

Technologies et langages pour le Web

Coté client
Z225DM05

Yannick Loiseau

Université Clermont-Auvergne

Licence Informatique 2^e année
Version du 16 janvier 2023

Yannick Loiseau

- ▶ `yannick.loiseau@uca.fr`
- ▶ bureau D008 (bâtiment ISIMA)

Technologies et langages pour le Web Client

- ▶ CM : 15h
- ▶ TP : 15h

Vidéos...

Qu'est-ce que le Web ?

Web \neq Net

~~site internet~~

Internet

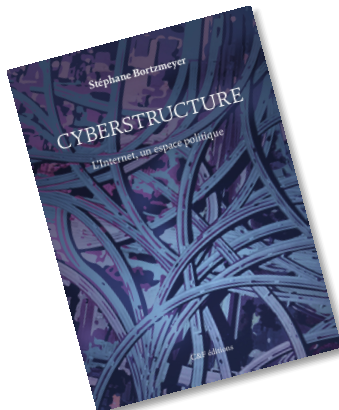
Réseau de réseaux \Rightarrow interconnexion de machines

Buts

- ▶ tolérance aux pannes
- ▶ abstraction de l'infrastructure physique
- ▶ « intelligence » à la périphérie (neutralité)

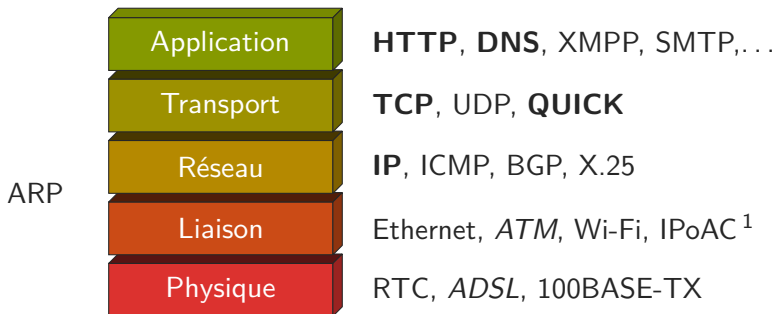
Cyberstructure : L'Internet, un espace politique (2018)

BORTZMEYER



⇒ protocoles en couches

- ▶ isolation
- ▶ indépendance des niveaux → évolution
- ▶ ⇒ connaissance limitée du système global



1. RFC 1149 ; 2001 : 9 paquets 5km, 55% perte, latence 1h

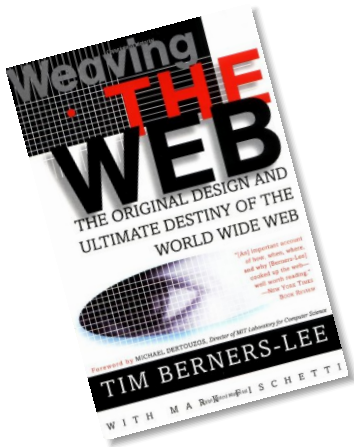
- ▶ ARPA (dpt. de la défense US)
- ▶ fin 60 début 70 :
premiers nœuds ARPANET : 1969 (RFC 1)
- ▶ RFC 675 – Internet transmission control program :
1974
- ▶ IP (Internet Protocol) et TCP (Transmission Control Protocol) (ou UDP – User Datagram Protocol) : RFC 791–RFC 793 (1981)

Suppose all information stored on computers everywhere were linked. Suppose I could program my computer to create a space in which anything could be linked to anything. There would be a single, global information space.

— Tim Berner-Lee (1980)

Weaving the Web (2000)

BERNERS-LEE





Le Web

Interconnexion de ressources \Rightarrow hypermédia

\Rightarrow couche applicative

Buts

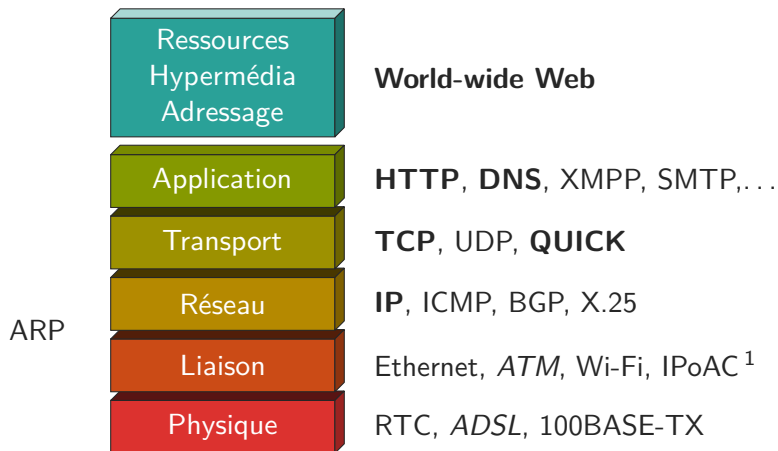
Système d'informations

- ▶ non/semi structurées
- ▶ global
- ▶ à grande échelle
- ▶ décentralisé
- ▶ non supervisé
- ▶ tolérant aux pannes
- ▶ extensible (évolutif)

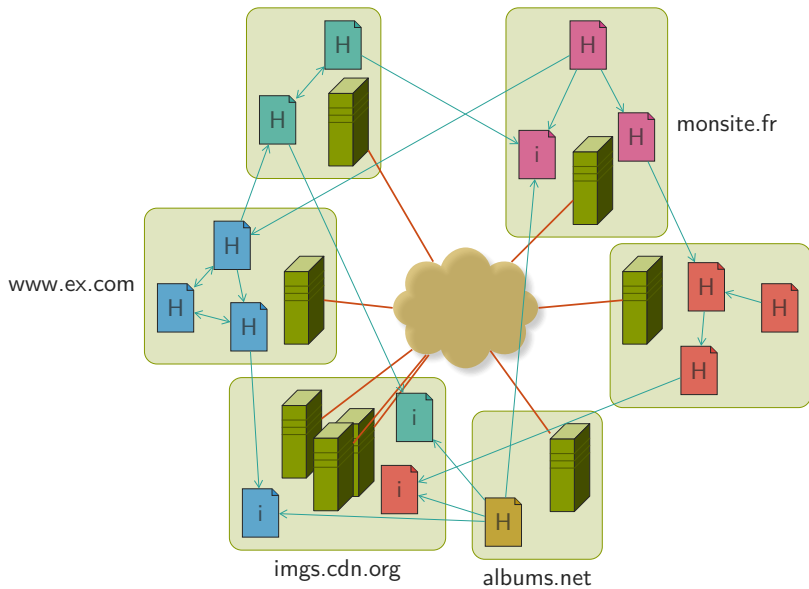
⇒ contraintes architecturales

FIELDING 2000 : « Architectural Styles and the Design of Network-based Software Architectures. »

- ▶ client/serveur
- ▶ faible couplage : client générique, hypermédia
- ▶ ressources
- ▶ représentation homogènes (HTML)
- ▶ protocole : interface uniforme (HTTP)
- ▶ adressage uniforme : indépendance (URL)



1. RFC 1149 ; 2001 : 9 paquets 5km, 55% perte, latence 1h

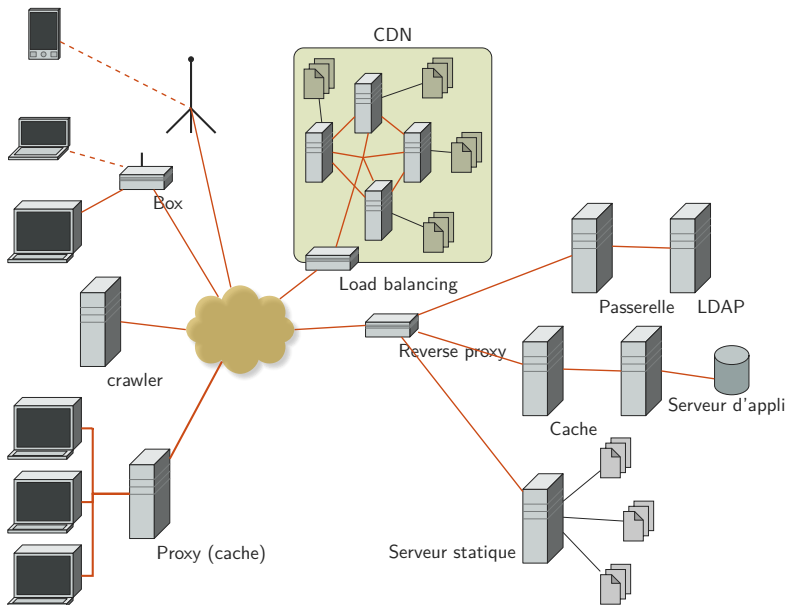


Site web ?

- ▶ ≠ machine
- ▶ ≠ domaine
- ▶ apparence ?
- ▶ ⇒ ensemble de ressources « cohérentes »

⇒ ressources / liens

- ▶ Hypermédia : 1962-65 (D. Engelbart, T. Nelson)
- ▶ T. Berner-Lee (CERN) : ENQUIRE (1980) ; proposition (1989)
- ▶ HTTP/0.9, HTML, httpd, navigateur (CERN 1990-91)
- ▶ Le CERN « libère » le Web en 1993 ; W3C (1994)
- ▶ HTTP (1996-99), URI (1998), Apache httpd (1995), libwww(-perl)
- ▶ REST (PhD Th. 2000)



Définition (Agent)

- ▶ client
- ▶ effectue les requêtes
- ▶ manipule les ressources (traitement, affichage)
- ▶ maintient l'état

Exemple

- ▶ navigateur (firefox, chrome, opera, IE)
- ▶ agrégateur (RSS, Podcasts)
- ▶ robots (indexation, extraction)
- ▶ téléchargement (wget, miroirs, ...)
- ▶ applications spécifiques (clients Twitter, Facebook, etc.)

Définition (Serveur)

- ▶ répond aux requêtes du client
- ▶ gère les ressources
 - ▶ état, création, destruction
 - ▶ représentations (génération)
 - ▶ adressage
- ▶ serveur statique / serveur d'application

Exemple

- ▶ Apache, Nginx, IIS, lighttpd, ...
- ▶ mod_php, node.js, tomcat, glassfish, gunicorn, ...

Définition (Passerelle)

- conversion de protocoles/formats
- à la fois serveur et client
- générique

≠ serveur d'appli. → logique métier *ad hoc*

Exemple

`mod_proxy_ftp`, `mod_proxy_ajp`

Définition (Cache)

- ▶ mémorisation des réponses
- ▶ performances
- ▶ tolérances pannes
- ▶ local \Rightarrow privé (navigateur)
- ▶ intermédiaire \Rightarrow partagé (proxy)
- ▶ \Rightarrow mutualisation

Exemple

Varnish, mod_cache

Définition (Proxy)

- ▶ intermédiaire
- ▶ explicite / transparent
- ▶ partage
- ▶ filtrage et contrôle d'accès (parental, pubs, ...)
- ▶ conversion (optim. mobile)
- ▶ cache
- ▶ masquage de source

Exemple

Squid, polipo, mod_proxy_http,

Définition (Reverse Proxy)

- ▶ intermédiaire coté serveur
- ▶ répartition de charge
- ▶ portail / intégration
- ▶ relais

Exemple

HAProxy, Apache, nginx, Pound, mod_proxy, mod_proxy_html, mod_proxy_balancer

Définition (CDN)

- ▶ *content delivey network*
- ▶ ensemble de serveurs
- ▶ miroirs
- ▶ répartis géographiquement
- ▶ public / privé
- ▶ \Rightarrow échelle

Exemple

Amazon, Google, Azure, OVH, ...

