Introduction à la programmation web

IT103 Programmation web

Philippe SWARTVAGHER
ph-sw.fr

BORDEAUX INP Enseirb-Matmeca

À propos de ce cours

Organisation

- 1 séance de CM (1h20)
- 7 séances de CI ($7 \times 2 \times$ 1h20)
- 4 séances de projet ($4 \times 2 \times$ 1h20)

Objectifs

- Connaître le fonctionnement de base d'un site Internet
- Avoir les notions de base (seulement) pour réaliser un site Internet
  - Listes de fonctionnalités non exhaustives ! RTFM !

Évaluation

- Projet : code, rapport et soutenance
- Partiel de 30 min

Pré-requis

- Avoir suivi les cours de IF110

Ressources et crédits

Ce cours est basé, entre autres, sur :

- Le cours de Jean-Rémy Falleri : https://www.labri.fr/perso/falleri/teaching/it103/

Un petit sondage

- Qui sait comment fonctionne un site web ?

Un petit sondage

- Qui sait comment fonctionne un site web ?
- Qui a déjà fait un site web ?

Un petit sondage

- Qui sait comment fonctionne un site web ?
- Qui a déjà fait un site web ?
- Qui connaît HTML et CSS ?

Un petit sondage

- Qui sait comment fonctionne un site web ?
- Qui a déjà fait un site web ?
- Qui connaît HTML et CSS ?
- Qui connaît PHP ?

Un petit sondage

- Qui sait comment fonctionne un site web ?
- Qui a déjà fait un site web ?
- Qui connaît HTML et CSS ?
- Qui connaît PHP ?
- Qui connaît SQL ?

Un petit sondage

- Qui sait comment fonctionne un site web ?
- Qui a déjà fait un site web ?
- Qui connaît HTML et CSS ?
- Qui connaît PHP ?
- Qui connaît SQL ?
- Qui connaît JavaScript ?

Sommaire

Fonctionnement d'un site Internet

L'architecture client/serveur

Client

- Demande des ressources (fichiers, contenu, ...) à un serveur
  - La demande est appellée *requête*
- **Exemple** : un navigateur web, n'importe quel outil capable d'interroger un serveur (commandes `nc`, `telnet`, `curl`, `wget`, ...)

Serveur

- Peut traiter et répondre à des requêtes venant de clients
  - La réponse est appelée... *réponse*
- Peut servir plusieurs clients simultanément
- Disponible tout le temps, pour n'importe quel client (ou pas)
- Ordinateur (presque) classique branché à un réseau

- Le client et le serveur sont souvent des machines distinctes
  - Communications réseau

Navigateurs web

- Permet d'interagir avec un serveur web : envoyer des requêtes et recevoir des réponses
  - Fonctionnalités pas spécifiques à un navigateur web
- Programme qui permet d'afficher des pages web
  - Comme un visionneur d'image permet d'afficher des images
- Programme qui permet d'exécuter du code (JavaScript)

Exemples

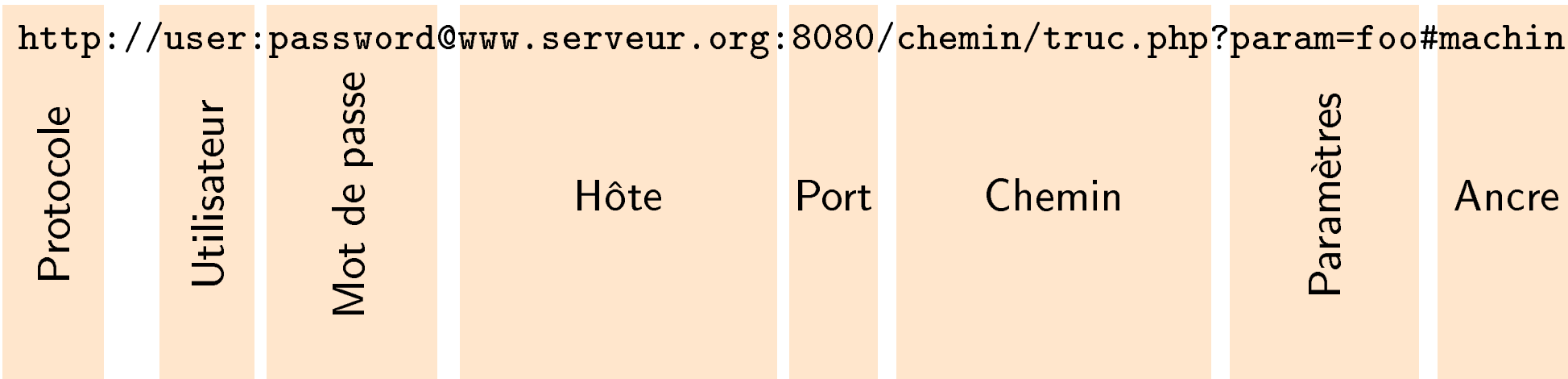
- Firefox
- Chrome
- Edge
- Opera
- Brave
- Mais aussi dans le terminal : Lynx

Obtenir une page web

Que se passe-t-il entre la saisie d'une adresse web et l'affichage de la page web correspondante ?

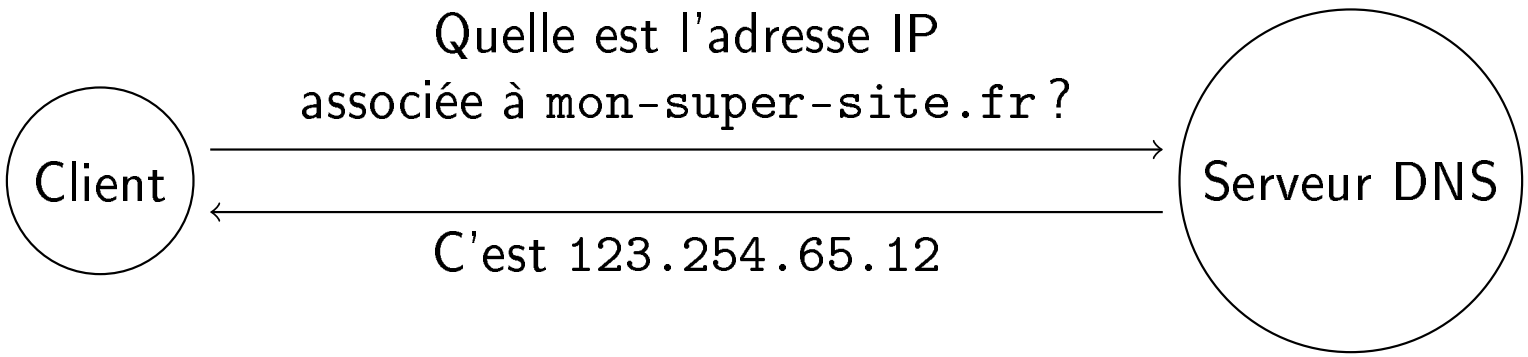
L'adresse web

URL : *Uniform Resource Locator*



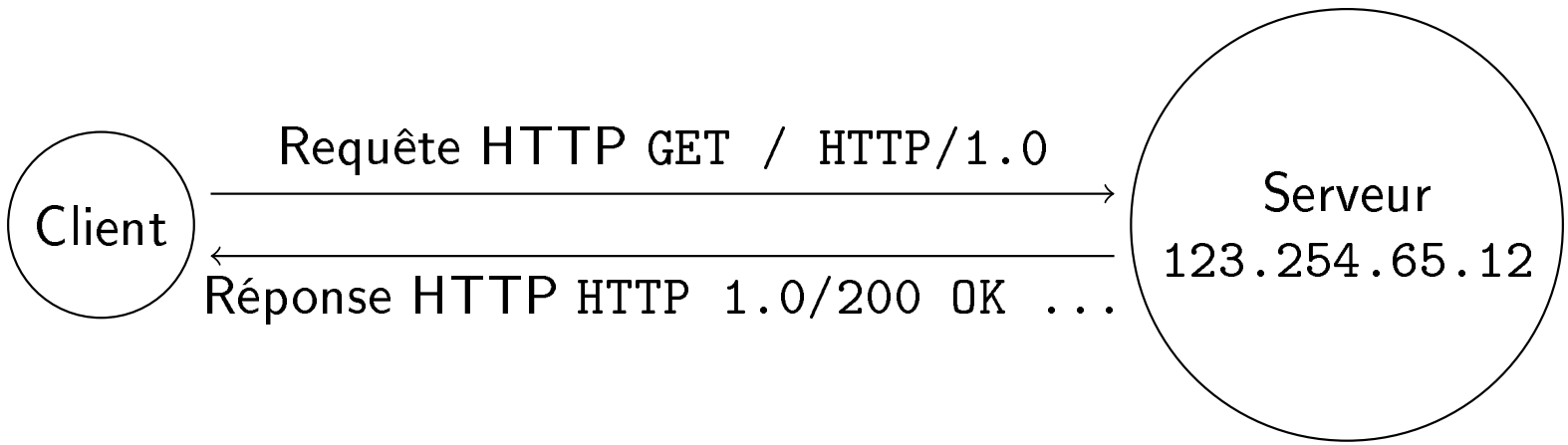
Obtenir une page web

Étape 1 : résolution de l'adresse IP de l'hôte en demandant à un serveur DNS (*Domain Name System*)



Obtenir une page web

Étape 2 : envoi au serveur d'une requête HTTP (*HyperText Transfer Protocol*)



Le protocole HTTP

- Protocole qui permet d'interagir avec un serveur web
- Le serveur écoute en TCP sur le port 80
- Protocole non sécurisé : quiconque intercepte le trafic peut en lire le contenu !

Principaux types de requêtes :

- `GET` : demande une ressource
- `POST` : envoie des données
- `DELETE` : supprime une ressource

Quelques réponses possibles :

- `200 OK`
- `404 Not Found`
- `500 Internal Server Error`

HTTPS

- *HyperText Transfer Protocol* ***Secure***
- Version sécurisée du protocole HTTP
- Échanges HTTP chiffrés avec le protocole TLS
- Le serveur écoute en TCP sur le port 443

Réponse HTTP du serveur

```
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/3.11.2
Date: Tue, 06 Feb 2024 17:28:38 GMT
Content-type: text/html
Content-Length: 330
Last-Modified: Tue, 06 Feb 2024 16:17:07 GMT

<!DOCTYPE html>
<html>
    <head>
        <title>Mon Super Site</title>
        <meta charset="utf-8">
    </head>
    <body>
        <div>
            <h1>Mon super site</h1>
            <p>Ceci est un super site.</p>
        </div>
    </body>
</html>
```

1. En-têtes de la réponse
2. Contenu de la réponse : ressource demandée
   - Code HTML dans ce cas

Deux types de sites web

Site statique

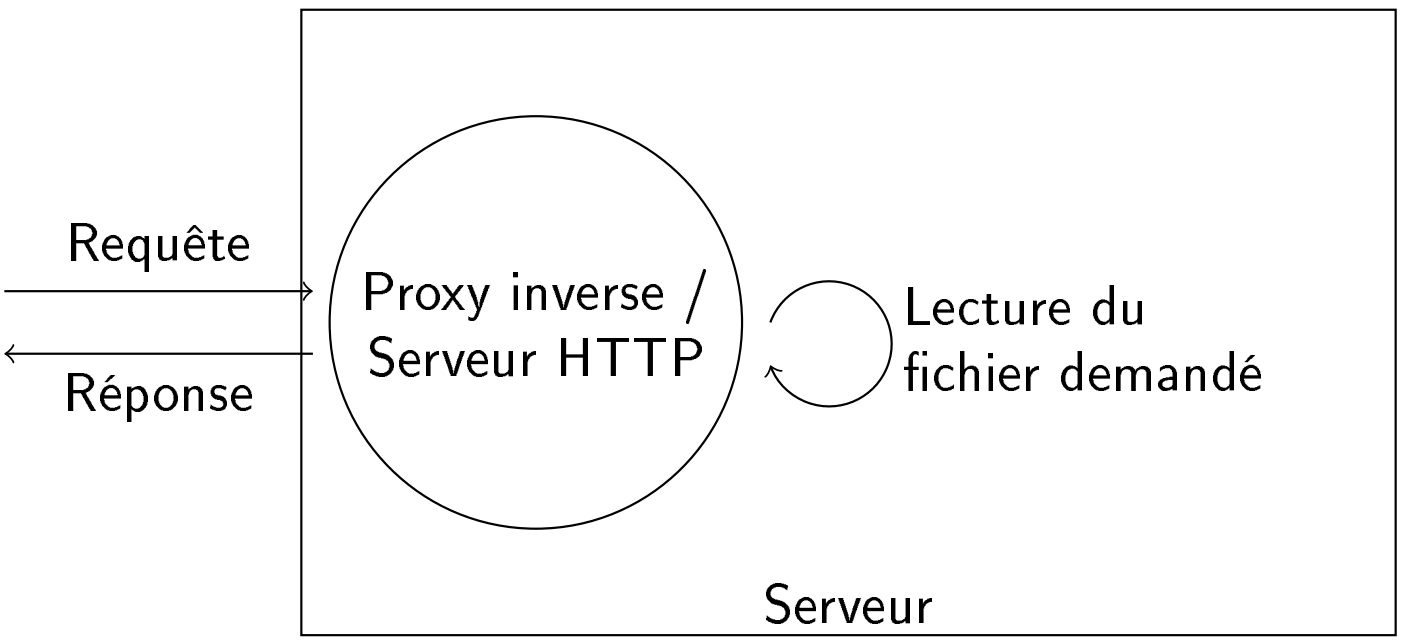
- Le serveur renvoie le contenu de fichiers déjà existant, qu'il ne génère pas.
- Fichiers HTML, CSS, JavaScript, images, ...

