

INTERFACES WEB AVANCEES - 1

3C- 2015/2016

Agenda

Session 1 Les bases

- Introduction
- CSS
- HTML
- Javascript

Session 2 La productivité

- Frameworks
- Outils de développement

Session 3 Allons plus loin

- ECMAScript 6 (Object.observe, Quick fonction, Promise)
- Introduction à d'autres langages
- Web component

<https://iwa2015.herokuapp.com/>

Slides, exemples de cours, démo, sujet du TP

World Wide Web

RÉTROSPECTIVE ET PERSPECTIVES

World Wide Web

The WorldWideWeb (W3) is a wide-area [hypermedia](#) information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an [executive summary](#) of the project, [Mailing lists](#), [Policy](#), November's [W3 news](#), [Frequently Asked Questions](#).

[What's out there?](#)

Pointers to the world's online information, [subjects](#), [W3 servers](#), etc.

[Help](#)

on the browser you are using

[Software Products](#)

A list of W3 project components and their current state. (e.g. [Line Mode](#), [X11 Viola](#), [NeXTStep](#), [Servers](#), [Tools](#), [Mail robot](#), [Library](#))

[Technical](#)

Details of protocols, formats, program internals etc

[Bibliography](#)

Paper documentation on W3 and references.

[People](#)

A list of some people involved in the project.

[History](#)

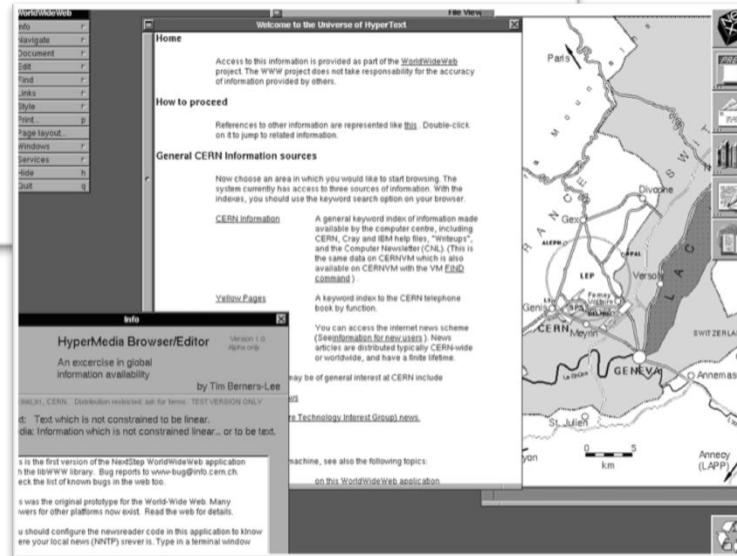
A summary of the history of the project.

[How can I help ?](#)

If you would like to support the web..

[Getting code](#)

Getting the code by [anonymous FTP](#), etc.



[TheProject.html](#), 6 août 1991

Tim Berners-Lee (Physicien -CERN)

Le commencement (début 90')

World Wide Web

- ▶ Système hypertexte
- ▶ Fondé sur Internet (TCP/IP, DNS, MIME, ...)
 - ▶ Transfert de fichiers via protocole HTTP
 - ▶ Pages écrites en HTML (Hypertext Markup Language)
 - ▶ Reliées par hyperliens ou URL (Uniform Resource Locator)



World Wide Web Consortium

- ▶ Organisme de standardisation
- ▶ Fondé en octobre 1994 au MIT/LCS
- ▶ Rôles :
 - ▶ Promouvoir l'universalité des technos web
 - ▶ Superviser le développement de standards (XML, CSS, PNG...)



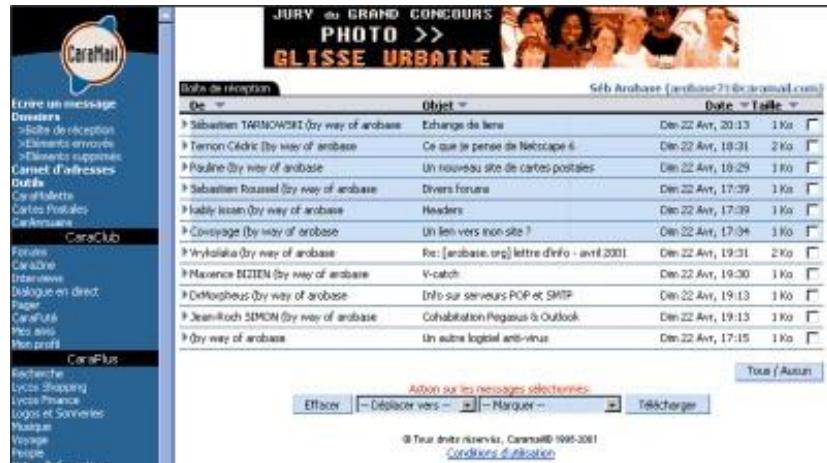
Montée en puissance (Din90')

Les applications interagissent avec le web

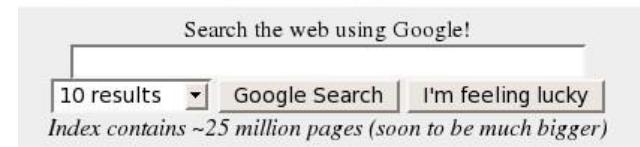
Généralisation des échanges (webmail, tchat)

Evolution des technologies du web :

- ▶ Enrichissement des langages
- ▶ Apparition des styles



Google!



[About Google!](#)

[Stanford Search](#) [Linux Search](#)

Get Google! updates monthly!

your e-mail [Subscribe](#) [Archive](#)

Copyright ©1997-8 Stanford University

L'explosion sociale (~ 2004)

Une forte croissance du nombre d'internautes (source : Banque mondiale)

Bouleversement des usages :

- ▶ Partage d'information (vidéo, photos, liens)
- ▶ Edition collaborative (wiki)
- ▶ Sites communautaires (blogs)
- ▶ Réseaux sociaux, mondes virtuels



The screenshot shows a Wikipedia article titled 'Association management company'. The page includes a sidebar with navigation links like 'Main Page', 'Contents', and 'Random article', as well as a search bar. The main content discusses the role of association management companies (AMCs) in managing associations, mentioning their growth in number and size, and their involvement in various industries. It highlights the complexity of managing associations due to changes in industry professionals, technology, and government regulations. The text also notes the shift from non-profit to for-profit management and the professionalization of association management.

Axé sur la donnée et l'expérience utilisateur :

- ▶ Applications riches
- ▶ Personnalisation
- ▶ Profiling

Simplicité et interactivité



- ▶ En quelques chiffres :
 - ▶ 3,3 milliards de requêtes sont effectuées chaque jour (100 milliards par mois).
 - ▶ 500 millions de tweets sont envoyés chaque jour
 - ▶ 6 milliards d'heures de vidéo vues par mois

Le « web 2.0 » démystifié

Au delà du **buzz marketing** et des **controverses** qu'il génère...

Terme générique utilisé en 2003 par Dale Dougherty

Médiatisé par O'Reilly Medias (conférences, publications)

Encore désigné par web « participatif » ou « social »

Les concepts clefs :

- ▶ Le web comme plateforme (orienté service ou WOA)
- ▶ Utilisation de l'intelligence collective via des communautés
- ▶ Importance des données et de leur mise à disposition
- ▶ Evolution transparente des services
- ▶ Syndication du contenu
- ▶ Applications web et interfaces enrichies

Le « web 2.0 » démystifié

Quelles technologies ?

- ▶ La syndication avec RSS ou ATOM
- ▶ Formats de données XML et JSON
- ▶ Feuilles de styles CSS
- ▶ Le modèle objet (DOM) et HTML
- ▶ Le langage JavaScript et les frameworks associés
- ▶ L'émergence d'AJAX (Asynchronous Javascript and XML)
- ▶ L'assemblage de composants (ou mashups)

Le « web 2.0 » démystifié

RIA versus RDA ?

