2. CSS
(Cascade StyleSheet)

CSS : *Cascade Style Sheets*

- **CSS** = règles + feuilles de style.
- **Règle** = état d'un aspect stylistique d'un ou plusieurs éléments.
- **Feuille de style** = ensemble de règles qui s'applique à un document HTML.
- **Exemple :** une feuille de style comportant une unique règle :

```
H1 { color : blue}
```

Introduction

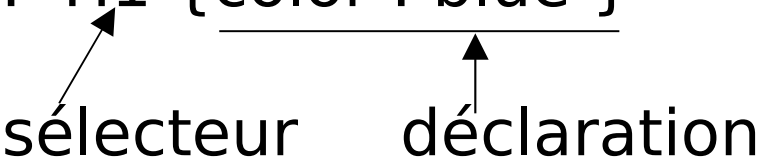
- **Objectif :** offrir aux concepteurs de sites Web, un outil puissant de présentation des documents HTML (et XML).
- **CSS :** *cascade style sheets*, est une mécanisme simple permettant d'ajouter du style aux documents HTML.
- Normes :
  - juillet 1997 : CSS 1
  - janvier 1998 : CSS 2
  - CSS 3 (en cours de validation).

Versions

- CSS 1
  - Recommendation du W3C
  - Uniquement pour HTML
  - Incomplet
- CSS 2
  - Pour XML et HTML
  - Recommendation W3C actuelle
- CSS 3
  - Toujours pas disponible
- Syntaxe non XML

Règle des CSS

- Une règle est constituée de 2 parties :
  - Un sélecteur (lien entre le document HTML et le style).
  - Une déclaration (une partie de la règle spécifiant une partie de la présentation).
  - Syntaxe : H1 {color : blue }



- Presque tous les éléments peuvent être utilisés comme une sélecteur (il n'y a par d'intérêt à utiliser un élément invisible, ex : br).
- Une déclaration se décompose en propriétés et valeurs (*color* est la propriété et *blue* est la valeur).

Groupement de règles

- Lorsque plusieurs sélecteurs possèdent une même définition, il est préférable de les regrouper.

  Ex : `H1, H2, H3 { font-weight : bold }`

- Lorsqu'une règle possède plus d'une propriété, on sépare les propriétés par des ";".

  Ex : `H4 { color : red ; font-weight : bold }`

Lien avec les documents HTML

- Il existe quatre méthodes pour attacher une feuille de style à un document HTML.
  - Appliquer à l'ensemble du document HTML à l'aide de la balise <style> dans l'en-tête.
  - Appliquer à une balise à l'aide de la balise <style>
  - Lier le document à une feuille de style externe à l'aide de la balise <link>
  - Lier le document à une feuille de style externe à l'aide de "@import".

Style de document

- Imbriquer la feuille de style entre commentaires pour les navigateurs non compatibles CSS.
- **Inconvénient :** Maintenance sur site volumineux.

```
<html><head>
<title>essai css</title>
<style type="text/css"> <!--
 h1 {color : blue}
 --> </style>
 </head>
 <body>
 <h1>premier essai avec
  CSS</h1>
 </body>
 </html>
```

Styles en lignes

- Méthode la plus simple d'associer un style à une balise.
- **Principe :** inclure un attribut style à la balise en plus de la liste habituelle des propriétés et des valeurs respectives.
- **Inconvénients :** code lourd et maintenance.

```
<html>
<head>
<title>Page styles en ligne</title>
</head>
<body>
<h1 style="color : blue; font-style
:italic"> Chapitre 1</h1>
</body>
</html>
```

Feuilles de style externes liées

- La balise <link> doit apparaître dans l'en-tête.
- L'URL de la feuille de style peut être absolu ou relatif à l'URL de base du document.
- **Avantage :** Maintenance sur des sites volumineux.

```
<html><head>
<title>page avec link</title>
<link rel="stylesheet" type="text/css"
    href="styles.css">
</head>
<body>
<h1>Chapitre 1</h1>
</body>
</html>
```

Feuilles de style importées

- @import doit apparaître dans la balise <style> dans l'en-tête.
- L'URL de la feuille de style peut être absolu ou relatif à l'URL de base du document.
- Peut aussi apparaître dans une feuille de style externe.

```
<html><head>
<title>page avec
    @import</title>
<style><!--
@import url(“style.css”);
-->
</style></head>
<body>
<h1>chapitre 1</h1>
</body>
</html>
```

