Introduction au développement Web

HTML, CSS, JavaScript et PHP

E. RAMAT

Université du Littoral - Côte d'Opale

20 avril 2015

Déroulement du module

- 4h de cours
- 42h de TP
- **chaque séance** de tp :
  - un compte rendu (votre code) à la fin de chaque partie
  - une note

- note du module = $\frac{1}{2}$ (examen + moyenne des notes TP)

Sommaire

Plan

Introduction

Objectifs

- comprendre le modèle du Web : client / serveur et le protocole HTTP
- aborder les langages de présentation (côté client) : HTML et CSS
- ajouter les aspects dynamiques côté client avec JavaScript
- introduire aux technologies serveur via PHP

Compléments abordés

Comment intégrer/utiliser un framework (côté client) ?

- Bootstrap : un framework responsive
- JQuery : un framework javascript
- canvas en HTML5 via un framework graphique 2D (d3.js)

Introduction

Mise en oeuvre

- développement d'une petite application d'e-commerce
  - première partie : la fiche produit (html + css)
  - deuxième partie : le panier (html + css + js)
  - troisième partie : les produits et la commande sur le serveur avec une base de données
- certains éléments seront donnés (le système d'identification, la base de données, ...)
- les outils :
  - l'environnement de développement : phpstorm de jetbrains
  - un navigateur et son debugger : chrome
  - un système de gestion de bases de données : sqlite

Définition : WWW

- WWW = World Wide Web = Web
- système d'information distribué à l'aide d'Internet (interconnexion de réseaux)
- un service (comme E-mail, DNS, ...)
- un ensemble de documents (et d'autres ressources : son, vidéo, ...) localisés sur différents serveurs
- accessible via divers protocoles standards : HTTP, HTTPS et FTP
- identifiés par une URL unique (`Unified Resource Location`)
- les serveurs Web fournissent du contenu Web
- les navigateurs Web affichent ce contenu

Structure

- Internet : fournisseur des canaux de communication à l'aide des protocoles TCP/IP, DNS et HTTP
- les clients (navigateurs Web)
- les serveurs Web : IIS de Microsoft, Apache, Tomcat, nodejs, ...

Composants

- un portocole : Hyper Text Transfer Protocol (HTTP)
- un langage : Hyper Text Markup Language (HTML)
- une localisation : Uniform Resource Locator (URL)
- des identifiants Uniform Resource Identifiers (URIs)

Définition

Une chaîne de caractères compose de :

- un protocole (HTTP, HTTPS, FTP, ...)
- un nom de serveur (ou d'une IP) + un port (optionnel) - par défaut, le port est 80 (ou 8080)
- un chemin accompagné d'un nom (`shop/index.html`, par exemple)

RFC

La syntaxe des URL sst définie par la RFC 1738.

Introduction

HTTP

Définition

- HTTP est un protocole client-serveur
- objectif : gérer le transfert de ressources (fichiers HTML, CSS, images, ...)
- prototype basé sur la notion de requête - réponse
- format basé texte
- fournit des métadonnées (encodage, par exemple)

Exemple

Requête

```
GET / HTTP/1.1
Host: www.google.fr
User-Agent: Mozilla/5.0 (Linux x86_64)
<CRLF>
```

Réponse

```
HTTP/1.1 200 OK
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Wed, 21 Jan 2015 17:52:55 GMT
Expires: -1
<CRLF>
<HTML>
...
</HTML>
```

Introduction

HTTP

client/serveur

- le programme client (le navigateur) envoie une requête au serveur (le serveur Web)
- le serveur répond en fournissant une ressource
- les requêtes sont typées par leur méthode : GET, POST, DELETE ou HEAD

Echange

Entête HTTP

L'entête d'une requête HTTP se compose :

- d'une ligne de requête avec le type de la méthode, l'URI et la version du protocole
- d'un ensemble de paramètres supplémentaires
- d'un corps (*body*) contenant les données (la valeur des champs d'un formulaire en mode POST, par exemple)

Format

```
<request method> <resource> HTTP/<version>
<headers>
<empty line>
<body>
```

Introduction

HTTP

Format

```
GET / HTTP/1.1
Host: www.google.fr
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:31.0) Gecko/20100101 Firefox/31.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr-fr
Accept-Encoding: gzip, deflate
Connection: keep-alive
<CRLF>
```

Explications

- la méthode d'accès est GET
- on tente d'accèder à la racine
- avec la version 1.1 du protocole HTTP
- le serveur est `www.google.fr`
- le navigateur est Iceweasel version 31.3.0 sous Linux Intel 64 bits
- le corps est vide

Entête HTTP

L'entête d'une réponse HTTP se compose :

- d'une ligne de requête avec la version du protocole, le code de retour et la phrase du code de retour
- des métadonnées
- le corps de la réponse (le code HTML de la page demandée)

Format

```
HTTP/<version> <status code> <status text>
<headers>
<CRLF>
<response body - the requested resource>
```

Introduction

HTTP

Format

```
HTTP/1.1 200 OK
Alternate-Protocol: 443:quic,p=0.02
Cache-Control: private, max-age=0
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Wed, 21 Jan 2015 17:52:55 GMT
Expires: -1
Server: gws
X-Frame-Options: SAMEORIGIN
X-XSS-Protection: 1; mode=block
X-Firefox-Spdy: 3.1
```

Explications

- la version du protocole HTTP de la réponse est 1.1
- le code de retour (`status code`) est 200 et la phrase est "OK"
- le contenu du corps est de type text/html

Méthodes

- GET : demande le téléchargement d'une ressource
- HEAD : récupère seulement l'entête d'une ressource
- POST : mets à jour une ressource ou fournit les données pour mettre à jour quelque chose
- DELETE : // TODO

REST

- // TODO un mot sur REST

Les réponses

- 1xx : information ("100 Continue")
- 2xx : succès ("200 OK")
- 3xx : redirection ("304 Not Modified")
- 4xx : erreur client ("404 Not Found")
- 5xx : erreur serveur ("503 Service Unavailable")

Les cookies

- les cookies sont des données stockées sur le client
- elles sont mises à jour à l'aide de la réponse du seveur
- si elle existe, elles sont envoyées au serveur lors des prochaines requêtes

Schéma

Cookie - réponse serveur

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache
Content-Type: text/html; charset=utf-8
Set-Cookie: connect.sid=s%3A3JwpEk2obi_K16k1Wi35w4sm7k2l6whL.YNKbVT3JgtssvtvrUy(
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Wed, 21 Jan 2015 18:13:10 GMT
Connection: keep-alive
Transfer-Encoding: chunked
```

Explications

- le serveur demande à enregistrer le cookie "connect.sid"
- c'est un identifiant de session
- il sera envoyé à la prochaine requête

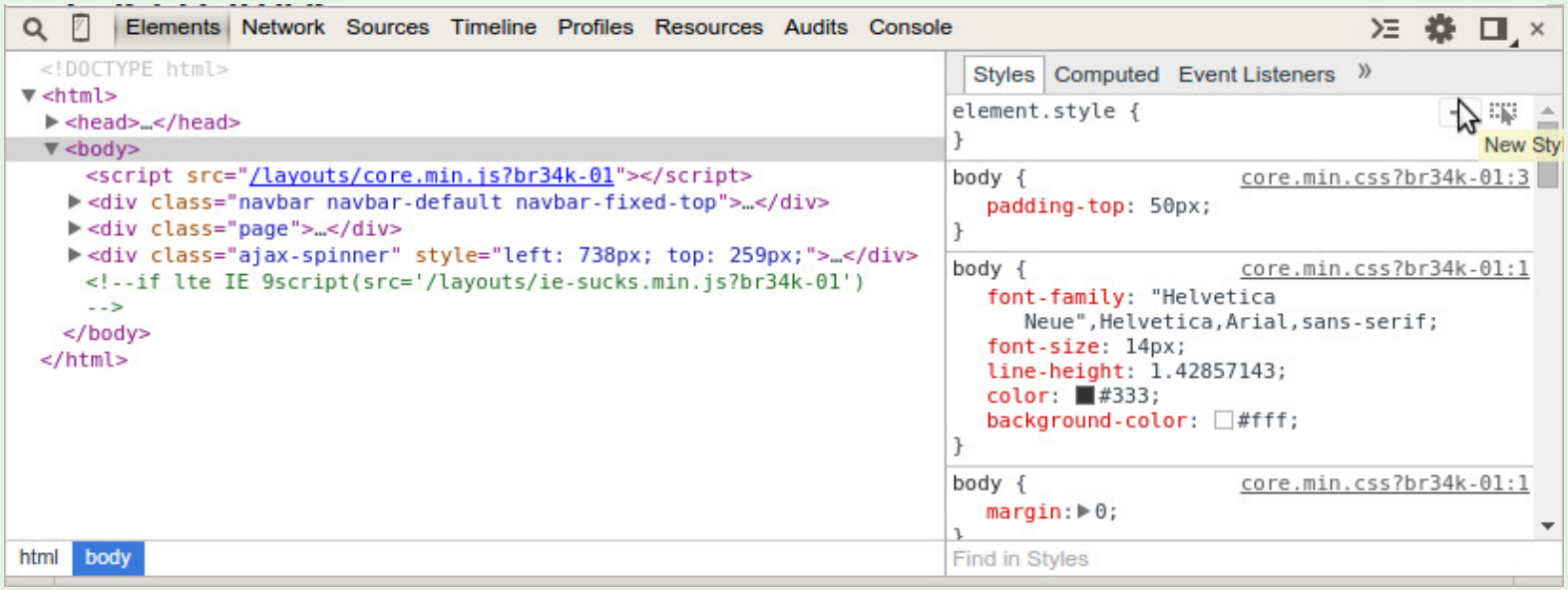
Le debugger

- sous Iceweasel/Mozilla, le plugin Firebug
- sous Chrome, c'est intégré

Possibilités

- navigation dans le DOM et modifier des éléments
- tracer les transferts réseaux (méthode, type de ressource, taille, temps, . . .)
- visualiser les sources javascript et les fichiers css téléchargés
- exécuter pas à pas des scripts JS, visualiser et modifier des variables JS
- visualiser et modifier les cookies, les bases de données locales (`localstorage` en HTML5)
- voir le contenu de la console (sortie des scripts JS)

Sous Chrome

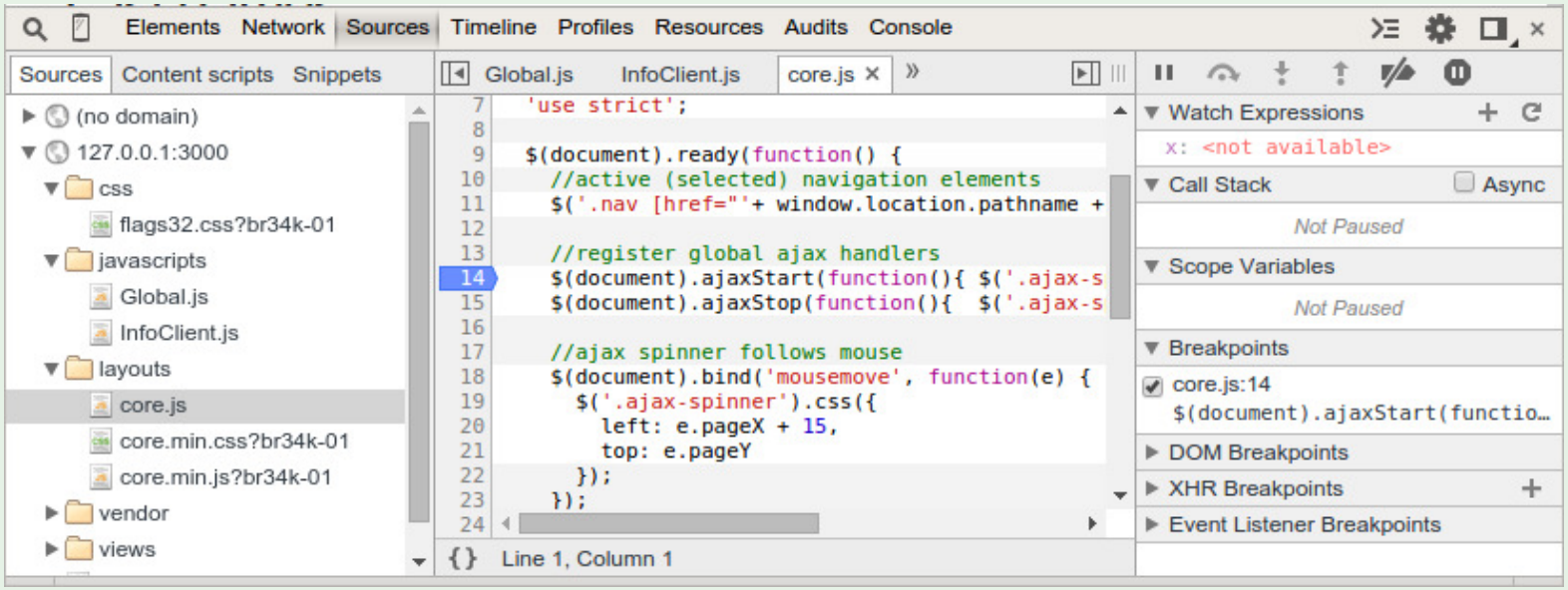


- la vue du DOM
- les éléments de CSS qui s'appliquent sur le document
- on peut aussi cliquer sur un élément d'une page pour obtenir sa représentation dans le DOM et les styles appliqués

Introduction

Les outils

Sous Chrome



- arborescence des fichiers téléchargés (.js et .css)
- source d'un fichier javascript
- définition d'un point d'arrêt
- suivi de variables

Introduction

Références

HTML

- *http ://www.w3schools.com/tags/*
- *https ://developer.mozilla.org/fr/docs/Web/HTML/Element*
- *http ://www.w3.org/TR/html-markup/* et *http ://www.w3.org/TR/html5/*
- *HTML5 Pocket Reference*, Jennifer Niederst Robbins, O'reilly
- *Head First HTML and CSS*, 2nd Edition, Elisabeth Robson, Eric Freeman

CSS

- *http ://www.w3schools.com/css/default.asp*
- *https ://developer.mozilla.org/fr/docs/Web/CSS/Reference*
- *http ://www.w3.org/TR/CSS2/*
- *http ://meiert.com/en/indices/css-properties/*

Introduction

Références

Javascript

- *http ://www.w3schools.com/js/default.asp*
- *https ://developer.mozilla.org/fr/docs/Web/JavaScript/Reference*
- *JavaScript : The Definitive Guide*, 6th Edition, David Flanagan, O'reilly

L'indispensable !

http ://stackoverflow.com/

L'indispensable !

Horizontally center a div in a div

1464

511

How do I horizontally center a `div` in a `div` with CSS (if it's possible at all)?

The outer `div` has 100%:

```
<div id="outer" style="width:100%">
    <div id="inner">Foo foo</div>
</div>
```

html css

share improve this question

edited Oct 31 '14 at 12:52

community wiki
9 revs, 7 users 55%
Lukas

4 like this jsfiddle.net/techsin/TfLTR – Muhammad Umer Aug 2 '13 at 5:02

show 5 more comments

41 Answers

active oldest votes

1 2 next

1675

You can apply this CSS to the inner div:

```
#inner {
    width: 50%;
    margin: 0 auto;
}
```

Of course, you don't have to set the width to 50%. Any width less than the containing div will work. The `margin: 0 auto` is what does the actual centering.

If you are targeting IE8+, it might be better to have this instead:

```
#inner {
    display: table;
    margin: 0 auto;
}
```

It will make the inner element center horizontally and it works without setting a specific width.

share improve this answer

edited Oct 29 '13 at 17:21

community wiki

Online editors

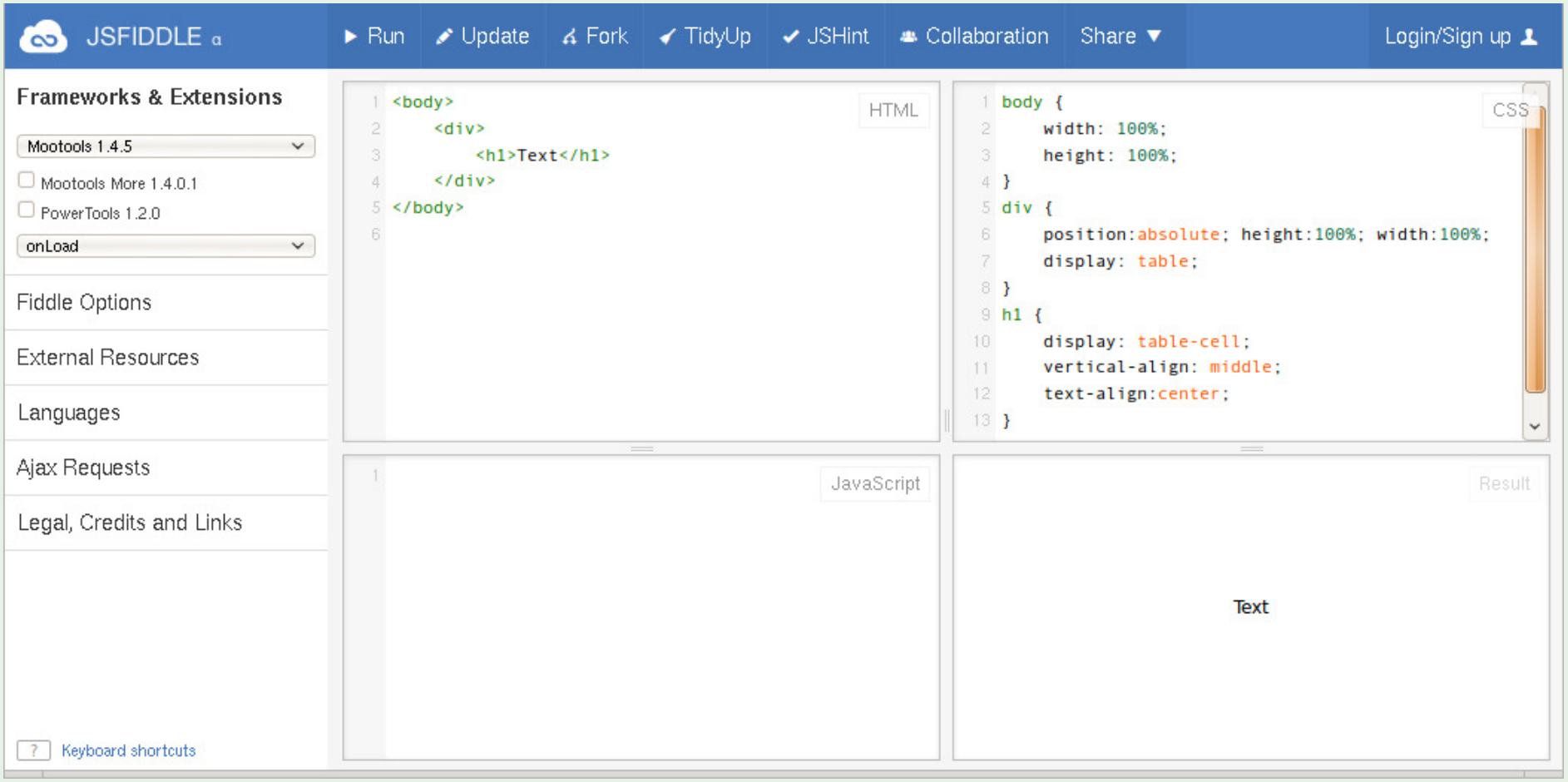
Les éditeurs de code online permettent de tester des morceaux de code HTML/CSS/Javascript avec ou sans bibliothèque

- *http ://jsfiddle.net/*
- *http ://js.do/*
- *http ://jsbin.com/ ?html,css,js,console,output/* (mode collaboratif)
- *http ://selfcss.org/* pour le CSS

Introduction

Online editors

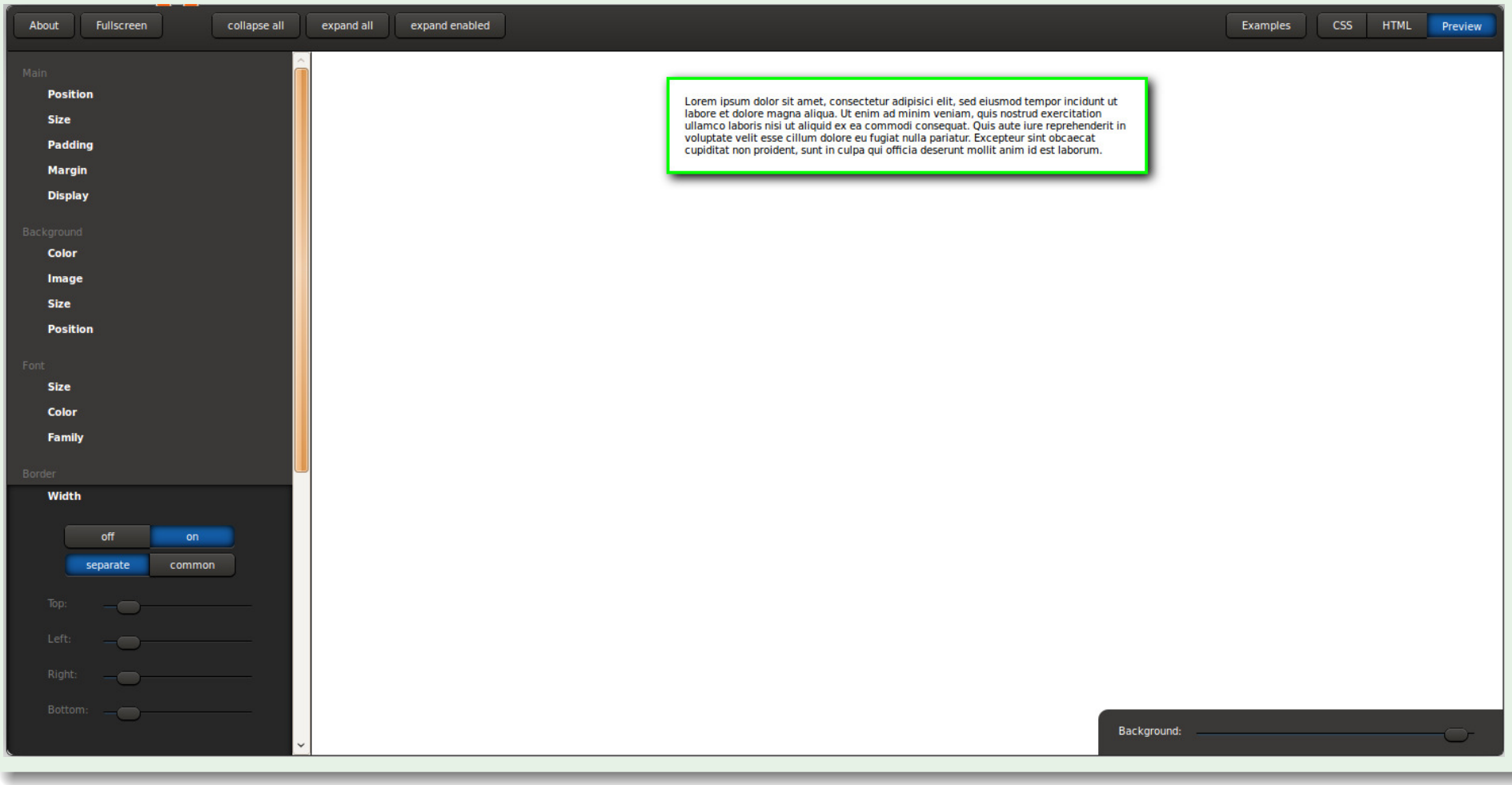
JSFiddle



Introduction

Online editors

selfCSS



Plan

Introduction

Les éléments abordés

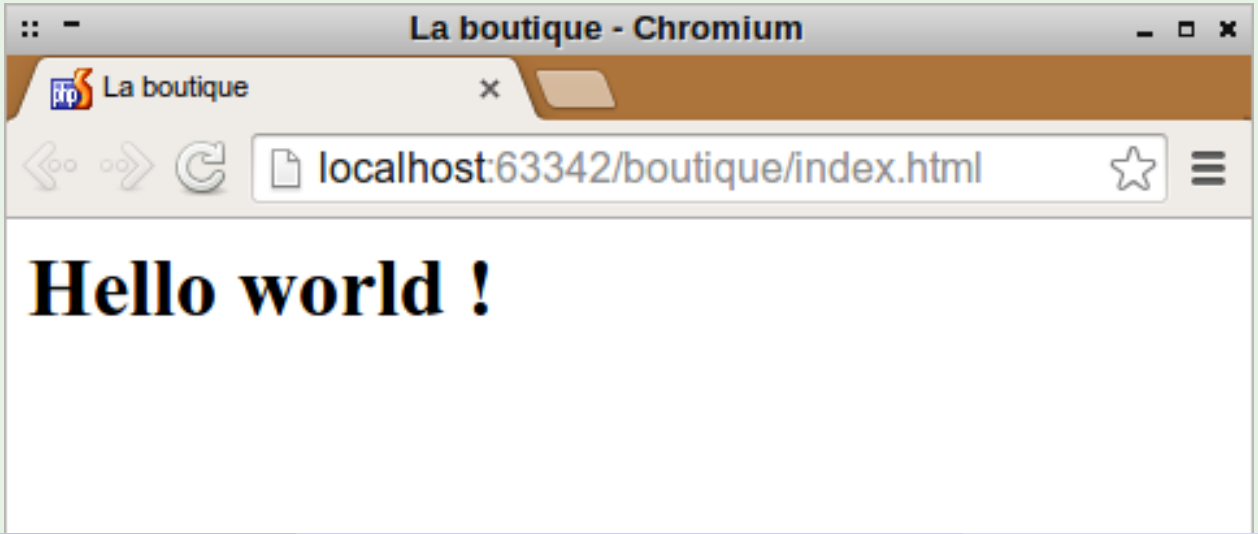
- la structure d'une page
- le texte et la mise en forme
- les listes
- les liens hypertextes
- les images
- les tableaux
- les div
- les formulaires
- les nouveautés HTML5

Introduction

Une page

```
<html>
<head lang="fr">
    <meta charset="UTF-8">
    <title>La boutique</title>
</head>
<body>
<h1>Hello world !</h1>
</body>
</html>
```

Sous Chrome



Quelques dates

- 1991 : première utilisation du terme HTML par Tim Berners-Lee
- 1993 : HTML première version publiée par IETF
- 1995 : HTML 2 normalisé par le W3C
- 1997 : HTML 3.2
- 1999 : sortie d'HTML 4.01
- 2001 : XHTML
- 2011 : HTML 5

Structure d'une page

Structure

- un document HTML est en réalité un document XML avec des balises prédéfinies

```
<!DOCTYPE html>
```

- la balise `html` marque le début du document

```
<html>
```

- la balise `head` contient des informations relatives à la page : le type d'encodage du texte, le titre, ...

```
<head lang="fr">
  <meta charset="UTF-8">
  <title>La boutique</title>
</head>
```

Structure d'une page

Structure

- la balise `body` contient le corps de la page (les éléments visuels de la page ; dans l'exemple, un texte mis en style h1)

```
<body>
  <h1>Hello world !</h1>
```

- le document se termine par la fermeture des balises `body` et `html`

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

Les balises

Structure d'une balise

```
<div style="text-color: red">xxxxx</div>
```

- de manière générale, une balise est divisée en deux parties :
  - la balise ouvrante (`<div>`)
  - la balise fermante (`</div>`)
- du texte ou d'autres balises peuvent être placés entre les deux parties
- un document HTML peut être représenté sous forme un arbre
- une balise est complètée par un ensemble d'attributs
- les attributs sont facultatifs
- une balise peut être vide