RIA (Rich Internet Application) :

Application web, ne nécessite que le browser pour fonctionner

RDA (Rich Desktop Application) :

Application embarquée dans une page web, nécessite une machine virtuelle sur le client pour fonctionner (Java et applets, Flash et Flex, .net et Silverlight,...)

Quelques faits :

Performances croissantes des navigateurs

Amélioration des capacités du langage HTML

Compatibilité des plateformes mobiles

Contraintes liées aux environnements propriétaires.

Diminution de projets RDA au profit de projets RIA !

Poussés par les géants du web...

Zoom sur RIA

Client léger

Les pages sont construites côté serveur

Le client se charge uniquement de la présentation

Chaque interaction est transmise au serveur qui va recalculer tout ou partie de la page

Framework MVC (Grails, GWT, Vaadin, ASP.net)

Client riche

Exécute une partie des traitements localement

Echanges d'objets métier avec le serveur (AJAX)

Framework MVC JavaScript (AngularJS, Backbone, ...)

Et le web de demain ?

L'internet des objets ?

Web sémantique ?

Web intelligent ?

Web « 3D » ?

Caractéristiques :

Mobilité

accessible depuis tout type de support

Universalité

abstraction des couches soft et hard

Accessibilité

à la fois pour l'Homme et la machine



HTML

HYPertext Markup Language

1.0 < HTML < 4.01

Dérivé de SGML (Standard Generalized Markup Language)

Exemple :

```
<TITLE>Exemple de HTML</TITLE>
Ceci est une phrase avec un <A HREF=cible.html>hyperlien</A>.
<P>Ceci est un paragraphe o&ugrave; il n'y a pas d'hyperlien.
```

Initialement, peu de balises : <TITLE>, <A>, <P>

Introduction de nouveautés par les navigateurs (Netscape)

... standardisées avec HTML 3.2 puis 4.0 :

- ▶ Eléments de présentation (formulaires, tableaux)
- ▶ Nouvelles balises (<h#>, , , , <i>...)
- ▶ Feuilles de style en cascade (CSS)
- ▶ Exécution de scripts (JavaScript)
- ▶ Structure en arbre pour chaque document (DOM)

Trois variantes du format (strict / transitional / frameset), DTD

XHTML ? Oui... mais non !

- ▶ Vers 2000, spécification HTML suspendue au profit de XHTML
 - ▶ Reformulation d'HTML 4.0 en application XML
 - ▶ Seul la syntaxe change (plus rigoureuse)
- ▶ Réticences des fabricants de navigateurs et des concepteurs de contenu web
- ▶ Finalement, spécification d'XHTML 2.0 abandonnée en 2009 au profit d'HTML5

Comparatif HTML et XHTML :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<title>Exemple HTML 4</title>
<ul>
  <li>Elément de liste.</li>
  <li>Autre élément de liste.
</ul>
<p>Texte <EM Class=important>mis en évidence</EM>.
<input type="checkbox" checked value="..." />

```

```
<!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">
<head><title>Exemple XHTML 1.0</title></head>
<body>
  <ul>
    <li>Elément de liste.</li>
    <li>Autre élément de liste.</li>
  </ul>
  <p>Texte <em class="important">mis en évidence</em>.
    <input type="checkbox" checked="checked" value="..." />
    
  </p>
</body>
</html>
```



HTML5 !

- ▶ Offline (cache, events)
- ▶ Stockage local (SQLite, key/value)
- ▶ Connectivité (WebSocket)
- ▶ Accès au système de fichier local
- ▶ Web sémantic (nouvelles balises, attributs, microdata)
- ▶ Audio/Vidéo (Canvas, SVG, WebGL)
- ▶ CSS3 (transformation 2D/3D, transitions, WebFonts)
- ▶ ETC.

	2012	2013	2014	2015	2016
HTML 5.0	Candidate Rec	Call for Review	Recommendation		
HTML 5.1	1st Working Draft		Last Call	Candidate Rec	Recommendation
HTML 5.2 ^[27]				1st Working Draft	

Retour sur HTML

Les éléments de HTML

- ▶ Structure générale :
 - ▶ Un en-tête (`<head>`) : contient les méta données, l'import de styles, de scripts et de bibliothèques JavaScript
 - ▶ un corps (`<body>`) : contient ce qui sera affiché
 - ▶ Mise en forme (`<h#>`, ``, `<i>`)
 - ▶ Listes (``, ``)
 - ▶ Tables (`<table>`, `<tr>`, `<td>`, `<th>`)
 - ▶ Hyperliens (`<a>`)
 - ▶ Images, d'applets et autres objets (``, `<iframe>` ...)
 - ▶ Eléments de regroupement (`<div>`, ``)
 - ▶ Cadres (`<frameset>`, `<frame>`)
 - ▶ Formulaires (`<form>`, `<input>`, `<textarea>`)

Retour sur HTML

Le Document Object Model

- ▶ **Interface** permettant à des programmes de parcourir et modifier la **structure, contenu et style** d'un document web.
 - ▶ JavaScript comme langage de manipulation (fonctions spécifiques)
 - ▶ Structure du document entièrement montée en mémoire
- ▶ Détection d'**événements** dans le document
 - ▶ Événements **page** et **fenêtre**
onabort, onerror, **onload**, onbeforeunload, onunload, onresize
 - ▶ Événements **souris**
onclick, ondblclick, onmousedown, onmousemove, onmouseout, **onmouseover**,
onmouseup
 - ▶ Événements **clavier**
onkeydown, onkeypress, onkeyup
 - ▶ Événements **formulaire**
onblur, **onchange**, onfocus, onreset, onselect, **onsubmit**

Retour sur HTML

Construire un document HTML valide

- ▶ Indiquer une DTD et s'y restreindre
- ▶ Indiquer l'encodage du texte
- ▶ Ecrire le document en respectant la sémantique
- ▶ Valider sur le site du W3C : validator.w3.org

Exemple avec DTD HTML 4.01 :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
  <head>
    <title>Exemple de HTML</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    <p>Ceci est une phrase avec un <a href="cible.html">lien</a>.</p>
  </body>
</html>
```



Listes et tableaux

► Listes non ordonnées :

```
<ul>
  <li>Coffee</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Milk

► Listes ordonnées :

```
<ol start="1" type="I">
  <li>Coffee</li>
  <li>Milk</li>
</ol>
```

- I. Coffee
- II. Milk

► Tableaux

```
<table border="1">
  <tr>
    <th>Header 1</th><th>Header 2</th>
  </tr>
  <tr>
    <td>row 1, cell 1</td><td>row 1, cell 2</td>
  </tr>
  <tr>
    <td>row 2, cell 1</td><td>row 2, cell 2</td>
  </tr>
</table>
```

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Formulaires

► Formulaire

```
<form name="myForm" action="html_form_action.asp" method="get">
    <input type="text" name="firstname" /><br>
    <input type="radio" name="sex" value="male" disabled /> Male
    <input type="radio" name="sex" value="female" /> Female<br>
    <input type="checkbox" name="vehicle" value="Bike" checked /> Bike
    <input type="checkbox" name="vehicle" value="Car" /> Car<br>
    <select name="cars">
        <option value="volvo" selected>Volvo</option>
        <option value="saab">Saab</option>
    </select><br>
    <textarea rows="10" cols="30">The cat was
        playing in the garden.</textarea><br>
    <input type="submit" value="Submit" />
    <input type="reset" value="Reset" />
</form>
```

The screenshot shows a web page with a form. At the top is a text input field. Below it are two radio buttons: 'Male' (disabled) and 'Female'. Underneath are two checkboxes: 'I have a bike' (checked) and 'I have a car'. A dropdown menu is set to 'Volvo'. A large text area contains the text: 'The cat was playing in the garden.' At the bottom are 'Submit' and 'Reset' buttons.

CSS

CASCADING STYLE SHEETS

Introduction

Pourquoi les CSS ?

- ▶ Séparer la présentation du contenu
 - Le code HTML n'est plus pollué par des balises de présentation (, <i>, , ...)
- ▶ Imposer une structure aux pages HTML
 - ▶ Permet une meilleure indexation par les moteur de recherche (SEO)
 - ▶ Le code est plus accessible
 - ▶ Et donc plus facile à maintenir
- ▶ Beaucoup plus de possibilités de design
- ▶ Facilité d'utilisation (et de réutilisation)
- ▶ Fichiers plus compacts, pages plus rapides à charger

Introduction

Exemple :

- ▶ Pur HTML : 14Ko et 213 lignes de code HTML
- ▶ HTML + CSS : 4Ko et 71 lignes de code HTML



HTML pour CSS

► Comment on faisait avant...

