

FORMATION WEB

PROTOCOLE HTTP & LANGAGE HTML5



http://



1959-68

Le programme DARPA naît pendant la **guerre froide** de la peur d'une guerre nucléaire :

- Pour contrer la faiblesse du **système centralisé** versus distribué
- Proposition d'un **maillage d'ordinateurs** (1964, P. Baran)
- 1^{ère} communication tél. **entre 2 machines** : 1965



1969

Création d'ARPANET (4 noeuds)

1970 à 1982

Apparition du **courrier électronique**.

Communications internationales (Angleterre, Norvège).

Apparition de **TCP/IP** (1974)

1983

TCP/IP adopté comme standard ARPANET qui devient **Internet**



1983 à 1989

Expansion du réseau (autoroutes de l'information)

- Utilisation importante par les scientifiques
- En 1989, ARPANET est remplacé par TCP/IP



1990

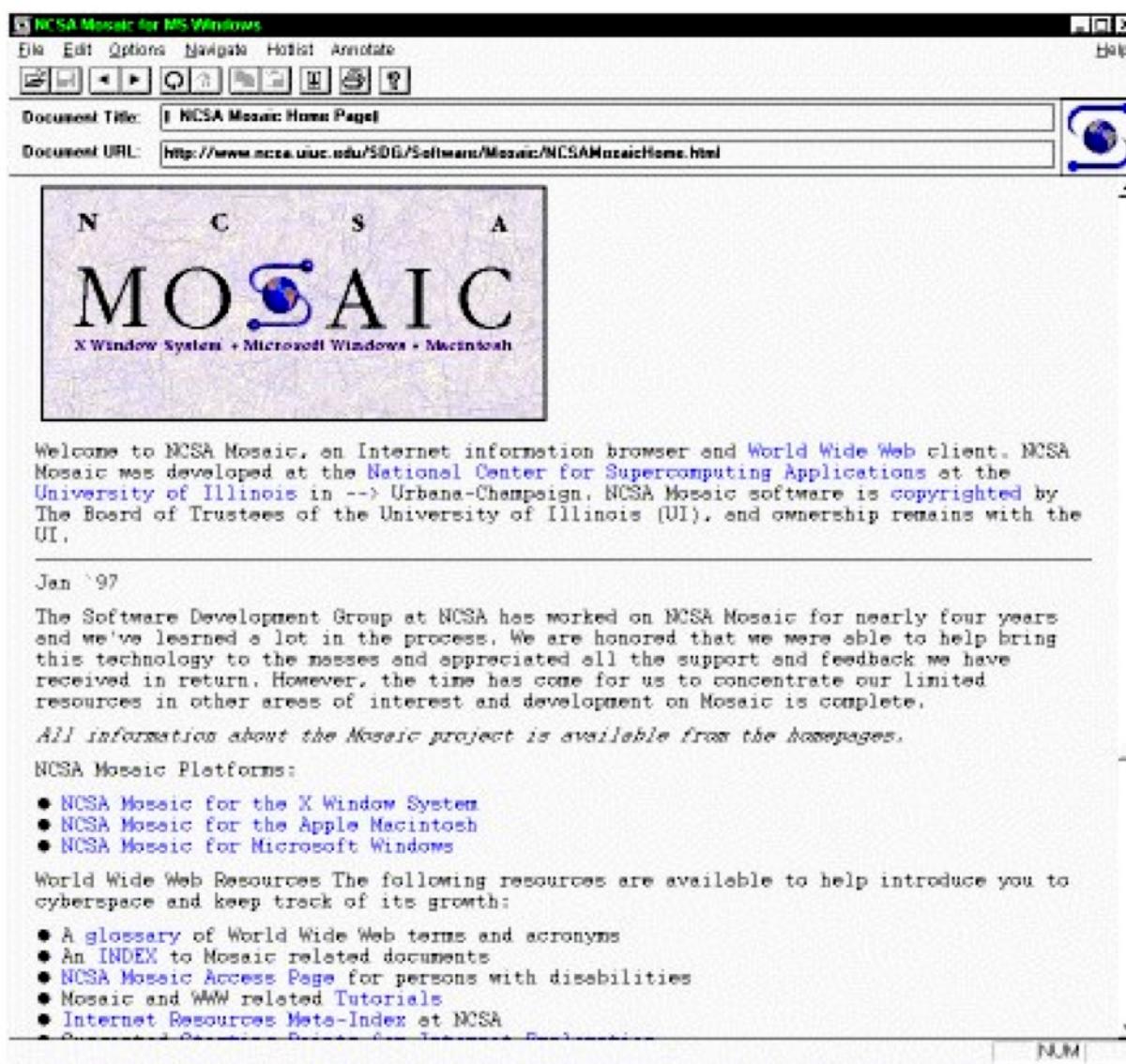
Le physicien Tim Berners Lee (CERN) étend le concept de lien **hypertexte** à Internet

- Naissance de HTML et HTTP
- **1er navigateur** : NCSA Mosaic

1995

Ouverture au **grand public** (Netscape et Internet Explorer)

LES NAVIGATEURS AU FIL DU TEMPS



NCSA Mosaic (1993)

LES NAVIGATEURS AU FIL DU TEMPS



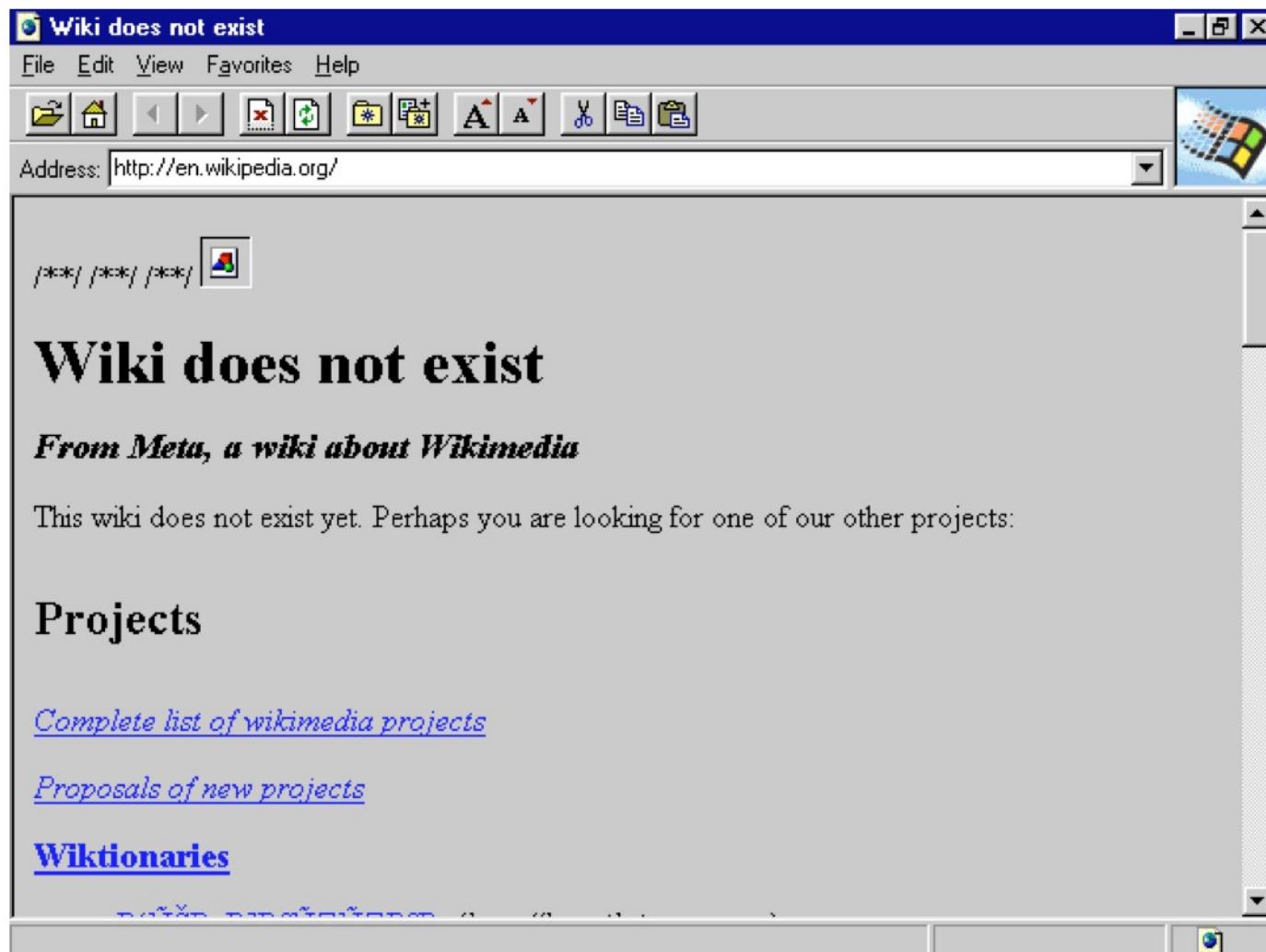
Netscape 1 (1994)

