

HTTP, HTML, CSS

kn@lri.fr
http://www.lri.fr/~kn

1 HTTP, HTML, CSS

1.1 Rappels

1.2 Internet et ses services

1.3 Fonctionnement du Web

1.4 Adressage des documents Web

1.5 Le protocole HTTP

1.6 UTF-8

1.7 HTML, le format des documents

1.8 Vue arborescente des documents

1.9 CSS : Boîtes

1.10 Sélectionneurs et application en cascade

Concepts de réseau informatique

Réseau : ensemble de **nœuds** reliés entre eux par des **liens** (ou canaux).

Réseau informatique : réseau où les nœuds sont des **ordinateurs**. Les liens sont **hétérogènes** (câbles, liaisons radio, liaisons satellites, ...)

Protocole : ensemble de **conventions** permettant d'établir une communication mais **qui ne font pas partie** du sujet de la communication.

Rappels sur le modèle TCP/IP

◆ Modèle organisé en 4 couches, avec 1 protocole par couche :

N	Nom	Description	Eq. OSI
4	<i>Application</i>	HTTP, BitTorrent, FTP, ...	5, 6, 7
3	<i>Transport</i>	TCP, UDP, SCTP, ...	4
2	<i>Internet</i>	IP (v4, v6), ICMP, IPsec, ...	3
1	<i>Liaison</i>	Ethernet, 802.11, ...	1, 2

◆ Les machines sont identifiées par leur(s) adresse(s) IP

◆ En IP version 4, les adresses sont constituées de 4 octets, représentés en décimal :
129.175.28.179

◆ Le protocole IP se charge du **routage** la transmission des données entre deux machines (en passant par des machines intermédiaires)

◆ Le protocole TCP utilise IP pour fournir une communication **bi-directionnelle, sans perte et avec garantie d'envoi/reception dans l'ordre**

- ◆ Modèle OSI (à 7 couches, plus « formel » que TCP/IP)
- ◆ Comment identifier les « voisins » d'un nœud (le sous réseau local) : classe d'adresses (obsolète) et masques de sous réseau
- ◆ Principes généraux du routage
- ◆ Fragmentation et réassemblage des paquets
- ◆ Épuisement du nombre d'adresses (survol d'IPv6)
- ◆ Communications en TCP
- ◆ ...

5 / 90

Plan

- 1 HTTP, HTML, CSS
 - 1.1 Rappels ✓
 - 1.2 Internet et ses services
 - 1.3 Fonctionnement du Web
 - 1.4 Adressage des documents Web
 - 1.5 Le protocole HTTP
 - 1.6 UTF-8
 - 1.7 HTML, le format des documents
 - 1.8 Vue arborescente des documents
 - 1.9 CSS : Boîtes
 - 1.10 Sélectionneurs et application en cascade

Domain Name System : permet d'attribuer un nom à une IP (annuaire). Double avantage :

- ◆ pour les humains, un nom est plus simple à retenir
- ◆ on peut changer d'adresse IP de manière silencieuse

Principe hiérarchisé :

- ◆ les serveurs DNS de premier niveau gardent les informations sur les TLD (Top Level Domain, domaine de premier niveau) : .com, .fr, .net, ...)
- ◆ pour chaque tld, il y a un ensemble de serveur DNS de niveau 2 qui fait correspondre le nom de domaine (google.com, u-psud.fr) à un DNS de niveau 3 (généralement le DNS de niveau 2 est chez le FAI)
- ◆ le DNS de niveau 3 donne l'IP d'une machine particulière sur son domaine : mail, www (le DNS de niveau 3 est administré localement)

6 / 90

Bref historique d'Internet (1/2)

- 1959-1968 :** ARPA (*Advanced Research Project Agency*) crée un réseau de quelques machines capable de résister à une attaque.
- 1969 :** ARPANET. Interconnexion des ordinateurs centraux des grandes universités et institutions américaines. Première utilisation du concept de paquet d'information.
- 1970-1982 :** Interconnexion avec la Norvège et le Royaume-Uni.
- 1982 :** Généralisation du protocole TCP/IP. Naissance de l'Internet actuel.

8 / 90

- 1986 :** « Autoroutes de l'information ». Des super-ordinateurs et les premières connexions à fibres optiques sont utilisées pour accélérer le débit d'Internet.
- 1987-1992 :** Apparition des premiers fournisseurs d'accès. Les entreprises se connectent.
- 1993-2000 :** Avènement du Web. Démocratisation du haut débit (vers 2000 pour la France).
- 2000-? :** Explosion des services en ligne, arrivée des réseaux sociaux, internet mobile, *Cloud* (stockage et calcul mutualisés accessible depuis internet).

On établit une connexion TCP, sur le port 4444 entre notre machine et la machine dont l'IP est 149.56.108.199.

La commande Unix `netcat` possède deux mode de fonctionnement :

- ◆ `netcat -l 4444` : se met en attente de connexion sur le port 4444. Toutes les données reçues sont écrites sur la sortie, toutes les données entrées sont envoyées au client connecté.
- ◆ `netcat 149.56.108.199 4444` se connecte à un serveur en TCP sur le port 4444. Gestion des lectures/écritures identiques.

- ◆ Ensemble de logiciels et protocoles basés sur *TCP/IP*
- ◆ Les applications suivent un modèle Client/Serveur
- ◆ Un serveur fournit un service:
 - ◆ courriel
 - ◆ transfert de fichier (ftp)
 - ◆ connexion à distance (ssh)
 - ◆ Web (http)
- ◆ Plusieurs services peuvent être actifs sur la même machine (serveur). Un *port (identifiant numérique)* est associé à chaque service. Sur Internet, un service est identifié par:
 - ◆ L'adresse IP de la machine sur laquelle il fonctionne
 - ◆ Le numéro de port sur lequel le programme attend les connexions

- ◆ Service de distribution de page *hypertexte*
- ◆ Une page *hypertexte* contient des références immédiatement accessibles à d'autres pages (pointeurs ou *liens hypertextes*)
- ◆ Les pages sont décrites dans le langage *HTML* (HyperText Markup Language)
- ◆ Architecture client/serveur:
 - ◆ Les pages sont stockées sur le serveur
 - ◆ Les pages sont envoyées au client (navigateur Web) qui en assure le rendu
- ◆ Utilise le protocole *HTTP* pour les échanges entre le client et le serveur

Concepts clé:

- URL :** localisation d'une page Web (« adresse de la page »)
- HTTP :** protocole de communication entre un client et un serveur Web
- HTML :** langage de description des pages Web

Évolutions récentes (Web 2.0, internet mobile, *Cloud*, ...)