```
<p>
<strong>
<font color="#0066FF" size="5" face="Verdana,
Arial, Helvetica, sans-serif">Urban Agrarian
Lifestyle</font></strong>
<br />
<font color="#FF3300" size="4" face="Georgia,
Times New Roman, Times, serif">
<em>
<strong>A Revolution in Indoor Agriculture
<br /></strong></em></font>
Lorem ipsum dolor sit amet...</p>
```

The Urban Agrarian Lifestyle

A Revolution in Indoor Agriculture

Lore ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure.

► Comment on ferait aujourd'hui... HTML + CSS

```
<h1>The Urban Agrarian Lifestyle</h1>
<h2>A Revolution in Indoor Agriculture</h2>
<p>Lorem ipsum dolor sit amet...</p>
```

HTML pour CSS

2 balises HTML – DIV and SPAN

- ▶ Les balises <DIV> et sont des conteneurs que l'on remplit avec du contenu
 - ▶ La balise <DIV> (pour division)
 - ▶ Marque un bloc de contenu, comme un paragraphe ou une ligne d'en-tête
 - ▶ N'applique aucune mise en forme par défaut
 - ▶ Permet de diviser une page en bloc logique (bannière, menu, pied de page)
 - ▶ Grâce à CSS, on peut positionner ces blocs les uns aux autres pour créer des pages sophistiquées
 - ▶ La balise
 - ▶ Créer des éléments alignés (mots ou phrases dans un paragraphe, <A>, sont des balises similaires)
 - ▶ Permet de mettre en évidence certaines portions du texte
 - ▶ N'applique aucune mise en forme par défaut
-
- ▶ Exemple :

```
<div id="footer">
    <p>Copyright 2006, <span class="bizName">CosmoFarmer.com</span></p>
    <p>Call customer service at 555-555-5501 for more information</p>
</div>
```

Best practices CSS

Oublier les mauvaises habitudes...

(ie. les mauvaises balises HTML !)

- ▶ Plus de balise `` pour formater le texte
- ▶ Ne plus utiliser `` et `<I>` pour emphaser le texte. Préférer `` ou ``
- ▶ Utiliser `<TABLE>` uniquement pour les données tabulaires
- ▶ Ne pas appliquer d'attributs à la balise `<BODY>`
- ▶ Ne pas utiliser de balises/attributs spécifiques à un navigateur
- ▶ Ne pas abuser de la balise `
`

Best practices CSS

... Pour en apprendre de nouvelles !

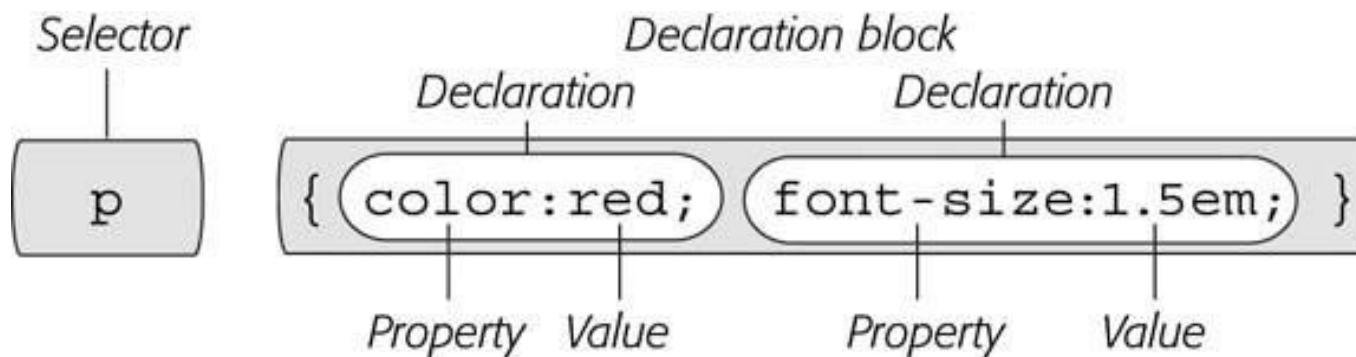
- ▶ Un seul tag <H1> par page
- ▶ Utiliser les balises de titre <Hx> en fonction de l'importance du contenu
- ▶ Utiliser les listes ordonnées ou non ordonnées
- ▶ Pour les citations utiliser <BLOCKQUOTE> ou <Q>
- ▶ Utiliser les balises peu connues comme <ADDRESS> ou <CITE>

- ▶ Utiliser HTML pour décrire le contenu de la page. Utiliser CSS pour la mise en forme ! La spécification HTML5 apporte de nouvelles balises comme <header> ou <footer>

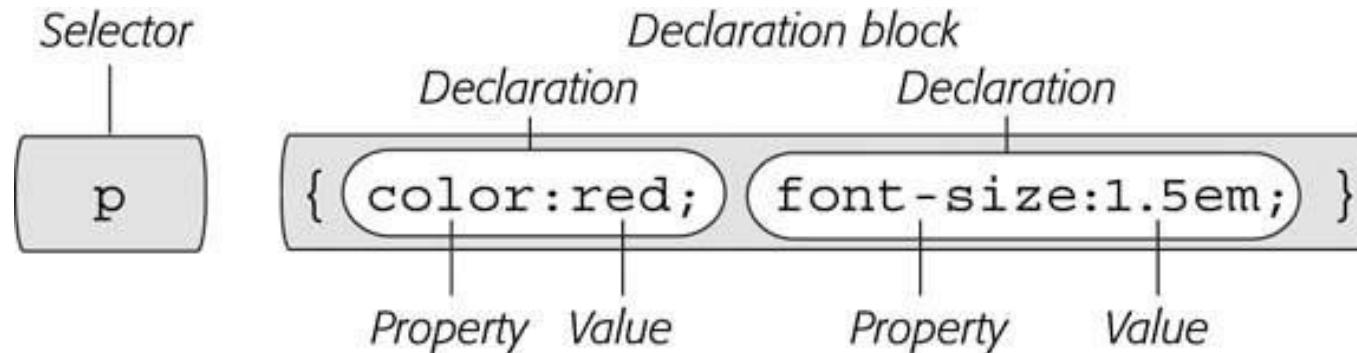
Créer des feuilles de styles

Un style est fait de 2 éléments :

1. L'élément HTML que le navigateur doit mettre en forme, désigné par un **selecteur**.
Toute balise HTML peut être candidate.
2. Les instructions de mise en forme, ou **bloc de déclaration**.
Par exemple : colorer un texte en bleu, épaisseur une bordure, centrer une image au centre

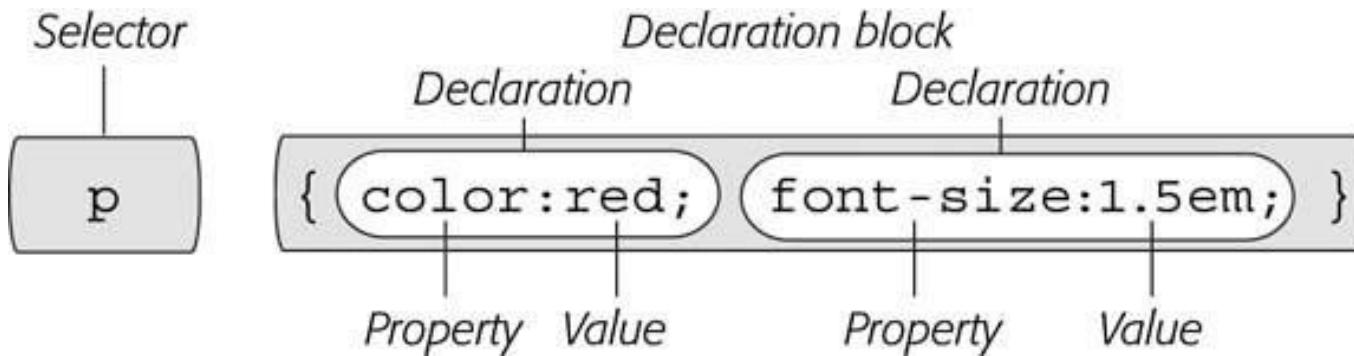


Anatomie d'un style



- ▶ **Selector:** désigne un ou plusieurs éléments dans le DOM à formatter
- ▶ **Declaration Block:** inclus toutes les mises en forme à appliquer pour ce sélecteur
- ▶ **Declaration :** Une ou plusieurs déclaration par bloc, chaque déclaration contient un attribut et une valeur, se termine par un point virugle.
- ▶ **Comments:** /* and */ style

Anatomie d'un style



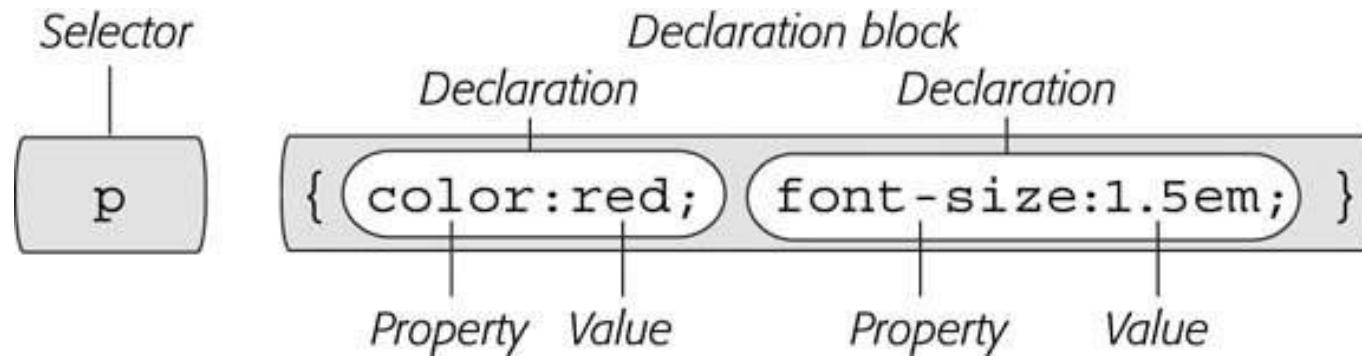
► **Property**

- Large choix d'attribut de mise en forme, appelé propriété.
- Par exemple, la propriété background-color permet de définir une couleur pour l'arrière plan.

► **Value**

- Plusieurs types de valeur :
 - une couleur (ex: red, #FF0000), une longueur (ex:18px, 2in, 5em), une URL (ex: images/background.gif), un mot clef (ex: top, center, bottom)

Anatomie d'un style



