonstration
(Firefox)

Tags de structuration

Éléments de section, article, nav, aside, header, footer

- Ce sont des éléments de structure, plus précis que `` ou `<div>`
- Etablis à partir de statistiques d'utilisation des ids et classes les plus populaires du web,
 - `<div class="aside">` devient `<aside>` pour les menus sur le côté
- En devenant standards, on peut plus facilement leur appliquer une CSS standard, partageable,
- Les navigateurs peuvent les reconnaître et proposer un rendu spécifique,
- Il devient plus simple de générer une table des matières, par exemple

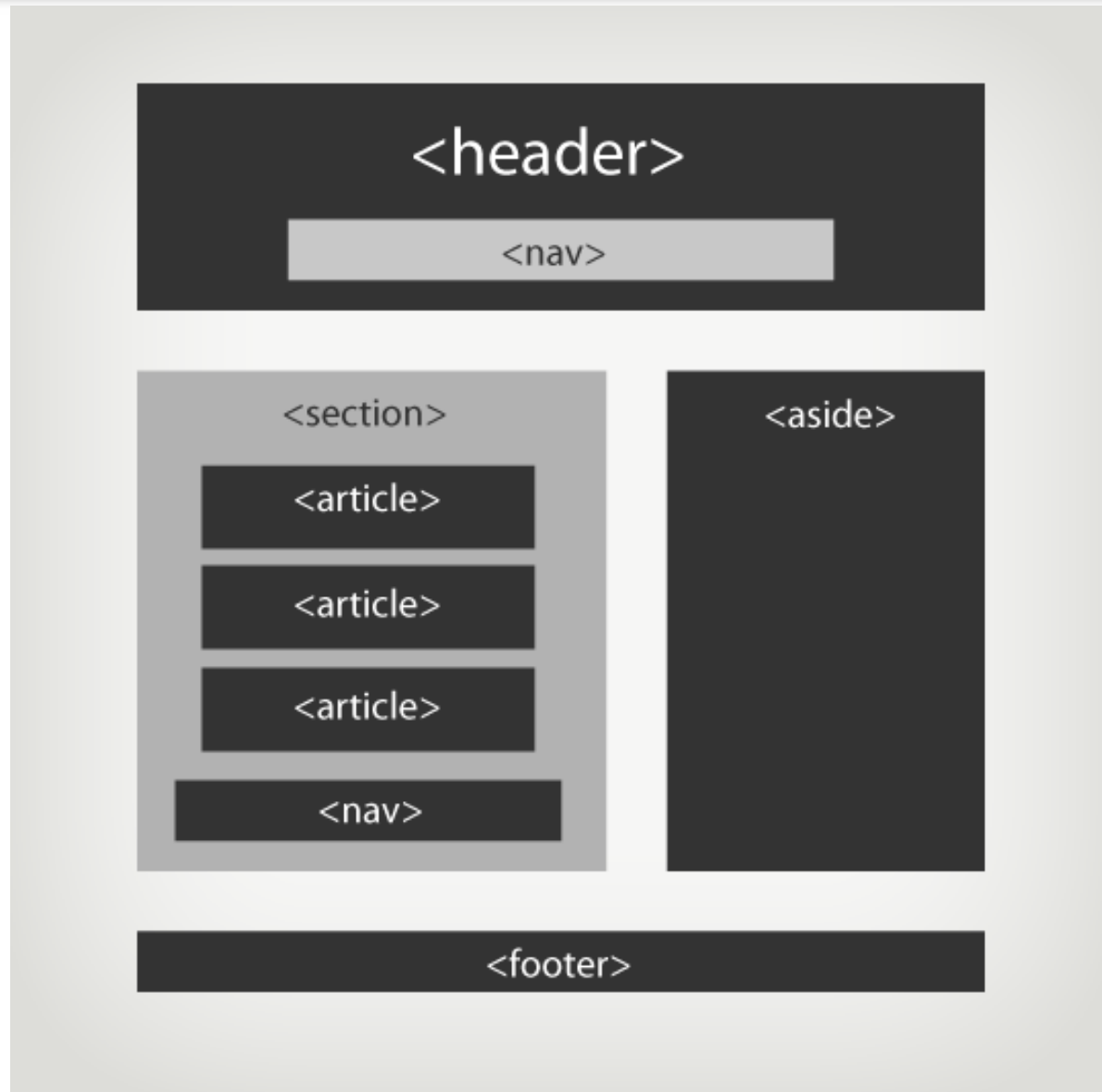
Synthèse (1)

■ Source :

<http://www.alsacreations.com/article/lire/1376-html5-section-article-nav-header-footer-aside.html>

Nom	Détails
<code><section></code>	Section générique regroupant un même sujet, une même fonctionnalité, de préférence avec un en-tête, ou bien section d'application web
<code><article></code>	Section de contenu indépendante, pouvant être extraite individuellement du document ou syndiquée (flux RSS ou équivalent), sans pénaliser sa compréhension
<code><nav></code>	Section possédant des liens de navigation principaux (au sein du document ou vers d'autres pages)
<code><aside></code>	Section dont le contenu est un complément par rapport à ce qui l'entoure, qui n'est pas forcément en lien direct avec le contenu mais qui peut apporter des informations supplémentaires.
<code><header></code>	Section d'introduction d'un article, d'une autre section ou du document entier (en-tête de page).
<code><footer></code>	Section de conclusion d'une section ou d'un article, voire du document entier (pied de page).

Synthèse (2)



Exemple de structure globale

```
<header>
<h1>Nouveaux éléments de section, article, header,
footer, aside, nav</h1>
</header>
<!-- nav principale -->
<nav>nav
  <ul>
    <li><a href="#">Rubrique 1</a></li>
    <li><a href="#">Rubrique 2</a></li>
    <li><a href="#">Rubrique 3</a></li>
    <li><a href="#">Rubrique 4</a></li>
  </ul>
</nav>

<!-- Main -->
<section id="main">
<article> .. </article>

</section>
```

Chaque article peut avoir sa propre structure

```
<article>
```

```
<header>
```

```
<h1>Titre de l'article</h1>
```

```
<p>Auteur : bidule</p>
```

```
</header>
```

```
<nav>
```

```
<ul>
```

```
<li><a href="index.html">Page d'accueil</a></li>
```

```
<li><a href="contact.html">Contact</a></li>
```

```
</ul>
```

```
</nav>
```

```
<p>Contenu de l'article</p>
```

```
...
```

```
<footer>
```

```
<p>Posté par Simon, le <time datetime="2012-02-02">2 février 2012</time> </p>
```

```
</footer>
```

```
</article>
```


Doit-on avoir honte d'utiliser des divs ?

- Ah, grande discussion...
- Les tags de structuration ont été pensés pour des sites webs généralistes (header, footer, nav)
- Pour des blogs, forums, cms (section, article, etc.)
- Pour le reste, vous êtes libres !
- Voir cette [discussion très intéressante](#)

<detail> et <summary>

- Similaire aux accordéons proposés par les libraries JavaScript

- ▶ `Comment tuer le boss ! Attention Spoiler !`

- On clique sur la flèche et « ça s'ouvre » :

- ▼ `Comment tuer le boss ! Attention Spoiler !`

- `Prenez l'épée Excalibur, levez votre bouclier`

- `Et balancez des boules de feu !`

- Nombreux styles applicables pour l'icône

- [Démonstration](#)

<detail> et <summary> suite...

```
<details>
```

```
<summary>Comment tuer le boss ! Attention Spoiler  
!</summary>
```

```
<p> - Prenez l'épée Excalibur, levez votre  
bouclier</p>
```

```
<p>Et balancez des boules de feu !</p>
```

```
</details>
```

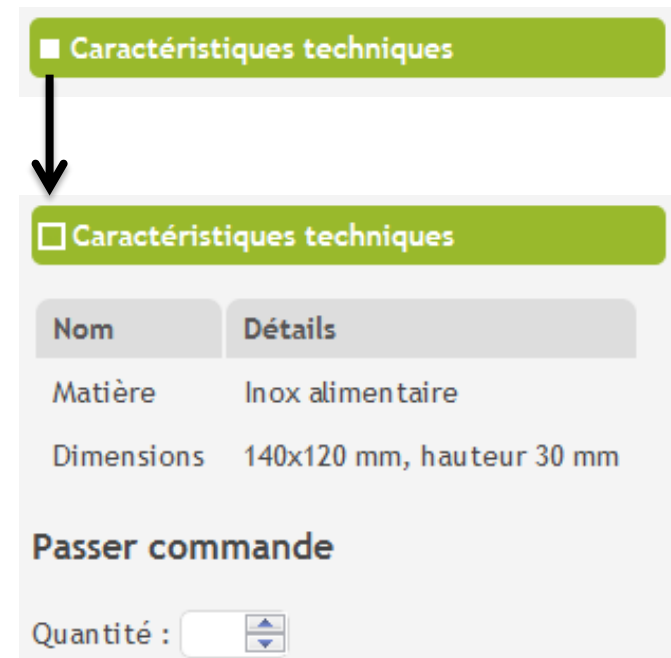
- Support pour le moment uniquement sur Browsers basés sur WebKit (Chrome, Safari, Android)
- On peut imbriquer des blocs <details> les uns dans les autres

<detail> et <summary>, CSS associées

- Il existe des pseudo classes CSS pour styler l'icône selon qu'on est dans l'état ouvert ou fermé

```
summary::-webkit-details-marker {  
    color:#ADCA48;  
    background:#ADCA48;  
}  
  
details[open]  
    summary::-webkit-details-marker {  
        color:#eaeaea;  
        background:#eaeaea;  
        outline: 2px solid #ADCA48;  
    }
```

- Démonstration



Changer l'icône par défaut

```
summary::-webkit-details-marker {  
  display: none  
}  
summary:after {  
  content: "+";  
  color: #ADCA48;  
  float: left;  
  font-size: 1.5em;  
  font-weight: bold;  
  margin: -5px 5px 0 0;  
  padding: 0;  
  text-align: center;  
  width: 20px;  
}  
details[open]  
  summary:after {  
    content: "-";  
  }
```



■ Démonstration

Le tag <mark>

- C'est un « highlighter », pour mettre du texte au stabilo !

<p>Le projet est à rendre<mark>au format .zip</mark> lundi prochain.</p>

- Donne :

Le projet est à rendreau format .zip lundi prochain.

- Démonstration

Le tag `<mark>` suite...

- Peut être utilisé à l'intérieur d'autres tags, par exemple dans un `<pre>`, un `<code>` un `` etc

```
<pre><code>
```

```
<mark>var</mark> i = 3;
```

```
</code></pre>
```

```
<p>Le mot-clé var permet de déclarer  
une variable en JavaScript.</p>
```

```
var i = 3;
```

Le mot-clé var permet de déclarer une variable en JavaScript.

Le tag <time>

- Sert à indiquer une date dans un format lisible par les machines
 - Ex : utilisable par extensions Chrome ou Firefoiw pour des alertes aux anniversaires, pour insertion automatique dans Google Calendar, etc
- Non supporté pour le moment... support partiel dans Opera

`<p>Nous ouvrons à <time>10:00</time> chaque matin.</p>`

`<p>J'ai rendez-vous le <time datetime="2012-02-14">lundi 14 février 2012</time>.</p>`

Le tag `<time>`, syntaxe très complète

- `<time datetime="1905">` L'année 1905
- `<time datetime="1905-11">` Novembre 1905
- `<time datetime="11-13">` Le 13 November
- `<time datetime="1905-W21">` semaine 21 de l'année 1905
- `<time datetime="1905-11-13T09:00">` : avec l'heure
- `<time datetime="1905-11-13 09:00">` : idem
- `<datetime="09:00Z">` 9h du matin, GMT.
- `<datetime="09:00-05">` 9h du matin, GMT – 5 heures.
- `<datetime="09:00+05:45">` 9h au [Nepal](#), qui a 5h45mn de décalage par rapport à GMT
- Nombreuses autres options pour exprimer des durées, des événements périodiques, voir <http://www.brucelawson.co.uk/2012/best-of-time/>

HTML5, l'API de selection dans le DOM

L'API JavaScript « Selectors »

- Cette API sert à « matcher » des éléments du DOM par leur Id, Tag, classe CSS etc...
- Existant : par Id, par Tag

```
var el = document.getElementById('header') ;  
el.focus() ;  
var els = document.getElementsByTagName('div') ;  
els[0].focus() ;
```

- Nouveauté HTML5 : par classe CSS

```
var els = document.getElementsByClassName('section') ;  
els[0].focus() ;
```

Requêtes complexes : `querySelector()` !

- Dès que l'on veut faire des requêtes complexes comme « le premier input situé dans un div », etc. on faisait appel à des frameworks JavaScript comme jQuery, Dojo, Mootools, etc.
- Nouveautés HTML5 :
 - `document.querySelector(req)` : retourne le premier élément qui satisfait la requête,
 - `document.querySelectorAll(req)` : retourne tous les éléments qui satisfont la requête
- Syntaxe des requêtes très inspirée de la syntaxe jQuery

querySelector() exemples...

■ HTML5 :

```
var el = document.querySelector('#nav ul li');  
var els = document.querySelectorAll('ul li:nth-child(even)');  
var els = document.querySelectorAll('form.test > tr > td');
```

■ Equivalents jQuery :

```
var el = $('#nav ul li').item(0);  
var els = $('ul li:nth-child(even)');  
var els = $('form.test > tr > td');
```

■ Spécification des selecteurs HTML5

- <http://www.w3.org/TR/selectors-api/>

Selecteurs : des exemples

- `querySelectorAll("p.warning, p.error");`
 - Tous les éléments qui ont pour classe `warning` ou `error`
- `querySelector("#foo, #bar");`
 - Le premier élément d'id égal à `foo` ou `bar`,
 - Renvoie `null` sinon
- `var div = document.getElementById("bar");`
`var p = div.querySelector("p");`
 - Le premier `p` d'un `div`
- `var elm = document.querySelector("ul li.red");`
 - Le premier élément `` de classe « `red` » d'une liste ``

Selecteurs : des exemples

```
<ul class="nav">  
  <li><a href="/">Home</a></li>  
  <li><a href="/products">Products</a></li>  
  <li><a href="/about">About</a></li>  
</ul>
```

- Voici un itérateur sur les des de la classe « nav »

```
var lis = document.querySelectorAll("ul.nav > li");  
  
for (var i = 0; i < lis.length; i++) {  
  process(lis[i]);  
}
```

Selecteurs : des exemples

```
<ul id="fruits">
  <li><input type="checkbox" name="fruit" value="apples"> Apples</li>
  <li><input type="checkbox" name="fruit" value="oranges"> Oranges</li>
  <li><input type="checkbox" name="fruit" value="bananas"> Bananas</li>
  <li><input type="checkbox" name="fruit" value="grapes"> Grapes</li>
</ul>
```

- On veut tous les inputs qui ont été cochés :

```
var list = document.querySelectorAll("#fruits input:checked");
```

- Autre exemple, on veut changer le background de tous les paragraphes sous le div

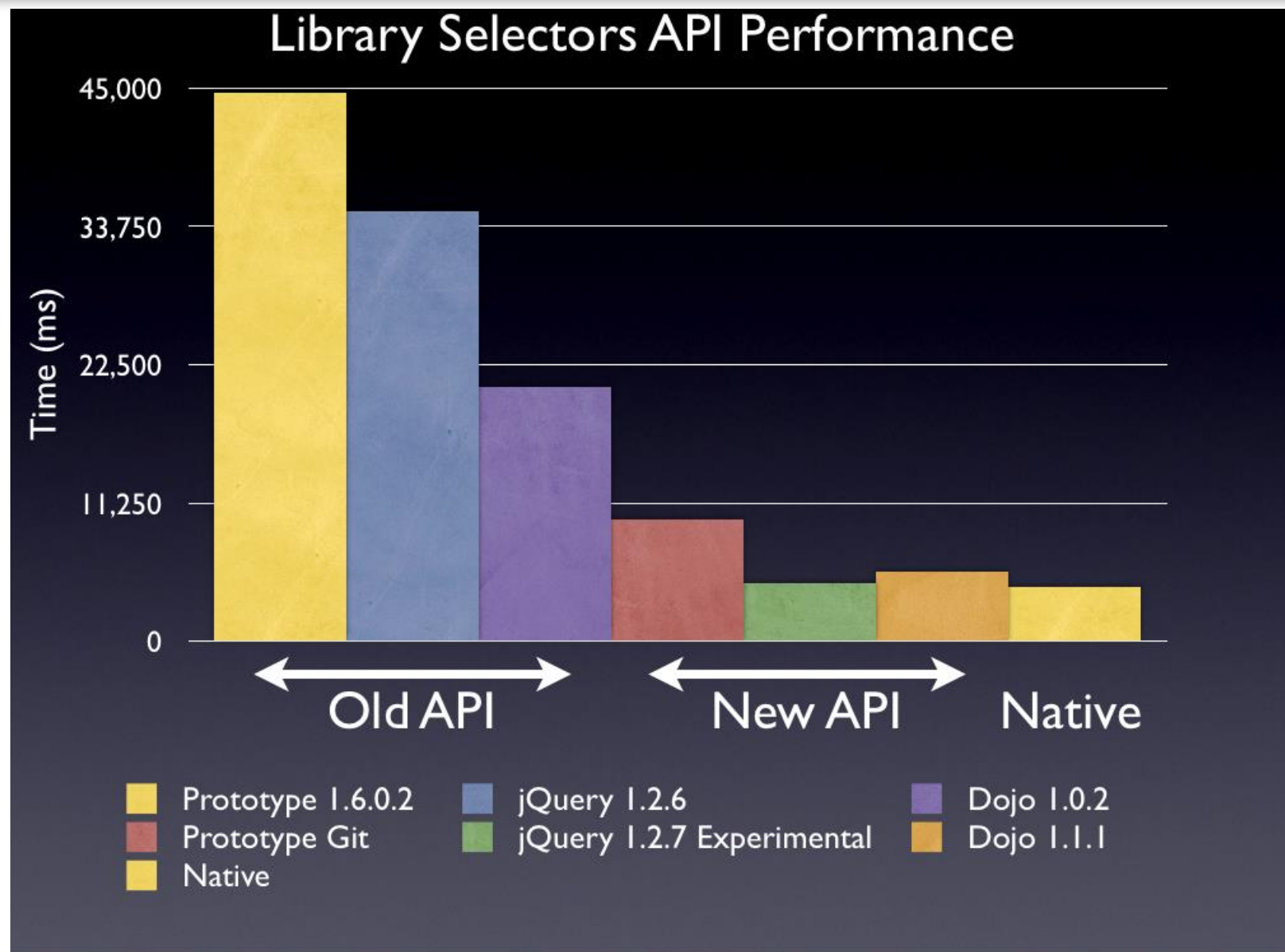
```
<div id="id" class="class">
  <p>First paragraph.</p>
  <p>Second paragraph.</p>
</div>
```

```
var p = document.querySelectorAll("#id p");
for ( var i = 0; i < p.length; i++ ) {
  p[i].style.backgroundColor = "red";
}
```


Doit-on cesser d'utiliser jQuery ou autre...