Définition (Hypermédia)

- ▶ information unitaire : ressource
- ▶ information structurée : liens (typés)
- ▶ non linéaire (graphe)
- ▶ « navigation » non séquentielle (*découverte dynamique*)
- ▶ inclusion (*transclusion*)
- ▶ \Rightarrow référence

Définition (Ressource)

- ▶ élément d'information
- ▶ abstrait $\Rightarrow \neq$ représentations
- ▶ adressable \Rightarrow URI

Exemple

- ▶ profil sur un réseau social
- ▶ article de blog ou d'actu
- ▶ « les conditions météo à Clermont-Ferrand aujourd'hui »

Les URI : identificateur de ressource

URI

Identifiant global unique pour une ressource

- ▶ URN : Uniform Resource Name
- ▶ URL : Uniform Resource Locator

URL

(RFC 3986, RFC 1738)

Définition (URL)

Identifiant de ressource

- ▶ unique : espace de nom par le DNS
- ▶ dérérérençable : suffisant pour retrouver le document
- ▶ définie par le serveur : sans gestion centrale
- ▶ opaque : compréhension (sémantique) uniquement pas le serveur

URL

Exemple

- ▶ `http://www.example.com/foo/bar`
- ▶ `ftp://file.server.org/some/path/readme.txt`
- ▶ `mailto:user@server.com`
- ▶ `file:///home/zaphod/Documents/myfile.pdf`
- ▶ `data:image/png;base64,iVBORw0KGgoAAAAN...`
- ▶ ...

comment (protocole) où (serveur) quoi (ressource locale au serveur)

http ://www.example.com :80 /foo/bar ?a=1&b=1 #ici

► scheme ●

► hostname ●

► port ●

► path ●

► query string ●

► fragment ●

- ▶ uri opaque : identifiant
- ▶ sémantique serveur :
 - ▶ ⇒ navigation / découverte
 - ▶ ⇒ pas de construction (ou le serveur donne les instructions)
 - ▶ ⇒ pas de compréhension par le client (autre que la structure)
 - ▶ ⇒ découplage client/serveur
 - ▶ ⇒ contrôle total du server

Cool URIs don't change

URIs don't change : people change them

`http://www.w3.org/Provider/Style/URI`

`http://www.example.net/index.php?type=article&id=1234`

- ▶ `http://www.example.net/articles/1234`
- ▶ `http://www.example.net/article-1234`
- ▶ `http://www.example.net/a531ae42-84c1-455e-86a3`

`http://www.example.net/monimage.jpg`

`http://www.example.net/monimage`

Principe : Design d'URL

- ▶ réflexion a priori
- ▶ espace d'adressage abstrait \Leftrightarrow représentation physique
- ▶ indépendante des informations variables

chemin, nom, domaine

- ▶ propriétaire (~john/article)
- ▶ techno (.php, /cgi-bin/, ...)
- ▶ format (.html, .jpg, ...)
- ▶ accès (public, private)
- ▶ titre, catégorie, structure de l'organisation
⇒ redirection URL canonique
- ▶ status (draft, etc.)
 - ✓ latest, today → uri canonique

Exemple

En favoris, scripts, etc. :

<http://weather.net/fr/clermont-ferrand/today>

redirige vers

<http://weather.net/fr/clermont-ferrand/2020-02-02>

si l'on est le 2 février 2020

redirige vers

<http://weather.net/fr/clermont-ferrand/2020-02-03>

le lendemain. . .

⇒ partage

Exemple

<http://doc.app.com/latest>

redirige vers

<http://doc.app.com/v1.2>

(si la version courante est la 1.2)

Propriétés intéressantes

URI \equiv API

- ▶ courte
- ▶ facile à retenir
- ▶ facile à écrire, dicter, lire
- ▶ « bidouillable »

« bidouillable » ⇒ changement « à la main »

Exemple

- ▶ <http://weather.net/fr/clermont-ferrant/2020-02-02>
 - ▶ <http://weather.net/fr/clermont-ferrant/2020-02-01>
 - ▶ <http://weather.net/fr/clermont-ferrant/2020-02-03>
 - ▶ <http://weather.net/fr/paris/2020-02-02>
- ▶ <http://journal.net/articles/1234>
 - ▶ <http://journal.net/articles/1235>
 - ▶ <http://journal.net/articles/>

Pas requis (opaque)

<http://weather.net/5211c671-a93d-4964-ab54-75ef18babf31>,
<http://weather.net/c4d1a05e-f2f4-47b7-a3fe-2cf1be6ea798>

client : découvre et suit les liens



⇒ liens typé ([next](#), [previous](#), [up](#), ...)



Niveau réseau : identification des machines par l'adresse IP

Exemple

93.184.216.34

- ▶  difficile à retenir
- ▶  dépendante de l'architecture réseau

DNS

- ▶ **niveau d'indirection**
- ▶ résolution nom de machine \Leftrightarrow adresse IP
- ▶ \approx plus facile à retenir
- ▶ \checkmark indépendance du réseau
- ▶ système hiérarchique
 - ▶ serveurs
 - ▶ structure

Exemple

`www.example.net` \Leftrightarrow `93.184.216.119`

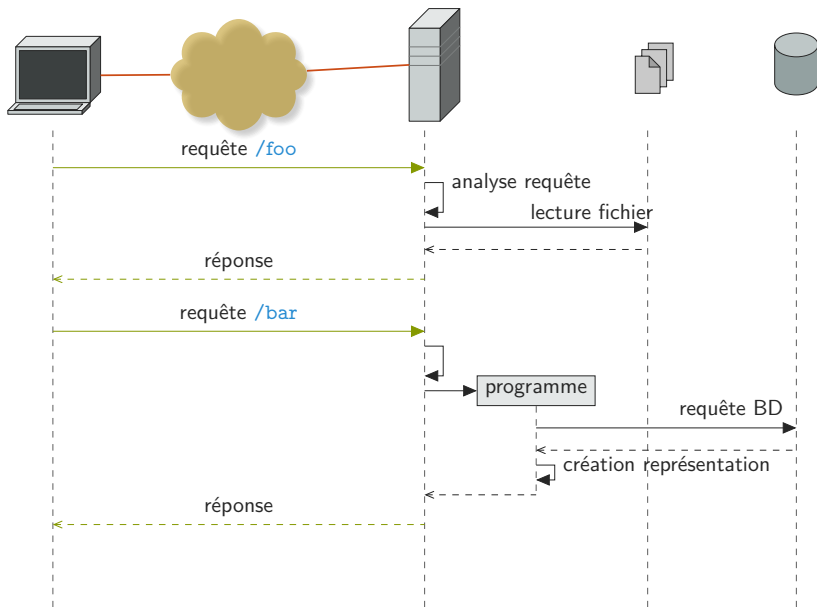
plus facile ?

`8.8.8.8` vs. `gtf0ks.f1x.sj`

Hypertext Transfert Protocol : (RFC 2616)

RFC 7230, RFC 7231, RFC 7232, RFC 7233, RFC 7234, RFC 7235

Manipulation des ressources transfert des représentations de l'état



start line

en-têtes

ligne vide
corps

version, type de message,...
Nom : valeur Nom : valeur ...
...

En-têtes généraux

- ▶ Cache-Control
- ▶ Connection : close, keep-alive
- ▶ Date
- ▶ Pragma
- ▶ Via : intermédiaires
- ▶ ...

Requête

start line : verbe identifiant version

Exemple

```
GET /foo HTTP/1.1
```

Manipulation de ressources

RFC 7231

- ▶ création : **POST**
- ▶ lecture : **GET** (**HEAD**)
- ▶ mise à jour : **PUT** – **PATCH** (RFC 5789)
- ▶ suppression : **DELETE**

Propriétés des méthodes

- ▶ idempotence
- ▶ sans effet de bord

Définition (Idempotence)

n requêtes \equiv 1 requête

Définition (sans effet de bord)

pas de modification de l'état de la ressource

Propriétés des méthodes

- ▶ indépendance des intermédiaires
- ▶ « cachabilité »
- ▶ reprise sur erreur
- ▶ automatisation / responsabilité

GET /articles/1234?action=delete

Propriétés des méthodes

Méthode	Idempotente	Sans effet de bord
GET	✓	✓
HEAD	✓	✓
PUT	✓	✗
DELETE	✓	✗
POST	✗	✗
PATCH	✗	✗

Propriétés des méthodes

À propos du POST

Pas de sémantique bien définie

- ▶ POSTa : append (commentaire, message de forum)
- ▶ POSTp : process (traitement quelconque)

RFC 2310 : en-tête **Safe** : (yes|no)

Méta informations, diagnostic et connexion

- ▶ **OPTIONS** : méta-informations
- ▶ **CONNECT** : tunnel
- ▶ **TRACE** : echo

Réponse

start line : version status message

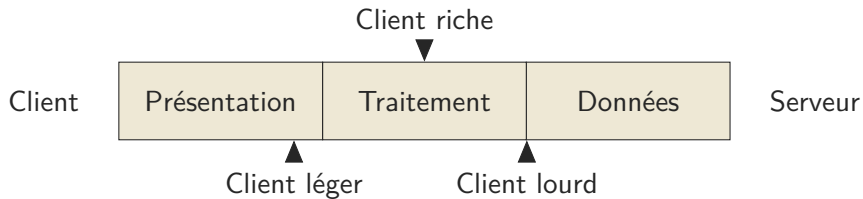
Exemple

HTTP/1.1 200 OK

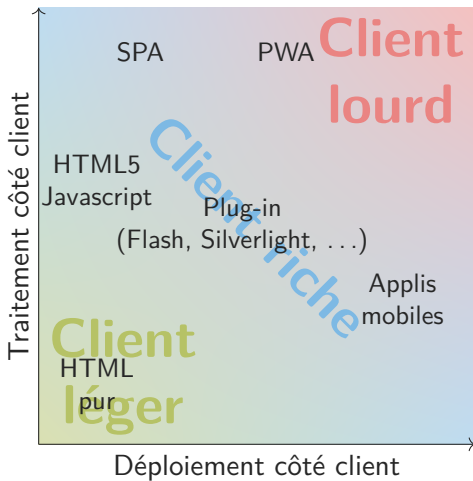
5 catégories :

- ▶ 1xx : Informations (100 [Continue](#), 101 [Switching Protocols](#))
- ▶ 2xx : Succès (200 [OK](#), 201 [Created](#), 204 [No Content](#))
- ▶ 3xx : Redirection ou pas de contenu (301 [Moved Permanently](#), 304 [Not Modified](#))
- ▶ 4xx : Erreur du client (404 [Not Found](#), 401 [Unauthorized](#), 418 [I'm a teapot](#)²)
- ▶ 5xx : Erreur du serveur (500 [Internal Server Error](#), 504 [Gateway Timeout](#))

<http://www.iana.org/assignments/http-status-codes/>



RIA



Migration client dédié (lourd) → navigateur générique + interface Web (légère)

Pourquoi ?

HTTP :

- ▶ tolérant aux pannes
- ▶ performant
- ▶ sécurisé

Exemple

connexion non persistante, cache et compression, reprise, proxy, SSL et authentification

Pourquoi ?

Navigateur comme plateforme :

- ▶ CoD (code on demand)
- ▶ pas de déploiement
- ▶ évolutivité
- ▶ indépendant du terminal

Pourquoi ?

Techno libres, documentées :

- facilité de mise en place
- composition, SOA

coté client (*frontend*) :

- ▶ données : HTML
- ▶ présentation : CSS
- ▶ traitement (comportement) : Javascript (ECMAScript)

Données : Html

HTML *HyperText Markup Language*

Langage à balise pour l'hypertexte

<http://www.w3.org/TR/html/>

- ▶ 1999 : html 4.01 <http://www.w3.org/TR/html401/>
- ▶ 2002 : xhtml 1.0
- ▶ 2010 : xhtml 1.1 <http://www.w3.org/TR/xhtml-basic/>
- ▶ 2014 : html 5 <https://www.w3.org/TR/html52/>
- ▶ WhatWG <https://html.spec.whatwg.org/>

Langage

- ▶ langage : grammaire / syntaxe
- ▶ API de manipulation (DOM, . . .)