```
p {  
    color: red;  
    font-size: 1.5em;  
}
```

Utiliser CSS

- ▶ Déclaration des feuilles de style (balise `<head>`)

- ▶ Depuis un fichier

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

- ▶ Dans le document HTML

```
<style type="text/css">
  h1 {
    color: #FF7643;
    font-face: Arial;
  }
</style>
```

- ▶ Utilisation dans les balises HTML ou inlining (corps `<body>`)

```
<h1 style="color: red; font-size: 4em;">
```

Les sélecteurs CSS

- ▶ Sélecteur universel (toutes les balises du DOM)

```
* { margin: 0; padding: 0; }
```

- ▶ Sélecteur d'ID (balise HTML vérifiant id=« contenu »)

```
#contenu { width: 960px; margin: auto; }
```

- ▶ Sélecteur de classe (balise HTML vérifiant class=« error »)

```
.error { color: red; }
```

- ▶ Sélecteur de pseudo-classe (tous les liens déjà visités)

```
a:visited { color: purple; }
```

- ▶ Sélecteur descendant (tout les liens dans une liste)

```
li a { text-decoration: none; }
```

- ▶ Sélecteur de type (tous les liens du DOM)

```
a { color: red; }
```

Les sélecteurs CSS

- ▶ Sélecteur adjacent (tous les paragraphes et listes)

```
ul + p { color: red; }
```

- ▶ Sélecteur d'enfant (toutes les listes directement descendantes d'une balise ayant pour id container)

```
#container > ul { border: 1px solid black; }
```

- ▶ Sélecteurs conditionnels

X[foo]	Présence d'un attribut
X[href="/foo"]	Egalité stricte
X[href*="foo"]	Présence de la chaîne foo
X[href^="foo"]	Commence par la chaîne foo
X[href\$="foo"]	Se termine par la chaîne foo
X[data-*="foo"]	Attribut commençant par data- contenant foo
X[foo~= "bar"]	Contient la valeur bar

- ▶ Autres exemples :

```
div:not(#contenu) { color: blue; }
div:hover { background: #e3e3e3; }
input[type=radio]:checked { border: 1px solid black; }
ul li:first-child { border-top: none; }
ul > li:last-child { color: green; }
```

Héritage CSS

- ▶ Les balises peuvent hériter des propriétés CSS de leur parent
- ▶ L'héritage fonctionne avec n'importe quel type de style, pas seulement les balises

Ex: quand on applique un style class, les propriétés CSS sont transmises aux balises enfants

- ▶ Certaines propriétés CSS ne sont pas passées aux descendants :
 - ▶ Positionnement, marge, couleur d'arrière plan, padding, bordure
 - ▶ Quand il y a un conflit, le style le plus spécifique gagne
- ▶ Le mécanisme de cascade régit quelles propriétés sont appliquées à quels éléments suivant un certain nombre de règles

La cascade

► Accumulation des styles hérités

```
body { font-family: Verdana, Arial, Helvetica, sans-serif; }
p { color: #999999; }
strong { font-size: 24px; }
```

► Styles appliqués à la balise strong