LES NAVIGATEURS AU FIL DU TEMPS



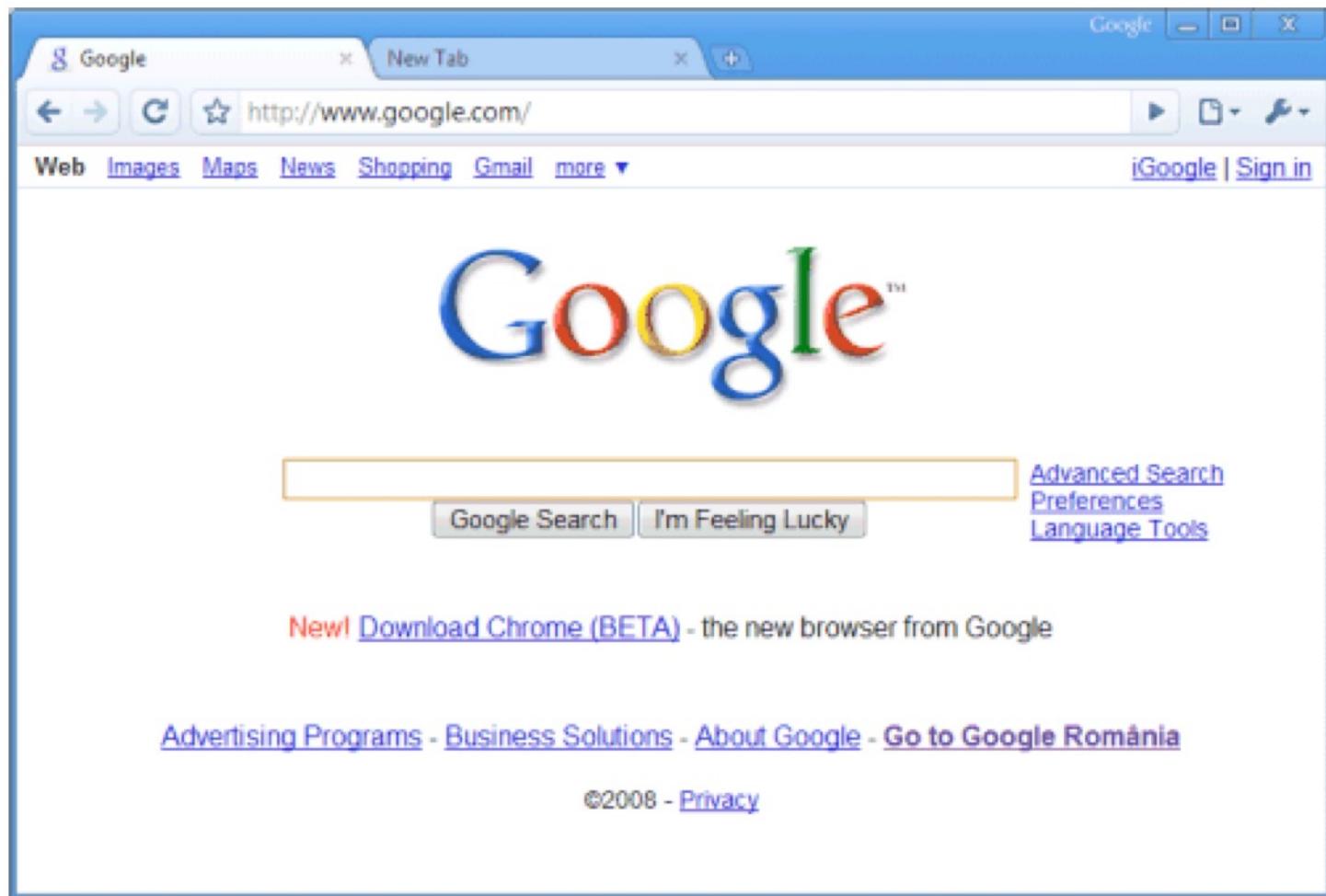
Mozilla 1

LES NAVIGATEURS AU FIL DU TEMPS



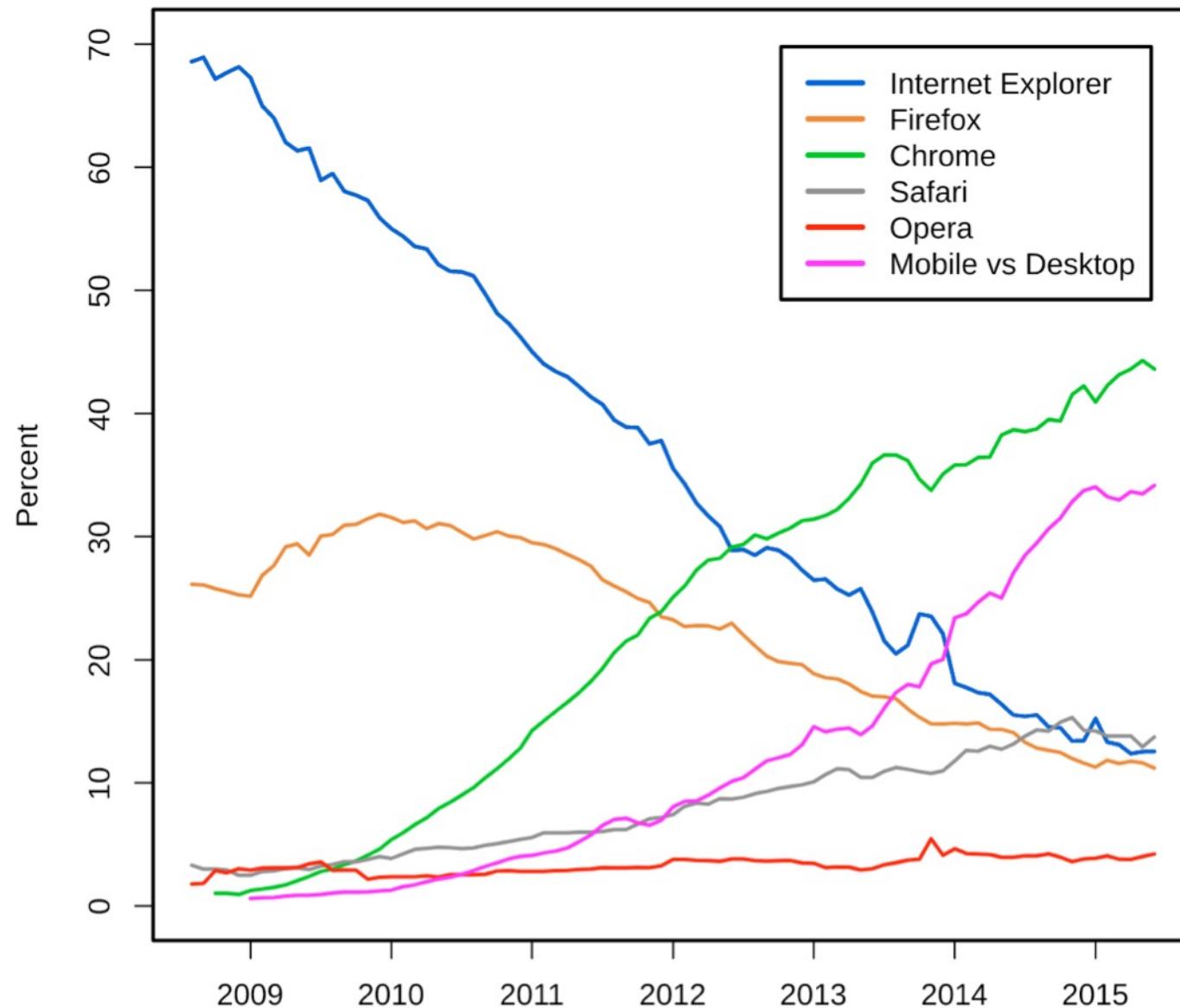
Internet Explorer 1 (1995)

LES NAVIGATEURS AU FIL DU TEMPS



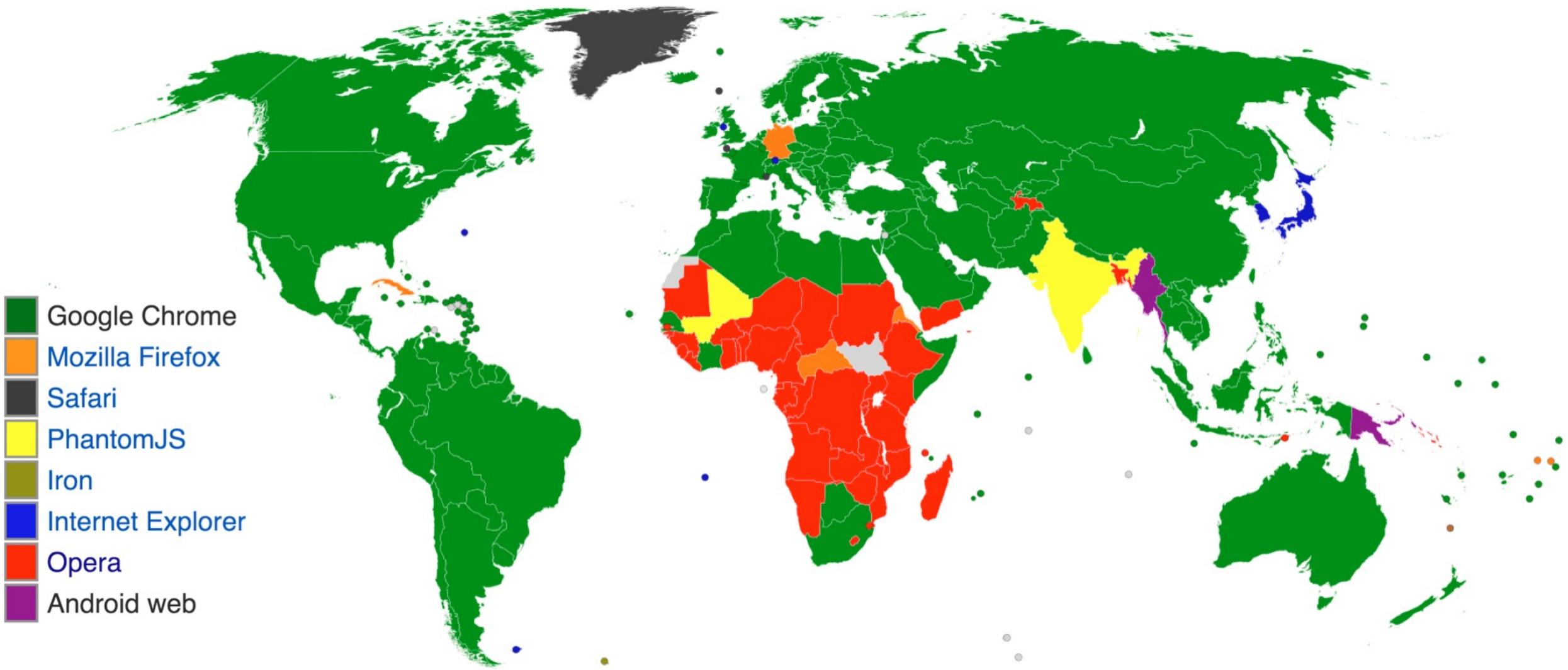
Google Chrome (2008)

ÉVOLUTION DE L'USAGE DES NAVIGATEURS



Source : StatCounter

PARTS DE MARCHÉ DES NAVIGATEURS



PhantomJS est un navigateur web sans interface graphique scriptable en JS utilisé pour automatiser des interactions avec des pages web.

CLIENT/SERVEUR

- Architecture Client/serveur
- Nécessité d'un protocole de communication : HTTP

Ressources statiques

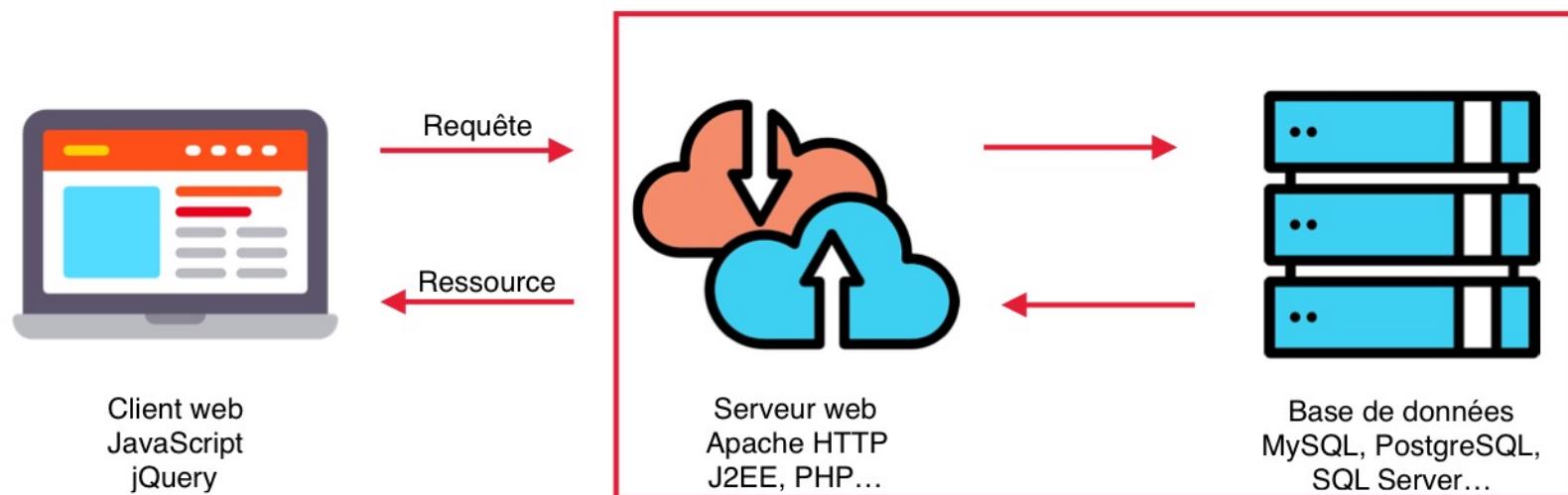
- HTML, images, son, vidéos, ressources dynamiques

Côté client

- Javascript/JQuery, Plugin, ActiveX...

Côté serveur

- Servlets/JSP, scripts serveur (php, ruby, .Net), ...





URL : Uniform Ressource Locator (Localisateur de Ressource Uniforme).

L'adresse d'un site internet est une URL.

Toutes les URL sont des URI, car il est par exemple possible d'identifier une personne avec son adresse, mais toutes les URI ne sont pas des URL.

ANALOGIE

Cela peut correspondre à une bibliothèque située dans un bâtiment.



URN : Uniform Ressource Name (Nom de Ressource Uniforme).

Identifiant unique pour une ressource.

La nomenclature des URN est précise et réglementée dans le but de maintenir l'unicité dans le temps et l'espace.

Une URN ne permet pas de savoir où est la ressource.