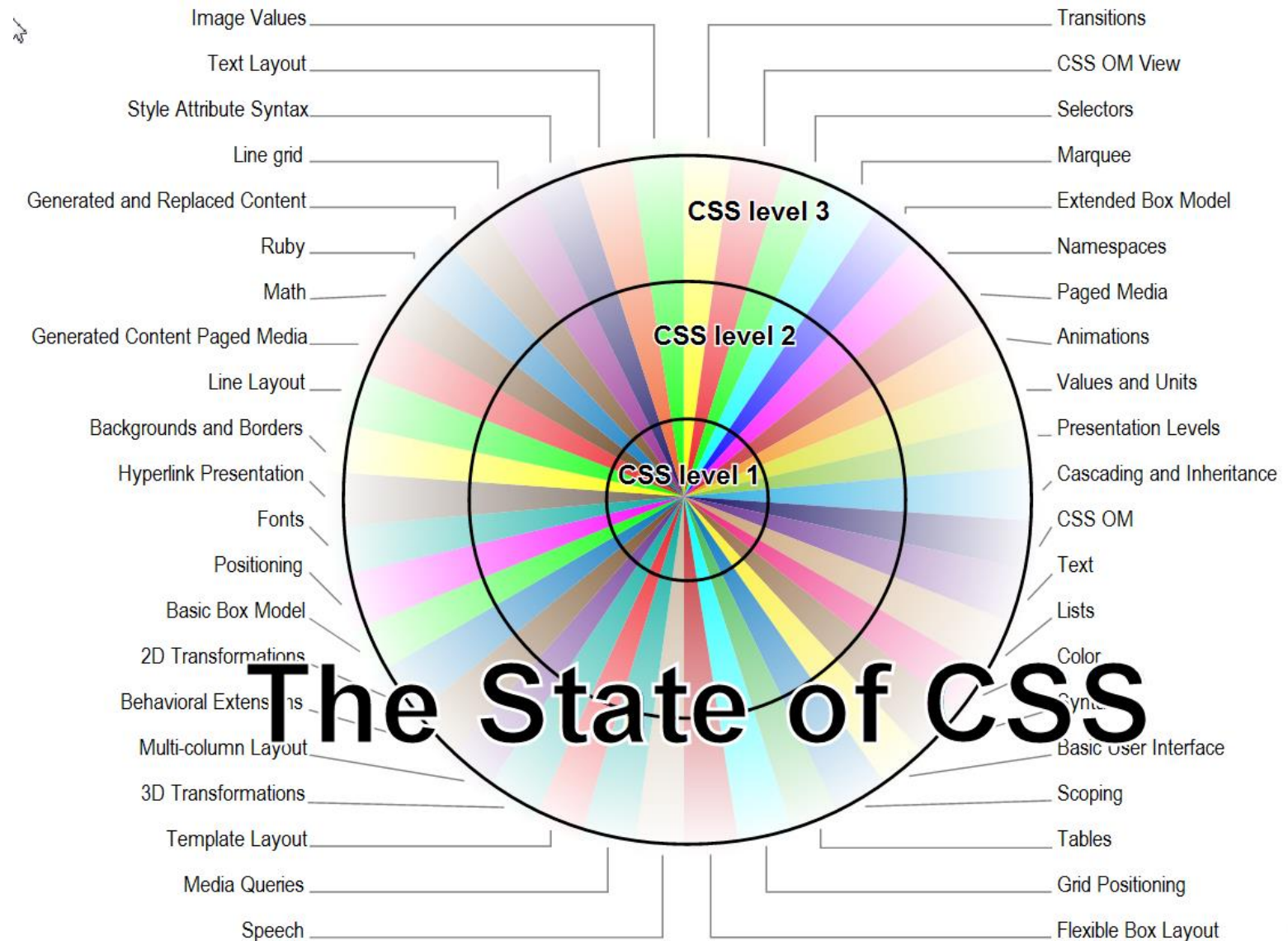
- Réponse : oui et non !
 - Déjà, au niveau syntaxe, le dernier exemple aurait été plus court avec jQuery !
`$ (' #id p ') .css (' background ' , ' #FF0000 ') ;`
- Oui : car l'implémentation native des sélecteurs DOM est beaucoup plus rapide que celle proposée par ces librairies auparavant
 - Voir le test interactif de performance [Slick Speed](#)
- Non : car les dernières versions utilisent l'implémentation native si elle est disponible.

Frameworks JavaScript : améliorations avec support natif HTML5



CSS3, nouveautés

Etat des lieux...

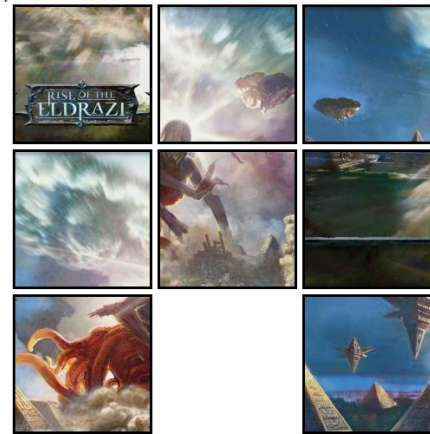


CSS3 transitions

- CSS3 introduit la notion de « transitions », c'est-à-dire passer d'un état à un autre
- Ressources :
<http://blogs.msdn.com/b/eternalcoding/archive/2011/11/01/css3-transitions.aspx>
- Spécification : <http://www.w3.org/TR/css3-transitions/>

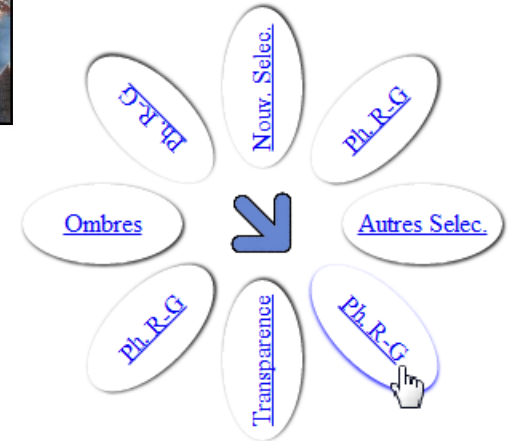
CSS3 transitions

- Ce sont des « animations simples » utiles par exemple pour donner un feedback visuel lorsqu'on clique ou lorsqu'on passe la souris sur un élément.
- Les nombreux « slides html5 » que l'on trouve sur le net (comme slides.html5rocks.com) utilisent des transitions CSS3 pour passer d'un transparent à l'autre.



Transitions Puzzle!

Mix it!
CSS3 Transitions activated
28.64s



CSS3 transitions : principe

- On définit une transition dans un fichier css ou dans un tag style
- La transition indique :
 1. Les propriétés sur lesquelles on va calculer par interpolation une transition,
 2. La durée de la transition,
 3. Le type d'interpolation (linéaire, autre)
- La transition démarre dès qu'une des propriétés indiquée change.
 - Ex : on change la couleur d'un objet bleu pour la mettre à rouge, et bien on va animer la couleur du bleu vers le rouge si il existe une transition

CSS3 transitions exemples

```
#id {  
    transition-property: all;  
    transition-duration: 0.5s;  
    transition-timing-function: ease;  
    transition-delay: 0s;  
}
```

- Indique que l'on veut animer toutes les propriétés supportées,
- Durée = 0,5s,
- Type de la transition = accélère-constant-ralentit (ease)
- La transition démarre sitôt qu'une des propriétés a changé (0s)

CSS3 exemple (1)

```
<head>
<style type="text/css">
div {
  width:100px;
  height:100px;
  background:red;
  transition:width 2s, height 2s;
  -moz-transition:width 2s, height 2s, -moz-transform 2s;
  -webkit-transition:width 2s, height 2s, -webkit-transform 2s;
  -o-transition:width 2s, height 2s, -o-transform 2s;
}

div:hover {
  width:200px;
  height:200px;
  transform:rotate(180deg);
  -moz-transform:rotate(180deg); /* Firefox 4 */
  -webkit-transform:rotate(180deg); /* Safari and Chrome */
  -o-transform:rotate(180deg); /* Opera */
}
</style>
</head>
<body>
<div>Hover over me to see the transition effect!</div>
</body>
</html>
```

Démonstration

Hover over me
to see the
transition effect!



CSS3 transitions remarques

- Beaucoup de duplications, car tant que la spécification n'est pas finalisée, on a affaire à des implémentations expérimentales.
 - Pour être sûr que ça marche : dupliquer
 - Ou bien utiliser des outils [CSS Scaffold](#), [LESS](#), mieux [SASS](#), qui évitent la duplications
 - Encore mieux : utiliser [Lea Verou's -prefix-free](#) qui ajoute les duplications à votre place au run-time.

CSS3 transitions syntaxes possibles

- Spécifier pour toutes les propriétés

```
transition-property: all;  
transition-duration: 0.5s  
transition-timing-function: ease;  
transition-delay: 0s;
```

- Ou la même chose sur une ligne

```
transition: all 0.5s ease 0s;
```

- Par propriétés séparées

```
transition-property: opacity left top;  
transition-duration: 0.5s 0.8s 0.1s;  
transition-timing-function: ease linear ease;  
transition-delay: 0s 0s 1s;
```

- Aussi sur une ligne (en séparant par des virgules)

```
transition: opacity 0.5s ease 0s, left 0.8s linear 0s;
```

CSS3 transitions, types d'interpolations

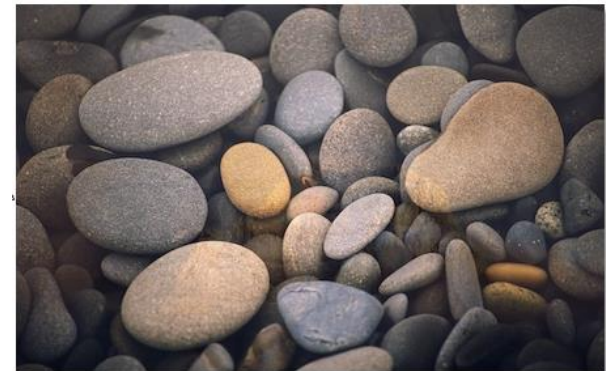
- linear, ease, ease-in, ease-out, ease-inout, custom
- Chacune de ces valeurs va donner une animation différente car les valeurs de propriétés seront interpolées différemment.
- Tester des démos pour mieux comprendre :
 - [Démonstration 1](#)
 - [Démonstration 2](#)
 - Générateur de fonctions custom (à partir de courbes de Bezier que l'on peut manipuler interactivement) : <http://matthewlein.com/ceaser/>

CSS3 transitions : amusons-nous !

■ Animations d'images en crossfading

```
<div id="cf">  
    
    
</div>
```

```
#cf {  
  position:relative;  
  height:281px;  
  width:450px;  
  margin:0 auto;  
}  
#cf img {  
  position:absolute;  
  left:0;  
  -webkit-transition: opacity 1s ease-in-out;  
  -moz-transition: opacity 1s ease-in-out;  
  -o-transition: opacity 1s ease-in-out;  
  -ms-transition: opacity 1s ease-in-out;  
  transition: opacity 1s ease-in-out;  
}  
#cf img.top:hover {  
  opacity:0;  
}
```



[Démonstration](#)

CSS3 transitions, **types** des propriétés qui peuvent être animées

- **color**: interpolation des valeurs R,G,B,A
- **length**: longueur, interpolées en entiers
- **percentage**: interpolés en réels
- **integer**: interpolées en entiers
- **number**: interpolés en réels
- **transform list**: see CSS Transforms
specification: <http://www.w3.org/TR/css3-2d-transforms/>
- **rectangle**: x,y,width, height
- **visibility**: en réels
- **Shadow**: color, x, y et blur
- **gradient**: interpolés uniquement si ils sont du même types et avec le même nombre de pas (steps)
- **paint server** (SVG): gradient vers gradient et color vers color

CSS3 transitions, noms des propriétés qui peuvent être animées

- background-color (*color*)
- background-image (*only gradients*)
- background-position (*percentage and length*)
- border-bottom-color (*color*)
- border-bottom-width (*length*)
- border-color (*color*)
- border-left-color (*color*)
- border-left-width (*length*)
- border-right-color (*color*)
- border-right-width (*length*)
- border-spacing (*length*)
- border-top-color (*color*)
- border-top-width (*length*)
- border-width (*length*)
- bottom (*length and percentage*)
- color (*color*)
- crop (*rectangle*)
- font-size (*length and percentage*)
- font-weight (*number*)
- grid-* (*various*)
- height (*length and percentage*)
- left (*length and percentage*)
- letter-spacing (*length*)
- line-height (*number, length and percentage*)
- margin-bottom (*length*)
- margin-left (*length*)
- margin-right (*length*)
- margin-top (*length*)
- max-height (*length and percentage*)
- max-width (*length and percentage*)
- min-height (*length and percentage*)
- min-width (*length and percentage*)
- opacity (*number*)
- outline-color (*color*)
- outline-offset (*integer*)
- outline-width (*length*)
- padding-bottom (*length*)
- padding-left (*length*)
- padding-right (*length*)
- padding-top (*length*)
- padding-top (*length*)
- right (*length and percentage*)
- text-indent (*length and percentage*)
- text-shadow (*shadow*)
- top (*length and percentage*)
- vertical-align (*keywords, length and percentage*)
- visibility (*visibility*)
- width (*length and percentage*)
- word-spacing (*length and percentage*)
- z-index (*integer*)
- zoom (*number*)

CSS3 transitions, pour aller plus loin

- http://css3.bradshawenterprises.com/transition_s/ : en bas de la page, menu avec de nombreux use-cases...
- <http://debray-jerome.developpez.com/> : articles en français sur CSS3

CSS3 animations

- Principe = évolution des transitions. Les mêmes propriétés sont animables.
- Exemples : [Démonstration 1](#) et [Démonstration 2](#)
- Différence entre une animation et une transition :
 - Une animation ce n'est plus entre deux états, mais entre n états,
 - On va définir un ensemble de KeyFrames (états), associés à un pourcentage (0% = début, 100% = fin, 50 % = milieu de l'animation, etc).
 - Pour chaque KeyFrame on donnera des valeurs de propriétés,
 - On donnera un nom à l'animation
 - On associera l'animation à des objets de la page, et on indiquera la durée de l'animation, si elle est en boucle etc.

CSS3 animation exemple simple avec deux frames

```
<style type="text/css">
h1 {
  -moz-animation-duration: 3s;
  -moz-animation-name: slidein;
}

@-moz-keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%
  }
  to {
    margin-left: 0%;
    width: 100%;
  }
}
</style>
```

- Cet exemple anime tous les titres en H1 en les faisant apparaître de droite à gauche, durée de l'animation 3s
- Note : dans cet exemple on a utilisé des préfixes –moz- mais qui devraient disparaître sitôt la spécification finalisée

Démonstration

CSS3 animation exemple simple avec trois frames

```
<style type="text/css">
h1 {
  -moz-animation-duration: 3s;
  -moz-animation-name: slidein;
}

@-moz-keyframes slidein {
  from {
    margin-left: 100%;
    width: 300%
  }
  75% {
    font-size: 300%;
    margin-left: 25%;
    width: 150%;
  }
  to {
    margin-left: 0%;
    width: 100%;
  }
}
</style>
```

- On a ajouté une position (frame) intermédiaire à 75% du temps de l'animation,
- la police de caractère est de taille 3 fois plus grande à cet endroit là : l'animation va progressivement grossir la taille entre la position initiale et celle-ci,
- Puis la diminuer jusqu'à la position finale
- **On parle d'interpolation !**

Démonstration

CSS3 animation exemple simple, animation infinie, alternée, etc...

```
<style type="text/css">
h1 {
  -moz-animation-duration: 3s;
  -moz-animation-name: slidein;
  -moz-animation-iteration-count: infinite;
  -moz-animation-direction: alternate;
}
```

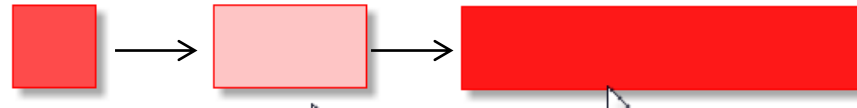
- Quand on associe l'animation à un objet du DOM (ici les tags H1), on peut dire si on veut une animation infinie (en boucle)
- Ou alternée (va et vient)

Démonstration

CSS3 animation autre exemple

```
<h2>Demo</h2>
<style>
@-webkit-keyframes resize {
  0% {
    padding: 0;
  }
  50% {
    padding: 0 20px;
    background-color:rgba(255,0,0,0.2);
  }
  100% {
    padding: 0 100px;
    background-color:rgba(255,0,0,0.9);
  }
}
#box {
  height:50px;
  width:50px;
  margin:0 auto;
  border:1px red solid;
  background-color:rgba(255,0,0,0.7);
}
#box:hover {
  -webkit-animation-name: resize;
  -webkit-animation-duration: 1s;
  -webkit-animation-iteration-count: 4;
  -webkit-animation-direction: alternate;
}
</style>
<div id="box"></div>
```

Note : dans cet exemple on a utilisé des préfixes – webkit- mais qui devraient disparaître sitôt la spécification finalisée



Vous pouvez copier-coller cet exemple et le tester, par exemple sur <http://css3.bradshawenterprises.com/animations/>

CSS3 animation, syntaxe

```
@keyframes nom_animation {  
  from {  
    propriété à animer 1 : valeur initiale;  
    propriété à animer 2 : valeur initiale;  
    ...  
  }  
  50% {  
    propriété à animer 1 : valeur intermédiaire;  
    propriété à animer 2 : valeur intermédiaire;  
    ...  
  }  
  to {  
    propriété à animer 1 : valeur finale;  
    propriété à animer 2 : valeur finale;  
    ...  
  }  
}
```

- NOTE / AU LIEU DE 0% et 100% on peut mettre `from` et `to`

CSS3 animation, syntaxe (1)

- On peut définir autant d'étapes que l'on veut, mais from/0% et to/100% sont obligatoires
- Si l'animation dure 20s, alors from/0% est l'état à t=0s, to/100% sera l'état à t=20s, 50% sera l'état à t=10s, etc...
- Une fois l'animation (@keyframes) définie il suffit de l'associer à un objet

```
#id_of_the_html_element {  
  animation-name: nom_animation;  
  animation-duration: nombre de secondes s;  
  animation-iteration-count: nombre| infinite;  
}
```

CSS3 animations : types d'interpolations

- Par défaut, l'interpolation d'une étape de l'animation à l'autre est linéaire, mais on peut utiliser les types déjà vus pour les transitions (ease, ease-in, etc...)

```
@keyframes demo {  
  from {  
    animation-timing-function: ease;  
    transform: translateX(0px);  
  }  
  50% {  
    animation-timing-function: ease-in;  
    transform: translateX(300px);  
  }  
  to {  
    animation-timing-function: ease-inout;  
    transform: translateX(900px);  
  }  
}  
#testCanvas { animation-delay: 0s; animation-duration: 6s;  
  animation-iteration-count: infinite;  
  animation-name: demo;  
}
```

Démonstration

CSS3 animations : événements

- Trois événements peuvent être déclenchés :
“*AnimationStart*”, “*AnimationEnd*” et
“*AnimationIteration*”
 - Note : pour que ça fonctionne aujourd’hui, utiliser aussi les versions préfixées : *ChromeAnimationEnd*, *MozAnimationEnd*, *MSAnimationEnd*, etc.
- Exemple pour chaîner deux animations :

```
elementToAnimate.addEventListener("AnimationEnd",  
    function () {  
        alert("the end !");  
    },  
    false);
```

CSS3 animation et événements

- Un exemple intéressant ici :
<https://developer.mozilla.org/samples/cssref/animations/animevents.html>

L'API de géolocalisation


API de géolocalisation

- Utilise plusieurs moyens : GPS, triangulation GSM/3G, Wifi, adresse IP
- C'est le navigateur qui implémente le code de géolocalisation, on peut parfois configurer des préférences sur l'ordre de priorité des moyens à utiliser
 - Le plus souvent si GPS activé alors GPS, sinon Wifi, sinon IP, sinon GSM/3G... ordre décroissant de précision
- Supporté par tous les navigateurs récents
- S'utilise via JavaScript

API de géolocalisation (2)

■ Appel asynchrone de `getCurrentPosition(callback)`

```
<!DOCTYPE html>
<html>
<body>
<p id="demo">Cliquez sur le bouton pour obtenir votre position</p>
<button onclick="getLocation()">Géolocalisez moi</button>
<script>
  var x=document.getElementById("demo");
  function getLocation(){
    if (navigator.geolocation) {
      navigator.geolocation.getCurrentPosition(showPosition);
    } else{
      x.innerHTML="Géolocalisation non supportée par votre
        navigateur.";
    }
  }
  function showPosition(position) {
    x.innerHTML="Latitude: " + position.coords.latitude +
      "<br />Longitude: " + position.coords.longitude;
  }
</script>
</body>
</html>
```