Feuilles de style externe

- Dans un document texte avec extension ".css".
- Réutilisation dans plusieurs documents HTML pour la gestion d'une charte graphique sur un site contenant beaucoup de pages (maintenance).

```
BODY { background-color :
    Yellow; color : Black }

H1 { font-weight : bold; color :
    red}
```

Hiérachie des styles

- **Ordre de priorité :**
  style en ligne > style de document > @import > link
- Différence entre @import et link :
  - Équivalence lorsqu'une seule balise <link>
  - Lorsqu'il y a plusieurs feuilles externes, @import est préférable, car il y a fusion des styles. La feuille du dernier import est prioritaire.

Classes de style

- Les classes de style permettent de définir plusieurs styles différents pour une ou plusieurs balises.
- On peut définir des classes régulières (attachées à une balise) ou bien génériques (pour l'ensemble des balises).
- **Définition d'une classe dans une feuille de document ou externe :**
  - ✗ Classe régulière d'un paragraphe : P.*nomclasse* {propriétés : valeurs ; etc.. }

  Classe générique : .*nomclasse* {propriétés : valeurs ; etc.. }
  - **Exploitation :**
    - ✗ Pour une classe régulière : <P class="nomclasse">abc</p>.
    - ✗ Pour une classe générique on peut utiliser les balises et .

Pseudo-classes

- Les pseudo-classes permettent de définir l'affichage de styles pour certains états de balises.
- Les noms sont prédéfinis.
- Elles sont rattachées au nom de la balise par ":" et non ".".
- Principales pseudo-classes : A:link, A:active, A:visited, P:first-letter, P:first-line.
- Attention à la compatibilité des navigateurs avec les pseudo-classes introduites dans CSS2 (A:hover, A:focus, :first-child, :lang).

Identificateur

- L'attribut id assigne à l'élément correspondant un identificateur unique au sein du document.
- On peut définir des identificateurs réguliers (attachés à une balise) ou bien génériques (pour l'ensemble des balises).
- **Définition d'un identificateur dans une feuille de document ou externe :** `#jaune {color : jaune}` ou `H1#bleu {color : blue}`
- **Exploitation :** `<span id="jaune">Jaune</span>` ou `<H1 id="jaune">Titre bleu</H1>`

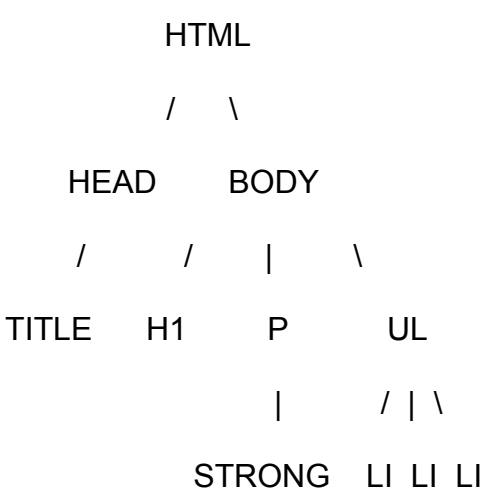
DIV et SPAN

- Balises introduites dans HTML 4.
- DIV : Crée une division au sein d'un document. Fonctionnement sur un bloc.
- SPAN : Permet de déléguer le formatage du texte entre les balises à une feuille de style. Fonctionnement sur la ligne.
- Attribut « id » pour identifier une zone
- Exemple: <div id=''menu''>...

Héritage

```
<HTML>  <HEAD><TITLE>CSS1</TITLE>
</HEAD>
 <BODY>
  <H1>Titre 1</H1>
  <P><STRONG>CSS1</STRONG> intro:</P>
  <UL>    <LI>selecteurs
   <LI>héritage
   <LI>exemple
  </UL>  </BODY></HTML>
```

Arbre du document HTML de droite



- L'organisation interne d'un document HTML est supportée par le DOM (Document Object Model) sous forme d'arborescence.
- Les balises qui se trouvent dans une section délimitée par d'autres balises héritent des propriétés et valeurs de la balise mère.