Définition (Grammaire)

- éléments du langage (vocabulaire)
- structure (construction)

⇒ syntaxe abstraite

syntaxe concrète → représentation « sérialisation »

- ▶ SGML (HTML \leq 4, HTML5)
- ▶ XML (xHTML, HTML5)

SGML *Standard Generalized Markup Language*

- ▶ Charles Goldfarb
- ▶ 1986
- ▶ ISO 8879 :1986

méta-langage

- syntaxe concrète
- grammaire quelconque

DTD *Document Type Definition* \Rightarrow grammaire

```
<!ENTITY % heading "H1|H2|H3|H4|H5|H6">
<!ENTITY % coreattrs
"id          ID          #IMPLIED
class       CDATA       #IMPLIED
style       %StyleSheet; #IMPLIED
title       %Text;      #IMPLIED"
>
<!ELEMENT LINK - O EMPTY>
<!ATTLIST LINK
%attrs;
charset     %Charset;    #IMPLIED
href        %URI;       #IMPLIED
hreflang    %LanguageCode; #IMPLIED
type        %ContentType; #IMPLIED
rel         %LinkTypes;  #IMPLIED
rev         %LinkTypes;  #IMPLIED
media       %MediaDesc;  #IMPLIED
>
<!ELEMENT P - O (%inline;)* >
<!ATTLIST P %attrs; >
<!ELEMENT UL - - (LI)+ >
<!ATTLIST UL %attrs; >
```

```
<a class="lien-a lien-b" href="bar.html" id=foo hidden>lien</a>
```

- ▶ élément : fermant optionnel, vide
- ▶ contenu
- ▶ attribut
 - ▶ booléen
 - ▶ valeur
 - ▶ multiple

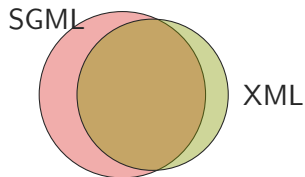
entités

- ▶ caractères « spéciaux » (cf. encodage)
- ▶ < : `<` ; & : `&` ;
- ▶ → : `&z#8594;` ;

analyse complexe : grammaire nécessaire

⇒ XML

XML



- ▶ dérivé de SGML
- ▶ *eXtensible Markup Language*
- ▶ 1996 → 2008
- ▶ W3C

XML

- ▶ extensions
 - ▶ espaces de nom (fusion de langages)
 - ▶ Processing Instructions
- ▶ encodage unicode par défaut
- ▶ syntaxe plus stricte

⇒ grammaire non nécessaire à l'analyse

bien formé vs. valide

Définition (Document bien formé)

respecte la syntaxe concrète

```
<a>foo <b>bar</a> baz</b>
```

```
<a>foo <b>bar</b> baz</a>
```

Définition (Document valide)

respecte la grammaire

p. ex. pas de `<p>` dans un ``

`foo <p>bar</p> baz`

⇒ bien formé, non valide

valide ?

- ▶ compatibilité
- ▶ \neq clients

syntaxe

- ▶ `<?xml version="1.0" ?>`
- ▶ valeurs d'attributs (obligatoire, quotes)
- ▶ éléments fermés : ``
- ▶ 1 élément racine

document → arbre DOM (Document Object Model)

séparation font / forme
donnée / présentation

- ▶ balises sémantiques
- ▶ vers le « responsive »

Structure générale

```
<?xml version="1.0" encoding="utf-8" ?>
<!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"
    lang="fr-FR" xml:lang="fr-FR">
<head>
...
</head>
<body>
...
</body>
</html>
```

En-têtes

méta-données

- ▶ `<meta name="" content="" />`
 - ▶ techniques : en-têtes HTTP, encodage, espace d'affichage...
 - ▶ documentaire : auteur, description, mots-clés...
- ▶ `<title></title>`
- ▶ `<link rel="" href="" />`

```
<!doctype html>
<html lang="fr">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta http-equiv="content-language" content="fr" />
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<meta name="author" content="Yannick Loiseau" />
<meta name="keywords" content="test,html" />
<meta name="description" content="page de test pour la structure html"/>

<title>Test structure</title>
</head>
```

Contenu

Structure

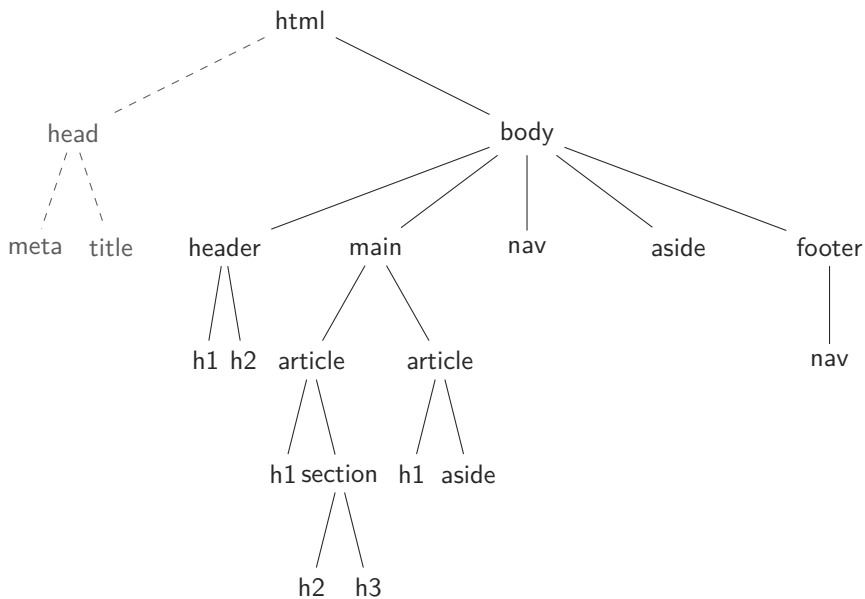
- ▶ `<header>`
- ▶ `<main>`
- ▶ `<footer>`
- ▶ `<article>`
- ▶ `<section>`
- ▶ `<nav>`
- ▶ `<aside>`
- ▶ `<h1>`, `<h2>`, ..., `<h6>`
- ▶ ...

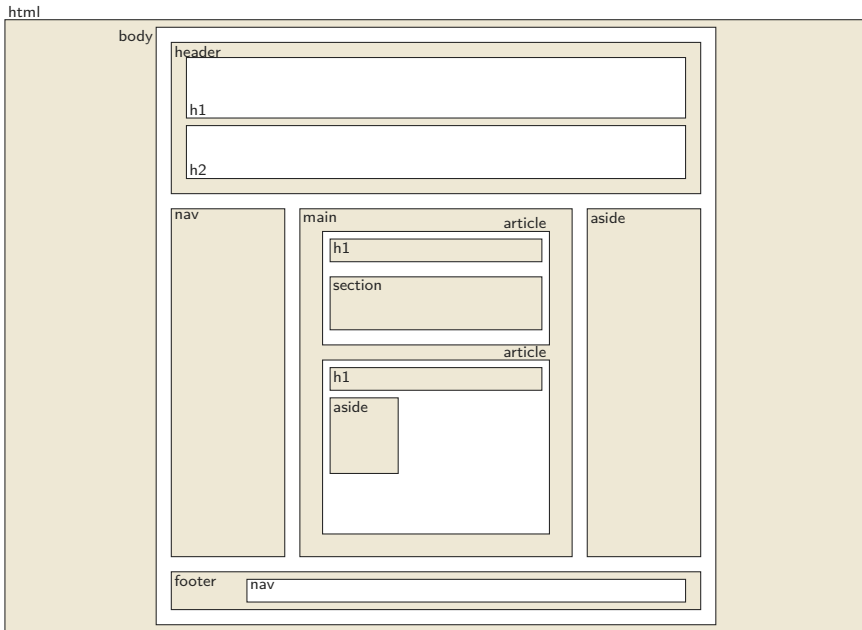

```
<body>
<header>
  <h1>Titre principal du site</h1>
  <h2>sous titre du site</h2>
</header>
<main>
<article><h1>Titre de l'article</h1>

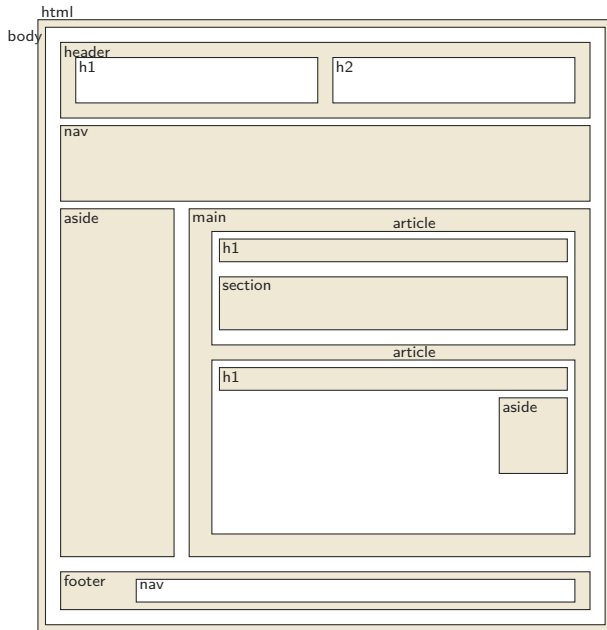
  <section><h2>Titre de section</h2>
    ...

    <h3>Titre de sous-section</h3>
    ...
  </section>
</article>
<article><h1>Un autre article</h1>
  <aside>Un encart dans l'article</aside>
</article>
</main>
```

```
<nav>
  divers liens de navigation sur le site
</nav>
<aside>
  informations annexes
</aside>
<footer>
  pied de page
  <nav>liens du pied de page</nav>
</footer>
</body>
</html>
```







par défaut → ordre d'apparition

⇒ + important d'abord

Contenu

Semantique

- ▶ `<p>`
- ▶ `important`
- ▶ `plus important`
- ▶ `<abbr title="Hypertext Markup Language">HTML</abbr>`
- ▶ `<blockquote>`
- ▶ `<time datetime="2001-01-01">millenium</time>`
- ▶ ...

<http://www.w3.org/TR/html/semantics>

Plus de structure ?