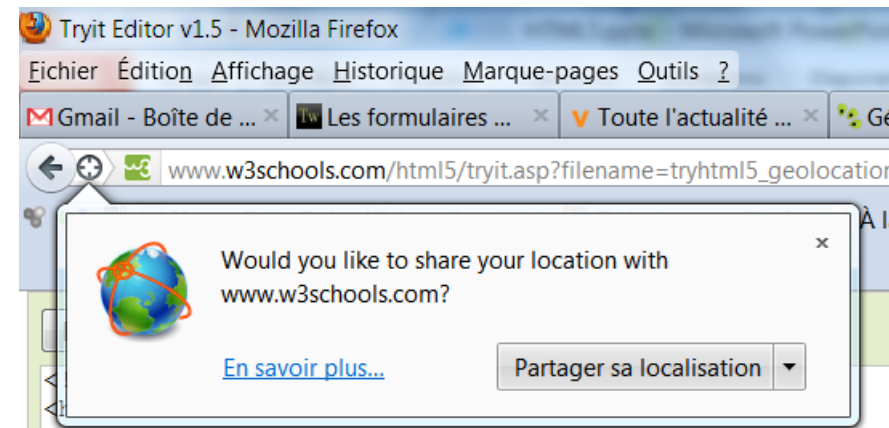
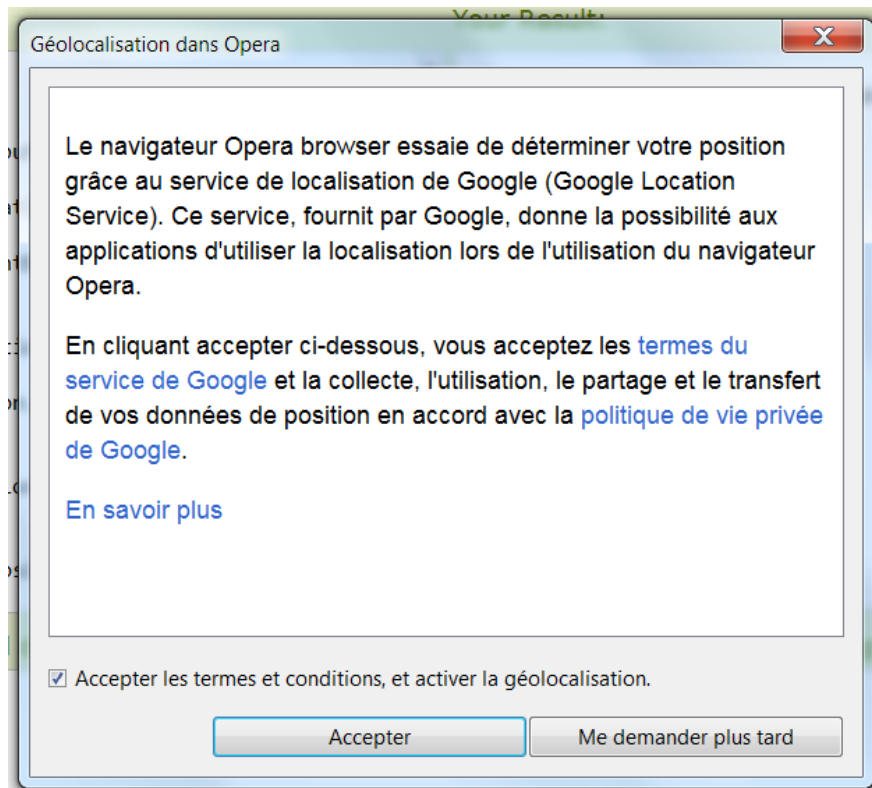
Démonstration

API de géolocalisation : l'objet position

- **Position.coords** permet d'accéder à
 - **latitude**, la latitude de la position ;
 - **longitude**, la longitude de la position ;
 - **altitude**, l'altitude de la position ;
 - **accuracy**, niveau de précision de la longitude et de la latitude (en mètres) ;
 - **altitudeAccuracy**: niveau de précision de l'altitude (en mètres) ;
 - **heading**, donne la position en degrés par rapport au nord ;
 - **speed**, affiche la vitesse actuelle de déplacement de la position (en mètres).
- Toutes ces valeurs ne sont pas encore toutes présentes dans les navigateurs
- Les plus répandues sont : latitude, longitude et altitude (unique à Firefox pour l'instant). Si une valeur n'est pas existante, elle retournera null.

API de géolocalisation (3)

- Le navigateur demande la première fois si on est d'accord pour partager sa position



API de géolocalisation : cas d'erreurs

```
<div id="maposition"></div>

...

// Fonction de callback en cas d'erreur
function erreurPosition(error) {
  var info = "Erreur lors de la géolocalisation : ";
  switch(error.code) {
    case error.TIMEOUT:
      info += "Timeout !";
      break;
    case error.PERMISSION_DENIED:
      info += "Vous n'avez pas donné la permission";
      break;
    case error.POSITION_UNAVAILABLE:
      info += "La position n'a pu être déterminée";
      break;
    case error.UNKNOWN_ERROR:
      info += "Erreur inconnue";
      break;
  }
  document.getElementById("maposition").innerHTML = info;
}
navigator.geolocation.getCurrentPosition(affichePosition, erreurPosition);
```


API de géolocalisation : google map

- On peut très facilement afficher une carte Google Map à partir de la position obtenue
- [Démonstration](#) et [Démonstration2](#)
- [Nombreux exemples d'utilisation de la Google Map API + l'API de Géolocalisation](#)
 - Ici aussi, [exemple interactif](#)

API de géolocalisation : watchPosition

- Pour un suivi en temps réel type application de navigation, il suffit de remplacer `getCurrentPosition` par `watchPosition`
- La fonction de callback sera appelée à intervalles réguliers

```
var idPos =  
    navigator.geolocation.watchPosition(surveillePosition);
```

- On arrêtera le suivi en appelant `clearWatch()` sur l'id obtenu :

```
idPos.clearWatch()
```

API de géolocalisation : options

- On peut passer aux fonctions `getCurrentPosition` et `watchPosition` un objet javascript qui contiendra une ou plusieurs options parmi :

- `enableHighAccuracy` : force l'utilisation du GPS (propose de l'activer, sur mobile)

```
navigator.geolocation.getCurrentPosition(maPosition, erreurPosition, {enableHighAccuracy:true});
```

- `Timeout` : au bout de combien de temps on appelle la fonction d'erreur

- `maximumAge` : pendant combien de temps la position demeure dans le cache (ms)

```
navigator.geolocation.getCurrentPosition(maPosition, erreurPosition, {maximumAge:600000, timeout:0});
```

Support accéléromètre ?

- L'API de géolocalisation permet d'avoir la position et l'orientation 2D (boussole), mais pas l'orientation 3D du device
- On l'obtient à l'aide d'un nouvel événement HTML5 sur l'objet window, depuis du JavaScript

```
window.addEventListener('deviceorientation', function(event) {  
    var a = event.alpha;  
    var b = event.beta;  
    var g = event.gamma;  
}, false);
```

- Démonstration

Le multimédia (vidéo, audio)

Jusqu'à aujourd'hui

■ Une vidéo c'est ça :

```
<object width="425" height="344">
  <param name="movie"
    value="http://www.youtube.com/v/9sEI1AUFJKw&hl=en_GB&fs=1">
  </param>
  <param name="allowFullScreen" value="true"></param>
  <param name="allowscriptaccess" value="always"></param>

  <embed src="http://www.youtube.com/v/9sEI1AUFJKw&hl=en_GB&fs=1"
    type="application/x-shockwave-flash"
    allowscriptaccess="always" allowfullscreen="true"
    width="425" height="344">
  </embed>
</object>
```

Maintenant : <video>

- Principe de base :

```
<video width="320" height="240"
controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  Votre navigateur ne supporte pas le tag video.
</video>
```

- L'attribut « controls » ajoute les boutons de contrôle (play, stop, etc.)
- Le premier format supporté sera lu !
- [Démonstration](#)



<video> : ressources web

- Une excellente référence est la documentation d'Opera, en bas de la page on trouve toutes les références importantes et de nombreux pointeurs vers des exemples ciblés (sous-titres, etc)
 - <http://dev.opera.com/articles/view/introduction-html5-video-fr/>

Le tag <video>

- Grande bataille d'influence autour de ce tag !
 - Flash vs HTML5
 - Codecs, brevets, contrôle, etc WebM (Google) vs H.264 (Apple, Microsoft) vs Ogg (Firefox) etc.

Browser	MP4	WebM	Ogg
Internet Explorer 9	YES	NO	NO
Firefox 4.0	NO	YES	YES
Google Chrome 6	YES	YES	YES
Apple Safari 5	YES	NO	NO
Opera 10.6	NO	YES	YES

- Une des solution consiste sur un serveur Linux à transcoder la vidéo à la volée dans un des format supporté par le navigateur client. [Le framework Gstreamer fait cela très bien](#)

<video> : contrôle depuis le DOM

- <video> possède des méthodes, des propriétés et des événements que l'on peut manipuler depuis JavaScript et l'API du DOM.
- Méthodes pour jouer, mettre en pause etc.
- Propriétés (durée, position courante, etc.) que l'on peut consulter/modifier
- Des événements sont générés durant le cycle de vie et on peut définir des écouteurs. On peut aussi envoyer des événements pour contrôler le lecteur vidéo.

<video> est un élément du DOM

- On peut le manipuler ou le créer depuis du code, via l'API de manipulation du DOM

```
var video = document.createElement('video');  
video.src = 'video.ogv';  
video.controls = true;  
document.body.appendChild(video);
```

<video> exemple

```
<video style='opacity: 0.5;
        position: absolute;
        top: 55%;
        left: 40%;
        width: 320px; height:240px;
        z-index: 1'
        src='/2008/Talks/26-video-plh/small.ogg'
        width='360' height='240'
        id="vid">
```

<p>This video cannot be displayed.

No support for HTML5?</p>

```
</video>
```

```
<span id='play' onclick="playVideo();">play</span>
```

```
<span id='pause' onclick="pauseVideo;">pause</span>
```

```
<span id='rewind' onclick="rewindVideo();">Retour à la position  
de départ</span>
```

```
<script>
```

```
    vid = document.getElementById("vid");
```

```
    play = document.getElementById("play");
```

```
    pause = document.getElementById("pause");
```

```
    rewind = document.getElementById("rewind");
```

```
</script>
```

<video> exemple

```
function playVideo() {  
    vid.play();  
    play.style.color='black';  
    pause.style.color='#999';  
    rewind.style.color='#999';  
}  
  
function pauseVideo() {  
    vid.pause();  
    play.style.color='#999';  
    pause.style.color='black';  
    rewind.style.color='#999';  
}  
  
function rewindVideo() {  
    vid.currentTime = 0;  
    play.style.color='#999';  
    pause.style.color='#999';  
    rewind.style.color='black';  
}
```

Démonstration

<video> attributs du tag

- **src** : source de la vidéo
- **width et height** : dimension de la vidéo. Si non spécifiés, valent la largeur et la hauteur du fichier vidéo. Si vous spécifiez une dimension mais pas l'autre, le navigateur va ajuster la taille de la dimension non spécifiée afin de préserver les proportions de la vidéo.
- **controls** : Si cet attribut booléen est présent, le navigateur affiche ses propres contrôles vidéo pour la lecture et le volume. Si non présent la première image est affichée (ou l'image poster spécifiée). Lecture via événement JavaScript seulement.
- **poster** L'attribut poster permet de spécifier une image que le navigateur utilisera alors que la vidéo est en cours de téléchargement, ou jusqu'à ce que l'utilisateur commence la lecture de la vidéo. Si cet attribut n'est pas spécifié, la première image de la vidéo sera utilisée à la place

<video> attributs du tag (2)

- **autoplay** Spécifie au navigateur de lancer la lecture de la vidéo automatiquement.
- **autobuffer** : spécifie au navigateur de commencer le téléchargement de la vidéo tout de suite, en anticipant sur le fait que l'utilisateur lira la vidéo. (Cette partie de la spécification est actuellement en pleine mutation et sujette à changement)
- **loop** : un autre attribut booléen, indique de lire la vidéo en boucle.
- Tous les attributs ne sont pas encore supportés par tous les navigateurs, mais les plus courants le sont.

<video> table des propriétés, méthodes, événements

- Certaines propriétés ne sont disponibles qu'après le chargement du début de la vidéo

Methods	Properties	Events
play()	currentSrc	play
pause()	currentTime	pause
load()	videoWidth	progress
canPlayType	videoHeight	error
	duration	timeupdate
	ended	ended
	error	abort
	paused	empty
	muted	emptied
	seeking	waiting
	volume	loadedmetadata
	height	
	width	

<video> autre exemple interactif

- Modification de la taille de la vidéo, pendant qu'elle joue, depuis JavaScript
- Démonstration



<video> et CSS

- Tous les styles CSS, notamment les effets de transition, transformations géométriques et animation de CSS3 s'appliquent
- [Démonstration](#) (le style ci-dessous fait tout)
Noter 4 fois la même ligne pour compatibilité

```
video {  
  width: 75px;  
  -o-transition: all 0.5s ease-in-out;  
  -webkit-transition: all 0.5s ease-in-out;  
  -moz-transition: all 0.5s ease-in-out;  
  transition: all 0.5s ease-in-out;  
}  
video:hover, video:focus { width:600px; }
```

<video> et <canvas>

- On étudiera le canvas plus tard, mais sachez qu'on peut superposer un canvas (une zone dans laquelle on dessine) à un élément <video>, il suffit qu'ils aient le même père
- Démonstration



<video> autres démonstrations

- Ici avec contrôles en SVG, incrustations de sous-titres dans l'image



- Ici avec déformation CSS



Le tag <audio>

- Similaire au tag <video>

```
<audio controls="controls">  
  <source src="">
```

- Donne :

- Démonstration



- Comme pour <video> l'attribut controls ajoute des boutons de contrôle

<audio> et le DOM

```
var audio = document.createElement('audio');  
audio.src = 'audio.oga';  
audio.controls = true;  
document.body.appendChild(audio);
```

■ Ou encore, on peut utiliser un constructeur :

```
var audio = new Audio('audio.oga');
```

■ Ou

```
var audio = new Audio();  
audio.src = 'audio.oga';
```

Codecs supportés

- Là aussi, grosse bataille, le mp3 n'est pas supporté partout par exemple

Browser	MP3	Wav	Ogg
Internet Explorer 9	YES	NO	NO
Firefox 4.0	NO	YES	YES
Google Chrome 6	YES	YES	YES
Apple Safari 5	YES	YES	NO
Opera 10.6	NO	YES	YES

- Votre serveur doit fournir plusieurs formats
 - [Test interactif de votre navigateur ici](#)
 - Une des solution consiste sur un serveur Linux à transcoder l'audio à la volée dans un des format supporté par le navigateur client. [Le framework Gstreamer fait cela très bien](#)

Attributs, méthodes et événements

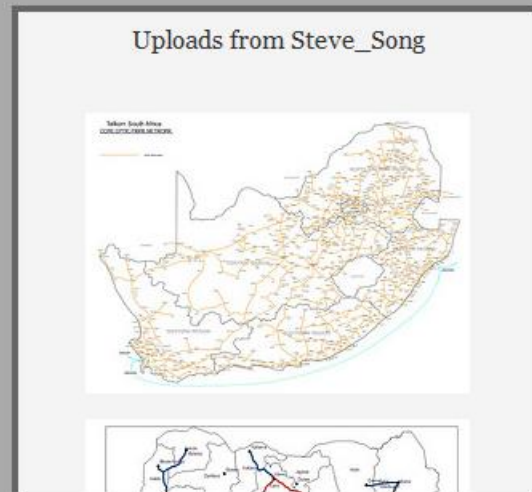
- Attributs : les mêmes que pour le tag vidéo à part « poster » : src, preload, autoplay, loop, controls
- Les méthodes sont les mêmes : play(), pause(), load()...
- Les événements sont les mêmes que ceux du tag vidéo, à 99%. On parle en fait de « media events » HTML5
 - Ex : onload, onpause, oncanplay, etc.
 - [Liste complète des Media Events](#)

Solution de fallback

- Il existe de nombreuses solutions pour que les tags audio et video fonctionnent même dans d'anciens navigateurs
 - Ces solutions utilisent un script qui va insérer dans le DOM un lecteur flash par exemple, si le format n'est pas supporté ou si HTML5 n'est pas supporté
- <http://html5media.info/> (open source, supporte les tags <audio> et <video>)
- Lecteur vidéo HTML5 + fallbacks
<http://videojs.com/>

Librairies multimedia évoluées

- [Popcorn.js](#) permet de synchroniser une vidéo ou un morceau de musique avec des effets sur le DOM de la page
 - Sous-titres, commentaires, transformer une vidéo en ascii, etc.



Web Sound API

- C'est tout nouveau, cela vient de sortir !
Supporté pour le moment par les dernières versions de Chrome ([sur le site des développeurs de Chrome](#))
- Cette API permet de générer du son synthétique mais aussi d'analyser en temps réel les sons qui sont joués.
- Démonstrations et explications :
<http://www.html5fivewow.com/slide55>

**Le tag <canvas> : dessin,
jeux, etc.**

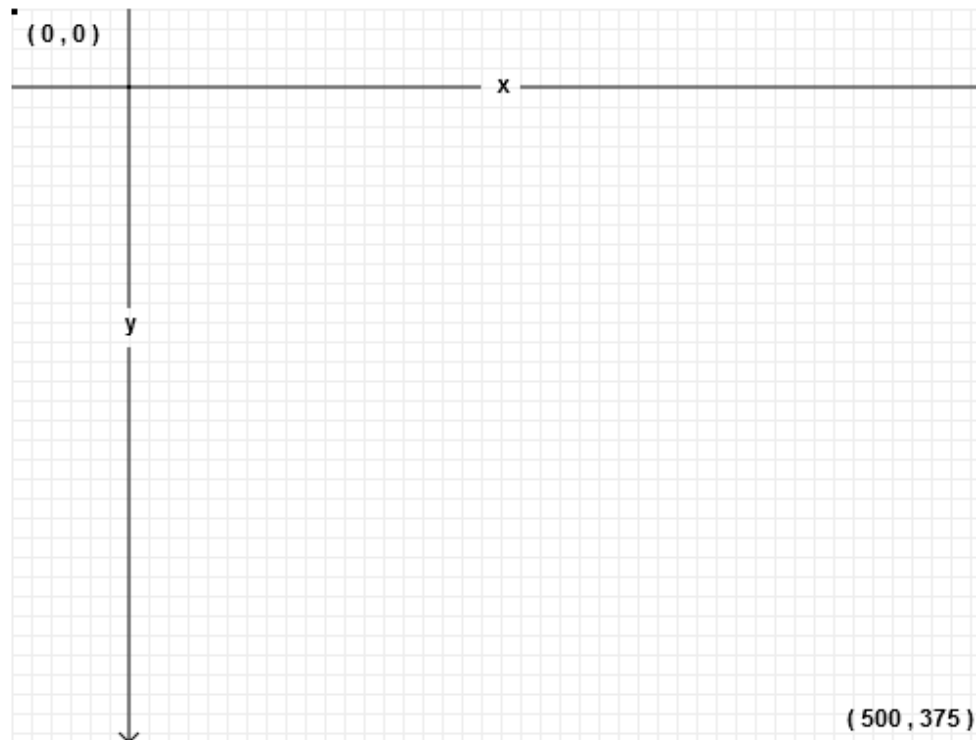
<canvas> permet de dessiner en bitmap

- Un des deux Flash-Killer de HTML5
- Dessin bitmap mais supporte l'accélération hardware (donc très performant)
- Fonctions proches de Java2D :
 - dessin de formes (droites, courbes, rectangles, texte),
 - Antialias, formes arrondies, support de la transparence,
 - Nombreux filtres (blur, etc),
 - Nombreuses fonctions de gestion d'images,
- Support de la 3D via WebGL, version web d'OpenGL

<canvas> système de coordonnées

- Source : un excellent tutorial sur le canvas : <http://diveinto.html5doctor.com/canvas.html>

Canvas coordinates diagram ~



<canvas> exemple simple

```
<canvas id="myCanvas" width="200"  
height="200">Votre navigateur ne supporte pas le  
tag canvas.</canvas>
```

```
<script type="text/javascript">  
  var canvas=document.getElementById('myCanvas');  
  var ctx=canvas.getContext('2d');  
  ctx.fillStyle='#FF0000';  
  ctx.fillRect(0,0,80,100);  
</script>
```



■ Démonstration

Principe d'utilisation

1. Déclarer le canvas, ne pas oublier l'attribut id et sa taille :

```
<canvas id="a" width="300" height="225"></canvas>
```

2. Récupérer dans une variable cet élément, dans du code JavaScript

```
var a_canvas = document.getElementById("a");
```

3. Récupérer le contexte graphique du canvas pour dessiner

```
var a_context = a_canvas.getContext("2d");
```

4. Dessiner à l'aide du contexte graphique

```
b_context.fillRect(50, 25, 150, 100);
```

- Le contexte est comme un “crayon” qui dessine dans l'objet qui a permis de l'obtenir

Principes du dessin

- Prenons l'exemple du rectangle des exemples précédents
 - **fillStyle** est une propriété du contexte, similaire à du CSS. Peut prendre comme valeur une couleur, une pattern (texture) ou un gradient (dégradé). Par défaut couleur = noir.

