

HTML & CSS Objectifs

version 2.1

Objectifs

- Savoir créer et structurer une page HTML
- Connaître les principes de CSS
- Savoir séparer la présentation à l'aide de CSS
- Savoir créer un site HTML
- Savoir utiliser les sélecteurs CSS
- Connaître les principales "recettes" CSS
- Savoir créer des formulaires HTML
- Connaître les ajouts de HTML5



Chapitres

0. Objectifs
1. Introduction
2. HTML - structures de base
3. CSS - principes de base
4. HTML - les formulaires
5. CSS - recettes
6. HTML - éléments avancés
7. Créer son site
8. Conclusion

copyleft

Support de formation créé par

Franck SIMON

<http://www.franck-simon.com>



copyleft

Cette œuvre est mise à disposition sous licence
Attribution

Pas d'Utilisation Commerciale

Partage dans les Mêmes Conditions 3.0 France.

Pour voir une copie de cette licence, visitez
<http://creativecommons.org/licenses/by-nc-sa/3.0/fr/>

ou écrivez à

Creative Commons, 444 Castro Street, Suite 900,
Mountain View, California, 94041, USA.



HTML & CSS

Introduction

Internet et le web

- Internet
 - inter net
 - liens entre des réseaux hétérogènes
 - réseau de réseaux
 - les réseaux sont reliés entre-eux par le protocole TCP/IP
 - il n'existe pas d'administration internet
 - comme pour un réseau téléphonique



TCP/IP

- Couches réseau

	modèle OSI	Pile TCP/IP
7	Application	HTTP FTP DNS
6	Présentation	
5	Session	
4	Transport	TCP, UDP, SCTP
3	Réseau	IP
2	Liaison	Ethernet, Token Ring, ...
1	Physique	ADSL, RTC, ...

Protocole IP

- IP – Internet Protocol
 - protocole non fiable
 - assure l'acheminement au mieux
 - ne se préoccupe pas du contenu envoyé
 - fournit une méthode pour délivrer le contenu à destination
 - pas de garantie sur les paquets envoyés
 - corruption de données, ordre d'arrivée
 - L'adresse IP est un numéro attribué à chaque équipement connecté

Protocole TCP

- TCP - Transmission Control Protocol
 - protocole de transport fiable
 - orienté connexion
 - délivre toutes les données correctement et en séquence
- TCP (comme UDP) utilise le concept de port pour identifier l'application
 - à chaque extrémité (client / serveur) est associé un numéro de port sur 16 bits

Protocole TCP

- Fonctionnement en 3 phases
 - établissement de la connexion
 - transfert des données
 - fermeture de la connexion
- La perte d'un segment est gérée par TCP
 - mécanisme de temporisation et retransmission

Le protocole HTTP

- HTTP – HyperText Transfert Protocol
 - protocole client-serveur inventé par Tim Berners-Lee
 - avec le langage HTML et les adresses Web
 - port 80 par défaut
 - HTTPS – variante sécurisée (port 443)
 - actuellement HTTP 1.1
 - depuis janvier 1997
 - RFC 2616
 - HTTP 1.1 bis
 - clarification de la RFC 2616
 - éclatement dans 8 documents RFC 7260 à RFC 7237

Le protocole HTTP 2

- Approuvé en mai 2015
 - RFC 7540
- Ajout d'un mécanisme permettant au client de basculer vers un autre protocole
- Amélioration des performances
 - compression des en-têtes
 - chargement des pages en parallèle sur une seule connexion TCP
 - push server avec les websockets
 - mis en place par Google (SPDY)

Protocole HTTP

- Protocole sans état
 - le client envoie une requête
 - qui comporte une commande : méthode HTTP

```
Requête
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
Connection: keep-alive
DNT: 1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:12.0) Gecko/20100101 Firefox/12.0
```

- le serveur lui répond

```
Réponse
Accept-Ranges: bytes
Content-Length: 9
Content-Type: text/html
Date: Thu, 02 Aug 2012 07:54:21 GMT
Etag: "1400000001e219-9-4b3e56b08114c"
Last-Modified: Mon, 12 Dec 2011 13:50:31 GMT
Server: Apache/2.2.21 (Win32) PHP/5.2.17
```

Le protocole HTTP

- Les méthodes HTTP
 - GET : demande de ressource
 - POST : soumission de données au serveur en vue d'un traitement
 - formulaire HTML
 - OPTIONS : permet d'obtenir les méthodes supportées par le serveur
 - CONNECT : utilisation d'un proxy comme tunnel de communication
 - TRACE : demande au serveur de retourner ce qu'il a reçu
 - PUT : demande de remplacer ou d'ajouter une ressource
 - DELETE : demande la suppression d'une ressource

Protocole HTTP

- HTTP permet l'identification
 - BASIC
 - mot de passe passé en clair (base 64)
 - DIGEST
 - souvent utilisé avec un hash MD5
- Utilisation possible du mode CLIENT-CERT
 - authentification mutuelle par échange de certificat

HTML

- Utilisé dans les navigateurs HTML est un langage de description des pages web
- HTML pour Hyper Text Markup Language
 - n'est pas un langage de programmation
 - basé sur un ensemble fini d'éléments
- Le navigateur interprète le document HTML et l'affiche comme une page web
- Les extensions standards pour les documents HTML sont *htm* et *html*
 - *index.html* ou *index.htm* sont souvent les pages par défaut d'un site web

HTML

- Évolution de HTML
 - 1989 -> 1992
 - description HTML informelle
 - 1993
 - HTML 1.0 non officiel
 - langage en pleine évolution
 - NCSA MOSAIC invente la balise IMG et FORM
 - 1994
 - apports de Netscape Navigator en terme de présentation
 - début de CSS (Cascading Style Sheet)

HTML

- Évolution de HTML
 - 1995 -> 1996
 - le W3C propose un brouillon de spécification HTML
 - RFC 1866 décrivant HTML 2.0 est finalisée fin 1995
 - 1997
 - la spécification HTML 3.2 est publié le 3 Janvier 1997
 - la spécification HTML 4.0 est publiée le 18 Décembre 1997
 - variante stricte (strict) qui exclut les éléments et attributs de présentation devant être remplacés par des styles CSS
 - variante transitoire (transitional) étend la variante stricte en reprenant les éléments et attributs de présentation
 - variante frameset qui normalise les jeux de cadres

HTML

- Évolution de HTML
 - 2000 - 2006
 - le développement de HTML est officiellement abandonné par le W3C au profit de XHTML
 - création du WHATWG (Web Hypertext Application Technology Working Group) pour relancer le développement HTML face à la proposition du W3C
 - 2007 à 2012
 - le W3C relance HTML pour
 - faire évoluer HTML pour décrire la sémantique des documents
 - parvenir à un langage extensible via XML
 - enrichir les IHM : menus, champs associés à des types, ...
 - les travaux du WHATWG sont adoptés par le W3C comme point de départ de la spécification HTML 5

HTML

- Juillet 2012
 - Ian Hickson (fondateur du WHATWG) prend ses distances avec le W3C
 - désaccord sur la décision du W3C de découper HTML 5 en sous-spécifications
 - API 2D canvas, gestion des événements, ...
 - le WHATWG fait évoluer sa branche de HTML 5
 - baptisée "Living Standard"
 - veut imposer une évolution de la spécification en phase avec le marché
 - en moyenne il y a une mise à jour toutes les 6 semaines de Chrome et Firefox

HTML

- HTML 4.01 strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
```

- HTML 4.01 transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
```