éléments génériques

- ▶ `<div>`
- ▶ ``

⇒ attributs `<div id="" class="" role="">`

Images

- ▶ ``
- ▶ `<svg>`, `<canvas>`
- ▶ `<figure>` et `<figcaption>`

```
<figure>  
    
  <figcaption>Description de l'image</figcaption>  
</figure>
```

Audio / Vidéo

- ▶ `<audio>` et `<video>`
- ▶ attributs : `src`, `preload`, `autoplay`, `mediagroup`, `loop`, `muted`, `controls`
- ▶ `<track src="uri" kind="type" />`
 - sous-titre : `subtitles`
 - sous-titre description : `captions`
 - audio description : `descriptions`
 - chapitres : `chapters`
- ▶ `<source src="uri" type="mime" media="query" />`

```
<video poster="myvid.jpg" controls="controls">
  <source src="video.ogv" type="video/ogg; codecs='theora,vorbis'"/>
  <source src="video.3gp" type="video/3gpp; codecs='mp4v.20.8, samr'"/>
  <source src="video.mp4" type="video/mp4; codecs='avc1.42E01E,
    mp4a.40.2'"/>
  <a href="myvid.webm">Download the video</a>
</video>
```

[https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_
and_embedding](https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding)

→ tableaux

✓ données

✗ mise en forme

```
<table><caption>Titre du  
  tableau</caption>  
<thead>  
  <tr><th>A</th><th>B</th></tr>  
</thead>  
<tbody>  
  <tr><td>1</td><td>5</td></tr>  
  <tr><td>2</td><td>6</td></tr>  
  <tr><td>3</td><td>7</td></tr>  
</tbody>  
<tfoot>  
  <tr><th>A</th><th>B</th></tr>  
</tfoot>  
</table>
```

Tableau – Titre du tableau

A	B
1	5
2	6
3	7
A	B

- ▶ `<td headers="a b"></td>`
- ▶ `<td colspan="2" rowspan="2"></td>`
- ▶ `<th id="a" scope="col" abbr="titre"></th>`
 - ▶ row
 - ▶ col
 - ▶ rowgroup
 - ▶ colgroup
- ▶ `<colgroup id="col1" span="3"/>`
- ▶ `<colgroup><col id="col1-2" span="2"/><col id="col3"/></colgroup>`

- ▶ <http://www.w3.org/TR/html/tabular-data>
- ▶ <https://developer.mozilla.org/en-US/docs/Learn/HTML/Tables>

Liens

► navigation :

- `<link rel="" title="" href="" type="" media="" hreflang="" />`
- ``
`_blank`, `_self`, `_parent`, `_top`

► transclusion :

- `<iframe src="uri" name="nom" seamless="seamless" />`
- ``
- `<audio src="" />`
- `<video src="" />`

Relations

- ▶ [alternate](#)
- ▶ [author](#)
- ▶ [bookmark](#)
- ▶ [help](#)
- ▶ [icon](#)
- ▶ [license](#)
- ▶ [nofollow](#)
- ▶ [noreferrer](#)
- ▶ [prefetch](#)
- ▶ [search](#)
- ▶ [tag](#)
- ▶ [next](#)
- ▶ [prev](#)

<http://www.iana.org/assignments/link-relations/link-relations.xhtml>

<http://microformats.org/wiki/existing-rel-values>

<https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/rel>

https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types

- ▶ `<form action="uri" method="get|post">`
- ▶ `<input name="" type="" value="" />`
- ▶ `<label for="">`
- ▶ `<fieldset> <legend>`
- ▶ `<textarea>`
- ▶ `<select> <optgroup> <option>`
- ▶ `<input list=""> <datalist id=""><option>`
- ▶ `<button>`
- ▶ `<output>`
- ▶ `<meter title="" min="" max="" value="" low="" high="" optimum="" />`
- ▶ `<progress max="100" value="50">50%</progress>`
- ▶ `<keygen>`

Fonctionnement

GET

```
<form action="/articles" method="GET">
  <label>Search :
    <input type="search" placeholder="Search..." name="q"/>
  </label>
  <input type="hidden" name="client" value="html" />
  <input type="submit" value="Search"/>
</form>
```

```
GET /articles?q=les+mots+de+ma+recherche&client=html HTTP/1.1
Content-Length : 0
[...]
```

Fonctionnement

POST

```
<form action="/articles/" method="POST">
  <fieldset><legend>Information</legend>
    <label for="name">Nom :</label>
    <input name="name" id="name" type="text"/>
    <label for="title">Titre :</label>
    <input name="title" id="title" type="text"/>
    <label><input name="date" type="date"/>Date</label>
  </fieldset>
  <input type="submit" value="Ok"/>
</form>
```

POST /articles/ HTTP/1.1

Content-Type : application/x-www-form-urlencoded

Content-Length : 41

[...]

name=moi&title=mon+titre&date=2001-01-01

Attributs

- ▶ `<input autocomplete="on|off"/>`
- ▶ `<input disabled/>`
- ▶ `<input multiple/>`
- ▶ `<input placeholder="texte"/>`
- ▶ `<input required/>`

Types de champs

- ▶ `text` : `minlength`, `maxlength`
- ▶ `hidden`
- ▶ `search`
- ▶ `tel`
- ▶ `url`
- ▶ `email`
- ▶ `password`
- ▶ `datetime`
- ▶ `date`
- ▶ `month`
- ▶ `week`
- ▶ `time`
- ▶ `datetime-local`
- ▶ `number`
- ▶ `range`
- ▶ `color`
- ▶ `checkbox`
- ▶ `radio`
- ▶ `file` : `accept`
- ▶ `submit`
- ▶ `image`
- ▶ `reset`
- ▶ `button`

Présentation : CSS

Cascading Style Sheet

Feuille de style en cascade

⇒ présentation

Syntaxe générale

```
selecteurs {  
  propriete : valeur ;  
  ...  
}
```

Sélecteurs

élément mis en forme

Exemple

- ▶ balise HTML
- ▶ attribut HTML
- ▶ structure (arbre)
- ▶ état de l'élément
- ▶ ...

Propriétés

caractéristiques de l'élément

Exemple

- ▶ couleur
- ▶ police
- ▶ position
- ▶ taille
- ▶ animation
- ▶ ...

Application au HTML

- ✗ intégration directe
- ≈ feuille embarquée
- ✓ feuille externe

Application au HTML

Intégration directe

```
<p style="color : red">Texte rouge</p>
```

- ▶ ✓ facile
- ▶ ✓ précis (locale)
- ▶ ≈ mélange fond / forme
- ▶ ✗ duplication entre les balises

⚠ exceptionnel

Application au HTML

Feuilles embarquées

```
<html><head>
<style>
  p {
    color : red;
  }
</style>
</head><body>
<p>Texte rouge</p>
</body></html>
```

- ▶ ✓ une seule définition (par document)
- ▶ ≈ duplication entre les pages
- ▶ ✓ / ✗ une seule ressource

⇒ styles locaux

Application au HTML

Feuilles externes

HTML : `<link rel="stylesheet" href="feuille.css"/>`

CSS : `@import url(feuille.css);`

- ▶ ✓ une seule définition globale
- ▶ ✓ séparation fond / forme
- ▶ ✓ ressource externe (performances)
- ▶ ≈ dépendance (résilience)
- ▶ ✗ lourd si aussi locaux

⇒ styles globaux

Application au HTML

Équilibre

- ▶ découpage local / global / média
- ▶ inclusion si nécessaire
- ▶ coût application des règles
- ▶ coût téléchargement
- ▶ déploiement
- ▶ cache (performance, résilience)
- ▶ redéfinition des règles (cascade)

Sélecteurs

- ▶ → élément mis en forme
- ▶ combinables \Rightarrow cascade

- ▶ valeurs par défaut (navigateur)
- ▶ ordre de définition
- ▶ + général → + spécifique
- ▶ héritage : **inherited** → élément parent
- ▶ **!important** → pas de surcharge

- ▶ tout : * {color : red;}
- ▶ nom : em {color : red;}
- ▶ classe : .maclasse {color : red;}
- ▶ id : #monid {color : red;}

em.maclasse p#monid

```
<!DOCTYPE html>
<html>
<head>
  <style type="text/css">
    .foo {background-color : blue; color :
    green}
    .bar {background-color : green}
    .foo .bar {background-color : yellow}
    .foo.bar {background-color : red}
  </style>
</head>
<body>

  <div class="foo">foo</div>
  <div class="bar">bar</div>
  <div class="foo">
    <div class="bar">foo/bar</div>
  </div>
  <div class="foo bar">foo bar</div>

</body>
</html>
```



Cascade

1. importance :
navigateur < utilisateur < auteur < a !important < u !important
2. spécificité : nom < structure < classe (attributs) < id
3. ordre

Héritage

```
<h1>Hello <em>World</em></h1>
```

```
h1 {color : red ;}
```

```
em {font-style : italic ;}
```

Hello *World*

attributs

- ▶ `a[href]`
- ▶ `a[type="application/pdf"]`
- ▶ `a[class~="bar"]` \equiv `a.bar`
- ▶ `a[href^="https://"] e[foo$="bar"] e[foo*="bar"]`
- ▶ `p :lang(fr)`

structure

- ▶ `article em` → `` dans `<article>`
- ▶ `article > h1` → `<h1>` fils direct de `<article>`
- ▶ `h1 + p` → `<p>` immédiatement précédé de `<h1>`
- ▶ `p ~ img` → `` suivant un `<p>`
- ▶ `:first-line`, `:first-letter`
- ▶ `:nth-child(n)`, `:nth-last-child(n)`, `:nth-of-type(n)`,
`:nth-last-of-type(n)`, `:first-child`, `:last-child`, `:first-of-type`,
`:last-of-type`, `:only-child`, `:only-of-type`, `:empty`

```
section :nth-child(4n+1) {background-color : red;}  
section :nth-child(4n+2) {background-color : green;}  
section :nth-child(4n+3) {background-color : blue;}  
section :nth-child(4n+4) {background-color : yellow;}
```

```
section > p :first-child :first-letter {font-size : 200%;}
```

Imbrication



imbrication / sélecteurs complexes

- ▶ dépendant de la structure → couplage
- ▶ plus complexe → plus lent

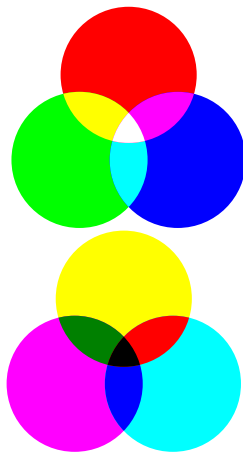
interaction

- ▶ `:link`
- ▶ `:visited`
- ▶ `:active`
- ▶ `:hover`
- ▶ `:focus`
- ▶ `:target`
- ▶ `:enabled`
- ▶ `:disabled`
- ▶ `:checked`

:not(S)









Codage

- ▶ additif : rouge, vert, bleu (RGB)
- ▶ soustractif : cyan, magenta, jaune, noir (CMYN)
- ▶ teinte, saturation, luminance (HSL)



Nommées

16 valeurs : 0, 128, 255

- ▶ transparent
- ▶  black  silver  gray white
- ▶  red  maroon
- ▶  fuchsia  purple
- ▶  blue  navy
- ▶  aqua  teal
- ▶  lime  green
- ▶  yellow  olive

RGB

`rgb(r,g,b)` `#RRGGBB`

`rgb(255,0,0)` = `rgb(100%,0%,0%)` = `#ff0000` = red 

`rgb(0,128,128)` = `rgb(0%,50%,50%)` = `#008080` = teal 

HSL



`hsl(h,s,l)`

`red = hsl(0,100%,50%)`

`teal = hsl(180,100%,25%)`

α

transparence

- ▶ `rgba(r,g,b,a)`
- ▶ `hsla(h,s,l,a)`

Propriété

- ▶ `color`
- ▶ `background-color`
- ▶ `border-color`
- ▶ `outline-color`

dégradés

`linear-gradient(direction, couleur position, ...)`

`linear-gradient(left, red 0%, blue 50%, green 100%)`



`linear-gradient(135deg, red 0%, green 100%)`



`radial-gradient(circle, yellow, green)`



Relatives

générale : %

```
h1 {font-size : 120% ;}
```

```
p {width : 90% ;}
```

Relatives

police

em : taille de la police

ex : *x-height*

rem : taille de la police de l'élément racine

```
h1 {font-size : 1.2em ;}
```

```
p {width : 80em ;}
```



MablpXo

Relatives

zone d'affichage

`vw` : 1% de la largeur

`vh` : 1% de la hauteur

`vmin` : $\min(vw, vh)$

`vmax` : $\max(vw, vh)$

Absolues

cm : centimètre

mm : millimètre

in : pouce (*inch*) $1\text{in} = 2.54\text{cm}$

px : pixel $1\text{px} = \frac{1}{96}\text{in}$

pt : point typographique $1\text{pt} = \frac{1}{72}\text{in}$

pc : pica $1\text{pc} = 12\text{pt}$

URL

```
<link rel="stylesheet"  
href="http://www.example.org/style/base.css"/>
```

```
body {background : url("fond.png")}
```