Site dynamique

- Le serveur passe la requête à un programme qui va générer le contenu de la réponse.
  - Interaction avec une base de données
  - Interaction avec un autre service
- Code HTML, fichiers CSS, JavaScript, images, ...

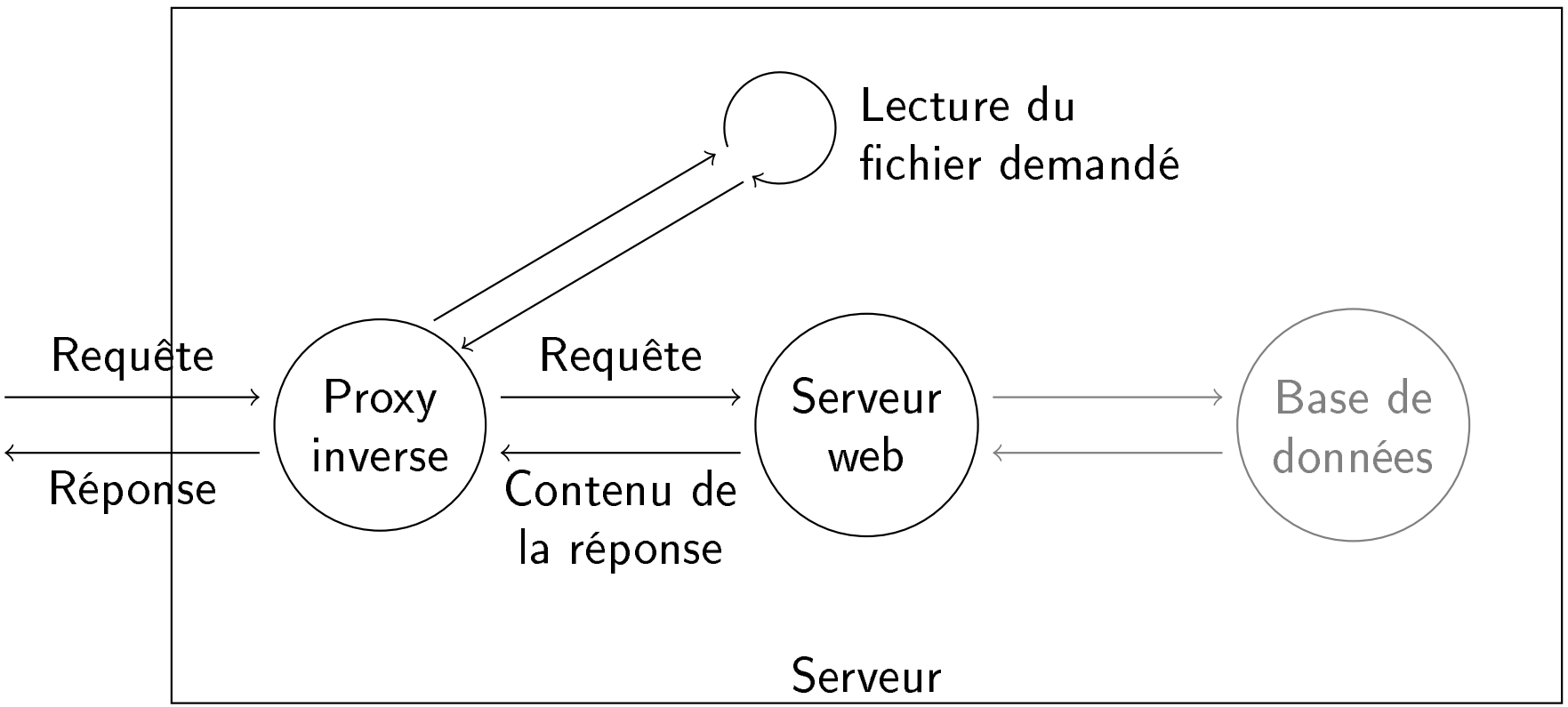
Rôle du serveur pour un site statique

- Renvoyer les fichiers demandés



Rôle du serveur pour un site dynamique

- Renvoyer les fichiers demandés
- *ou* appel à un autre programme pour **générer** les réponses
  - Exécution de code pour traiter la requête et générer la réponse correspondante



Logiciels

Proxy inverse / Serveur HTTP

- Apache
- Nginx
- lighthttpd
- HAProxy
- Traefik
- ...

Serveur web dynamique

- PHP
- Gunicorn (Python)
- Tomcat (Java)
- Puma (Ruby)
- ...

La suite de ce cours

- **HTML** : le langage qui permet de décrire ce que le navigateur va afficher
- **CSS** : le langage qui permet de mettre en forme ce qui est affiché
- **PHP** : langage de serveur web dynamique qui permet de générer des pages
- **SQL** : langage qui permet d'interagir avec des bases de données
- **JavaScript** : langage qui permet d'exécuter du code dans le navigateur, pour interagir avec le contenu affiché

Le seul cours où vous allez apprendre 5 langages d'un coup !

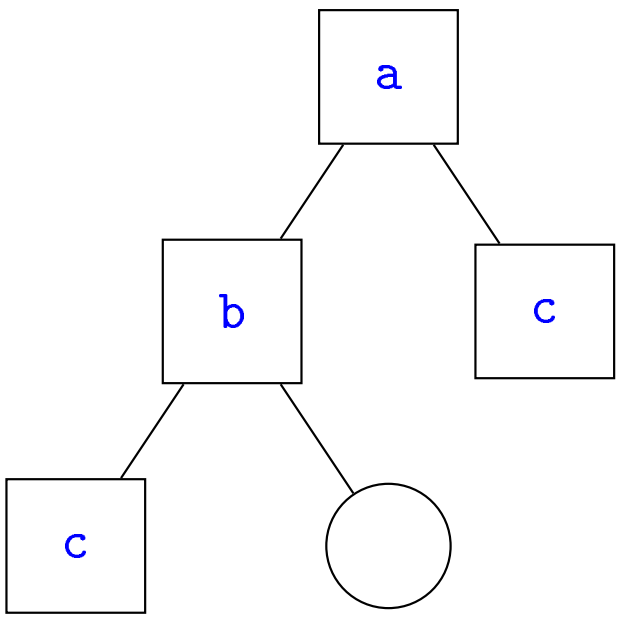
Structuration du contenu des pages web avec HTML

Le langage HTML

- Langage de description
- Inventé en 1989
- Code stocké dans des fichiers avec l'extension `.html`
- Actuellement version 5 du langage

Détour par un autre langage : XML

- *eXtensible Markup Langage*
- Code stocké dans des fichiers avec l'extension `.xml`
- Utilisé par exemple comme fichiers de configuration, pour représenter des données structurées, pour les flux RSS, ...
- Organise les données sous forme d'arbre




```
<a foo="bar" baz="oof">
    <b>
        <c></c>
        A free text!
    </b>
    <c piz="za">
</a>
```

Vocabulaire

```
<a foo="bar">Truc</a>
```

- `<a>` : balise ouvrante `a`
- `</a>` : balise fermante `a`
- `foo="bar"` : attribut `foo` ayant pour valeur `bar`
- `Truc` : contenu de la balise

Règles d'écriture

- Toute balise ouvrante doit être fermée
  - En HTML, certaines balises (*void elements* : `<br>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`, ...) n'ont pas besoin d'être fermées [1]
  - Ancienne syntaxe parfois rencontrée : `<br />`
- Les balises imbriquées doivent être fermées dans l'ordre inverse qu'elles ont été ouvertes
  - ✗ `<a><b>foo</a></b>`
  - ✓ `<a><b>foo</b></a>`
- Style de code : mettre le contenu d'une balise sur une nouvelle ligne et indenter suivant la profondeur dans le graphe

1. Un peu de lecture : https://jakearchibald.com/2023/against-self-closing-tags-in-html/

Commentaires et caractères particuliers

Commentaires

```
<!-- Ceci est un commentaire -->
```

Caractères particuliers

Comment afficher < ou > ?
⇒ utilisation des *HTML entities* :

- `<` pour <
- `>` pour >
- `&` pour &
- ` ` pour une espace insécable
- Il y en a d'autres...

XML et HTML

L'HTML est du XML !

HTML 5

On fait de l'HTML **5** :

- Beaucoup de nouvelles choses par rapport aux versions précédentes
- Beaucoup de choses dépréciées également !

Attention aux différentes ressources que vous pouvez trouver sur Internet !

Code de base d'une page HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titre de la page</title>
        <meta charset="utf-8">
    </head>
    <body>
    </body>
</html>
```

- `<!DOCTYPE html>` : c'est de l'HTML 5
- `<html></html>` : tout se passe là
- `<head></head>` : informations sur la page
  - Titre de la page (ce qui sera affiché comme titre de l'onglet et/ou de la fenêtre du navigateur)
  - Encodage de la page
- `<body></body>` : ce qui sera affiché

Deux types d'éléments

Éléments **blocs**

- Peuvent contenir d'autres éléments (bloc ou en ligne)
- Occupent (par défaut) toute la largeur possible
- Deux blocs consécutifs seront présentés (par défaut) verticalement (le deuxième bloc va à la ligne)
- Exemples : un paragraphe (<p>), une image (<img>), ...

Éléments **en ligne** (*inline*)

- Peuvent contenir d'autres éléments en ligne uniquement
- Impossible d'imposer des dimensions
- Deux blocs consécutifs seront présentés horizontalement (pas de nouvelle ligne)
- Exemples : un lien (<a>), du texte mis en évidence (), ...

Attention !

Longues listes de balises dans les prochaines diapositives...

Jetez un œil à la référence exhaustive :
https://developer.mozilla.org/fr/docs/Web/HTML/Element

Structurer le contenu de la page - balises bloc

Balise	Description
`<header>`	En-tête de la page
`<footer>`	Pied de page
`<section>`	Section générique, commençant généralement par un titre
`<article>`	Section indépendante, qui peut être séparée du reste du contenu
`<aside>`	*Aparté* : contentu au rapport indirect avec le contenu principal
`<nav>`	Navigation : lien vers d'autres pages, un menu, ...
`<div>`	Balise bloc générique

Contenus textuels - balises bloc

Balise	Description
`<p>`	Paragraphe de texte
`<h1>`, `<h2>`, ..., `<h6>`	Titres (1 : très important ; 6 : peu important)
`<blockquote>`	Longue citation
`<pre>`	Code HTML à l'intérieur pas interprété, présenté dans un style « code »

Autres contenus en balises bloc

Balise	Description
`<table>`	Tableau
`<ul>`	Liste sans ordre particulier (liste à puces)
`<ol>`	Liste numérotée
`<address>`	Adresse de contact

Balises en ligne pour le texte

Balise	Description
<abbr>	Abbréviation ou acronyme
<b>	Élément qui doit attirer l'attention du lecteur mais qui n'est pas important
 	Nouvelle ligne
<code>	Code
	Emphase : élément sur lequel on souhaite insister
<i>	Contenu qui se différencie du texte principal
<kbd>	Touche de clavier
<mark>	Marque du texte avec une pertinence particulière
<samp>	Sortie textuelle d'un programme
<small>	Petit texte
	Texte particulièrement important
<sub>	Texte en indice
<sup>	Texte en exposant
	Balise en ligne générique

Liens

La balise en ligne <a> permet de faire un lien :

```
<a href="https://ph-sw.fr">Le meilleur prof</a>
```

La cible du lien est à indiquer dans l'attribut `href`

Sur la page accédée à l'adresse https://foo.com/a/b/truc.html :

Contenu de `href`	**Adresse générée par le navigateur**
https://foo.com/a/b/page.html	https://foo.com/a/b/page.html
//bar.com/a/b/page.html	https://bar.com/a/b/page.html
/bar.html	https://foo.com/bar.html
bar.html ou ./bar.html	https://foo.com/a/b/bar.html
../bar.html	https://foo.com/a/bar.html

Ancres

Chaque balise peut avoir un attribut `id` :

```
<h2 id="conclusion">Conclusion : vive le web !</h2>
```

Attention ! Il ne peut pas y avoir deux attributs `id` avec la même valeur au sein d'une même page

Lien pour aller directement à cet élément :

```
<a href="#conclusion">Aller à la conclusion</a>

<a href="foo.html#conclusion">
    la conclusion d'un autre article
</a>
```

Images

Pour afficher une image :

```
<img src="panda.png" alt="Panda mangeant du bambou">
```

- `src` : adresse de l'image (comme `href` pour `<a>`)
- `alt` : texte de description affiché en cas d'échec du chargement de l'image (**obligatoire !**)

Image avec légende :

```
<figure>
    <img src="panda.png" alt="Panda mangeant du bambou
        ">
    <figcaption>
        Les pandas aiment manger du bambou.
    </figcaption>
</figure>
```

Toutes les images à afficher seront téléchargées par le client : optimisez leur taille !

Formulaires

```
<form method="post" action="connexion.php">
    <!-- champs du formulaire -->
</form>
```

- `method` : quel type de requête HTTP utiliser pour envoyer le formulaire
  - `GET` : les données seront envoyées comme paramètres de l'URL (exemple : moteur de recherche `search.php?q=web`)
  - `POST` : les données seront envoyées comme données de la requête HTTP
- `action` : vers quelle URL envoyer les données

Pour le traitement des données envoyées : *cf* partie sur PHP

Champs de formulaires : zones de texte