- HTML 4.01 frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
<html>
```

HTML

- XHTML 1.0 strict

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

- XHTML 1.0 transitional

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

- XHTML 1.0 frameset

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

- XHTML 1.1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

HTML

- HTML 5
 - Simplification de la syntaxe

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

Technologies web

- Pour créer une application web il est nécessaire de mettre en œuvre un ensemble de technologies
 - HTTP et TCP/IP
 - HTML
 - différentes versions
 - représentation par le navigateur sous forme de DOM
 - CSS
 - JavaScript
 - et autres langages
 - XML, XSL, XUL, Java, Flex, ...

Technologies web

- Difficultés des applications web
 - créer une application homogène avec des technologies hétérogènes
 - tenir compte des différentes versions de navigateur
 - évolution rapide des demandes des utilisateurs
 - en fonctionnalités supplémentaires
 - en fréquentation du site => montée en charge
 - en fluidité d'utilisation
 - IHM limitée
 - nécessité d'un framework

HTML & CSS

Structures de base HTML

HTML - page minimale

- Structure d'un document HTML

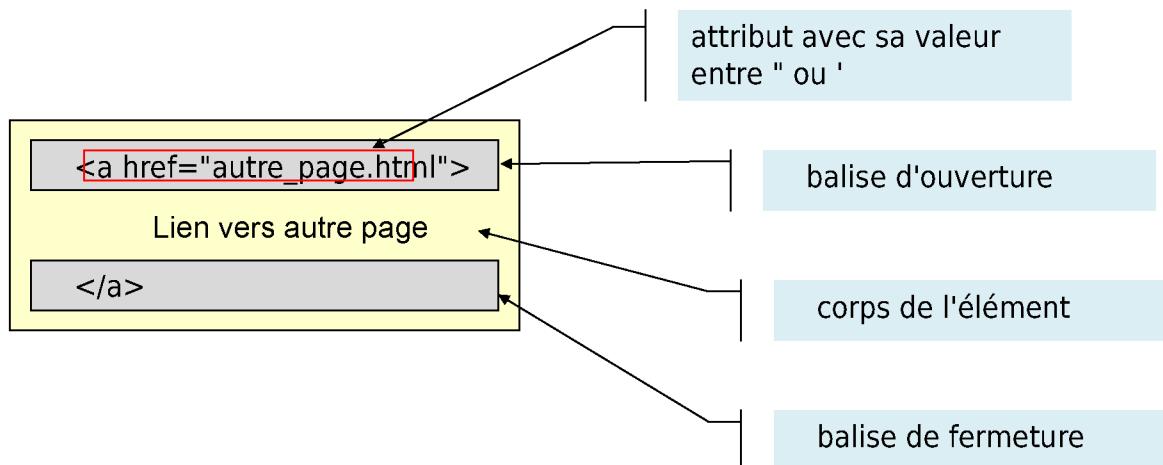
```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title>page minimale</title>
  </head>
  <body>
    Hello, World
  </body>
</html>
```

- `<!DOCTYPE ...>` grammaire utilisée
- `<html>...</html>` décrit la page web
- `<head>...</head>` contient des informations pour le document - les métadonnées (metadata)
- `<body>...</body>` est le contenu visible de la page

Élément HTML

- Un tag élément HTML est constitué
 - d'une balise d'ouverture (start tag)
 - qui peut contenir des attributs
 - d'un corps d'élément
 - qui peut être vide, contenir du texte et/ou d'autre éléments
 - d'une balise de fermeture (end tag)
- Une balise est entourée par les caractères < et >
- Normalement toute balise ouverte devrait-être fermée
 - il existe une écriture simplifiée pour les balises sans corps
 - <p />

Élément HTML



Élément HTML

- Le corps d'un élément peut contenir d'autres éléments
 - les éléments doivent être imbriqués



```
<div>
  <h2>Hello, World</h2>
</div>
```



```
<div>
  <h2>Hello, World
</div>
</h2>
```

Élément HTML

Commentaires

- ne sont pas affichés
- peuvent-être sur plusieurs lignes
- ne peuvent pas être imbriqués
- commence par `<!--` et fini par `-->`



```
<!-- <h2>Hello, World</h2> -->
<!-- commentaire
    sur plusieurs lignes -->
```



```
<!--
  <!-- <h2>Hello, World</h2> -->
  <!-- commentaire
      sur plusieurs lignes -->
-->
```

Section <head>

- <title> : titre affiché dans l'onglet du navigateur
- <meta> : métadonnées
 - fournissent des informations supplémentaires
 - utilisées par les navigateurs, serveurs et proxy
 - cf. slide suivant
- <script> : scripts JavaScript
- <link> : déclaration d'une ressource liée au document
 - le plus souvent une feuille CSS

Section <head>

- Exemple

```
...
<head>
  <title>Formulaire</title>
  <meta charset="utf-8">
  <link rel="stylesheet" type="text/css" href="form.css" />
  <script type="text/javascript" src="form.js"></script>
</head>
...
```

Section <body>

- Contient le contenu qui sera affiché par le navigateur
 - le navigateur analyse les éléments HTML (et les CSS) pour calculer le rendu
- Les éléments peuvent être de type
 - texte
 - regroupement de contenu
 - définition de section
 - tableau
 - formulaire
 - déclaration d'objet

Éléments de type *block* et *inline*

- Un élément de type *block* démarre sur une nouvelle ligne
 - div, address, article, aside, footer, ...
- Les éléments de type *texte* sont *inline*
- Généralement un élément de type *block* regroupe une structure plus large qu'un élément de type *inline*
 - dans les documentations
 - block <=> flow content
 - inline <=> phrasing content
- cf. https://developer.mozilla.org/en-US/docs/Web/HTML/Block-level_elements

Éléments texte

- <a> : création d'un lien hypertext
-
 : retour à la ligne
- <code> : exemple de code source
- : élément générique
 - pas de sémantique particulière
- : texte important
- <var> : variable de code source
- cf. une liste exhaustive sur www.w3schools.com

Éléments texte

- Exemple

```
...
<body>
    <strong>Exemple de code</strong><br/>
    <code>
        Calculatrice calc = new Calculatrice();
    </code><br/>
    <a href="code.html">Récuperer le source</a>
</body>
...
```

Exemple de code

Calculatrice calc = new Calculatrice();
[Récuperer le source](#)

Éléments de regroupement

- <div> : élément générique
- : listes
- : item d'une liste , ou <menu>
- <pre> : préserve la mise en page
- <p> : paragraphe
- <hr/> : rupture de paragraphe
- cf. documentation pour liste exhaustive

Éléments de regroupement

- Exemple

```
...
<pre>
public static void main(String[] args)
{
    int a=3,b=6;

    if(args.length==2)
    {
        a = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);
    }
    Calculatrice calc = new Calculatrice();
    System.out.println(a" + "b" = "+calc.additionner(a, b));
}
</pre>
...
```

Exemple de code

```
public static void main(String[] args)
{
    int a=3,b=6;

    if(args.length==2)
    {
        a = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);
    }
    Calculatrice calc = new Calculatrice();
    System.out.println(a" + "b" = "+calc.additionner(a, b));
}
```

Éléments de regroupement

- Exemple

```
...  
    <ul>  
        <li>item 1</li>  
        <li>item 2</li>  
        <li>item 3</li>  
    </ul>  
  