⇒ <http://www.example.org/style/fond.png>

indep. du document html

Unités

angles : `deg`, `grad`, `rad`, `turn`

temps : `s`, `ms`

fréquences : `Hz`, `kHz`

résolutions : `dpi`, `dpcm`, `dppx`

Fonctions

calcul : `width : calc(50% - 2em)`

cycles : `em {font-style : toggle(italic, normal);}`

attribut : `a : :after {content : "[" attr(href) "];}`

Famille

font-family : Helvetica, sans-serif;

- serif
- sans-serif
- cursive
- fantasy
- monospace

Calpsb Calpsb

Calpsb Calpsb

Calpsb

Famille

✗ limité polices locales

⇒ Web fonts

```
@font-face {  
  font-family : principale;  
  src : local(Foobar),  
        url(Foobar.woff) format("woff"),  
        url(Foobar.otf) format("opentype"),  
        url(Foobar.ttf) format("truetype");  
}  
  
p { font-family : principale, serif; }
```

font-weight, font-style

Épaisseur

`font-weight : bold ;`

- ▶ 100 → 900
- ▶ `normal` (400)
- ▶ `bold` (700)
- ▶ `bolder` (relatif)
- ▶ `lighter` (relatif)

Calpsb

Calpsb

Calpsb

Calpsb

Forme

`font-style : italic ;`

- `normal`
- `italic`
- `oblique`

Calpsb

Calpsb

Calpsb

Taille

```
font-size : 12pt;
```

- ▶ `xx-small x-small small medium large x-large xx-large`
- ▶ `larger smaller`
- ▶ taille absolue (cm, pt, ...)
- ▶ taille relative (cascade) (em, ex, %)

Variante

`font-variant : small-caps ;`

- `normal`
- `small-caps`

Calpsb

CALPSB

Alignement

`text-align : justify ;`

- ▶ `center`
- ▶ `left`
- ▶ `right`
- ▶ `justify`

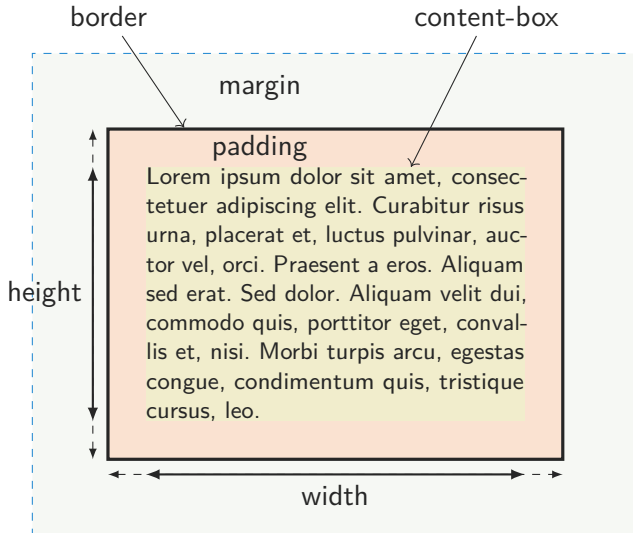
Texte de test pour
illustrer
l'alignement au
centre

Texte de test pour
illustrer l'alignement
justifié

Texte de test pour
illustrer
l'alignement à
gauche

Texte de test pour
illustrer
l'alignement à
droite

- ▶ `text-indent`
- ▶ `text-decoration` : `none` | `underline` | `overline` | `line-through` | `blink`
- ▶ `letter-spacing` `word-spacing`
- ▶ `text-transform` : `capitalize` | `uppercase` | `lowercase` | `none`



display

- ▶ none
- ▶ inline
- ▶ block
- ▶ inline-block

position

- ▶ `static`
- ▶ `relative`
- ▶ `absolute`
- ▶ `fixed`
- ▶ `sticky`

offsets

- ▶ `top`
- ▶ `right`
- ▶ `bottom`
- ▶ `left`

z-index

Float

float

- ▶ left
- ▶ right

clear

- ▶ left
- ▶ right
- ▶ both

Taille

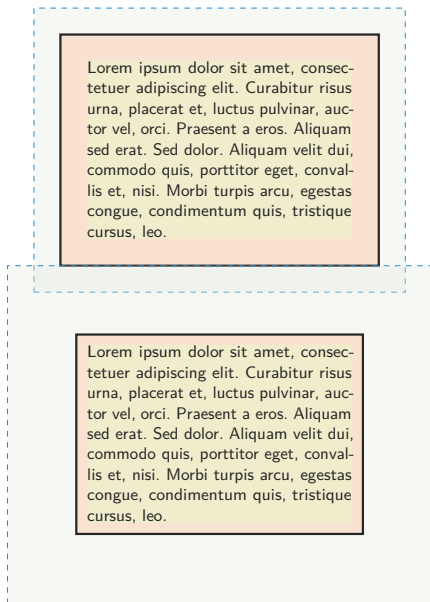
- ▶ `height`, `width`
- ▶ `max-height`, `max-width`
- ▶ `min-height`, `min-width`

`box-sizing`

- ▶ `border-box`
- ▶ `content-box`

Overflow

- ▶ `visible`
- ▶ `hidden`
- ▶ `scroll`
- ▶ `auto`



Marges

- ▶ `margin`
- ▶ `padding`

`-top`, `-right`, `-bottom`, `-left`

`auto`

$\text{offset} + \text{marges} + \text{taille} + \text{bordure} = \text{largeur du bloc englobant}$

Fond

background

- ▶ -color
- ▶ -image
 - ▶ -repeat : repeat-x, repeat-y, repeat, space, round
 - ▶ -attachment : scroll, fixed, local
 - ▶ -position
 - ▶ -clip : border-box, padding-box, content-box
 - ▶ -origin : border-box, padding-box, content-box
 - ▶ -size

Bordure

border

- ▶ `-color`
- ▶ `-style` `none`, `hidden`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`
- ▶ `-width`
- ▶ `-top`
- ▶ `-right`
- ▶ `-bottom`
- ▶ `-left`

border-top-color

Bordure

`border-radius` : `top-left`, `top-right`, `bottom-right`, `bottom-left`

rayon horizontal, rayon vertical

`box-shadow` : h offset, v offset, rayon de flou, distance (taille)

changement de propriété ([hover](#), [focus](#)) → progressif

Propriétés

transition-property : **prop1**, **prop2**, ...

boîte, fond, bordure

- ▶ couleur, opacité
- ▶ taille
- ▶ position
- ▶ police (taille et épaisseur)
- ▶ marges et padding
- ▶ visibilité
- ▶ z-index
- ▶ **all**

<http://www.w3.org/TR/css3-transitions/#animatable-css>

Durée

`transition-duration` : 11, 12, ...

Aller-retour

`transition` → vers la cible

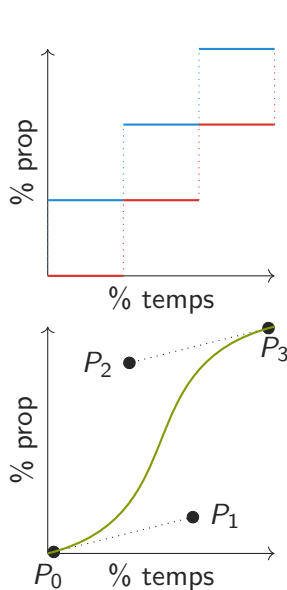
Délai

`transition-delay` : `d1`, `d2`, ...

Fonction de transition

transition-timing-function \Rightarrow interpolation

- ▶ `steps(n, start|end)`
- ▶ `cubic-bezier(p1x, p1y, p2x, p2y)`
- ▶ `ease`
- ▶ `linear`
- ▶ `ease-in`
- ▶ `ease-out`
- ▶ `ease-in-out`
- ▶ `step-start`
- ▶ `step-end`



Raccourcis

`transition : prop1 l1 f1 d1, prop2 l2 f2 d2, ...`

Exemple

`transition : color 2s, width 4s;`

- ▶ définition de transitions
- ▶ étapes (*keyframes*)
- ▶ déclenchement automatique

```
@keyframes test {  
  0% {  
    color : red;  
    width : 2em;  
  }  
  
  50% {  
    color : green;  
    width : 6em;  
  }  
  
  100% {  
    color : blue;  
    width : 4em;  
  }  
}  
  
#elt {  
  animation : test 5s linear 0s infinite;  
}
```


- ▶ utilisation conditionnelle de règles
- ▶ affichage \neq média

Inclusion

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

⚠ ordre → cascade

```
@import url(print.css) print ;
```

⚠ URL relative ≠

Déclaration

```
@media print { /* ... */ }
```

Média

- ▶ `print`
- ▶ `screen`
- ▶ `speech`
- ▶ `all`

Fonctionnalités

propriété : valeur + min-, max-

- ▶ width
- ▶ height
- ▶ orientation : portrait | landscape
- ▶ aspect-ratio
- ▶ color
- ▶ color-index
- ▶ monochrome
- ▶ resolution
- ▶ scan : progressive | interlace
- ▶ grid
- ▶ hover
- ▶ pointer
- ▶ prefers-color-scheme : dark | light
- ▶ ...

Mots-clés

- ▶ only
- ▶ not
- ▶ and

```
@media (min-width : 50em) and (orientation : landscape){  
  /* mode tablette */  
}
```

- ▶ Stylus : <http://learnboost.github.com/stylus/>
- ▶ Sass : <http://sass-lang.com/>
- ▶ Less : <http://lesscss.org/>

Ajout de fonctionnalités

- ▶ variables
- ▶ calculs et fonctions
- ▶ héritage et mixins
- ▶ imbrication des définitions

Surchouches et pré-processeurs pour CSS

Sass

lib.scss

```
$colors-base : hsl(210deg,40%,10%);  
$colors-background : lighten($colors-base, 20%);  
$colors-text : darken(desaturate($colors-base, 10%), 40%)  
  
%menu {  
    display : block;  
    vertical-align : middle;  
    background-color : $colors-background;  
    color : $colors-text;  
}
```

Surchouches et pré-processeurs pour CSS

Sass

```
@import "lib.scss";

#nav {
  @extend %menu;

  a {
    display : block;
    text-decoration : none;
    font-size : 150%;

    & :active, & :hover, & :focus {
      background-color : $colors-base;
    }
  }
}
```

Scalable and Modular Architecture for CSS (2012)

SNOOK



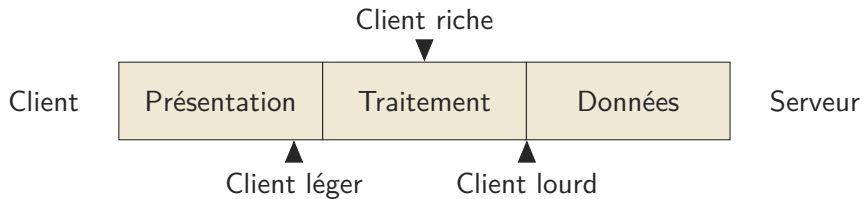
code à la demande (CoD)

⇒ ajout de fonctionnalités au client (générique)

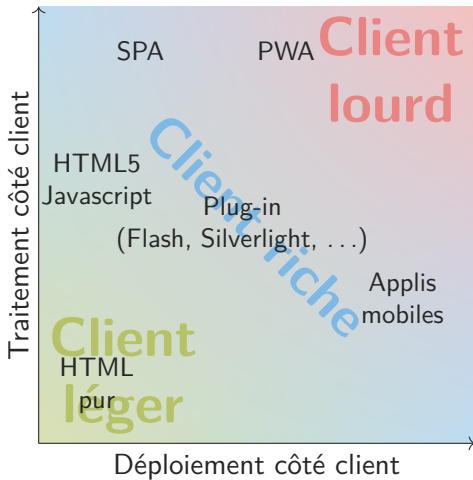
applet, plug-in, script embarqué

fonction spécifique : → client riche

⇒ navigateur ≡ plateforme



RIA



Plus précisément

- ▶ interactivité (IHM)
- ▶ moins d'échange C/S → réactivité
 - ▶ trie de tableau
 - ▶ vérif. formulaire
 - ▶ requêtes HTTP (rafraichissement partiel)
- ▶ *mash-up*

Historique



Brendan Eich



1995 : Netscape

LiveScript → Javascript

1996 : Microsoft → JScript

1997 : ECMA-262 → EcmaScript

1999 : XMLHttpRequest (IE5)

2001 : JSON

2005 : Ajax

2011 : EcmaScript 5.1

2015 : EcmaScript 6



- ▶ <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>
- ▶ <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/>
- ▶ <http://exploringjs.com/es6/index.html>

Environnement

- ▶ langage : EcmaScript
 - ▶ syntaxe
 - ▶ types
 - ▶ sous-typage (prototypes, héritage)
 - ▶ bibliothèques standards
- ▶ VM
- ▶ intégration navigateur
 - ▶ DOM
 - ▶ événements
 - ▶ API (historique, fenêtres, requêtes HTTP, géoloc., ...)

langage interprété \Rightarrow bytecode \rightarrow JIT

- ▶ SpiderMonkey : Mozilla (Firefox, CouchDB, Adobe)
- ▶ V8 : Google (Chrome, Node.js)
- ▶ Chakra, SquirrelFish, ...