```
strong {
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #999999;
    font-size: 24px;
}
```

Heading 1

This is the tag--
strong emphasized text within
a paragraph of text. Lorem ipsum dolor sat.

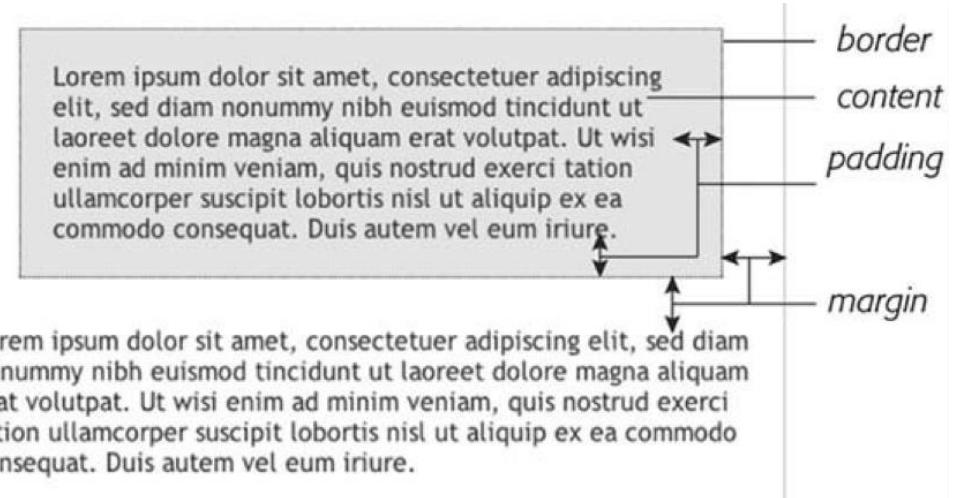
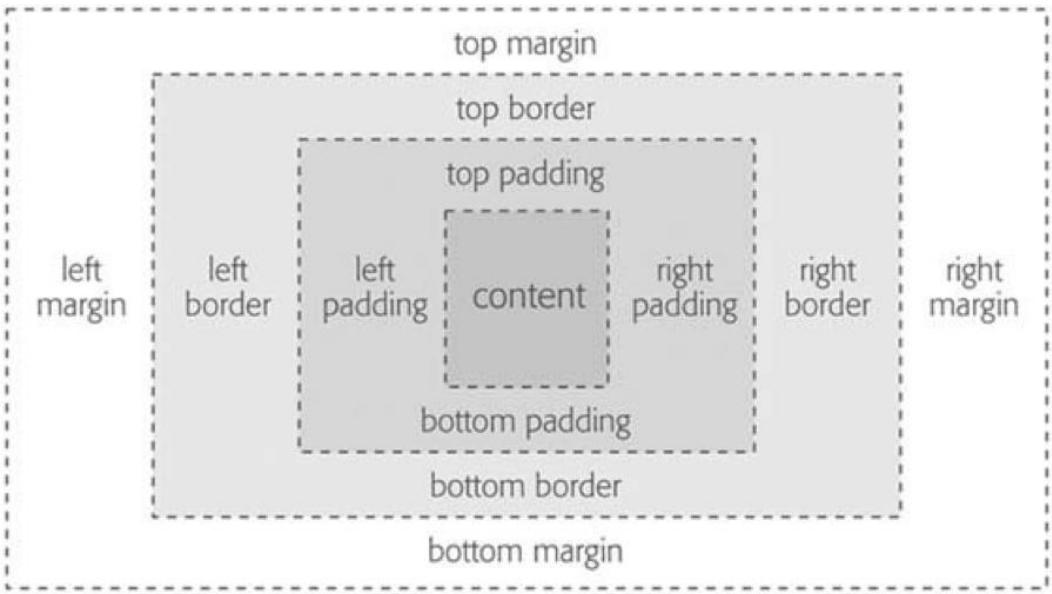
- Bulleted List item 1
- Bulleted List item 2
- Bulleted List item 3
- Bulleted List item 4

La cascade

- ▶ Quelques règles
 - ▶ Le style directement appliqué gagne en cas de conflit
 - ▶ Une balise et plusieurs règles de style combinaison, tant qu'il n'y a pas de conflit
 - ▶ Si conflit il y a, ce dernier est résolu suivant la règle suivante :
 - ▶ **inline** style > **id** selector > **class** selector > **tag** selector
- ▶ Exception:
`p.dark {color: #333 !important; background: white;}`

CSS box model

- ▶ Chaque élément HTML est un conteneur

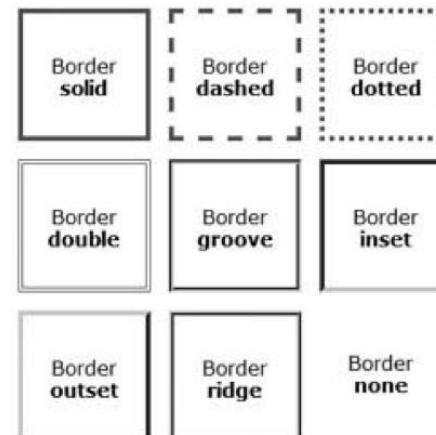


CSS box model

- ▶ Contrôle du **margin** et **padding** sur chaque côté
 - ▶ **marging**: margin-top, margin-right, margin-bottom, and margin-left.
 - ▶ **padding**: padding-top, padding-right, padding-bottom, and padding-left.
- ▶ Racourcis :

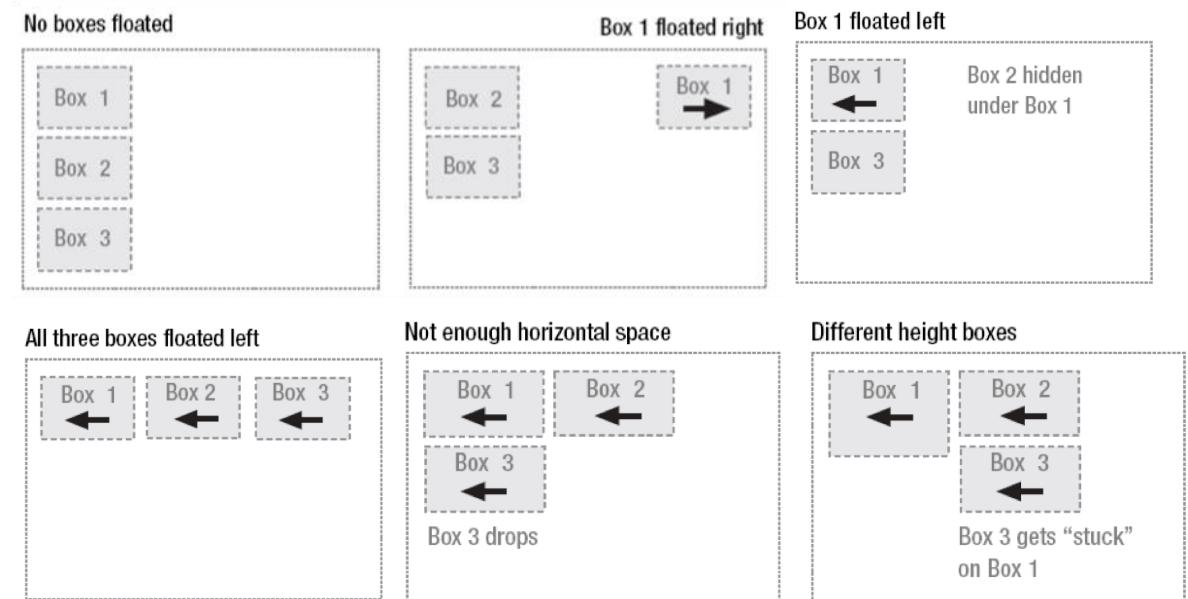
```
/* top, right, bottom, left */  
margin: 0 10px 10px 20px;  
padding: 10px 5px 5px 10px;
```

- ▶ Contrôle du **border**
 - ▶ **Style**: solid, dotted, dashed, double, groove, ridge, inset, outset, none, et hidden



CSS box model

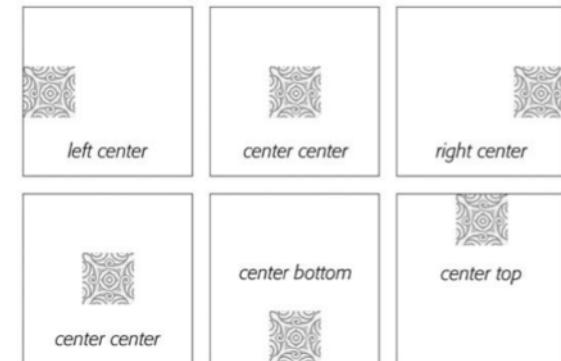
- ▶ Objets flottants :
 - ▶ Par défaut, les objets (balises) sont ajoutés les uns après les autres en partant du haut de la page
 - ▶ La propriété `float` permet de les déplacer à gauche et à droite
 - ▶ **float** : left, right, none
 - ▶ **clear** : left, right, both, none



CSS box model

- ▶ Ajout d'une **background-image** (image de fond)
 - ▶ **background-repeat** : repeat, no-repeat, repeat-x, and repeat-y
 - ▶ **background-position**:
 - ▶ Position horizontale : left, center, right
 - ▶ Position verticale: top, center, bottom