    <ol>  
        <li>item 1</li>  
        <li>item 2</li>  
        <li>item 3</li>  
    </ol>  
...
```

- item 1
- item 2
- item 3

1. item 1
2. item 2
3. item 3

Définition de section

- `<h1> ... <h6>` : titre
- nouveautés HTML 5
 - `<header>, <footer>, <section>, <aside>, <address>, <details>, <nav>, ...`
 - permet de donner une sémantique à une section
 - ne définit pas une mise en page par défaut
- cf. documentation pour liste exhaustive

Définition de section

- Exemple

```
...  
<h1>Titre H1</h1>  
<h2>Titre H2</h2>  
<h3>Titre H3</h3>  
<h4>Titre H4</h4>  
<h5>Titre H5</h5>  
<h6>Titre H6</h6>  
...
```

Titre H1

Titre H2

Titre H3

Titre H4

Titre H5

Titre H6

Tableau

- Les tableaux ne doivent être utilisés que pour la présentation de données
 - pas pour faire de la mise en page
 - utiliser les sections, `<div>`, ``, ... et les styles
 - `<table>` : tableau
 - `<tr>` : ligne de cellules
 - `<th>` : cellule de type en-tête
 - `<td>` : cellule de type contenu
 - cf. documentation pour liste exhaustive

Tableau

- Exemple

```
...


| Légende |   |   |
|---------|---|---|
| titre   | a | a |
|         | a | a |
|         | a | a |


...

```

Légende		a	a
titre	a	a	a
	a	a	a

Formulaire

- Seront vus plus en détail dans un prochain chapitre
- **<form>** : **encapsule les champs du formulaire**
 - contient l'URL de la ressource devant traiter les données du formulaire
- **<input>** : **champ de saisie sur une ligne**
 - de type text, password, date, ...
 - ou permettant une interaction avec l'utilisateur
 - de type radio, checkbox, submit, reset, ...
- **<textarea>** : **champ de saisie sur plusieurs lignes**

Formulaire

- Exemple

```
...
<form action="http://localhost/PHP/echo.php" method="post">
    Nom :<input type="text" name="nom" value="Toto"/><br/>
    Password :<input type="password" name="prenom" /><br/>
    Email :<input type="text" name="email" /><br/>
    <input type="submit" value="Envoyer" /><input type="reset" value="RAZ" />
</form>
...
```

A screenshot of a web browser window displaying a simple HTML form. The form consists of three text input fields: 'Nom' with value 'Toto', 'Password' (empty), and 'Email' (empty). Below the inputs are two buttons: a blue 'Envoyer' button and a grey 'RAZ' button.

Objets embarqués

- Se sont des objets embarqués dans la page
 - images, son, audio, zone de dessin, video, plugin, ...
- : déclare une image
- <iframe> : document HTML encapsulé dans la page principale
- <audio>, <video>, <track> : gestion du multimédia
- ...

Objets embarqués

- Exemple

```
...
<body>
 <br/>
<audio controls>
  <source src="tnt.mp3" type="audio/mp3" />
  Votre navigateur ne supporte pas le lecteur audio HTML 5.
</audio>
</body>
..
```



Attributs des éléments

- Plusieurs attributs peuvent être positionnés pour un même élément
 - toujours dans la balise d'ouverture
 - un attribut est unique
 - un certains nombre d'attributs sont communs à tous les éléments de rendu

```
<a href="navigation.jsp?cde=1" id="nav_1" class="navigation_anchor">Acheter</a>
```

Attributs des éléments

- Attributs booléens

- le fait que l'attribut soit présent affecte le comportement de l'élément

- 3 syntaxes

```
<input disabled/>
<input disabled="" />
<input disabled="disabled" />
```

- Attribut personnalisé (depuis HTML 5)

- doivent-être préfixés par data-

```
<a href="navigation.jsp?cde=1" id="nav_1" class="navigation_anchor"
    data-id="1" data-pays="canada">Acheter</a>
```

Attributs communs

- Les plus utilisés

- `id` : identifiant unique de l'élément dans la page
 - permet la récupération de l'élément en JavaScript par la méthode `document.getElementById(...)`
- `class` : référence d'une classe de style CSS
 - le style est défini dans la feuille de style

```
<div class="rouge"></div>
```

- `style` : application directe d'un style à l'élément
 - le style est directement décrit

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

Attributs communs

- `accesskey` : raccourci clavier
 - permet de donner le focus par Alt+lettre
- `contenteditable` : rend un élément éditable
- `draggable`, `dropzone` : support du drag and drop
- `spellcheck` : active le correcteur orthographique
- `title` : tool tip
- `tabindex` : contrôle de la navigation entre les champs avec la touche Tab
- ...

Entités HTML

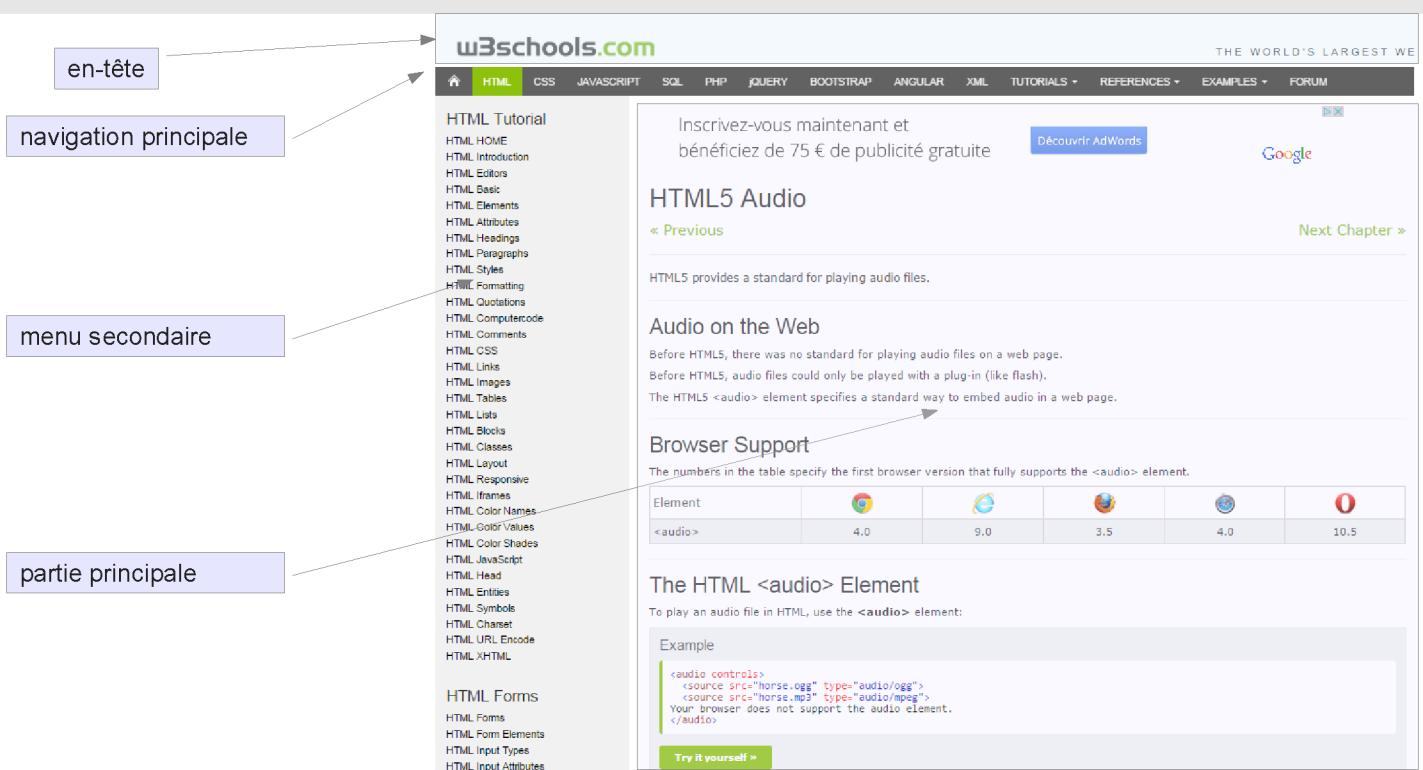
- Certains caractères ont une signification particulière
 - signes `<`, `>`, `&`, ...
 - ils peuvent être remplacé par des entités
 - une entité débute par `&` et finit par `;`

	Caractère	Entité	Valeur
exemples d'entités	<code><</code>	<code>&lt;</code>	<code>&#60;</code>
	<code>></code>	<code>&gt;</code>	<code>&#62;</code>
	<code>&</code>	<code>&amp;</code>	<code>&#30;</code>
	<code>€</code>	<code>&euro;</code>	<code>&#8364;</code>
	<code>£</code>	<code>&pound;</code>	<code>&#163;</code>
	<code>©</code>	<code>&copy;</code>	<code>&#169;</code>
	<code>®</code>	<code>&reg;</code>	<code>&#174;</code>

Structure d'une page

- En général une page comporte plusieurs parties
 - en-tête
 - contenant un bandeau, une image, un logo, ...
 - une partie de navigation
 - contenant un menu composé de liens, boutons, ...
 - positionné en haut ou sur le côté
 - une partie principale
 - c'est cette partie qui est mise à jour
 - un pied de page
 - adresse, copyright, mentions légales

Structure d'une page



Structure d'une page

- Il faut donc créer une structure d'éléments HTML qui permet le rendu visuel de la structure de page imaginée
- Il faut aussi maintenir l'état du menu sélectionné par rapport au contenu visualisé
 - en JavaScript
 - ou code côté serveur
- Les éléments structurant la page ont beaucoup évolués
 - attention aux exemples trouvés sur internet
 - se référer aux dernières bonnes pratiques

Structure d'une page

- La mise en forme des pages HTML utilisaient
 - des tableaux `<table>`
 - des frames `<frame>`
 - inconvénients
 - non respect de l'accessibilité des sites
 - les machines braille ne peuvent pas en déduire une structure
 - la mise en page était effectuée par les attributs des éléments
 - le rendu et le contenu sont mélangés
 - problème de maintenance du site
 - pas de sémantique associée aux éléments
- NE PAS FAIRE

Structure d'une page

- Petit à petit la structure des pages a évoluée
 - HTML + CSS [+ JavaScript] dans des fichiers séparés
 - meilleure maintenance, cohérence et maintenabilité
 - utilisation des `<div>` pour structurer la partie HTML
 - utilisation des attributs `id`, `class` pour les styles

Structure d'une page

- Exemple sans feuille de style

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
  "http://www.w3.org/TR/html4/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>structure_base_4</title>
<meta name="author" content="franck" />
</head>
<body>
<div id="header">en-tête de la page</div>
<div id="nav">Menu de navigation entre les pages</div>
<div id="main">partie principale de la page</div>
<div id="footer">pied de page</div>
</body>
</html>
```