- ▶ au chargement
- ▶ *inline* → URL
- ▶ événementiel

Chargement

exécution dans l'ordre



ordre



performances \Rightarrow bloquant

inclusion en fin de document

Inclusion

`<script>`

- ▶ *inline* :

```
<script type="text/javascript">alert("hello");</script>
```

- ▶ *externe* :

```
<script src="filename.js"></script>
```

- ▶ `defer`, `async`

`<noscript>`

URI

javascript :alert("coucou");

✗ lien :

✓ *bookmarklet*

Événementiel

- ▶ approche IHM
- ▶ fonction → événement
- ▶ `onclick`, `onmouseover`, ...

✗ ``

≈ `link.onclick = function(evt) {alert("hello")};`

✓ `link.addEventListener("click", (evt) => alert("hello"));`

Paradigmes du langage

- ▶ impératif
- ▶ fonctionnel
- ▶ objet (prototype)

Paradigmes du langage

- ▶ « interprété » (déploiement des sources)
- ▶ évaluation dynamique
- ▶ typage dynamique (structurel, duck typing)
- ▶ typage faible (transtypage automatique)

- ▶ commentaires : C-like : *// commentaire* ou */* commentaire */*
- ▶ fin d'instruction : `;`



insertion automatique

```
return {  
  "answer" : 42  
};
```

```
return  
{  
  "answer" : 42  
};
```

Variables

- ▶ non typées
- ▶ mots-clés : `var` portée fonction (hoisting)
- ▶ `let` et `const` ES6 portée block
- ▶ pas d'initialisation → `undefined`
- ▶ valeur nulle : `null`

```
> var a;  
> a === undefined;  
true  
> typeof a;  
"undefined"  
> a === null;  
false  
> a = 1;  
1  
> typeof a;  
"number"  
> a = "Toto";  
"Toto"  
> typeof a;  
"string"
```

Variables

Portées

```
> if (true) { var x = 42; } ; console.log(x);  
42  
> if (true) { let y = 42; } ; console.log(y);  
ReferenceError : y is not defined
```



pas de préfixe \Rightarrow global

\Rightarrow contextes et *hoisting*

Définition (hoisting)

« déplace » la déclaration au début du scope

Variables

Hoisting

```
> function say(to) { console.log(hello + " " + to); }  
> say("world");  
ReferenceError : hello is not defined  
> var hello = "Hello";  
> say("world");  
Hello world
```

Variables

Hoisting

```
> var f = function() {console.log("Hello " + t);}
```

```
> f()
```

```
ReferenceError : t is not defined
```

```
> var f = function() {console.log("Hello " + t); var t = "World"}
```

```
> f()
```

```
Hello undefined
```


Variables

Déclaration

raccourci :

```
var a = 1; var b = 2;
```

≡

```
var a = 1, b = 2;
```

Variables

Identifiants

identifiants valides :

- ▶ 1^{er} : lettre unicode, \$, __
- ▶ autres : idem + chiffres unicode

Exemple




```
var foo;
```

```
var $foo123;
```

```
var $;
```

```
var __;
```

Opérateurs

- ▶ affectation : `=`
- ▶ arithmétique : `+`, `-`, `*`, `/`, `%`, ...
 transtypage sur `+`
- ▶ binaires : `&`, `|`, `^`, `~`, `>>`, `<<`
- ▶ affectation combinée : `+=`, `|=`, ...
- ▶ booléens : `&&`, `||`, `!`
 transtypage
- ▶ comparaison : `===`, `!==`, `<`, `>`, `>=`, `<=`
 `==`, `!=`

Transtypage

Addition

```
> 2 + 3
```

```
5
```

```
> "a" + "b"
```

```
"ab"
```

```
> "2" + 2
```

```
"22"
```

```
> 2 + "2"
```

```
"22"
```

```
> +"2" + 2
```

```
4
```

```
> "1" + 2 + 3
```

```
"123"
```

```
> 1 + 2 + "3"
```

```
"33"
```

Transtypage

Égalité

```
> "2" == "2"  
true  
> 2 == 2  
true  
> "2" == 2  
true  
> "2" === 2  
false
```

```
> false == "false"  
false  
> false == "0"  
true  
> "" == 0  
true  
> 0 == "0"  
true  
> "" == "0"  
false  
> null == false  
false  
> false == undefined  
false  
> null == undefined  
true
```

Transtypage

Utilisation de la méthode `valueOf`

Transtypage

```
> let a = new Object();  
> a.valueOf = function() { return 42; }  
> a.toString = function() { return "Answer"; }  
> a + 0  
42  
> 'a' + a  
"a42"  
> 'answer is ${a}'  
"answer is Answer"  
> a == 42  
true  
> a == 'Answer'  
false  
> a === 42  
false
```

Instructions

► Conditionnelles :

- Si : `if (expr) {instr} else {instr}`
- Cas : `switch(variable) {case val : instr break ;}`

► Itératives :

- `while (expr) {instr}`
- `do instr while (expr) ;`
- `for (init ; cond ; inc) {instr}`
- `for (var attr in expr) {instr}` ⚠
- `for (var elt of iter) {instr}` ES6

Instructions

Boucles

⚠ `for (var i in expr)`

```
a = ["a", "b", "c"];  
for (var i in a) { console.log(a[i]); }
```

a
b
c

```
a.foo = "ahahah";  
Array.prototype.foo = "pwned";  
for (var i in a) { console.log(a[i]); }
```

a
b
c
ahahah
pwned

```
for (var e of a) { console.log(e); }
```

a
b
c

Primitifs littéraux

- ▶ nombres : `var taille = 1.75;`
`var age = 35;`
- ▶ `Infinity`, `NaN`
- ▶ booléens : `true`, `false`
- ▶ Chaînes de caractères : `'foo'` ou `"foo"`,
`'hello ${name}'`

```
> typeof 1.75;  
"number"  
> typeof 35;  
"number"  
> typeof NaN;  
"number"  
> typeof true;  
"boolean"  
> typeof "foo";  
"string"  
> typeof undefined;  
"undefined"  
> typeof null;  
"object"
```

Remarques

 nombre : IEEE 754 (64 bits)

```
> 0.1 + 0.2 == 0.3 ;
```

```
false
```

```
> 0.1 + 0.2;
```

```
0.30000000000000004
```

Primitif objet

- ▶ `var age = new Number(35);`
- ▶ `var cond = new Boolean(false);`
- ▶ `var nom = new String("Zaphod");`

```
> typeof age;  
'object'  
> age instanceof Number;  
true  
> age == 35  
true  
> typeof cond  
'object'  
> cond == false  
true  
> typeof nom  
'object'  
> nom == "Zaphod"  
true
```

⚠ pas de type \Rightarrow pas de autoboxing

```
> "foo" === new String("foo");
```

```
false
```

```
> typeof "foo";
```

```
"string"
```

```
> typeof (new String("foo"));
```

```
"object"
```

```
> 2 === new Number(2);
```

```
false
```

```
> typeof 2;
```

```
"number"
```

```
> typeof (new Number(2));
```

```
"object"
```

Autres objets

Date

```
var maDate = new Date (annee, mois, jour, h, min, s);
```

✓ `getHours(), getMinutes(), getSeconds()`

⚠ `getDate() → 1–31`

⚠ `getDay() → 0–6 (0 = dimanche)`

⚠ `getMonth() → 0–11`

⚠ Y2K

```
> (new Date()).getFullYear();
```

122

```
> (new Date()).getFullYear();
```

2022

```
> typeof (new Date());
```

"object"

```
> typeof Date();
```

"string"

Autres objets

Expressions régulières

`var re = /exp/` ou `var re = new RegExp("exp")`

- ▶ `RegExp.exec`, `RegExp.test`,
- ▶ `String.match`,
- ▶ `String.search`,
- ▶ `String.replace`,
- ▶ `String.split`

- ▶ `^`, `$`
- ▶ `[abc]`, `[^abc]`, `[a-z]`
- ▶ `\d`, `\w`, `\s`, ...
- ▶ `*`, `+`, `?`, `{n}`, `{n,m}`
- ▶ `foo|bar`
- ▶ `(\w+)\1`


```
> /[abc]{3}/.test("abc")
true
> /[abc]{3}/.test("aaa")
true
> /[abc]{3}/.test("abd")
false
> /[abc]{3}/.test("aa")
false
> "Foo Bar".replace(/(\w+)\s(\w+)/, "$2 et $1")
"Bar et Foo"
```

```
> tel = /\d{2}([-\/. ]?)\d{2}\1\d{2}\1\d{2}\1\d{2}/  
{}  
> tel.test("0123456789")  
true  
> tel.test("01-23-45-67-89")  
true  
> tel.test("01/23/45/67/89")  
true  
> tel.test("012345678")  
false  
> tel.test("01/23-45/67-89")  
false
```

Tableaux

Déclaration

littéral ✓ `var tab = [1, 2, 'a', ['answer', 42]];`

objet ≈

- ▶ `var tab = new Array(1, 2, 'a');`
- ▶ `var tab = new Array();` → tableau vide
- ▶ `var tab = new Array(5);` → tableau de 5 éléments

factory ✓

- ▶ `Array.from(iterable)` (copie)
- ▶ `Array.of(v1, v2, v3,...)`


Tableaux

Accès

```
> let t = [1, 2, 3]
> t.length
3
> t[0]
1
> t[5] = 42
42
> t.length
6
> t[5]
42
> t[4] === undefined
true
```

Tableaux

Méthodes

- ▶ `join(sep)`
- ▶ `reverse()`, `sort()`
 transtypage
> `[1, 2, 10].sort()`
`[1, 10, 2]`
- ▶ `concat(...)`, `slice(debut, fin)`
- ▶ `push(val)`, `pop()`, `unshift(val)`, `shift()`
- ▶ ...