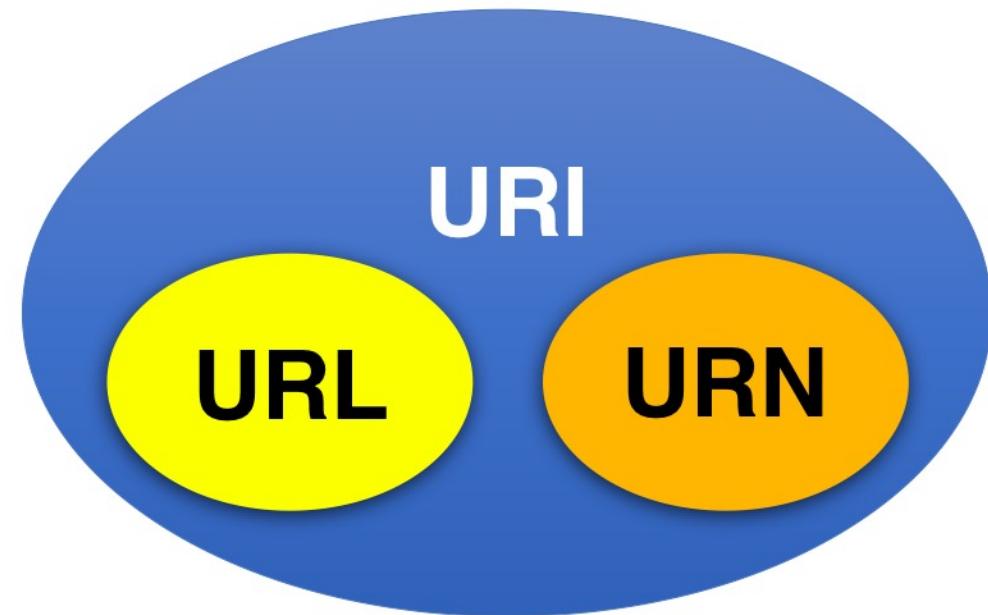
ANALOGIE

Cela peut correspondre au numéro ISBN d'un livre.

URI : Uniform Ressource Identifier (Identifiant de Ressource Uniforme).

Ce terme désigne un élément permettant d'identifier une ressource.

URI = URL + URN





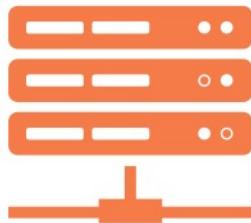
EXEMPLE

Cette « URL » de Gmail pointe vers un email spécifique est une URI.

URI : <https://mail.google.com/mail/u/0/#inbox/15bc536ba25bae26>

URL : <https://mail.google.com/mail/u/0/#inbox/15bc536ba25bae26>

URN : [15bc536ba25bae26](#)



<protocole>://<serveur>:<port>/<chemin>/<ressource>

Certains caractères doivent être encodés par % suivi de leur valeur hexadécimale en ISO Latin ou ASCII (ex : doc#2.html => doc%232.html)

EXEMPLES

http://www.novup.fr:80/fr/index.html

ftp://ftp.novup.fr/

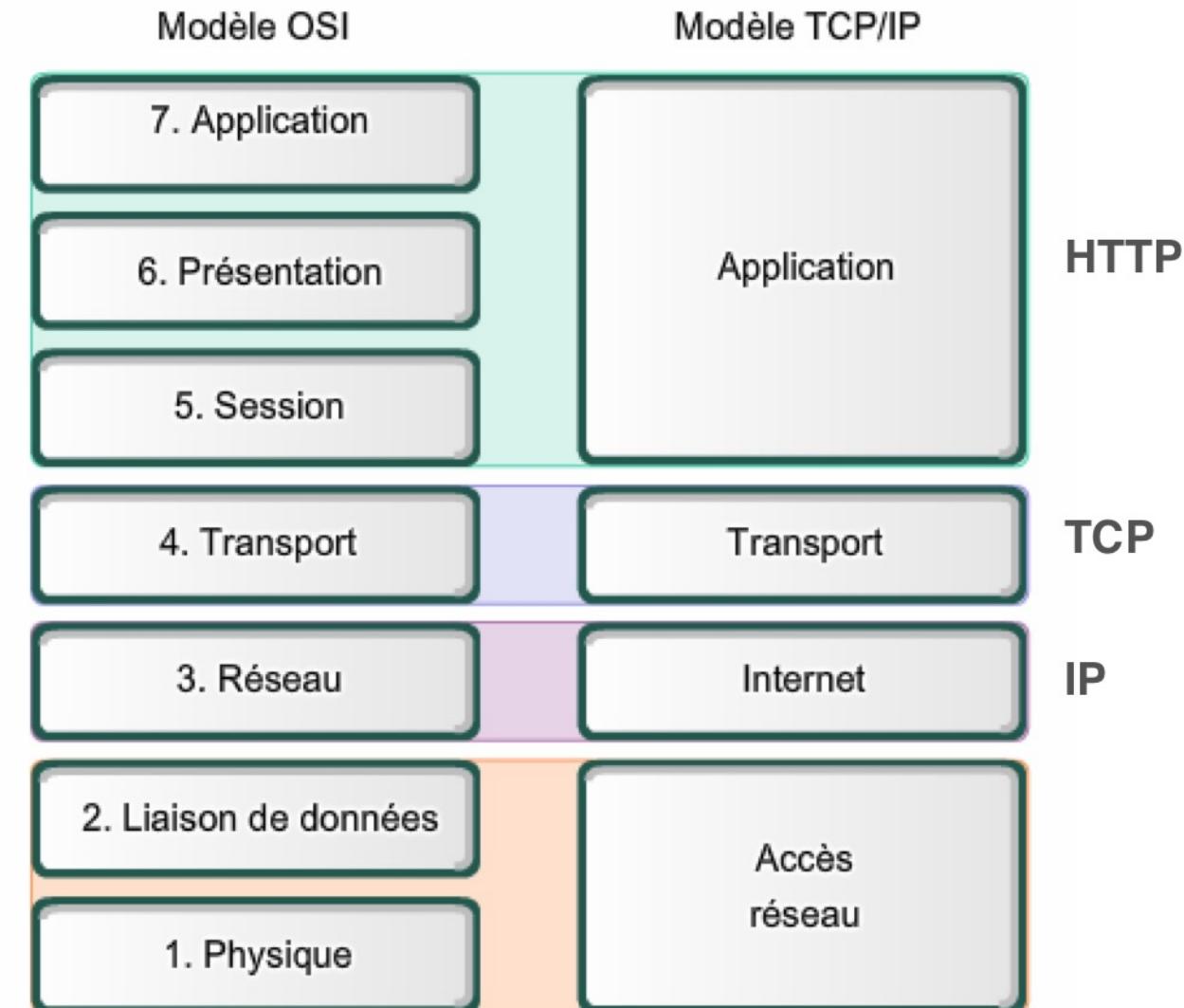
sftp://famisse@novup.fr/

file://home/cours/

mailto:francois.amisse@novup.fr

HTTP

- Hyper Text Transfer Protocol
- Est au-dessus de TCP/IP
- Transmission de ressources : morceau d'information identifié par une URL
- Fonctionnement client-serveur (requête/réponse)



LES MÉTHODES

GET

Demande une ressource.

HEAD

Demande des informations sur la ressource, sans demander la ressource elle-même.

POST

Modifie la ressource.

OPTIONS

Obtient les options de communication d'une ressource ou du serveur en général.

CONNECT

Utilise un proxy comme tunnel de communication.

TRACE

Demande au serveur de retourner ce qu'il a reçu pour tester et effectuer un diagnostic sur la connexion.

PUT

Ajout d'une ressource sur le serveur.

DELETE

Suppression d'une ressource du serveur.

http

Ligne de statut : HTTP-Version Status-Code Reason-Phrase

- Status-Code : code numérique représentant le succès ou l'échec de la requête
- Reason-Phrase : texte expliquant le Status-Code

Code	Description	Code	Description
200	OK	400	Bad Request
201	Created	401	Unauthorized
202	Accepted	403	Forbidden
301	Moved Permanently	404	Not Found
303	See Other	410	Gone
304	Not Modified	500	Internal Server Error
307	Temporary Redirect	503	Service Unavailable



Après la méthode HTTP, vient l'URL qui est l'adresse de la page sur le serveur.

Elle est du type : "/repertoire/ressource.ext" (le fichier "ressource.ext" dans le dossier "repertoire" qui se situe à la racine du serveur).

Le dernier élément de Ligne de commande est la version de HTTP utilisée, par exemple : "HTTP/1.1".

Exemple complet

GET /repertoire/page.html HTTP/1.1

Pour terminer la requête, on envoie le corps de requête.

Il peut contenir, par exemple, le contenu d'un formulaire HTML envoyé en POST.

Dans le cas du formulaire, les variables ont un nom et une valeur, comme l'en-tête Cookie, et les différentes variables sont séparées par des *espaces blanches* : "&" (notez que les variables passées par GET sont également séparées par "&").

Exemple

variable=valeur&variable2=valeur2

HEAD demande l'envoi de l'en-tête de la réponse sans son contenu. Il est intéressant si on veut, par exemple, récupérer des informations sur un fichier ou sur le serveur.

Exemple

HEAD /fichier.ext HTTP/1.1

Host: www.site.com

Connection: Close <nouvelle ligne>

GET est la méthode permettant de récupérer le contenu d'un fichier.

Exemple

GET /fichier.ext HTTP/1.1

Host: www.site.com

Connection: Close <nouvelle ligne>

Pour transmettre les variables, il faut remplacer la ligne de commande par quelque chose du type :

GET /fichier.ext?var1=valeur1&var2=valeur2 HTTP/1.1

En séparant la liste de variables de l'adresse de la page par un "?" et en séparant chaque variable de la suivante par "&".

POST envoie des données via le corps de la requête en spécifiant le type des données.

Pour un formulaire HTML, il y a 2 valeurs possibles :

- **application/x-www-form-urlencoded** qui est la valeur par défaut
- **multipart/form-data** utilisée notamment pour le download de fichiers.

Il vous faudra également spécifier la longueur du corps.

Exemple

POST /fichier.ext HTTP/1.1

Host: www.site.com

Connection: Close

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

Content-Length: 33

<nouvelle ligne>

variable=valeur&variable2=valeur2

Le Corps de réponse contient le contenu du fichier comme le HTML d'une page par exemple.

Exemple de réponse

HTTP/1.1 200 OK

Date: Thu, 11 Jan 2007 14:00:36 GMT

Server: Apache/2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4

Connection: close

Transfer-Encoding: chunked

Content-Type: text/html; charset=ISO-8859-1

<!DOCTYPE html>

<html lang="fr">

<head>

<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />

<meta http-equiv="pragma" content="no-cache" />

<title>Bienvenue !</title>

Le protocole HTTPS est un **protocole crypté** qui permet l'échange des données de manière sécurisée.

On peut vérifier qu'un site est bien en protocole HTTPS en regardant simplement l'URL dans la barre d'adresse.

https://www.google.fr

Le protocole est le même que celui du HTTP mais le serveur fournit une clé cryptographique au client.

Le cryptage et le décryptage a donc lieu entre le serveur et le client.

REST

REST est un style d'architecture qui repose sur le protocole HTTP.

On accède à une ressource (par son URI unique) pour procéder à diverses opérations HTTP :

- **GET** (lecture)
- **POST** (écriture)
- **PUT** (modification)
- **DELETE** (suppression)

Supposons que nous voulions réaliser un serveur REST pour gérer une pizzéria.

Nous devrions pouvoir ajouter (**POST**), modifier (**PUT**), lire (**GET**) et supprimer (**DELETE**) des clients, des commandes ou des pizzas.

L'adresse de notre pizzéria représente le « **point terminal** » (**endpoint**) :

<http://pizzeria/>

Le point terminal n'est ni plus ni moins que l'adresse du webservice.

Notre pizzéria contient des ressources, en particulier des pizzas, qui pourront être manipulés comme ceci :

http://point_terminal/nom_de_ressource/

Soit dans notre exemple **<http://pizzeria/pizzas/>**

Nous pourrons effectuer plusieurs manipulation sur ces pizzas :

- **Voir la fiche produit** : Requête de type GET sur **<http://pizzeria/pizzas/>** **ID_LA_PIZZA**
- **En créer** : Requête de type POST sur **<http://pizzeria/pizzas/>**. Le corps du message POST représente le contenu de la nouvelle pizza à créer. À la charge de la pizzéria d'affecter un identifiant à notre nouvelle pizza.
- **Les modifier** : Requête de type PUT sur **<http://pizzeria/pizzas/>** **ID_LA_PIZZA**. Le corps du message PUT représente le contenu modifié de la pizza d'identifiant ID_LA_PIZZA.
- **Les supprimer** : Requête de type DELETE sur **<http://pizzeria/pizzas/>** **ID_LA_PIZZA**.

FORMAT D'ÉCHANGE

REST n'impose pas de format d'échange particulier entre client et serveur.

Vous êtes libre de représenter vos données en **XML**, en **JSON** (JavaScript Object Notation) ou dans tout autre format de votre propre choix.