```
<input type="text" name="pseudo">
```

- `type` : le type de champ. Influe sur la façon dont le champ est présenté, le clavier présenté (sur téléphone ou tablette) et peut faire une première vérification des contraintes
  - `text` : simple texte
  - `password` : mot de passe
  - `email` : adresse mail
  - `url` : URL
  - `tel` : numéro de téléphone
  - `number` : nombre
  - `range` : nombre (mais affiche un curseur à déplacer)
  - `color` : couleur
  - `date` : date
  - `search` : champ de recherche
- `name` : le nom qui permettra de récupérer la valeur de ce champ lors du traitement des données
- D'autres attributs existent

Champs de formulaires : grandes zones de texte

```
<textarea name="comment" rows="10" cols="50">
</textarea>
```

Champs de formulaire : options

Cases à cocher

Plusieurs choix possibles

```
<input type="checkbox" name="haricots">
<input type="checkbox" name="epinards">
```

Zones d'option

Un seul choix possible : on ne peut choisir qu'une valeur parmi les champs qui ont le même attribut `name`, l'attribut `value` permet d'identifier l'option choisie

```
<input type="checkbox" name="couleur" value="rouge">
<input type="checkbox" name="couleur" value="vert">
```

Listes déroulantes

```
<select name="couleur">
    <option value="rouge">Rouge</option>
    <option value="vert">Vert</option>
</select>
```

Titre des champs de formulaires

```
<label for="login">Login</label> :
<input type="text" name="login" id="login">
```

La valeur de l'attribut `for` du `<label>` doit correspondre à l'`id` de l'`<input>`

- Permet de mettre le focus sur le champ en cliquant sur le label
- $\Rightarrow$ Ergonomie et accessibilité

Envoyer le formulaire

Ajout d'un bouton pour valider le formulaire :

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

Navigation dans les formulaires au clavier

Les balises <input> peuvent prendre un attribut tabindex qui indique dans quel ordre le focus est passé aux champs du formulaire lors de l'appui sur la touche Tab

```
<input type="text" name="login" tabindex="1">
<input type="password" name="password" tabindex="2">
```

⇒ Ergonomie et accessibilité

Organisation des fichiers

- Une page web $\Leftrightarrow$ un fichier HTML (pour un site statique)
- Les serveurs sont généralement configurés pour que la page par défaut (« page d'accueil ») soit `index.html`

L'importance de la sémantique du langage

- Les balises et leurs attributs ont une signification
- Important pour de nombreux aspects :
  - Moteurs de recherche
  - Accessibilité
  - Affichage d'une page web sans mise en forme (pas depuis un navigateur, par exemple)
- $\Rightarrow$ Utilisez les balises selon leur sémantique, pas selon leur rendu !
- Exemples :
  - On n'utilise pas une balise pour la façon dont elle va être affichée : pas de balise titre de niveau 1 pour afficher quelque chose en très gros
  - On n'utilise pas des balises pour faire un tableau seulement parce que ça facilite la mise en page

Afficher et explorer le code source

- Il est possible d'**afficher** le code HTML de n'importe quelle page affichée dans un navigateur !
  - Clic droit > Code source de la page
  - Commande `wget`, `curl`, ...
  - Affiche ce qu'a renvoyé le serveur
- Il est possible d'**afficher**, **explorer** et **manipuler** le code HTML de n'importe quelle page affichée dans un navigateur !
  - En utilisant les outils de développement intégrés au navigateur
  - Clic droit > Inspecter
  - Affiche ce qu'a interprété le navigateur

Validateur W3C

- W3C (*World Wide Web Consortium*) : organisme de standardisation de certaines technologies du web
- Service en ligne de validation du code HTML : `https://validator.w3.org/`

Toutes vos pages HTML doivent être valides !

Ressources

- Le portail développeur de Mozilla : https://developer.mozilla.org/fr/
- W3Schools : https://www.w3schools.com/html/
- Grafikart : https://grafikart.fr/formations/html
- L'accessibilité numérique : https://simulation-accessibilite.inria.fr/

Exercice : article de blog

Faites une page qui affiche un article de blog. Elle contiendra les éléments suivants :

- Menu (accueil, à propos, contact, ...)
- Titre du blog
- Titre de l'article
- Le contenu de l'article avec des sous-titres
- Une date de publication
- Un auteur
- Des commentaires
  - Contenu du commentaire
  - Auteur du commentaire
  - Date du commentaire
- Un formulaire pour poster un commentaire

Pour générer du faux texte, on pourra utiliser le *lorem ipsum*.

Blog de Tartanpion

- Accueil
- Articles
- À propos
- Contact

Vive les pandas

Gérard Menvuça

Publié le jeudi 15 février 2024

Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Quis varius quam quisque id diam. Duis at tellus at urna condimentum mattis pellentesque. Tortor aliquam nulla facilisi cras fermentum odio eu feugiat pretium. Sed vulputate mi sit amet mauris commodo quis imperdiet massa. Ut enim blandit volutpat maecenas volutpat. Scelerisque in dictum non consectetur a erat nam. At in tellus integer feugiat scelerisque varius morbi enim nunc. Sit amet nisl suscipit adipiscing bibendum est ultricies integer. Malesuada bibendum arcu vitae elementum curabitur vitae nunc sed. Suscipit adipiscing bibendum est ultricies integer quis auctor elit. Vitae tempus quam pellentesque nec nam aliquam sem et tortor.

Panda roux mangeant du bambou

Section 1

Sed vulputate odio ut enim blandit. Amet mattis vulputate enim nulla aliquet porttitor lacus luctus accumsan. Egestas pretium aenean pharetra magna ac placerat vestibulum lectus. Mus mauris vitae ultricies leo integer malesuada nunc vel. Odio eu feugiat pretium nibh. Eget lorem dolor sed viverra ipsum nunc aliquet. Eu lobortis elementum nibh tellus. Cursus turpis massa tincidunt dui. Facilisi cras fermentum odio eu feugiat pretium nibh ipsum. Fermentum iaculis eu non diam phasellus. Nibh venenatis cras sed felis eget velit. Vel facilisis volutpat est velit egestas.
Viverra tellus in hac habitasse platea. Convallis posuere morbi leo urna molestie. Habitasse platea dictumst vestibulum rhoncus est pellentesque elit ullamcorper. Mus mauris vitae ultricies leo integer. Lobortis scelerisque fermentum dui faucibus in ornare quam. Velit scelerisque in dictum non consectetur. Neque volutpat ac tincidunt vitae semper quis. Elit sed vulputate mi sit amet mauris commodo. Lorem dolor sed viverra ipsum nunc. Nulla aliquet porttitor lacus luctus accumsan tortor posuere ac. Bibendum at varius vel pharetra vel turpis nunc. Amet cursus sit amet dictum. Faucibus pulvinar elementum integer enim neque.
Urna porttitor rhoncus dolor purus non enim praesent elementum. Elit sed vulputate mi sit amet mauris commodo quis. Pulvinar elementum integer enim neque volutpat ac tincidunt vitae semper. Molestie a iaculis at erat pellentesque adipiscing commodo elit at. Et netus et malesuada fames ac. Dignissim diam quis enim lobortis scelerisque. Tortor vitae purus faucibus ornare. Porttitor eget dolor morbi non. Dignissim sodales ut eu sem integer. Odio pellentesque diam volutpat commodo sed. Venenatis cras sed felis eget velit aliquet sagittis. Laoreet non curabitur gravida arcu. Sed vulputate odio ut enim blandit volutpat maecenas volutpat. Eget egestas purus viverra accumsan in.

Section 2

Fermentum leo vel orci porta non. Nunc non blandit massa enim nec dui. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Scelerisque viverra mauris in aliquam sem fringilla ut morbi tincidunt. Vitae et leo duis ut diam. Bibendum ut tristique et egestas quis ipsum suspendisse ultrices gravida. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt tortor. Amet consectetur adipiscing elit duis tristique sollicitudin nibh. Interdum varius sit amet mattis vulputate enim. Risus viverra adipiscing at in. Massa id neque aliquam vestibulum morbi blandit cursus risus. Risus in hendrerit gravida rutrum. Tortor consequat id porta nibh venenatis cras sed. Aenean vel elit scelerisque mauris pellentesque pulvinar. Feugiat in fermentum posuere urna. Magna etiam tempor orci eu lobortis. Mi bibendum neque egestas congue quisque egestas diam in.

Conclusion

Fermentum leo vel orci porta non. Nunc non blandit massa enim nec dui. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Scelerisque viverra mauris in aliquam sem fringilla ut morbi tincidunt. Vitae et leo duis ut diam. Bibendum ut tristique et egestas quis ipsum suspendisse ultrices gravida. Phasellus egestas tellus rutrum tellus pellentesque eu tincidunt tortor. Amet consectetur adipiscing elit duis tristique sollicitudin nibh. Interdum varius sit amet mattis vulputate enim. Risus viverra adipiscing at in. Massa id neque aliquam vestibulum morbi blandit cursus risus. Risus in hendrerit gravida rutrum. Tortor consequat id porta nibh venenatis cras sed. Aenean vel elit scelerisque mauris pellentesque pulvinar. Feugiat in fermentum posuere urna. Magna etiam tempor orci eu lobortis. Mi bibendum neque egestas congue quisque egestas diam in.

Commentaires

Fermentum leo vel orci porta non. Nunc non blandit massa enim nec dui. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Scelerisque viverra mauris in aliquam sem fringilla ut morbi tincidunt. Vitae et leo duis ut diam.

Toto

Publié le vendredi 16 février 2024

Fermentum leo vel orci porta non. Nunc non blandit massa enim nec dui. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Scelerisque viverra mauris in aliquam sem fringilla ut morbi tincidunt. Vitae et leo duis ut diam.

Toto

Publié le vendredi 16 février 2024

Fermentum leo vel orci porta non. Nunc non blandit massa enim nec dui. Sit amet venenatis urna cursus eget nunc scelerisque viverra. Scelerisque viverra mauris in aliquam sem fringilla ut morbi tincidunt. Vitae et leo duis ut diam.

Toto

Publié le vendredi 16 février 2024

Ajouter un commentaire

Pseudo Adresse email Commentaire Envoyer
Mentions légales
© 2024 - Tartanpion

À vous de jouer !

Mise en forme des pages web avec CSS

Mise en forme

- HTML pur $\Rightarrow$ le navigateur met en forme selon son style par défaut
- Mise en forme avec des règles CSS (*Cascading Style Sheets*)
- CSS 3 actuellement (couple HTML5/CSS3)

Support des navigateurs

- L'affichage d'une page web par un navigateur dépend du moteur de rendu utilisé
- Support et interprétation de certaines fonctionnalités différentes selon les moteurs de rendus (vrai pour HTML et CSS)
- Le style par défaut peut varier d'un navigateur à l'autre[2]
- D'une manière générale : **il faut tester le rendu de son site Internet sur les différent navigateurs existants**

2. Il y a des solutions pour uniformiser le rendu par défaut, par exemple : https://necolas.github.io/normalize.css/

Où mettre le code CSS ?

Dans un fichier dédié

Méthode préférée : tout le code CSS dans un fichier `.css`, puis dans la balise `<head>` :

```
<link rel="stylesheet" href="style.css">
```

Directement dans le fichier HTML

Dans la balise `<head>` :

```
<style>
/* règles CSS */
</style>
```

Directement comme attribut des balises concernées

À éviter

```
<p style="color: white; backgroud-color=black;">
    Blabla
</p>
```

Règles CSS

```
selecteur {
    propriete1: valeur;
    propriete2: valeur;
}

autre_selecteur {
    propriete1: valeur;
    propriete2: valeur;
}

/* autres règles (et exemple de commentaire au
   passage) */
```

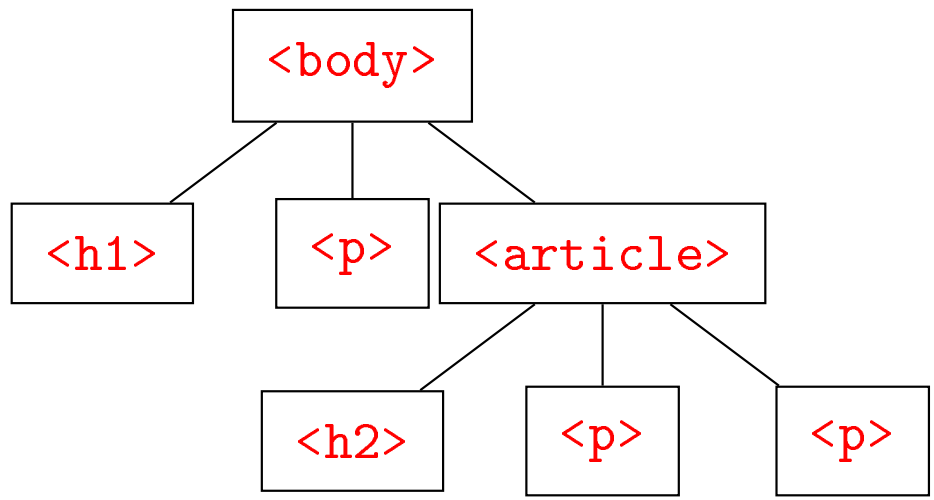
Sélecteurs

- Permet de choisir à quels éléments appliquer la règle
- Quand applicable, les éléments enfants appliquent les règles de leurs parents
- Référence :
  https://developer.mozilla.org/fr/docs/web/css/css_selectors

Sélecteurs

Sélectionner tous les éléments

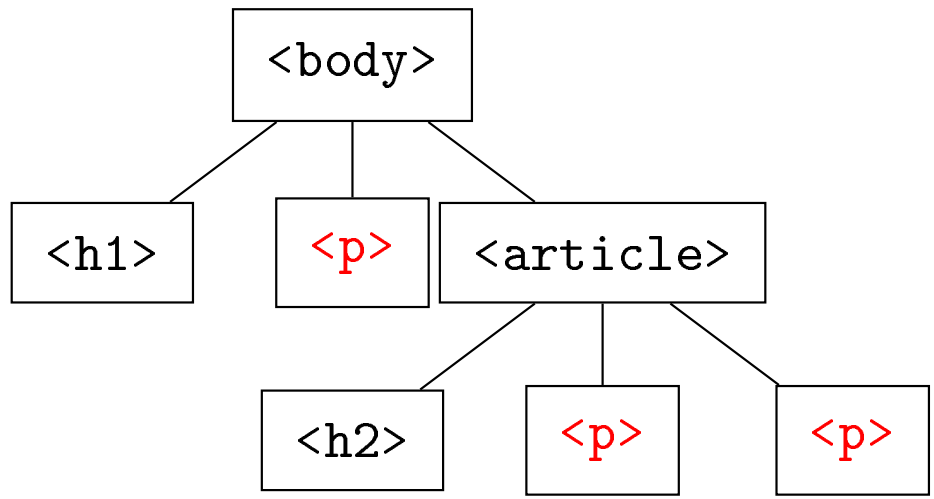
```
* {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélectionner un type de balise