- ◆ Standardisation du contenu multimédia (images, vidéos et sons en *streaming*)
- ◆ Contenu interactif avancé (stockage de fichier coté client, rendu 3D, ...)
- ◆ Uniformisation de nombreuses extensions *ad-hoc*: HTML5

1 HTTP, HTML, CSS

1.1 Rappels ✓

1.2 Internet et ses services ✓

1.3 Fonctionnement du Web

1.4 Adressage des documents Web

1.5 Le protocole HTTP

1.6 UTF-8

1.7 HTML, le format des documents

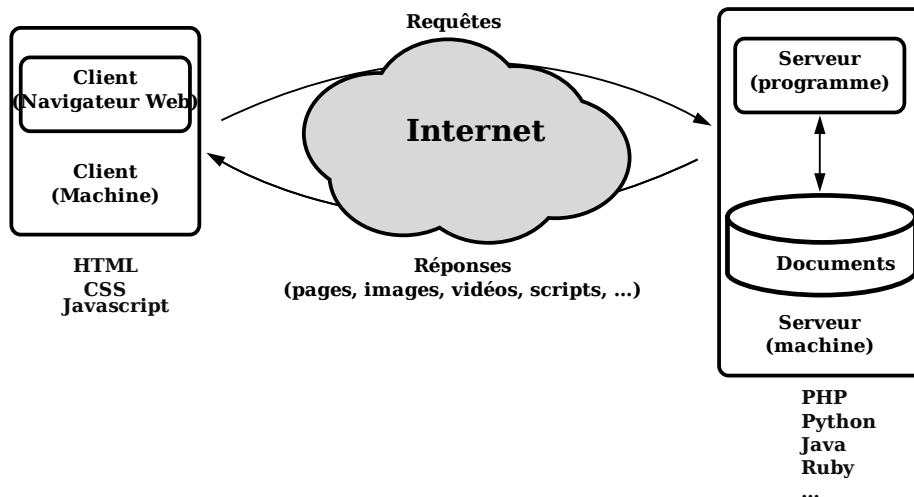
1.8 Vue arborescente des documents

1.9 CSS : Boîtes

1.10 Sélectionneurs et application en cascade

13 / 90

Fonctionnement du Web



15 / 90

Côté client

Le navigateur :

- ◆ Analyse l'URL demandée
- ◆ Obtient l'adresse IP auprès du serveur DNS
- ◆ Établit une connexion (potentiellement sécurisée) avec le serveur
- ◆ Envoie une *requête HTTP* au serveur
- ◆ Récupère la page envoyée par le serveur dans sa *réponse*
- ◆ Analyse la page et récupère les éléments référencés : images, sons, ...
- ◆ Effectue le traitement du code client
- ◆ Met en forme le contenu et l'affiche dans la fenêtre

16 / 90

- ◆ Un *serveur Web* attend les connexions sur un port par défaut (80 dans le cas de HTTP)
- ◆ À chaque nouvelle connexion, le serveur crée un nouveau *sous-processus* de traitement et se remet en attente
- ◆ Le sous-processus de traitement vérifie la validité de la requête :
 - ◆ le document demandé existe ?
 - ◆ le client est autorisé à accéder au document ?
 - ◆ ...
- ◆ Le sous-processus de traitement répond à la requête :
 - ◆ Exécution de code côté serveur, récupération de données dans une BDD, ...
 - ◆ Envoi de la page au client

Adressage des documents Web (1/3)

URL : *Uniform Resource Locator* identifie un document sur internet

Une URL se décompose en 3 parties

- ◆ *protocole* (comment ?)
- ◆ *adresse* (où ?)
- ◆ *document* (quoi ?)

Syntaxe (simplifiée) :

protocole://*adresse*/*document*

Exemple :

http://www.lri.fr/~kn/teach_fr.html

- 1 HTTP, HTML, CSS
 - 1.1 Rappels ✓
 - 1.2 Internet et ses services ✓
 - 1.3 Fonctionnement du Web ✓
 - 1.4 Adressage des documents Web
 - 1.5 Le protocole HTTP
 - 1.6 UTF-8
 - 1.7 HTML, le format des documents
 - 1.8 Vue arborescente des documents
 - 1.9 CSS : Boîtes
 - 1.10 Sélectionneurs et application en cascade

Adressage des documents Web (2/3)

On peut aussi préciser un **numéro de port**, des paramètres et un **emplacement** :

protocole://*adresse*:*port*/*document*?*p1*=*v1*&*p2*=*v2*#*emp1*

Exemple :

http://www.youtube.com:80/results?search_query=cat#search-results

Le serveur utilise les paramètres passés par le client dans l'URL pour *calculer* le contenu de la page (changer la chaîne « cat » ci-dessus et essayer)

La *racine* d'un site Web (ex: `http://www.lri.fr/`) correspond à un répertoire sur le disque du serveur (ex: `/var/www`). Le fichier

```
http://www.lri.fr/index.html
```

se trouve à l'emplacement

```
/var/www/index.html
```

Le serveur Web peut aussi effectuer des *réécritures d'adresses* :

```
http://www.lri.fr/~kn/index.html
```

devient

```
/home/kn/public_html/index.html
```

1 HTTP, HTML, CSS

1.1 Rappels ✓

1.2 Internet et ses services ✓

1.3 Fonctionnement du Web ✓

1.4 Adressage des documents Web ✓

1.5 Le protocole HTTP

1.6 UTF-8

1.7 HTML, le format des documents

1.8 Vue arborescente des documents

1.9 CSS : Boîtes

1.10 Sélectionneurs et application en cascade

21 / 90

Caractéristiques du protocole HTTP

Format des messages HTTP

- ◆ Sans connexion permanente:
 - ◆ Le client se connecte au serveur, envoie sa requête
 - ◆ Le serveur envoie sa réponse et ferme la connexion
- ◆ Indépendant du contenu : permet d'envoyer des documents (hyper) texte, du son, des images, ...
- ◆ Sans *état*: chaque paire requête/réponse est indépendante (le serveur ne maintient pas d'information sur le client entre les requêtes)
- ◆ Protocole en mode *texte*

Les messages ont la forme suivante

- ◆ Ligne initiale CR LF
- ◆ zéro ou plusieurs lignes d'option CR LF
- ◆ CR LF
- ◆ Corp du message (document envoyé, paramètres de la requête, ...)
- ◆ *Requête* la première ligne contient un nom de *méthode* (GET, POST, HEAD, ...), le paramètre de la méthode et la version du protocole
- ◆ *Réponse* la version du protocole, le code de la réponse (200, 404, 403, ...) et un message informatif

Remarque : dans les chaînes de caractères Python (ainsi que dans d'autres langages) le caractère CR (*carriage return*) s'obtient par `\r` et le caractère LF (*line feed*) par `\n`.