Les services REST permettent au client d'indiquer le format dans lequel ils souhaitent dialoguer :

- sous la forme d'un paramètre supplémentaire dans l'URL,
- ou grâce aux en-têtes HTTP dans le content-type.

L'architecture REST utilise les spécifications originelles du protocole HTTP, plutôt que de réinventer une sur-couche (comme le font SOAP ou XML-RPC par exemple).

Règle n°1 : l'URI comme identifiant des ressources

Règle n°2 : les verbes HTTP comme identifiant des opérations

Règle n°3 : les réponses HTTP comme représentation des ressources

Règle n°4 : les liens comme relation entre ressources

Règle n°5 : un paramètre comme jeton d'authentification

L'URI comme identifiant des ressources

REST se base sur les URI (Uniform Resource Identifier) afin d'identifier une ressource.

Ainsi une application se doit de construire ses URI (et donc ses URL) de manière précise, en tenant compte des contraintes REST.

Il est nécessaire de prendre en compte la hiérarchie des ressources et la sémantique des URL pour les éditer.

Liste de livres

NOK : <http://domaine.com/book>

OK : <http://domaine.com/books>

Filtre et tri sur les livres

NOK : <http://domaine.com/books/filtre/policier/tri/asc>

OK : <http://domaine.com/books?filtre=policier&tri=asc>

Affichage d'un livre

NOK : <http://domaine.com/book/display/87>

OK : <http://domaine.com/books/87>

Tous les commentaires sur un livre

NOK : <http://domaine.com/books/comments/87>

OK : <http://domaine.com/books/87/comments>

En construisant correctement les URI, il est possible de les trier, de les hiérarchiser et donc d'améliorer la compréhension du système.

Un commentaire pour un livre

<http://domaine.com/books/87/comments/1568>

Tous les commentaires pour un livre

<http://domaine.com/books/87/comments>

Un livre

<http://domaine.com/books/87>

Tous les livres

<http://domaine.com/books>

Les verbes HTTP comme identifiants des opérations

Il y a généralement 4 opérations possibles (CRUD) sur une ressource :

- Créer (create)
- Afficher (read)
- Mettre à jour (update)
- Supprimer (delete)

HTTP propose les verbes correspondant :

- Créer (create) => **POST**
- Afficher (read) => **GET**
- Mettre à jour (update) => **PUT**
- Supprimer (delete) => **DELETE**

Exemple d'URL pour une ressource donnée (un livre par exemple) :

Créer un livre

NOK : POST http://domaine.com/books/create

OK : POST http://domaine.com/books

Afficher un livre

NOK : GET http://domaine.com/books/display/87

OK : GET http://domaine.com/books/87

Mettre à jour un livre

NOK : PUT http://domaine.com/books/editer/87

OK : PUT http://domaine.com/books/87

Supprimer un livre

NOK : DELETE http://domaine.com/books/87/delete

OK : DELETE http://domaine.com/books/87

Les réponses HTTP comme représentation des ressources

Il est important d'avoir à l'esprit que la réponse envoyée n'est pas une ressource, c'est la représentation d'une ressource.

Ainsi, une ressource peut avoir plusieurs représentations dans des formats divers : HTML, XML, CSV, JSON, etc.

C'est au client de définir quel format de réponse il souhaite recevoir via l'entête **Accept**.

Il est possible de définir plusieurs formats.

Exemples

Réponse en HTML

GET /books

Host: domaine.com

Accept: **text/html**

Réponse en XML

GET /books

Host: domaine.com

Accept: **application/xml**

Les liens comme relation entre ressources

Les liens d'une ressource vers une autre ont tous une chose en commun : ils indiquent la présence d'une relation.

Il est cependant possible de la décrire afin d'améliorer la compréhension du système.

L'attribut **rel** doit être spécifier sur tous les liens.

La liste complète des relations sur le site de l'IANA : <http://www.iana.org/assignments/link-relations/link-relations.xml>

* Internet Assigned Numbers Authority (IANA) - Responsable de la coordination des routeurs DNS, de l'adressage IP et des ressources du protocole IP

On peut alors parler d'hypermedias.

Exemple de réponse en XML d'une liste paginée de livres

```
<?xml>
<search>
  <link rel="self" title="self" href="http://domaine.com/books?q=policier&page=1&c=5" />
  <link rel="next" title="next" href="http://domaine.com/books?q=policier&page=2&c=5" />
  <link rel="last" title="last" href="http://domaine.com/books?q=policier&page=4&c=5" />
  <book>
    //...
  </book>
</search>
```

Un paramètre comme jeton d'authentification

Le jeton d'authentification permet d'authentifier une requête REST.

Chaque requête est envoyée avec un jeton (token) passé en paramètre GET de la requête.

Ce jeton temporaire est obtenu en envoyant une première requête d'authentification puis en le combinant avec les requêtes suivantes.

RÈGLE #5 Un paramètre comme jeton d'authentification

1. Demande d'authentification

GET /users/123/authenticate?

pass=lkdnnssdf54d47894f5123002fds2sd360s0

<?xml>

<user>

<id>1234</id>

<name>Lucy GOLDBERG</name>

</user>

<token>

fsd531gfd5g5df31fdg3g3df45

</token>

RÈGLE #5 Un paramètre comme jeton d'authentification

2. Accès aux ressources

Ce token est ensuite utilisé pour générer un hash de la requête de cette façon :

hash = SHA1(token + requete)

hash = SHA1(fsd531gfd5g5df31fdg3g3df45 + "GET /books")

hash = 456894ds4q15sdq156sd1qsd1qsd156156

C'est ce hash qui est passé comme jeton afin de valider l'authentification pour cette requête :

GET /books?

user=123&hash=456894ds4q15sdq156sd1qsd1qsd156156

HTML5



PRÉAMBULE

HYPertext Markup Language

Le HTML fait son apparition dès **1991** lors du lancement du Web pour gérer et organiser le contenu.

Le langage xHTML est une variante du HTML qui se veut plus rigoureuse et qui est donc un peu plus délicate à manipuler.

C'est un format de données standardisé par le W3C qui permet de représenter une page web.

Le **HTML5** est la dernière variante du langage HTML.

QU'EST-CE QUE L'HYPertexte ?

Un document hypertexte est un document qui contient des **hyperliens** et des **nœuds**.

On dit que le HTML est un **langage de balisage** car il contient des balises (comme `<html>` ou `<body>`) qui sont utilisées pour organiser et structurer le texte, les images, les liens, etc. et donner du sens (on parle de HTML sémantique) aux informations incluses dans une page web.

Il est souvent utilisé conjointement avec le format de présentation **CSS** (Cascading Style Sheets ou « feuilles de style en cascade »).

QU'EST-CE QUE L'HYPertexte ?

HTML



Structure

CSS



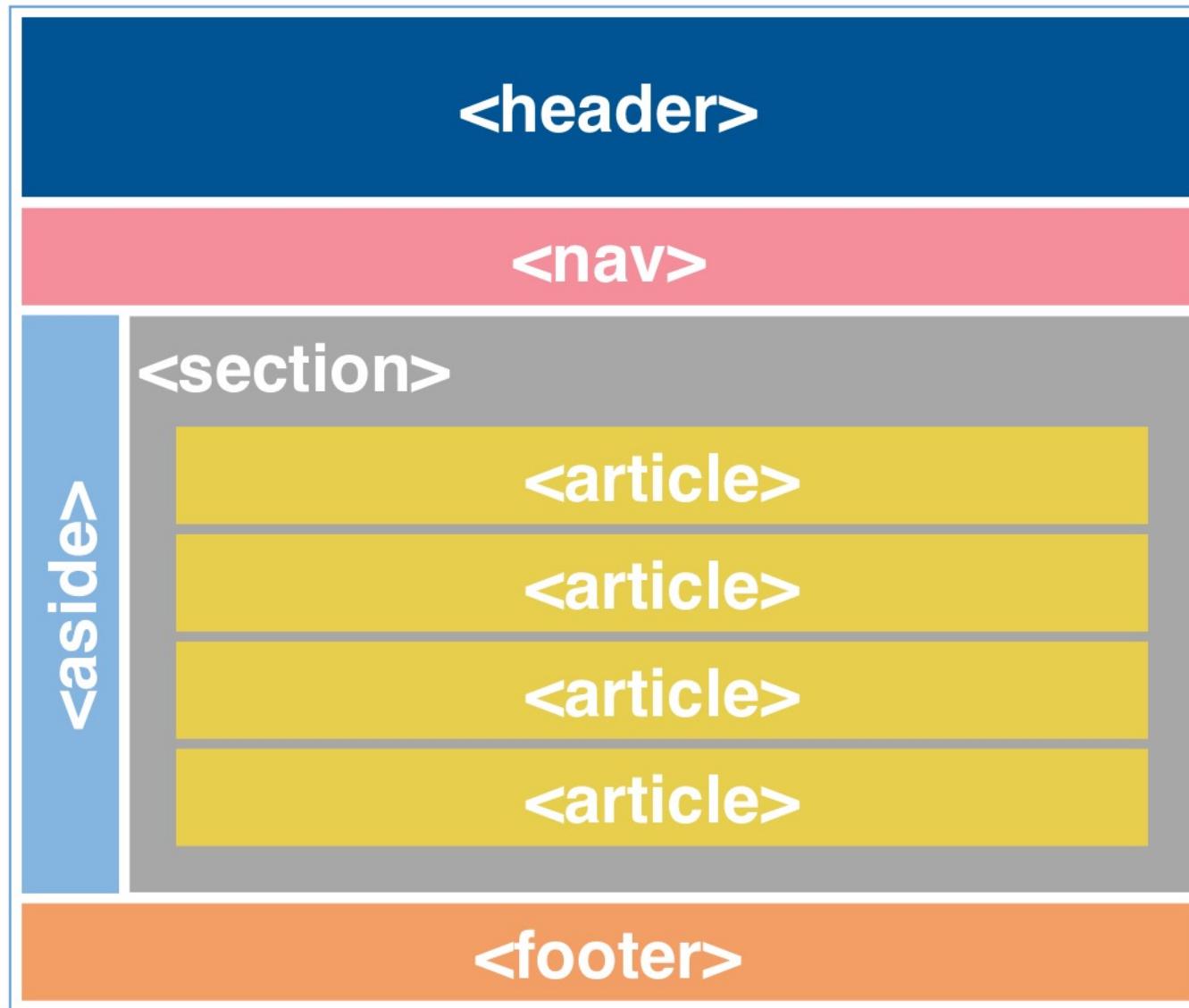
Présentation

LES ÉLÉMENTS SÉMANTIQUES DE STRUCTURATION

Le HTML5 introduit des éléments permettant de structurer et de segmenter des portions de votre page HTML :