Propriétés sur les caractères

```
element     {
  font-family: Helvetica, Arial;
  font-size: x-large;
  font-style: italic;
  font-variant: small-caps;
  font-weight: 900;
  font-stretch: semi-expanded
}
```

Propriétés sur le texte

```
element      {
  text-indent: 0.5in;
  text-align: center;
  text-decoration: underline;
  text-transform: capitalize;
  white-space: normal
}
```

Propriétés sur les couleurs

```
element      {
 color: #00FF00;
background-color: rgb(43, 43,43);
border-color: black
}
```

Unités

- Les valeurs de certaines propriétés peuvent être exprimées dans plusieurs unités.
- **Couleur :** *nom d'une couleur de la palette web, #rgb, #rrggbb, rgb (0-255,0-255,0-255), rgb (0-100%,0-100%,0-100%).*
- **Taille :** on distingue deux types : relative et absolue.
  - Relative : em, ex, px et %.
  - Absolue : in, cm, mm, pt, pc.

Les listes

- 2 types de liste en html: non ordonnée (ul) et ordonnée (ol)
- CSS permet d'ajouter du style aux listes
- Pour ul: ul.a {list-style-type : x; }
  - Avec x= {circle, none, disc (défaut), square}
- Pour ol: ol.b{list-style-type: y;}
  - Y= {decimal, lower-alpha, lower-greek, lower-roman..}

Les listes (suite)

- On peut mettre une image: ul {list-style-image: url(''abc.png'');}
-

CSS float

- Un élément peut être déplacer vers la gauche ou la droite et avoir d'autres éléments l'encadrant. Utile pour les images.
  - Exemple : im {float: left;}
- Pour annuler float, on utilise clear
  - Ex: .text {clear:both;}, both indique que l'on annule à droite et à gauche.
- Valeurs possibles: {left, right, both, none, inherit}

Un point sur les propriétés

- Les grandes classes des propriétés de CSS 1 : police de caractères, disposition du texte, couleurs, fonds de document et boîtes.

- Le CSS2 étend le jeu de propriétés au positionnement des images, effets visuels, media, etc.. L'exploitation de ces propriétés se posent au niveau de la compatibilité des navigateurs.

- Normes CSS et propriétés sur le site du w3c (www.w3c.org).

- Articles sur webreview (www.webreview.com), HTMLHelp, le site css.nu (http;//css.nu).

- Outils : TopStyle , Style Master, CSS Validator : http://jigsaw.w3.org/css-validator/

3. Javascript

Présentation

- Développé pour ajouter de l'interactivité aux pages HTML.
- Souvent inséré directement dans les pages HTML.
- Un langage de script (un langage de programmation léger)
- Langage interprété, orienté objet, typage dynamique

Possibilités

- Pour ajouter du texte dynaniquement dans une page
- Pour réagir aux événements
- Pour lire et écrire des éléments HTML
- Pour valider des données saisies
- Pour créer des cookies
- Pour détecter le navigateur du client
- ...

Balises DIV, SPAN et LAYER

- Positionnement relatif contre absolu:
  - `<DIV  ID = ''name'' STYLE= ''position: absolute; top 30px; left 10px; ''>`
  - `<DIV STYLE= ''float: left;''>`
-

Inclure le code Javascript

- Inclusion dans la page HTML (dans head ou body)

```
<body>
   <script type=''text/javascript''>
      Alert('Hello'); </script>
</body>
```

- Ou référence à un fichier externe

```
<body>
   <script language=''javascript''
   type=''text/javascript'' src=''text.js''>
   </script>
</body>
```

Variable

- Avec ou sans var:

  var i = valeur ou i = valeur

- Une variable (locale) déclarée dans une fonction limite l'accès de la variable à l'intérieur de la fonction.
- La déclaration d'une variable (globale) hors d'une fonction permet l'accès de celle-ci dans toutes le fonctions
- Durée de vie d'une variable : début à la déclaration et fin lors de la fermeture de la page.

Popup

- Alert box
  - Utilisateur doit cliquer sur « OK » pour poursuivre
  - alert("message")
- Confirm box
  - Utilisateur doit cliquer sur « Ok » ou « Cancel » pour poursuivre
  - confirm("message")
- Prompt box
  - Utilisateur doit cliquer sur « Ok » ou « Cancel » pour poursuivre après une saisie
  - prompt("message","valeur par défaut")

Elément du langage

- Conditionnelle
  - if, if.. else, switch
- Boucle
  - for, while
- try...catch
- throw

Javascript et les objets

- Comme en POO, Object est un type dont les autres objets dérivent.
- La comparaison avec le POO s'arrête là.
- Création:
  - var monObjet = new Object();
- Pas de méthodes, pas propriétés.