```
<br />
```

Balises pour le textes

Les paragraphes

- <p>
-
- <hr>
- <blockquote> et <q>

Mise en forme

- gras, italique, strong, em, ...
- indice, exposant, ...
- <cite>, <dfn>
- <pre>

Les styles

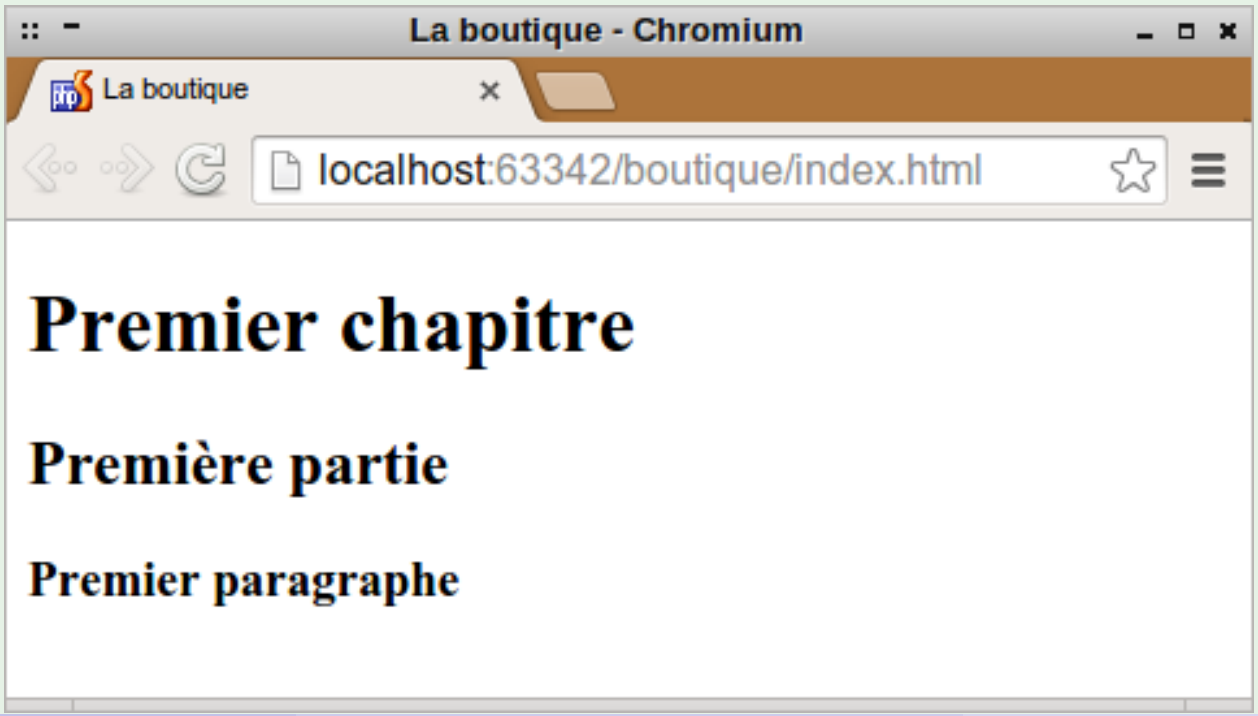
Les styles vont permettre aussi d'appliquer des mises en forme.

Balises pour le textes

Les titres

- les titres sont définis par la balise `<hx>`
- où x désigne un niveau
- il existe 6 niveaux : 1 étant associé à la taille de caractères la plus grande et 6 la plus petite

Exemple



Balises pour le textes

Les caractères spéciaux

- le standard HTML n'autorise que l'encodage ASCII 7 bits
- les caractères accentués ne sont pas autorisés
- il faut utiliser soit des constantes prédéfinies soit un code hexadécimal
- le format est : &xxxx ; (par exemple pour l'espace :   ; ou &#32 ;)

Exemple

Caractère	Code	Caractère	Code
&	&amp;	¶	&para;
€	&euro;	À	&Agrave;
<	&lt;	Ç	&Ccedil;
>	&gt;	È	&Egrave;
Espace		É	&Eacute;
§	&sect;	Ê	&Ecirc;
©	&copy;	À	&agrave;
®	&reg;	é	&eacute;
±	&plusmn;	è	&egrave;

Les listes

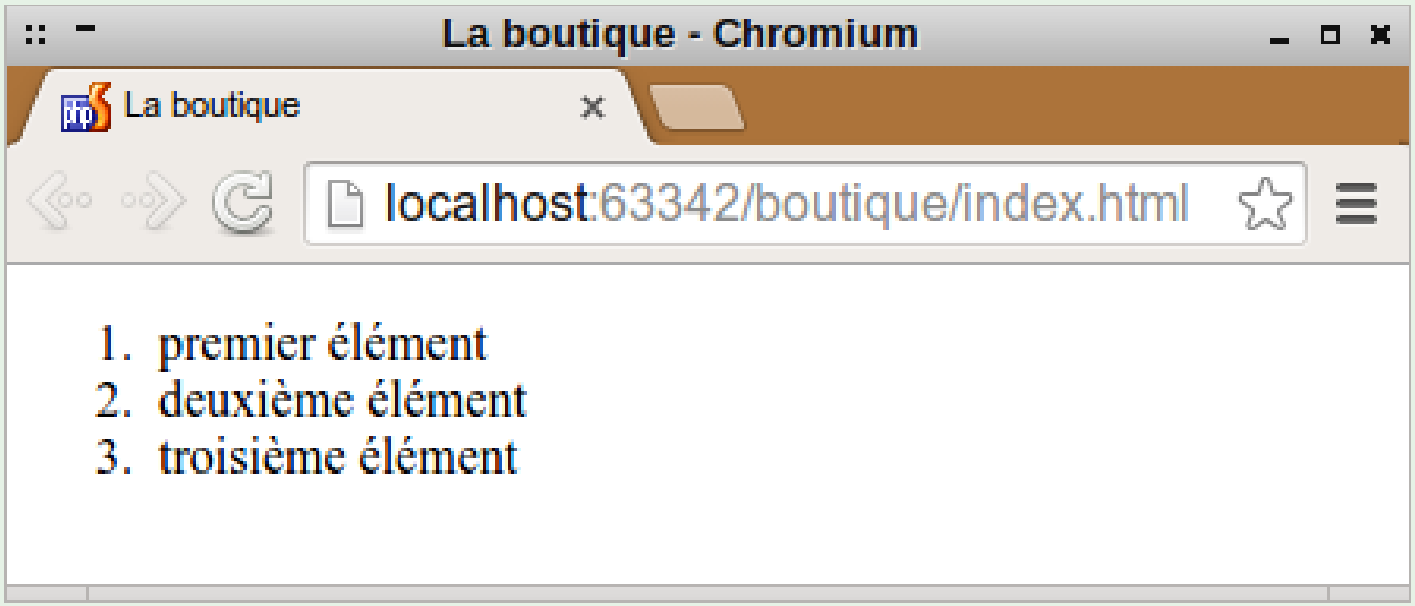
Définition

- il existe plusieurs types de listes : ordonnées ou non ou de définitions
- les listes ordonnées contiennent une séquence d'éléments dont la position est importante
- les listes non ordonnées contiennent des éléments dont l'ordre n'a pas d'importance
- une liste de définition contient des termes et leurs définitions
- une liste peut contenir d'autres listes

Les listes

Exemple 1

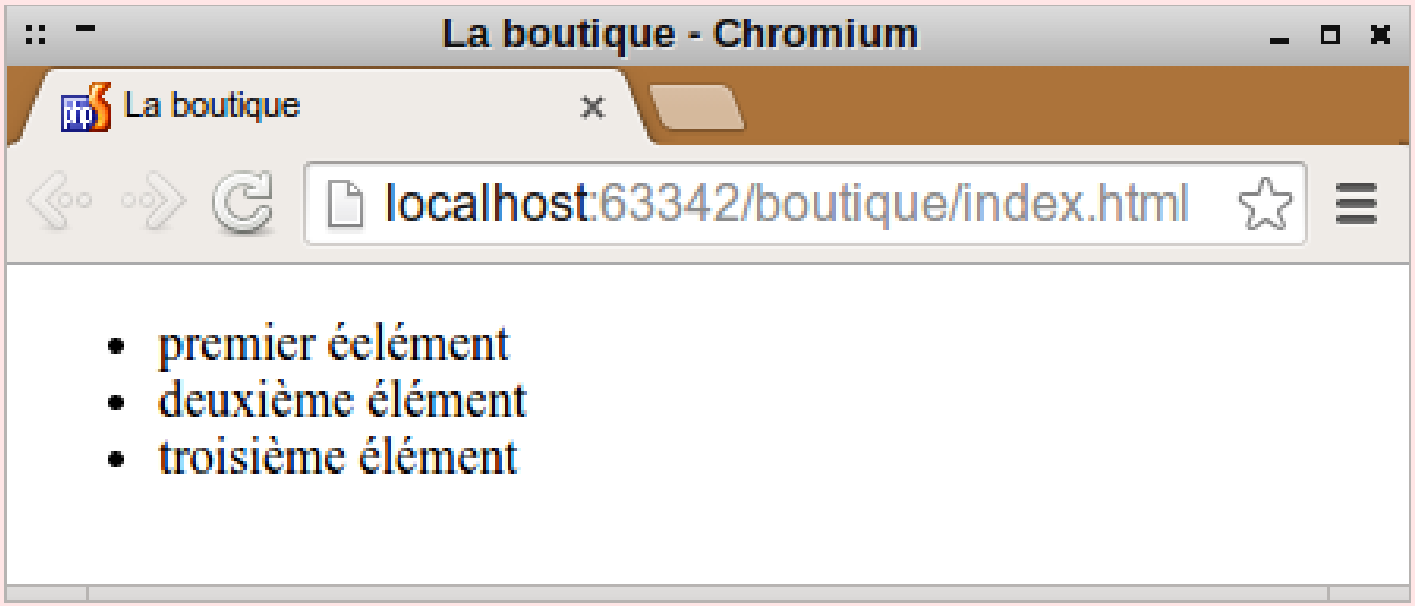
```
<ol>
  <li>premier &eacute;l&eacute;ment</li>
  <li>deuxi&egrave;me &eacute;l&eacute;ment</li>
  <li>troisi&egrave;me &eacute;l&eacute;ment</li>
<ol>
```



Les listes

Exemple 2

```
<ul>
  <li>premier &eacute;l&eacute;ment</li>
  <li>deuxi&egrave;me &eacute;l&eacute;ment</li>
  <li>troisi&egrave;me &eacute;l&eacute;ment</li>
<ul>
```



Les liens

Définition

- la notion de lien est au coeur du Web : référencer au sein d'une page d'autres pages ou ressources
- la balise utilisée est <a>
- l'adresse cible (ou URL) est précisée par l'attribut `href`
- un lien est représenté par l'élément placé entre <a> et
- cet élément peut être un texte mais aussi une image

Exemple

```
<a href="http://www.univ-littoral.fr">Universit&eacute; du Littoral</a>
```

Les liens

URL

- toute ressource du Web possède un identifiant unique : l'URL (`Unified Resource Locator`)
- il existe 2 types d'URL : absolue et relative
- une URL absolue commence par le nom de domaine du serveur contenant la ressource
- pour désigner une ressource appartenant au même serveur que la ressource qui la référence : utilisation des URL relatives
- on utilise alors la même strcuture que le système de fichiers
- on ne mentionne pas alors le nom de domaine
- on peut aussi faire référence à une partie d'une page au sein de la même page

Les liens

Exemples

Un lien :

- sur une page du même répertoire

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

- sur une page contenue dans un sous-répertoire

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

- sur une page contenue dans le répertoire parent

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

Liens via un module de routage

Attention, la plupart des frameworks de développement Web réécrivent les URLs
$\rightarrow$ l'URL relative n'a donc pas de lien avec les chemins du système

Les liens

Liens pour les mails

- une URL débutant par `mailto` demande au navigateur de lancer un client mail
- l'adresse du destinataire est précisée après `mailto`

Exemples

```
<a href="mailto:ramat@lisic.univ-littoral.fr">Eric RAMAT</a>
```

Cible

- par défaut, le clic sur un lien provoque le remplacement de la page précédente
- l'attribut `target` permet de préciser où sera affiché la nouvelle page :
  - `target="_blank"` : une nouvelle fenêtre s'ouvre
  - ...

Les liens

Liens vers une partie de la page

- un lien peut pointer sur un élément particulier dans une page ou d'une autre page
- cliquer sur ce type de lien provoque sa visualisation
- très pratique pour des pages très longues
- syntaxe : `<a href="#id1">xxx</a>` ou `<a href="autre_page.html#id1">xxx</a>` ou `<a href="www.univ-littoral.fr#id1">xxx</a>`
- la cible doit être identifiée par une balise contenant l'attribut id

Les images

Organisation

- un site utilise beaucoup d'images de tout type et de toute taille
- il est important de structurer leur stockage : dans un répertoire `img`, par ex
- les formats reconnus sont : jpeg, png, gif et svg

Critères de choix du format

- jpeg : idéal pour les photos et illustrations complexes contenant des millions de couleurs ; attention à la compression !
- png : format sans compression ; 8 bits ou 24 bits ; possibilité de transparence
- gif : pratique pour des petites images (logo, par exemple) ; 256 couleurs max ; gestion de la transparence et de l'animation
- svg : format vectoriel ; ensemble d'objets graphiques ; pas de perte de qualité ; plus léger et plus rapide

Outils libres

gimp pour les images pixelisées et **inkscape** pour les images vectorisées

Les images

La balise img

- l'insertion d'une image dans une page est possible à l'aide de la balise `img`
- deux attributs :
  - src : URL de l'image (absolue ou relative)
  - alt : un texte alternatif en cas de non chargement de l'image

Exemple

```
<img src="img/mon_image.jpg" alt="Mon image" />
```

Les images

Poids des images

- faire attention au “poids” d'une image → énorme impact sur le temps de chargement d'une page
- jouer sur la résolution (72dpi, par exemple), la taille et le taux de compression (pour le format jpeg)

Taille des images

- par défaut, une image s'affiche avec sa taille (largeur / hauteur)
- la largeur et la hauteur peuvent être modifiées via les attributs `width` et `height`
- attention aux proportions !

Exemple

```
<img src="img/mon_image.jpg" alt="Mon image" width="100" height="100" />
```

Les images

Placement des images

- en HTML4 et inférieur, la balise `align` permet de spécifier la position relative d'une image vis à vis du texte
- cette balise est maintenant obsolète
- il faut utiliser les CSS

Les images

SVG

- il est possible d'insérer directement du svg dans une page
- utilisation de la balise <svg></svg>

Exemple

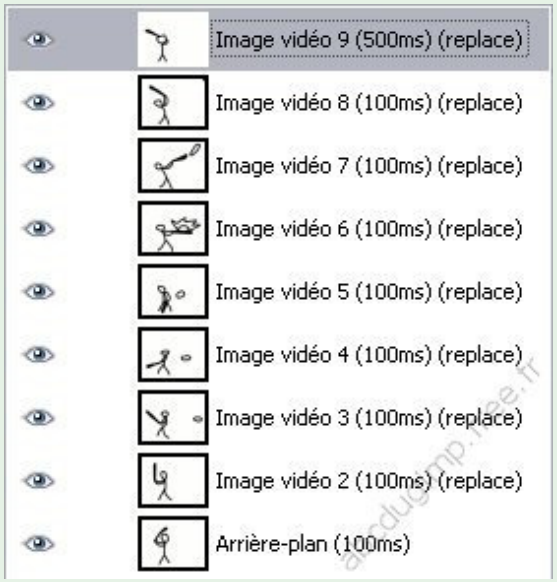
```
<svg width="400" height="180">
  <rect x="50" y="20" width="150" height="150"
        style="fill:blue;stroke:pink;stroke-width:5;fill-opacity:0.1;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣stroke-opacity:0.9">
</svg>
```

Les images

gif animé

- une solution pour introduire des petites animations : les gif animés
- un gif animé est une collection d'images gif affichées avec un délai entre chaque image
- la construction d'un gif animé est possible avec gimp à l'aide de la notion de calque : une image est un calque et on remplace un calque par un autre avec un délai

Exemple



Les tableaux

Définition

- les tableaux permettent d'organiser les éléments d'une page sous forme tabulaire
- un tableau est un ensemble de lignes
- une ligne est un ensemble de cellules

Exemple

Number	First Name	Last Name	Points
1	Eve	Jackson	94
2	John	Doe	80
3	Adam	Johnson	67
4	Jill	Smith	50

Les tableaux

Les balises

- la balise <table> pour délimiter le tableau
- <tr> pour définir une ligne
- <td> pour une cellule
- le tableau s'adapte aux éléments placés dans les cellules
- par défaut, aucun élément graphique indique les séparations entre les cellules

Exemple

```
<table style="width:100%">
  <tr>
    <td>Jill</td><td>Smith</td><td>50</td>
  </tr>
  <tr>
    <td>Eve</td><td>Jackson</td><td>94</td>
  </tr>
</table>
```

Les tableaux

Entête, corps et pied

- il est possible de "typer" les lignes :
  - `<theader>` pour l'entête
  - `<tbody>` pour le corps
  - `<tfoot>` pour le pied
- de plus, les cellules peuvent être typées "entête" avec la balise `<th>` et l'attribut `scope`

Exemple

```
<table>
<thead>
 <tr><th>Month</th><th>Savings</th></tr>
</thead>
<tfoot>
 <tr><td>Sum</td><td>$180</td></tr>
</tfoot>
<tbody>
 <tr><td>January</td><td>$100</td></tr>
 <tr><td>February</td><td>$80</td></tr>
</tbody>
</table>
```

Les tableaux

Fusion

- afin d'améliorer la présentation, on peut fusionner des colonnes ou des lignes
- deux attributs :
  - `colspan` : n colonnes d'une ligne donnée sont fusionnées
  - `rowspan` : n lignes d'une colonne donnée sont fusionnées

Les tableaux

Exemple

```
<table>
  <tr>
    <th>Month</th><th>Savings</th><th>Savings for holiday!</th>
  </tr>
  <tr>
    <td>January</td><td>$100</td><td rowspan="2">$50</td>
  </tr>
  <tr>
    <td>February</td><td>$80</td>
  </tr>
</table>
```

Month	Savings	Savings for holiday!
January	$100	$50
February	$80	

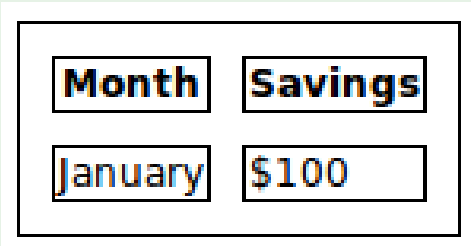
Les tableaux

Espacement et remplissage

- les attributs `cellspacing` et `cellpadding` définissent l'espacement entre les cellules et le remplissage d'une cellule
- attention, non supporté par HTML5 et donc utiliser les CSS

Exemple

```
<table cellspacing="10">
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
</table>
```



Les blocs

Définition

- les blocs ont pour objectif de structurer un contenu
- deux balises :
  - `span` : définit un bloc dans le flux du code HTML
  - `div` : définit un bloc rectangulaire

Exemple