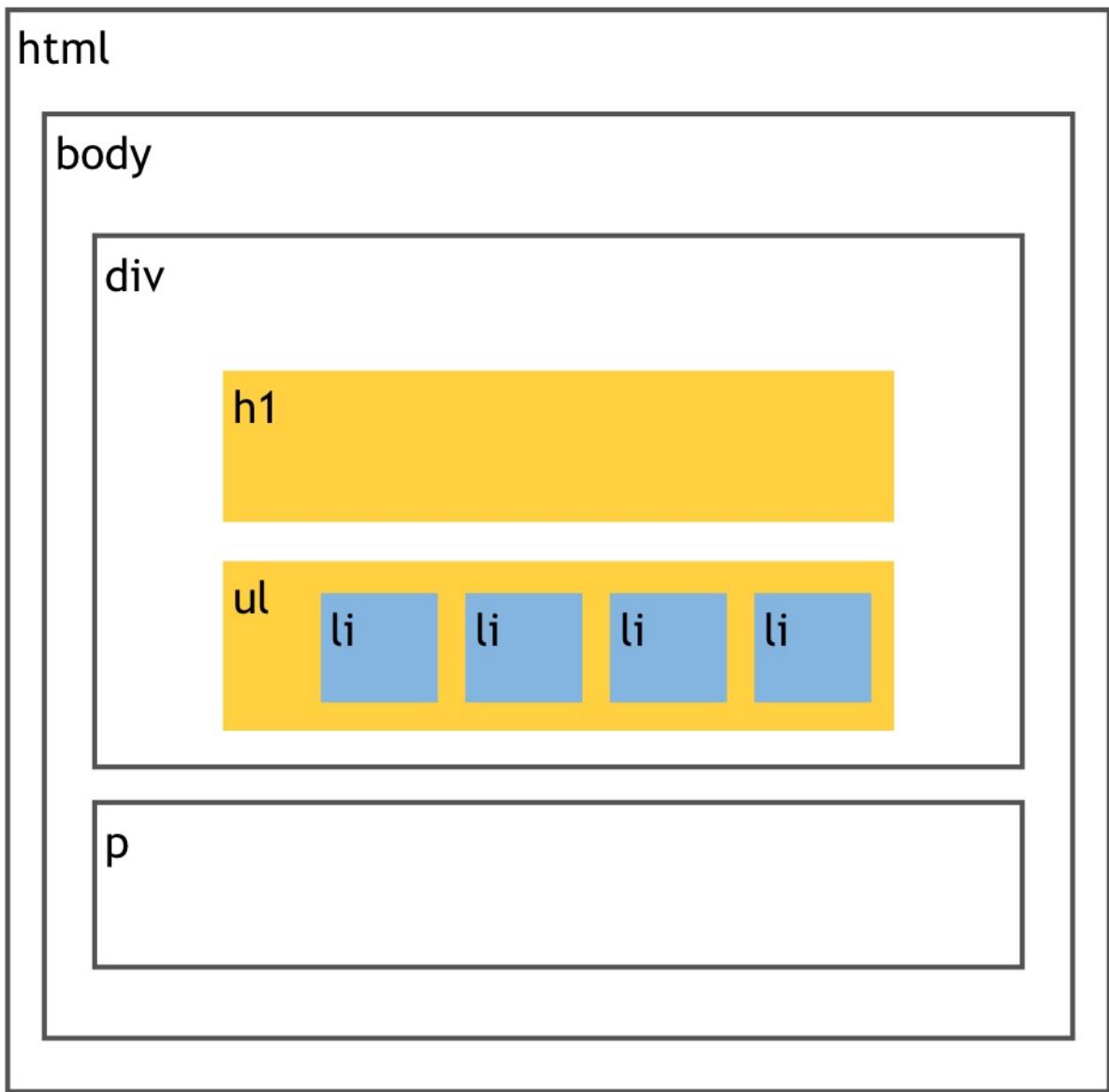
- **HEADER** pour l'entête,
- **NAV** pour la barre de navigation,
- **ASIDE** pour la barre latérale,
- **FOOTER** pour le pied de page,
- **ARTICLE** pour les articles,
- **SECTION** pour délimiter des zones...

LES ÉLÉMENTS SÉMANTIQUES DE STRUCTURATION



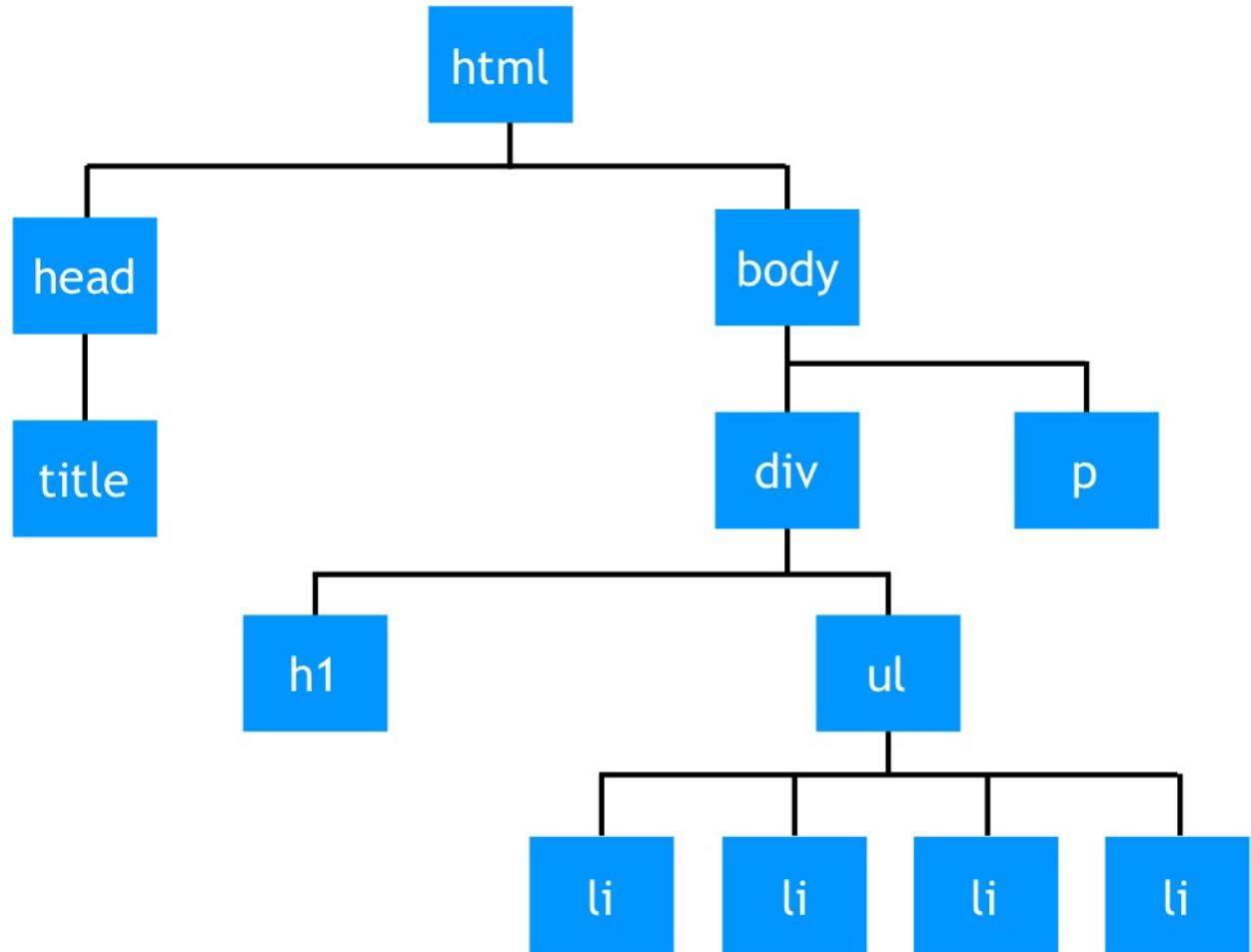
STRUCTURE D'UNE PAGE HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
  </head>
  <body>
    <div id="header">
      <h1>Titre de la page</h1>
      <ul id="menu">
        <li>Accueil</li>
        <li>Portfolio</li>
        <li>Blog</li>
        <li>Contact</li>
      </ul>
    </div>
    <p>Ceci est une page d'exemple.</p>
    <!-- Commentaire pour l'exemple -->
  </body>
</html>
```



STRUCTURE D'UNE PAGE HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
  </head>
  <body>
    <div id="header">
      <h1>Titre de la page</h1>
      <ul id="menu">
        <li>Accueil</li>
        <li>Portfolio</li>
        <li>Blog</li>
        <li>Contact</li>
      </ul>
    </div>
    <p>Ceci est une page d'exemple.</p>
    <!-- Commentaire pour l'exemple -->
  </body>
</html>
```



```
<!doctype html>
```

```
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
  </head>
  <body>
    <div id="header">
      <h1>Titre de la page</h1>
      <ul id="menu">
        <li>Accueil</li>
        <li>Portfolio</li>
        <li>Blog</li>
        <li>Contact</li>
      </ul>
    </div>
    <p>Ceci est une page d'exemple.</p>
    <!-- Commentaire pour l'exemple -->
  </body>
</html>
```

Déclaration d'un DOCTYPE

Le DOCTYPE ou document type declaration est la grammaire, la syntaxe à utiliser pour interpréter votre fichier HTML.

C'est une indication donnée au navigateur.

Le DOCTYPE est la grammaire utilisée par une page web.

Il en existe plusieurs :

- **Doctype HTML 4.01**

**<!DOCTYPE HTML PUBLIC “-//W3C//DTD HTML 4.01//EN”
“http://www.w3.org/TR/html4/strict.dtd”>**

- **Doctype XHTML 1.0**

**<!DOCTYPE HTML PUBLIC “-//W3C//DTD xHTML 1.0//EN”
“http://www.w3.org/TR/xhtml1/strict.dtd”>**

- **Doctype HTML5**

<!DOCTYPE html>

Balises en paires (ouvrantes et fermantes)

<balise>contenu</balise>

<p>Du contenu ici. Google.</p>

<div>Du contenu ici</div>

Balises orphelines (autofermantes)

<balise />

<meta charset="utf-8" />

BALISES INDISPENSABLES

```
<!doctype html>  
  
<html>  
  <head>  
    <meta charset="UTF-8" />  
    <title>Bonjour</title>  
  </head>  
  <body>  
    <div id="header">  
      <h1>Titre de la page</h1>  
      <ul id="menu">  
        <li>Accueil</li>  
        <li>Portfolio</li>  
        <li>Blog</li>  
        <li>Contact</li>  
      </ul>  
    </div>  
  
    <p>Ceci est une page d'exemple.</p>  
    <!-- Commentaire pour l'exemple -->  
  </body>  
</html>
```

Englobe tout le document

BALISES INDISPENSABLES

```
<!doctype html>
<html>

  <head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
  </head>

  <body>
    <div id="header">
      <h1>Titre de la page</h1>
      <ul id="menu">
        <li>Accueil</li>
        <li>Portfolio</li>
        <li>Blog</li>
        <li>Contact</li>
      </ul>
    </div>

    <p>Ceci est une page d'exemple.</p>
    <!-- Commentaire pour l'exemple -->
  </body>
</html>
```

Métadonnées sur la page
Encodage

Titre de la page

*Éventuellement les scripts ou
la feuille de style*

BALISES INDISPENSABLES

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Bonjour</title>
  </head>
  <body>
  <div id="header">
    <h1>Titre de la page</h1>
    <ul id="menu">
      <li>Accueil</li>
      <li>Portfolio</li>
      <li>Blog</li>
      <li>Contact</li>
    </ul>
  </div>
  <p>Ceci est une page d'exemple.</p>
  <!-- Commentaire pour l'exemple -->
  </body>
</html>
```

Balise principale de la page.
Contient les données affichées par le navigateur.

EXEMPLES

Code HTML

Text normal

```
<strong>Texte GRAS</strong>
```

```
<i>Texte ITALIQUE</i>
```

```
<u>Texte SOULIGNE</u>
```

```
<table>
```

```
<tr><td>1</td><td>2</td></tr>
```

```
<tr><td>3</td><td>4</td></tr>
```

```
</table>
```

```
<hr> ou <hr /> (en xHTML)
```

```
<img src='picture.jpg'>
```

Résultat

Text normal

Texte GRAS

Texte ITALIQUE

Texte SOULIGNÉ

1	2
3	4



BALISES PRINCIPALES

Balise d'en-tête	<code><link /></code> <code><meta /></code> <code><script></code> <code><style></code>	Liaison avec une feuille de style Métadonnées de la page (charset, mots clés, etc.) Code JavaScript Code CSS
Balises sectionnantes <u>Intérêt</u> <ul style="list-style-type: none">• Lisibilité du code pour les développeurs• Référencement (SEO)	<code><header></code> <code><nav></code> <code><footer></code> <code><section></code> <code><article></code> <code><aside></code>	En-tête Liens principaux de navigation Pied de page Section de page Article (contenu autonome) Informations complémentaires
Balises génériques	<code><div></code> <code></code>	Balise générique de type block Balise générique de type inline

BALISES PRINCIPALES

Balise de structuration du texte	<h1>...</h6> <p> <blockquote> <q> <sup> <sub> <a> <pre>	Titres de niveaux 1 à 6 Paragraphe Citation longue Citation courte Exposant Indice Mise en gras fort Lien hypertexte Retour à la ligne Affichage formaté (respect des espaces et tabulations)
Balises sectionnantes	 	Liste à puces Liste numérotée Élément de liste

BALISES PRINCIPALES

Balise de tableaux	<code><table></code> <code><caption></code> <code><tr></code> <code><th></code> <code><td></code> <code><thead></code> <code><tbody></code> <code><tfoot></code>	Tableau Titre du tableau Ligne de tableau Cellule d'en-tête Cellule Section de l'en-tête du tableau Section du corps du tableau Section du pieds du tableau
Balises de formulaires	<code><form></code> <code><input /></code> <code><textarea></code> <code><select></code> <code><option></code>	Formulaire Champs de formulaire (texte, nombre, email, ...) Zone de saisie multilignes Liste déroulante Élément d'une liste déroulante

BALISES PRINCIPALES

Balises de médias	<code></code> <code><video></code> <code><audio></code> <code><source></code>	Image Vidéo Son Lien et format des sources des balises vidéo et son
Commentaires	<code><!-- ... --></code>	Commentaires sur 1 ou +ieurs lignes

Les attributs peuvent être vus comme les options des balises. Ils viennent compléter pour donner des informations supplémentaires.

L'attribut se place après le nom de la balise ouvrante et a le plus souvent une valeur, comme ceci :

<balise attribut="valeur">

Exemple pour afficher une image

LES TABLEAUX

Aujourd’hui, les tableaux HTML ne sont utilisés que pour des **tableaux de données**.

Il est plutôt conseillé de faire la mise en page en CSS avec l'aide de `<div>`.

Plus **souple**, plus **précis**, plus **facile** et surtout plus **évolutif**, tout ce qui est visuel doit être reporté sur le CSS.

Un tableau HTML ne doit servir que de base d'un tableau.

Créer un tableau avec la balise <table>