```
body {  
    /* Chemin relatif au fichier CSS */  
    background-image: url(bg_page.jpg);  
    background-repeat: no-repeat;  
    background-position: center center;  
}
```



Annexe

CSS Font Properties

Property	Value	Example(s)
<i>font {value order sensitive}</i>	<i>mix</i> of following font-family font-size font-style font-variant font-weight	[style] [variant] [weight] size family name of font abs unit number % 12pt, medium, large, +1, 120% name of style keyword number strength 100, 200, light[er], normal, bold[er]

CSS Text Properties

Property	Value	Example(s)
letter-spacing	normal <length>	5em, 2pt, normal
line-height	normal <length> % number	20pt, 120%, normal
text-align	keyword	left, right, center, justify
text-decoration	keyword	underline, overline, line-through, blink
text-indent	number percentage	2in, 5%, 3em, .5cm, 5mm
text-transform	keyword	capitalize, lowercase, uppercase, none
vertical-align	keyword	top, middle, bottom, sub, super, baseline
white-space	keyword	normal, nowrap, pre
word-spacing	normal <length> inherit	5em, 2pt, normal

Annexe

CSS Background/Color Properties		
Property	Value	Example(s)
<i>background</i>	<i>mix</i> of following	red black/white URL(paper.gif)
background-attachment	keyword	fixed, scroll
background-color	color_name hex keyword	red, #FFCCFF, #FCF, transparent
background-image	valid url none	URL(paper.gif)
background-position	keyword	bottom right, top, center, left
background-repeat	keyword	repeat, repeat-x, repeat-y, no-repeat
<i>color {of text}</i>	color_name hexcode	red, gray, #FF00FF, #cf9

CSS Box Properties		
Property	Value	Example(s)
<i>border</i>	width style color	2px solid red
<i>margin</i>	keyword % abs unit	auto, 10%, 0.5in, 2em, 5mm
<i>padding</i>	keyword % abs unit	auto, 10%, 0.5in, 2em, 5mm
clear	keyword	left, right, none, both
float	keyword	left, right, none
height	keyword % abs unit	auto, 10%, 0.5in, 2em, 5mm
width	keyword % abs unit	auto, 10%, 0.5in, 2em, 5mm

Javascript

Introduction

- ▶ Langage de script
 - ▶ Interprété par le navigateur
 - ▶ Orienté objet à prototype
 - ▶ Faiblement typé
 - ▶ Multiplateforme
-
- ▶ Lancé par Netscape en 1995 (LiveScript)
 - ▶ Actuellement en version ECMASCIPT 6 (2015)
 - ▶ Standardisé par l'ECMA (European Computer Manufacturers Association)
 - ▶ Inspiré de Java et Python



JavaScript n'est pas Java !

JavaScript

- ▶ Interprété
- ▶ Langage à base d'objets, liste d'objets prédéfinis, pas d'héritage
- ▶ Code intégré dans HTML
- ▶ Typage faible
- ▶ Liaison dynamique des objets (runtime)
- ▶ Conçu pour les RIA*
- ▶ Accessibilité du code

Java

- ▶ Pseudo-compilé
- ▶ Langage orienté objets, héritage, définitions de classes
- ▶ Code dans applets
- ▶ Typage fort
- ▶ Liaison statique des objets (compilation)
- ▶ Conçu pour les RDA
- ▶ Confidentialité du code

Utiliser JavaScript

- ▶ Chargement du code JavaScript (balise <head>)
 - ▶ Depuis un fichier

```
<script language="Javascript" src="url/fichier.js"></script>
```

- ▶ Directement dans le document HTML

```
<script language="Javascript">
<!--
alert("Voici un message d'\alerte!");
// --
</script>
```

- ▶ Appel d'une fonction depuis un événement DOM

```
<input type="button" onclick="javascript:maFonction();">
<p>Ceci est un <a href="javascript:maFonction();">lien</a>.</p>
```

Do it yourself! via un shell javascript : <http://jsconsole.com>

Syntaxe

- ▶ Sensible à la casse (Case-sensitive)
- ▶ Les instructions se terminent par ; (semicolon)
- ▶ Commentaire : // ou /* */

```
0,c;c=za[b];b++)if(0(a,c))return c},Ga=function(a){a=a.relatedTarget;var b;a:{try{la(a.parentNode);b=f;break a}catch(c){}b=l}if(b)return a;r e&&g&&d.blur()}else b.type=="mouseout"&&L(c,a)}catch(h){z(h,"sb","moaoh")},Ha=function(a,b){if(a==b)return l;for(;b&&b!==a;)b=b.parentNode;p[2])while(c==0)return c},La=function(a,b){if(a<b) return -1;else if(a>b) return 1;return 0};var S=function(a){q("gbar.pcm",o(this.fa,this));I("gbmpsB");b&&H(b,"click",o(this.pa,this),f);if(c&&d){H(c,"click",o(this.U,this));H(d,"click",o(this.U,this))}this.F=f}if(!this.ja)if((b=I(Ia(c);g.appendChild(h);c.appendChild(d);c.appendChild(e);b.style.display=""))){if(a.xp){a=I("gbg4");b=I("gbg6");a&&H(a,"mouseUp");function(a){try{a=a||window.event;a.cancelBubble=f;a.stopPropagation&&a.stopPropagation();a.preventDefault&&a.preventDefault();var b=I("m.za=function(){var a=I("gbmpdv"),b=I("gbmps");if(a&&b){a.style.display="";b.style.display="none";if(!this.z){var c=I("gbmpal"),d=I("gbpm");m.ea=function(a,b,c,d,e,g,h,i){try{var k=I("gbmpas");if(k){var s="gbmtc";if(a)s+="gbmpmta";var w=R("div",s),x=R("div","gbmpfh");w.appendChild(this.ia.replace("%$s",e);D.appendChild(document.createTextNode(a));k.appendChild(w)}catch(M){z(M,"sp","aa")}};var Na=function(a,b){var c=I(S.prototype.C=function(){try{if(!this.T){this.T=f;var a=I("gbmpi");if(a&&this.w)a.src=this.w}catch(b){z(b,"sp","spp")}};S.prototype.Ba=function(){try{var var b=I("gbmpi"),c=I("gb4i");this.w=a(96);if(b.b.src=a(96);if(c.c.src=a(24)){try{var d=z(d,"sp","spp")}};S.prototype.pro e=d.join("");if(!window.gbar.logger._itl(e))return e}return b};v("er",init:function(){window.gbar.logger._aem=ab});var Ya=function(a){var Za=function(){for(var a=[],b=arguments.callee.caller,c=0;b&&c<20;){var d;d=(Function.prototype.toString.call(b).match(eb))?d[1]:"";var e=b;b;this.D=c;this.ya=d;this.P=e},$a=function(a){var b=[a.H?a.H+":","","",a.name?a.name:"anonymous",a.ya,a.D?" [ as "+a.D+"] ":""];if(a.P){b.push("a:{var rb="",V;if(T&&n.opera){var sb=n.opera.version;rb=typeof sb=="function"?sb():sb}else{if(ob)V=/rv\:([^:]+)(\)|/;else if(U)V=/MSIE\s+this))/a&&Q(a,o(this.qa,this));if(a=I("gbz")){d=o(this.Da,this>window.location.hostname);H(a,"mouseover",d)}if(document.body.style.MozTransform=d;k=R("div","gbmasc");Gb(this,k,i);i=k;i.id="gbm"+g;c.insertBefore(i,b);k=Hb(this,i).i.replaceChild(h,k)}this.c=undefined;r.ach("gbz",o(this.m.V=function(a){try{Aa(a);if(this.a&&this.h){clearTimeout(this.h);this.h=0;if(this.a){X(this);r.logger.il(30)}else{this.A(a.type,m.A=function(a){try{var b=a=="keydown";if(!this.a){var c=I("gbqlw"),d=I("gbz"),e=I("gbqla"),g=I("gbq1");if(d&&c&e&&g){var h=d.style;qa(j,c,X=function(a){try{if(a.a){Y(a);Mb(a,f);Ib(a);Jb(a);Y(a,0);var b=I("gbz"),c=I("gbqla");if(b&&c){K(b);var d=b.style,e=c.style;P(b,"qbzt")W.prototype.G=function(a){try{var b=a||window.event,c=(b.target||b.srcElement).id,d=b.shiftKey;if(b.keyCode==9&&!d)if(c=="gbztm")this.b||X(tW.prototype.ta=function(a){try{Nb(this,a);if(!this.t)try{if(!this.a&&!this.o){var b;var c=I("gbqlw");if(!c||c.clientHeight)b=500;else{var d=d/2)e=45}b=Math.min(500,1E3*d/e)}this.o=window.setTimeout(o(this,A,this,a.type),b);r.logger.il(28)}}catch(k){z(k,"as","oasad")}}catch(s){z(s var Ob=function(){var a=document.activeElement;if(a){var b=I("gbz"),c=I("gbd");if(b&&c){if(Ha(b,a)||Ha(c,a).a.blur())}Kb=function(a,b,c,d){c var Ib=function(a){clearTimeout(a.o);a.o=0;clearTimeout(a.h);a.h=0;clearTimeout(a.n);a.n=0},Jb=function(a,b){Mb(a,b);if(a.k){clearTimeout(a.W.prototype.Ha=function(a){try{Aa(a);if(this.j){if(this.b){this.b=2}else this.b=Y(this);this.o(a)}catch(b){z(b,"as","tmm")}};W.prototype.O=function(a){try{var b=I("gbz"),c=I("gbd");if(b&&c&&!this.b){Mb(this,f);var d=c.style,e=K(b).height-2,g=K(c).width;b={};b.height=var Y=function(a,b){var c=I("gbd");if(c&&a.b){Mb(a,f);var d=c.style,e=a.f;b!=undefined?b:400:0;a.f&&P(c,"gbdat");d.width="auto";d.overflow=W.prototype.B=function(){var a=I("gbz"),b=I("gbd");if(a&&b){var c=b.style;c.visibility=this.b?"visible":"hidden";c.overflow="";L(b,"gbdat");b:0};W.prototype.Da=function(a){try{if(!r.gpcc&&r.gpcc())&&!this.K){this.K=f;var b=a.indexOf("google");b>0&&Fb.set("OGP","",604800,"/",a.substring,Z.prototype.Fa=function(){try{var a=document.getElementById("gbqfqw");a&&L(a,"gbqfqw")}catch(b){z(b,"sf","stb")}};Z.prototype.Aa=function(){function(){y(t.Z);var a,b;for(a=0;b=r.bnc[a];++a){if(b[0]=="m")break;if(b&&b[1].l){a=function(){for(var c=r.mdc,d=r.mdi||[],e=0,g;g=ka[e];+f;y(t.Y);a:{for(c=0;d=r.bnc[c];++c){if((d[1].auto||d[0]=="m")&&!d[1].l){c=l;break}a.c=f}c&&y(t.W)};if(!b[1].libs||r.agl&&r.agl(b[1].libs))a()}};
```

Les variables

Les variables peuvent être déclarées :

- ▶ De façon **explicite** avec le mot clef var (portée locale)
- ▶ De façon **implicite**, sans mot clef (portée globale)

Les variables n'ont pas besoin d'être initialisées

- ▶ Le moteur leur affecte une valeur par défaut

Les variables n'ont pas de type (typage faible)

- ▶ Conversion implicite si nécessaire
- ▶ La chaîne de caractères est le type dominant

Une même variable peut contenir des valeurs de différent type dans le temps

Best practice:

- ▶ Utiliser la notation CamelCase