```
<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes
and my father has <span style="color:darkolivegreen;font-weight:bold">
dark green</span> eyes.</p>
```

Rendu

My mother has **blue** eyes and my father has **dark green** eyes.

L'attribut `style` permet de définir les propriétés des éléments définis dans le bloc (`span`). C'est du CSS inline.

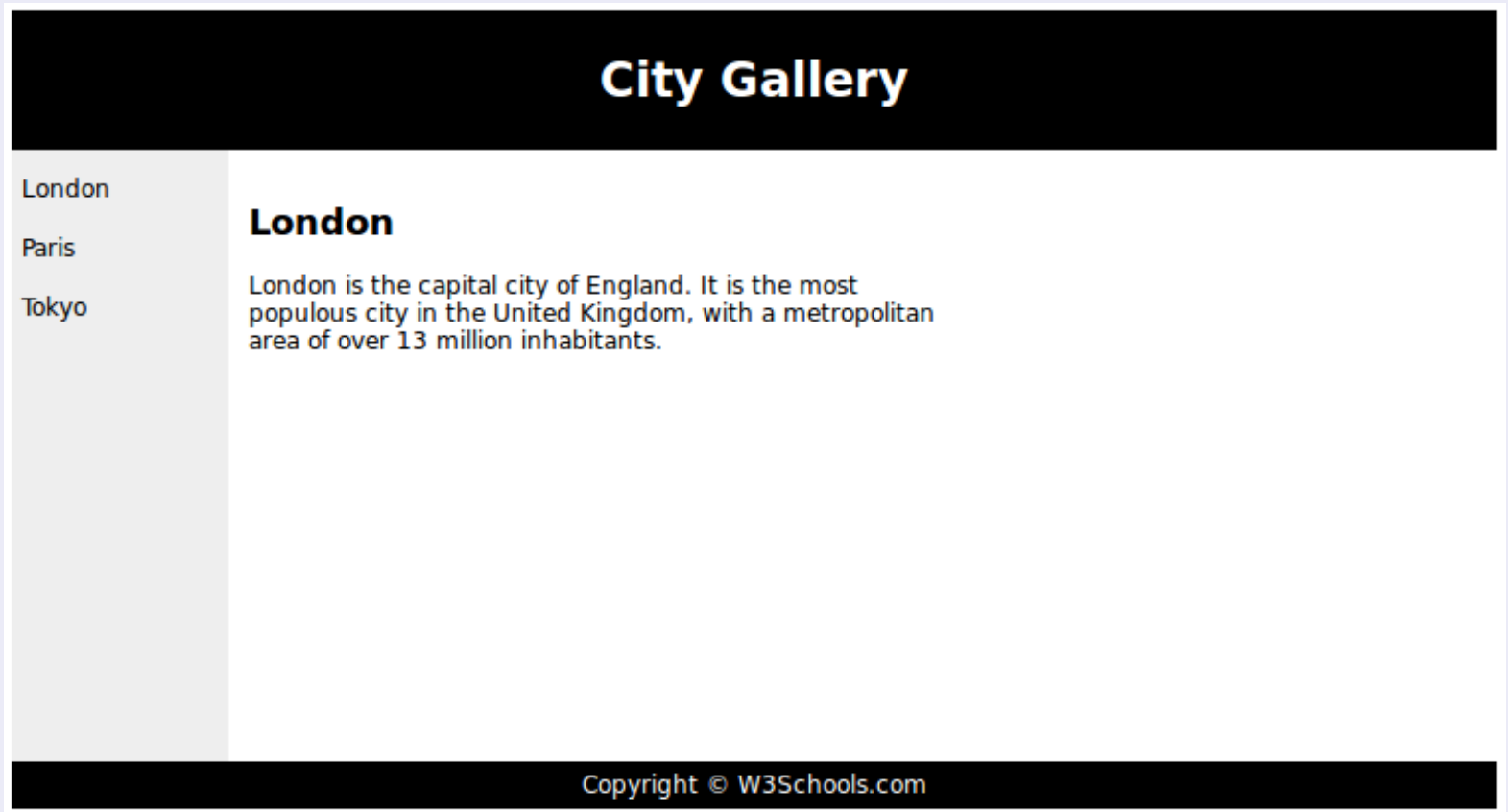
Les blocs

Exemple

```
<div style="background-color:black;color:white;text-align:center;padding:5px;">
  <h1>City Gallery</h1>
</div>

<div style="line-height:30px;background-color:#eeeeee;height:300px;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣width:100px;float:left;padding:5px;">
  London<br>
  Paris<br>
  Tokyo<br>
</div>

<div style="width:350px;float:left;padding:10px;">
  <h2>London</h2>
  <p>London is the capital city of England. It is the most populous city
  in the United Kingdom, with a metropolitan area of over 13 million
  inhabitants.</p>
</div>

<div style="background-color:black;color:white;clear:both;text-align:center;
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣padding:5px;">
  Copyright (c) W3Schools.com
</div>
```

Les blocs

Rendu



L'attribut `style` permet de définir les propriétés des éléments définis dans le bloc (`div`). C'est du CSS inline.

Les blocs

Dimensions

Les dimensions d'un bloc div sont définies :

- soit par son contenu
- soit par les propriétés `height` et `width` du style

Plan

Introduction

Les éléments abordés

- définition
- *cascading*
- les selecteurs
- le lien avec HTML
- les couleurs
- le texte et les polices de caractères
- les blocs
- la disposition
- les images
- les transitions et les transformations
- les animations

Editeur online

Utiliser selfcss pour définir vos CSS.

Introduction

Définition

- première version en 1996, v2 en 1998, v2.1 en 2004 et finalement v3 en 2012 (en partie) auprès du w3c (*http ://www.w3.org*)
- introduit en HTML4
- un langage de règles pour appliquer des styles aux éléments d'un document HTML
- un style se compose de multiple choses : une couleur, une police de caractères pour du texte, une position, ...
- les définitions peuvent être interne au code HTML ou externe en utilisant des fichiers CSS

Pourquoi ?

- objectif : séparer le contenu de la présentation
- le code HTML définit un contenu (texte, image, ...)
- le CSS définit les styles de présentation (couleur et taille du texte d'un paragraphe, par exemple)

Introduction

Fonctionnement

- chaque élément est dans une zone rectangulaire
- deux modes :
  - en bloc (<p>, <h1>, , ...)
  - *inline* ()
- le CSS va s'appliquer sur ces zones

Une zone peut contenir des zones → le CSS va donc s'appliquer par héritage (le *cascading*)

Introduction

Cascading

Comment déterminer le style qui doit s'appliquer sur un élément ?

- chaque élément d'un document HTML appartient à un arbre (DOM)
- il hérite des styles de ses parents
- il peut les surcharger (les redéfinir)

Ordre d'application

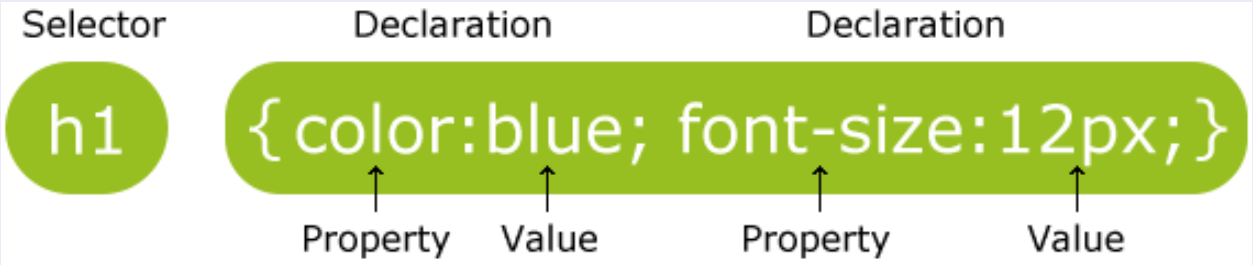
- la règle d'héritage s'applique aussi sur le mode de définition
- du moins prioritaire au plus prioritaire :
  - les styles par défaut du navigateur
  - les styles définis à l'extérieur du document
  - les styles définis dans l'entête du document
  - les styles *inline* (dans un élément HTML via l'attribut `style`)

Toutes les propriétés ne sont pas héritées :

- les propriétés relatives aux *box*
- les styles de positionnement

Syntaxe

Syntaxe



Explications

La définition d'un style se décompose en :

- un sélecteur qui désigne sur quel élément ou sur quel type d'éléments le style va s'appliquer
- un bloc de déclaration de style, lui-même composé d'un ensemble de couples composés de :
  - une propriété
  - une valeur

Syntaxe

Exemple

```
<html>
<head>
<style>
p {
    color: red;
    text-align: center;
}
</style>
</head>
<body>
  <p>Hello World!</p>
  <p>This paragraph is styled with CSS.</p>
</body>
</html>
```

Explications

- un nouveau style est défini pour la balise p
- la couleur rouge sera utilisée pour le texte
- le texte sera centré dans la page

Sélecteur

Définition

Un sélecteur permet de désigner un élément du document ou un ensemble d'éléments :

- tous les éléments d'une balise spécifique
- tous les éléments d'une classe donnée
- l'élément avec un id spécifique

Classe et id

Classe

Tout élément HTML peut être rattachée à une classe. Par exemple, le paragraphe a pour classe (`class`) c1.

```
<p class="c1">Hello World!</p>
```

Une classe peut être utilisée plusieurs fois au sein d'une même page.

Identifiant

Tout élément HTML peut possèder un id. Par exemple, le paragraphe a pour identifiant (id) para1.

```
<p id="para1">Hello World!</p>
```

Il doit être unique dans une page.

Sélecteur

CSS pour une balise

```
p {
    text-align: center;
    color: red;
}
```

CSS pour une classe

```
.c1 {
    text-align: center;
    color: red;
}
```

CSS pour un id

```
#para1 {
    text-align: center;
    color: red;
}
```

Sélecteur

Un CSS pour la classe d'une balise

- une classe peut être utilisé au sein de plusieurs balises
- on peut définir un CSS spécifique pour un couple donné classe + balise

```
p.c1 {
     text-align: center;
     color: red;
}
```

Un style pour plusieurs sélecteurs

- un même style peut être appliqué à plusieurs sélecteurs
- on définit alors un groupe de sélecteurs

```
p, .c1, #para1 {
     text-align: center;
     color: red;
}
```

Sélecteur

Sélection des sous-éléments

- sélectionne tous les éléments fils d'un élément (ou d'un type d'éléments)
- attention, ce sont les fils directs

Exemple

- sélectionne tous les éléments dont la balise parent est .
- affecte la couleur jaune au fond <p>

```
div * {
    background-color: yellow;
}
```

Sélecteur

Sélection des sous-éléments

- sélectionne tous les éléments fils d'un certain type de balise (ou d'une certaine classe, . . .) d'un élément (ou d'un type d'éléments)
- attention, ce sont les fils directs

Exemple

- sélectionne tous les <p> dont la balise parent est .
- affecte la couleur jaune au fond du texte contenu dans la balise <p>

```
div > p {
    background-color: yellow;
}
```

Sélecteur

Sélection des éléments “frères”

- sélectionne tous les éléments situés **juste** après un élément (ou un type d'éléments)
- les éléments sont au même niveau dans le DOM et possèdent un parent commun

Exemple

- sélectionne tous les <p> dont la balise précédente est .
- affecte la couleur jaune au fond du texte contenu dans la balise <p>

```
div + p {
    background-color: yellow;
}
```

Sélecteur

Sélection des voisins

- sélectionne tous les éléments situés après d'un élément (ou un type d'éléments)
- les éléments sont au même niveau dans le DOM et possèdent un parent commun

Exemple

- sélectionne tous les <ul> dont une balise <p> les précède.
- affecte la couleur jaune au fond du texte contenu dans la balise <p>

```
p ~ ul {
    background-color: yellow;
}
```

Sélecteur

Sélection sur les valeurs des attrributs

- sélectionne tous les éléments dont la valeur d'un attribut respecte une condition
- plusieurs types de condition sont possibles :
  - égalité (=)
  - contient (~= en CSS2 ou *= en CSS3)
  - commence par (|= en CSS2 ou ^= en CSS3)
  - finit par ($=)

Exemple

- sélectionne toutes les balises <a> dont l'attribut `href` commence par *https*
- affecte la couleur jaune au fond du texte associé au lien <p>

```
a[href^="https"] {
    background: #ffff00;
}
```

Sélecteur

Plus loin avec les sélecteurs

- les pseudo-classes d'état
- les pseudo-sélecteurs définissant des éléments relatifs à un autre
- les placements relatifs (une balise dans une autre balise)
- les correspondances sur les attributs

Pseudo-sélecteurs

- *:first-child* : premier fils d'une balise
- *:last-child* : dernier fils d'une balise
- *: :first-line* : première ligne du bloc, paragraphe, tableau, ...
- *: :first-letter* : première lettre du bloc, paragraphe, ...

Exemple

La couleur de fond du premier élément de la liste est jaune.

```
<html>
<head>
<style>
li:first-child {
    background: yellow;
}
</style>
</head>
<body>
<ul>
  <li>Coffee</li><li>Tea</li><li>Coca Cola</li>
</ul>
</body>
```

Pseudo-éléments

Modification par insertion d'éléments

- *: :before* : ajout d'un élément **avant** l'élement
- *: :after* : ajout d'un élément **après** l'élement

Exemple

```
<html>
<head>
<style>
p::after {
    content: "␣-␣Remember␣this";
    background-color: yellow;
    color: red;
    font-weight: bold;
}
</style>
</head>
<body>
<p>Hello world!</p>
</body>
</html>
```

Hello world! **- Remember this**

Pseudo-classes d'état

- *:hover* : si la souris passe au-dessus de l'élément
- *:focus* : si l'élément possède le focus
- *:active* : l'utilisateur clique sur l'élément
- *:checked* : l'élément est coché (boîte à cocher, par exemple)
- ...

Exemple

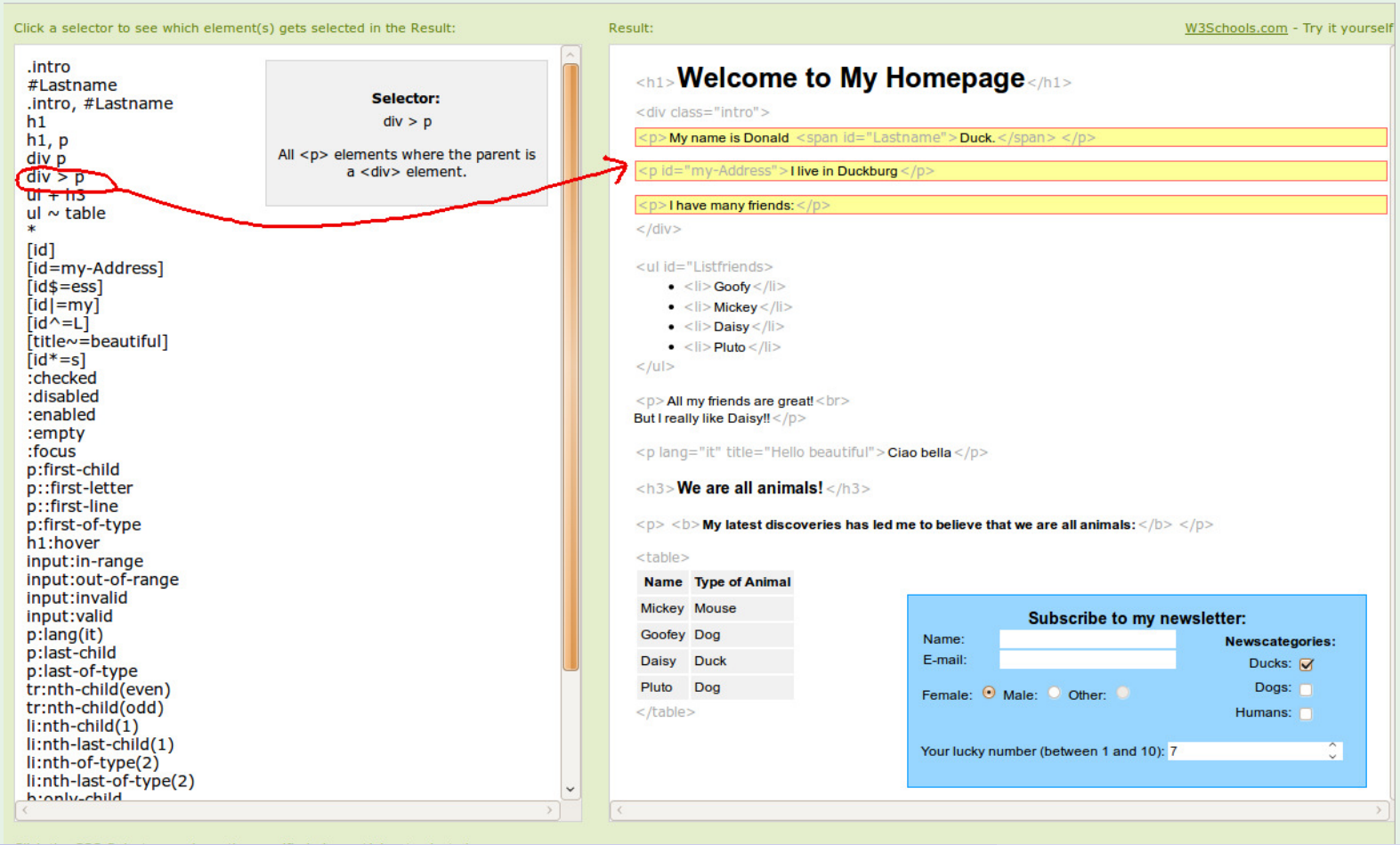
La couleur de fond des éléments h1 passe en jaune lorsque la souris le survol.