```
<table>  
<tr>  
<th>Entête 1</th>  
<th>Entête 2</th>  
</tr>  
<tr>  
<td>Cellule 1 de la ligne  
2</td>  
<td>Cellule 2 de la ligne  
2</td>  
</tr>  
</table>
```

En tête 1	Entête 2
Cellule 1 de la ligne 2	Cellule 2 de la ligne 2

Colspan, fusionner des cellules horizontalement

```
<table>
<tr>
<td>Cellule 1 de la ligne 1</td>
<td>Cellule 2 de la ligne 1</td>
</tr>
<tr>
<td colspan="2">
FUSION de celulle
</td>
</tr>
<tr>
<td>Cellule 1 de la ligne 3</td>
<td>Cellule 2 de la ligne 3</td>
</tr>
</table>
```

Cellule 1 de la ligne 1	Cellule 2 de la ligne 1
FUSION de celulle	
Cellule 1 de la ligne 3	Cellule 2 de la ligne 3

Rowspan, fusionner des cellules verticalement

```
<table>
<tr>
<td>Cellule 1 de la ligne 1</td>
<td>Cellule 2 de la ligne 1</td>
</tr>
<tr>
<td rowspan="2">
FUSION cellule
</td>
<td>Cellule 2 de la ligne 2</td>
</tr>
<tr>
<td>Cellule 2 de la ligne 3</td>
</tr>
</table>
```

Cellule 1 de la ligne 1	Cellule 2 de la ligne 1
FUSION cellule	Cellule 2 de la ligne 2
	Cellule 2 de la ligne 3

Légende en haut des tableaux HTML

```
<table>  
<caption>Légende en haut</caption>  
<tr>  
<td>Cellule 1 de la ligne 1</td>  
<td>Cellule 2 de la ligne 1</td>  
</tr>  
<tr>  
<td>Cellule 1 de la ligne 2</td>  
<td>Cellule 2 de la ligne 2</td>  
</tr>  
</table>
```

Légende en haut	
Cellule 1 de la ligne 1	Cellule 2 de la ligne 1
Cellule 1 de la ligne 2	Cellule 2 de la ligne 2

Mise en forme du tableau HTML

Il est possible de faire de la mise en forme en HTML pour les tableaux, mais je vous conseille plutot de le faire en CSS. Vous pouvez donc definir la taille d'une cellule ou la taille de la bordure. Cette information est purement informative, ne l'utilisez pas.

```
<table border="10">  
  
<tr>  
  
  <td width="10">10px</td>  
  <td>Cellule 2 de la ligne 1</td>  
</tr>  
  
<tr>  
  
  <td height="200">200px</td>  
  <td>Cellule 2 de la ligne 2</td>  
</tr>  
  
</table>
```

10px	Cellule 2 de la ligne 1
200px	Cellule 2 de la ligne 2

Les alignements

Pour aligner sur l'axe horizontal, utiliser l'attribut **ALIGN**.

Les valeurs possibles sont : **LEFT, CENTER, RIGHT**

Exemple

```
<td align="center">CONTENU</td>
```

CONTENU

Pour aligner sur l'axe vertical, utiliser l'attribut **VALIGN**.

Les valeurs possibles sont : **TOP, MIDDLE, BOTTOM**

Exemple

```
<td align="center" valign="middle">CONTENU</td>
```

CONTENU

LES LISTES

Liste non ordonnée

Comme nous avons vu dans le chapitre balise, il est possible de créer des listes d'item en HTML.

Code

```
<ul>  
  <li>Item 1</li>  
  <li>Item 2</li>  
  <li>Item 3</li>  
  <li>Item 4</li>  
</ul>
```

- Item 1
- Item 2
- Item 3
- Item 4

Liste ordonnée

Code

```
<ol>
```

```
    <li>Item 1</li>
```

```
    <li>Item 2</li>
```

```
    <li>Item 3</li>
```

```
    <li>Item 4</li>
```

```
</ol>
```

1. Item 1

2. Item 2

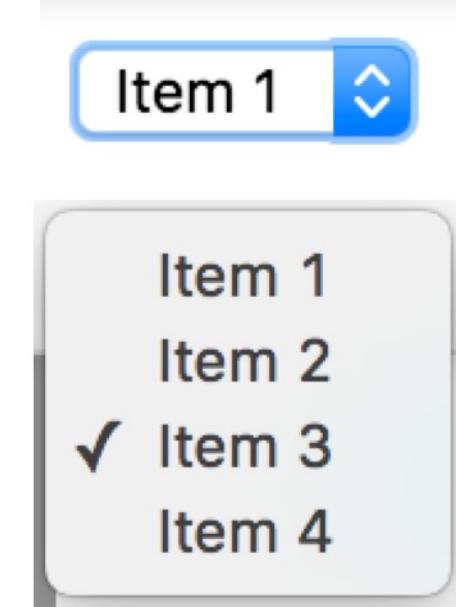
3. Item 3

4. Item 4

Listes déroulantes

Code

```
<select>  
  <option>Item 1</option>  
  <option>Item 2</option>  
  <option selected>Item 3</option>  
  <option>Item 4</option>  
</select>
```



LE MULTIMÉDIA

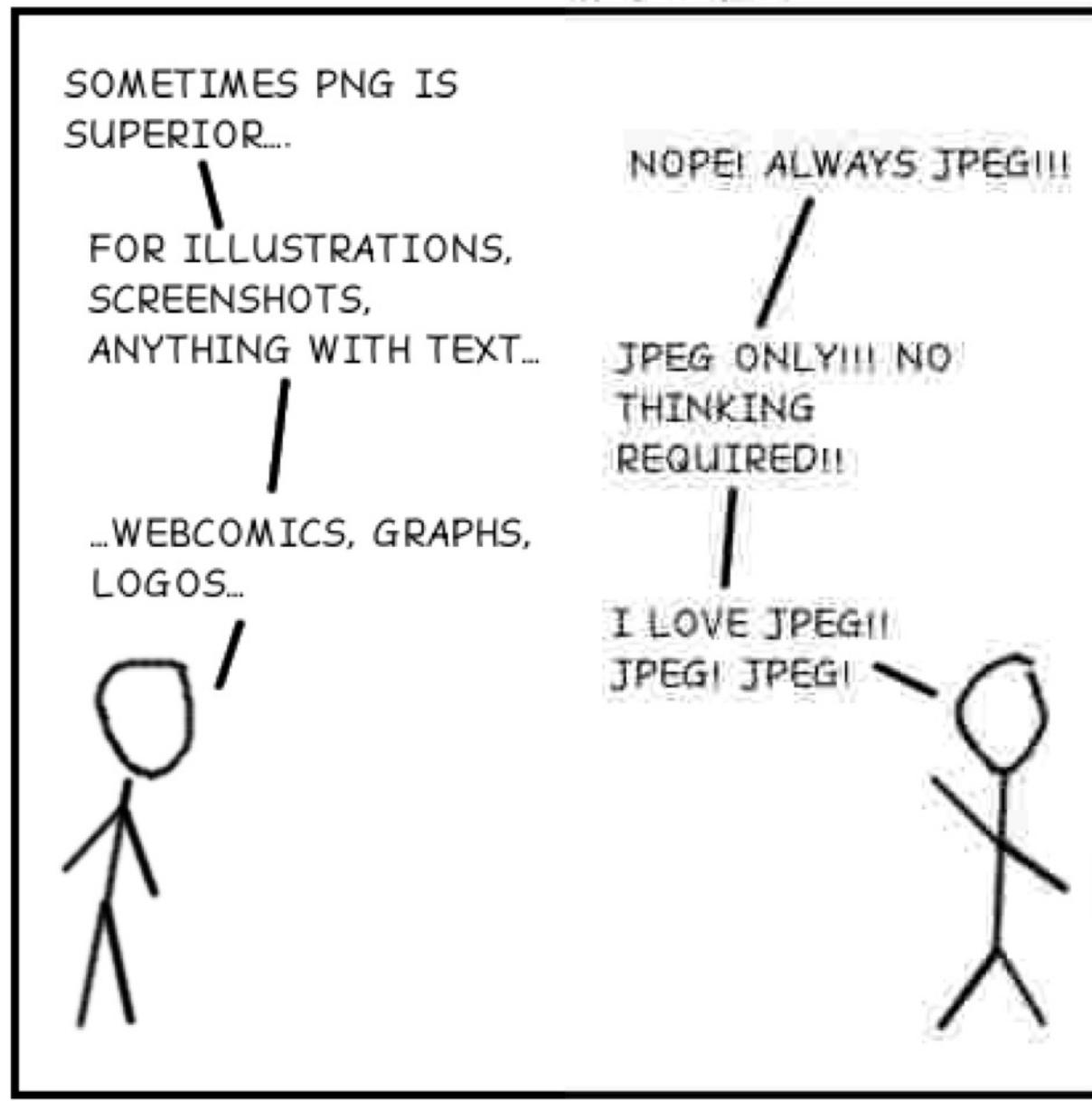
LES IMAGES

Les formats d'images : JPG et PNG

Il existe plusieurs formats sur le web, nous étudierons les plus connus : **PNG** et **JPG**.

Pour faire simple :

- **PNG** est optimisé pour des **petits formats** comme des logos,
- **JPG** est plus intéressant pour les **grosses images** comme les photos.



Les formats d'images : JPG et PNG

Pour une même qualité d'image, un fichier JPG peut être 3 fois plus léger qu'un PNG.

Pour d'autres types d'images, comme un simple dégradé le PNG sera plus léger.

De plus le PNG a tendance à mieux respecter la qualité de l'image, les JPG sont souvent « **pixélisés** » lorsqu'ils sont trop compressés.

A noter aussi que le PNG gère les **transparencies**.

Insérer l'image du même projet

```

```



Insérer l'image d'un site externe

```

```



Ajouter un titre à l'image

Pour afficher un texte au survol de l'image, il suffit d'indiquer le texte souhaiter dans l'attribut title. Il est également intéressant de remplir l'attribut alt qui permet d'afficher un texte si le navigateur n'arrive pas à charger l'image.

Code

```

```



Photo Égypte Ancienne

L'AUDIO & LA VIDÉO

HTML5 ajoute des éléments `<audio>` et `<video>`, qui permettent de jouer des sons et d'exécuter des vidéos nativement, sans plugins tels que Flash, QuickTime ou même Windows Media Player.

Les éléments `<audio>` et `<video>` se ressemblent fortement. D'ailleurs, ils sont représentés par le même objet, à savoir [HTMLMediaElement](#).

Comme ils dérivent du même objet, ils en possèdent les propriétés et méthodes.

L'insertion d'un élément <audio> est très simple :

```
<audio id="audioPlayer" src="muzic.mp3"></audio>
```

Ce bout de code suffit à insérer un lecteur audio qui lira le son "muzic.mp3"

Il est également possible d'utiliser <source> :