```
var myTestValue = 10;  
myTestValue = "une chaîne";  
myTestValue = 10.0;  
myTestValue = 9.5e+10;
```

Quelques types primitifs

Nombre (Number)

- ▶ entier (décimal, héxadécimal 0x, octale 0)
- ▶ réel

Booléen (Boolean) : true ou false

Chaîne de caractères (String)

- ▶ Les codes \t (tabulation) \n (à la ligne) \r (retour chariot) \b (backspace) \f (saut de page) sont reconnus

Types particuliers :

- ▶ Null (null),
- ▶ Undefined (undefined)
- ▶ Nombres particuliers: Infinity, NaN (Not a Number)

Conversion String / Number avec **parseInt()** et **parseFloat()**

Opérateurs, conditions, boucles

► Opérateurs :

- **Arithmétiques :** + - * / %
- **Conditionnels :** == != <= >= < > === !==
- **Logiques :** && || !
- **Autres :** += -= *= /= %= <<= >>= - ++ --

```
if (var1 == var2) { ... }
else { ... }
```

► Boucles :

```
for (i=0; i<5; i++) { ... }
for (x in someArray) { ... }

while (a < b) { ...}
do { ... } while (a<b)
```

Les fonctions

► Définition de fonction

```
function maFonction() { ... }
var maFonctionBis = function maFonction() { ... }
```

► « Lambda fonction »

- objet fonction
- stockable dans une variable
- peut se substituer à une autre fonction

```
link = document.createElement("a");

var onclick = function(num){ return function(){alert(num)} };
link.onclick = onclick(i);

link.onclick = function(){alert(i);};

link.onclick = (function(num){ return function(){alert(num)} })(i);
```

Notions d'Objet

- ▶ Pas de vraie classe au sens Java (pseudo-classe)
 - ▶ pas de sous-classe, ni d'héritage
- ▶ Créations d'objets + possibilité de définir des propriétés et des méthodes via des "prototype"
- ▶ propriétés (objet.propriété)
- ▶ méthodes (objet.methode())
- ▶ Création d'un objet par définition de son constructeur (fonction du nom de la pseudo-classe avec affectation des propriétés à partir des paramètres et déclaration des méthodes)
- ▶ Existence d'objets prédéfinis dans le langage

Les Objets JavaScript

- ▶ **Prédefinis (ECMAScript) :**
 - ▶ **Array** object, Boolean object, Date object, **Math** object, Number object, **String** object, RegExp object
- ▶ **Liés au navigateur :**
 - ▶ Window object, Navigator object, Screen object, History object, Location object
- ▶ **Liés au document HTML :**
 - ▶ **Document** object, Event object, HTMLElement object, Body object, Button object, Form object, Input Button/Text/... Object, Table object, ...

[JavaScript and HTML DOM Reference \(w3schools\)](#)

Objet String

- ▶ Propriétés :

- ▶ **length** : longueur de la chaîne

- ▶ Méthodes :

- ▶ **indexOf(searchstring, start)**
 - ▶ **substring(from, to), charAt(index)**
 - ▶ **lastIndexOf(string), toLowerCase()**
 - ▶ **toUpperCase(), split(), toString()**

- ▶ Exemples :

```
var chaine = "Bonjour";
chaine.indexOf("o");      // 1
chaine.lastIndexOf("o");// 4
chaine.charAt(3);        // j
chaine.substring(3,7);   // jour
chaine.toUpperCase();    // BONJOUR
```

Objet Math

- ▶ **Propriétés** (constantes mathématiques) :
 - ▶ E, PI, SQRT2, SQRT1_2, LN2, LN10, LOG2E, LOG10E
- ▶ **Méthodes (fonctions usuelles)** :
 - ▶ Trigo : cos(x), sin(x), tg(x), acos(x), atan(x), asin(x), atan2(x)
 - ▶ Valeur absolue : abs(x)
 - ▶ Entier INF : ceil(x); Entier SUP : floor(x); Entier + proche : round(x)
 - ▶ exp(x), log(x), sqrt(x), pow(x, a)
 - ▶ max(a, b), min(a, b)
 - ▶ Tirage aléatoire : random() (0 < résultat < 1)
- ▶ **Exemples** :

```
var tirage = Math.random(); // 0 < r < 1
var x = Math.PI;           // 3.14...
var y = Math.sqrt(16);     // 4
```

Objet Date

- ▶ **Propriétés :** ø
- ▶ **Méthodes** (essentiellement getter) :
 - ▶ getDate(), getDay(), getFullYear(), getHours(),
getMilliseconds(), getMinutes(), getMonth(), getSeconds(),
getTime() + setters
 - ▶ toLocaleDateString(), toLocaleTimeString(), toLocaleString()
 - ▶ parse(datestring)

- ▶ **Exemples :**

```
// Date courante
var today = new Date()
// Timestamp (since epoch 1970/01/01)
var d1 = new Date(timestamp)
// new Date(dateString)
var d1 = new Date("October 13, 1975 11:13:00")
// Date(year, month, day, hours, minutes, seconds, milliseconds)
var d3 = new Date(79,5,24,11,33,0)
```

Objet Array

- ▶ **Propriétés :**
 - ▶ length : taille du tableau
- ▶ **Méthodes (fonctions usuelles) :**
 - ▶ concat(array1, array2,...), indexOf(), join(separator), slice(start, end), splice(index, howMany)
 - ▶ **Stack** : pop(), push(), shix() unshix(),
 - ▶ **Tri** : sort(sortFunction), reverse()

▶ Exemples :