en-tête de la page
Menu de navigation entre les pages
partie principale de la page
pied de page

Structure d'une page

- Exemple avec feuille de style
 - la structure précédente est restée la même, mais avec ajout d'une feuille de style

```
...  
<head>  
...  
    <link rel="stylesheet" href="strcuture_4.css" />  
</head>  
...
```



Structure d'une page

- Exemple
 - feuille de style de la page précédente

```
#header{  
background-color: #FFA07A;  
width: 50%;  
height: 50px;  
margin: auto;  
text-align: center;  
font-family: "Arial";  
font-size: xx-large;  
font-weight: bolder;  
}  
  
#nav{  
background-color: #FFDAB9;  
width: 50%;  
height: 50px;  
margin: auto;  
}  
  
#main{  
background-color: #DCDCDC;  
width: 50%;  
height: 600px;  
margin: auto;  
}  
  
#footer{  
background-color: #ADFF2F;  
width: 50%;  
height: 50px;  
margin: auto;
```

Structure d'une page

- HTML 5 ajoute des éléments sémantiques pour structurer la page
 - <header>, <nav>, <main>, <article>, <footer>, <aside>, ...
 - vérifiez le support de ces balises
 - par exemple <main> n'est pas supporté par IE
 - les navigateurs ne font pas de placement par défaut
 - il faut donc associer des styles pour effectuer le rendu

Structure d'une page

- Exemple HTML 5 sans CSS

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="utf-8">
  <title>structure_base_5</title>
</head>
<body>
  <div>
    <header>
      <h1>EN TETE</h1>
    </header>
    <nav>
      <a href="/">Home</a> <a href="/contact">Contact</a>
    </nav>
    <div id="main">
    </div>
    <footer>&copy; Copyright by franck</footer>
  </div>
</body>
</html>
```

EN TETE

[Home](#) [Contact](#)
© Copyright by franck

Structure d'une page

- Exemple HTML 5 avec CSS

The diagram illustrates the structure of a web page. At the top is a header section labeled "EN TETE" in orange. Below it is a navigation bar with links "Home" and "Contact". The main content area is shown as a large gray rectangle. At the bottom is a footer section labeled "© Copyright by franck" in a green bar.

```
...  
<head>  
...  
  <link rel="stylesheet" href="structure_5.css" />  
</head>  
...
```

Lien hypertexte

- Balise `<a>`
 - par défaut
 - un lien non visité est sous-ligné et bleu
 - un lien visité est sous-ligné et pourpre
 - un lien actif est sous-ligné et rouge
 - Permet de naviguer
 - d'un document à l'autre
 - le nouveau document peut remplacer l'ancien ou être afficher dans un nouvel onglet (ou nouvelle fenêtre)
 - au sein d'un même document

Lien hypertexte

- attribut href
 - cible à atteindre
 - URL et/ou fragment d'URL (commence par #)
 - href="autre-page.html"
 - href="http://www.w3schools.com"
 - href="#label1" **cible une ancre passive**
 - href="#top" **renvoie vers le haut de la page**
 - **non sécurisé :**
 - href="mailto://toto@titi.net"
 - href="file:///exemple.txt"

Lien hypertexte

- attribut target
 - spécifie où sera ouvert le document
 - si absent, dans le cadre (frame) courant
 - **valeurs :**
 - _blank : nouvelle fenêtre
 - _self : dans le frame courant
 - _parent : dans le frame parent
 - _top : dans la fenêtre en cours
 - *framename* : dans le frame nommé *framename*

iframe

- Permet de créer un cadre interne et d'y injecter un document
 - un document HTML est intégré dans le document courant
 - les contextes sont indépendants
 - historique, session