23 / 90

24 / 90

- ◆ On réutilise le programme netcat pour explorer un peu le protocole HTTP
- ◆ On utilise d'abord netcat -l 4444 et on s'y connecte avec un navigateur Web
- ◆ En suite, on utilise netcat idea.nguyen.vg 80 pour récupérer une page.

25 / 90

Point méthode

- ◆ On peut faire des exercices débranchés sur la partie IP (voire TCP)
- ◆ Faire des TPs est super intéressant (sous Unix avec netcat ou en Python, cf. feuille d'exercices)
- ◆ Les TPs sont très très dépendants de la configuration réseau (présence d'un pare-feu, obligation de passer par un proxy, ...)
- ◆ Les TPs « Système et Réseau » doivent être fait sous idle (ou avec un autre éditeur) et le programme lancé localement. Ils fonctionneront difficilement avec un environnement de type jupyter

27 / 90

◆ Client :

```
GET /test.html HTTP/1.1
Host: idea.nguyen.vg:80
```

◆ Serveur :

```
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 13 Jun 2019 18:59:49 GMT
Content-Type: text/html
Content-Length: 128
Last-Modified: Wed, 12 Jun 2019 10:15:01 GMT
Connection: close
ETag: "5d00d0a5-80"
Accept-Ranges: bytes
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Une page de test</title>
</head>
<body>
```

} ← code de retour

} ← type de contenu

} ← longueur du contenu

} ← contenu (128 octets)

26 / 90

Plan

1 HTTP, HTML, CSS

1.1 Rappels ✓

1.2 Internet et ses services ✓

1.3 Fonctionnement du Web ✓

1.4 Adressage des documents Web ✓

1.5 Le protocole HTTP ✓

1.6 UTF-8

1.7 HTML, le format des documents

1.8 Vue arborescente des documents

1.9 CSS : Boîtes

1.10 Sélectionneurs et application en cascade

Avant de représenter des documents complexes, on s'intéresse aux textes (sans structure particulière)

Problématique: comment représenter du texte réaliste ?

Exemple de texte réaliste:

" و عليكم السلام ,Здравей,¡Hola!, 你好,Góðan daginn,... "

29 / 90

... et pendant ce temps là, ailleurs dans le monde

Encodage multi-octets:

- ◆ Encodages spécifiques pour le Chinois (Big5, GB, ...)
- ◆ Encodages spécifiques pour le Japonais (Shift-JIS, EUC, ...)

Impossibilité de mettre plusieurs « alphabets » dans un même texte

Chaque logiciel « interprétait » les séquences d'octet de manière prédéfinie

31 / 90

Encodage 1 caractère = 1 octet (8 bits) :

- ◆ Encodage ASCII sur 7 bits (128 caractères)
- ◆ ASCII étendu 8 bits (256 caractères, dont 128 de « symboles »)
- ◆ Latin 1 : ASCII 7 bits + 128 caractères « ouest-européens » (lettres accentuées française, italienne, ...)
- ◆ Latin 2 : ASCII 7 bits + 128 caractères « est-européens » (Serbe, Hongrois, Croate, Tchèque, ...)
- ◆ Latin 3 : ASCII 7 bits + 128 caractères turques, maltais, espéranto,
- ◆ Latin 4 : ASCII 7 bits + 128 caractères islandais, lituanien, ...
- ◆ ...
- ◆ Latin 15 : Latin 1 avec 4 caractères « inutiles » remplacés (par exemple pour « € » à la place de « ¤ »)

30 / 90

UTF-8

Universal (Character Set) Transformation Format 8 bit

- ◆ Un organisme (ISO) donne un code à chaque symbole. C'est le standard Unicode (la version actuelle, 12.1 référence 137 994 symboles).
- ◆ Encodage à taille variable « universel » (contient tous les alphabets connus)
- ◆ Compatible avec ASCII 7 bits

Encodage

Nombre d'octets	Octet 1	Octet 2	Octet 3	Octet 4
1	0xxxxxxx			
2	110xxxxx	10xxxxxx		
3	1110xxxx	10xxxxxx	10xxxxxx	
4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

32 / 90

A → 65₁₀ → 0100 0001₂ (représenté sur un seul octet)

ë → 7877₁₀ → 0001 1110 1100 0101₂ (représenté 3 octets) :

11100001 1011 10 11 1000 0101 ≡ 225 187 133



→ 128053₁₀ → ... ≡ 240 237 220 181

Avantages :

- ◆ compatible ASCII 7 bits (d'anciens documents texte en anglais sont toujours lisibles)
- ◆ pas d'espace gaspillé (à l'inverse d'UTF-32 ou tous les caractères font 32 bits)

Inconvénients

- ◆ Caractères à taille variable: il faut parcourir le texte pour trouver le n^{ème} caractère
- ◆ Les vieux logiciels doivent être adaptés

33 / 90

Encodage/Décodage

```
>>> mot = "éléphant"
>>> len(mot)
8

>>> octets = mot.encode()
>>> octets
b'\xc3\xa9\xc3\x9c\xc3\xa9\xc3\xphant'

>>> len(octets)
10

>>> octets.decode()
'éléphant'

>>> invalide = b"\xff\xff"
>>> invalide.decode()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 0:
invalid start byte
```

35 / 90

Les chaînes de caractères sont encodées en *UTF-8*. La fonction `len(...)` renvoie le nombre de caractères dans la chaîne (et non pas le nombre d'octets).