Toute le dessin en mode « plein » se fera avec cette propriété activée : les rectangles pleins seront noirs, les cercles pleins seront noirs, etc. Tant qu'on ne modifie pas cette propriété, tous les ordres de dessin la prendront en compte, c'est comme une variable globale du contexte

Principe du dessin

- **fillRect(x, y, width, height)** : dessine un rectangle plein. On indique le point en haut à gauche, la largeur et la hauteur. Utilise le fillStyle courant.
- **strokeStyle** est comme fillStyle mais pour les formes « en fil de fer », non pleines. Ex : un cercle dont on ne veut que le contours. Mêmes valeurs possibles que pour fillStyle.
- **strokeRect(x, y, width, height)** : idem fillRect mais rectangle en fil de fer, non plein. Utilise le strokeStyle courant.
- **clearRect(x, y, width, height)** : efface le rectangle courant (couleur = noir transparent, en fait le remet dans l'état initial)

Faire « reset » d'un <canvas>

- Oui, on peut remettre simplement le canvas dans l'état initial aussi en resettant la largeur ou sa hauteur

```
var b_canvas =  
document.getElementById("b");
```

```
b_canvas.width = b_canvas.width;
```

- Cela efface le canvas mais remet aussi son contexte dans l'état initial

Principe du dessin de lignes par chemin

- Contrairement à de nombreuses approches, pour dessiner des formes dans un canvas on utilise la notion de chemin (path).
 - On met le crayon à un endroit donné (moveTo)
 - On choisit la couleur et ce que l'on veut dessiner (strokeStyle ou fillStyle =)
 - On dit jusqu'où on veut dessiner (lineTo par exemple)
 - Encore un coup (lineTo, on trace deux lignes jointives...)
 - On dessine (stroke ou fill)
 - On aurait pu choisir la couleur juste avant le stroke ou le fill

Exemple de tracés de lignes

```
// Lignes verticales

for (var x = 0.5; x < 500; x += 10) {
    context.moveTo(x, 0);
    context.lineTo(x, 375);
}

// Lignes horizontales

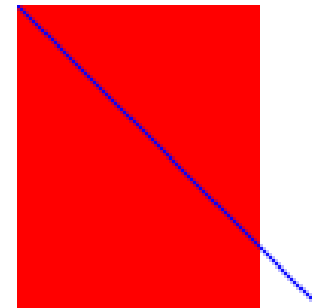
for (var y = 0.5; y < 375; y += 10) {
    context.moveTo(0, y);
    context.lineTo(500, y);
}

// Rien n'a été encore dessiné !!!
context.strokeStyle = "#0000FF"; // en bleu
context.stroke(); // Ah, là ça y est, on dessine !
```

Exemple complet

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas">Your browser does not support
the canvas tag.</canvas>
<script type="text/javascript">
  var canvas=document.getElementById('myCanvas');
  var ctx=canvas.getContext('2d');
  ctx.fillStyle='#FF0000';
  ctx.fillRect(0,0,80,100);

  ctx.moveTo(0,0);
  ctx.lineTo(100, 100);
  ctx.lineTo(100,0);
  ctx.strokeStyle = "#0000FF";
  ctx.stroke();
</script>
</body>
</html>
```



Attributs du dessin « fil de fer »

- Largeur du trait : **context.lineWidth=10;**
- Couleur du trait : **context.strokeStyle=couleur**,
gradient ou texture (exemple plus loin)
 - `context.strokeStyle = "#ff0000";`
 - `context.strokeStyle = "red";`
- Arrondis aux extrêmités : **context.lineCap=[value];**
 - `context.lineCap = "butt";`
 - `context.lineCap = "round";`
 - `context.lineCap = "square";`



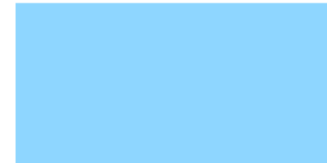
Gestion des ombres

■ Quatre attributs pour cela :

- `context.shadowColor`
- `context.shadowBlur`
- `context.shadowOffsetX`
- `context.shadowOffsetY`

■ Exemple :

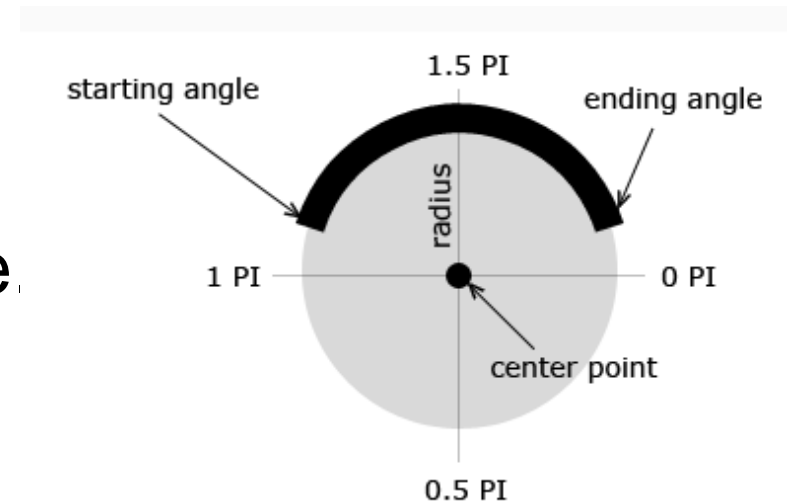
```
context.rect(188, 40, 200, 100);  
context.fillStyle = "#8ED6FF";  
context.shadowColor = "#bbbbbb";  
context.shadowBlur = 20; // floutage de l'ombre  
context.shadowOffsetX = 15; // décalage en X  
context.shadowOffsetY = 15; // décalage en Y  
context.fill();
```



Dessiner un Arc (de cercle)

```
context.arc(centreX, centreY, rayon,  
            angleDepart, angleArrivee,  
            sensInverseAiguillesMontre);
```

- Les angles en radians, le dernier paramètre est booléen. True indique qu'on travaille dans le sens inverse des aiguilles d'une montre quand on trace.
- Le sens n'est pas le sens trigonométrique classique, Piège !!!!



Piège du sens de progression des angles

```
context.arc(centerX, centerY, radius, 0, Math.PI/4, false);  
context.lineWidth = 15;  
context.strokeStyle = "black"; // line color  
context.stroke();
```

■ Donne :



■ Mais :

```
context.arc(centerX, centerY, radius, 0, Math.PI/4, true);  
context.lineWidth = 15;  
context.strokeStyle = "black"; // line color  
context.stroke();
```

■ Donne :

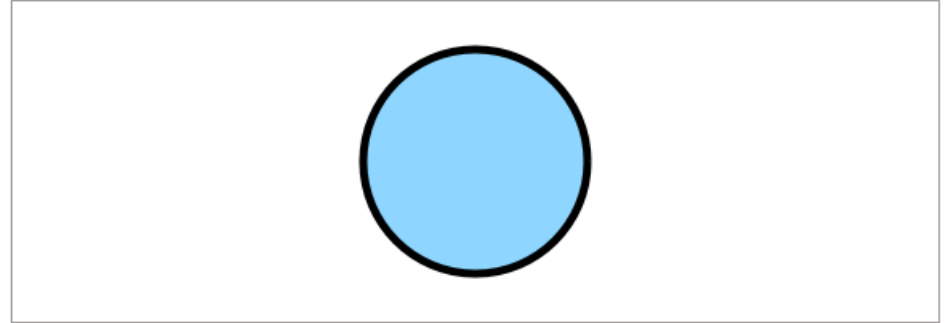


Dessiner un cercle

```
<!DOCTYPE HTML>
<html>
<head>
<style>
  #myCanvas {
    border: 1px solid #9C9898;
  }
</style>
<script>
  window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var centerX = canvas.width / 2;
    var centerY = canvas.height / 2;
    var radius = 70;

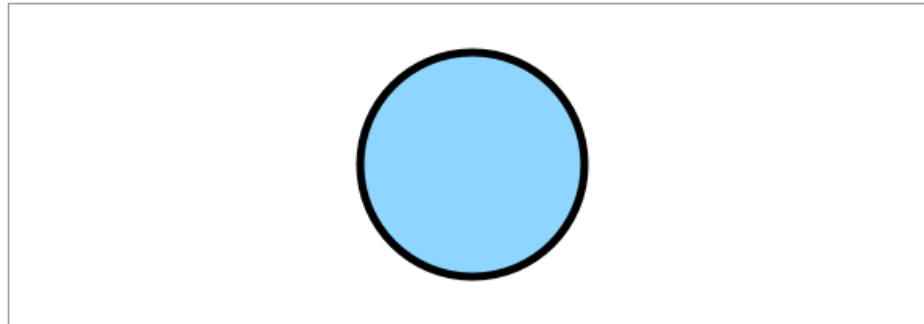
    context.beginPath();
    context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);
    context.fillStyle = "#8ED6FF"; // Cercle bleu plein
    context.fill(); // on dessine tout ce qui est en attente en plein
    context.lineWidth = 5;          // Cercle fil de fer noir, largeur trait 5
    context.strokeStyle = "black";

    context.closePath();
    context.stroke(); // on dessine tout ce qui est en attente en fil de fer
  };
</script>
```



Dessiner un cercle (fin)

```
<body>  
  <canvas id="myCanvas" width="578" height="200"></canvas>  
</body>  
</html>
```



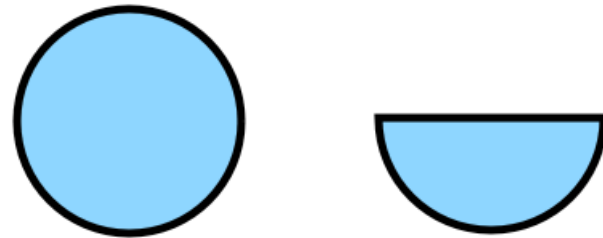
■ Démonstration !

Dessiner un demi-cercle

- Il suffit de remplacer dans l'exemple précédent

```
context.arc(centerX, centerY, radius, 0, 2*Math.PI, false);
```

- Par :



```
context.arc(centerX, centerY, radius, 0, Math.PI, false);
```

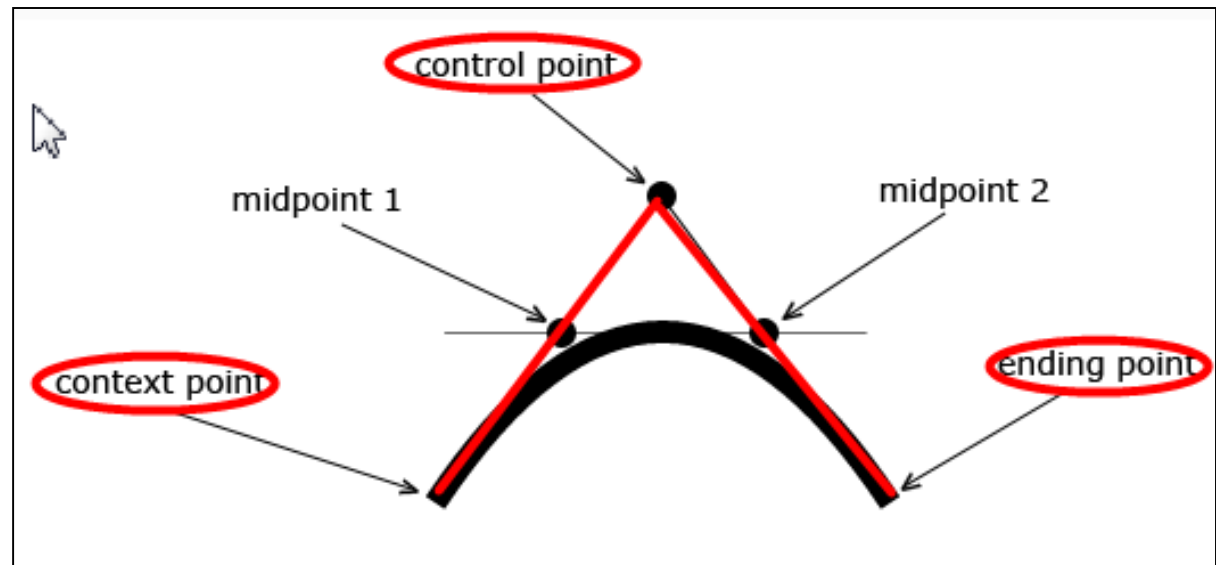
- Le dernier paramètre, dans le cas d'un cercle partiel, indique s'il vaut « true » que l'on désire dessiner non pas en allant dans le sens des aiguilles d'une montre, mais dans le sens inverse. On aurait eu le demi-cercle allant de 0 à π par le haut (le demi-cercle supérieur)

Courbes quadriques

```
context.moveTo(startX, startY);  
context.quadraticCurveTo(controlX, controlY,  
                           endX, endY);
```

- Ci-dessous, le « context point » est le dernier point du chemin en cours.

- Démonstration

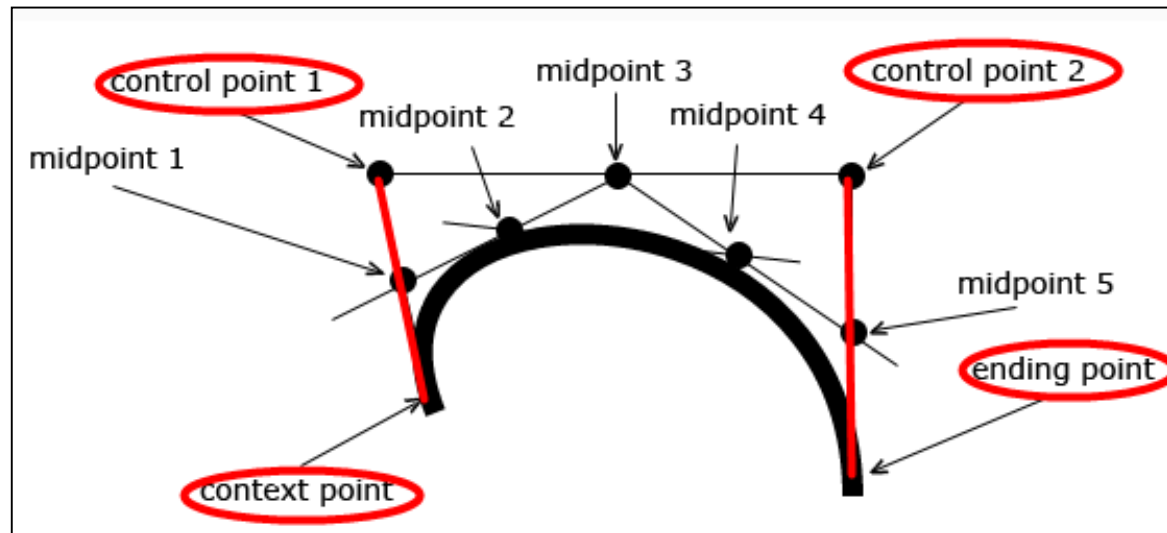


Courbes de Bezier

```
context.moveTo(startX, startY);  
context.bezierCurveTo(controlX1, controlY1,  
controlX2, controlY2,  
endX, endY);
```

Ci-dessous, le « context point » est le dernier point du chemin en cours.

■ Démonstration



Formes personnalisées

- On peut créer des formes personnalisées à l'aide des méthodes `beginPath()` et `closePath()` du contexte.

```
var canvas = document.getElementById("myCanvas");  
var context = canvas.getContext("2d");  
context.beginPath(); // début de la forme personnalisée  
  
... Ici des ordres de dessin  
  
context.closePath(); // fin de la forme  
  
// ensuite on dessine la forme, par exemple :  
context.lineWidth = 5;  
context.strokeStyle = "#0000ff";  
context.stroke();
```


Formes personnalisées (suite)

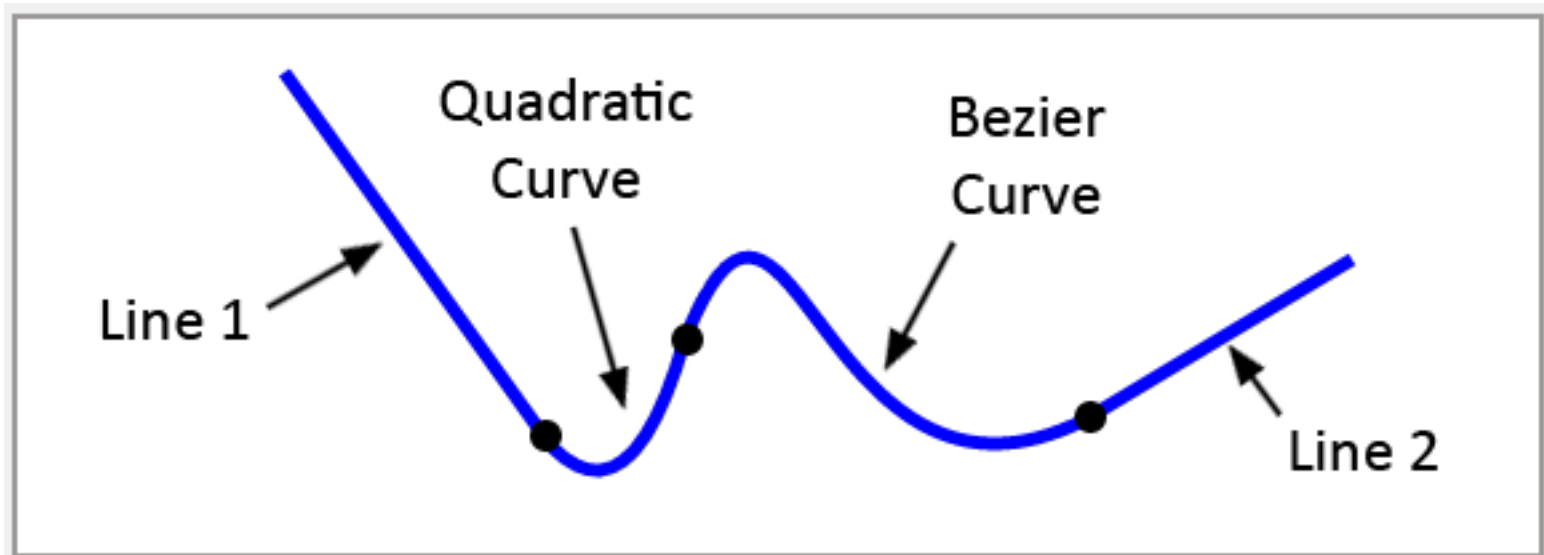
- Dans la partie entre le `beginPath()` et le `closePath()` on peut utiliser
 - `lineTo()`, `arcTo()`, `quadraticCurveTo()`, ou `bezierCurveTo()`

```
context.beginPath(); // début d'une forme personnalisée
context.moveTo(170, 80);
context.bezierCurveTo(130, 100, 130, 150, 230, 150);
context.bezierCurveTo(250, 180, 320, 180, 340, 150);
context.bezierCurveTo(420, 150, 420, 120, 390, 100);
context.bezierCurveTo(430, 40, 370, 30, 340, 50);
context.bezierCurveTo(320, 5, 250, 20, 250, 50);
context.bezierCurveTo(200, 5, 150, 20, 170, 80);
context.closePath(); // fin de la forme
context.lineWidth = 5;
context.strokeStyle = "#0000ff";
context.stroke();
```



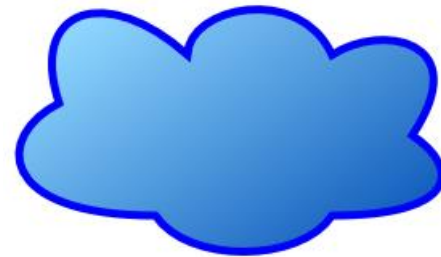
Formes personnalisées (suite)

```
context.beginPath();  
context.moveTo(100, 20);  
context.lineTo(200, 160);  
context.quadraticCurveTo(230, 200, 250, 120);  
context.bezierCurveTo(290, -40, 300, 200, 400, 150);  
context.lineTo(500, 90);  
context.closePath();  
context.lineWidth = 5;  
context.strokeStyle = "#0000ff";  
context.stroke();
```



Dessin de dégradés

- On appelle un dégradé « gradient » en anglais.
- On peut créer des objets de ce type pour des dégradés « linéaires » ou « radiaux »
- Exemple de forme avec dégradé radial et linéaire comme couleur de remplissage



- On peut cliquer ces exemples pour voir les démos

Création d'un gradient linéaire

- Gradient linéaire

```
var my_gradient =  
context.createLinearGradient(0, 0, 300, 0);
```

- On donne le point de départ (0,0) et le point d'arrivée (300, 0), situé 300 pixels plus à droite que le point de départ, dans cet exemple

- Puis on donne un ensemble de couleurs que le dégradé devra interpoler :

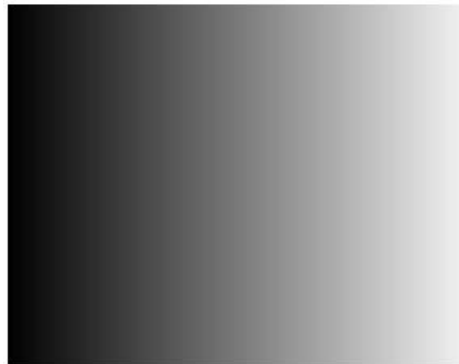
```
my_gradient.addColorStop(0, "black");  
my_gradient.addColorStop(1, "white");
```



Utilisation d'un gradient

- Il reste à mettre ce gradient comme valeur d'un fillStyle ou d'un strokeStyle