```
<iframe src="http://www.w3schools.com"></iframe>
```



HTML & CSS

Principes de base des CSS

Introduction

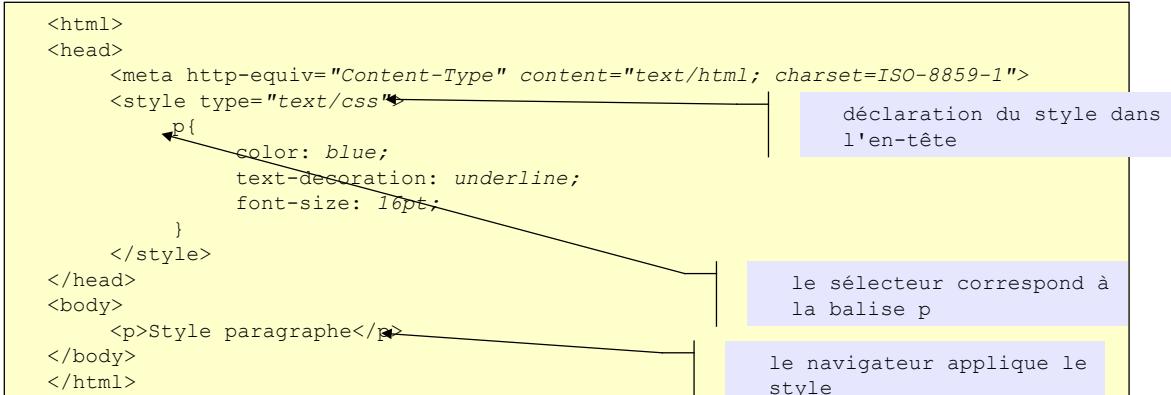
- Les styles CSS (Cascading Style Sheet) permettent de séparer la forme et le contenu
 - avant l'utilisation des styles CSS la mise en forme des pages HTML était effectuée grâce aux balises HTML de présentation
- Les intérêts sont nombreux :
 - mutualisation des style de l'ensemble d'un site
 - chargement unique des feuilles de style par mise en cache par le navigateur
 - adaptation au média

Mise en place

- Syntaxe de base

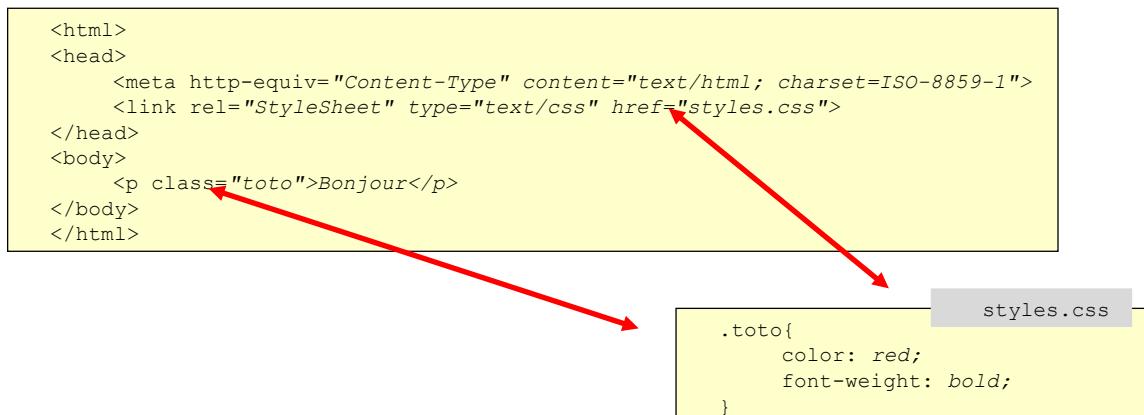
- sélecteur {propriété:valeur; propriété:valeur}
 - exemple : H2{color:blue; font: 18px }

- Mise en place des styles en interne



Mise en place

- Feuille de style externe



- Style intra-ligne

```
<p style="color: blue;font-style: italic">Autre style</p>
```

Règles

- Une règle est composé d'un sélecteur et d'un bloc de déclaration

```
sélecteur {propriété:valeur; propriété:valeur}
```

- Le sélecteur identifie le style

- presque tous les éléments HTML sont des sélecteurs potentiels
- des noms de sélecteurs peuvent être créés
 - le nom est précédé par un point

- Des commentaires peuvent être insérés

- entre /* et */

Règles

- Sélecteur classe

- il est possible d'ajouter une classe à un sélecteur
- il est aussi possible d'utiliser # pour spécifier l'id d'un élément

```
<html>
<head>
    <style type="text/css">
        p.cadre {
            border: 1px;
            border-style: solid;
            border-color: blue;
        }
    </style>
</head>
<body>
    <p>Texte non encadré</p>
    <p class="cadre">Texte encadré</p>
</body>
</html>
```

Règles

- Les pseudo-classes sont des classes spécifiques permettant d'obtenir des effets sur des éléments sans passer par l'attribut `class`
 - `:first-child`
 - `:link, :visited`
 - `:hover, :active, :focus`
 - `:lang`