Comme on veut parfois manipuler des séquences d'octets arbitraires (par exemple 255 255 255) qui peuvent ne pas être des séquences UTF-8 valide, Python propose le type `bytes`. Il utilise la même syntaxe que les chaînes mais :

- ◆ Elle est préfixée d'un `b`
- ◆ Elle ne contient que des caractères ASCII 7 bits (pas d'accent en particulier). Les caractères non directement représentables sont échappés et leur code est donné en hexadécimal :
- ◆ La méthode `.encode()` sur une chaîne permet de convertir en `bytes`(fonctionne toujours).
- ◆ La méthode `.decode()` sur un `bytes` permet de convertir en chaîne

Attention, beaucoup de fonctionnalités « système » (et réseau en particulier) utilisent des `bytes`

34 / 90

Plan

- 1 HTTP, HTML, CSS
 - 1.1 Rappels ✓
 - 1.2 Internet et ses services ✓
 - 1.3 Fonctionnement du Web ✓
 - 1.4 Adressage des documents Web ✓
 - 1.5 Le protocole HTTP ✓
 - 1.6 UTF-8 ✓
 - 1.7 HTML, le format des documents
 - 1.8 Vue arborescente des documents
 - 1.9 CSS : Boîtes
 - 1.10 Sélectionneurs et application en cascade

HyperText Markup Language : langage de mise en forme de documents hypertextes (texte + liens vers d'autres documents). Développé au CERN en 1989.

1991 : premier navigateur en mode texte

1993 : premier navigateur graphique (mosaic) développé au NCSA (National Center for Supercomputing Applications)

Historique du langage HTML

1973 :	GML, Generalised Markup Language développé chez IBM. Introduction de la notion de balise.
1980 :	SGML, Standardised GML, adopté par l'ISO
1989 :	HTML, basé sur SGML. Plusieurs entreprises (microsoft, netscape, ...) interprètent le standard de manière différente
1996 :	XML, eXtensible Markup Language norme pour les documents semi-structurés (SGML simplifié)
2000 :	XHTML, version de HTML suivant les conventions XML
2008 :	Première proposition pour le nouveau standard, HTML5
2014 :	Standardisation de HTML 5.0
2017 :	Standardisation de HTML 5.2

- ◆ est un document *semi-structuré*
- ◆ dont la structure est donnée par des *balises*

Exemple	Rendu par défaut
Un texte <code>en gras</code>	Un texte en gras
<code>Un lien</code>	Un lien
<code> Premièrement Deuxièmement </code>	<ul style="list-style-type: none"> ◆ Premièrement ◆ Deuxièmement

On dit que `<toto>` est une balise *ouvrante* et `</toto>` une balise *fermante*. On peut écrire `<toto/>` comme raccourci pour `<toto></toto>`.

XHTML vs HTML

Différences avec HTML:

- ◆ Les balises sont *bien parenthésées* (`<div>` `` `` `</p>` est interdit)
- ◆ Les balises sont en minuscules

Les avantages sont les suivants

- ◆ HTML autorise les mélanges majuscule/minuscule, de ne pas fermer certaines balise ... Les navigateurs corrigent ces erreurs de manières *différentes*
- ◆ Le document est *structuré* comme un programme informatique (les balises ouvrantes/fermantes correspondent à { et }). Plus simple à débogger.

Convention pour le cours

Afin d'être compatible à la fois XHTML et HTML5, on utilisera les conventions suivantes :

- ◆ Les balises suivantes et celle-ci **uniquement** doivent ne pas avoir de contenu :

area, base, br, col, command, embed, hr, img, input, keygen, link, meta, param, source, track, wbr

Exemple :

```
<img src='toto.jpg' alt='une image' />
```

Toutes les autres balises doivent **obligatoirement** être de la forme:

```
<toto ... > ... </toto>
```

- ◆ Les noms de balises sont toujours en **minuscule**
- ◆ Le *doctype* (balise spéciale indiquant le type de document) est :

```
<!DOCTYPE html>
```

41 / 90

Exemple de document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Un titre</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <h1>Titre de section</h1>
    <p> premier paragraphe de texte. On met
    un <a href="http://www.lri.fr">lien</a> ici.
    </p>
    <!-- on peut aussi mettre des commentaires -->
  </body>
</html>
```

43 / 90

Rôle d'HTML

Séparer la *structure* du document de son *rendu*. La structure donne une *sémantique* au document :

- ◆ ceci est un titre
- ◆ ceci est un paragraphe
- ◆ ceci est un ensemble de caractères importants

Cela permet au navigateur d'assurer un rendu en fonction de la sémantique. Il existe différents types de rendus:

- ◆ graphique interactif (Chrome, Firefox, Internet Explorer, ...)
- ◆ texte interactif (Lynx, navigateur en mode texte)
- ◆ graphique statique (par ex: sur livre électronique)
- ◆ rendu sur papier
- ◆ graphique pour petit écran (terminal mobile)

42 / 90

Structure d'un document HTML

Pour être *valide* un document HTML contient **au moins** les choses suivantes :

- ◆ Une indication <!DOCTYPE html> en début de fichier
- ◆ Une balise <html> qui est la **racine** (elle englobe toutes les autres balises). La balise html contient deux balises filles: <head> et <body>
- ◆ La balise <head> représente l'en-tête du document. Elle peut contenir diverses informations (feuilles de styles, titre, encodage de caractères, ...). La seule balise **obligatoire** dans head est le titre (title). C'est le texte qui est affiché dans la barre de fenêtre du navigateur ou dans l'onglet.
- ◆ Il faut aussi souvent ajouter une balise <meta charset="utf-8" /> dans la balise head (si le navigateur n'a pas pu obtenir l'information par un autre moyen, par exemple dans les en-tête de la requête HTTP)
- ◆ la balise body représente le contenu de la page. On y trouve diverses balises (div, p, table, ...) qui formatent le contenu de la page

44 / 90

Titres

Les balises `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, permettent de créer des titres de section, sous-section, sous-sous-section, ...

Titre de niveau 1

Titre de niveau 2

Titre de niveau 3

Titre de niveau 4

Titre de niveau 5

Titre de niveau 6

45 / 90

Mise en forme du texte

Les balises `` (*bold*, gras), `<i>` (*italic*, italique), `<u>` (*underlined*, souligné) `` (*emphasis*, important) et beaucoup d'autres permettent de décorer le texte.