```
<html>
<head>
<style>
h1:hover {
    background-color: yellow;
}
</style>
</head>
<body>
<h1>Welcome to My Homepage</h1>
</body>
```

Sélecteur

Testeur

Pour tester les différentes formes de sélecteurs :
http ://www.w3schools.com/cssref/trysel.asp



Trois possibilités d'intégration

- un ou plusieurs fichiers externes
- dans l'entête du document
- directement dans la balise à l'aide de l'attribut `style`

Exemples

- fichier externe :

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

- dans l'entête :

```
<head>
  <style>
h1 {
    color: maroon;
    margin-left: 40px;
}
  </style>
</head>
```

- dans la balise :

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

Les attributs

- chaque balise accepte un certain nombre d'attributs
- plusieurs catégories :
  - background
  - texte et police de caractères
  - liens
  - listes
  - tableaux
  - "box" (bordure, marge, ...)
  - dimension, position et alignement
  - affichage

background

- la propriété `background` permet de définir le fond d'un élément (page, titre, ...)
- le fond est défini par :
  - une couleur (*background-color*)
  - une image (*background-image*)
  - une propriété de répétition (*background-repeat*)
  - une position (dans le cas d'une image) (*background-position*)

Exemple

- l'image contenu dans le fichier img_tree.png va être appliquée comme image de fond pour la page
- l'image n'est pas répétée ; même si elle est trop petite pour remplir toute la page
- l'image est placée en haut à droite de la page

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

Texte et couleur

- la propriété Color permet de fixer la couleur du texte
- une couleur est définie selon l'un des 5 modes :
  - une valeur hexadécimale : *#ff0000*
  - un code RGB : *rgb(255, 0, 0)*
  - un code RGB + un niveau d'opacité : *rgb(255, 0, 0, 0.5)* (0 étant le transparent)
  - un code HSL (*hue*, *saturation*, *lightness* - Teinte, Saturation et Luminosité) : *hsl(0, 100%, 50%)* pour le rouge
  - un nom de couleur (*red*, par exemple)

Exemple

Le texte des titres de type `h2` aura pour couleur le rouge.

```
h2 {
     color: rgb(255,0,0);
}
```

Texte et alignement

- il est possible de définir le placement d'un texte horizontalement
- 4 modes : calé à gauche ou à droite, justifié ou centré
- en mode justifié, les espaces sont dimensionnés afin que le texte remplisse tout l'espace

Exemple

```
h1 {
    text-align: center;
}
```

Texte et décor

- la notion de décor correspond aux éléments que l'on peut ajouter au texte comme le souligné
- 5 modes :
  - *none* : aucun décor de texte n'est utilisée
  - *underline* : souligné (déconseillé car utilisé pour les liens)
  - *overline* : texte surligné avec une ligne au-dessus
  - *line-through* : texte barré
  - *blink* : texte clignotant.

Exemple

Le titre sera souligné

```
h1 {
    text-decoration: overline;
}
```

Texte et transformation

- le texte peut être transformé en minuscule (lowercase) ou en majuscule (uppercase)
- la première lettre de chaque mot peut aussi être mise en majuscule (capitalize)

Exemple

Tout le texte du paragraphe sera en minuscule

```
p {
    text-transform: lowercase;
}
```

Texte et police de caractères

- la propriété relative à la police de caractères utilisée permet de définir la famille, le style et la taille
- une famille est un ensemble de polices de caractères de même type (rendu très proche) : `serif`, `sans-serif` et `monospace`
- lors du choix de `serif`, on peut spécifier plus précisement la police que l'on désire : Times, par exemple
- si le navigateur ne connait pas Times alors serif sera utilisé

Exemple

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

Taille des caractères

- la taille est soit spécifiée en absolu ou en relatif
- le mode absolu empêche l'utilisateur de modifier la taille des caractères via son navigateur
- il peut être utilisé si l'on connait la taille du dispositif d'affichage
- en mode relatif, la taille est définie en fonction des éléments environnants
- par défaut, la taille est 16px (ou 1em)
- l'unité em permet de définir des tailles propotionnelles à la taille de base (16px) : par exemple, 2.5em = 40px

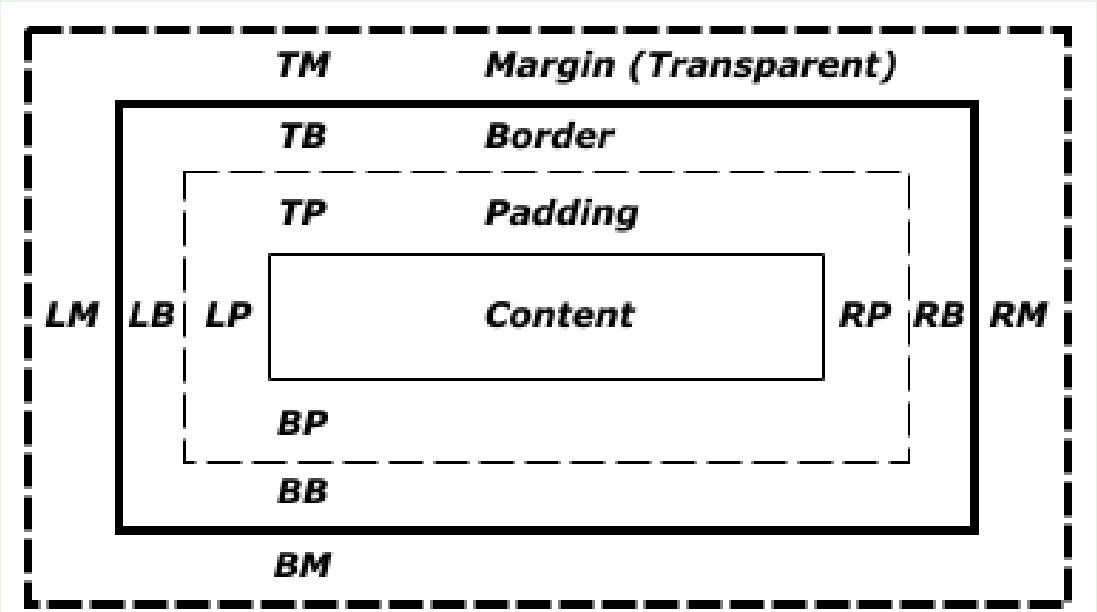
Exemple

```
h1 {
    font-size: 40px;
}

h2 {
    font-size: 1.875em;
}
```

"Box"

- tout élément HTML peut être vu comme une boîte
- cette boîte possède un ensemble de propriétés :
  - un contenu
  - un *padding* (l'espace entre le contenu et la bordure
  - une bordure
  - une marge (l'espace entre la bordure et les éléments environnants)

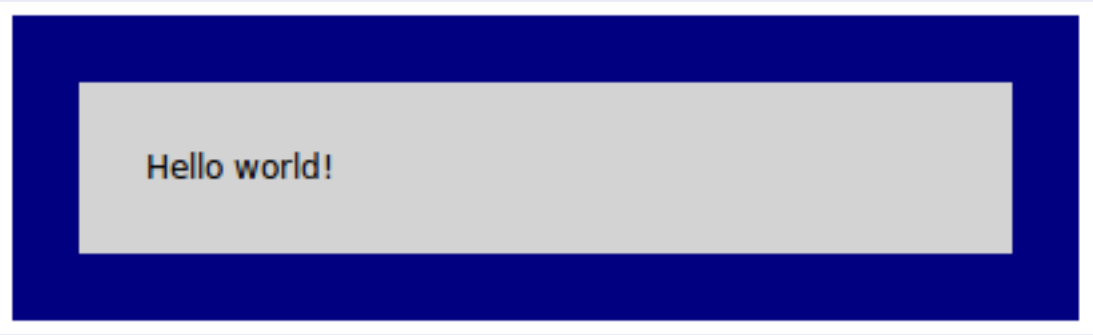
Modèle



Un exemple

- la largeur de la div est fixée à 300 pixels
- l'espace entre le contenu et la bordure est de 25 pixels
- la bordure est pleine, d'épaisseur 25 pixels et de couleur navy
- l'espace entre la bordure et les autres éléments est de 25 pixels

```
div {
    width: 300px;
    padding: 25px;
    border: 25px solid navy;
    margin: 25px;
}
```



"Box"

- les propriétés peuvent être globales ou relatifs à l'un des 4 côtés
- par exemple, `border-top-xxx` fixe un élément de style à la bordure supérieure
- le style, la couleur et la taille de la bordure peut être spécifiés

Exemple

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

Plan

Introduction

Objectifs

- des CSS compatibles avec l'ensemble des navigateurs (IE, Firefox, Chrome, Opera, ...)
- un système de conception de pages basé sur des grilles (*grid*)
- un support multi-écrans (desktop, tablette, mobile, ...)
- un framework de définition d'interface utilisateur
- des plugins JavaScript

Les éléments abordés

- le système de grille
- des CSS pour les tableaux, les formulaires, les boutons, ...
- des composants : barre de navigation, icônes, panels, ...
- des plugins Javascript : boîte de dialogue, carrousel, ...

Introduction

Installation

Deux possibilités :

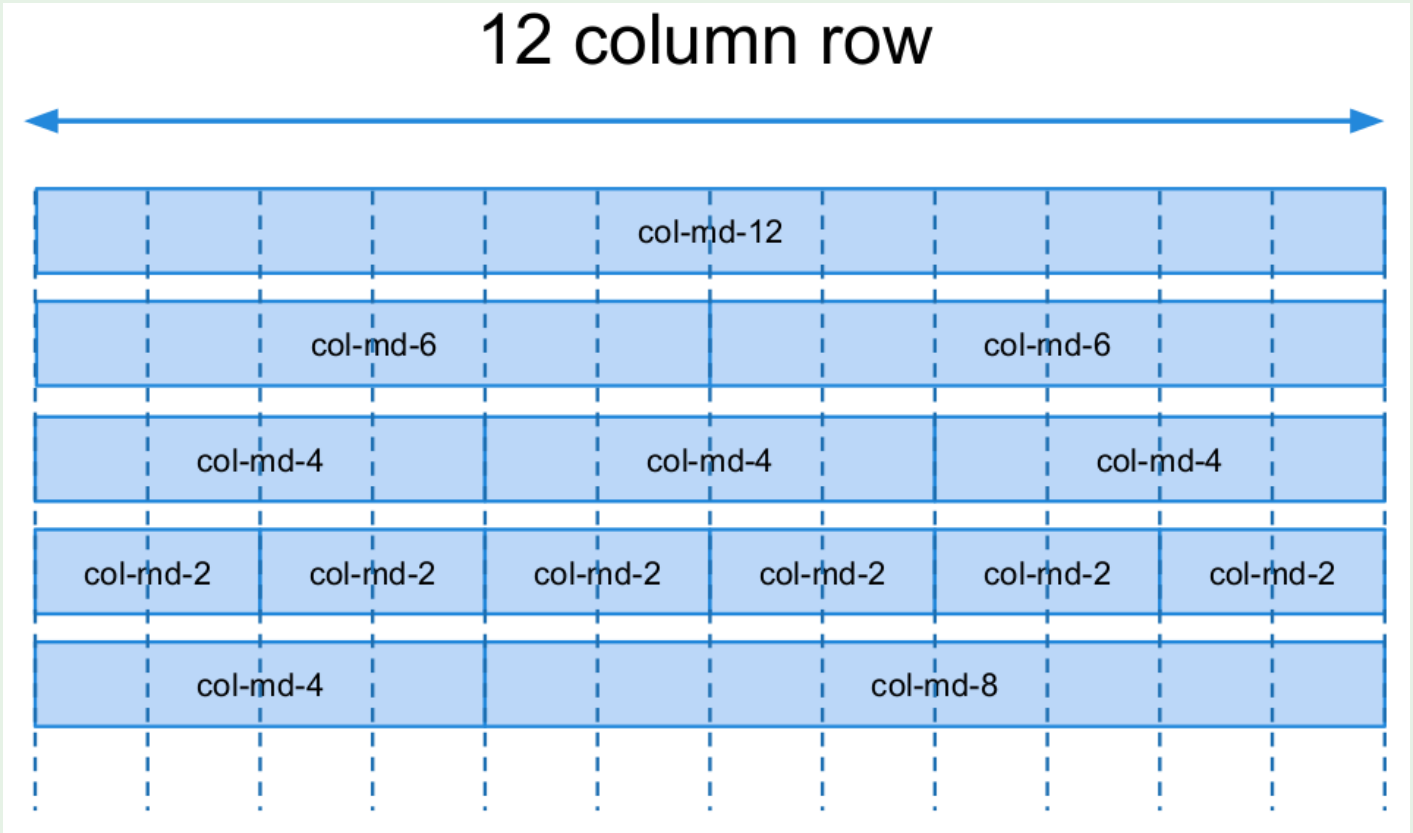
- télécharger le fichier css et le fichier Javascript depuis le site de Bootstrap (`http://www.getbootstrap.com/`)
- utiliser la version CDN (`content delivery network`)

```
<html>
<head>
  <link href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css"
        rel="stylesheet">
  <script src="//netdna.bootstrapcdn.com/bootstrap/3.0.0/js/bootstrap.min.js">
  </script>
</head>
<body>
...
</body>
</html>
```

Le système de grille

Définition

- une page est divisée en un ensemble de “lignes”
- une ligne est subdivisée en 12 colonnes
- pour chaque ligne, le contenu peut étendre sur une ou plusieurs colonnes



Le système de grille

Responsive

- Bootstrap permet de définir la disposition des contenus en fonction de la taille du dispositif (desktop, tablette, mobile, ...)
- 4 systèmes de dimension sont disponibles :
  - col-xs-[num] : pour les *extra small devices*
  - col-sm-[num] : pour les *small devices* - largeur $>$ 768px
  - col-md-[num] : pour les *medium devices* - largeur $>$ 992px
  - col-lg-[num] : pour les *large devices* - largeur $>$ 1200px

num correspond au nombre de colonnes occupées par le contenu

- la page va s'adapter automatiquement à la taille de l'écran

Le système de grille

Exemple

```
<div class="row">
  <div class="col-md-4 col-xs-6">
    <!-- content -->
  </div>
  <div class="col-md-8 col-xs-6">
    <!-- content -->
  </div>
</div>
```

Explications

- si l'écran est de taille *medium* alors la ligne est divisée en 2 parties : l'une sur 4 colonnes et l'autre sur 8 colonnes
- si l'écran est de taille *small* alors la ligne est divisée en 2 parties : 6 colonnes + 6 colonnes

Le système de grille

Container

Les lignes doivent être définies dans un container.

Exemple

```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-xs-6">
      <!-- content -->
    </div>
    <div class="col-md-8 col-xs-6">
      <!-- content -->
    </div>
  </div>
</div>
```

Attention

Les div de type `col` doivent être obligatoirement les fils immédiats de div de type `row`.

Le système de grille

Offset

- il est possible de décaler les contenus d'une ou plusieurs colonnes sur la droite
- la classe du div contient alors à la fois un *col-xx-y* et un *col-xx-offset-y*

Exemple