```
<html>
<head>
    <style type="text/css">
        a:VISITED {color: blue;}
        a:LINK { color:blue;}
        a:ACTIVE { color:blue;}
        a:HOVER { color:red;}
    </style>
</head>
<body>
    <a href="#">Lien</a>
</body>
</html>
```

Règles

- Les pseudo-éléments permettent d'agir sur du contenu impossible à identifier en HTML
 - `::first-line`
 - `::first-letter`
 - `::after, ::before`
- Règles spéciales
 - `@import`
 - `@media`
 - `@page`

Les sélecteurs

Sélecteur	Description	CSS
*	tous les éléments	2
type	éléments du type spécifié	1
.class	éléments de la classe spécifiée	1
#id	élément avec l'id spécifié	1
[attr]	éléments ayant l'attribut attr, quelque soit la valeur de l'attribut	2
[attr="val"]	éléments dont l'attribut attr vaut val	2
[attr^="val"]	éléments dont l'attribut attr vaut un valeur commençant par val	3
[attr\$="val"]	éléments dont l'attribut attr vaut un valeur finissant par val	3
[attr~="val"]	éléments dont la liste des valeurs de l'attribut attr contient val	3
[attr*="val"]	éléments dont la valeur l'attribut attr contient la chaîne val	3
[attr]="val"]	éléments dont la liste des valeurs de l'attribut attr commence par val	3
selector1, selector2	union de plusieurs sélecteurs	1
selector1 selector2	éléments qui correspondent à selector2 et dont un ancêtre correspond à selector1	1

Les sélecteurs

Sélecteur	Description	CSS
select1 > select2	éléments qui correspondent à select2 et dont le parent correspond à select1	2
select1 + select2	éléments qui correspondent à select2 et qui sont immédiatement suivis par un élément correspondant à select1	2
select1 ~ select2	éléments qui correspond à select2 et qui sont suivis par un élément correspondant à select1	3
::first-line	sélectionne la première ligne d'un bloc de texte	1
::first-letter	sélectionne la première lettre d'un bloc de texte	1
:before	sélectionne avant l'élément	2
:after	sélectionne après l'élément	2
:root	sélectionne l'élément racine du document	3
:first-child	sélectionne le premier élément enfant	2
:last-child	sélectionne le dernier élément enfant	3
:only-child	sélectionne les éléments qui sont les seuls enfants de leur parents	3

Les sélecteurs

Sélecteurs	Description	CSS
:only-of-type	sélection des éléments qui sont seul de leur type dans leur conteneur	3
:nth-child(n)	sélection des nième enfants	3
:nth-last-child(n)	sélection des nième enfants en partant du dernier	3
:nth-of-type(n)	sélection des nième enfants de leur type	3
:nth-last-of-type(n)	sélection des nième enfants de leur type en partant de du dernier	3
:enabled	sélection des éléments qui sont activés	3
:disabled	sélection des éléments qui sont désactivés	3
:checked	sélection des éléments qui sont sélectionnés	3
:default	sélection des éléments par défaut	3
:valid	sélection des input dont les règles de saisies sont valides	3
:invalid	sélection des input dont les règles de saisies sont pas valides	3
:in-range	sélection des input qui sont dans les bornes spécifiées	3
:out-of-range	sélection des input qui ne sont pas dans les bornes spécifiées	3

Les sélecteurs

Sélecteurs	Description	CSS
:required	sélection des input qui sont obligatoires	3
:optional	sélection des input qui sont optionnels	3
:link	sélection des éléments <a>	1
:visited	sélection des éléments <a> visités	1
:hover	sélection des éléments survolés par la souris	2
:active	sélection des éléments qui sont activés par l'utilisateur (quand un bouton est cliqué par exemple)	2
:focus	sélection de l'élément qui possède le focus	2
:not(selector)	négation d'une sélection (sélectionne les éléments qui ne sont pas sélectionnés par selector)	3
:empty	sélection des éléments ne possédant pas de fils	3
:lang(language)	sélection des éléments dont l'attribut lang vaut language	2
:target	sélection de l'élément référencé par l'URL	3

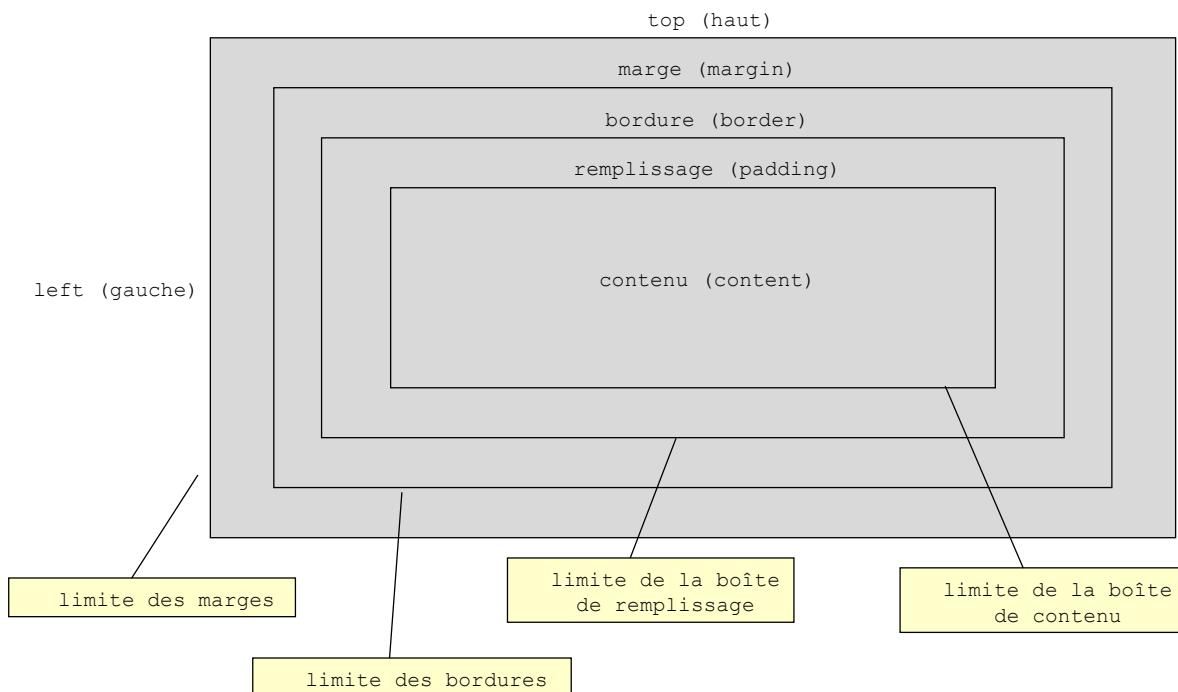
Éléments blocs et lignes

- On distingue deux types d'éléments
 - les éléments blocs comme `<h1>`, `<div>`, ``, ...
 - après la création d'un bloc il y a un retour à la ligne
 - les éléments en ligne : `<a>`, ``, ``
 - après la création de l'élément il n'y a pas de retour à la ligne
- Certaines propriétés de style ne sont applicables que sur un type d'élément
 - `vertical-align` s'applique sur les éléments en ligne

Boîte

- Une boîte (box) permet de définir la surface sur laquelle sont appliquées des propriétés
 - le contenu des éléments d'un document est inséré dans une boîte
 - les boîtes peuvent être imbriquées
- Chaque boîte est composée de plusieurs rectangles ayant des noms et des rôles spécifiques
 - les marges (margin)
 - les bordures (border)
 - la boîte de remplissage (padding)
 - la boîte de contenu (content)

Boîte



Unités de mesure

- Une unité de mesure est composée d'un nombre suivi d'une abréviation indiquant l'unité
- Unités de longueurs relatives
 - `em` : relatif à la taille de caractère employé dans l'élément parent (`1.2em` vaut 120%)
 - `ex` : relatif à la hauteur du caractère employé dans l'élément parent
 - `px` : relatif à la résolution du support visuel

Unités de mesure

- Unités de longueurs absolues
 - `in` : pouce (inch)
 - `cm` : centimètre
 - `mm` : millimètre
 - `pt` : point ($1\text{pt} = 1/72\text{in}$)
 - `pc` : pica ($1\text{pc} = 12\text{pt}$)
- Les pourcentages (%) sont relatifs à d'autre valeurs

Couleurs

- Une couleur peut être définie par
 - un nom : aqua, black, fuchsia, gray, green, lime, ...
 - un code couleur RGB : `#RRGGBB`
 - RR, GG, BB sont exprimés en hexa de 0 à FF
 - fonction `rgb`
 - `rgb(r, g, b)` où r, g et b sont un nombre entre 0 et 255
 - `rgb(r%, g%, b%)` où r%, g% et b% sont le pourcentage de chaque couleur
 - autres fonctions `rgba()`, `hsl()`, `hsla()`
 - cf. documentation

Propriétés

- La plupart des propriétés de style sont normalisées
- Certaines propriétés sont propres à un navigateur particulier
 - elles possèdent un préfixe
 - Chrome / Safari : -webkit-
 - Opera : -o-
 - Firefox : -moz-
 - Internet Explorer : -ie-

Propriétés

- Les propriétés sont composées
 - d'un nom principal, ex. : border
 - éventuellement de sous propriétés, séparés par un trait d'union -
 - border-bottom, border-bottom-color
- Des valeurs sont affectées aux propriétés
 - valeur ou résultat de fonction, ex. `rgb(...)`
 - plusieurs valeurs peuvent être affectées à une propriété