```
<p><b>Lorem ipsum dolor</b> sit amet, consectetur  
adipiscing elit, sed do eiusmod tempor incididunt ut labore et  
dolore magna aliqua. <u>Ut enim ad minim veniam</u>, <em>quis</em> nostrud  
exercitation ullamc</p>  
<p><i>Nouveau</i> paragraphe</p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamc

Nouveau paragraphe

47 / 90

Paragrapes

Des paragraphes de textes sont introduits avec les balises `<p>`. Par défaut chaque paragraphe implique un retour à la ligne:

```
<p>Lorem ipsum dolor sit amet, consectetur  
adipiscing elit, sed do eiusmod tempor incididunt ut labore et  
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud  
exercitation ullamc</p>  
<p>Nouveau paragraphe</p>
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamc

Nouveau paragraphe

Remarque : par défaut, les espaces, retour à la ligne, ... sont ignorés et le texte est reformaté pour remplir la largeur de la page.

46 / 90

Tableaux

On peut formater des tables en utilisant :

- ◆ La balise `<table>` pour délimiter la table
- ◆ La balise `<tr>` pour délimiter une ligne de la table
- ◆ La balise `<th>` pour délimiter une tête de colonne
- ◆ La balise `<td>` pour délimiter une case
- ◆ L'attribut `colspan` permet de fusionner des colonnes

```
<table>  
<tr> <th>Nom</th> <th>Prénom</th> <th>Note 1</th> <th>Note 2</th></tr>  
<tr> <td>Foo</td> <td>Bar</td> <td> 15</td> <td>12</td> </tr>  
<tr> <td>Doe </td> <td>Jonh</td> <td colspan="2">Absent</td></tr>  
</table>
```

Nom	Prénom	Note 1	Note 2
Foo	Bar	15	12
Doe	Jonh	Absent	

48 / 90

Listes

On peut créer des listes énumérées (avec ``, *ordered list*) ou non énumérées (avec ``, *unordered list*). Chaque ligne est limitée par une balise `` (*list item*)

```
<ul>
  <li> Un élément </li>
  <li> <ol> <li> Un autre élément </li>
    <li> <ol> <li> Un sous-élément</li>
      <li> Un autre sous-élément</li>
    </ol>
  </li>
</ol>
<li>Le dernier</li>
</ul>
```

- Un élément
- 1. Un autre élément
- 2. 1. Un sous-élément
2. Un autre sous-élément
- Le dernier

49 / 90

Remarques générales

- ◆ On n'a normalement pas le droit de mettre n'importe quel élément n'importe où (i.e. pas de `` tout seul)
- ◆ Il existe une spécification précise de HTML 5 (plusieurs dizaines de pages uniquement pour les balises)
- ◆ Il existe aussi des validateurs, il faut les utiliser le plus possible
- ◆ De manière générale, les espaces sont ignorés, on prendra donc bien soit de les utiliser judicieusement pour rendre le code de la page lisible
- ◆ Tous les éléments ont un style (moche) par défaut. On verra comment modifier ce style grâce à des propriétés CSS.

51 / 90

Liens hyper-texte

On peut faire référence à une autre **ressource** en utilisant un lien hyper-texte (balise `<a/>` et son attribut `href`). La cible du lien peut être absolue (une URL complète avec le protocole, par exemple `https://www.lri.fr`) ou relative (par exemple `foo.html`). Si l'URL est relative, le chemin est substitué à la dernière composante de l'URL de la page courante. Si l'URL commence par un `#` elle référence, l'attribut `id` d'un élément de la page:

```
<a href="https://www.lri.fr">Le LRI</a>
<a href="../../../../index.html">Un lien</a>
<a href="#foo">On va vers le titre</a>
...
<h1 id="foo">Le titre</h1>
```

[Le LRI](#) [Un lien](#) [On va vers le titre](#) ...

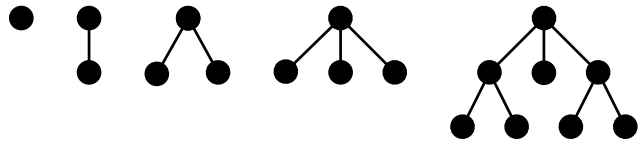
50 / 90

Plan

- 1 HTTP, HTML, CSS
 - 1.1 Rappels ✓
 - 1.2 Internet et ses services ✓
 - 1.3 Fonctionnement du Web ✓
 - 1.4 Adressage des documents Web ✓
 - 1.5 Le protocole HTTP ✓
 - 1.6 UTF-8 ✓
 - 1.7 HTML, le format des documents ✓
 - 1.8 Vue arborescente des documents
 - 1.9 CSS : Boîtes
 - 1.10 Sélectionneurs et application en cascade

Mots bien parenthésés et arbres

[] [[]] [[[]]] [[[[]]]] [[[[[]]]]]



Différentes types de paires de parenthèses = différentes étiquettes pour les nœuds de l'arbre

Un document HTML peut être vu comme un arbre !

53 / 90

Navigation et propriété des arbres

On manipule ici des arbres **enracinés**, i.e. ayant un nœud particulier appelé **racine** (*root*) (on le dessine en général en haut !)

On appelle profondeur d'un nœud sa distance (en nombre d'arrête à) la racine

Les **fil** (*children*) d'un nœud **n** sont les nœuds relié directement à **n** et de profondeur plus grande.

Le **parent** d'un nœud **n** est le nœud relié directement à **n** et de profondeur inférieure.

Un nœud sans fils est appelé une **feuille**. L'**unique nœud** sans parent est la **racine**.

Les frères (*siblings*) d'un nœud **n** sont tous les nœud qui ont le même parent que **n**.

Les **descendants** d'un nœud sont ses fils et les fils de ses fils et ... (jusqu'aux feuilles).

55 / 90

Pourquoi est-ce utile ?

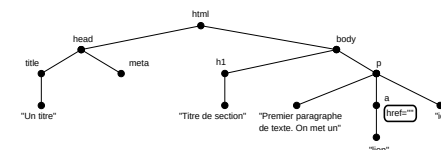
Voir un document HTML comme un arbre est fondamental :

- ◆ Pour écrire des feuilles de style complexes
- ◆ (Pour manipuler le HTML depuis un langage de programmation tel que Java ou Javascript)

54 / 90

Modèle d'arbre pour XHTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Un titre</title>
    <meta charset="utf-8" />
  </head>
  <body>
    <h1>Titre de section</h1>
    <p> premier paragraphe de texte. On met
    un <a href="http://www.lri.fr">lien</a> ici.
  </p>
  </body>
</html>
```