```
context.fillStyle = my_gradient;  
context.fillRect(0, 0, 300, 225);
```



- En changeant les points de départ et d'arrivée du gradient on peut faire un dégradé vertical, en changeant les couleurs un dégradé arc en ciel etc.

Création d'un gradient radial

- On utilise :

```
var grd=  
    context.createRadialGradient(startX, startY, startRadius,  
                                endX,    endY,    endRadius);
```

- On définit deux cercles imaginaires, un pour le départ et un pour l'arrivée
- On ajoute des « stop colors » comme pour le gradient linéaire (1^{er} paramètre entre 0 et 1)

```
grd.addColorStop(0, "red");  
grd.addColorStop(0.17, "orange");  
grd.addColorStop(0.33, "yellow");  
grd.addColorStop(0.5, "green");  
grd.addColorStop(0.666, "blue");  
grd.addColorStop(1, "violet");
```



Textures

- On peut utiliser également des textures avec `strokeStyle` et `fillStyle`

```
<head>
<script>
  window.onload = function(){
    var canvas = document.getElementById("myCanvas");
    var context = canvas.getContext("2d");
    var imageObj = new Image();

    // Fonction de callback asynchrone appelée par le chargement de l'image
    imageObj.onload = function(){
      // On entre ici lorsque l'image est chargée
      var pattern = context.createPattern(imageObj, "repeat");
      context.rect(10, 10, canvas.width - 20, canvas.height - 20);
      context.fillStyle = pattern;
      context.fill();
    };

    imageObj.src = "wood-pattern.png";
  };
</script>
</head>

<body>
  <canvas id="myCanvas" width="578" height="200"></canvas>
</body>
```



Textures

- La création de la texture :
`context.createPattern(imageObj, "repeat") ;`
- Valeurs possibles pour le dernier paramètre :
repeat (par défaut), repeat-x, repeat-y, ou no-repeat
- Démonstration

Images

- En fait, on vient de voir un exemple de chargement d'image asynchrone

```
var imageObj = new Image();  
  
// Fonction de callback asynchrone appelée par le chargement de l'image  
imageObj.onload = function(){  
    // On entre ici lorsque l'image est chargée, on dessine en 0,0 ici  
    context.drawImage(imageObj, 0, 0);  
};  
  
imageObj.src = "darth-vader.jpg";
```

- Démonstration

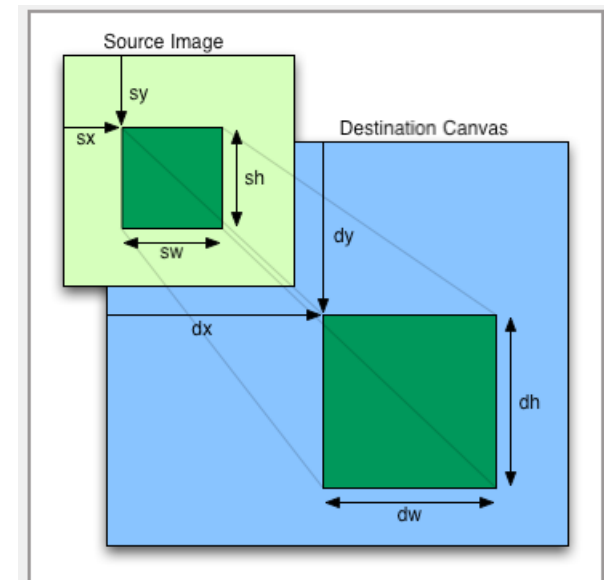


Nombreuses options pour drawImage

- Il existe plusieurs variantes pour drawImage
 - drawImage(img, x, y) : dessine l'image à la position x,y, l'image conserve sa taille
 - drawImage(img, x, y, tailleX, tailleY) : l'image est retournée au moment où elle est dessinée.

Démonstration.

- drawImage(img, sx, sy, sw, sh, dx, dy, dw, dh); permet de ne dessiner qu'une partie de l'image source dans le canvas, tout en spécifiant la taille. Démonstration.



Reference: www.whatwg.org

Dessiner l'image d'une vidéo dans un canvas avec drawImage()

```
<video id="video1" height="115px" width="140px  
autoplay="true" controls="true" src="snoop.mp4">...</video>
```

```
<canvas id="canvasDest"  
        height="115px" width="140px"></canvas>
```

```
<script type="text/javascript">  
  var video1 = document.getElementById("video1");  
  var videoWidth = video1.width;  
  var videoHeight = video1.height;  
  var canvasDest = document.getElementById("canvasDest");  
  var ctxDest = canvasDest.getContext("2d");  
  ctxDest.drawImage(video1, 0, 0, videoWidth, videoHeight);  
</script>
```

- On peut faire cela à la fréquence vidéo !
- Démonstration

Manipulation des pixels d'une image

- A partir du contexte d'un canvas on peut obtenir l'ensemble des pixels qui le composent.

```
bufferImage= ctx.getImageData(0, 0, width, height);
```

- La variable résultat est un tableau à 2 dimensions dont chaque élément est un pixel (R,G,B,A). Exemple pour accéder au ième pixel

```
var r = bufferImage[i + 0];  
var g = bufferImage[i + 1];  
var b = bPixels[i + 2];  
var a = bufferImage[i + 3];  
// on peut tester si le pixel est vert  
if (r == 0 && g == 255 && b == 0 && a == 255) {...}
```

Manipulation des pixels d'une image (suite)

- Application à l'incrutation vidéo... on extrait en temps réel le buffer image d'une vidéo,
- On extrait celui d'une autre vidéo,
- On regarde chaque pixel du buffer destination, s'il est vert, on le remplace par le même pixel du buffer source.
 - Résultat : on incruste la source dans la destination !
- Démonstration (chrome, IE, car vidéos en H.264)
- Explications

Dessin de texte

- Cas simple :

```
context.font = "40pt Calibri";
```

```
context.fillText("Hello World!", x, y);
```

Hello World!

- On peut utiliser les valeurs de `strokeStyle` et `fillStyle` déjà vues

Dessin de texte, exemples

```
context.font = "60pt Calibri";  
context.lineWidth = 3;  
context.strokeStyle = "blue";  
context.strokeText("Hello World!", x, y);
```

Hello World!

- Mais on peut faire du fill et du stroke !

```
context.fillStyle = "red";  
context.fillText("Hello World!", x, y);  
context.strokeStyle = "blue";  
context.strokeText("Hello World!", x, y);
```

Hello World!

Alignement de texte

- On peut « aligner » le texte dessiné par rapport à une ligne verticale imaginaire qui passe par la coordonnée x du (x,y) utilisé pour indiquer l'endroit où on dessine avec `strokeText()` ou `fillText()`
- `context.textAlign=[value];`
- Valeurs possibles : `start`, `end`, `left`, `center`, ou `right`
- [Démonstration](#)

Texte, concepts avancés

■ Non étudiés :

- Baseline, pour alignement des polices sur une ligne horizontale (pour les lettres montantes et descendantes)
- Calcul des boîtes englobantes (pour centrer un texte dans une zone par exemple) pour la justification (text wrap) d'un long texte à dessiner dans une zone rectangulaire.

Détection d'événements dans un canvas

- Le canvas peut générer de nombreux événements, notamment pour capturer les clics souris et les déplacements
 - Étudiés dans le TP sur le programme de paint

```
<script>
function writeMessage(canvas, message){
    var context = canvas.getContext('2d');
    context.clearRect(0, 0, canvas.width, canvas.height);
    context.font = '18pt Calibri';
    context.fillStyle = 'black';
    context.fillText(message, 10, 25);
}
window.onload = function(){
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');
    canvas.addEventListener('mousemove', function(evt){
        var message = "Mouse position: " + evt.clientX + ", "
        + evt.clientY;
        writeMessage(canvas, message);
}, false);
};
</script>
```

Détection d'événements dans un canvas

- Valeurs possibles très nombreuses, notamment mousedown, mouseup, mousemove mais aussi tous les événements de type Keyboard ou Mouse
 - Voir http://www.w3schools.com/html5/html5_ref_eventattributes.asp

Intéressant pour aller plus loin

- Le site <http://www.html5canvastutorials.com/> possède de très nombreux tutoriaux
 - Traitement d'image (filtres, etc)
 - Animation via `window.requestAnimationFrame` (équivalent du `repaint()` de Java)
 - Transformations géométriques
- Il existe de nombreuses librairies et framework pour utiliser des canvas avec une approche « de haut niveau »
 - processingJS.org
 - Outils commerciaux : bibliothèques JS, outils Adobe, Unity3D, etc.

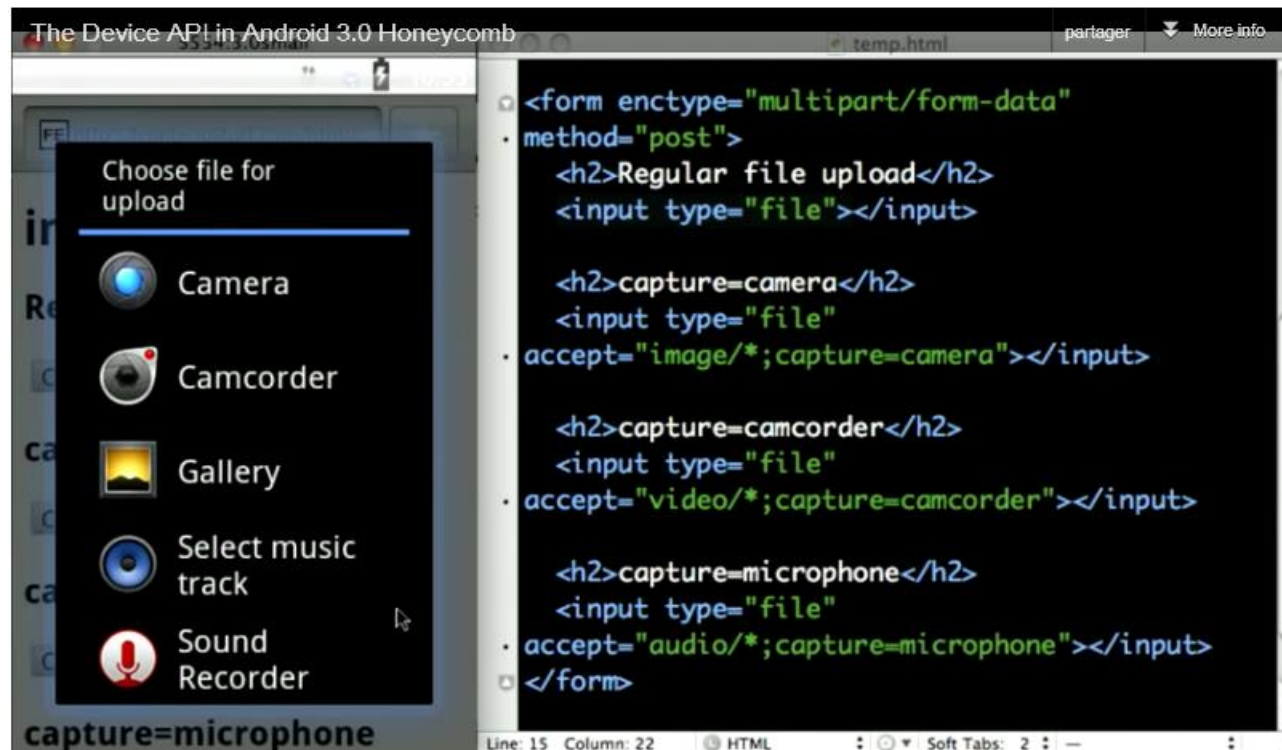
Utilisation des APIs de capture audio/video

Capture audio/vidéo

- Pendant des années : Flash, Silverlight, plugins DirectX et autres solutions ad-hoc...
- HTML5 propose déjà un accès au hardware
 - Géolocalisation, Orientation, WebGL (GPU), WebAudio API, etc.
- Et depuis peu (2012): capture audio/video via l'API `navigator.getUserMedia()`
- Resources :
 - <http://html5doctor.com/getusermedia/>
 - <http://www.html5rocks.com/en/tutorials/getusermedia/intro/>

Historique (1) : 2011, augmenter <input>

- [W3C Device APIs Policy \(DAP\) Working Group](#) essaie de mettre de l'ordre dans les APIs



- Support : Androïd 3.0 et Chrome pour Androïd 4.0/ICS

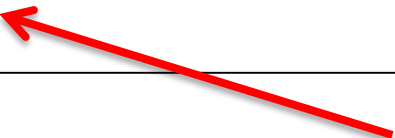
Historique (2) : augmenter <input>

- Problème de cette solution : difficile d'appliquer des effets temps réel
 - Rendu dans un <canvas> impossible,
 - Appliquer des filtres CSS ou via WebGL impossible
- Ne permet que l'enregistrement dans un fichier ou la prise d'un snapshot image
- Probable que ce ne soit jamais implémenté par Opera, Firefox, IE, etc. au profit de `navigator.getUserMedia()`

Historique (3) : 2011, le tag <device>

- La proposition précédente a été modifiée au profit du tag <device>
 - Pourra supporter de futur devices de capture
 - Première implémentation dans Opera, mais...

Last week we wrote a blog post discussing our internal prototyping of [web camera streaming in the browser](#). **On the very same day,** the proposed standard interface on which that was built [changed considerably](#).

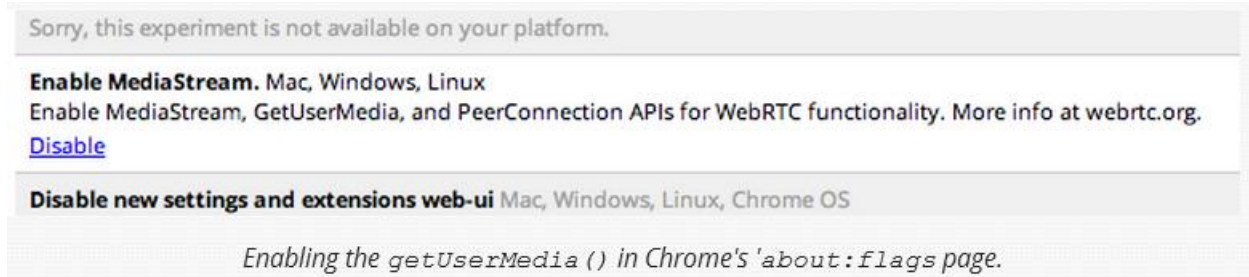


- Bye bye <device> ! **Un jour de vie !**

```
<device type="media" onchange="update(this.data)"></device>  
<video autoplay></video>  
<script>  
  function update(stream) {  
    document.querySelector('video').src = stream.url;  
  }  
</script>
```

Aujourd'hui : getUserMedia API

- [W3C WebRTC Working Group](#) : Web Real Time Communication, en charge de cette API
 - Google, Opera, Firefox + autres...
- getUserMedia : API JavaScript de capture, utilisant les couches RTC : [DEMONSTRATIONS](#)
- Implémenté dans Chrome 18+, activer via
 - about:flags et activer « MediaStream »
 - Ou lancer avec --enable-media-stream



getUserMedia API : test du support

- Opera Desktop et Mobile : support dans les [builds expérimentaux](#)
- Test du support :

```
function hasgetUserMedia() {  
    // Note: Opera builds are unprefixed.  
    return !! (navigator.getUserMedia ||  
                navigator.webkitgetUserMedia ||  
                navigator.mozgetUserMedia ||  
                navigator.msgetUserMedia);  
}  
  
...  
if (hasgetUserMedia()) {  
    // Good to go!  
} else {  
    alert('getUserMedia() not supported');  
}
```

getUserMedia : principe d'utilisation

```
<video autoplay></video>

...
<script>
  var onFailSoHard = function(e) {
    console.log('Reeeejected!', e);
  };

  // Not showing vendor prefixes.
  navigator.getUserMedia('video', function(localMediaStream) {
    var video = document.querySelector('video');
    video.src = window.URL.createObjectURL(localMediaStream);

    // Note: onloadedmetadata doesn't fire in Chrome when using it
    // with getUserMedia.
    // See crbug.com/110938.
    video.onloadedmetadata = function(e) {
      // Ready to go. Do some stuff.
    };
  }, onFailSoHard);
</script>
```

getUserMedia : remarques

- On peut passer plusieurs types de media en paramètres :

```
navigator.getUserMedia('video,  
audio'), successCb, errorCallback);
```

- Fonctionne en conjonction avec <audio> et <video>
- On positionne l'attribut src de ces tags avec un « blob URL »

```
window.URL.createObjectURL(localMedia  
Stream);
```

getUserMedia : encore des inconsistances

```
var video = document.querySelector('video');
if (navigator.getUserMedia) { // Opera
    navigator.getUserMedia({audio: true, video: true},
        function(stream) {
            video.src = stream;
        }, onFailSoHard);
} else if (navigator.webkitGetUserMedia) { // Chrome
    navigator.webkitGetUserMedia('audio, video',
        function(stream) {
            video.src =
                window.webkitURL.createObjectURL(stream);
        }, onFailSoHard);
} else {
    video.src = 'somevideo.webm'; // fallback.
}
```

getUserMedia : encore des inconsistances

- Cela ne devrait pas durer, les implémentations vont s'aligner
- En attendant : utiliser [gUM Shield](#), qui normalise les appels à getUserMedia(...)

Exemple : prendre une image instantanée

```
<video autoplay></video>
```

```
<canvas></canvas>
```

```
...
```

```
var video = document.querySelector('video');
```

```
var canvas = document.querySelector('canvas');
```

```
var ctx = canvas.getContext('2d');
```

```
var localMediaStream = null;
```

```
function snapshot() {
```

```
    if (localMediaStream) {
```

```
        ctx.drawImage(video, 0, 0);
```

```
    }
```

```
}
```

```
video.addEventListener('click', snapshot, false);
```

```
navigator.getUserMedia({video: true}, function(stream) {
```

```
    video.src = window.URL.createObjectURL(stream);
```

```
    localMediaStream = stream;
```

```
}, onFailSoHard);
```


On peut aussi appliquer des filtres CSS

```
<style>
    .grayscale { +filter: grayscale(1); }
    .sepia { +filter: sepia(1); }
    .blur { +filter: blur(3px); }
    ...
</style>
```

<video autoplay></video>

Démonstration

```
<script>
  var idx = 0;
  var filters = ['grayscale', 'sepia', 'blur', ''];

  function changeFilter(e) {
    var el = e.target;
    el.className = '';
    var effect = filters[idx++ % filters.length];
    if (effect) {
      el.classList.add(effect);
    }
  }
  document.querySelector('video').addEventListener('click',
    changeFilter, false);
</script>
```

HTML5 et le support pour mobiles/tablettes

CSS3 Media Queries ?

- C'est la possibilité d'adapter l'interface au périphérique utilisé :
 - Résolution,
 - Profondeur d'écran (nombre de couleurs),
 - Type d'écran : TV, encre électronique, tactile, etc
 - Braille, synthèse vocale, impression
 - Etc...
- Il existait déjà un attribut « media » dans CSS2

Ce que proposait CSS2

```
<head>
<link rel="stylesheet" media="screen"
      href="screen.css" type="text/css" />
<link rel="stylesheet" media="print"
      href="print.css" type="text/css" />
</head>
```

- En fonction de la valeur de l'attribut media, on chargeait une feuille de style adaptée
- Valeurs possibles pour media : screen, **handheld**, print, aural, braille, embossed (imprimante braille), projection (slides), tty (terminal ascii), tv, all

CSS2 : trop limité

- Autre syntaxe possible (dans le fichier CSS) :

```
@media print {  
    body { font-size:120%; color:black; }  
    ...  
}
```

- C'est le navigateur qui interprète cet attribut
- Le simple « handheld » pour devices portables, par exemple, est bien trop vague...
- Il est d'ailleurs ignoré par la majorité des navigateurs modernes pour smartphones/tablettes qui se déclarent de type « screen » (comme sur pc desktop)

CSS3 Media Queries : une évolution

- CSS3 propose une manière plus souple et plus précise de tester certains critères en combinant les contraintes
 - C'est pour cela qu'on parle de « requêtes » (queries)...

■ Exemples

```
<link rel="stylesheet" media="screen and (max-device-width: 640px)" href="smallscreen.css" type="text/css" />
```