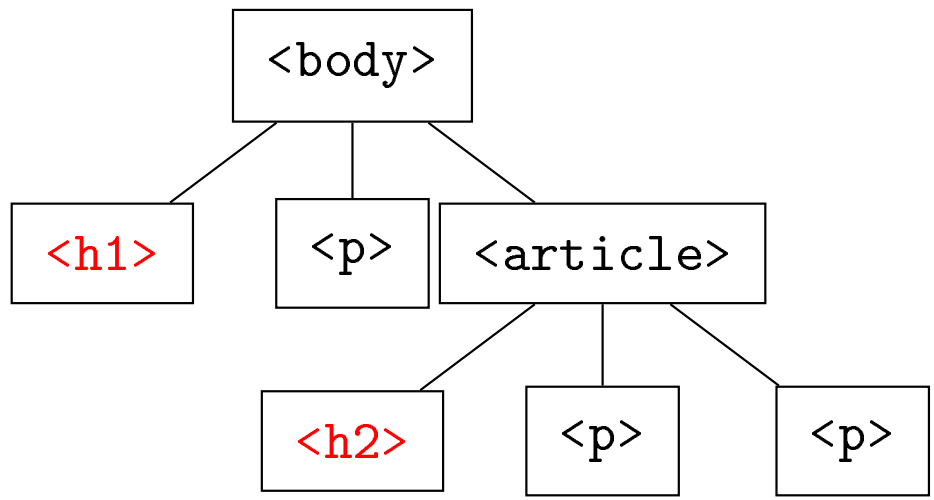
```
p {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Union de sélecteurs

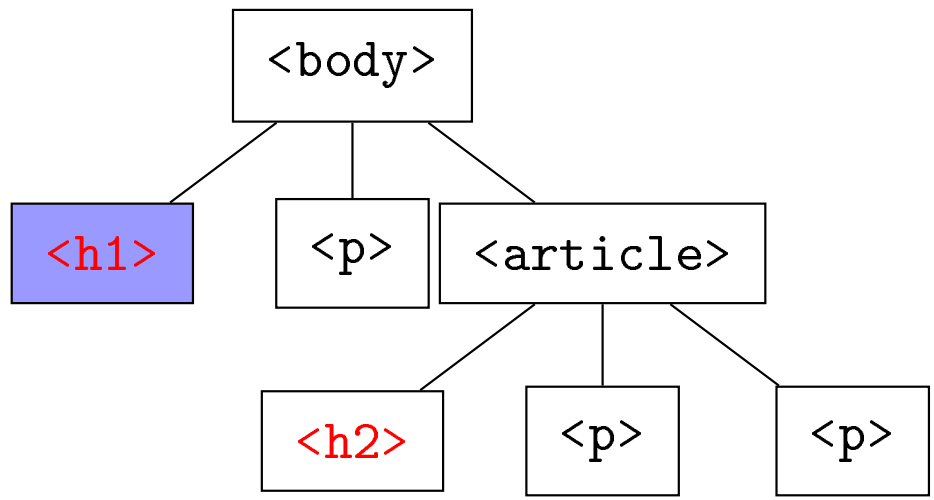
```
h1, h2 {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Combinaison de règles

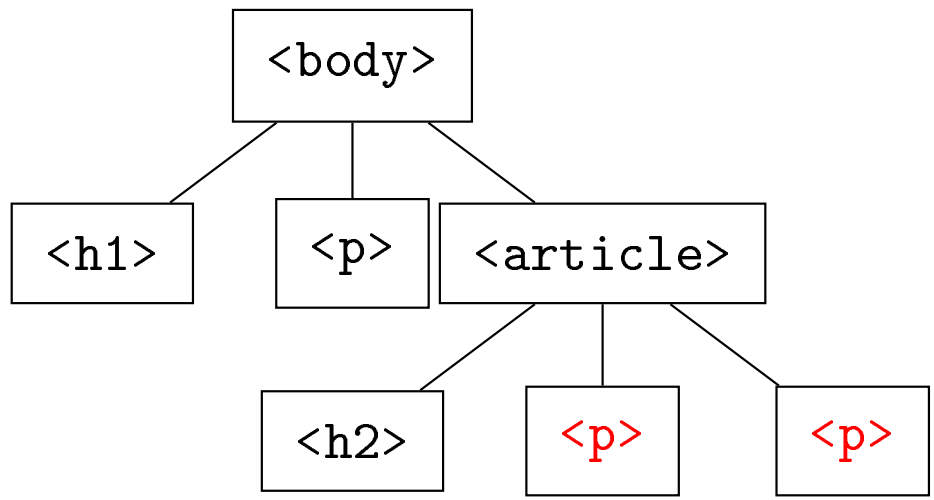
```
h1, h2 {
    color: red;
}

h1 {
    background-color:
        blue;
}
```



Sélecteurs

Sélection de tous les descendants

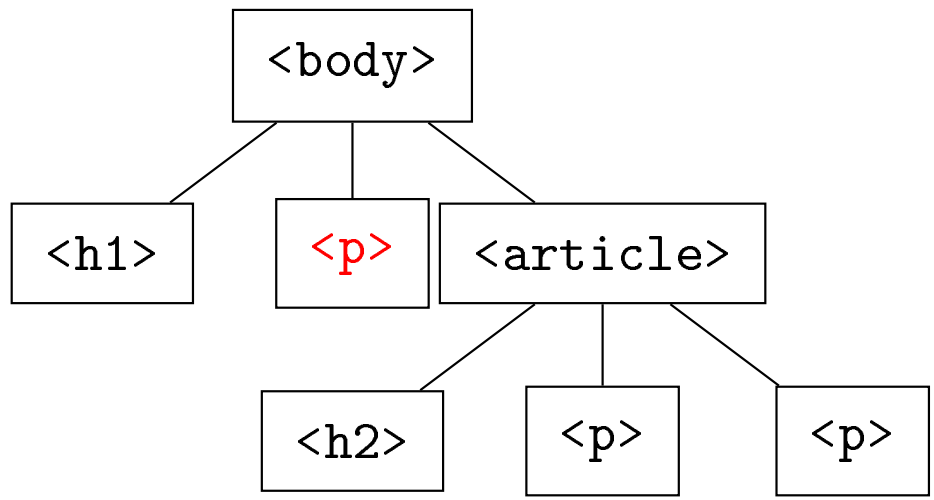
```
article p {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélection des enfants directs

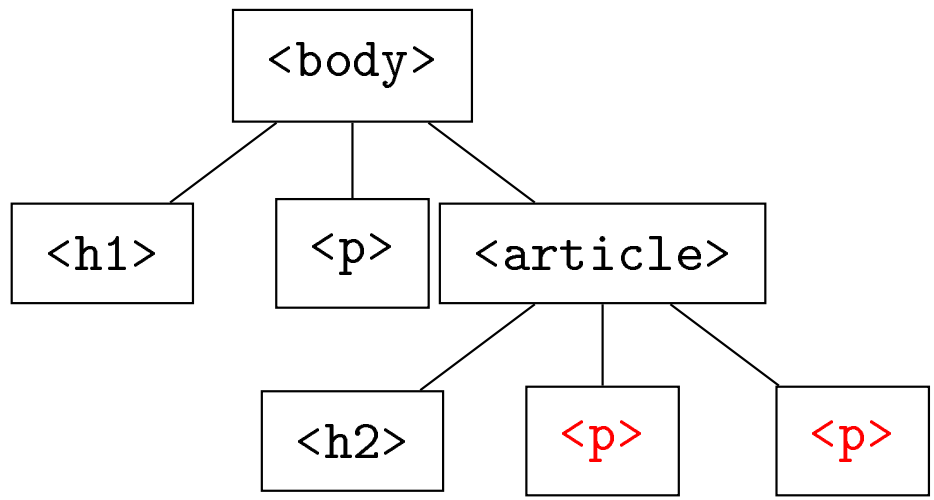
```
body > p {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélection des voisins

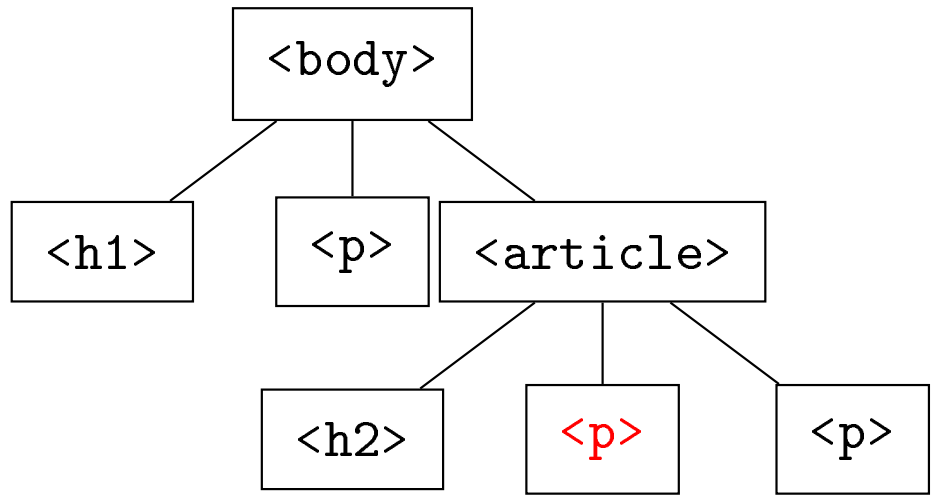
```
h2 ~ p {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélection du premier voisin

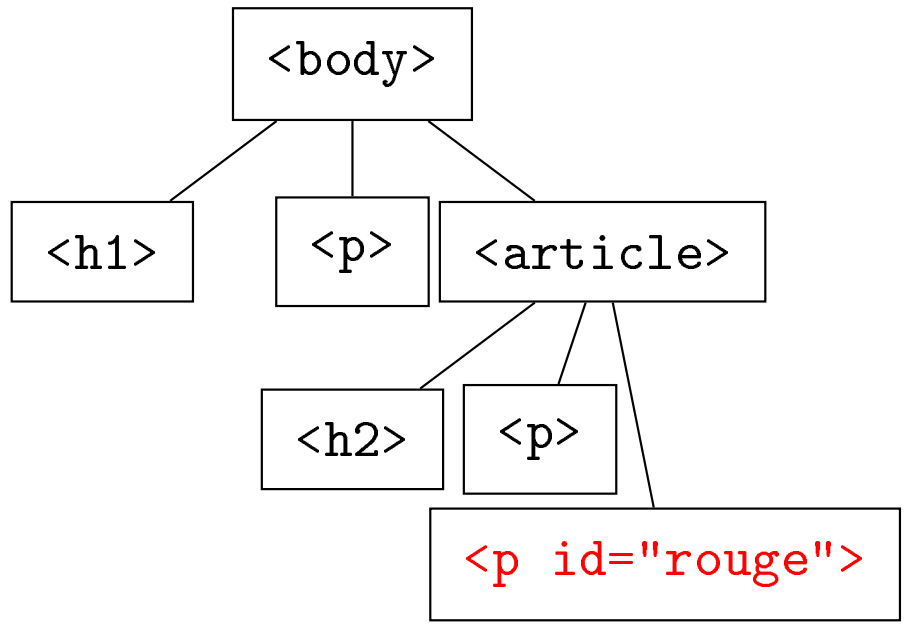
```
h2 + p {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélection selon l'ID

```
#rouge {
    /* texte en rouge */
    color: red;
}
```



Sélecteurs

Sélection selon un attribut

```
input[required] {
    /* Tous les <input required> */
}

input[type="password"] {
    /* Tous les <input type="password"> */
}
```

Classes

Toutes les balises peuvent prendre un attribut `class` :

```
<h2 class="rouge">Un titre en rouge</h2>
<p class="rouge">Du texte en rouge</p>
```

Il est possible de sélectionner en CSS directement selon la valeur de cet attribut en préfixant avec un point :

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

Pseudo-classes

- Sélectionne un élément dans un état particulier
- Exemples :
  - `:hover` : la souris est sur l'élément
  - `:focus` : l'élément a le focus (par exemple le curseur est dans ce champ de formulaire)
  - `:visited` : ce lien a déjà été visité
  - `:nth-child(3)` : le 3ème enfant
- Référence :
  https://developer.mozilla.org/fr/docs/Web/CSS/Pseudo-classes