```
<audio id="audioPlayer">  
  <source src="muzic.ogg">  
  <source src="muzic.mp3">  
</audio>
```

Si le navigateur est capable de lire le format **.ogg**, il le fera.
Sinon, il lira le format **.mp3**.

Ça permet une plus grande interopérabilité (compatibilité entre les navigateurs et les plates-formes).

Vous pouvez compléter la balise des attributs suivants :

- **controls** : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement.
- **width** : pour modifier la largeur de l'outil de lecture audio.
- **loop** : la musique sera jouée en boucle.
- **autoplay** : la musique sera jouée dès le chargement de la page.

- **preload** : indique si la musique peut être préchargée dès le chargement de la page ou non :
 - **auto (par défaut)** : le navigateur décide s'il doit précharger toute la musique, uniquement les métadonnées ou rien du tout.
 - **metadata** : charge uniquement les métadonnées (durée, etc.).
 - **none** : pas de préchargement. Utile si vous ne voulez pas gaspiller de bande passante sur votre site.

L'apparence du lecteur audio change en fonction du navigateur.

La figure suivante représente par exemple le lecteur audio dans Google Chrome.



Il suffit d'une simple balise <video> pour insérer une vidéo dans la page :

```
<video src="ma_video.webm" controls  
poster="sintel.jpg" width="600"></video>
```

Si le navigateur est capable de lire le format **.ogv**, il le fera.
Sinon, il lira le format **.mp4**.

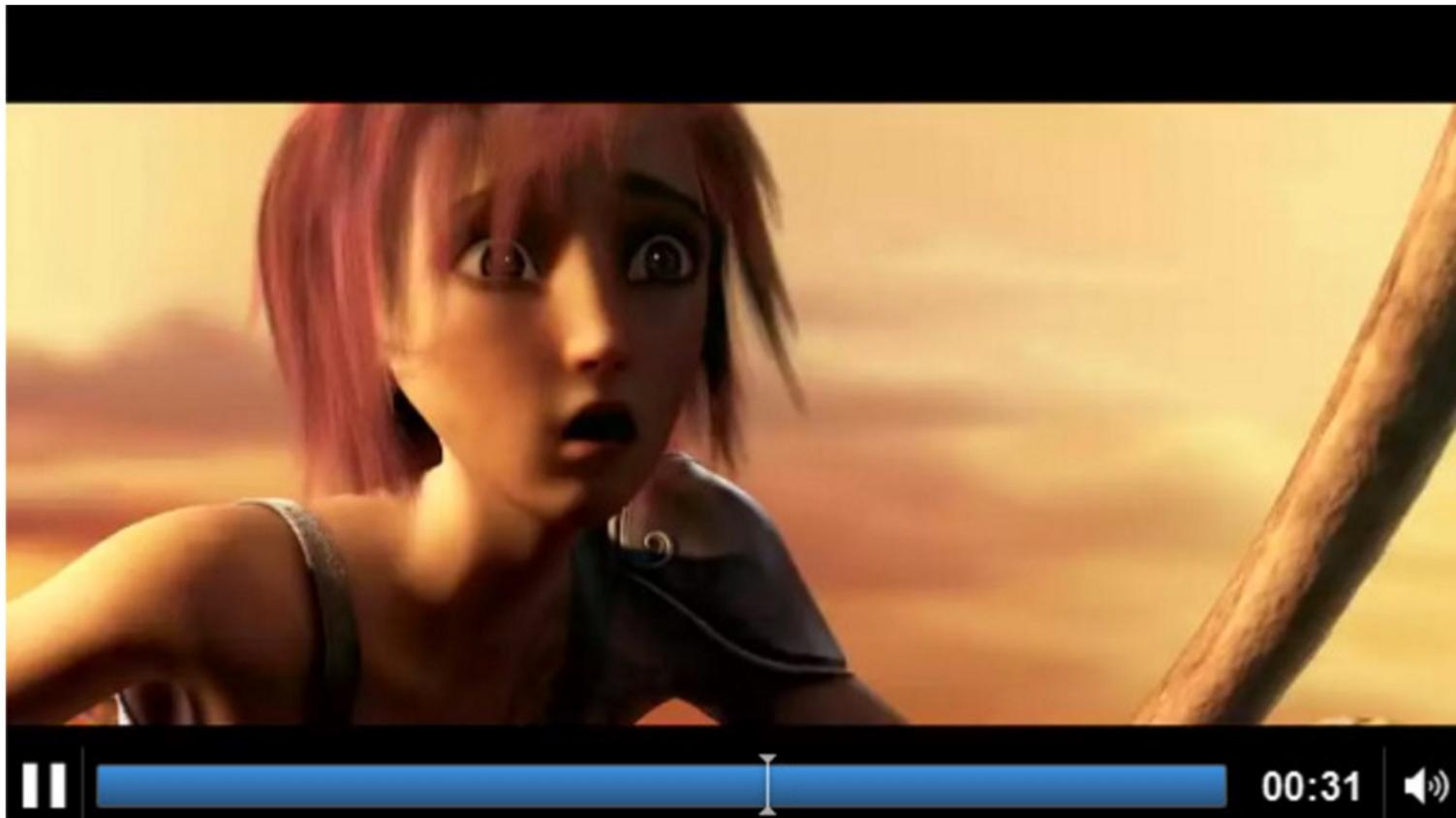
Ça permet une plus grande interopérabilité (compatibilité entre les navigateurs et les plates-formes).

Rajoutons quelques attributs (la plupart sont les mêmes que pour la balise `<audio>`) :

- **poster** : image à afficher à la place de la vidéo tant que celle-ci n'est pas lancée.
- **controls** : pour ajouter les boutons « Lecture », « Pause » et la barre de défilement.
- **width** : pour modifier la largeur de la vidéo.
- **height** : pour modifier la hauteur de la vidéo.
- **loop** : la vidéo sera jouée en boucle.

- **autoplay** : la vidéo sera jouée dès le chargement de la page.
- **preload** : indique si la vidéo peut être préchargée dès le chargement de la page ou non. Cet attribut peut prendre les valeurs :
 - **auto (par défaut)** : le navigateur décide s'il doit précharger toute la vidéo, uniquement les métadonnées ou rien du tout.
 - **metadata** : charge uniquement les métadonnées (durée, dimensions, etc.).
 - **none** : pas de préchargement.

Résultat



LES LIENS

Les liens hypertexte

La gestion des liens hypertexte se fait avec la balise `<a>` et l'attribut `href`.

Cette balise est peut être la plus importante du web, sans elle impossible de passer de site en site.

Le lien de base aura cette forme :

```
<a href="http://www.egypte-antique.com">EGYPTE ANCIENNE</a>
```

L'exemple ci-dessus affichera :

EGYPTE ANCIENNE

Les liens hypertexte

On remarquera que lorsqu'on clique dessus, on quitte le site pour aller sur un autre.

Pour éviter de perdre la page en cours on peut donner à la balise `<a>` l'attribut target qui indiquera la cible de destination.

Le lien de base aura cette forme :

```
<a href="xxx" target="_blank">YYY</a>
```

`_blank` ouvrira une nouvelle fenêtre du navigateur.

```
<a href="xxx" target="_top">YYY</a>
```

`_top` ouvrira le contenu dans sa totalité sur la page.

```
<a href="xxx" target="_parent">YYY</a>
```

`_parent` affichera le contenu dans le cadre qui enveloppe la structure courante.

```
<a href="xxx" target="_self">YYY</a>
```

`_self` affichera le contenu dans le cadre courant, c'est la valeur par défaut.

Les ancrages

La gestion des ancrages se fait avec l'attribut name.

Le fait de nommer cette balise la transforme en ancre.

Il s'agit d'un point de repère qu'on peut atteindre via l'URL.

L'ancre se présente ainsi :

```
<a name="nom-de-votre-ancre"></a>
```

Le pointeur qui permet d'atteindre l'ancre se créer ainsi :

```
<a href="#nom-de-votre-ancre"></a>
```

Lorsque l'on clique sur le pointeur, le navigateur scroll jusqu'à l'ancre.

Créer un lien sur une image

Pour créer un lien sur une image, il faut encadré l'image des balises `<a>`

Exemple :

```
<a href="http://www.egypte-antique.com" target="_blank">  
    
</a>
```

LES INPUTS

Le champ texte

HTML possède des éléments de formulaire qui permettent aux utilisateurs d'interagir avec le site.

Input

`<input />`

Résultat

Cliquez sur l'élément et vous pourrez entrer du texte.

Par défaut l'input est un champ texte mais on peut lui donner d'autres formes telles que des boutons radio, des checkbox, des listes, etc.

Les libellés

Cette zone de texte permet d'indiquer une information.

```
<label>Libellé</label>
```

Généralement, le libellé est lié à une zone de texte.

Pour lier le label au champ, il faut lui donner un attribut for qui a la même valeur que l'id du champ :

```
<form method="post"
action="traitement.php">

    <p>
        <label
            for="pseudo">Votre
            pseudo</label> : <input
            type="text" name="pseudo"
            id="pseudo" />
    </p>
</form>
```

Quelques attributs supplémentaires

On peut ajouter un certain nombre d'autres attributs à la balise `<input />` pour personnaliser son fonctionnement :

- On peut agrandir le champ avec **size**.
- On peut limiter le nombre de caractères que l'on peut saisir avec **maxlength**.
- On peut pré-remplir le champ avec une valeur par défaut à l'aide de **value**.
- On peut donner une indication sur le contenu du champ avec **placeholder**.

Quelques attributs supplémentaires

Exemple

```
<form method="post" action="/traitement">  
    <p>  
        <label for="pseudo">Votre pseudo :</label>  
        <input type="text" name="pseudo" id="pseudo"  
               placeholder="Ex : Bobby" size="50" maxlength="15" />  
    </p>  
</form>
```

Mot de passe

Vous pouvez facilement faire en sorte que la zone de texte se comporte comme une « zone de mot de passe », c'est-à-dire une zone où on ne voit pas à l'écran les caractères saisis.

Pour créer ce type de zone de saisie, utilisez l'attribut `type="password"`.

Exemple

```
<form method="post" action="/traitement">  
    <p>  
        <label for="pseudo">Votre pseudo :</label>  
        <input type="text" name="pseudo" id="pseudo" />  
        <br />  
        <label for="pass">Votre mot de passe :</label>  
        <input type="password" name="pass" id="pass" />  
    </p>  
</form>
```

Votre pseudo : famisse
Votre mot de passe :

Checkbox

Exemple de checkbox :

```
<input type="checkbox" name="vehicle1" value="Bike">J'ai  
un vélo
```

```
<input type="checkbox" name="vehicle2" value="Car">J'ai  
une voiture
```

Résultat

J'ai un vélo J'ai une voiture

Ce qui nous frappe lors de ce test de checkbox c'est qu'on peut cocher les deux valeurs.

C'est assez idiot de répondre à une question par oui et par non en même temps.

Il existe donc un input qui permet de choisir qu'une seule réponse possible.

Radio

Exemple de radio :

```
<input checked name="question1" type="radio" /> OUI
```

```
<input name="question1" type="radio" /> NON
```

Résultat

OUI NON

Textarea

Il existe un type d'input qui permet d'écrire du texte sur plusieurs lignes

```
<textarea>Exemple</textarea>
```

Résultat



Bien qu'il soit conseillé de changer sa taille en CSS, on peut modifier sa taille en lui précisant son nom de colonnes et de lignes.