```
<div class="row">
  <div class="col-md-4">XXXX</div>
  <div class="col-md-4 col-md-offset-4">YYY</div>
</div>
<div class="row">
  <div class="col-md-3 col-md-offset-3">ZZZZZZZZ</div>
  <div class="col-md-3 col-md-offset-3">WWWWWWWW</div>
</div>
```

Explications

- la première ligne débute par un contenu sur 4 colonnes puis un espace de 4 colonnes et ensuite un contenu sur 4 colonnes
- la deuxième ligne débute par un espace de 3 col., un contenu sur 3 col. puis un espace de 3 col. et ensuite un contenu sur 3 col.

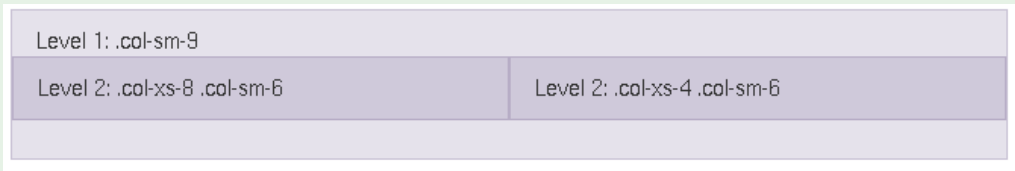
Le système de grille

Des colonnes dans des colonnes

- à l'intérieur de colonnes, des lignes peuvent être définies
- le système des 12 colonnes s'applique de nouveau à la zone définie par les colonnes englobantes

Exemple

```
<div class="row">
  <div class="col-sm-9">
    Niveau 1
    <div class="row">
      <div class="col-sm-6">
        Niveau 2
      </div>
      <div class="col-sm-6">
        Niveau 2
      </div>
    </div>
  </div>
</div>
```



Les boutons

Boutons

- la classe des boutons est applicables sur plusieurs balises : <a>, <button> et <input>
- 6 styles sont possibles : default, primary, success, info, warning et danger
- chaque style est associé à une couleur
- le système de taille est applicable

Exemple

```
<a href="#" class="btn␣btn-primary␣btn-lg" role="button">Un lien</a>
```

Explications

- un bouton apparaît à la place d'un lien classique grâce à l'attribut `role`
- la couleur est bleu : type `primary`
- la taille est large

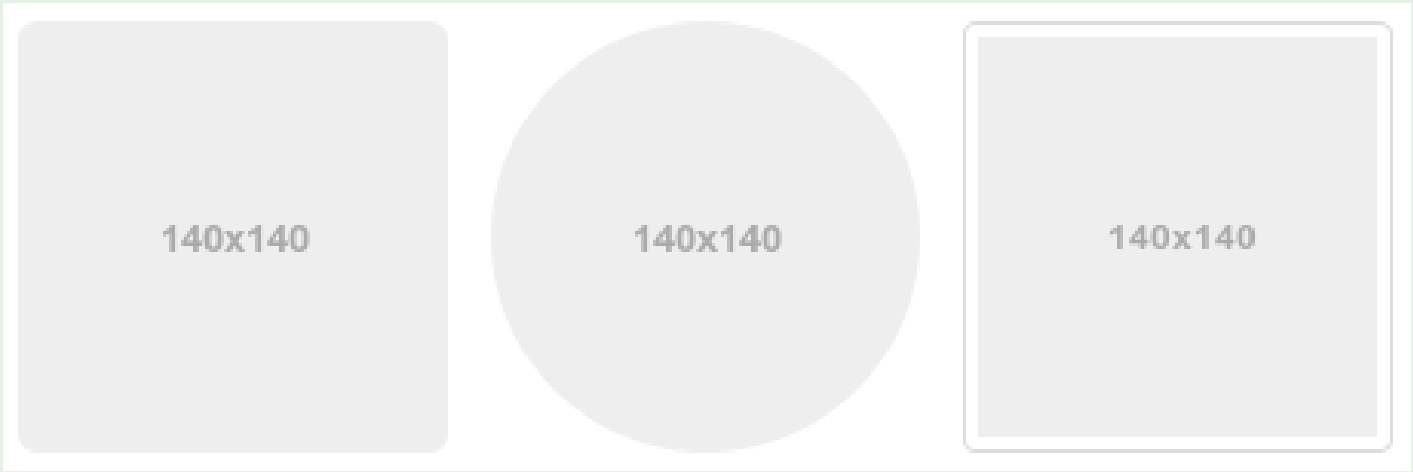
Les images

Définition

- un cadre autour des images peut être défini
- 3 types :
  - arrondi
  - en cercle
  - en miniature

Exemple

```
<img src="..." alt="..." class="img-thumbnail">
```



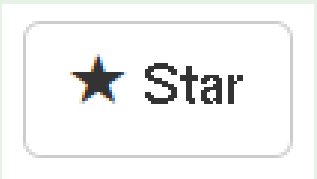
Les icônes

Glyphicon Halflings

- la police de caractères *Glyphicon Halflings* est disponible avec Bootstrap
- cette police de caractères propose des icônes qui peuvent être utilisées dans les menus, les boutons, ...
- la taille est gérée par le même système que les colonnes

Exemple

```
<button type="button" class="btn␣btn-default␣btn-lg">
  <span class="glyphicon␣glyphicon-star" aria-hidden="true"></span> Star
</button>
```



Attention, vérifier bien que le répertoire `fonts` est installé !

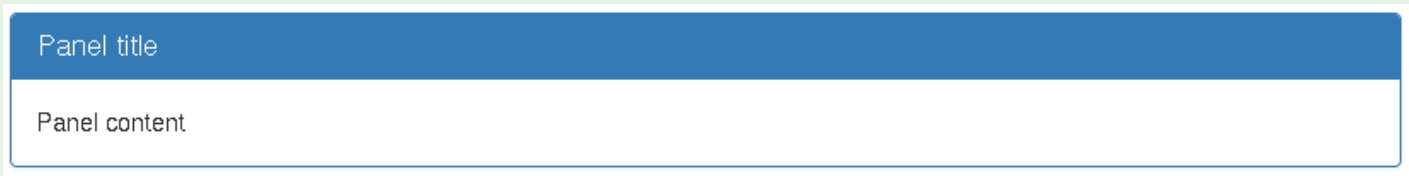
Les panels

Panels

- un panel est une boite avec potentiellement une entête et un “pied de page”
- la boite a un contour avec des coins arrondis

Exemple

```
<div class="panel panel-primary">
  <div class="panel-heading">Panel title</div>
  <div class="panel-body">
    Panel content
  </div>
</div>
```



Plan

Introduction

Les éléments abordés

- les valeurs, les types, les variables et les opérateurs
- les tableaux
- les instructions de contrôle
- les fonctions
- les classes et les objets
- this
- le DOM et javascript

HTML et Javascript

Le Web lie de manière étroite HTML et Javascript :

- HTML : le langage de présentation
- Javascript : le langage de manipulation et de contrôle

Introduction

Caractéristiques

- langage simple, flexible et interprété
- peu typé : les variables sont typées par affectation
- peut être utilisé comme un langage objet
- technologie du côté client (et maintenant du côté serveur avec nodejs)

Utilisations

- pour valider des éléments d'un formulaire
- réagir aux actions de l'utilisateur
- changer une image lors d'une action avec la souris
- faire apparaître ou disparaître des éléments d'une page
- charger dynamiquement des éléments
- interagir avec un serveur sans recharger la page

Introduction

Capacités

Javascript peut :

- capturer les évéments (clavier, souris, . . .)
- lire et écrire des éléments HTML dans le DOM
- valider les données d'un formulaire
- accèder et modifier les cookies d'un navigateur
- détecter le type de navigateur et de systèmes d'exploitation
- utiliser comme un langage orientée objets

Introduction

Un exemple simple

- la balise <script> permet d'introduire du code Javascript dans une page HTML
- le script est exécuté au chargement de la page
- une popup s'ouvre avec le message "Hello world !"

```
<html>

<body>
  <script type="text/javascript">
    alert('Hello world!');
  </script>
</body>

</html>
```

Un second exemple simple

- via l'objet document, le texte "Hello world !" est ajouté à la page

```
<html>

<body>
  <script type="text/javascript">
    document.write('Hello world!');
  </script>
</body>

</html>
```

Introduction

Localisation du code

Le code Javascript peut être placé :

- dans la partie entête (HEAD)
- dans la partie entête (BODY) $\rightarrow$ non recommandé
- dans un fichier externe

Recommandation

Utilisé les fichiers externes

- le lien se fait à l'aide de la balise <SCRIPT> dans l'entête
- les fichiers peuvent alors être mis dans le cache du navigateur

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

Introduction

Execution

Les scripts sont exécutés :

- lors du chargement de la page
- lors d'un évenement

Définition

Toutes les instruction sont exécutées mais certaines ne sont que des déclarations de fonctions.

Evénement

Le code et les fonctions peuvent être appelés lors d'événements via des attributs spéciaux

```
<img src="logo.gif" onclick="alert('clicked!')" />
```

- une popup s'ouvre lorsque l'on clique sur l'image
- le code est directement mis dans l'attribut `onclick`

Introduction

Fonction et événement

- des fonctions sont définies dans l'entête
- l'appel se fait lors de l'événement

Evénement

```
<html>
  <head>
    <script type="text/javascript">
      function test (message) {
        alert(message);
      }
    </script>
  </head>

  <body>
    <img src="logo.gif" onclick="test('clicked!')" />
  </body>
</html>
```

Introduction

Le langage

Très similaire à Java et C#

- un ensemble d'opérateurs arithmétiques et logiques (+, *, =, !=, &&, ++, ...)
- les variables à typage par affectation (typage faible)
- des structures conditionelles (if, else)
- des structures de boucle (for, while)
- les tableaux associatifs (my_array['abc']) ou non (my_array[0])
- les fonctions avec retour (et la notion de variable de type fonction)
- ...et les objets

Types simples

4 types

- les chaînes de caractères : String (codage 16 bits – utilisation des simples et doubles quotes)
- les numériques : Number (double, 64bit floating point),
- les booléens : Boolean
- les expressions régulières : RegExp

Booléen

- Faux : false, null, undefined, '', 0, NaN
- Vrai : tout le reste !

Types simples

Les expressions régulières

- TODO !!!

Les opérateurs

Arithmétique

- unaire : +expr -expr
- multiplication et division : * / %
- addition et soustraction : + -,
- postfix : expr++ expr−
- décalage : >> << >>>

Logique et relationnel

- logique : & | && ||
- test : ? :
- égalité : == != === !==
- relationnel : in instanceof

Les opérateurs

Egalité : == ou ===

=== : même type et même valeur

- 5 === 5 // => true
- 5 === 5.0 // => true
- 'a' === "a" // => true
- 5 === '5' // => false
- [5] === [5] // => false
- new Number(5) === new Number(5) // => false
- var a = new Number(5) ;
- *a* === *a* // => true
- false === false // => true

[5], "new Number(5)" et a sont des objets

Les opérateurs

Affectation

- simple : =
- arithmétique : += -= *= /= %=
- logique : &= ≙ |= <>= >>>=

Objet

- new : création
- delete

Les variables

Définition

- prédéclarée avant l'utilisation (sauf mécanisme de Hoisting)
- pas de déclaration de type
- typage dynamique = par la valeur
- la portée est limitée à la fonction

Exemple

```
var a = new Number(5);
var b = 5;

function f (x) {
  if (x > 0) {
    var y = 1;
  }
  return y;
}
```

Les variables

Objet

- toute variable est considéré comme un objet
- donc, des méthodes s'appliquent

Exemple

```
var test = "some string";

alert(test[7]); // retourne 'r'
alert(test.charAt(5)); // retourne 's'
alert("test".charAt(1)); // retourne  'e'
alert("test".substring(1,3)); // retourne 'es'
```

$\rightarrow$ charAt et substring sont des méthodes que l'on peut appliquer sur des chaînes de caractères

les tableaux

Définition

- un type intégré au langage
- plusieurs façons de créer :
  - par new
  - par un simple [ ]
- ajout :
  - par une simple affectation
  - par la fonction push
- retrait :
  - comme une pile : pop
  - splice
- taille : length

Exemple

```
// creation
var empty = new Array(2);
// un tableau vide de taille  2
var array = ["1", "2"];
// lecture
empty[0]; // => undefined
empty[5]; // => undefined
// ecriture
empty[0] = "Test";
// empty est maintenant :
// ["Test", undefined]
empty[2] = "Test";
// empty est maintenant :
// ["Test", undefined, "Test"]
```

Les instructions de contrôle

Liste

- conditionel : switch, if/else,
- itération : while, do/while, for,
- branchement : return, break, continue,
- exception : throw, try/catch/finally

if/else

```
if (a === b) {
} else {
}
```

for

```
for (var i = 0; i < n; ++i) {
}

for (var i in L) {
}
```

Exceptions

Les exceptions permettent de contrôler les erreurs d'exécution.

Les fonctions

Définition

- pas de type de retour
- pas de typage des paramètres
- pas de contrôle du nombre de paramètre
  - si appel avec insuffisamment de valeur, alors paramètre = undefined
  - si appel avec trop de valeurs, alors ignoré
- retourne une valeur à l'aide de `return`

Les fonctions

Fonctions

- deux formes :
  - déclaration d'une fonction
  - définition via une variable
- invocation identique quelle que soit la déclaration

Déclaration

```
function myFunction(a, b) {
  return a * b;
}

var x = function(a, b) {return a * b};
```

Appel

```
r = myFunction(2, 2);

u = x(2, 2);
```

Les classes et les objets

Définition

- une classe est une structure de données associée à un ensemble de méthodes
- une méthode est une fonction qui manipule les données (attributs) de la structure
- un objet est une instantiation d'une classe ou non

Déclaration des classes

- deux formes :
  - les fonctions
  - les prototypes
- instantiation via l'opérateur new

Déclaration

```
var maClasse = function(a) {
  this.maMethode = function(x) {
    this.unAttribut = 0;
  }

  this.unAttribut;

  var _a = a;
}
```

var / this

var déclare les attributs/méthodes "privés" et this, les "publiques".

Les classes et les objets

Les objets via les tableaux

- déclaration à l'aide de
- association clé $\rightarrow$ valeur :
  - clé : le nom d'un attribut ou d'une méthode
  - valeur : la valeur d'un attribut ou une fonction

Déclaration

```
> var pt = { x: 10, y: 10, w: function() { return this.x + this.y; } };
> pt.w();
20
> typeof pt
'object'
```

Les classes et les objets

Une classe via une fonction

- une classe est une fonction avec des variables locales et des fonctions
- instantiation à l'aide de l'opérateur `new`

Déclaration

```
var Person = function(name) {
  var private_name = name;

  this.get_name = function() {
    return private_name;
  };
  this.set_name = function(new_name) {
    private_name = new_name;
  };
};

var p = new Person('toto');
```

Problème mémoire

Si deux objets sont issus de Person, tout est dupliqué variables ET méthodes

Les classes et les objets

Une classe via les prototypes

Les prototypes permettent de partager les méthodes entre les objets d'une même classe

Déclaration

```
var Person = function(name) {
  this.name = name;
};

Person.protoype.get_name = function() {
    return this.name;
};

Person.protoype.set_name = function(new_name) {
    name = new_name;
};

var p1 = new Person('toto');
var p2 = new Person('titi');
```

On a perdu les attributs privés !

This

Définition

- le mot clé `this` référence le “proprétaire” de la fonction
- dans le cas des fonctions d'un objet, this désigne l'objet
- dans un contexte de Javascript embarqué dans de l'HTML, this désigne l'élément dans lequel est le code (par exemple, un formulaire)

DOM

Définition

- le DOM est une présentation arborescente d'une page Web
- chaque noeud est une balise HTML et correspond à un élément de la page
- Javascript propose des méthodes d'accès au DOM
- tous les éléments d'une page Web est manipulable
- **avantage : faire des modifications sans rechargement de la page**

Exemple

- accès via l'attribut id

```
var elem = document.getElementById("un_id")
```

- accès via un nom (par exemple, pour un champ d'un formulaire)

```
var elems = document.getElementsByName("un_nom")
```

- accès via un type de balise

```
var elems = document.getElementsByTagName("img")
```

Exemple

On peut lire la valeur d'un attribut et modifier sa valeur.