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

Dans le même genre, les *pseudo-éléments* permettent de sélectionner seulement une partie de l'élément ciblé

Propriétés de formatage de texte

Propriété	Rôle
`font-family`	Police de caractères
`font-size`	Taille du texte
`font-weight`	Graisse du texte
`font-style`	Italique
`text-decoration`	Souligné, barré, ...
`text-transform`	Forcer la casse
`text-shadow`	Ombre de texte
`font`	Propriété qui permet de spécifier plusieurs propriétés ci-dessus d'un coup

Propriétés d'alignement du texte

Propriété	Rôle
`text-align`	Alignement horizontal
`vertical-align`	Alignement vertical
`line-height`	Hauteur de ligne
`text-indent`	Alinéa

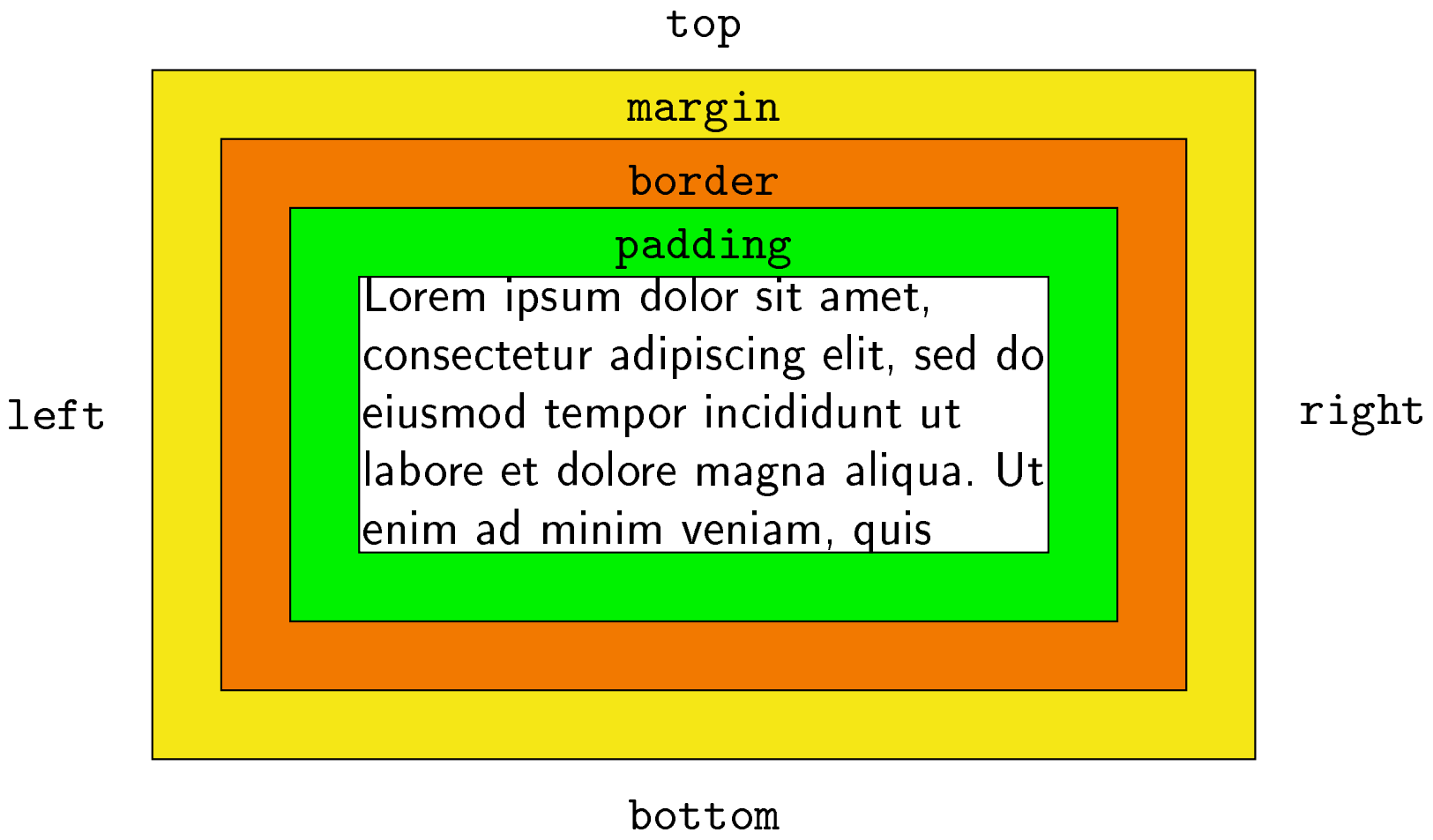
Propriétés de couleurs

Propriété	Rôle
`color`	Couleur du texte
`background-color`	Couleur d'arrière-plan

Valeurs de couleurs possibles

- Nom anglais de la couleur : `black`, `blue`, `red`, ...
- Notation hexadécimale : `#ff0000`
- Notation RGB : `rgb(255, 0, 0)`
- Notation RGB avec transparence : `rgb(255, 0, 0, 0.8)`
- Et d'autres :
  https://developer.mozilla.org/en-US/docs/Web/CSS/color_value

Le modèle de boîte



Propriétés des boîtes

Propriété	Rôle
{min-,max-,}width	Largeur (minimale, maximale) de la boîte
{min-,max-,}height	Hauteur (minimale, maximale) de la boîte
margin	Taille de la marge extérieure
padding	Taille de la marge intérieure
margin-left	Taille de la marge extérieure gauche
border-width	Épaisseur de la bordure
border-color	Couleur de la bordure
border-style	Style de la bordure (trait plein, tirets, ...)
border-top-style	Style de la bordure au nord
border	Propriété qui combine toutes les propriétés des bordures
border-top	Propriété qui combine toutes les propriétés des bordures au nord

Unités des tailles

- Pourcentage : relatif à la taille du parent
  - `10%`
- Nombre de pixels :
  - `10px`
- Relatif à la taille de la police :
  - `1.2em`
- Référence :
  https://developer.mozilla.org/fr/docs/Web/CSS/length

Positionement des blocs

Avec la propriété `position` :

- `static` : par défaut
- `absolute` : position absolue par rapport à toute la page
- `fixed` : position absolue par rapport à la fenêtre, donc bloc immobile lors du défilement de la page
- `relative` : position par rapport à la position par défaut
- `sticky` : le bloc reste entièrement visible à l'écran tant qu'une partie de son bloc parent est visible à l'écran

Pour `absolute`, `fixed`, `relative` et `sticky`, nécessité d'ajouter les coordonnées :

- `top` ou `bottom`
- `left` ou `right`

Référence : https://developer.mozilla.org/fr/docs/Web/CSS/position

Flottants

```
<section>
    <div class="left"></div>
    <p>
    Lorem ipsum dolor sit ...
    </p>
</section>
```

```
.left {
  float: left;
  background: pink;
  width: 60px;
  height: 35px;
  margin: 0 5px;
  border: blue 2px solid;
}
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi tristique sapien ac erat tincidunt, sit amet dignissim lectus vulputate. Donec id iaculis velit. Aliquam vel malesuada erat. Praesent non magna ac massa aliquet tincidunt vel in massa. Phasellus feugiat est vel leo finibus congue.

Colonnes

L'ancienne façon

L'élément n'est plus de type bloc, mais un type mi-bloc mi-en ligne

```
<div>
    <div class="colonne">
        Colonne 1
    </div>
    <div class="colonne">
        Colonne 2
    </div>
    <div class="colonne">
        Colonne 3
    </div>
    <div class="colonne">
        Colonne 4
    </div>
</div>
```

```
.colonne {
    background-color: pink;
    margin: 3px;
    height: 50px;
    display: inline-block;
    width: 23%;
}
```

Colonnes

La nouvelle façon

Les *flex-box* : ça fait le café !

```
<div class="parent">
    <div class="colonne">
        Colonne 1
    </div>
    <div class="colonne">
        Colonne 2
    </div>
    <div class="colonne">
        Colonne 3
    </div>
    <div class="colonne">
        Colonne 4
    </div>
</div>
```

```
.parent {
    display: flex;
}

.colonne {
    background-color: pink;
    margin: 3px;
    height: 50px;
    flex: 1;
}
```

Autres propriétés en vrac

- Image de fond
- Listes
- Tableaux
- Curseur de la souris
- Animations

Media queries

- Problème : largeur des écrans très hétérogène (ordinateur, tablette, téléphone, ...)
- La mise en page est différente sur un écran de téléphone ou un écran 27" !
- *Media queries* pour appliquer des règles CSS suivant la largeur de l'écran (entre autres)

```
@media (max-width: 800px) {
  .truc {
      width: 100%;
  }
}

@media (min-width: 800px) {
  .truc {
      width: 800px;
  }
}
```

Outils du navigateur pour le CSS

- Comme pour inspecter le code HTML
- On peut :
  - Voir quelles propriétés CSS sont appliquées...
  - ... à partir de quelles règles
  - Ajouter des propriétés, changer des valeurs
  - Redimensionner la fenêtre

Validateur

https://jigsaw.w3.org/css-validator/

Exercice : donnez du style à votre blog

Créez une feuille de style pour votre blog. Au choix :

- Faites un blog très beau
- Faites un blog très moche

À vous de jouer !

Préprocesseurs CSS

- Le code CSS peut vite devenir compliqué
- Préprocesseurs CSS qui améliorent la syntaxe CSS et génèrent ensuite du CSS :
  - Variables (ça devient aussi possible en CSS3)
  - *Mixins* (extraits de code qu'on peut insérer à différents endroits)
  - Imbrication
  - Fonctions et opérations mathématiques
- Exemples : LESS, SASS, ...

Frameworks CSS

- Le code CSS (avec ou sans préprocesseur) peut vite devenir compliqué
- Faire un style CSS à peu près élégant et fonctionnel prend du temps, demande de l'expérience (c'est un métier à part entière)
- $\Rightarrow$ utilisation de *frameworks* CSS !
- Fournissent des feuilles de style CSS prêtes à l'emploi, il n'y a plus qu'à appliquer les classes CSS aux balises HTML !
- Exemples :
  - Bootstrap (le plus connu ; utilise SASS !)
  - Foundation
  - Tailwind
  - *cf* https://usablica.github.io/front-end-frameworks/compare.html

Programmation web côté serveur avec PHP

Programmation web côté serveur

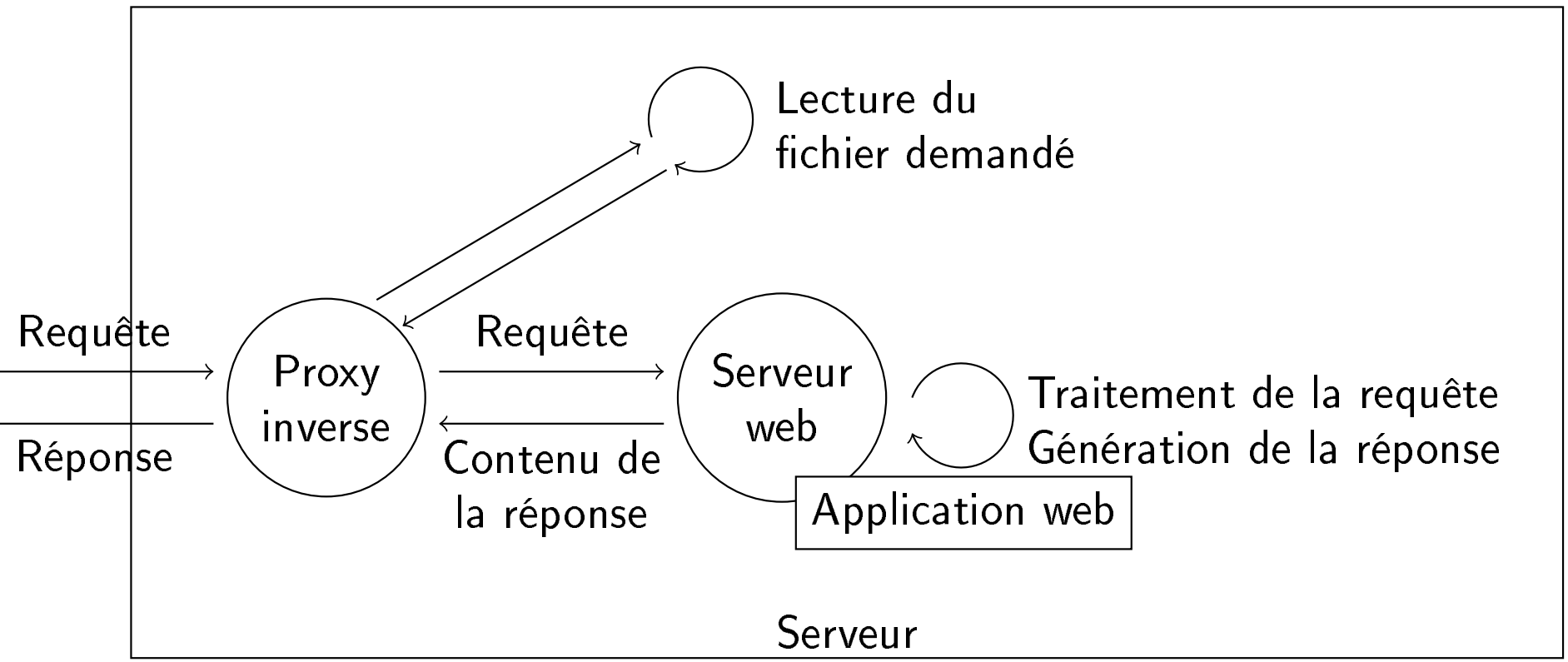
Fonctionnalités d'un serveur web dynamique :

- Écoute sur un port TCP, une socket
- Analyse des requêtes reçues (*parsing* des requêtes HTTP)
- **Traitement** de la requête
- **Génération** du contenu de la réponse
- Envoi de la réponse

On distingue :

- le (logiciel) serveur web dynamique
  - Pas vu dans ce cours
- le code exécuté pour générer le contenu de la réponse
  - *Application web*
  - Vu dans la suite de ce cours

Rappel : rôle du serveur pour un site dynamique



Langages de programmation web côté serveur

- **PHP**
- Python
- Java
- Ruby
- Perl
- JavaScript
- C (bon courage !)
- et d'autres...

- Concepts fondamentaux identiques

Distinction *front-end* / *back-end*

Dans les offres d'emploi liées à la programmation web, vous pouvez trouver les termes :

Front-end

- Tout ce que va recevoir et interpréter le navigateur
- HTML, CSS, JavaScript
- Ce qu'on a vu jusqu'à maintenant

Back-end

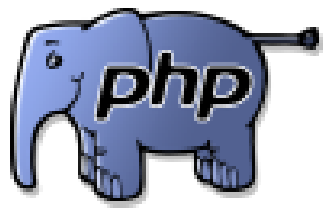
- Tout ce qui est exécuté sur le serveur, pour générer la page
- Ce qu'on va voir maintenant

Full-stack

- Une personne à l'aise à la fois en back-end et en front-end

PHP

- *PHP : Hypertext Preprocessor*
- Naissance en 1994 pour compter le nombre de visiteurs sur un site web
- Utilisé par la majorité des sites Internet (exemples : Facebook, Wikipedia, Wordpress, ...)
- Multiplateformes
- Version 8 actuellement
- Langage impératif, orienté objet, interprété, faiblement typé



`https://www.php.net/`

Astuce : `php.net/foo` mène à la documentation de la fonction `foo`

PHP à l'ENSEIRB-MATMECA

Activation

- Suivre les instructions à l'adresse https://zzz.bordeaux-inp.fr/
  - « *La page de gestion de votre compte* » est accessible à l'adresse https://ccc.bordeaux-inp.fr/moncompte/
- Accessible uniquement depuis le réseau de l'école

Utilisation

1. Depuis l'explorateur de fichier Thunar : Ctrl+L ou menu *Aller* puis *Ouvrir l'emplacement*
2. Saisir l'adresse `sftp://login@zzz.bordeaux-inp.fr/data/www/`
3. Le mot de passe qui vous est demandé est **le mot de la passe que vous avez renseigné à l'activation de votre espace web !**
4. Depuis Thunar, vous pouvez faire un clic droit et *Ouvrir un terminal ici* pour obtenir un terminal placé dans le dossier monté
5. Site accessible à l'adresse https://login.zzz.bordeaux-inp.fr/

PHP sur votre machine personnelle

Plusieurs solutions :

- Tout installer comme sur un vrai serveur web
  - Transforme votre machine en véritable serveur web
- Utiliser une machine virtuelle, un conteneur Docker, ...
- **Utiliser XAMPP** : logiciel tout-en-un qui contient tout ce dont on a besoin

Installer et utiliser XAMPP sur votre machine

1. Téléchargez le programme depuis
   https://www.apachefriends.org
2. Dans un terminal :