```
@media screen and (max-device-width: 640px) {...}
```

```
@media screen and (min-device-width: 200px) and (max-device-width: 640px) {...}
```

CSS3 Media Queries : démonstrations

- <http://www.html5rocks.com/en/> retailer le navigateur ou mieux, tester sur un smartphone/tablette !
- Nombreuses démonstrations (pas forcément très ergonomiques) sur : <http://mediaqueri.es/>
- <http://webdesignerwall.com/demo/adaptive-design/final.html>
- <http://designshack.net/articles/css/20-amazing-examples-of-using-media-queries-for-responsive-web-design>

CSS3 Media Queries : syntaxe

- Opérateurs logiques : `and`, `only` et `not`, pas de `or`, il faut à la place plusieurs média queries à la suite, séparées par des virgules.
- On combine en général en premier le type de device avec un `and` suivi des contraintes sur des propriétés :

```
media="screen and (max-device-width: 640px) "
```

```
media="print and (min-width: 5in) "
```

(impression si le support est plus grand que 5
pouces)

CSS3 Media Queries : syntaxe

- Les propriétés suivantes peuvent être utilisées. Elles peuvent être préfixées par min- ou max- si elles ont des valeurs numériques (pixels, inches, pourcentage).

`color, colorindex, aspect-ratio, device-aspect-ratio, device-height, device-width, grid, height, width, monochrome, orientation, resolution, scan`

- Les dimensions peuvent être en px ou em
- Les ratios en fractions ex : 16/9
- La résolution en dpi (points par pouce) ou en dpcm (points par centimètres)
- Orientation vaut `paysage` ou `landscape`

CSS3 Media Queries : syntaxe, exemples

- `@media screen and (device-aspect-ratio: 16/9) { ... }`
- `@media (orientation:portrait) { ... }`
- `@media (color) { /* Ecran couleur */ }`
- `@media (min-color-index: 256) {
 /* écran support au moins 256
 couleurs */ }`
- `@media (monochrome) {
 /* L'écran est monochrome */ }`
- `@media (min-monochrome: 2) {
 /* L'écran a au moins 4 niveaux de
 gris */ }`
- `@media (tv, screen and (device-aspect-ratio: 16/9),
screen and (device-aspect-ratio: 16/10) {
 /* Ecran ou TV à écran large */ }`

CSS3 et Media Queries : bonnes pratiques

- Tester plutôt max-device-width et max-device-height plutôt que max-width et max-height, modifier le contenu d'une page sur un écran desktop quand on retaille la fenêtre peut-être perturbant.
- Si la page contient un canvas et que celui-ci est retaillé, il sera réinitialisé
 - A éviter car on perd son contenu !

CSS3 : astuce

- Pris sur [le site html5rocks](#), section « [mobifying your site](#) »

```
<link rel="stylesheet" media="all"
      href="/static/css/base.min.css" />
```

```
<link rel="stylesheet" media="only screen and
      (max-width: 800px) "
      href="/static/css/mobile.min.css" />
```

- Le **only** indique que la Media Query ne sera prise en compte que par les navigateurs qui les reconnaissent
- **screen** est utilisé par les navigateurs de smartphones/tablettes
- base.css est la feuille de style par défaut, mobile.min.css *redéfinit des règles CSS pour écran tel ou tablette*

Bon à savoir : les <meta> tags pour WebKit

- Bon, WebKit c'est le moteur HTML de 99% des navigateurs sur smartphones et tablettes (Android, IOS)
- Il reconnaît certains tags <meta>, le plus connu : name=viewport
- En positionnant ce tag, on indique :
 1. Comment le contenu doit remplir l'écran,
 2. Que le site est optimisé pour mobile.

<meta name=viewport content=... />

- Un exemple :

```
<meta name="viewport" content="width=device-width,  
initial-scale=1.0, user-scalable=yes">
```

- Initialement le viewport a la largeur de l'écran de l'appareil mobile, échelle 1 (pas de zoom)
- Le zoom est autorisé (à deux doigts si l'écran est multitouch)
- Comportement ok si il s'agit d'un site web avec du texte, etc.
- Pas bon si on a un jeu qui tourne dans un canvas (canvas retaillé = réinitialisé)

<meta name=viewport content=... />

- Autre exemple :

```
<meta name=viewport content="width=device-width,  
initial-scale=1.0, minimum-scale=0.5 maximum-  
scale=1.0">
```

- On donne des limites au zoom min et max
- Remarque : width peut prendre une valeur absolue en pixels. Par ex 320px, ce qui irait pour un iphone 3
 - Déconseillé, aujourd'hui on trouve des résolutions très différentes sur les devices disponibles.

Les tags <meta> spécifiques aux smartphones/tablettes

- Par rapport aux devices Apple, les navigateurs sous Android supportent un attribut pour indiquer la résolution pour laquelle le site a été développé :

```
<meta name="viewport" content="target-densitydpi=device-dpi">
```

- Valeurs possibles pour `target-densitydpi` : `device-dpi`, `high-dpi`, `medium-dpi`, `low-dpi`.

Android, astuces

- Android : si on veut modifier la page web pour différentes densités d'écran
 - Utiliser `-webkit-device-pixel-ratio` dans une media query CSS3 et/ou la positionner la propriété `window.devicePixelRatio` depuis JavaScript,
 - Ensuite, positionnez l'attribut `target-densitydpi` du tag `meta` à `device-dpi`.
 - Ceci indique à Android de ne pas rescaler la page web, et vous permet de faire les ajustements pour chaque densité depuis CSS ou JavaScript.
- Voir la documentation [WebView d'Android](#) pour plus de détails sur la gestion de la résolution pour des applications web.

IOS : astuces spécifiques, full screen browsing, déguiser une web app en native

```
<meta name="apple-mobile-web-app-capable"  
      content="yes">
```

```
<meta name="apple-mobile-web-app-status-bar-style"  
      content="black-translucent">
```

- Ces deux tags sont spécifiques à IOS/Apple
- Le premier indique que si la page est lue depuis un iPhone/iPad, le rendu de la page sera similaire à une application native (navigateur caché),
- Le second indique que la barre de statut doit être translucide.
- Voir [la documentation de référence de Safari](#) pour connaître les autres tags meta spécifiques IOS

Astuce pour cacher la barre d'adresse

- En scrollant la page, même de 1 pixel, on cache la barre d'adresse.

```
window.addEventListener('load', function(e) {  
    setTimeout(function() {  
        window.scrollTo(0, 1);  
    }, 1);  
}, false);
```

A ne faire que si on est sur un device mobile !

Android et iOS : icônes sur le Home

- iOS et Android reconnaissent des valeurs de l'attribut `rel` du tag `link` pour spécifier des icônes de raccourcis en cas d'ajout du site dans els favoris.

- Apple :

```
<link rel="apple-touch-icon"  
      href="/static/images/identity/HTML5_Badge_64.png" />
```

- Android :

```
<link rel="apple-touch-icon-precomposed"  
      href="/static/images/identity/HTML5_Badge_64.png" />
```

Bonne pratiques

- Penser à limiter le nombre de requêtes,
- Utiliser un « minifieur » de code JavaScript comme [Juicer](#), [YUI compressor](#) ou [Closure Compiler](#).
- Ne pas démarrer de vidéo, audio, dès le lancement...
- Voir :
<http://www.html5rocks.com/en/mobile/mobifying>
pour plus de conseils

Autres approche

- Phonegap.com propose de transformer une webapp en application native IOS ou Android
- Vous codez en JavaScript + HTML5, mais vous avez en plus accès aux API natives (webcam, contact, numérotation, sms, notifications, etc.)
- En gros : vous développez des web apps natives, mais avec les langages du web !

Evenements JavaScript sur mobiles

- **Attention, les événements « classiques » ont un comportement différent !**
- **Mouseover** et **Mousemove** ne réagissent pas car le mouvement du doigt sur l'écran fait défiler la page.
- **Keypress**, **Keyup** et **Keydown** ne sont pas pris en compte hors des *textareas*.
- **OnClick**, **Mousedown** et **Mouseup** sont étrangement appelés en même temps quand le doigt quitte l'écran.
- **DoubleClick** ne se déclenche pas, le *Double-Tap* effectue un zoom quoi qu'il arrive et un `event.preventDefault()` n'y changera rien.
- Le *One-finger panning* déclenche un **onScroll** une fois le mouvement stoppé.
- Le *Two-finger panning* déclenche un **mouseWheel** si l'élément est *scrollable*

Evenements « touch »

- Remplacent les « mouse events » sur les terminaux tactiles, bien que ces derniers fonctionnent toujours.
- **touchstart** (\approx mousedown) : déclenché lorsqu'un doigt touche l'écran
- **touchmove** (\approx mousemove) : déclenché lorsqu'un doigt touche l'écran puis se déplace
- **touchend** (\approx mouseup) : déclenché lorsqu'un doigt quitte l'écran.
- **touchcancel**, **touchleave** : déclenchés lorsque le doigt quitte la zone

Evenements « touch », exemple

- Exemple de traitement des événements par du code JavaScript :

```
document.addEventListener( 'touchstart',  
                           onTouchStart, false );  
document.addEventListener( 'touchmove',  
                           onTouchMove, false );  
document.addEventListener( 'touchend',  
                           onTouchEnd, false );
```

- Si on veut tester ce genre d'événements dans un canvas, s'assurer que le canvas utilise toute la surface

Evenements « touch »

- Tester sur un smartphone ou une tablette cet exemple :
<https://developer.mozilla.org/samples/domref/touchevents.html>
- Le site contient un excellent tutorial sur les touch events

Evènements « touch »

```
el.addEventListener("touchstart", handleStart, false);  
  
function handleStart(evt) {  
  
    evt.preventDefault(); // stoppe la propagation  
    var touches = evt.changedTouches;  
  
    for (var i=0; i<touches.length; i++) {  
        ongoingTouches.push(touches[i]);  
        var color = colorForTouch(touches[i]);  
        ctx.fillStyle = color;  
        ctx.fillRect(touches[i].pageX-2,  
                      touches[i].pageY-2, 4, 4);  
    }  
}
```

- L'évènement reçu possède des propriétés que l'on peut consulter. Ici la position du doigt.

Objet Touch

- Représente un point de contact entre un ou plusieurs doigts et l'écran
- `evt.changedTouches` = un objet de type `TouchList`
- Attributs possibles (en lecture seulement) :
 - `identifier` : id de l'événement,
 - `screenX`, `screenY` : coordonnées de l'endroit cliqué, par rapport à l'écran, n'inclut pas d'offset de scroll,
 - `clientX`, `clientY` : idem relatif au viewport, sans offset,
 - `pageX`, `pageY` : idem, mais inclut un offset si l'écran est scrollé,
 - `radiusX`, `radiusY` : rayons de l'ellipse correspondant à la zone touchée,
 - `rotationAngle` : angle de l'ellipse de rayons précédents,
 - `force` : pression entre 1 et 10, `target` : élément du DOM

Evenements « gesture »

- Pas d'équivalents sur ordinateur.
- Servent à contrôler le zoom ou la rotation sur un élément, le défilement etc.
- **gesturestart** : déclenché lorsqu'au moins deux doigts touchent l'écran
- **gesturechange** : déclenché lorsqu'au moins deux doigts bougent sur
- **gestureend** : déclenché lorsqu'au moins deux doigts quittent l'écran

Exemple d'utilisation via jQuery

```
$('element').bind('touchstart', function(event) {  
    var e = event.originalEvent ;  
    event.preventDefault() ; // ne pas propager  
    var firstFinger = e.touches[0].pageX ;  
})  
;  
$('element').bind('gesturechange', function(event) {  
    var e = event.originalEvent ;  
    event.preventDefault() ;  
    $(this).css('-webkit-transform',  
                'scale('+e.scale+')') ;  
})  
;
```

- position en x du premier doigt : e.touches[0].pageX
- facteur de multiplication pour le zoom : e.scale
- Event.preventDefault() coupe la propagation des événements touch, gesture ou mouse

HTML5, Web Sockets

Communication temps réel full-duplex

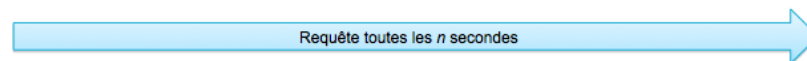
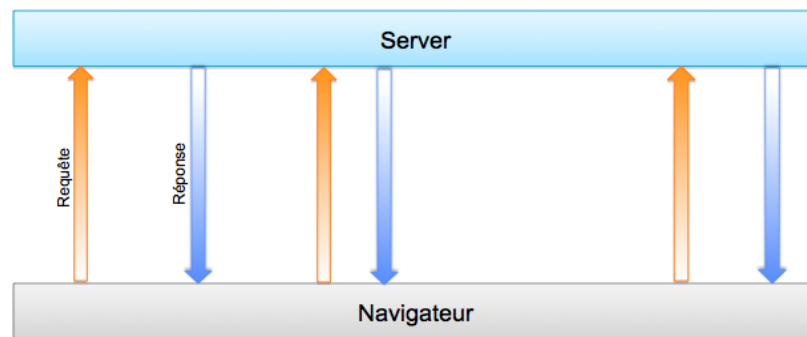
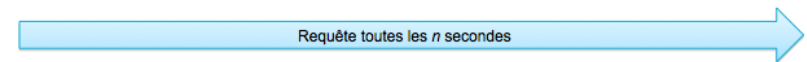
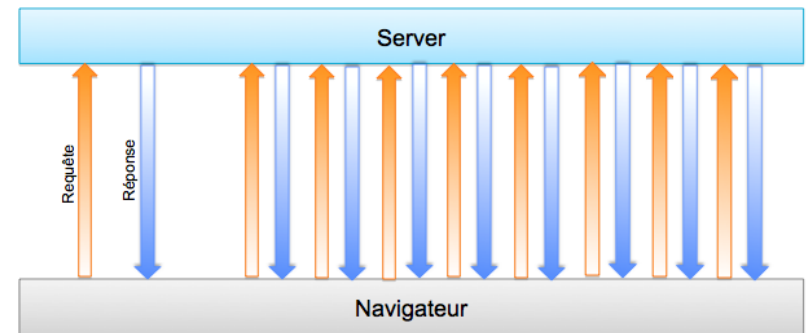
- Pour les jeux temps réel : Flash Sockets

- Ajax

- Requêtes/réponses
- Ouverture/fermeture de connexions

- Long Polling (COMET)

- Moins de requêtes,
- Connexion ouverte plus longtemps



Ajax/COMET / WebSockets

■ Avant tout les performances

- Header HTTP envoyé avec chaque requête/réponse
- Ouverture/fermeture de connexions,
- Pas de push

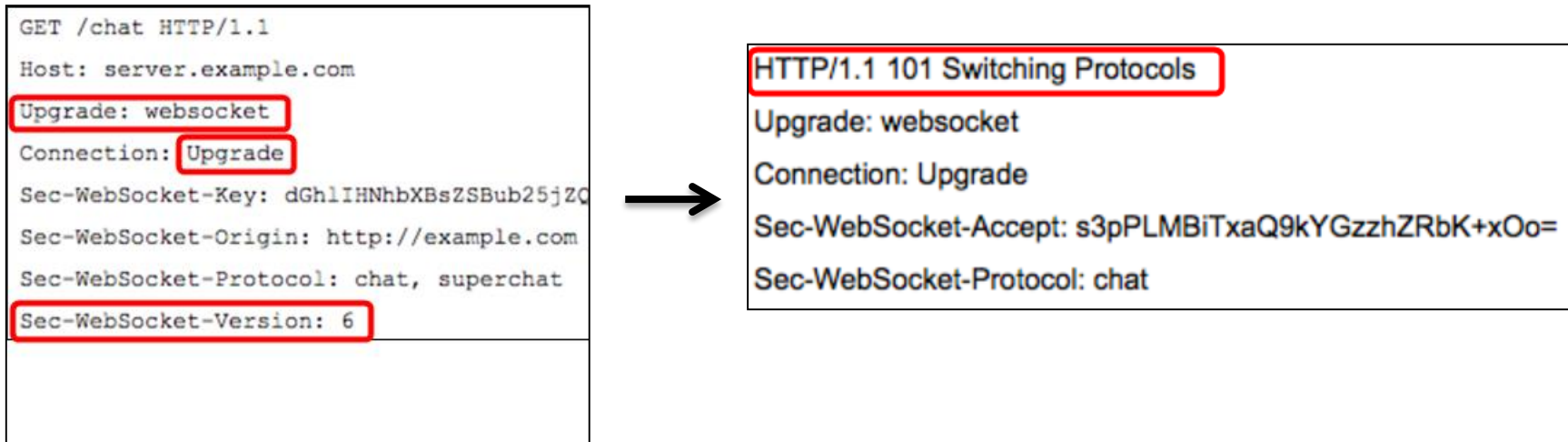
```
Request URL:http://test.com
Request Method:GET
Status Code:200 OK
Request Headersview source
Accept:text/html, */*; q=0.01
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Authorization:Basic c3Jlbm5vOnNlY3VyaXR5
Connection:keep-alive
Host:test.com
Referer:http://test.com
User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_1) AppleWebKit/53
X-Requested-With:XMLHttpRequest
Query String Parameters (4)
Response Headersview source
Connection:Keep-Alive
Content-Encoding:gzip
Content-Length:22
Content-Type:text/html; charset=UTF-8
Date:Mon, 19 Sep 2011 19:55:20 GMT
Keep-Alive:timeout=5, max=100
Server:Apache
Vary:Accept-Encoding,User-Agent
X-Powered-By:Servlet/2.5 Server: Sun Java System Application Server
```

■ WebSockets

- canal de communication bidirectionnel,
- Transparent pour les firewalls, proxy, et routeurs.
- Partage des ports 80 et 443.
- Etend n'importe protocole basé sur TCP/UDP
- 2 octets par trame (données entre 0x00 et 0xFF)
- Connexion permanente

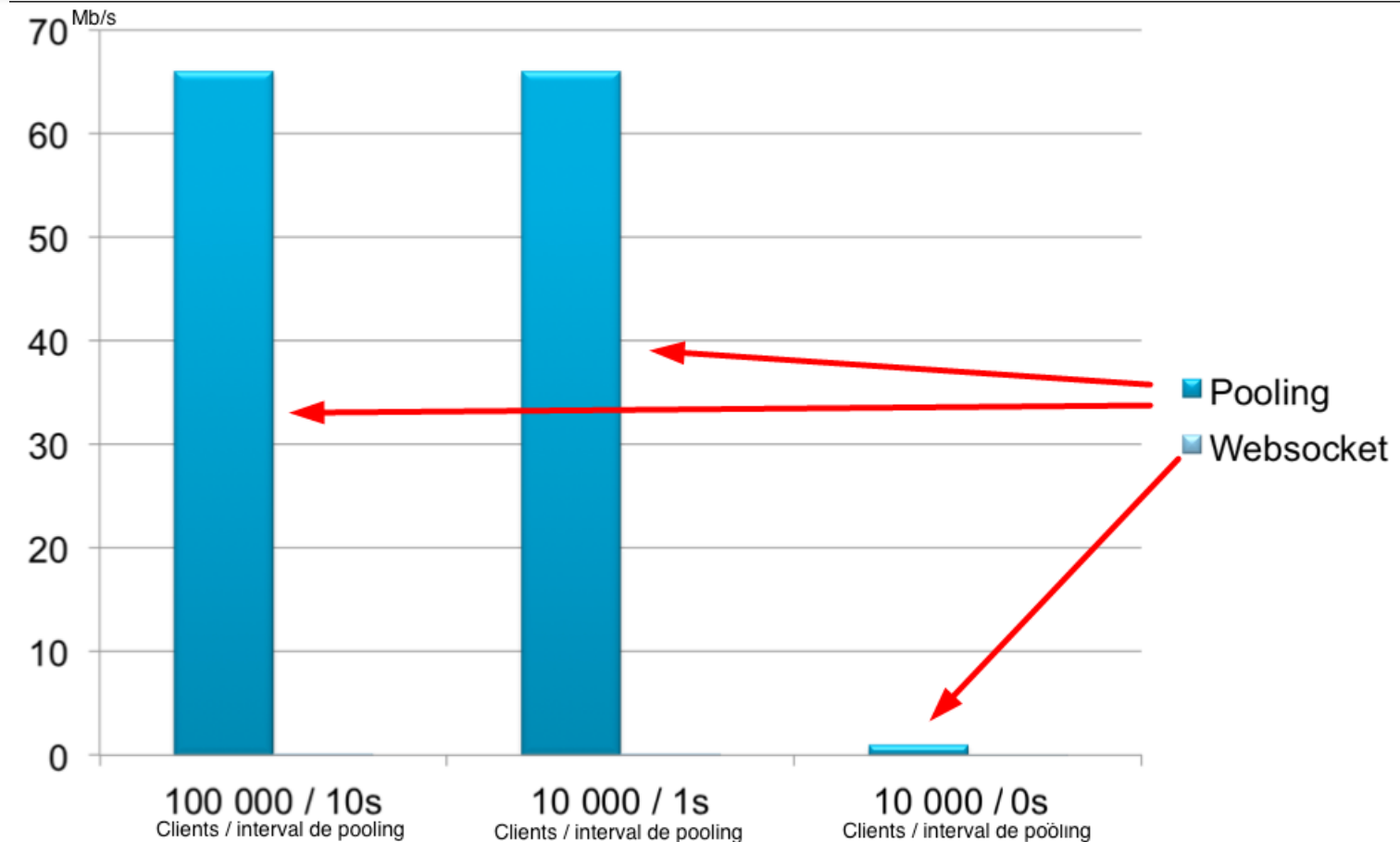
WebSockets

- Handshake sur HTTP lors de l'ouverture de connexion



- Nombreuses versions en 2011 (13 différentes !)
- Données encodées en UTF-8 (pas de binaire donc)

Performances



- On a simulé un chat avec 10.000 personnes simultanées avec un serveur de WebSockets dans NodeJS sur un laptop : ça marche !

Mise en œuvre : support des navigateurs

- Côté client, il faut que le navigateur supporte les WebSockets
 - Tester sur www.websocket.org



- Pas de panique, [on peut les activer dans Opera](#) !
- Support dans IE10, mais pour installer IE10, il faut installer... Windows 8 (huh !)

Mise en œuvre : API cliente

- Initialisation de la connexion