```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".png";

  var statusDiv = document.getElementById("statusDiv");
  statusDiv.innerHTML = "The␣lamp␣is␣" + state;
}
```

Commentaires

- premier bloc :
  - le nom du fichier image est changé
  - il est construit à partir d'une variable state passée en paramètre
  - l'image est chargée et elle est remplacée dans la page
- deuxième bloc : le code HTML de la div *statusDiv* est changé

DOM

Attributs

- la pluplart des attributs ont le même nom que les attributs des balises :
  - id, name, href, alt, title, src, ...
- l'attribut *style* donne accès au style CSS de l'élément (en mode inline) :
  - style.width, style.marginTop, style.backgroundImage, ...
- l'attribut *className* donne accés à l'attribut HTML *class*
- l'affectation de *innerHTML* remplace la totalité du code HTML à l'intérieur de l'élément (par exemple, d'une div)
- certains attributs sont en lecture seule :
  - tagName, offsetWidth, offsetHeight, scrollHeight, scrollTop, nodeType, ...

Navigation dans le DOM

- on peut se déplacer dans l'arbre DOM
- les méthodes s'appliquent à partir d'un noeud de l'arbre
- quelques méthodes :
  - element.childNodes : liste des noeuds fils
  - element.parentNode : le noeud parent (s'il existe)
  - element.firstChild : premier noeud fils
  - element.lastChild : dernier noeud fils
  - element.nextSibling : le noeud suivant vis à vis du noeud père et du noeud courant
  - element.previousSibling : le noeud précédent vis à vis du noeud père et du noeud courant

Exemple

```
var el = document.getElementById('form_div');
alert(el.childNodes[0].value);
alert(el.childNodes[1].getElementsByTagName('span').id);

<div id="form_div">
  <input type="text" value="test␣text" />
  <div>
    <span id="test">test span</span>
  </div>
</div>
```

On affiche dans une popup :

- la valeur du champ de saisie
- l'id de la balise span

DOM

Manipulation du DOM

- on peut aussi créer, supprimer et modifier des éléments du DOM
- quelques fonctions :
  - createElement : créé un noeud Element
  - createTextNode : créé un noeud Text
  - appendChild : ajoute un nouvel enfant au noeud courant
  - insertBefore : insère un nouvel enfant avant le noeud de référence
  - removeChild : retire le noeud du DOM
  - setAttribute(nom, valeur) : ajoute ou modifie un attribut à l'élément courant

JQuery

- via l'API de base, la manipulation du DOM est assez complexe
- JQuery permet de simplifier ces opérations

DOM et les événements

Déclaration

- les événements sont capturés par le navigateur
- le navigateur exécute la fonction associée à l'événement
- un événement est attaché à un objet (le document, un bouton, une image, ...)
- il existe deux façons de déclarer les fonctions rattachées aux événements :
  - à l'aide d'un attribut spécifique à un événement dans une balise HTML
  - à l'aide du DOM (et au chargement de la page, par exemple)

DOM et les événements

Exemple 1

- on considère que la fonction *imageClicked* est au préalable définie
- la fonction *imageClicked()* sera appelé si on clique sur l'image

```
<img id="monImage" src="test.gif" onclick="imageClicked()" />
```

Exemple 2

La fonction `imageClicked()` sera appelé si on clique sur l'image dont l'identifiant est *monImage*

```
var img = document.getElementById("monImage");

img.onclick = imageClicked;
```

Les événements

Paramètre

- tous les événements reçoivent un paramètre
- il contient les informations relatives à l'événement
- il contient le type de l'événement (clic souris, appui sur une touche, ...)
- il peut contenir des informations de localisation lors d'un clic
- une référence sur l'objet mis en jeu est aussi contenue (le bouton sur lequel l'utilisateur a cliqué)
- lors de l'appui sur une touche, l'état des touches shift, alt et ctrl sont disponibles

Exemple

```
//TODO
```

Les événements

Souris

- onclick : clic
- onmousedown : l'un des boutons de la souris est enfoncé
- onmouseup : l'un des boutons est relaché
- onmouseover : le curseur vient d'entrer dans la zone graphique de l'élément
- onmouseout : le curseur vient de sortir de la zone graphique de l'élément
- onmousemove : la souris a bougé

Clavier

Les événements clavier valident sur des champs de formulaire et pour les canvas

- onkeypress : l'utilisateur vient d'appuyer sur une touche
- onkeydown : une touche vient d'être enfoncée
- onkeyup : une touche vient d'être relachée

Les événements

Interface

Il existe des événements liés à l'interface

- onblur : //TODO ? ? ? ?
- onfocus : l'élément vient d'obtenir le focus (par exemple, un champ de saisie d'un formulaire)
- onscroll : l'utilisateur a fait *scroller* la fenêtre

Formulaire

- onchange : un champ de saisie vient de changer de valeur
  rightarrow permet de mettre à jour un élément en fonction du contenu d'un champ
- onsubmit : le formulaire vient d'être soumis
  rightarrow permet de déclencher une fonction de validation des valeurs saisies

Les événements

Exemple

```
function checkForm()
{
  var valid = true;

  if (document.monFormulaire.nom.value === "") {
    alert("Nom␣obligatoire␣!");
    document.getElementById("ErreurNom").style.display = "inline";
    valid = false;
  }
  return valid;
}


<form name="monFormulaire" onsubmit="return␣checkForm()">
  <input type="text" name="nom" />
</form>
```

- le formulaire contient un champ texte
- la fonction *checkForm* teste si le champ texte est vide
- si le champ est vide alors l'élément dont l'identifiant est ErreurNom change de style (`display` est affecté à `online`)

Les événements

Chargement et déchargement

- lorsqu'une page est chargée ou déchargée, *onload* et *unload* sont générés
- la déclaration est faite exclusivement dans la balise body
- on peut définir des actions lors du chargement d'une page (affichage d'une popup, par exemple)

Exemple

```
<html>
<head>
  <script type="text/javascript">
    function hello() {
      alert("Page␣chargee");
    }
  </script>
</head>
<body onload="hello()" >
</body>
</html>
```

Les objets prédéfinis

Définition

Les navigateurs proposent plusieurs objets globaux :

- window :
  - référence le noeud racine du DOM
  - représente la fenêtre du navigateur
- document : le document chargé
- screen : les propriétés de l'écran d'affichage (résolution, ...)
- browser : les informations concernant le navigateur (le type, le système d'explotation, ...)

JSON

Définition

- JSON = *JavaScript Object Notation*
- utilisé pour la définition des tableaux associatifs
- utilisé aussi pour l'échange de données entre le client et le serveur sous forme de chaînes de caractères
- deux fonctions de serialisation :
  - JSON.stringify(x) : production d'une chaîne de caractères à partir d'un tableau associatif (ou d'un objet)
  - JSON.parse(data) : reconstruction d'un tableau associatif (ou d'un objet) à partir d'une chaîne de caractères au format JSON

Ajax

Problématique

- lorsqu'une page est chargée, le contenu peut être modifié à l'aide :
  - d'actions (événements)
  - de fonctions locales déclenchées par des timers
- c'est à dire, à partir d'éléments purement locaux
- question : comment réaliser une modification :
  - sans rechargement de la page
  - en fonction du résultat d'une requête exécutée sur le serveur

La solution

Ajax : *Asynchronous JavaScript and XML* // TODO

Ajax

Exemple

```
function doAjaxCall() {
  var request = null;
  if (window.XMLHttpRequest) {
    request = new XMLHttpRequest();
  }
  if (request) {
    request.open("GET", "http://127.0.0.1/foo.php");
    request.onreadystatechange = function() {
      if (request.readyState === 4) {
        document.getElementById("resultDiv").innerHTML = request.responseText;
      }
    }
    request.send(null);
  }
}
```

- vérification du support d'Ajax par le navigateur
- création de la requête et ouverture en mode GET
- définition de la fonction de récupération de la réponse
- envoi de la requête

Plan

Introduction

C'est quoi ?

- c'est une bibliothèque JavaScript
- simple : une seule fonction jQuery(...) / un seul opérateur $
- multi-navigateur
- communauté très active

Les éléments abordés

Manipulation du DOM et des événements

- sélectionner les éléments : **trouver**
- faire quelque chose avec

Introduction

Et les autres ?

Il existe d'autres bibliothèques :

- Prototype (http ://prototypejs.org/)
- Dojo (http ://dojotoolkit.org/)
- Yui de Yahoo (http ://yuilibrary.com/)
- Mootools (http ://mootools.net/)

Utilisation de jquery

- en mode CDN (*Content Delivery Network*) : ajouter dans l'entête du fichier HTML :

```
<script src="http://code.jquery.com/jquery-1.11.2.min.js"></script>
```

- en téléchargeant le source minimisé ou non :

```
<script src="js/jquery-1.11.2.min.js"></script>
```

Recherche

Recherche

- sélectionner une div avec un *id*

```
$('div#id')
```

- ou plus simplement via l'id

```
$('#id')
```

- sélectionner un élément dont la classe est *class*

```
$('.class')
```

Remarque

On peut remplacer le symbole $ par jQuery(...).

Recherche

Collection

- si la recherche sélectionne plusieurs éléments alors on obtient une collection (une liste)
- on peut accèder un élément par l'opérateur []
- on peut obtenir la taille à l'aide de l'attribut length ou de la méthode *size*

```
list = $('div')
list[0]
list.length
list.size()
```

Recherche

Parcours

On peut aussi parcourir la liste :

- à l'aide d'une boucle *for*
- via la méthode *each* et une fonction anonyme :

```
$('div').each(function() { this.hide(); });
```

Each

- la méthode each parcourt l'ensemble des éléments de la collection
- appelle la fonction anonyme pour chacun d'eux
- *this* référence l'élément en cours de traitement

C'est une forme de programmation très courante en Javascript.

Manipulation

Modification

- le texte contenu dans une balise peut être changé

```
$('div').text('Hello world!');
```

- le code HTML contenu dans une balise peut l'être aussi

```
$('div').html('<p>Hello world!</p>');
```

Modification des attributs

- la valeur des attributs des balises est modifiable

```
$('img#mini').attr('src', 'img/img_small.jpg');
```

- l'attribut src de la balise img dont l'id est mini est modifié
- il est possible d'en modifier plusieurs à l'aide d'un unique appel

```
$('img#mini').attr({'src': 'img/img_small.jpg', 'width': '100px'});
```

- on utilise la syntaxe de définition d'un tableau associatif : la clé est le nom de l'attribut et la valeur, la nouvelle valeur de l'attribut

Manipulation

Attribut

- on peut récupérer la valeur d'un attribut

```
$('img#mini').attr('src');
```

- la suppression est aussi possible

```
$('img#mni').removeAttr('width');
```

Manipulation

Classe

Deux méthodes spécifiques pour les classes

- si un style est défini via un css, on peut dynamiquement le rattacher à un élément

```
$('p').addClass('new');
```

- un style est défini pour les balises a via la classe new
- la suppression est aussi possible

```
$('p').removeClass('new');
```

- la classe est retirée à tous les éléments de type p et par conséquent, le style qui y était rattaché

CSS

- il est possible de définir des styles à la volée

```
$('p').css({'color': 'red', 'font-size': '20pt'});
```

- tous les paragraphes passent en rouge avec des caractères de taille 20pt

Manipulation

Création de balises

- l'appel à jQuery avec une balise en paramètre déclenche sa création :

```
var p = $('<p/>');
```

- l'élément créé est ensuite rattaché à un autre élément du DOM

```
var row = $('<div/>', { class: 'row' });
row.appendTo($('#main'));
```

- une div est créée, sa classe est row et elle est rattachée à un élément dont l'id est main

Il est possible de construire une page Web entièrement à l'aide de jQuery.

Evénements

Démarrage

On peut spécifier une fonction qui sera exécutée lors du chargement d'une page

```
$(document).ready(function() {
    alert('Hello world!');
});
```

Clic

- la réponse aux événements est réalisée via des fonctions anonymes
- il faut sélectionner le ou les éléments puis faire appel à la fonction relative à l'événement dont on veut répondre

```
$('p').click(function() {
    console.log('Clic');
});
```

- une deuxième technique est possible :

```
$('p').on('click', function() {
    console.log('Clic');
});
```

Evénements

Evénements multiples

Plusieurs événements peuvent être possibles pour un même élément

```
$('p').on({
  'click': function() { console.log('clicked!'); },
  'mouseover': function() { console.log('hovered!'); }
});
```

Fonctions nommées

La réponse à un événement peut être définie dans une fonction non anonyme

```
var my_function = function(event) {
  console.log('clicked!');
};

$('p').click(my_function);
```

Pour tout événement, un paramètre peut être passé ; il contient les informations liés à l'événement.

Plan

Introduction

Les éléments abordés

- les éléments d'un formulaire
- les méthodes
- la vérification avec HTML5 et Javascript/jQuery

Définition

Un formulaire est :

- un élément composé de zones de saisie et de boutons
- objectif : envoyer des informations au serveur Web
- côté serveur : lancer un traitement, stocker des informations, ...

Eléments

Les éléments d'un formulaire

- les zones de saisie texte simple ou multi-lignes
- les boutons radio
- les boîtes à cocher
- les listes déroulantes
- les boutons d'envoi

Un exemple simple

Un formulaire avec un champ de saisie et un bouton pour "soumettre" le formulaire.

```
<form method="post" action="create.php">
  <p>Name: <input type="text" name="name" /></p>
  <input type="submit" />
</form>
```

Un champ de saisie est identifié par un nom (attribut *name*).

Eléments

Zone de saisie texte

- deux types : texte "visible" (*text*) et texte "invisible" (*password*)
- possibilité de contrôle du nombre de caractères max (*maxlength*)
- la balise `textearea` permet la saisie de texte multilignes → il faut préciser le nombre de lignes (attribut *rows*) et le nombre de colonnes (attribut *cols*)

Exemple multilignes

```
<p>Comments:</p>
<textaera name="comments" rows="5" cols="20" />
```

Eléments

Les boutons radio

- objectif : faire un choix parmi plusieurs possibilités
- des balises `input` dont le type est *radio*
- chaque possibilité possède une valeur qui sera transmis au serveur
- par défaut, aucun bouton n'est coché
- l'attribut *checked* permet d'indiquer un choix par défaut

Exemple

```
<input name="language" value="html" type="radio" checked />HTML
<input name="language" value="js" type="radio" />JavaScript
<input name="language" value="php" type="radio" />PHP
```

Eléments

Les cases à cocher

- objectif : faire un ou plusieurs choix parmi plusieurs possibilités
- des balises `input` dont le type est *checkbox*
- chaque possibilité possède une valeur (attribut *value*) qui sera transmis au serveur sous forme d'une liste
- par défaut, aucune case n'est cochée
- l'attribut *checked* permet d'indiquer un choix par défaut

Exemple

```
<input name="language" value="html" type="checkbox" checked />HTML
<input name="language" value="js" type="checkbox" />JavaScript
<input name="language" value="php" type="checkbox" />PHP
```

Eléments

Les listes déroulantes

- objectif : faire un ou plusieurs choix parmi plusieurs possibilités
- les choix se présentent sous forme d'une liste déroulante
- deux balises sont utilisées :
  - `select` : la balise qui encadre la liste des possibilités
  - `option` : la balise des possibilités ; chaque possibilité possède une valeur (attribut *value*) qui sera transmis au serveur
- par défaut, aucune option n'est sélectionnée
- l'attribut *selected* permet d'indiquer un choix par défaut

Eléments

Exemple

```
<p>Languages:<br />
<select name="language">
  <option value="html">HTML</option>
  <option value="js">JavaScript</option>
  <option value="php">PHP</option>
</select>
</p>
```

Multiple

Pour les listes à sélection multiple, il faut ajouter l'attribut *multiple*.

La liste peut se présenter soit :

- sous la forme d'un champ avec un bouton clicable pour obtenir les choix
- sous la forme d'une liste avec une barre de défilement $\rightarrow$ il faut alors ajouter l'attribut *size* qui indique le nombre d'éléments visibles

Eléments

Les champs invisibles

- possibilité de définir les champs qui ne sont pas visibles
- utilité : stocker des données qui pourront être envoyé au serveur
- exemple : des identifiants de session ou des valeurs de clés primaires associées aux données du formulaire, par exemple
- le type devient alors *hidden*
- pour voir les variables et leurs valeurs, il faut regarder le code

Exemple