```
chmod u+x xampp-linux-x64-8.2.12-0-installer.run
sudo ./xampp-linux-x64-8.2.12-0-installer.run
sudo apt install net-tools
```

3. Lancement des services :

```
sudo /opt/lampp/xampp start
```

4. Tous vos fichiers doivent être dans `/opt/lampp/htdocs/`
   - S'attribuer les droits sur ce dossier :

```
chmod -R 777 /opt/lampp/htdocs/
```

   - Faire un dossier par projet dans ce dossier, par exemple : `/opt/lampp/htdocs/it103`
   - Ensuite, dans le navigateur le site est accessible à l'adresse : http://localhost/it103/

L'incontournable *Hello world*

Dans un fichier `hello.php` :

```
<?php echo "Hello world"; ?>
```

Accédez à http://adressedevotreserveur/dossier/hello.php

L'incontournable *Hello world*

Dans un fichier `hello.php` :

```php
<?php echo "Hello world"; ?>
```

Accédez à http://adressedevotreserveur/dossier/hello.php

- `<?php` : balise ouvrante : « *ce qui suit va être du code PHP* »
- `echo "Hello world"` : « *affiche `Hello world`* »
- `;` : fin d'une instruction (comme en C)
- `?>` : balise fermante, fin du code PHP. Pas nécessaire si c'est la fin du fichier

Code PHP au milieu de code HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titre de la page</title>
        <meta charset="utf-8">
    </head>
    <body>
        <p>
            <?php echo "Hello world"; ?>
        </p>
    </body>
</html>
```

- Le rôle du serveur PHP est de remplacer tout le code `<?php ... ?>` par ce qu'il affiche

Inclusion de fichiers

```
<!DOCTYPE html>
<html>
    <head>
        <title>Titre de la page</title>
        <meta charset="utf-8">
    </head>
    <body>
        <?php include("header.php"); ?>
        <p>
            <?php echo "Hello world"; ?>
        </p>
        <?php include("footer.php"); ?>
    </body>
</html>
```

- Les fichiers `header.php` et `footer.php` contiennent le contenu commun à toutes les pages du site
- On peut inclure n'importe quel type de fichier textuel (HTML, par exemple)

Afficher les messages d'erreur

- Selon votre configuration de PHP, les avertissement et messages d'erreurs peuvent s'afficher directement dans la réponse HTTP (et donc dans le navigateur)
- Tous ces messages sont normalement sauvegardés dans un fichier de *logs*, par exemple dans `/opt/lampp/logs/php_error_log`
- Pour activer l'affichage des erreurs, au choix :
  - Ajouter au début de chaque fichier PHP :

```php
ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);
```

  - Changer ces valeurs de configuration directement dans le fichier de configuration de PHP, par exemple dans `/opt/lampp/etc/php.ini` (il faudra redémarrer le serveur)
  - Pas forcément possible sur toutes les installations
  - Uniquement pour du développement ! Pas en production !

La syntaxe PHP en 5 diapos : 1. Les variables

- Les noms de variables commencent par $
- Remarquez au passage la syntaxe pour les commentaires

```php
$verite = true; // booléen
$age = 24; // entier
$temperature = 15.3; // flottant
$texte = "Texte"; // chaîne de caractères

/* Opérations arithmétiques comme en C : */
$resultat = $age * $temperature - 13;
$age++;
$age *= 2;

/* Constantes: */
define("FOO", 67);
$resultats = 67 * FOO; // pas de $

$valeur_nulle = null;
```

Plusieurs balises PHP sur une même page

- Les balises PHP d'une page partagent toutes le même espace mémoire

```
<?php
$prenom = "Toto";
?>

<!-- du code HTML... -->

<p>
    Bonjour <?php echo $prenom; ?> !
</p>
```

Affichera `Bonjour Toto`

Concaténation

Concaténer des chaînes de caractères et des variables :

```php
<?php
$prenom = "Toto";
$nom = "Bonne Question";
$nom_complet = $prenom . " " . $nom;
$nb_billes = 8;
$billes = $prenom . " a " . $nb_billes . " billes";
?>

<!-- du code HTML... -->

<p>
    <?php echo "Bonjour " . $nom_complet; ?> !<br>
    <?php echo $billes; ?> !
</p>
```

La syntaxe PHP en 5 diapos : 2. Les conditions

- Comparaison : ==, <, <=, >, >=, !=
- Logique : !$var, &&, ||

```php
if ($a == $b)
{
    echo "a == b";
}
elseif ($a == $c)
{
    echo "a == c";
}
else
{
    echo "a != b et c";
}
```

```php
// Condition ternaire :
$truc = $var ? "vrai" : "faux
    ";

/* Raccourci pour
   $x ? $x : "faux" : */
$truc = $x ?: "faux";

switch ($x) {
    case 0:
        echo "Zéro";
        break;
    case "un":
        echo "1";
    default:
        echo "défaut";
}
```

Conditions et code HTML

```
<p>
    <?php if ($est_connecte) { ?>
    Bienvenue <?php echo $prenom; ?> !<br>
    <a href="logout.php">Se déconnecter</a>
    <?php } else { ?>
    <a href="login.php">Se connecter</a>
    <?php } ?>
</p>
```

La syntaxe PHP en 5 diapos : 3. Les boucles

```php
$i = 0;
while ($i < 12)
{
    echo $i;
}

for ($i = 0; $i < 12; $i++)
{
    echo $i;
}

do {
    echo $i;
} while ($i < 12);
```

- `break` et `continue` fonctionnent comme en C

La syntaxe PHP en 5 diapos : 4. Les tableaux

```php
$tableau = array("pg109", "pg110");
$tableau = ["pg109", "pg110"];
echo $tableau[0];
$tableau[1] = "PG110";
array_push($tableau, "it103");

for ($i = 0; $i < count($tableau); $i++)
{
    echo $tableau[$i];
}

foreach ($tableau as $e)
{
    echo $e;
}
```

- Beaucoup de fonctions pour manipuler les tableaux

La syntaxe PHP en 5 diapos : 4. Les tableaux associatifs

```php
$cours = array(
    "pg109" => "C",
    "pg110" => "Projet"
);
$cours = ["pg109" => "C", "pg110" => "Projet"];
echo cours["pg110"];
cours["it103"] = "Web";

foreach ($cours as $nom)
{
    echo $nom;
}

foreach ($cours as $code => $nom)
{
    echo $code . " : " . $nom;
}
```

- Beaucoup de fonctions pour manipuler les tableaux associatifs

La syntaxe PHP en 5 diapos : 5. Les fonctions

```php
function truc($param1, $param2)
{
    // fait des trucs...

    // pas obligatoire si rien à renvoyer
    return $result;
}

$r = truc($a, $b);
```

Exercice : utilisez PHP pour votre blog

1. Reprenez votre (magnifique) blog fait lors de la dernière séance
2. Créez une deuxième page (contact, à propos, n'importe quoi) qui aura le même en-tête et pied-de-page
3. Ajoutez des liens qui permettent d'aller d'une page à l'autre
4. Utilisez PHP et la fonction `include()` pour factoriser le code commun entre les deux pages

À vous de jouer !

Paramètres passés par l'URL

`https://serveur.com/fichier.php?param1=valeur1&param2=valeur2`

- `?` pour séparer l'adresse des paramètres
- `&` pour séparer ensuite les couples nom de paramètre et valeur

- `fichier.php` récupère ces paramètres et leurs valeurs dans le tableau associatif `$_GET` :

```php
echo $_GET["param1"];

if ($_GET["param2"] == "value2")
{
    // ...
}

if (isset($_GET["param3"])
{
    // il y a un paramètre 'param3'
}
```

Exercice : afficher ou masquer les commentaires

Ajoutez un lien en bas de votre article de blog qui permet d'afficher la même page avec ou sans les commentaires. Par défaut les commentaires seront masqués.

L'URL pour afficher les commentaires sera de la forme `...blog.php?show_comments=on`

À vous de jouer !

Récupérer les données des formulaires

- Requête HTTP `POST`
- Utilisé par la majorité des formulaires

```
<form method="post" action="XXX">
    <input type="text" name="pseudo" required>
    <textarea name="comment"></textarea>

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

À quelle adresse envoyer les données d'un formulaire ?

À la même page

- La page va traiter les données, puis générer la réponse
- La page doit fonctionne en `GET` et en `POST`
  - ⇒ Besoin de connaître le type de requête dans le code PHP
- Facilite l'affichage d'erreur

À une autre page

- La page va traiter les données, puis (potentiellement) rediriger vers une autre page
- La page doit fonctionner uniquement en `POST`
  - ⇒ Besoin de connaître le type de requête dans le code PHP
    - Renvoyer une erreur si pas de type `POST`

Connaître le type de requête HTTP

- Stocké dans `$_SERVER["REQUEST_METHOD"]`

```php
if ($_SERVER["REQUEST_METHOD"] == "POST")
{
    // C'est une requête POST
}
```

Faire une redirection

```php
header("Location: " . $url);
exit;
```

- `header()` positionne une en-tête de la réponse
- Doit être appelée avant que du contenu de la réponse ne soit généré
- C'est le navigateur qui réalise la redirection en voyant l'en-tête `Location`
  - Aucune garantie que le navigateur fasse la redirection !
- Pourquoi c'est mieux de mettre `exit` ?

Récupérer les données tranmises en POST

- Comme pour `$_GET`, les données POST sont dans le tableau `$_POST`
- Les clés du tableaux associatifs correspondent aux attributs `name` des champs du formulaire

```php
echo $_POST["pseudo"];

if (isset($_POST["comment"])
{
    // on a bien reçu un commentaire
}
```

- Pour transformer les retours à la ligne dans le `<textarea>` en balises `<br>` : fonction `nl2br()`

Types des paramètres

- Tous les valeurs des tableaux `$_GET` et `$_POST` sont des **chaînes de caractères**
- Conversion vers un nombre entier possible avec la fonction `intval()`

Combiner paramètres GET et POST

- Il est tout à fait possible d'utiliser les deux types paramètres en même temps

```
<form method="post" action="blog.php?article_id=12">
    <input type="text" name="pseudo" required>
    <textarea name="comment"></textarea>

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

```
$article_id = intval($_GET["article_id"]);
$comment = $_POST["comment"];
```

Exercice : afficher le nouveau commentaire

Modifiez le code de la page qui affiche l'article de blog pour qu'une fois le formulaire pour commenter validé, la même page s'affiche, mais avec le nouveau commentaire en plus.