Ajouter des propriétés

- On peut ajouter des propriétés à notre instance de Object.
  - `monObjet.annee = 2009;`
  - `monObjet.etat = 'neuf';`
  - `monObjet.achat = new Date(2010,1,1);`
- Chaînage de références
  - `var createur = new Object();`
  - `createur.nom = 'pierre';`
  - `monObjet.createur = createur;`
- Accès
  - `var nomCreateur = monObjet.createur.nom;`

JSON (Javascript Object Notation)

- Notation plus compacte et facile d'utilisation.

```
var monObjet = {
    annee: 2009,
    achat: new Date(2010,2,1),
    etat: 'neuf',
    createur: { nom : 'pierre'},
    valeurs: {2,3,5,7,9}
};
```

Variable et propriétés

- Stockage d'une référence à un objet JS:
  - `var maVar = 'salut';`
  - `unObjet.unePropriété = 'bonjour';`
- Lorsque `var` est utilisé au plus haut niveau d'une page HTML (hors d'une fonction), il correspond à une propriété de `window`. Donc, il y a équivalence entre:
  - `var foo = bar;`
  - `window.foo=bar;`
  - `foo = bar;`
- `bar` est non qualifié donc c'est une propriété de `window`.

Synthèse

- Un objet Javascript est une collection non ordonnée de propriétés
- Une propriété consiste en un couple nom/valeur.
- JSON facilité le déclaration d'objets
- Les variables de haut-bniveau d'un document sont des propriétés de window.

Les fonctions

- Les fonctions comme objet central de javascript
- Les fonctions JS sont considérées comme des objets.
- Elles peuvent être:
  - Affectées à des variables, des propriétés d'un objet, passées comme paramètre, retournées comme résultat, déclarées à en utilisant un formalisme comme JSON.

Les fonctions (suite)

- Une fonction Javascript contient du code qui sera exécuté seulement lors de son invocation
- On peut invoquer une fonction depuis n'importe où dans la page ou depuis une autre page si la fonction se trouve dans un fichier externe.
- On peut définir une fonction dans la balise head ou body. La convention est de les définir dans head.
- Le code ne se trouvant pas dans une fonction est exécuté dès le chargement de la page.

Example

```
<html>
<head>
<script type="text/javascript">
  function displaymessage() {
      alert("Hello World!")
  }
</script>
</head>
<body>
<form>
<input type="button" value="Click me!"
onclick="displaymessage()" >
</form>
</body>
</html>
```

Function comme un objet

- Le code suivant ne crée pas une fonction 'test':
  - function test { alert('faire un truc');}
- Cela va créer un instance Function et l'affecter à une propriété window:
  - test = function() { alert('faire un truc');}

Equivalence

- Les déclarations suivantes sont équivalentes:
  - `function hello() { alert('hello');}`
  - `hello = function() {alert('hello');}`
  - `window.hello = function() {alert('hello');}`

Asynchrone

- La nature d'une page web est asynchrone.
- La notion de *callback* est prépondérante dans la programmation asynchrone.
- Exemple:

```
function hello() {alert('hello');}

setTimeout(hello,5000);
```

- La fonction hello est un paramètre de setTimeout. Après 5sec. appelle hello (callback).
- On pouvait aussi écrire:

```
setTimeout(function {alert('hello');},5000);
```

Fonction comme propriété d'un objet

- Le mot clé *this* donne l'accès à l'objet associé à une fonction.
- Différence avec utilisation en POO (ex Java)
  - *this* ne pointe pas sur l'instance où a été déclarée la méthode (POO) mais sur l'instance d'où elle est invoquée. La notion de contexte d'appel est importante.

Fonction et contexte

- On peut fixer le contexte d'une fonction avec les méthodes **call()** et **apply()**.
- Ce sont des méthodes définies par le constructeur de function.
- call() invoque la fonction en spécifiant (1er argument) l'objet devant servir de contexte. Les arguments suivant de call sont les paramètres de la fonction.
- Apply() fonctionne de manière similaire à call()mais le second paramètre est un tableau d'objets qui deviendront les paramètres de la fonction.