```
<textarea cols="30" rows="4">Exemple</textarea>
```

Résultat



Liste déroulante

Lorsque l'on propose à l'utilisateur de sélectionner une valeur stricte, on lui propose une liste déroulante.

<select>

<option>OPTION 1</option>

<option>OPTION 2</option>

</select>

Résultat



OPTION 1

✓ OPTION 1
OPTION 2

Email

Vous pouvez demander à saisir une adresse e-mail :

```
<input type="email" />
```

Le champ vous semblera a priori identique mais votre navigateur sait désormais que l'utilisateur doit saisir une adresse e-mail.

Il peut afficher une indication si l'adresse n'est pas un e-mail :

A screenshot of a web form. At the top, there is a text input field containing the text "toto" and a "Valider" button to its right. A tooltip or validation message box is displayed below the input field, containing an exclamation mark icon and the text: "Veuillez inclure "@" dans l'adresse e-mail. Il manque un symbole "@" dans "toto".

Email

Certains navigateurs, comme les navigateurs mobiles sur iPhone et Android, affichent un clavier adapté à la saisie d'email :



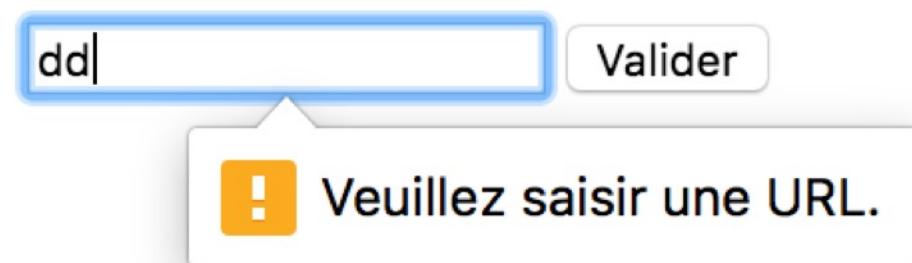
Clavier de saisie d'e-mail sur un iPhone

URL

Avec le type url, on peut demander à saisir une adresse absolue (commençant généralement par http://) :

```
<input type="url" />
```

Même principe : si le champ ne vous semble pas différent sur votre ordinateur, sachez que celui-ci comprend bel et bien que le visiteur est censé saisir une adresse.



URL

Les navigateurs mobiles affichent par exemple un clavier adapté à la saisie d'URL :



Nombre

`<input type="tel" />`

Sur iPhone, par exemple, un clavier adapté s'affiche lorsqu'on doit remplir le champ :



Clavier de saisie de numéro de téléphone sur un iPhone

N° de téléphone

```
<input type="number" />
```

Sur iPhone, un clavier adapté s'affiche lorsqu'on doit remplir le champ :



Vous pouvez personnaliser le fonctionnement du champ avec les attributs suivants :

- **min** : valeur minimale autorisée,
- **max** : valeur maximale autorisée,
- **step** : c'est le « pas » de déplacement. Si vous indiquez un pas de 2, le champ n'acceptera que des valeurs de 2 en 2 (par exemple 0, 2, 4, 6...).

Curseur

```
<input type="range" />
```

Le type range permet de sélectionner un nombre avec un curseur (aussi appelé slider) :



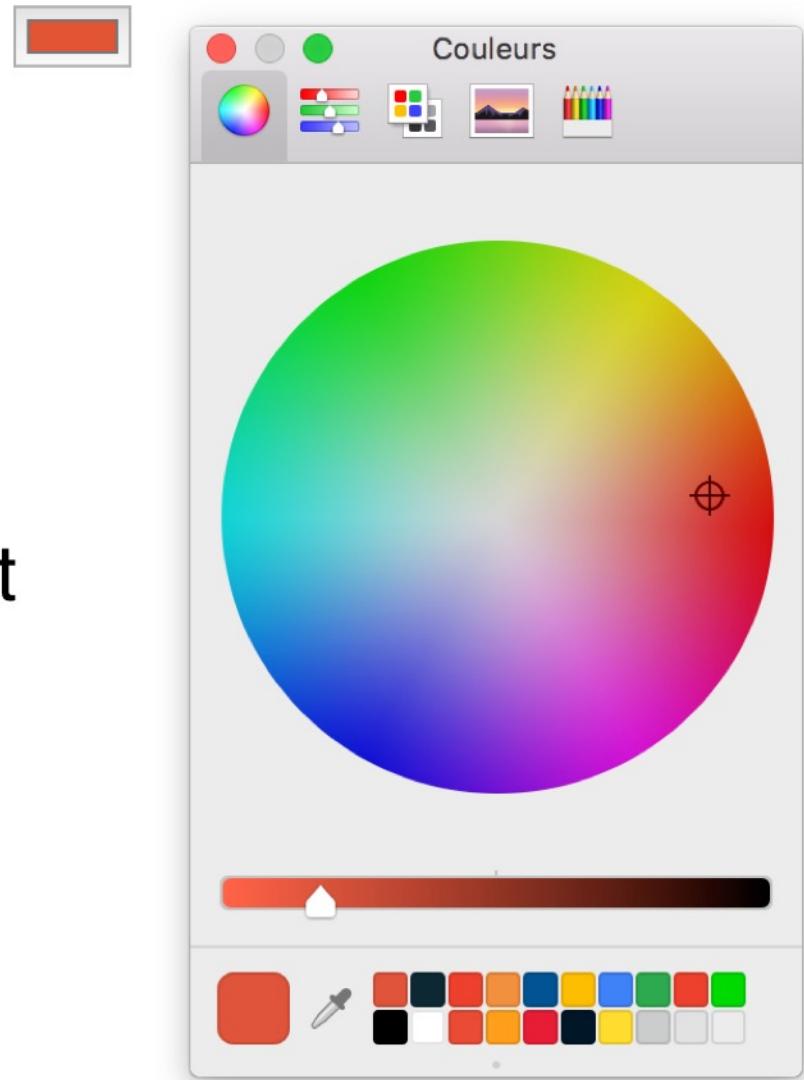
Vous pouvez utiliser les attributs **min (0)**, **max (100)**, **step** et **value** pour restreindre les valeurs disponibles.

Couleur

```
<input type="color" />
```

Le type « color » permet de sélectionner une couleur.

Attention tous les navigateurs ne connaissent pas encore ce type de champ, mais la compatibilité progresse.

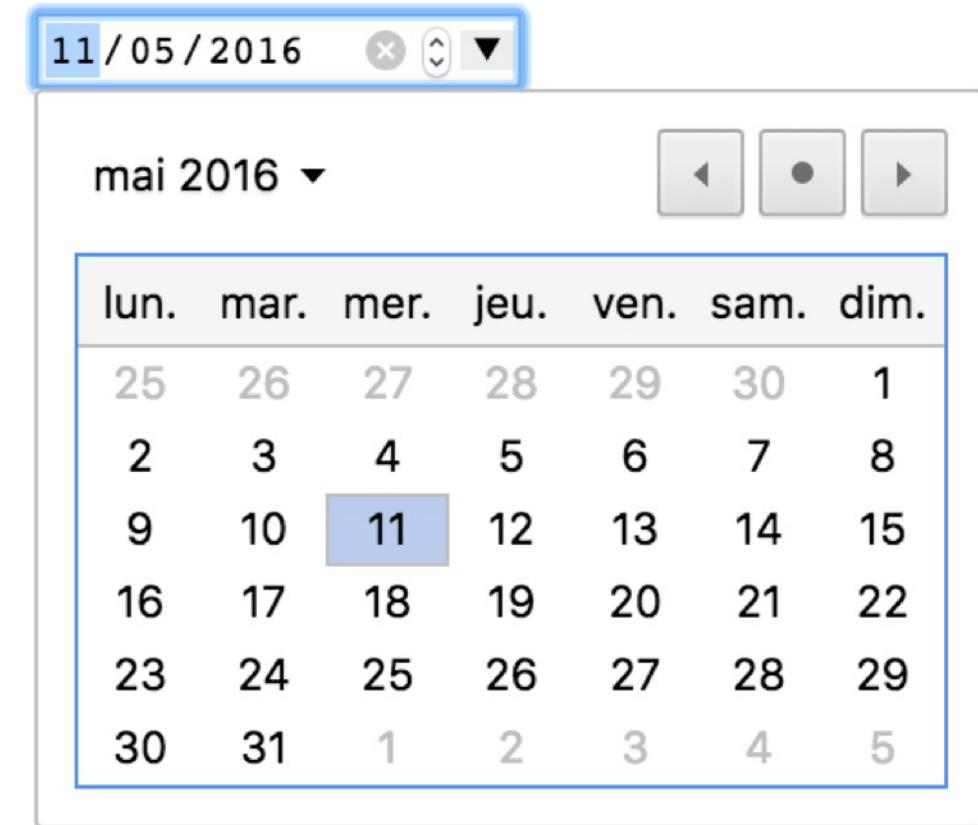


Date

```
<input type="date" />
```

Différents types de champs de sélection de date existent :

- **date** : pour la date (05/08/1985 par exemple) ;
- **time** : pour l'heure (13:37 par exemple) ;
- **week** : pour la semaine ;
- **month** : pour le mois ;
- **datetime** : pour la date et l'heure (avec gestion du décalage horaire) ;
- **datetime-local** pour la date et l'heure (sans gestion du décalage horaire).



Recherche

```
<input type="search" />
```



Le navigateur décide comment afficher le champ de recherche.

Ainsi, il peut ajouter une petite loupe au champ pour signifier que c'est un champ de recherche et éventuellement mémoriser les dernières recherches effectuées.

Fichier

```
<input type="file" />
```

Le champs permet de choisir un fichier sur votre ordinateur et l'envoie au serveur.

Aucun fichier choisi

Sélection automatique d'un champ

Vous pouvez placer automatiquement le curseur dans l'un des champs de votre formulaire avec l'attribut autofocus.

Dès que la page est chargée, le curseur se placera dans ce champ.

Par exemple, pour que le curseur soit par défaut dans le champ prenom :

```
Prénom : <input type="text"  
name="prenom" id="prenom" autofocus  
><br />
```

```
Nom : <input type="text" name="nom"  
id="nom" >
```

Prénom : |

Nom : |

Rendre un champ obligatoire

Vous pouvez faire en sorte qu'un champ soit obligatoire en lui donnant l'attribut required.

```
<input type="text" name="prenom" id="prenom" required>  
<input type="text" name="nom" id="nom" >
```

Prénom :

Nom :

 Veuillez renseigner ce champ.

Les boutons

Il existe 4 types de bouton :

- **type="submit"** : le principal bouton d'envoi de formulaire.
- **type="reset"** : remise à zéro du formulaire.
- **type="image"** : équivalent du bouton submit, présenté cette fois sous forme d'image. Rajoutez l'attribut src pour indiquer l'URL de l'image.
- **type="button"** : bouton générique, qui n'aura (par défaut) aucun effet. Est géré en JavaScript pour exécuter des actions.

Les boutons

```
<input type="submit" value="Envoyer" >
```

Prénom :

Nom :

LES FORMULAIRES

L'utilité d'un formulaire HTML

Le formulaire HTML est un ensemble d'éléments que l'utilisateur soumet au serveur.

Ces éléments sont essentiellement les inputs (voir le chapitre précédent).

Cela permet une communication entre l'utilisateur et le site web.

La balise <form>

Les formulaires sont encadrés par les balises <form> et possèdent ces attributs :

- **METHOD**, indique le type d'envoi des données. Soit en POST (données envoyées via le corps de la requête) ou en GET qui envoie les données via l'URL (255 car. max.).
- **ACTION** indique la localisation du script. Cela peut être un script ou directement une adresse mail, exemple: <mailto:exemple@example.com>

L'utilité d'un formulaire HTML

Exemple

```
<form method="POST" action="/page-html-formulaire">  
Nom <input type="text" name="nom" />  
Prénom <input type="text" name="prenom" />  
<input type="submit" value="Valider" />  
</form>
```

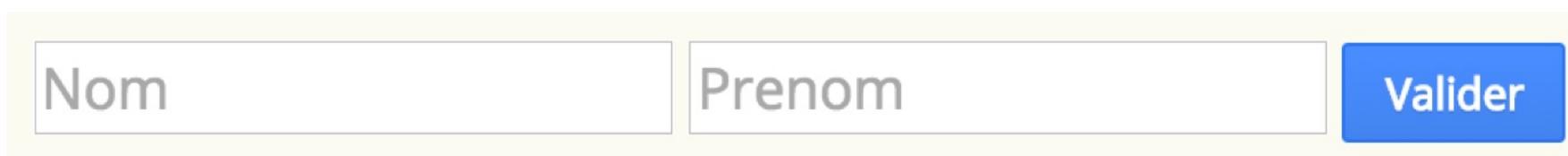
Résultat

Nom Prénom Valider

L'utilité d'un formulaire HTML

Exemple avec Bootstrap (que nous verrons plus tard)

```
<form method="POST" action="/page-html-formulaire">  
    <input type="text" name="nom" placeholder="Nom" />  
    <input type="text" name="prenom"  
placeholder="Prenom" />  
    <input type="submit" class="btn" value="Valider" />  
</form>
```



The form is displayed on a light yellow background. It contains two text input fields side-by-side, each with a placeholder label above it. To the right of the inputs is a blue rectangular button with the word "Valider" in white capital letters.

L'ordre du focus dans les formulaires

```
<input type="text" placeholder="Prénom" tabindex="1" />
<input type="text" placeholder="Nom" tabindex="4" />
<input type="text" placeholder="Email" tabindex="3" />
<button tabindex="2">Envoyer</button>
```