56 / 90

1 HTTP, HTML, CSS

1.1 Rappels ✓

1.2 Internet et ses services ✓

1.3 Fonctionnement du Web ✓

1.4 Adressage des documents Web ✓

1.5 Le protocole HTTP ✓

1.6 UTF-8 ✓

1.7 HTML, le format des documents ✓

1.8 Vue arborescente des documents ✓

1.9 CSS : Boîtes

1.10 Sélecteurs et application en cascade

CSS : Langage permettant de décrire le *style graphique* d'une page HTML

On peut appliquer un style CSS :

- ◆ À un élément en utilisant *l'attribut style*
- ◆ À une page en utilisant l'élément `<style>...</style>` dans l'en-tête du document (dans la balise `<head>...</head>`).
- ◆ À un ensemble de pages en référençant un fichier de style dans chacune des pages

58 / 90

L'attribut style

```
<a href="http://www.u-psud.fr" style="color:red">Un lien</a>
```

Apperçu:

Un lien

Inconvénients :

- ◆ il faut copier l'attribut style pour tous les liens de la page
- ◆ modification de tous les éléments difficiles

59 / 90

L'élément style

```
<html>
  <head>
    <title>...</title>
    <style>
      a { color: red; }
    </style>
  </head>
  <body>
    <a href="...">Lien 1</a> <a href="...">Lien 2</a>
  </body>
</html>
```

Apperçu :

Lien 1 Lien 2

Inconvénient : local à une page

60 / 90

Fichier style.css:

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

Fichier test.html:

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

Modifications & déploiement aisés

61 / 90

Unités de longueur

CSS permet de spécifier des longueurs comme valeurs de certaines propriétés (position et taille des éléments, épaisseur des bordures, ...). Les longueurs *doivent* comporter une unité. Les unités reconnues sont:

px : pixel
in : pouce (2,54cm)
cm : centimètre
mm : millimètre
pt : point (1/72ème de pouce, 0,35mm)
pc : pica (12 points)
em : facteur de la largeur d'un caractère de la police courante
ex : facteur de la hauteur d'un caractère « x » de la police courante
% : pourcentage d'une valeur particulière (définie par propriété)
vh : *viewport height* (% de la hauteur de la partie visible de la page) CSS3
vw : *viewport width* (% de la largeur de la partie visible de la page) CSS3
vmin : plus petite valeur entre vw et vh CSS3
vmax : plus grande valeur entre vw et vh CSS3

63 / 90

Une *propriété* CSS est définie en utilisant la syntaxe:

```
nom_prop : val_prop ;
```

◆ Si on utilise l'attribut style d'un élément:

```
<a href="..." style="color:red;border-style:solid;border:1pt;">Lien 1</a>
```

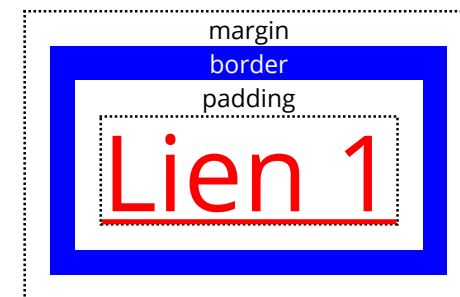
◆ Si on utilise un fichier .css ou une feuille de style:

```
a {
  color : red;
  border-style: solid;
  border: 1pt;
}
h1 { /* Le style des titres de niveau 1 */
  text-decoration: underline;
  color: green;
}
```

62 / 90

Boîte

Chaque élément de la page HTML possède une *boîte rectangulaire* qui délimite le contenu de l'élément:



La zone entre le contenu et la bordure s'appelle le *padding* (« remplissage »). La zone autour de la bordure s'appelle *margin* (« marge »).

64 / 90

Marge, bordure, ajustement

On peut spécifier jusqu'à 4 valeurs:

- ◆ 1 valeur: toutes les dimensions égales à cette valeur
- ◆ 2 valeurs: haut et bas égal à la première valeur, gauche et droite égale à la deuxième
- ◆ 3 valeurs: haut à la première valeur, gauche et droite égale à la deuxième, bas égal à la troisième
- ◆ 4 valeurs: haut, droit, bas, gauche

```
span {
  padding:10pt 20pt 5pt 0pt;
  margin:10pt 5pt;
  border-width:3pt;
  border-color:red blue green;
  border-style:solid dotted;
}
```

Du  dans une boîte

65 / 90

Modes d'affichage

La propriété *display* contrôle le mode d'affichage d'un élément:

- none :** l'élément n'est pas dessiné et n'occupe pas d'espace
- inline :** l'élément est placé sur la ligne courante, dans le flot de texte. La taille du contenu (avec les marges, ajustements et bordures) dicte la taille de la boîte, *height* et *width* sont ignorés (*<i>*, **, **, **, ... sont *inline* par défaut).
- block :** l'élément est placé seul sur sa ligne. La taille est calculée automatiquement mais peut être modifiée par *width* et *height* (*<div>*, *<h1>*, *<p>*, ... sont *block* par défaut)
- inline-block :** positionné comme *inline* mais la taille peut être modifiée comme pour *block*

67 / 90

Calcul de la taille d'une boîte CSS3

L'attribut CSS *box-sizing* permet de spécifier le mode de calcul de la taille d'une boîte. Deux valeurs sont possible :

content-box

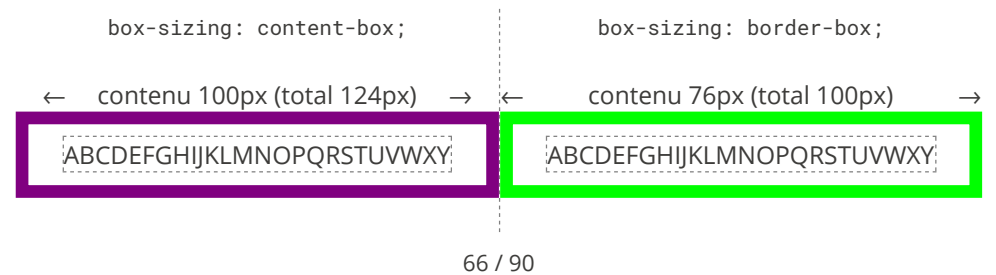
(valeur par défaut) signifie que la taille d'une boîte (telle que donnée par *width* ou *height*) est uniquement celle de son contenu.

border-box

signifie que la taille d'une boîte (telle que donnée par *width* ou *height*) est uniquement celle de son contenu, de l'ajustement et des bordures.

la marge n'est jamais prise en compte dans le calcul de la taille.