Tableaux associatifs

- mapping clé \mapsto valeur
- clé : chaîne (ou `o.toString()`)
- valeur : primitif ou objet

```
> t = {} ;  
{}  
> t["coucou"] = "salut" ;  
"salut"  
> t[5] = [1, 2, 3] ;  
[1,2,3]  
> t ;  
{"5" : [1,2,3], "coucou" : "salut"}  
> t["coucou"] ;  
"salut"  
> t["foobar"] === undefined ;  
true
```

Tableaux associatifs

Notation littérale

```
var t = {  
  "a" : "coucou",  
  "b" : "salut",  
  "c" : [1, 2, 3],  
  "sous" : {  
    "foo" : "bar",  
    "spam" : [  
      1,  
      2,  
      {"egg" : "42"}  
    ]  
  }  
};
```

```
console.log(t["sous"]["spam"][2]["egg"]);
```

```
42
```

```
for ( k in t ) {  
  console.log(k + ' : ' + t[k]);  
}
```

```
a : coucou  
b : salut  
c : 1,2,3  
sous : [object Object]
```

Aussi `Map` and `Set`

- ▶ `function nom(param1, param2){...}`
- ▶ `function nom(param1=default1, param2=default2){...}`
- ▶ `return`

Appel variadique

```
> function foo(a, b) { console.log('a = ${a}; b = ${b}'); }  
> foo()  
a = undefined; b = undefined  
> foo(1)  
a = 1; b = undefined  
> foo(1, 2)  
a = 1; b = 2  
> foo(1, 2, 3)  
a = 1; b = 2
```

Vararg

- ▶ `arguments`
- ▶ `arguments[1]`
- ▶ `var args = Array.prototype.slice.call(arguments);` ES5
- ▶ `const args = Array.from(arguments);` ES6

ES6

```
function foo(a, b, ...rest)
```

1^{re} classe

fonction \equiv valeur

- affectée
- passée en paramètre
- retournée

```
function foo(a, b) {  
  return a + b;  
}
```

```
var toto = foo;
```

```
toto(1, 2);
```

Anonymes

λ

- à la demande
- locales
- GC

```
var hello = function(name){ return "Hello " + name; };  
  
hello("World");
```

ES6 :

```
const hello = (name) => "Hello " + name;  
  
hello("World");
```

Closure

variables libres \Rightarrow capture dans la définition

```
function count() {  
  var c = 0;  
  return function() {  
    c += 1;  
    return c;  
  };  
}
```

```
> c1 = count();  
> c1()  
1  
> c1()  
2  
> c2 = count();  
> c2()  
1  
> c1()  
3  
> c2()  
2
```

Closure

```
function say(pref, suff) {  
    return (name) => `${pref} ${name} ${suff}`;  
}
```

```
var hello = say("Hello", "!");  
var bye = say("Goodbye", "...");
```

```
hello("Arthur");  
bye("Trillian");
```

Paramètre

```
function say(display, pref, suff) {  
  return function(name) {  
    display(`${pref} ${name} ${suff}`);  
  }  
}
```

```
let hello = say(console.log, "Hello", "!");  
hello("Arthur");
```

```
let bye = say(function (t) {  
  document.querySelector("body").appendChild(  
    document.createTextNode(t)  
  )  
}, "Goodbye", "...");  
bye("Trillian");
```

```
say(window.alert, "Yo", "\uD83D\uDC4B")("Zaphod")
```


Hoisting

lever, hisser

```
var a = 1;  
var f = function(b) { return a + b; }  
f(2); // -> 3
```

```
var f = function(b) { return a + b; }  
f(2); // -> erreur  
var a = 1;  
f(2); // -> 3
```

Sucre

mapping → notation pointée, pas de quote

`var o = {"a" : 1} ⇔ var o = {a : 1};`

`o["a"] ⇔ o.a`

Construction littérale

fonction comme valeur

```
var obj0 = {  
  myA : 1,  
  myB : 2,  
  hello : function () { return "Hello !"; }  
};
```

```
> obj0.myA ;  
1  
> obj0.hello();  
"Hello !"
```

Fonction de construction

```
var MyObject1 = function (a, b) {  
  return {  
    myA : a,  
    myB : b,  
    hello : function () { return "Hello !"; }  
  };  
};
```

```
> obj1 = MyObject1(1, 2);  
{ "myA" : 1, "myB" : 2 }  
> obj1.myA;  
1  
> obj1.hello();  
"Hello !"
```

Closure

```
var MyObject2 = function (a, b) {  
  return {  
    myA : a, myB : b,  
    foo : function () {return a + b; },  
    bar : function (c) {return a + c; }  
  };  
};
```

```
> obj2 = MyObject2(1, 2);  
{"myA" :1,"myB" :2}  
> obj2.foo();  
3  
> obj2.bar(3);  
4  
> obj2.myA = 4; obj2;  
{"myA" :4,"myB" :2}  
> obj2.foo();  
3
```

Closure en mieux

```
var MyObject3 = function (a, b) {  
  var obj = { myA : a, myB : b };  
  obj.foo = function () { return obj.myA + obj.myB; };  
  obj.bar = function (c) { return obj.myA + c; };  
  return obj;  
};
```

```
> obj3 = MyObject3(1, 2);  
{ "myA" : 1, "myB" : 2 }  
> obj3.myA;  
1  
> obj3.foo();  
3  
> obj3.bar(3);  
4  
> obj3.myA = 4; obj3;  
{ "myA" : 4, "myB" : 2 }  
> obj3.foo();  
6
```

Privé ?

```
var MyObjectPriv = function (a) {  
  var priv = a;  
  return {  
    getPriv : function() { return priv; },  
    setPriv : function(val) { priv = val; }  
  };  
};
```

```
> objpriv = MyObjectPriv(4);  
{}  
> objpriv.getPriv();  
4  
> objpriv.setPriv(5); objpriv.getPriv();  
5  
> objpriv.priv === undefined;  
true
```

```
> obj3b = MyObject3(3, 4);  
{ "myA" : 3, "myB" : 4 }  
> obj3.foo === obj3b.foo;  
false
```

⇒ fonction extérieures

Fonctions extérieures

```
var MyObject4Meth = {  
  hello : function(s) { return "Hello " + s; },  
  bye : function() { return "Goodbye..."; }  
};  
  
var MyObject4 = function (a, b) {  
  return { myA : a, myB : b,  
    hello : MyObject4Meth.hello,  
    bye : MyObject4Meth.bye };  
};
```

```
> obj4 = MyObject4(1, 2);  
{ "myA" : 1, "myB" : 2 }  
> obj4.hello("World");  
"Hello World"  
> obj4.bye();  
"Goodbye..."  
> obj4b = MyObject4(2, 3);  
{ "myA" : 2, "myB" : 3 }  
> obj4.hello === obj4b.hello;  
true
```

Et l'accès ? **X**

```
var MyObject3 = function (a, b) {  
  var obj = { myA : a, myB : b };  
  obj.foo = function () { return obj.myA + obj.myB; };  
  obj.bar = function (c) { return obj.myA + c; };  
  return obj;  
};
```

```
var MyObject4Meth = {  
  hello : function(s) { return "Hello " + s; },  
  bye : function() { return "Goodbye..."; }  
};  
var MyObject4 = function (a, b) {  
  return { myA : a, myB : b,  
    hello : MyObject4Meth.hello,  
    bye : MyObject4Meth.bye };  
};
```

⇒ *hoisting* + contexte

Contexte

- ▶ **this** → contexte d'exécution
- ▶ objet (dictionnaire)
- ▶ défaut : objet global (selon plateforme)
- ▶ `call(contexte, args...)` redéfinition du contexte
- ▶ `apply(contexte, [args])` idem avec un tableau

Contexte

```
> test = function (a, b) {return this.foo + " " + a + b;};  
> test("aarrgg", " :");  
"undefined aarrgg :)"  
> foo = "bar";  
"bar"  
> test("baz", "zzz");  
"bar bazzzz"  
> test.call({foo : "spam"}, "egg", "ggg");  
"spam eggggg"  
> test.apply({foo : "hello"}, ["world", "!!!"]);  
"hello world!!!"  
> ({ f : test, foo : 42 }).f("is the ", "answer");  
"42 is the answer"
```

⇒ **this** est lié au contexte

new ?

```
var __new = function(f, args) {  
  var obj = {};  
  f.apply(obj, args);  
  return obj;  
};
```

```
var MyObject5Meth = {  
  hello : function(s) { return "Hello " + s; },  
  foo : function () { return this.myA + this.myB; }  
};  
var MyObject5 = function (a, b) {  
  this.myA = a;  
  this.myB = b;  
  this.hello = MyObject5Meth.hello;  
  this.foo = MyObject5Meth.foo;  
};
```

```
> obj5 = __new(MyObject5, [1, 2]);  
{ "myA" :1, "myB" :2}  
> obj5.hello("World");  
"Hello World"  
> obj5.foo();  
3  
> obj5b = __new(MyObject5, [2, 3]);  
{ "myA" :2, "myB" :3}  
> obj5b.foo();  
5  
> obj5.foo === obj5b.foo;  
true
```

Initialisation

- ▶ \approx copies des méthodes
- ▶ `__new` initialise avec une copie
- ▶ `Object.create(obj)` → « copie » de `obj`

```
__new2 = function(f, meths, args) {  
  var obj = Object.create(meths);  
  f.apply(obj, args);  
  return obj;  
};
```



```
var MyObject6 = function (a, b) {  
  this.myA = a;  
  this.myB = b;  
};
```

```
> obj6 = __new2(MyObject6, MyObject5Meth, [1, 2]);  
{ "myA" : 1, "myB" : 2 }  
> obj6.hello("World");  
"Hello World"  
> obj6.foo();  
3  
> obj6b = __new2(MyObject6, MyObject5Meth, [2, 3]);  
{ "myA" : 2, "myB" : 3 }  
> obj6b.foo();  
5  
> obj6.foo === obj6b.foo;  
true
```

Objet méthodes ?

fonction → objet ⇒ attributs

```
var MyObject7 = function (a, b) {  
  this.myA = a;  
  this.myB = b;  
};  
MyObject7.methodes = {  
  hello : function(s) { return "Hello " + s; },  
  foo : function () { return this.myA + this.myB; }  
};
```

```
var __new3 = function(f, args) {  
  var obj = Object.create(f.methodes);  
  f.apply(obj, args);  
  return obj;  
};
```

```
> obj7 = __new3(MyObject7, [1, 2]);  
{ "myA" :1, "myB" :2}  
> obj7.hello("World");  
"Hello World"  
> obj7.foo();  
3
```

Encore un peu de magie ?

arguments et closure. . .

```
var __new4 = function(f) {  
  return function() {  
    var obj = Object.create(f.methodes);  
    f.apply(obj, arguments);  
    return obj;  
  }  
};
```

```
var MyObject8 = __new4(MyObject7);
```

```
> obj8 = __new4(MyObject7)(1, 2);
```

```
{"myA" :1,"myB" :2}
```

```
> obj8.foo();
```

```
3
```

```
> obj8b = MyObject8(1, 2);
```

```
{"myA" :1,"myB" :2}
```

```
> obj8b.foo();
```

```
3
```

Objet JS

- ▶ `new`
- ▶ `Object.prototype`
- ▶ `instanceof` → égalité des prototypes

```
var MyObjFinal = function (a, b) {  
  this.myA = a;  
  this.myB = b;  
};  
MyObjFinal.prototype.hello = function(s) { return "Hello " + s; };  
MyObjFinal.prototype.foo = function () { return this.myA + this.myB; };
```

```
> objfin = new MyObjFinal(1, 2);  
{ "myA" : 1, "myB" : 2 }  
> objfin.hello("World");  
"Hello World"  
> objfin.foo();  
3  
> objfin instanceof MyObjFinal;  
true
```

 constructeur

`this` → objet global si pas `new`

appel direct « constructeur » ⇒ pollution espace global

Héritage

chaînage de prototype

```
var MySubObj = function (a, b, c) {  
    MyObjFinal.call(this, a, b);  
    this.myC = c;  
};  
MySubObj.prototype = Object.create(MyObjFinal.prototype);  
MySubObj.prototype.constructor = MySubObj;  
MySubObj.prototype.bar = function() { return this.myA + this.myC;};  
MySubObj.prototype.bye = function(s) { return "Goodbye " + s;};
```

```
> subobj = new MySubObj(1, 2, 3);  
{ "myA" :1,"myB" :2,"myC" :3}  
> subobj.hello("World");  
"Hello World"  
> subobj.foo();  
3  
> subobj.bar();  
4  
> subobj.bye("All...");  
"Goodbye All..."  
> subobj instanceof MySubObj;  
true  
> subobj instanceof MyObjFinal;  
true
```


sélection des méthodes héritées

```
Toto.prototype = Object.create(SuperClasse.prototype);  
Toto.prototype.foo = Foo.prototype.bar;  
Toto.prototype.hello = Message.prototype.hello;  
Toto.prototype.match = String.prototype.match;
```

Modification

- ▶ prototype → tous les objets
- ▶ objet (instance)

```
> Obj = function(a) { this.a = a; }; o = new Obj(1);  
{ "a" : 1 }  
> o.a;  
1  
> o.foo === undefined;  
true  
> o.bar === undefined;  
true  
> Obj.prototype.foo = function() { return "My a is " + this.a };  
o.foo();  
"My a is :1"  
> o.bar = function(b) { return this.a + b; }; o.bar(2);  
3
```

prototype \Rightarrow délégation automatique

- ▶ ES6 \Rightarrow syntaxe plus « habituelle »
- ▶ sucre syntaxique
- ▶ « dessous » : fonctions et prototypes !