```
var arr = { "un": 1, "deux": 2, "trois": 3 }; // Tableau associatif
var myCars = new Array(); // new Array(3);
myCars[0] = "Saab";
myCars[1] = "Volvo";
myCars[2] = "BMW";
myCars = new Array("Saab", "Volvo", "BMW");
myCars = ["Saab", "Volvo", "BMW"];
console.log(myCars[0]); // Saab
```

Objet Window

- ▶ Propriétés :
 - ▶ closed, **onload**, location, navigator, status
- ▶ Méthodes (fonctions usuelles) :
 - ▶ close(), open(), focus()
 - ▶ createPopup(), prompt(), alert()
 - ▶ moveBy(), moveTo(), scrollBy(), **scrollTo()**

- ▶ Exemples :

```
while(true) {window.alert("Salut!");} // à éviter...
window.open("http://www.google.com/");
var saisie = prompt("Saisissez votre texte :", "Texte par défaut");
```

Objet Document

- ▶ Représente le DOM de la page courante
- ▶ **Propriétés :**
 - ▶ domain, referrer, title, URL, ...
- ▶ **Méthodes :**
 - ▶ *getElementById(idAsString)*] Retourne un objet HTMLElement
 - ▶ *getElementsByName(nameAsString)*] Retourne un tableau de HTMLElement
 - ▶ *getElementsByTagName(tagNameAsString)*]
 - ▶ *write(string)*
 - ▶ *writeln(string)*
- ▶ **Exemples :**

```
// <h1 id="myHeader" onclick="getValue()">Click me!</h1>
var x1 = document.getElementById("myHeader");
// <input name="x" type="text" size="20" />
var x2 = document.getElementsByName("x");
// <input type="text" size="20" />
var x3 = document.getElementsByTagName("input");
```

Objet HTMLElement

- ▶ Représente un objet du DOM
- ▶ **Propriétés :**
 - ▶ `disabled`, `id`, `innerHTML`, `lastChild`, `firstChild`, `nodeName`, `nodeType`,
`nodeValue`
- ▶ **Méthodes :**
 - ▶ `insertBefore(newChild)`, `appendChild(newChild)`
 - ▶ `focus()`, `blur()`, `click()`,
 - ▶ `getElementsByTagName(name)`
 - ▶ `removeChild(childToRemove)`, `replaceChild(childToReplace)`
 - ▶ `getAttribute(attrName)`, `setAttribute(attrName)`,
`removeAttribute(attrName)`

Fil rouge

TODO LIST

Première étape

- ▶ Réaliser la structure en HTML5 (`<!DOCTYPE html>`)
- ▶ Utiliser des balises de mise en forme (`<h1>, <h2>`)
- ▶ Utiliser des listes (`, `)
- ▶ Utiliser des tableaux (`<table>, <tr>, <td>`)
- ▶ Utiliser des champs de formulaire :
 - ▶ Type `text` : `<input type="text">`
 - ▶ Type `button` : `<input type="button">`
 - ▶ Type `checkbox` : `<input type="checkbox">`

Ma TODO list !

Menu

- Afficher la TODO list d'une personne
- Ajouter un TODO

Mes TODO

- Faire les courses
- Nourrir le chat
- Passer boire une bière chez Farid !

Deuxième étape

- ▶ Ajouter une feuille de style CSS au document



Troisième étape

- ▶ Rendre les boutons actifs :
 - ▶ Ajout d'un TODO via le bouton « Ajouter »
 - ▶ Suppression de TODO via le bouton « Supprimer la sélection »
 - ▶ Effacer toutes les TODO via le bouton « Tout effacer »
- ▶ Appel de fonction JavaScript bindé sur l'événements DOM « onclick ».
- ▶ Utiliser les Objets document et HTMLElement pour manipuler le DOM

Ma TODO list !

Menu

- Afficher la TODO list d'une personne
- Ajouter un TODO

Mes TODO

- Faire les courses
- Nourrir le chat
- Passer boire une bière chez Farid !

Solution – Première étape

```
<!DOCTYPE html>
<html>
<head>
    <title>Ma TODO list !</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
    <h1>Ma TODO list !</h1>
    <h2>Menu</h2>
    <ul>
        <li>Afficher la TODO list d'une personne<br>
            <input type="text" id="personId" size="20">
            <input type="button" value="Afficher">
        </li>
        <li>Ajouter un TODO<br>
            <input type="text" id="newTodo" size="20">
            <input type="button" value="Ajouter">
        </li>
    </ul>
    <h2>Mes TODO</h2>
    <ul style="list-style-type:none;">
        <li><input type="checkbox"> Faire les courses</li>
        <li><input type="checkbox"> Sortir le chat</li>
        <li><input type="checkbox"> Passer prendre une bière chez Farid
    </li>
    </ul>
    <p><input type="button" value="Supprimer"> la sélection</p>
</body>
</html>
```

Solution – Deuxième étape (styles.css)

```
body {  
    background-color: #eee;  
    font-family: Arial;  
    margin:0; padding:0;  
}  
  
h1, h2 {  
    font-family: "Trebuchet MS";  
    margin: 5px 0px 5px 0px;  
    padding: 5px 10px 5px 10px;  
}  
  
h1 {background-color: #000; color: #fff;}  
h2 {background-color: #444; color: #fff;}  
  
div {  
    font-size: 10pt;  
    padding-left: 20px;  
}  
  
ul {  
    list-style-type:none;  
    padding-left: 0px;  
}  
  
ul > li {  
    background-color: #ddd;  
    margin: 2px; padding: 2px;  
}
```

Solution – Troisième étape (index.html)

```
<!DOCTYPE html>
<html>
<head>
    <title>Ma TODO list !</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <script language="javascript" src="functions.js"></script>
</head>
<body>
    <h1>Ma TODO list !</h1>
    <h2>Menu</h2>
    <ul>
        <li>Afficher la TODO list d'une personne<br>
            <input type="text" id="personId" size="20" disabled>
            <input type="button" value="Afficher" disabled>
        </li>
        <li>Ajouter un TODO<br>
            <input type="text" id="todoMessage" size="20">
            <input type="button" value="Ajouter"
onclick="javascript:addTodo();">
        </li>
    </ul>
    <h2>Mes TODO</h2>
    <ul id="todoList"></ul>
    <p><input type="button" value="Supprimer la sélection"
onclick="javascript:removeSel();">
        <input type="button" value="Tout effacer"
onclick="javascript:removeAll();"></p>
</body>
</html>
```

Solution – Troisième étape (functions.js)

```
// Variable globale
var counter = 0;

function addTodo() {
    // Récupération du message
    var input = document.getElementById("todoMessage");
    var todoMessage = input.value;
    // On efface le champs texte
    input.value = "";
    // On récupère l'élément liste
    var listContainer = document.getElementById("todoList");
    // Creation du nouvel élément de la liste
    var newElement = document.createElement("li");
    var newElementId = counter++;
    newElement.setAttribute("id", newElementId);
    newElement.innerHTML = "<input type='checkbox' name='todoCheck' value='"
        + newElementId + "'> " + todoMessage;
    // Ajout du nouvel élément ,à ö, Ät la liste
    listContainer.appendChild(newElement);
}
```

Solution – Troisième étape (functions.js)

```
function removeAll() {
    // On récupère l'élément liste
    var listContainer = document.getElementById("todoList");
    // On vide son contenu
    listContainer.innerHTML = "";
}

function removeSel() {
    // On récupère les éléments du DOM
    var listContainer = document.getElementById("todoList");
    var checkboxes = document.getElementsByName("todoCheck");
    // On boucle sur chaque élément
    for(var i = checkboxes.length-1; i >= 0; i--) {
        // Si la checkbox est cochée...
        if(checkboxes[i].checked) {
            // On supprime la ligne !
            var todo = document.getElementById(checkboxes[i].value);
            todo.parentNode.removeChild(todo);
        }
    }
}
```