Exemple : *border: 2px; padding: 10px; width: 100px;*



66 / 90

Modes d'affichage (exemples)

```
a { display: inline; ... }
```

Le [lien 1](#), le [lien 2](#) et le [lien 3](#).

```
a { display: none; ... }
```

Le , le et le .

```
a { display: block; ... }
```

Le

[lien 1](#)

, le

[lien 2](#)

et le

[lien 3](#)

68 / 90

Positionnement

Le type de positionnement est donné par la propriété `position`

- static :** positionnement « automatique »
- fixed :** positionnement par rapport à la fenêtre du navigateur (la boîte est supprimée du flot)
- relative :** positionnement « relatif » par rapport à la position normale
- absolute :** positionnement « absolu » par rapport à l'ancêtre le plus proche qui n'est pas *static*

Pour `fixed`, `relative` et `absolute`, les propriétés `top`, `bottom`, `left` et `right` dénotent les décalages respectifs.

69 / 90

Gestion du débordement

L'attribut `overflow` permet de gérer le débordement. Il peut prendre les valeurs `visible`, `hidden` et `auto` :

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint

71 / 90

Positionnement (exemple) fixed (right:10pt,top:10pt)

```
a { position: static;
... }
a { position: fixed;
right:10pt;
top: 10pt;
}
a { position: relative;
left: 10pt;
bottom: -5pt;
... }
a { position: absolute;
right:0pt;
bottom: 10pt;
}
<ul style="position:relative;"><li>...</li> ...</ul>
```

- ◆ Positionnement `static`
- ◆ Positionnement
- ◆ Positionnement `relative (left:10pt,bottom:-5pt)`
- ◆ Positionnement `absolute (right:10pt,bottom:10pt)`

70 / 90

Couleurs

Les couleurs peuvent être données:

- ◆ par nom symbolique: `red`, `blue`, `purple`, ...
- ◆ en hexadécimal: `#xyyyzz`, avec $00 \leq xx, yy, zz \leq ff$
- ◆ en décimal: `rgb(x, y, z)`, avec $0 \leq x, y, z \leq 255$
- ◆ en décimal avec transparence: `rgba(x, y, z, a)`, avec $0 \leq x, y, z \leq 255$ et $0 \leq a \leq 1$

[css3](#)

On peut définir la **couleur de fond** d'une boîte avec la propriété `background` et la couleur du texte avec la propriété `color`

72 / 90

Propriétés du texte

Certaines propriétés permettent d'alterer le rendu du texte d'un élément

direction :	ltr ou rtl (orientation du texte)
text-transform :	capitalize, uppercase, lowercase
text-decoration :	underline, overline, line-through
text-align :	left, right, center, justify
text-indent :	longueur du retrait de paragraphe
vertical-align :	alignement vertical par rapport à la ligne de texte baseline, top, bottom, middle ou une longueur.

73 / 90

Plan

1 HTTP, HTML, CSS

1.1 Rappels ✓

1.2 Internet et ses services ✓

1.3 Fonctionnement du Web ✓

1.4 Adressage des documents Web ✓

1.5 Le protocole HTTP ✓

1.6 UTF-8 ✓

1.7 HTML, le format des documents ✓

1.8 Vue arborescente des documents ✓

1.9 CSS : Boîtes ✓

1.10 Sélectionneurs et application en cascade

Propriétés de la police

font-family :	liste de nom de polices séparées par des virgules (Helvetica, sans, "Times New Roman")
font-style :	normal, italic
font-weight :	normal, lighter, bold, bolder
font-size :	soit une longueur soit xx-small, x-small, small, medium, large, x-large, xx-large

On peut aussi spécifier un descripteur de police [css3](#)

```
@font-face {
  font-family: Toto;
  src: url(toto.ttf);
}
a { font-family: Toto; }
```

74 / 90

Sélectionneurs

On peut sélectionner finement les éléments auxquels un style s'applique :

* :	tous les éléments
x :	tous les éléments dont la balise est x
.foo :	tous les éléments dont l'attribut class contient foo
x.foo :	(sans espace) tous les éléments dont l'attribut class contient foo et dont la balise est x
#foo :	l'élément dont l'attribut id vaut foo (les id doivent être uniques)
X Y :	tous les éléments sélectionnés par Y qui sont des descendants des éléments sélectionnés par X
X, Y :	tous les éléments sélectionnés par X ou par Y
X > Y :	tous les éléments dont sélectionné par Y qui sont des fils d'éléments sélectionnés par X
a:visited :	les liens déjà visités
a:link :	les liens non visités
X:hover :	élément sélectionné par X et survolé par la souris

La spécification CSS3 en définit beaucoup d'autres ...

76 / 90

Attributs id et class

On veut souvent pouvoir appliquer un style à un unique élément d'une page, ou à un groupe d'éléments bien défini. On utilise pour cela l'attribut id de l'élément ou l'attribut class commun à plusieurs éléments:

Fichier CSS:

```
-----  
#toto123 { background:red; }  
.bluetext { color: blue; }  
.border { border: 1pt solid green; }  
-----
```

L'élément d'id toto123 a un fond rouge.
Les éléments de classe bluetext sont bleus.
Les éléments de classe border ont une bordure verte.

Fichier HTML

```
-----  
<html>  
  <head> ... </head>  
  <body>  
    <div class="bluetext"> AAA </div>  
    <div id="toto123" class="border"> BBB </div>  
    <div class="border bluetext"> CCC </div>  
  </body>  
</html>
```

AAA
BBB
CCC

77 / 90

Cascade ? (priorités)

Que se passe-t-il quand un élément est sélectionné par plusieurs sélecteurs ?

- ◆ Les propriétés **présentent exclusivement** dans l'un ou l'autre **sont appliquées**
- ◆ Pour les propriétés présentes dans plusieurs sélecteurs (ex: color:blue; et color:red;):

1. Les propriétés des attributs **style** ont une priorité plus forte que celles de l'élément **<style >** de l'en-tête, qui a lui-même une priorité plus forte que les feuilles de styles **.css** référencées.
2. On prend ensuite pour chaque règle le triplet :
(nombre d'id (#foo), nombre de classes (.bar), nombre de balises)
On sélectionne les règles ayant **la plus forte valeur**, comparé composantes par composantes (**ordre lexicographique**). S'il reste plusieurs règles possibles, on prend la dernière déclaration dans l'**ordre du fichier**.