```
border : solid black 1px;
```

 - border-bottom-color : red;

Principales propriétés des boîtes

- Gestion de la taille
 - width, min-width, max-width
 - height, min-height, max-height
- Gestion des remplissages et marges
 - margin, padding
 - et leurs sous-propriétés : bottom, left, right, top
- Autres
 - visibility, float, display
 - float permet de gérer les décalages à droite et gauche par rapport au conteneur

Principales propriétés de placement (layout)

- position : positionnement
 - par défaut le navigateur ajoute les les éléments les un après les autres dans un flux
 - valeur possibles
 - static : rendu par défaut
 - relative : l'élément est positionné relativement à sa position normale
 - absolute : l'élément est positionné relativement à son premier ancêtre qui à une position autre que static
 - fixed : le positionnement est fixé relativement à la fenêtre du navigateur

Principales propriétés de placement (layout)

- Gestion des offsets
 - right, top, left, bottom
- Gestion des la profondeur
 - z-index
- Gestion des tableaux
 - column et toutes ses propriétés filles

Propriété background

- Gestion du fond de la boîte
- Propriétés secondaires
 - color : couleur du fond
 - image : image
 - size : taille de l'image
 - repeat : répétition de l'image
 - attachment, clip, position, origin : gestion fine du positionnement de l'image, du scroll de zone, ...

Propriété border

- Gestion de la bordure
- Propriétés secondaires
 - bottom, left, right, top
 - propriétés filles : color, style, width, left-radius, right-radius, ...
 - image
 - propriétés filles : repeat, source, width, outset, slice
 - color
 - radius

Propriété du texte

- Police de caractères
 - font et ses sous-propriétés
 - family, size, style, weight, variant
- Gestion du texte
 - text et ses sous-propriétés
 - align, decoration, indent, justify, spacing, transform, shadow
- Autres
 - letter-spacing, word-spacing
 - line-height, witespace

Propriétés d'animation

- Gestion des animations
 - animation et ses sous-propriétés
 - delay, direction, duration, timing-function,...
- Gestion des transformations
 - transform, transform-origin
- Gestion des transitions
 - transition est ses sous-propriétés
 - delay, duration, property, timing-function

Résolution des styles

- Le navigateur possède des styles par défaut
 - user agent style
- L'utilisateur du navigateur peut définir ses propres styles
 - Firefox : option → Contenu
- Le développeur créer ses styles

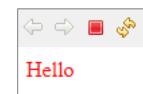
Résolution des styles

- Le navigateur va rechercher un style dans l'ordre suivant :
 - style définit en ligne - attribut `style` de la balise
 - style embarqué - élément `<style>` dans la page HTML
 - style externe - feuille de style liée par `<link>`
 - style définit par l'utilisateur
 - style du navigateur

Résolution des styles

- Exemple
 - le style inline prend le pas sur le style global

```
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>css-resolution</title>
    <style type="text/css">
        div{
            color : blue;
        }
    </style>
</head>
<body>
    <div style="color: red">Hello</div>
</body>
</html>
```



Résolution des styles

- Exemple
 - ici la résolution est forcée via le marqueur !important

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>css-resolution</title>
  <style type="text/css">
    div{
      color : blue !important;
    }
  </style>
</head>
<body>
  <div style="color: red">Hello</div>
</body>
</html>
```



Résolution des styles

- Si plusieurs styles sont applicables à un élément
 - le navigateur prend en compte
 - l'identifiant du sélecteur de style
 - les pseudo classes dans le sélecteur
 - les noms des éléments et des pseudo éléments
- Le style d'une balise enfant est hérité du style de sa balise parent
 - les styles liés à l'apparence sont hérités
 - couleur, police de caractères, ...
 - les styles liés au positionnement ne sont pas hérités

Résolution des styles

- L'héritage d'un style peut être forcé en attribuant la valeur `inherit` à une propriété

```
...
<style type="text/css">
  p{
    color:white;
    background:grey;
    border: solid black;
  }
  span{border: inherit;}
</style>
</head>
<body>
  <p>Hello, <span>world</span></p>
</body>
...

```

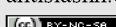


HTML & CSS

Les formulaires HTML

Les formulaires

- Mécanisme HTML pour envoyer des données vers le serveur
 - il faut donc un programme côté serveur qui récupère les données envoyées afin de les traiter
- HTML 5 a largement amélioré les caractéristiques des formulaires
 - ajout de types de données
 - ajout des validations



Formulaire de base

```
<body>
  <form method="post" action="http://localhost:8080/echo/">
    Votre nom : <input name="nom"/>
    <button>Identification</button>
  </form>
</body>
```

Votre nom : Identification

- Un formulaire est composé au minimum

- d'une balise `<form>` qui encapsule le formulaire
- d'un champ de saisie - ici l'élément `<input>`
- d'un bouton d'envoi - ici l'élément `<button>`
 - pourrait aussi être un élément :
`<input type="submit" value="Identification" />`

Élément `<form>`

- Attributs

- `action` : traitement invoqué côté serveur
- `method` : méthode d'envoi des données du formulaire
 - `get` : envoi en mode GET - dans l'URL
 - `post` : envoi en mode POST - dans le corps
 - indispensable pour l'envoi de fichiers
- `enctype` : encodage de l'envoi du formulaire
 - `application/x-www-form-urlencoded` : par défaut
 - ne permet pas l'envoi de fichiers
 - `multipart/form-data` : pour l'envoi de fichiers
 - `text/plain` : dépend des navigateurs

Élément <form>

- Attributs (suite)

- name : nom du formulaire (différent de l'`id`)
- accept-charset : encodage utilisé pour la soumission du formulaire
 - valeur `UNKNOWN` par défaut, l'encodage du document est alors utilisé
- target : dans quelle fenêtre la réponse doit s'afficher
 - `_blank`, `_parent`, `_self` (par défaut), `-top`, `<frameName>`

Élément <form>

- Attributs (suite)

- ajoutés avec HTML 5 :

- `novalidate` : le formulaire ne doit pas être validé à la soumission
- `autocomplete` : la saisie des champs peut-être auto-complétée ou non
 - valeurs : `on`, `off`

Élément <input>

- Attributs principaux

- name : nom du champ
 - c'est avec ce nom que la valeur pourra être récupérée côté serveur
- disabled : interdit la saisie dans le champ
- type : type de champ de saisie
 - sera vu plus loin
 - text (par défaut), password, ...

Élément <button>

- Attribut type

- submit : envoi du formulaire (par défaut)
 - remplace `<input type="submit" value="Identification" />`
- reset : reset du formulaire
 - remplace `<input type="reset" value="Annuler" />`
- button : bouton générique
 - remplace `<input type="button" value="Cliquez" />`

Élément <button>

- D'autres attributs permettent de redéfinir le comportement du formulaire
 - formaction : redéfini action du formulaire
 - formenctype : redéfini enctype du formulaire
 - formmethod : redéfini method du formulaire
 - formtarget : redéfini target du formulaire
 - formnovalidate : redéfini novalidate du formulaire

Récupérer les données du formulaire

- Cela nécessite un code côté serveur
 - exemple JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core_rt" %>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ECHO</title>
        <style type="text/css">
            tr:nth-child(even){
                background: #CCC;
            }
            tr:nth-child(odd){
                background: #FFF;
            }
        </style>
    </head>
    <body>
        <h1>Données du formulaire</h1>
        <table>
            <tr><th>Keys</th><th>Values</th></tr>
            <c:forEach items="${param }" var="p">
                <tr><td>${p.key }</td><td>${p.value }</td></tr>
            </c:forEach>
        </table>
    </body>
</html>
```

Données du formulaire

Keys	Values
name	Franck

Regroupement de champs

- <fieldset> permet de regrouper les champs d'un formulaire
 - ajoute un cadre autour des champs regroupés
 - l'élément <legend> permet d'ajouter un titre au regroupement