Classes

```
var Person = function(name) {  
  if (!(this instanceof Person)) {  
    throw new TypeError("Classes can't be function-called");  
  }  
  this.name = name;  
}  
Person.prototype.describe = function() {  
  return "My name is " + this.name;  
}
```

```
class Person {  
  constructor(name) {  
    this.name = name;  
  }  
  describe() {  
    return `My name is ${this.name}`;  
  }  
}
```

Classes

```
> typeof Person
"function"
> Person.prototype.constructor === Person
true
> z = new Person("Zaphod")
{"name" : "Zaphod"}
> z instanceof Person
true
> typeof z
"object"
```

- ▶ `Object.defineProperty`
- ▶ `static`
- ▶ `get` / `set`

Extension

```
var Student = function(name, id) {  
  Person.call(this, name);  
  this.studentNumber = id;  
}  
  
Object.setPrototypeOf(Student, Person);  
Student.prototype = Object.create(Person.prototype);  
Student.prototype.constructor = Student;  
Student.prototype.describe = function() {  
  return Person.prototype.describe.call(this) + " (" + this.studentNumber + ")";  
}
```

```
class Student extends Person {  
  constructor(name, id) {  
    super(name);  
    this.studentNumber = id;  
  }  
  describe() {  
    return super.describe() + '(id : ${this.studentNumber})';  
  }  
}
```


- ▶ API navigateur
- ▶ DOM
- ▶ modèle événementiel

- ▶ séparation comportement / structure
- ▶ action utilisateur → événement
- ▶ association (objet + événement) → fonction (ou `EventListener` → `handleEvent`)
- ▶ objet reçoit événement \Rightarrow exécution fonction

Association

✗ `<button onclick="action()" />`

≈ `theButton.onclick = action`

✓ `theButton.addEventListener('click', action, false)`

Handler

- ▶ fonction de 1 seul paramètre : objet `Event`
- ▶ instance de `EventListener` : méthode `handleEvent`

 `this`

Propagation

deux phases :

- ▶ capture : descente du DOM
- ▶ bubble : (default) remontée du DOM

Propagation

- ▶ `obj.dispatchEvent(evt)` envoie d'un événement
- ▶ `evt.preventDefault()` ; annule l'action normale
- ▶ `evt.stopPropagation()` ; interrompt la phase de remontée
- ▶ capture vs. bubble : `addEventListener(evt, fun, capture)`

Temps

- ▶ `setTimeout(fun, delais) ;`
- ▶ `setInterval(fun, periode) ;`

accès au navigateur

- ▶ fenêtres
- ▶ historique
- ▶ géolocalisation
- ▶ stockage local
- ▶ orientation
- ▶ réseau
- ▶ ...

 compatibilité

⇒ bibliothèques + Polyfills

p.ex. Modernizr (<http://modernizr.com/>)

Fenêtre

- ▶ objet `Window`
- ▶ 1 instance par zone d'affichage
- ▶ variable globale : `window`
- ▶ environnement par défaut (`this`)

Fenêtre

Attributs

- ▶ `console`
- ▶ `name`
- ▶ `navigator`
- ▶ `location`
- ▶ `status`
- ▶ `innerHeight` `innerWidth`

Fenêtre

Méthodes

- ▶ `alert(msg)`, `prompt(msg, def)`, `confirm(msg)`
- ▶ `open(url, nom, options)`
 - ▶ `toolbar=yes/no`
 - ▶ `location=yes/no`
 - ▶ `status=yes/no`
 - ▶ `menubar=yes/no`
 - ▶ `titleBar=yes/no`
 - ▶ `alwaysRaised=yes/no`
 - ▶ `resizeable=yes/no`
 - ▶ `screenX=N`, `screenY=N`, `outerWidth=N`, `outerHeight=N`
- ▶ `close()`, `focus()`



bloquage



Historique

- ▶ objet `History`
- ▶ `window.history`
- ▶ `go(x)` (relatif)
- ▶ `back()` \equiv `go(-1)`
- ▶ `forward()` \equiv `go(1)`

Historique

État

manipulation de la pile

-  HTML5
- ▶ `pushState(obj, titre, url)`  même origine
 - ▶ `replaceState`
 - ▶ attribut `state`
 - ▶ événement `window.onpopstate`

Historique

Pourquoi ?

Principe :

Ne pas perturber l'utilisateur

⇒ maintenir les conventions

single page app

⇒ navigation Ajax

✓ maintient l'historique (bouton)

Et aussi...

- ▶ geoloc
- ▶ stockage local
- ▶ orientation
- ▶ notifications
- ▶ ...

<https://developer.mozilla.org/en-US/docs/WebAPI>

Ajax

- requêtes HTTP en JS
- objet `XMLHttpRequest`

- ▶ pas de rechargement
- ▶ asynchrone
- ▶ passage objets (JSON)

```
var req = new XMLHttpRequest();
req.onreadystatechange = function() {
  if (req.readyState == 4) {
    if (req.status == 200) {
      //...
    } else {
      // err
    }
  }
};
req.open('GET', 'http ://www.example.com/', true);
req.send(null);
```

État

- ▶ 0 (uninitialized)
- ▶ 1 (loading)
- ▶ 2 (loaded)
- ▶ 3 (interactive)
- ▶ 4 (complete)

Réponse

- ▶ `responseText`
- ▶ `responseXML`
- ▶ `getResponseHeader(nom)`
- ▶ `setRequestHeader(nom, valeur)`

Fetch

- ▶ plus simple que `XMLHttpRequest`
- ▶ utilise les promesses pour l'asynchrone plutôt que l'événementiel
- ▶ https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

DOM

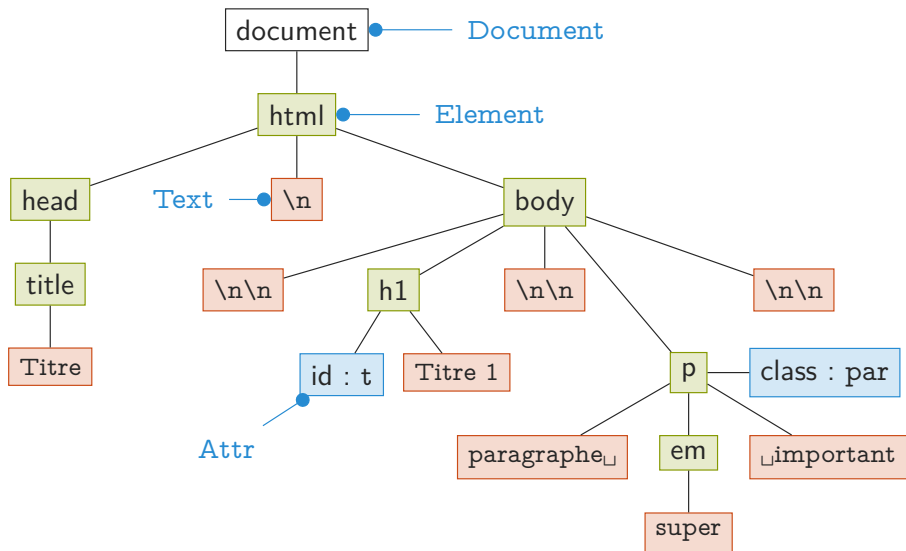
- ▶ *Document Object Model*
- ▶ représentation objet du document
- ▶ arbre de nœuds ([Node](#))
- ▶ langage de programmation quelconque
- ▶ xml quelconque

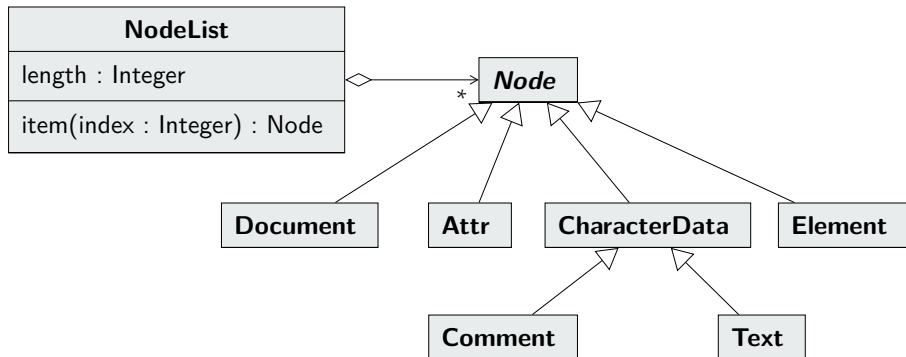
```
<html>
<head><title>Titre</title></head>
<body>

<h1 id="t">Titre 1</h1>

<p class="par">paragraphe <em>super</em> important</p>

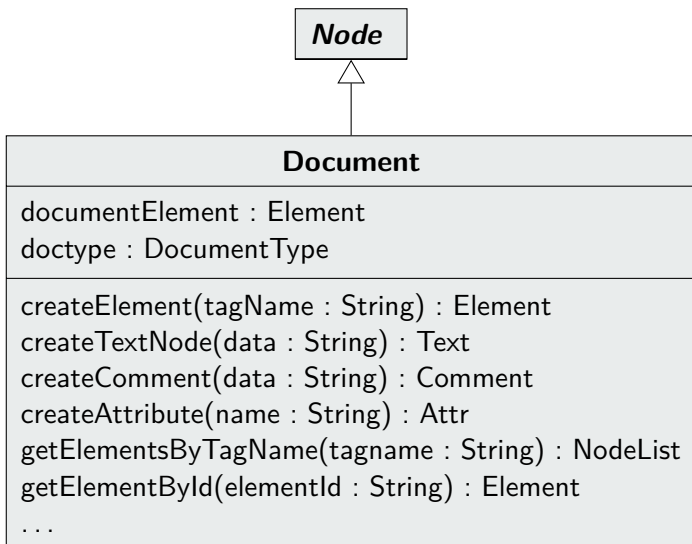
</body>
</html>
```



Node

nodeName : String
nodeValue : String
nodeType : Integer
parentNode : Node
childNodes : NodeList
firstChild : Node
lastChild : Node
previousSibling : Node
nextSibling : Node
attributes : NamedNodeMap
ownerDocument : Document



Element

- ▶ `tagName`
- ▶ `hasAttribute(nom)`, `getAttribute(nom)`, `getAttributeNode(nom)`
- ▶ `getElementsByTagName`, `getElementById`,
`getElementsByClassName`, `querySelector`, `querySelectorAll`

- ▶ <https://www.w3.org/TR/dom/>
- ▶ https://developer.mozilla.org/fr/docs/Web/API/Document_Object_Model