```
var ws = new WebSocket("ws://host:port");
```

- Évènement déclenché lors d'une connexion réussie

```
ws.onopen = function(evt) { alert("Connection open!"); };
```

- Évènement déclenché lors de la réception d'un message

```
ws.onmessage = function(evt) { alert( "Message reçu: " + evt.data); };
```

- Évènement déclenché lors de fermeture du WebSocket

```
ws.onclose = function(evt) { alert("Connection closed."); };
```

- Évènement déclenché lors d'une erreur

```
ws.onerror = function(evt) { alert("WebSocket error : " + evt.data) };
```

- Pour envoyer une donnée au serveur :

```
ws.send(MyMessage);
```

- Pour fermer le WebSocket

```
ws.close();
```

Mise en œuvre côté serveur

- De nombreux serveurs proposent en standard le support pour les WebSockets
 - [mod_pyWebSocket](#) (extension en Python / Apache)
 - Glassfish à partir de la version
 - [Jetty](#) (Java) à partir de la V7
 - [Netty](#) (Framework Java client serveur)
 - Tomcat 7
 - [Web Socket Ruby](#) (Ruby)
 - Kaazing WebSocket Gateway (commercial)
 - [JWebSocket](#) (Java)
 - [Socket.IO-Node](#) (NodeJS)
 - [NowJS](#) (NodeJS)

Très populaire : socket.io

- Bibliothèques client/serveur en JavaScript pour le micro-serveur NodeJS
- Code serveur écrit en JavaScript
- Socket.io propose des mécanismes de fallback si le navigateur ne supporte pas les WebSockets
 - Bascule vers Flash Sockets, sinon vers COMET, sinon vers Ajax...etc.
 - Supporte les déconnexions/reconnexions

Objets distribués sur WebSockets : NowJS

- Construit au-dessus de socket.io
- Permet de partager des objets JavaScript entre le client et le serveur,
- Code très concis et naturel
- Comparer le code d'une application de chat
 - [Version Java](#),
 - [Version PHP](#),
 - [Version NowJS](#)

Chat avec NowJS

■ Dans le client

```
<script src="/nowjs/now.js"></script>

<script>
$(document).ready(function() {
    now.receiveMessage = function(name, message){
        $("#messages").append("<br>" + name + ": " + message);
    }
    $("#send-button").click(function() {
        now.distributeMessage($("#text-input").val());
        $("#text-input").val("");
    });
    // Poppe une fenêtre de dialogue et demande le nom
    now.name = prompt("What's your name?", "");
});
</script>
```

■ Code Serveur

```
var html = require('fs').readFileSync(__dirname+'/simpleChat.html');
var server = require('http').createServer(function(req, res){
    res.end(html);
});
server.listen(8080);

var nowjs = require("now");
var everyone = nowjs.initialize(server);

everyone.now.distributeMessage = function(message){
    everyone.now.receiveMessage(this.now.name, message);
};
```

NowJS : démonstrations

- Ok, place à des démos, et n'oubliez pas :
 - NowJS est basé sur socket.io, donc marche dans tout navigateurs, mêmes si les WebSockets ne sont pas supportés,
 - Gère les déconnexions
 - Sérialisation/désérialisation JSON en coulisse,
 - RPC JavaScript naturel
 - Nombreux mini projets réalisés par les étudiants de Master informatique.

HTML5, Web Fonts

CSS3 @font-face

- Utilisation de polices de caractères customs !
 - Polices via URLs
 - Manipulables via CSS : transformations 2D et 3D, line-height, letter-spacing, text-shadow, text-align, selecteurs ::first-letter et ::first-line
 - Tout en restant “searchables (ctrl-F)”, selectionnables, etc.



Utilisation via CSS

- Charger la police à partir d'un URL et lui donner un identifiant pour le reste de la CSS

```
@font-face {  
    font-family: Chunkfive;  
    src: url('fonts/Chunkfive.otf') ;  
}
```

- Utiliser cette police dans une font-stack classique, avec fallbacks

```
h1, h2, h3 { font-family: Chunkfive,  
Georgia, Palatino, Times New Roman,  
serif} ;
```



Utilisation via CSS

- Ajoutant un peu de piment

```
h1, h2, h3 {  
    font-family: Chunkfive, Georgia,  
                Palatino, Times New Roman, serif;  
    text-shadow: rgba(0, 0, 0, 0.5) 0 7px 7px;  
};
```

Un exemple de H1

- Et si on rajoute : `-webkit-box-reflect: below 10px;`

Un exemple de H1
Un exemple de H1

Plusieurs types de polices / support

- **Embedded OpenType (EOT):** IE5-9
- **True Type (TTF):** FF 3.5+, Chrome 4+, Safari 4+, IE9
- **Open Type (OTF):** FF 3.5+, Chrome 4+, Safari 4+, IE9, Opera 10.5+
- **Web Open Font (WOFF):** Firefox 3.6+, IE9
- Les navigateurs mobile comme Safari sur iPad et iPhone requierent **SVG**.
- **ET ALORS ?????? ON FAIT COMMENT ???**

Support multi-browsers

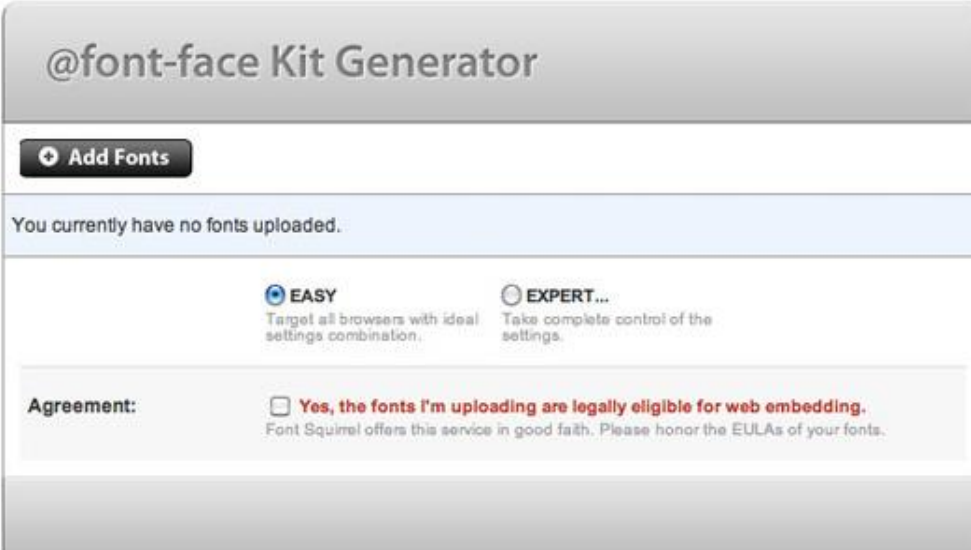
- Solution en attendant mieux : Paul Irish's [Bulletproof @font-face syntax.](#)

```
@font-face {  
  font-family: 'ChunkFiveRegular';  
  src: url('Chunkfive-webfont.eot');  
  src: local(,òf'),  
        url('Chunkfive-webfont.woff') format('woff'),  
        url('Chunkfive-webfont.ttf') format('truetype'),  
        url('Chunkfive-webfont.svg#webfont') format('svg');  
  font-weight: normal;  
  font-style: normal;  
}
```

- Astuce : désactiver **les polices locales** ([donne en général de mauvais résultats](#))
- Initialiser **weight** et **style** force les valeurs par défaut (Chrome et IE n'ont pas les mêmes)

Fournisseurs de Web Fonts gratuites

- [Google Web Fonts](#) (TTF), [Fontex](#) (OTF)
- [Font Squirrel](#) fournit les polices dans tous les formats + la CSS qui va avec (cf slide précédent)
- Si on a une police dans un seul format :



The screenshot shows the '@font-face Kit Generator' web interface. At the top, there's a header with the title '@font-face Kit Generator'. Below it is a dark button labeled 'Add Fonts'. A light blue message box states 'You currently have no fonts uploaded.' Below this, there are two radio button options: 'EASY' (selected) with the description 'Target all browsers with ideal settings combination.', and 'EXPERT...' with the description 'Take complete control of the settings.' At the bottom, there is an 'Agreement:' section with a checkbox that is checked, followed by the text 'Yes, the fonts I'm uploading are legally eligible for web embedding.' and a smaller line of text: 'Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.'

WebFont as a service

■ Google Font API :

```
<html>
<head>
  <link rel="stylesheet" type="text/css"
    href="http://fonts.googleapis.com/css?family=Lobster">
  <style>
    h1 {
      font-family: 'Lobster', serif; font-size: 48px;
    }
  </style>
</head>
<body>
  <h1>Hello World !h1>
</body>
</html>
```

■ Possibilité de demander plusieurs polices à la fois :

<http://fonts.googleapis.com/css?family=Vollkorn|Yanone>

Fournisseurs commerciaux

- De nombreuses polices sont copyrightées, la gestion des droits d'utilisation dans le contexte du Web est un cauchemar !
- Des prestataires proposent des licences simplifiées pour les polices les plus populaires (au site, à l'année, etc.)
 - [TypeKit](#), [Typotheque](#), [Fonts Live](#), [Kernest](#) (similaire à Google Web Font API), etc.

HTML5, drag'n'drop

Principe : 1) déclarer un élément draggable

- On ajoute à un élément draggable l'attribut **draggable=true**
- On indique une fonction JavaScript qui sera appelée dès qu'on commence à déplacer l'objet : **ondragstart =...**

```

```

[Démonstration](#)

Principe : 2) indiquer ce que l'on déplace comme données

- On spécifie ce qu'il va se passer lorsque l'élément est déplacé (draggué)

```
function drag(ev) {  
    ev.dataTransfer.setData("Text", ev.target.id) ;  
}
```

- L'attribut ondragstart appelle la fonction `drag(event)`, qui indique la donnée qui va être “dragguée”.
- La méthode `dataTransfer.setData()` positionne le type et la valeur de la donnée qui va être transférée
 - Dans l'exemple, type=Text et valeur = l'id de l'image

Principe : 3) indiquer où on droppe, ondragover

- On indique à l'aide d'un attribut **ondragover** le ou les éléments qui peuvent recevoir le drop

```
<div id="div1" ondrop="drop(event)"  
  ondragover="allowDrop(event)">  
</div>
```

```
function allowDrop(ev) {  
  ev.preventDefault();  
}
```

- Lors d'un drop c'est la fonction indiquée par l'attribut **ondrop=** qui est appelée (page suivante)
- Par défaut, le drop n'est pas autorisé, il faut indiquer qu'on l'autorise dans la fonction de drop :
event.preventDefault()
 - La valeur par défaut = ouvrir en tant que lien, pas récupérer les données

Principe : 4) que faire dans le drop

```
function drop(ev) {  
    var data=ev.dataTransfer.getData("Text");  
    ev.target.appendChild(document.getElementById(data));  
    ev.preventDefault();  
}
```

- On récupère les données, celles positionnées par le setData, elles doivent être du même type que dans le getData(), ici type="Text".
- On rajoute l'élément droppé comme fils de l'élément récepteur

HTML5 et la persistance

Introduction

- Il existe plusieurs moyens de faire de la persistance via des APIs HTML5.
- Certaines, comme WebSQL ne sont pas encore standardisées car toutes les implémentations utilisent la même base de données (SQLite), or pour que le W3C valide, il en faut au moins deux

WebStorage (Local et Session Storage)

- Sortes de “super cookies”
 - Plus sûr que les cookies, plus rapide,
 - Les données ne sont plus renvoyées au serveur avec chaque requête, mais à la demande,
 - Plus de limitation de taille ni de nombre (4k pour cookies, 20 par site, 300 en tout).
- Les données sont stockées sous la forme de paires clé/valeur et une page ne peut accéder qu’aux données qu’elle a stockée elle-même.
- LocalStorage : pas de date d’expiration
SessionStorage : durée = la session

WebStorage : support

- Supporté par tous les navigateurs récents
- Test par :

```
if (typeof(Storage) !== "undefined") {  
    // C'est supporté !  
    // Traitement  
} else {  
    // Désolé votre navigateur ne supporte pas le Web Storage  
}
```

L'objet LocalStorage

```
<div id="result"></div>
<script>
if(typeof(Storage) !== "undefined") {
    localStorage.lastname="Smith"; // un set !
    // dans la ligne ci-dessous localStorage.lastname fait
    // un get
    document.getElementById("result").innerHTML=
        "Last name: " + localStorage.lastname;
} else {
    document.getElementById("result").innerHTML="Sorry,
        your browser does not support web storage...";
}
</script>
```

Démonstration

L'objet sessionStorage

- S'utilise de la même manière si ce n'est que les données disparaissent lorsqu'on ferme le navigateur. [Démonstration](#)

```
<script>
function clickCounter() {
    if(typeof(Storage) !== "undefined") {
        if (sessionStorage.clickcount) {
            sessionStorage.clickcount=
                Number(sessionStorage.clickcount)+1;
        }else {
            sessionStorage.clickcount=1;
        }
        document.getElementById("result").innerHTML=
            "You have clicked the button " +
                sessionStorage.clickcount + " time(s) in
            this session.";
    } else { // pas de support }
}
</script>
```

IndexedDB

- Pas encore standardisé, supporté par Chrome et FireFox
- IndexedDB est un « object store », pas une BD relationnelle avec des tables.
- Les objets stockés sont en JSON, et peuvent être indexés en fonction de leurs attributs.
- Pas de SQL propose des requêtes sur des indexs, on obtient un curseur que l'on va utiliser pour itérer sur les résultats.

IndexedDB : fonctionnement asynchrone

- Chaque opération va renvoyer un résultat via une fonction de callback. Ainsi chaque opération rend la main immédiatement, elle est non bloquante.
- Les opérations sont transactionnelles. On ne peut manipuler de données en dehors d'une transaction.
- L'exemple illustré ici est celui [des slides de html5rocks](#)
 - Voir explications : <http://www.html5rocks.com/en/tutorials/indexeddb/to-do/>

IndexedDB : informations importantes

- Spécification encore en Béta version
- Voir les très bons slides :
<http://www.slideshare.net/mikewest/intro-to-indexeddb-beta>
- Nous explicitons ici un exemple bien connu.

Exemple étudié

Output

- Faire le ménage [Delete]
- Nettoyer la cage des oiseaux [Delete]

What do you need to do?

Add Todo Item

IndexedDB : ouverture d'une base

```
// On fait de l'objet. On associe un namespace dans ces
// exemples
var html5rocks = {};
html5rocks.indexedDB = {};

html5rocks.indexedDB.db = null;
html5rocks.indexedDB.open = function() {
    var request = indexedDB.open("todos");
    request.onsuccess = function(e) {
        html5rocks.indexedDB.db = e.target.result;
        // ... traitement
    };

    request.onfailure = html5rocks.indexedDB.onerror;
};
```

- En rouge la fonction d'ouverture/création d'une BD, en vert les fonctions de callback

Création d'un object store (eq table...)

```
request.onsuccess = function(e) {
  var v = "1.0";
  html5rocks.indexedDB.db = e.target.result;
  var db = html5rocks.indexedDB.db;
  // Création seulement dans une transaction setVersion
  if(v!= db.version) {
    var setVrequest = db.setVersion(v);
    // setVrequest.onsuccess est le seul endroit où on
    // peut créer un objectStore

    setVrequest.onfailure = html5rocks.indexedDB.onerror;
    setVrequest.onsuccess = function(e) {
      var store = db.createObjectStore("todo",
                                      {keyPath: "timeStamp"});
      html5rocks.indexedDB.getAllTodoItems();
    };
  }
  html5rocks.indexedDB.getAllTodoItems();
};
request.onfailure = html5rocks.indexedDB.onerror;
```

Création d'un object store

- Dans l'exemple précédent, on vérifie que la BD est bien en version 1.0, si c'est un succès, alors on crée un object store intitulé « todo » et qui contient la date de création (objet JSON : `{keyPath: "timeStamp"}`)

Ajout d'éléments dans l'object store

```
html5rocks.indexedDB.addTodo = function(todoText) {  
    var db = html5rocks.indexedDB.db;  
    var trans = db.transaction(["todo"], IDBTransaction.READ_WRITE, 0);  
    var store = trans.objectStore("todo");  
    var request = store.put({  
        "text": todoText,  
        "timeStamp" : new Date().getTime()  
    });  
  
    request.onsuccess = function(e) {  
        // On relit tous le contenu pour affichage  
        html5rocks.indexedDB.getAllTodoItems();  
    };  
  
    request.onerror = function(e) {  
        console.log(e.value);  
    };  
};
```

Récupération d'éléments dans un object store

```
html5rocks.indexedDB.getAllTodosItems = function() {
  var todos = document.getElementById("todoItems");
  todos.innerHTML = "";
  var db = html5rocks.indexedDB.db;
  var trans = db.transaction(["todo"], IDBTransaction.READ_WRITE, 0);
  var store = trans.objectStore("todo");

  // Récupérer tous les résultats
  var keyRange = IDBKeyRange.lowerBound(0); // on cherche depuis l'index
                                              // KeyRange = 0... KeyRange est
                                              // une sort de clé primaire
                                              // pour les objets que l'on a
                                              // stockés. Le KeyPath
                                              // indiqué à la création de la
                                              // base est un timestamp (date)

  var cursorRequest = store.openCursor(keyRange);

  cursorRequest.onsuccess = function(e) {
    var result = e.target.result;
    if(result == false)
      return;

    renderTodo(result.value);
    result.continue(); // car le callback onsuccess est juste pour un
                      // résultat, en appelant continue on passe au résultat suivant...
  };

  cursorRequest.onerror = html5rocks.indexedDB.onerror;
```

Manipulation des résultats pour affichage

```
function renderTodo(row) {  
    var todos = document.getElementById("todoItems");  
    var li = document.createElement("li");  
    var t = document.createTextNode();  
    t.data = row.text;  
  
    // On ajoute l'élément à la liste <ul> pointée  
    // par la variable todos  
    li.appendChild(t);  
    li.appendChild(a);  
    todos.appendChild(li)  
}
```


Suppression d'éléments dans un object store

```
html5rocks.indexedDB.deleteTodo = function(id) {  
    var db = html5rocks.indexedDB.db;  
    var trans = db.transaction(["todo"], IDBTransaction.READ_WRITE, 0);  
    var store = trans.objectStore("todo");  
  
    var request = store.delete(id);  
  
    request.onsuccess = function(e) {  
        html5rocks.indexedDB.getAllTodoItems(); // rafraichir affichage  
    };  
  
    request.onerror = function(e) {  
        console.log(e);  
    };  
};
```

WebSQL

- Une vraie BD relationnelle dans le navigateur (aujourd'hui SQLite),
- Supporté par Chrome, Safari, Firefox et Opera
- API peu adaptée à JavaScript
- Un peu sur-dimensionné d'utiliser SQL depuis du JavaScript où on manipule surtout du JSON
- Spécification plus ou moins « au point mort » par le W3C (attente d'une seconde implémentation pour valider)

WebSQL utilisation

■ Création d'une base

- `var db = openDatabase('mydb', '1.0', 'my first database', 2 * 1024 * 1024);`
- Nom, version, description, taille estimée de la BD

■ Création d'une table et insertion d'un élément