Exemple

```
<html>  <head>   <script>
    var o1 = {handle:'o1'};
    var o2 = {handle:'o2'};
    var o3 = {handle:'o3'};
    window.handle = 'window';
    function whoAmI() { return this.handle;}
    o1.identifyMe = whoAmI;
    alert(whoAmI());
    alert(o1.identifyMe());
    alert(whoAmI.call(o2));
    alert(whoAmI.apply(o3));
  </script> </head> <body></body> </html>
```

Affiche séquentiellement: window, o1, o2 puis o3

Synthèse

- En javascript, on ne peut pas dire qu'une fonction est une méthode d'un objet.
- On peut dire:
  - Une fonction f agit comme la méthode d'un objet o lorsque o sert comme contexte de la fonction f.
- Exemple:
  - alert(o1.identifyMe.call(o3));

Evénements et leurs gestions

- Chaque élément d'une page web a des événements qui peuvent déclenchés une action.
- Des attributs sont insérés dans les éléments HTML pour définir événements et gestionnaires.
- Exemple:
  - Clic souris
  - Chargement d'une page ou d'une image
  - Curseur passant sur une zone donnée
  - Sélection d'une boîte d'un formulaire
  - Soumission d'un formulaire
  - Frappe clavier

Evénements

- onabort – chargement d'une image est interrompu
- onblur – un élément perd le focus
- onchange – le contenu d'un champ a changé
- onclick – Clic souris sur un objet
- ondblclick – Double clic souris sur un objet
- onerror – Une erreur survient lors du chargement d'une image ou d'un document
- onfocus – Un élément prend le focus
- onkeydown – Une touche clavier est pressée

Evénements (suite)

- onkeypress – une touche clavier est pressée ou maintenue
- onkeyup – relâchement d'un touche clavier
- onload – fin de chargement d'une page/image
- onmousedown – bouton de la souris est pressé
- onmousemove – la curseur a bougé
- onmouseout – la curseur de la souris sort d'un élément
- onmouseover – le curseur se déplace au dessus d'un élément
- onmouseup – le bouton de la souris est relâché

Evénements (fin)

- onreset – le bouton reset en cliqué
- onresize – une fenêtre est redimensionnée
- onselect – sélection d'un texte
- onsubmit – clic sur un bouton submit
- onunload – l'utilisateur quitte une page

onload et onUnload

- Les evts onload et onUnload sont déclenchés lorsque l'utilisateur entre ou quitte une page
- L'evt onload est souvent utilisé pour détecter le type et la version du navigateur de l'utilisateur et charger la version adéquate de la page.
- Ils sont aussi utilisés pour gérer les cookies

OnFocus, onBlur et onChange

- Les evts onFocus, onBlur et onChange sont souvent utilisés pour la validation des formulaires
- Exemple: la fonction checkEmail() va être invoquée à chaque fois que l'utilisateur change le contenu d'un champ:

```
<input type="text" size="30"
 id="email" onchange="checkEmail();">;
```

Exemple onBlur

```
<html>
<head>
<script type="text/javascript">
    function upperCase() {
      var x=document.getElementById("fname").value;

document.getElementById("fname").value=x.toUpperCase();
    }
</script>
</head>
<body>
Enter your name:
<input type="text" id="fname" onblur="upperCase()">
</body>
</html>
```

onSubmit

- L'evt onSubmit est utilisé pour valider les champs d'un formulaire avant sa soumission
- Exemple: la fonction checkForm() sera appelé lorsque l'utilisateur cliquera sur le bouton submit du formulaire. Si la valeur d'un champ n'est pas accepté, la soumission sera annulé. La fonction retourne vrai ou faux. Si c'est vrai, le formulé sera envoyé sinon, le submit sera annulé.

```
<form method="post" action="xxx.html" onsubmit="return checkForm()">
```

Exemple onSubmit

```
<html>
<head>
<script type="text/javascript">
 function validate() {
   // return true or false based on validation logic
 }
</script>
</head>
<body>
    <form action="tryjs_submitpage.htm" onsubmit="return validate()">
      Name (max 10 chararcters): <input type="text" id="fname"
size="20"><br />
      Age (from 1 to 100): <input type="text" id="age" size="20"><br />
      E-mail: <input type="text" id="email" size="20"><br />
      <br />
      <input type="submit" value="Submit">
    </form>
</body>
</html>
```

OnMouseOver et onMouseOut

- Les evts onMouseOver et onMouseOut sont utilisés pour créer de boutons « animés ».
- Exemple : Une AlertBox apparaît lorsque un onMouseOver evt est détecté:

```
<a href="http://www.w3schools.com"
onmouseover="alert('An onMouseOver
event');return false">

<img src="w3schools.gif" width="100"
height="30">

</a>
```

4. Ajax