79 / 90

Composition de sélecteurs

Les sélecteurs CSS peuvent être composés afin de créer des **chemins de sélection** :

```
div li a { ... } /* tous les a descendants d'un li descendants d'un div */  
  
#id100 > p .foo { ... } /* tous les éléments de class foo se trouvant  
à n'importe quelle profondeur sous un p  
lui-même étant un sous-élément direct  
de l'élément d'id id100 */  
  
p.bar { ... } /* toutes les balises p qui ont la classe bar */  
  
p .bar { ... } /* toutes la balises ayant la classe bar placées  
sous une balise p (attention à l'espace !) */  
  
a, li * /* tous les a ou tous les éléments placés sous un li */
```

78 / 90

Exemple de sélecteurs ambigus

```
* { color: blue; } /* score (0, 0, 0) */  
#id101 li b { color: red; } /* score (1, 0, 2) */  
.class ol li b { color: pink; } /* score (0, 1, 3) */  
#id101 ol * b { color: green; } /* score (1, 0, 2) */
```

Le score le plus élevé est (1, 0, 2). Il y a deux sélecteurs qui ont ce score, on choisit le dernier dans l'ordre du fichier donc le texte des éléments sélectionnés sera **vert**.

80 / 90

```
<ul class="menu">
  <li>Entrée 1
    <ul class="sous-menu"> <li>Sous-entrée 1.1 </li>
      <li>Sous-entrée 1.2 </li>
      <li>Sous-entrée 1.3 </li>
    </ul>
  </li>
  <li>Entrée 2
    <ul class="sous-menu"> <li>Sous-entrée 2.1 </li>
      <li>Sous-entrée 2.2 </li>
      <li>Sous-entrée 2.3 </li>
    </ul>
  </li>
</ul>
```

81 / 90

exemple : menu dépliant (démonstration)



83 / 90

Pour que le menu soit « dépliant » lors du survol de la souris, on souhaite que :

- ◆ Par défaut, les éléments de sous-menu soient cachés (`display : none`)
- ◆ Les éléments se trouvant sous un élément survolé (`hover`) soient visibles (`display : block`)

82 / 90

Style CSS du menu

```
li { padding: 10pt 0pt 10pt 0pt;
  display:block;
  background:orange;
  color:blue;
}
.sous-menu { display : none; }

ul.menu > li:hover ul.sous-menu { display: block; }
ul.menu > li:hover ul.sous-menu > li {
  background:blue;
  color:orange;
}
```

84 / 90

Valeurs symboliques de propriétés

Pour n'importe quelle propriété (color, height, ...) on peut donner des valeurs symboliques :

- inherit : utilise la même valeur que celle de l'élément parent
- initial : utilise le style par défaut
- unset : annule le style spécifique et utilise inherit

Fichier CSS:

```
div { background: red; }  
span { background: blue; }  
div span { background: unset; }
```

Fichier HTML

```
<html>  
<head> ... </head>  
<body>  
  <div > AAA </div>  
  <span> BBB </span>  
  <div> <span> CCC </span></div>  
</body>  
</html>
```

Tous les divs sont en rouge
Tous les spans sont en bleu
Les spans sous un div n'ont pas le couleur de fond, donc sont en rouge

AAA
BBB
CCC

85 / 90

Animations

On peut demander à CSS d'animer les changements de propriétés au moment de l'application d'un style, grâce à la propriété transition :

transition: prop₁ duree₁ fun₁ del₁, ..., prop_n duree_n fun_n del_n

- prop : un nom de propriété (height, background-color, ...)
- duree : une durée d'animation en secondes
- fun : une fonction d'animation parmi linear, ease, ...
- del : un délais de démarrage en secondes

Attention, on ne peut animer que si les propriétés de départ et d'arriver ont des valeurs numériques (i.e. pas de inherit, unset, auto, ...).

87 / 90

Propriété transform

On peut appliquer une transformation géométrique à un élément avec la propriété transform. On peut utiliser une liste des valeurs suivantes :

- rotate (angle) : applique une rotation. angle peut être donné en degrés (45deg), en radians (3.14rad) ou en tours (0.1turn)
- scale (fx, fy) : passe à l'échelle la largeur d'un facteur fx et la hauteur d'un facteur fy.
- skew (anglex, angley) : déforme la boîte selon un angle horizontal (anglex) et vertical (angley)

Il existe aussi une notion de transformation 3D plus complexe (application d'une matrice de transformation). Attention, les transformations s'appliquent à toute la boîte (bordure comprise).

font-size: 150%

transform: scale(2,2)

86 / 90

Animations (démonstration)

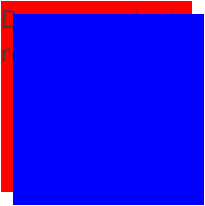
On dispose à chaque fois d'un carré avec une bordure noire et un fond bleu clair, d'une largeur et hauteur de 10vmin. On anime à chaque fois une grandeur dans :hover.

```
.box {  
  width:10vmin;  
  height:10vmin;  
  background-color: #ddf;  
  border: solid 0.1vmin black;  
}  
#b1:hover { width:20vmin; transition: width 1s linear; }  
...
```



88 / 90

Il arrive que certaines boîtes se recouvrent :



On peut utiliser la propriété `z-index` pour définir l'ordre d'empilement (plus elle est élevée, plus la boîte est en avant plan). Le `z-index` par défaut vaut zéro, on a donné la propriété `z-index:2` ; à la boîte contenant du texte :



On peut charger un style CSS de manière conditionnelle grâce à l'attribut `media` de la balise `link`. La valeur de l'attribut est une **formule logique** où l'on peut tester le type de support d'affichage ainsi que ces caractéristiques physiques :

```
<link rel="stylesheet" type="text/css" href="style1.css" media="all" />  
<link rel="stylesheet" type="text/css" href="style2.css" media="print" />  
<link rel="stylesheet" type="text/css" href="style3.css" media="screen and landscape" />  
<link rel="stylesheet" type="text/css" href="style4.css" media="screen and min-width:480px" />  
<link rel="stylesheet" type="text/css" href="style5.css" media="screen and max-width:479px" />
```

Cela permet d'appliquer des styles spécifiques lors de l'impression d'une page ou pour des terminaux mobiles (ayant une petite taille d'écran) ou de changer de style si l'orientation de l'écran est modifiée.