```
db.transaction(function (tx) {  
    tx.executeSql('CREATE TABLE foo (id  
                                     unique, text)');  
    tx.executeSql('INSERT INTO foo (id,  
                                     text) VALUES (1, "synergies")');  
});
```

WebSQL utilisation, suite

■ Récupération d'éléments, utilisation d'une fonction de callback

```
tx.executeSql('SELECT * FROM foo', [], function (tx,
                                                    results) {
    var len = results.rows.length, i;
    for (i = 0; i < len; i++) {
        alert(results.rows.item(i).text);
    }
});
```

■ Attention, results n'est pas un tableau JavaScript !

- Si on avait un âge comme colonne,
results.rows.item(i).age

File API

- Support par Opera, Chrome récent, Firefox
- Support pour l'upload / download de fichiers, y compris par drag'n'drop
- On peut créer et manipuler un FileSystem complet associé à une page web,
- Crypté, mode « sandbox » (le code JS ne pourra lire que dans le FileSystem créé par la page)
- Possibilité de gérer des répertoires, des fichiers, d'uploader des fichier dans le FileSystem (wget like)
- Les fichiers et répertoire ont des URIs

Exemple d'obtention d'informations sur des fichiers

```
<input type="file" accept="image/*" multiple onchange="filesProcess(this.files)"
name="selection"/>

Sélectionnez plusieurs images<br/>
<br/>
<div id="result">...</div>

<script>
  function filesProcess(files) {
    selection = "<table><tr><th>Nom</th><th></th><th>Octets</th><th></th><th>Type
MIME</th></tr>"

    for(i=0;i<files.length;i++){
      file = files[i]
      selection += "<tr><td>"+file.name+"</td><td> | </td><td style=\"text-align:right\">"
                    +file.size+"</td><td> | </td><td>"
                    +file.type+"</td></tr>"

      selection += "</table>"
      document.getElementById("result").innerHTML = selection
    }
  }
</script>
```

Copier/coller cet exemple sur un site de test de code JS et l'exécuter ! [Démonstration](#)

Analyse de répertoire entier

- Le navigateur fournit un objet directory auquel on a accès depuis l'API JavaScript (WebKit only pour le moment)
- Cliquer l'image pour exécuter

```
<input type="file" id="dir-select" webkitdirectory />
```

```
document.querySelector('#dir-select').onchange = function(e) {  
  var out = [];  
  for (var i = 0, f; f = e.target.files[i]; ++i) {  
    out.push(f.webkitRelativePath);  
  }  
  document.querySelector('output').textContent = out.join('/n');  
};
```

Select a directory

View as: [LIST](#)

copy-of-light-field-cameras-2bfoyp69lxf-033_131743_116230

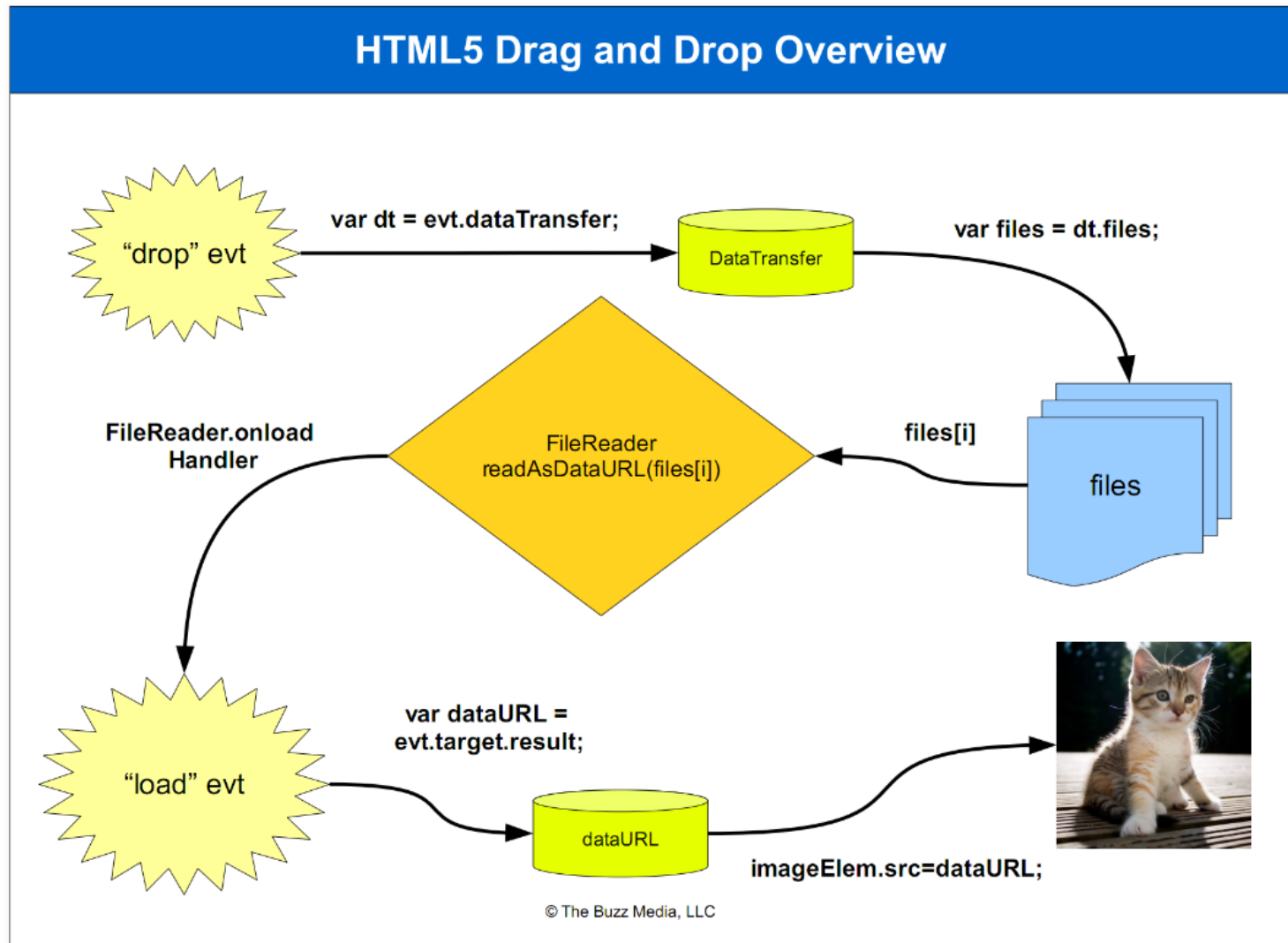
data

prezi.app

prezi.exe

Un exemple concret : drag'n'drop de fichiers multiples pour affichage/upload

- Schéma explicatif (source : <http://www.thebuzzmedia.com/html5-drag-and-drop-and-file-api-tutorial/>)

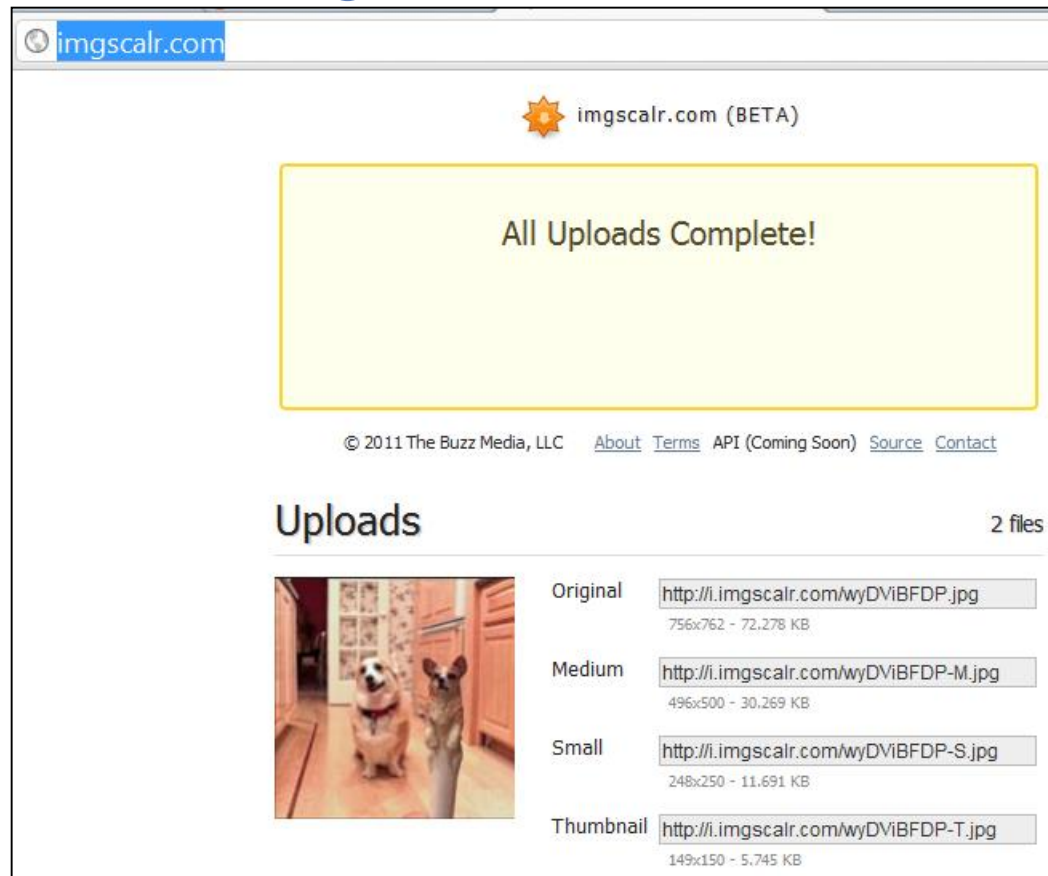


Explications lors du drag'n'drop

1. (DnD API) démarre avec un drop event quand l'utilisateur relâche la souris (mouseup)
2. (DnD API). On récupère un objet de type `DataTransfer` depuis le drop event
3. (File API) On appelle `DataTransfer.files` pour récupérer un objet `FileList`, qui représente la liste des fichiers droppés.
4. (File API) On itère sur les fichiers et on utilise un `FileReader` pour lire leur contenu.
5. (File API) On utilise `FileReader.readAsDataURL(file)`, chaque fois qu'on a lu un fichier, un nouvel objet "data URL" (RFC 2397) est créé et un événement `onload` est lancé par l'objet `FileReader`
 - L'objet de type data URL est le contenu en base 64.
 - Exemple : ``
 - `AAAFCAyAAACNbybIAAAAHIEQVQI12P4//8/w38GIAXDIBKE0DHxgljNBAAO" data-bbox="184 724 774 748"/>`
 - `9TXL0Y4OHwAAAABJRU5ErkJggg==" alt="Red dot">`
6. (File API) Maintenant on peut utiliser l'objet "data URL" avec un attribut `src` d'une `` pour afficher l'image dans la page, ou l'uploader sur un serveur.
7. Supporté par : Chrome, Firefox, Safari 6+ with Opera 11.x, IE 10

Lancement de l'exemple

- Démonstration
- Code téléchargeable et tutorial détaillé



Extraits du code

```
$(document).ready(function() {  
    var dropbox = document.getElementById("dropbox"  
    dropbox.addEventListener("drop", drop, false);  
});  
  
function drop(evt) {  
    var files = evt.dataTransfer.files;  
    var count = files.length;  
    // Only call the handler if 1 or more files was dropped.  
    if (count > 0)  
        handleFiles(files);  
}  
  
function handleFiles(files) {  
    var file = files[0];  
    var reader = new FileReader();  
    // init the reader event handlers  
    reader.onload = handleReaderLoadEnd;  
    // begin the read operation  
    reader.readAsDataURL(file);  
}
```

Extrait du code (suite)

```
function handleReaderLoadEnd(evt) {  
    var img = document.getElementById("preview");  
    img.src = evt.target.result;  
}
```

File API, pour aller plus loin

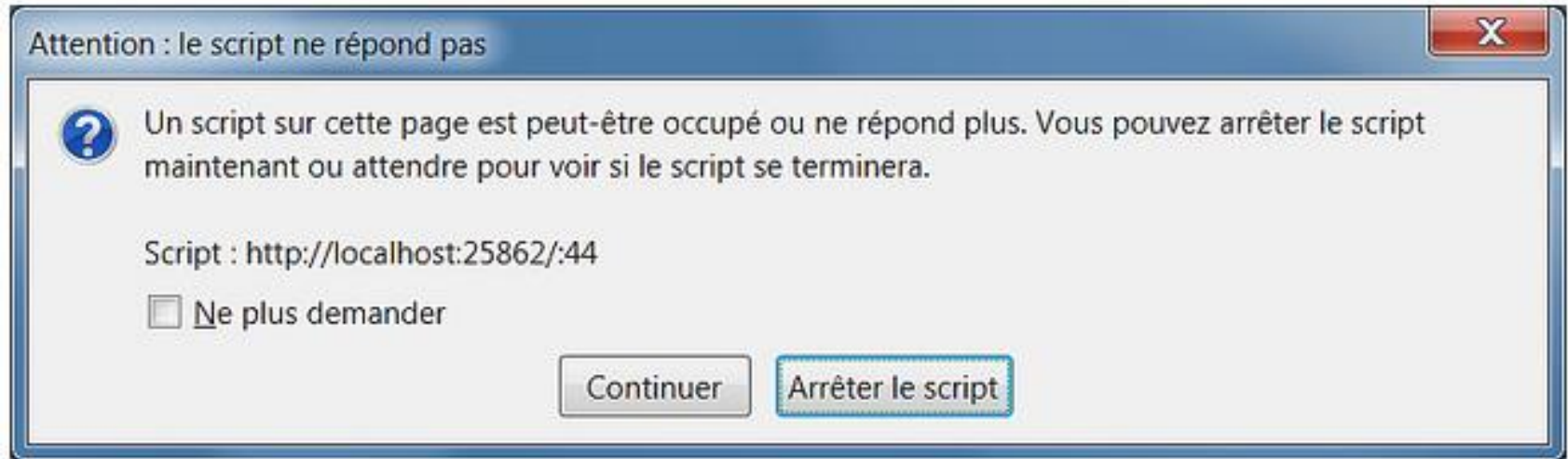
- Tutorial très complet sur :
<http://hacks.mozilla.org/2009/12/w3c-fileapi-in-firefox-3-6/>
- Voir des exemples assez incroyables sur
<http://www.htmlfivewow.com/slide19>

HTML5 WebWorkers

HTML5 WebWorkers

- Limitation de JavaScript : toute l'exécution s'exécute dans un seul et même thread !
 - Si à un moment donné on exécute une action longue -> tout est bloqué, y compris l'interface utilisateur (UI Thread)
 - D'où la programmation asynchrone par callbacks (appels HTTP Ajax, etc.) ou des animations à l'aide de setInterval(...)
 - mais... dans certains cas pouvoir exécuter une tâche en fond est intéressant et...
 - Avec les processeurs multi-cœurs, il est peu coûteux de pouvoir faire des actions en parallèle !

HTML5 WebWorkers : mon script dure trop longtemps !



- Problème : est-ce dû à une erreur ou bien est-ce que le traitement est voulu ?
- Même avec les appels asynchrones ou `setInterval()` tous les traitements sont effectués de manière séquentielle (on parle de file d'événements). Il n'y a aucun parallélisme.

HTML5 WebWorkers

- Les [APIs des Web Workers](#) définissent justement un moyen d'exécuter des scripts en tâche de fond (multithread)

- Création d'un WebWorker :

```
var w = new Worker('monworker.js');
```

- On crée un écouteur pour les sorties du Worker

```
w.addEventListener("message", function (event) {  
    // faire quelque chose, modifier le DOM par ex  
}, false);
```

- Démarrage du WebWorker

```
w.postMessage('hello');
```

Besoins pour des traitements en tâche de fond

- Traitement d'image dans des Canvas
- Traitement de données reçus après des appels Ajax
- Entrées-sorties multiples et concurrentes via les solutions de persistance de HTML5 (IndexedDB ou autre)
- Traitement du DOM lourd (analyse de texte, correcteur orthographique, etc)
- Jeu (IA, moteur physique, etc.), voir par exemple cet article sur [la bibliothèque de simulation physique Box2D en WebWorkers](#)

HTML5 WebWorker : code du worker

```
function messageHandler(event) {  
    // On récupère le message envoyé par la page principale  
    var messageSent = event.data;  
  
    // On prépare le message de retour  
    var messageReturned = "Bonjour " + messageSent + "  
                           depuis un thread séparé !";  
  
    // On renvoie le tout à la page principale  
    this.postMessage(messageReturned);  
}  
  
this.addEventListener('message', messageHandler, false);
```

- Exemple complet [sur cette page de blog](#)
- Notez qu'un message envoyé à un WebWorker peut être un objet JavaScript

HTML5 WebWorker : support JSON

```
function WorkerMessage(cmd, parameter) {  
    this.cmd = cmd;  
    this.parameter = parameter;  
}  
  
// On démarre le worker en lui envoyant la commande  
// 'init'  
monWorker.postMessage(new WorkerMessage('init', null));
```

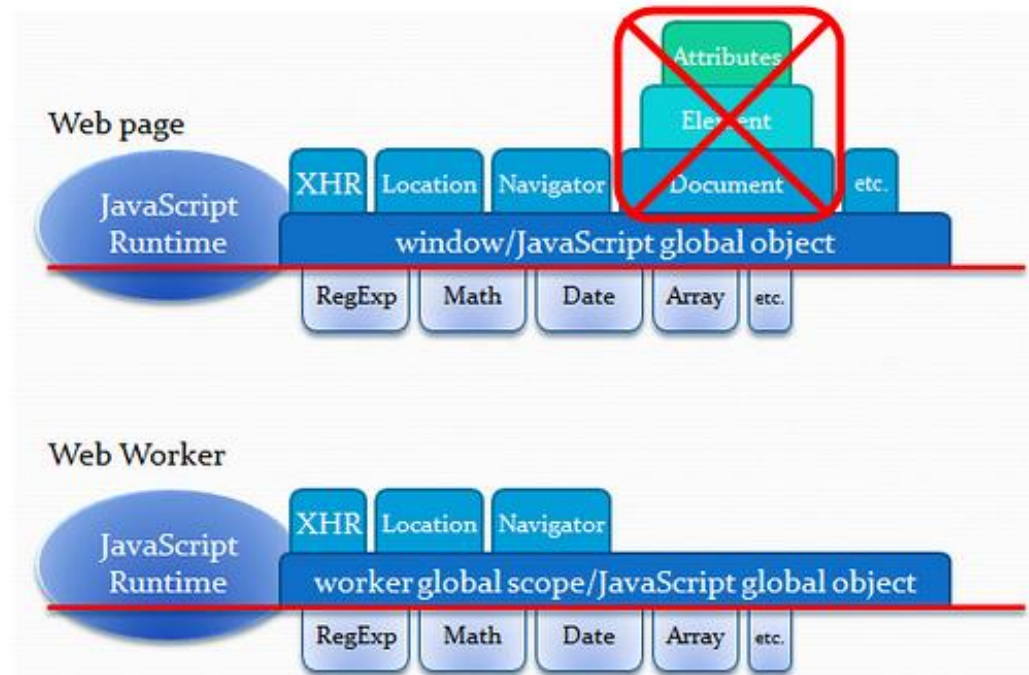
Et dans le worker :

```
function messageHandler(event) {  
    var msg= event.data;  
    switch (msg.cmd) {  
        case 'init':  
            // faire quelque chose...  
            break;  
        case 'hello':  
            var messageReturned = "Bonjour " +  
                msg.parameter + " depuis un thread séparé !";  
            this.postMessage(messageReturned);  
            break;  
    }  
}
```

Note : les messages
sont sérialisés et
désérialisés en JSON
automatiquement.

HTML5 WebWorkers : scopes

- Attention, on a pas accès à tous les objets standards dans le code JavaScript d'un Worker, notamment, **on a pas accès au DOM de la page !**



HTML5 WebWorkers : pas de console.log()

- On n'a pas accès à l'objet console !
- Pour que cela marche quand même : utiliser [console.log\(\) for WebWorkers](#) qui simule console.log() à l'aide de messages.

HTML5 WebWorker : debug

- On peut s'abonner à l'événement « error » :

```
monWorker.addEventListener("error", function (event) {  
    output.textContent = event.data;  
}, false);
```

- ... ou utiliser les consoles de debug comme Firebug, console intégrée à Chrome, Opera ou IE10