Le formulaire enverra les données vers la même page. On n'a pas encore vu comment sauvegarder des données, donc le nouveau commentaire disparaitra si on recharge la page.

Pour afficher la date du commentaire, on peut utiliser :

```php
<?php echo date("l j F Y"); ?>
```

(il est possible que la date ne s'affiche pas en français, il est possible de corriger ça, mais c'est un peu casse-pieds pour ce simple exercice, je vous laisse chercher)

À vous de jouer !

Ne jamais faire confiance à l'utilisateur !

Vous ne pouvez pas faire confiance aux données fournies par l'internaute !

Vous n'avez aucune garantie que l'internaute va utiliser votre site web de façon « normale » (de façon intentionnelle ou non) :

1. les requêtes HTTP ont-elles le bon type ?
2. les paramètres `GET` et/ou `POST` attendus sont-ils bien présents ?
3. les paramètres présents sont-ils corrects ? Exemple pour `article_id` :
   - le type est-il correct ?
   - l'article avec l'ID fourni existe-t-il ?
   - l'internaute a-t-il le droit de poster un commentaire pour cet article ?
4. les valeurs des paramètres ne contiennent-elles rien de *dangereux* ?

$\Rightarrow$ Il faut vérifier tout ces points avant de faire confiance aux données envoyées par l'utilisateur !

Notifier à l'internaute que sa requête n'est pas valide

- Afficher la page avec un message d'erreur (ex. : formulaire invalide)
- Renvoyer un code HTTP d'erreur et tout arrêter :

```php
http_response_code(400);
exit;
```

  - Le proxy inverse peut être configuré pour intercepter ces codes d'erreur et afficher une page d'erreur spécifique

Code	Signification
400	*Bad Request*
401	*Unauthorized*
403	*Fobidden*
404	*Not Found*
405	*Method Not Allowed*
418	*I'm a teapot*
Et beaucoup d'autres[3]	

3. https://fr.wikipedia.org/wiki/Liste_des_codes_HTTP

Contourner les restrictions des formulaires HTML

Rien de plus facile !

```
<form method="post" action="truc.php">
    <input type="text" name="pseudo" minlength="3"
        maxlength="20" required>
    <input type="email" name="email" required>

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

- Les bons navigateurs vont empêcher d'envoyer le formulaire en cas de donées invalides
  - $\Rightarrow$ Côté client : aucune garantie !
- Avec les outils du navigateur, facile de :
  - Enlever les attributs HTML contraignants
  - Renvoyer la même requête en changeant les données
  - Récupérer la commande `curl` à exécuter dans un terminal

Il faut refaire toutes les vérifications côté serveur !

Expressions régulières

- « *Regex* »
- Langage pour décrire des motifs de chaînes de caractères
- Plusieurs utilisations :
  - S'assurer qu'une chaîne de caractères respecte un format précis
  - Extraire des éléments d'une chaîne de caractères
  - Remplacer des éléments d'une chaîne de caractères
- Pas propre à PHP, utilisable dans tous les langages de programmation (ou simplement avec la commande Linux `sed`)
- Site web pour tester ses regex : https://regex101.com/

Exemples

```php
if (preg_match("#^0[1-9]([-. ]?[0-9]{2}){4}$#", $_POST["tel"]))
{
    echo "C'est bien un numéro de téléphone !";
}

// convertit "**foo**" en "<b>foo</b>" (à la Markdown) :
$comment = preg_replace(
    "/\*\*(.+)\*\*/msU",
    "<b>$1</b>",
    $_POST["comment"]
);
```

Données de l'utilisateur dangereuses : faille XSS

- *Cross-Site Scripting*
- L'internaute envoie des données qui contiennent du code que le navigateur va interpréter
  - Balises HTML
  - Plus dangereux : code JavaScript

Exemple

Validez le formulaire pour envoyer un commentaire avec les valeurs suivantes :

- Pseudo : `<b>Seul en gras</b>`
- Adresse mail : ce que vous voulez (on ne l'affiche pas)
- Commentaire :
  `Blabla <script>alert("Boum");</script>blabla`

Données de l'utilisateur dangereuses : faille XSS

Protection

Utilisez la fonction `htmlentities()` qui convertit tous les caractères qui ont un équivalent en entités HTML

- Exemple : < → `<`

Question : faut-il faire
`nl2br(htmlentities($_POST["comment"]))`
ou
`htmlentities(nl2br($_POST["comment"]))` ?

Cookies

- Couples clé/valeur que le serveur peut stocker dans le navigateur, pour une durée déterminée
- Permet de conserver des informations lors des changements de pages
- Client : donc aucune confiance ! Depuis le navigateur, on peut :
  - Afficher les cookies
  - Créer des cookies
  - Modifier leurs valeurs
  - Les supprimer
- D'un point de vue HTTP :
  - le serveur envoie un en-tête dans la réponse : `SetCookie: PSEUDO=toto;`
  - le navigateur inclut dans chaque requête un en-tête qui contient les cookies qu'il connaît pour ce site : `Cookie: PSEUDO=toto;`

Cookies en PHP

- Créer un cookie :

```php
setcookie("pseudo", "toto", time() + 7*24*3600);
```

  - 3ème paramètre : date d'expiration du cookie
  - Doit être appelée avant que du contenu de la réponse ne soit généré

- Accéder à un cookie :

```php
// tester l'existence, sécuriser la valeur, etc
echo $_COOKIE["pseudo"];
```

- Modifier la valeur d'un cookie :

```php
setcookie("pseudo", "titi", time() + 7*24*3600);
```

- Supprimer un cookie :

```php
setcookie("pseudo", "", 0);
```

Sessions

- Similaires aux cookies, mais sont uniquement sur le serveur
- Utilisation : retenir que l'internaute est connecté, retenir son panier d'achat, *etc*

Fonctionnement

- Pour chaque client, PHP génére un identifiant unique
- Cet identifant est transmis au client par un cookie
- Lors de la prochaine requête du client, le serveur récuperera l'identifiant du client dans le cookie et pourra récupérer les données associées à cet identifiant
- Plusieurs techniques possibles pour stocker les sessions :
  - Dans la mémoire du processus du serveur
  - Dans une base de données
  - Dans une base de données non persistente (Redis, memcached, ...)

Sessions en PHP

- Initilisation du système de sessions :

```php
session_start();
```

  - Doit être appelée avant que du contenu de la réponse ne soit généré
  - À appeler sur toutes les pages qui vont accéder aux sessions

- On manipule ensuite `$_SESSION` comme n'importe quel tableau associatif :

```php
$_SESSION["pseudo"] = "toto";
echo $_SESSION["pseudo"];
```

Instructions pour aider au déverminage

- Activer l'affichage des erreurs et des avertissements (*cf* plus haut)
- Afficher tout le contenu d'une variable, d'un tableau, d'un objet :
  - Fonctions `var_dump()` et `print_r()`
  - Appeler ces fonctions dans des balises `<pre>` génére un meilleur rendu :

```
echo "<pre>";
print_r($t);
echo "</pre>";
```

- Des débuggueurs existent : par exemple Xdebug
- Servez-vous de l'onglet *Réseau* de votre navigateur

Exercice : le jeu du plus ou moins dans le navigateur

Faites une page PHP qui permet de jouer au jeu du plus ou moins.

1. Premier chargement de la page : PHP choisit un nombre entre 1 et 100 (fonction `rand()`). Un formulaire est affiché pour deviner le nombre.
2. Les données du formulaire sont envoyées à la même page :
   - si les données sont invalides, on affiche un message d'erreur en plus du formulaire
   - si le nombre essayé n'est pas le nombre à trouver, on affiche si c'est plus ou moins
   - si le bon nombre est trouvé, on affiche un message de victoire contenant le nombre d'essais pour trouver le bon nombre et un lien pour faire une nouvelle partie

Bonus

- Pouvoir abondonner une partie en cours
- Ajouter une page qui permet de démarrer une partie en choisissant le nombre maximal possible
- Afficher au fur et à mesure les nombres essayés

À vous de jouer !

Ressources

- La documentation officielle : https://www.php.net/manual/fr/
- Une synthèse du langage : https://learnxinyminutes.com/docs/fr-fr/php-fr/
- Grafikart : https://grafikart.fr/formations/php

Base de données relationnelles avec SQL

Comment stocker des données ?

- Directement dans de simples fichiers textes ?

Mais...

- Comment stocker (efficacement) des données structurées ?
- Comment accéder, ajouter, éditer, chercher, supprimer (toujours efficacement) des données structurées ?
- *Efficacement* : le plus rapidement possible, en utilisant le moins de mémoire possible
- Comment gérer les accès concurrents ?

Bases de données

⇒ Utilisation de **bases de données**

- On s'intéresse ici aux base de données *relationnelles*
  - Données structurées qui ont des liens entre elles
  - « *Tableaux avec des lignes et des colonnes* »

⇒ Cette partie n'est pas propre à la programmation web !

Exemple

Articles

ID	Titre	Contenu	Auteur	Date
...	...	...	...	...
...	...	...	...	...

Commentaires

ID	Commentaire	Auteur	Date	Article ID
...	...	...	...	...
...	...	...	...	...

SGBD

- *Système de Gestion de Bases de Données*
- Logiciel utilisé pour interagir avec une base données
- Suit (généralement) le modèle client/serveur
- Langage spécifique pour parler au serveur

Les plus connus :

- MySQL / **MariaDB**
- PostgreSQL
- Oracle
- Microsoft SQL Server
- SQLite

Connexion à un SGBD

$\Rightarrow$ quel client utiliser ?

- Dans le terminal : commande `mysql`
- Une interface web dédiée : phpMyAdmin, Adminer, ...
- Un logiciel dédié : MySQL Workbench, Sequel Pro, ...
- Via une API dans un langage de programmation

Informations à fournir :

- Adresse du serveur
- Port (si pas celui par défaut)
- Nom d'utilisateur
- Mot de passe
- Nom de la base de données à utiliser (optionnel)

Un SGBD peut gérer plusieurs bases de données simultanément !

SQL

- *Structured Query Language*
- Langage qui permet de formuler les requêtes adressées à SGBD
- Dialectes différents selon les SGBD pour les fonctionnalités avancées

Exemple

« *Donne-moi tous les articles dont* `Toto` *est l'auteur* » :

```
SELECT * FROM articles WHERE author="Toto";
```

- Dans cette partie : langage SQL
- Partie suivante : utilisation de SQL depuis du code PHP

SQL à l'ENSEIRB-MATMECA

Nécessite d'activer ses pages web, comme pour PHP.

Une interface phpMyAdmin est disponible à l'adresse
https://zzz.bordeaux-inp.fr/aaa/phpmyadmin/

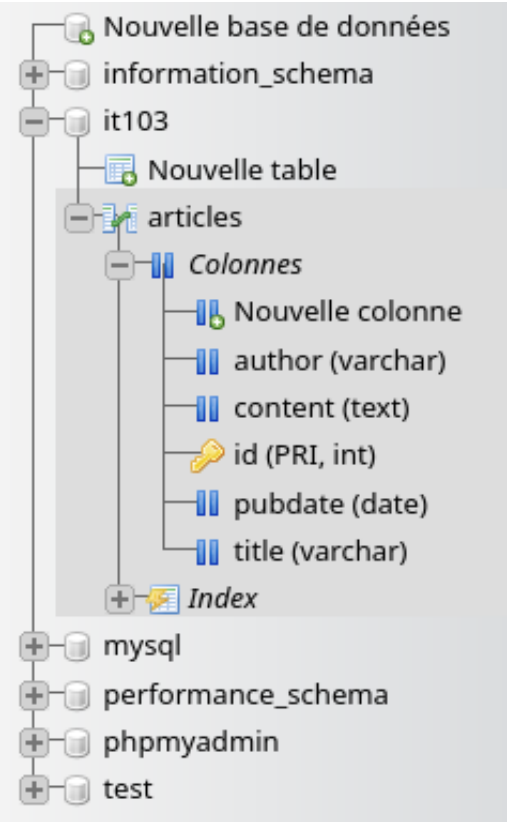
- Mot de passe que vous avez défini lors de l'activation de vos pages web
- Différent de votre de mot de passe pour copier les fichiers et du mot de passe CAS !

SQL sur votre machine

- Avec XAMPP, une interface phpMyAdmin est accessible à l'adresse `http://localhost/phpmyadmin/`

Structure d'une base données

- Un **SGBD** gère plusieurs **bases de données**
- Une **base de données** contient plusieurs **tables**
- Chaque **table** a des colonnes définies et on peut y ajouter des lignes
- Les **lignes** des tables sont les **données** : une ligne = une entité
  - Par exemple : un article est une ligne de la table `article`



Structure d'une table

Une table possède :

- un **nom**
- des **colonnes**

Chaque colonne possède :

- un **nom**
- un **type de données** :
  - `TEXT` pour de longs textes
  - `VARCHAR` pour du texte court (il faut préciser la taille limite)
  - `INT`
  - `DATE`
  - ...
- des **attributs** :
  - Valeur par défaut
  - Est-ce que la valeur `NULL` est permise
  - Incrémentation automatique
  - ...

Pourquoi une colonne `ID` ?

- Contient un entier pour identifier de façon unique la ligne
- Servira dans les relations avec les autres tables
  - Ce commentaire concerne quel article ?
- Servira comme paramètre dans de nombres URL !
  - Je souhaite voir l'article avec tel `ID`
- Appelé **clé primaire**

Fonctionnalités SQL

- Possibilité de définir une colonne comme clé primaire
- Support de l'incrémentation automatique : le serveur SQL se chargera de définir la valeur d'ID de chaque nouvelle ligne

Création de la table pour stocker les articles