```
<form>
  <fieldset>
    <legend>Identifiez-vous</legend>
    Identifiant :
    <input type="text" id="user" name="user"/> *
    ...
  </fieldset>
</form>
```

Identifiant : *

Mot de passe : *

S'identifier

Identifiez-vous

Identifiant : *

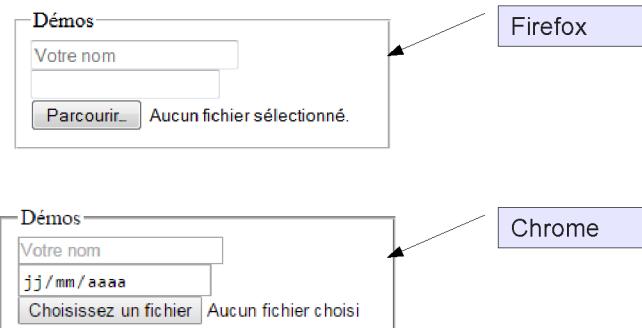
Mot de passe : *

S'identifier

Élément <input>

- L'attribut `type` permet de préciser le type de saisie
 - le rendu peut-être différent selon le type
 - le rendu peut aussi varier selon le navigateur
 - des attributs complémentaires sont spécifiés suivant le type

```
<form>
  <fieldset>
    <legend>Démos</legend>
    <input type="text" placeholder="Votre nom"/>
    <input type="date" />
    <input type="file" />
  </fieldset>
</form>
```



Association des labels

- L'élément `<label>` associe un label à un champ de saisie
 - attribut `for` : correspond à l'`id` du champ auquel le label est associé

```
<form>
  <fieldset>
    <legend>Identifiez-vous</legend>
    <label for="user">Identifiant : </label><input type="text" id="user" name="user"/><br/>
    <label for="pswd">Mot de passe : </label><input type="password" id="pswd"
                                                   name="pswd"/><br/>
    <input type="submit" value="S'identifier" id="submit" onclick="return checkForm()"/>
  </fieldset>
</form>
```

<input> de type text

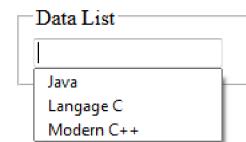
- Quelques attributs supplémentaires
 - `list` : `id` de la `<datalist>`
 - `maxLength` : nombre de caractères maximal
 - `pattern` : expression régulière à appliquer
 - `placeholder` : renseigne l'utilisateur sur la saisie attendue
 - `readonly` : en affichage seulement
 - `required` : obligatoire
 - `size` : largeur du champ en caractères
 - `value` : valeur initiale du champ

<input> de type text

- Utilisation de <datalist>

```
<form>
  <fieldset>
    <legend>Data List</legend>
      <input name="language" list="languages" />
    </fieldset>
</form>

<datalist id="languages">
  <option value="Java" />
  <option value="C" label="Langage C" />
  <option value="C++">Modern C++</option>
</datalist>
```



<input> de type text

- Utilisation d'une expression régulière

```
CB : <input type="text" name="master_card_number" pattern="^5[1-5]\d{14}$" />
<button type="submit">Envoyer</button>
```

A screenshot of a web browser showing an input field. The input field has a red border and contains the text "zzzz". To the right of the input field is a button labeled "Envoyer". Below the input field is a tooltip box with the text: "Veuillez modifier la valeur pour correspondre au format demandé.".

<input> de type password

- Fonctionne comme le type text
 - les saisies sont masquées
 - attributs :
 - maxlength, pattern, placeholder, readonly, required, size, value
 - pas d'attribut list

<input> pour créer des boutons

- types
 - submit : soumet le formulaire
 - reset : initialise le formulaire
 - button : bouton générique

```
<input type="submit" value="Envoyer">
<input type="reset" value="Annuler" />
<input type="button" value="Cliquer" />
```

Envoyer Annuler Cliquer

<input> de type checkbox

- Permet d'effectuer plusieurs choix
- L'attribut `name` doit être le même pour tout les checkbox appartenant au même groupe
- Si `value` n'est pas présent, c'est le texte affiché qui est envoyé au serveur

```
<input type="checkbox" name="groupe" value="choix1"/>choix1<br/>
<input type="checkbox" name="groupe" value="choix2"/>choix2<br/>
<input type="checkbox" name="groupe" value="choix3"/>choix3<br/>
```

choix1
 choix2
 choix3

<input> de type radio

- Permet d'effectuer un seul choix
- L'attribut `name` doit être le même pour tout les radio appartenant au même groupe
 - permet le comportement exclusif
- Si `value` n'est pas présent, c'est le texte affiché qui est envoyé au serveur

```
<input type="radio" name="choix" value="choix1"/>choix1<br/>
<input type="radio" name="choix" value="choix2"/>choix2<br/>
<input type="radio" name="choix" value="choix3"/>choix3<br/>
```

choix1
 choix2
 choix3

<input> de type hidden

- Le champ n'est pas affiché
- Permet de positionner des valeurs récupérées ensuite par le serveur

```
<input type="hidden" name="article_id" value="123" />
```

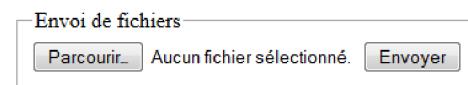
<input> de type file

- Permet d'envoyer des fichiers vers le serveur
 - encodage *multipart/form-data*
- Attribut
 - accept : types MIME acceptés
 - multiple : permet l'envoi de plusieurs fichiers
 - required : obligatoire

<input> de type file

- Exemple

```
<form enctype="multipart/form-data" method="post"
    action="http://localhost:8080/echo/" target="_blank" >
    <fieldset>
        <legend>Envoi de fichiers</legend>
        <input type="file" name="fichiers" multiple="multiple"/>
        <button type="submit">Envoyer</button>
    </fieldset>
</form>
```



<input> de type image

- Permet de faire un bouton de soumission avec une image
- Attributs
 - src : URL de l'image
 - formaction : redéfini action du formulaire
 - formenctype : redéfini enctype du formulaire
 - formmethod : redéfini method du formulaire
 - formtarget : redéfini target du formulaire
 - formnovalidate : redéfini novalidate du formulaire

<input> de type image

- Les coordonnées de l'image sont envoyées au serveur

```
<form method="post" action="http://localhost:8080/echo/" target="_blank" >
  <fieldset>
    <legend>Démos</legend>
      <input name="nom" type="text" size="10" maxlength="10" placeholder="Votre nom"/>
      CB : <input type="text" name="master_card_number" pattern="^5[1-5]\d{14}$" />
      <input type="image" src="ok01.png" width="20" name="submit"/>
  </fieldset>
</form>
```

Keys	Values
master_card_number	
nom	
submit.x	5
submit.y	11

<input> autres types

- D'autres types peuvent être utilisés
 - number
 - range : saisie d'un nombre avec bornes, pas
 - boolean
 - date, time
 - color
 - email, tel, url

<select>

- Permet la création de boîtes déroulantes et boîtes à liste
 - les différents items de la liste sont énumérés dans des éléments <option>
 - l'attribut `multiple` permet d'avoir le fonctionnement d'une liste à sélection multiple
 - l'attribut `size` : nombre de lignes à afficher

<select>

- Exemples

```
<select name="langage">
    <option value="java">Java</option>
    <option value="c">Langage C</option>
    <option value="cpp">C++ 14</option>
</select>
```

Java ▾

```
<select name="fruits" size="5">
    <option value="orange">Orange</option>
    <option value="pomme">Pomme</option>
    <option value="poire">Poire</option>
    <option value="fraise">Fraise</option>
    <option value="cerise">Cerise</option>
    <option value="citron">Citron</option>
</select>
```

Orange ▾
Pomme
Poire
Fraise
Cerise ▾

<textarea>

- Permet d'obtenir une zone de saisie sur plusieurs lignes
 - attributs
 - cols : nombre de colonnes
 - rows : nombre de lignes

```
<textarea name="message" cols="20" rows="5" spellcheck="false">  
</textarea>
```