```
<input name="id" value="534" />
```

Soumission

Soumission

- un formulaire est défini dans une balise *form*
- deux attributs sont obligatoires :
  - *action* : url où sera envoyé les données du formulaire
  - *method* : la façon dont les données sont envoyées (*POST* ou *GET*)
- un bouton de soumission doit être présent

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

Soumission

La méthode GET

- les données sont ajoutées à la fin de l'URL
- à utiliser pour des formulaires avec peu de données
- pour des données peu sensibles

La méthode POST

- les données sont placées dans la requête HTTP
- non visible directement dans l'URL
- possibilité de protection des données

Validation

Validation

- les données d'un formulaire ont en général besoin d'être validé avant l'envoi au serveur
- par exemple, vérifier qu'une adresse mail est correctement définie
- la validation peut se faire aussi du côté serveur
- côté client, deux possibilités :
  - en Javascript
  - en HTML5

En HTML5

Trois attributs :

- *required* : indique que le champ est obligatoire
- *pattern* : une expression régulière pour vérifier que la valeur saisie possède le bon format
- *placeholder* : une indication de ce que l'on attend dans une info bulle

Validation

En HTML5 - suite

- il est possible de préciser la nature de l'information à saisir en changeant le type (par défaut, texte)
- pour les champs numériques, on peut indiquer un minimum et un maximum via les deux attributs *min* et *max* ; le type devient alors `number`
- on peut préciser : date, time, url, email, search et color

En Javascript

Deux possibilités :

- ajout de l'attribut *onsubmit* dans la balise `form`
  - avant l'envoi du formulaire, un évément *submit* est généré
  - la fonction Javascript doit se terminer par un *return true ;* ou *return false ;* afin d'indiquer si le formulaire est correctement rempli
- ajout de l'attribut *onchange* dans les balises `input` et `select`
  - dès que l'utilisateur réalise une modification dans un champ ou dans une liste, l'événement *change* est généré
  - la réponse peut consister à ajouter un message à côté du champ

Validation

En jQuery

- accès à la valeur d'un champ :

```
<p>Phone number:
<input id="phone_number" name="phone" type="text" onchange="check_phone();"
<span id="phone_msg"></span>
</p>


function check_phone()
{
  var value = $('#phone_number').val();

  if (value.length != 10) {
    $('#phone_msg').show();
    $('#phone_msg').html('Incorrect␣length');
  }else {
    $('#phone_msg').hide();
  }
}
```

- si la longueur du numéro de téléphone n'est pas égale à 10 alors un message est affiché

Plan

Introduction

Les éléments abordés

- historique et caractéristiques
- les types
- les variables et les constantes
- entrée-sortie
- les opérateurs et les structures de contrôle
- les fonctions
- les bases de données

Les références

- le site de référence : `http://www.php.org`
- le livre de référence : `K. Tatroe, P. MacIntyre et R. Lerdorf, Programming PHP, édition o'reilly, 540 pages`

Eléments non abordés

Les classes et objets

Introduction

Historique

- créé par Rasmus Lerdorf pour créér dynamiquement des *home pages* et traiter des formulaires
- première release 1.0 en 1995 et la deuxième (2.0) en 1996
- PHP devient ensuite un langage de traitement pour les documents hypertextes
- analyseur réécrit par Andi Gutmans et Zeev Suraski : *Zend engine*
- version 3.0 en 1998, version 4.4 en 2005
- en 2004, changement de modèles de développement : modèle objet avec la version 5
- puis en 2009, version 5.3, en 2011, version 5.4, en 2013 version 5.5 et finalement la version 5.6 en 2014

Quelques références

- des sites web : Facebook et Wikipedia
- des CMS comme Wordpress et Drupal

Introduction

Caractéristiques

- langage faiblement typé
- orienté objet
- interprété mais génération à la volée de “bytecode” en cache
- utilisé principalement pour le développement d'applications web

Autres utilisations

- *scripting* système comme Perl, shell, ...
- écriture d'appplications avec interface graphiques (GUI - PHP-GTK)

PHP et HTML

La balise PHP

Le code PHP est placé à l'intérieur de :

- la balise `<?php ...?>`
- ou `<script language="php"> ...</script>`

Fichiers PHP

- l'extension des fichiers contenant du PHP est .php
- l'extension permet au serveur web comme Apache par exemple de les reconnaître
- le code PHP est alors interprété par le moteur PHP

Syntaxe de base

Syntaxe de base

- le langage est sensible à la "case" : un nom de variable écrit en majuscule ou en minuscule est considéré comme différent
- toutes les instructions se terminent par un point virgule → les sauts de ligne et les espaces n'ont pas d'effet (sauf dans les chaînes de caractères)
- les commentaires sont comme :
  - en C (multilignes) : /* ...*/
  - en C++ (monoligne) : // ...
  - shell/Perl (monoligne) : # ...

Exemple "Hello world"

```
<?php echo "hello world"; ?>
```

Syntaxe de base

Inclusion de code

- il est possible d'inclure un autre fichier (html ou php) dans un fichier php
- *include* :
  - inclusion du fichier → un warning apparaît si le fichier à inclure n'est pas trouvé
  - utilisé pour l'inclusion de fichiers HTML
  - permet la réutilisation de fichiers html pour plusieurs pages (par exemple une entête de page ou un pied de page)
- *require* :
  - inclusion du fichier et erreur fatale si le fichier n'est pas trouvé
  - utilisé pour inclure des fichiers de définition de fonctions php utilisées dans le script

Syntaxe de base

Inclusion de code html

```
<?php include "header.html"; ?>
...
<?php include "footer.html"; ?>
```

Inclusion de code php

```
<?php require "design.php";
header(); ?>
...
<?php footer(); ?>
```

Syntaxe de base

Littéraux

Les littéraux sont les données constantes :

- les numériques : `2015`, `0xFF` ou `3.1415`
- les chaînes de caractères : "Hello world" ou 'Hello world'
- true / false
- null

Identifiants

- le nom d'une variable commence par le symbole $ et est suivi par une lettre ou un *underscore* puis des lettres, des chiffres et l'*underscore*
- exemples : $str, $_name, $an_instance
- les noms de fonction suivent la même logique mais sans le $ devant
- rappel : le php est *case sensitive*

Syntaxe de base

Les types

- les types simples (scalaires) : entiers, réels, chaînes de caractères et booléen
- les collections : les tableaux et les objets
- les types spéciaux : ressource et `null`

Entiers

- valeur comprise entre -2^{16} et $2^{-16} - 1$
- peut être notée dans une autre base que 10 :
  - 16 (hexa) : `0x3F`
  - 8 (octal) : `0755`
  - 2 (binaire) : `0b00110111`
- possibilité de tester si une variable contient une valeur entière

```
<?php if (is_int($x) { ... } ?>
```

Syntaxe de base

Réels

- valeur comprise entre $1.7e-308$ et $1.7e+308$ (comme le type double du langage C)
- deux formats :
  - usuel (avec une virgule) : 1.3, -0.7
  - scientifique : $1.3e-7$
- **la représentation d'un réel est une approximation !**
  $\rightarrow 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1 \neq 1$

Syntaxe de base

Chaînes de caractères

- une chaîne de caractères est un ensemble de caractères
- deux types de délimiteurs : double quote et simple quote
- si la chaîne est délimitée par des simples quotes alors elle n'est pas interprétée

```
<?php
$name = "world";
echo "Hello␣$name!\n";
?>
```

- interprété : la variable $name est remplacée par sa valeur

Caractères d'échappement

- comme en C : \ n \ r \ t \ \\ "
- \ $ pour le distinguer du marqueur de début de nom de variable
- et aussi : \ { \ } \

\

Syntaxe de base

Booléens

- valeur : vrai (`true`) ou faux (`false`)
- la valeur faux peut aussi être :
  - l'entier 0 ou le réel 0.0
  - la chaîne de caractères vide ou contenant le caractère 0
  - le tableau avec aucun élément
  - la valeur `null`
- tout ce qui n'est pas faux est vrai

Test

Possibilité de tester si une variable contient une valeur booléenne :

```
<?php if (is_bool($x) { ... } ?>
```

Syntaxe de base

Les tableaux

- un tableau est un ensemble de valeurs indexées
- l'index peut être :
  - un entier ; la première position est indexée par 0
  - une chaîne de caractères ; on parle alors de tableau associatif

```
$list[0] = "Hello";
$list[1] = "world!";

$dic["Hello"] = "Bonjour";
$dic["world!"] = "le␣monde␣!";
```

Syntaxe de base

Création

- la création d'un tableau est possible grâce à la fonction `array()`
- *$list = array() ;* créé un tableau vide
- création d'un tableau avec des éléments :
  - index entier :

```
$list = array("Hello", "world!");
```

  - associatif :

```
$dic = array("Hello" => "Bonjour", "world!" => "le monde !");
```

Count

On peut connaître la taille d'un tableau grâce à la fonction *count*.

```
$n = count($list);
```

Syntaxe de base

Les variables

- inutile de déclarer les variables avant de les utiliser
- une variable définit à l'extérieur d'une fonction est considérée comme globale
- lorsqu'une variable est définie dans une fonction, celle-ci a une durée de vie limitée à l'exécution de la fonction
- pour allonger la durée de vie, deux solutions :
  - variable globale : variable définie en dehors des fonctions ou utilisation du mot-clé `global`
  - variable statique : variable définie dans une fonction → portée est limitée à la fonction **mais** lorsque la fonction est exécutée une seconde fois la variable reprend la valeur qui lui a été assigné à l'appel précédent

Syntaxe de base

Les constantes

- les constantes sont identiques aux variables mais elles peuvent être affectées qu'une seule fois
- PHP crée automatiquement certaines constantes (PHP_VERSION, par exemple)
- on peut créer manuellement des constantes grâce à la fonction *define*

```
define("BONJOUR", "Hello, World !");
echo BONJOUR;
```

Syntaxe de base

Entrée-sortie

- trois fonctions permettent de générer du texte pour la page web : *echo*, *print* et *printf*
- *echo* est plus un opérateur qu'une fonction
- *echo* prend n'importe quel nombre de paramètres, séparés par des virgules ; chaque paramètre est converti en chaîne de caractères

```
echo "Premiere chaine", 2, 3.0, "derniere chaine";
```

- la fonction *print* est identique qu'*echo* sauf qu'elle nécessite des paranthèses et qu'elle admets qu'une seule chaîne de caractères en paramètre
- *printf* est similaire à la fonction du C ; le premier paramètre est une chaîne de formatage

```
printf('%02d/%02d/%04d', $month, $day, $year);
```

Expression

Opérateurs arithmétiques

- unaire : +expr -expr
- multiplication, division et modulo : * / %
- addition et soustraction : + -
- refix : ++$var –$var
- postfix : $var++ $var–
- décalage : >> <<

Logique et relationnel

- logique : ! & | && || and or xor
- test : ? :
- égalité avec *type casting* : == !=
- égalité sans *type casting* : === !==

Expression

Chaîne de caractères

- concaténation : .

```
$a = "Hello␣"; $a = $a . "world!";
```

Affectation

- simple : =
- arithmétique : += -= *= /= %=
- logique : &= ≙ |=
- chaîne de caractères (concaténation) : . =

Les instructions de contrôle

Liste

- conditionel : switch, if/else/elseif
- itération : while, do/while, for, foreach
- branchement : break, continue
- exception : try/catch

if/else

```
if ($a > $b) {
} elseif ($a < $b) {
} else {
}
```

for

```
for ($i = 0; $i < $n; ++$i) {
}
```

for

```
$i = 0;
while ($i < $n) {
   ...
   ++$i;
}
```

Les instructions de contrôle

foreach

- parcours d'un tableau

```
foreach ($list as $current) {
  ...
}
```

- parcours d'un tableau associatif

```
foreach ($list as $key => $value) {
  ...
}
```

Les fonctions

Fonction

- une fonction peut être définie en utilisant la syntaxe suivante :

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
  ...
  return $retval;
}
```

- peut ne pas avoir de paramètres (exemple : *function foo() ...)* ; son appel se fera aussi sans paramètre (*foo()*)
- les paramètres, comme les variables, ne sont pas typés
- toute fonction doit être préalablement définie avant d'être utilisées
- si une fonction doit retourner une valeur alors la dernière instruction de la fonction sera *return*

Les fonctions

Passage des paramètres

- par défaut, les paramètres sont passés par valeur
- les paramètres sont considérés comme des variables locales de la fonction et sont affectés par les valeurs passées à la fonction
- si le paramètre est précédé du signe & alors le paramètre est passé par référence
- le paramètre par référence est un alias de la variable passée
- toute modification du paramètre dans la fonction se répercute sur la variable passée en paramètre
- un paramètre peut possèder une valeur par défaut

Les fonctions

Passage par référence

```
function add_some_extra(&$string)
{
    $string .= ' world!';
}

$str = 'Hello';
add_some_extra($str);
```

Valeur par défaut

```
function get($index = 0)
{
  ...
}

$value = get();
```

PHP et HTML

Intégration PHP / HTML

- du code PHP dans les balises HTML

```
<label>Name:<br>
<input name="<?php␣echo␣$name␣?>" type="text" />
```

- du code HTML généré par du PHP

```
echo '<table>';
for ($row = 0; $row < 10; ++$row) {
  echo '<tr>';
  for ($col = 0; $col < 10; ++$col) {
    echo '<td>', ($row * $col), '</td>';
  }
  echo '</tr>';
}
echo '</table>';
```

Remarque

- le deuxième cas est aussi réalisable en Javascript/jQuery
- pourquoi utiliser alors PHP ?

Les formulaires et PHP

Les données

- selon la méthode de soumission d'un formulaire, les données sont stockées dans des variables spéciales : *$_GET* et *$_POST*
- ce sont des tableaux associatifs
- une entrée du tableau par attribut *name* du formulaire

Les données multivaluées

- si le champ est de type multiple pour les *checkbox* ou les *select* alors l'entrée est un tableau
- le nom associé au champ doit comporte des crochets

```
<select name="languages[]" multiple>
  <option value="html">HTML</option>
  <option value="js">JavaScript</option>
  <option value="php">PHP</option>
</select>


foreach ($_GET["$languages"] as $language) {
  ...
}
```

Les bases de données et PHP

PDO

- PDO = *PHP Data Objects*
- interface légère d'accès aux bases de données
- un driver par type de base de données (mysql, postgresql, ...)
- le langage SQL est utilisé pour les requêtes
- le résultat des requêtes SQL est bufferisé

Un exemple

```
$db = new PDO("mysql:host=localhost;dbname=boutique", "toto", "password_toto");
$db->query("UPDATE␣product␣SET␣id=4␣WHERE␣name='odroid␣c3'");
```

php.ini

Ajouter la déclaration du driver dans `/etc/php5/cgi/php.ini` :
extension=mysql.so

Les bases de données et PHP

Résultat

- les requêtes *SELECT* produisent des résultats
- ces résultats sont bufferisés → des enregistrements
- ils sont parcourus via la méthode *fetch*
- un enregistrement est un tableau associatif
- chaque entrée est un champ

Un exemple

```
$statement = $db->prepare( "SELECT␣*␣FROM␣products");
$statement->execute();
while ($row = $statement->fetch()) {
  print_r($row);
}
$statement = null;
```

print_r

La méthode la plus simple pour visualiser un tableau associatif : *print_r(...)*.

Les bases de données et PHP

sqlite

- sqlite est une base de données légère, compact → un simple fichier
- disponible de manière intégrée à PHP
- une classe et des méthodes dédiées

Un exemple

- ouverture de la base de données

```
$db = new SQLiteDatabase("/home/xxx/boutique.sqlite");
```

- une requête *SELECT*

```
$result = $db->query("SELECT␣*␣FROM␣products␣WHERE␣id=1");
while ($row = $result->fetch()) {
  print_r($row);
}
```

Copyright

Auteur

Éric Ramat *ramat@lisic.univ-littoral.fr*