```
CREATE TABLE articles (
  id int(11) NOT NULL,
  title varchar(255) NOT NULL,
  content text NOT NULL,
  author varchar(255) NOT NULL,
  pubdate date NOT NULL
);

ALTER TABLE articles
  ADD PRIMARY KEY (id);

ALTER TABLE articles
  MODIFY id int(11) NOT NULL
       AUTO_INCREMENT;
```

- `CREATE TABLE` pour créer une table
- `ALTER TABLE` pour modifier la structure d'une table
- Une requête SQL termine par un point virgule

phpMyAdmin propose un beau formulaire pour créer des tables

Stockage de l'auteur

Imaginons :

- On souhaite stocker le pseudo, le nom, le prénom, l'adresse mail de l'auteur d'un article
  - $\Rightarrow$ Ajoutons les colonnes nécessaires à la table `articles` !
- Un auteur souhaite maintenant changer son adresse mail :
  - $\Rightarrow$ Il faut modifier son adresse mail pour tous ses articles ! ☹
- On souhaite stocker les mêmes informations pour les commentaires. Un auteur d'article peut aussi commenter des articles.
  - $\Rightarrow$ Les informations vont être dupliquées dans la base de données ! ☹

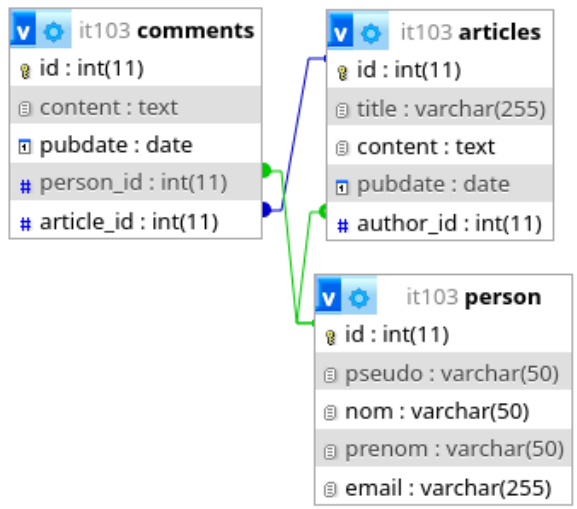
Que faire pour stocker les informations des auteurs ?

La solution : les clés étrangères

- Créer une table `personnes` qui contient toutes les informations propres à une personne
- Chaque personne aura un ID (clé primaire)
- Modifier la table `articles` pour avoir une colonne contenant l'ID de l'auteur dans la table `personnes`

Fonctionnalités SQL

- La colonne contenant l'ID de l'auteur dans la table `articles` peut être définie comme **clé étrangère**
- Le SGBD s'assure que les données sont cohérentes : l'ID correspond bien à un auteur existant, par exemple
- On peut configurer le comportement du SGBD si l'auteur est supprimé de la table `auteurs` :
  - supprimer ses articles ?
  - autoriser des articles sans auteurs ?
  - remplacer l'ID par une valeur par défaut ?



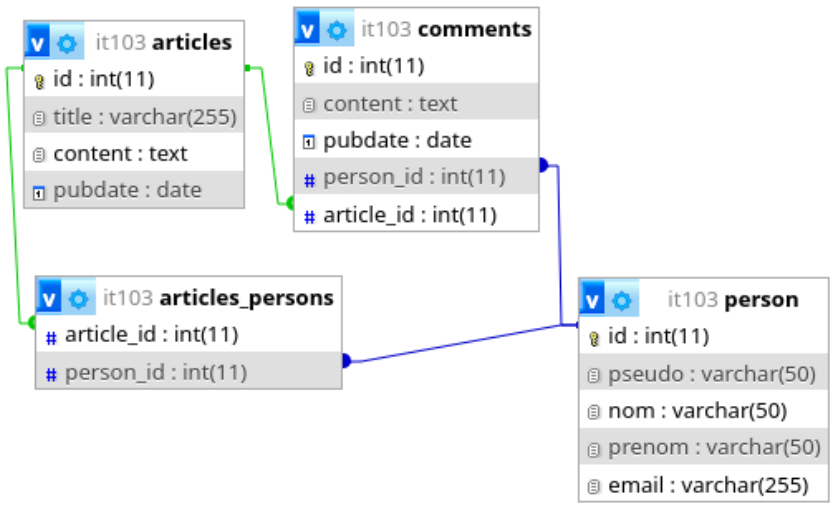
Stockage des auteurs

- Maintenant, un article peut être écrit par plusieurs auteurs !

Comment stocker la relation entre articles et auteurs ?

Solution : une table intermédiaire

- Table intermédiaire `articles_persons` avec deux colonnes :
  - ID de l'article
  - ID de la personne
- La table `articles` n'a plus de colonne pour l'ID de l'auteur



- Relation *Many to many*
- Un `X` peut avoir plusieurs `Y`, mais un `Y` peut aussi avoir plusieurs `X`
  - Exemple : `X` = article et `Y` = catégorie

Exercice : Marmit'eirb

Décrivez sur une feuille de papier le schéma de la base de données d'une application de partage de recettes de cuisine :

- chaque recette possède un titre, un texte, un temps de cuisson, des catégories (entrée, plat principal, dessert, potage, pâtisserie, ...), un auteur, des ingrédients et leurs quantités respectives
- on peut commenter les recettes
- on peut indiquer les recettes qu'on a réalisées
- on peut créer des menus : ensemble de recettes
- ... et laissez libre cours à votre imagination pour d'autres fonctionnalités !

À vous de jouer !

Instructions SQL pour la manipulation des tables

- Toutes les actions de manipulation des tables (création, modification, suppression, gestion des clés étrangères, ...) sont faisables en SQL
- `CREATE TABLE`, `ALTER TABLE`, ...
- phpMyAdmin transforme vos actions dans l'interface en requêtes SQL
- Dans la vraie vie, des outils génèrent ces requêtes pour vous

$\Rightarrow$ on va seulement voir les requêtes pour manipuler les données

Remarques sur le code SQL

Commentaires

```
-- Commentaire sur une ligne

/* Commentaire
   sur plusieurs
   lignes */
```

Style de code

- Pas une obligation, mais on a tendance à écrire les mots-clés SQL tout en majuscules

Code SQL dans un fichier

- Ce sera rarement le cas, mais il est possible de mettre du code SQL dans un fichier et ensuite exécuter ce code :

```
mysql -u too -p database < code.sql
```

Ajout de données

```
INSERT INTO articles
(id, title, content, pubdate, author_id)
VALUES (12, "Vive les pandas", "Lorem ipsum...",
    "2024-02-12", 15);
```

- Pas besoin de préciser le nom des colonnes si les valeurs sont dans l'ordre des colonnes
- Si la colonne `id` est configurée pour être incrémentée automatiquement : pas besoin de préciser sa valeur, ou bien mettre la valeur `NULL`

Sélection de données

- Récupérer toutes les colonnes et toutes les lignes :

```
SELECT * FROM articles;
```

- Récupérer seulement certaines colonnes et toutes les lignes :

```
SELECT title, content FROM articles;
```

Sélection de données - Restriction

Récupérer seulement les lignes correspondant à un critère :

```
SELECT * FROM articles WHERE author_id = 15;
```

Critères possibles :

- `WHERE author_id = 15 AND pubdate >= '2024-01-01'`
- `WHERE author_id = 15 OR author_id = 16`
- `WHERE author_id IN (15, 16)`
- `WHERE author_id != 15`
- `WHERE content IS NOT NULL`
- `WHERE title LIKE "%Pandas%"`
- ... et d'autres opérateurs : `BETWEEN`, ...

Sélection de données - Ordre

Ordonner les résultats par date de publication croissante :

```
SELECT * FROM articles ORDER BY pubdate;
```

- `ORDER BY pubdate DESC` : par date de publication décroissante
- `ORDER BY pubdate DESC, title` : par date de publication décroissante et en cas d'égalité par ordre lexicographique du titre

Sélection de données - Nombre de résultats

Récupérer seulement les 10 premiers articles :

```
SELECT * FROM articles LIMIT 0, 10;
```

- Le premier nombre indique à partir de quelle ligne parmi tous les résultats on commence à récupérer les lignes
- Le deuxième nombre indique combien de lignes on souhaite récupérer au maximum

- `LIMIT 5, 10` : de la sixième à la quinzième entrée
- `LIMIT 10, 2` : onzième et douzième entrées

Très utile pour gérer les paginations !

Combinaison des options

Attention ! L'ordre des clauses doit être le suivant :

1. `WHERE`
2. `ORDER BY`
3. `LIMIT`

Les 10 derniers articles les plus récents (les plus récents en premier) publiés par la personne avec l'ID 15 :

```
SELECT * FROM articles
WHERE author_id = 15
ORDER BY pubdate DESC
LIMIT 0, 10;
```

Modification de données

```
UPDATE articles
SET
    titre = "Pas ouf les pandas",
    pubdate = "2024-03-02"
WHERE id = 12;
```

Attention ! Sans clause `WHERE`, toutes les entrées de la table `articles` sont modifiées !

Suppression de données

```
DROP FROM articles WHERE id = 12;
```

Attention ! Sans clause `WHERE`, toutes les entrées de la table `articles` sont supprimées !

Fonctions

```
SELECT
    UPPER(nom) AS nom_upper,
    prenom,
    pseudo,
    email
FROM person;
```

Les résultats aurant une colonne nommée `nom_upper` avec les noms de personnes tout en majuscules.

Quelques fonctions :

- `LOWER()` : convertit tout en minuscules
- `LENGTH()` : donne le nombre de caractères
- `ROUND()` : arrondit un nombre flottant
- `RAND()` : renvoie un nombre aléatoire
- ...

Aggrégations

- Appliquer une fonction qui prend plusieurs valeurs (venant de lignes différentes) et renvoie une unique valeur

Obtenir le nombre de caractères moyen de tous les articles :

```
SELECT
    AVG(LENGTH(content)) AS longueur_moyenne
FROM articles;
```

Quelques fonctions d'aggrégation :

- `SUM()`
- `MIN()` : donne le nombre de caractères
- `MAX()` : arrondit un nombre flottant
- `COUNT()`
- ...

Attention ! On ne peut pas sélectionner d'autres colonnes non-aggrégées

Grouper les données pour les aggréger

Pour obtenir la longueur moyenne des articles de chaque auteur :

```
SELECT
    AVG(LENGTH(content)) AS longueur_moyenne,
    author_id
FROM articles
GROUP BY author_id;
```

Quelques précisions sur `COUNT()`

- Compter le nombre de lignes dans la table :

```
SELECT COUNT(*) AS nb FROM articles;
```

- Compter le nombre d'articles écrit par la personne avec l'ID 15 :

```
SELECT COUNT(*) AS nb FROM articles WHERE
    author_id = 15;
```

- Obtenir le nombre d'articles écrits par chaque auteur :

```
SELECT COUNT(*) AS nb, author_id FROM articles
    GROUP BY author_id;
```

- Compter le nombre de valeurs **non nulles** dans une colonne :

```
SELECT COUNT(article_id) AS nb_authors FROM
    articles;
```

- Obtenir le nombre de personnes ayant écrit au moins un article :

```
SELECT COUNT(DISTINCT author_id) AS nb FROM
    articles;
```

Filtres sur les aggrégations

- `WHERE` filtre les lignes avant de les aggréger
- `HAVING` filtre les lignes résultant de l'aggrégation

Sélectionner les IDs des auteurs ayant écrit au moins 10 articles :

```
SELECT COUNT(*) AS nb, author_id
FROM articles
GROUP BY author_id
HAVING nb >= 10;
```

Sélectionner les IDs des auteurs ayant écrit au moins 10 articles en 2024 :

```
SELECT COUNT(*) AS nb, author_id
FROM articles
WHERE YEAR(pubdate) = 2024
GROUP BY author_id
HAVING nb >= 10;
```

Jointures

- Permettent de récupérer dans une unique requête des données provenant de plusieurs tables

Exemple

- Repartons de l'exemple avec deux tables `comments` et `persons`
- La table `comments` a une clé étrangère vers la table `persons` pour savoir qui a écrit chaque commentaire

$\Rightarrow$ Jointure pour récupèrer en une seule requête les commentaires et toutes les informations sur leurs auteurs

- Les colonnes `comments.person_id` et `persons.id` permettent de faire correspondre les lignes des deux tables

Deux types de jointures

- Jointures **internes** : seules les données qui ont une correspondance entre les deux tables sont renvoyées
- Jointures **externes** : toutes les données sont sélectionnées, même s'il n'y a pas de correspondance avec les données de l'autre table

Jointures internes

Récupérer tous les commentaires de l'article 15 avec le pseudo et l'ID de la personne qui a écrit le commentaire, trié par ordre chronologique :

```
SELECT c.id, c.content, c.pubdate, p.id, p.pseudo
FROM comments c
INNER JOIN persons p
ON p.id = c.person_id
WHERE c.article_id = 15
ORDER BY c.pubdate;
```

Il est possible de faire plusieurs jointures par requête !

Jointures externes

- Même construction que pour `INNER JOIN`
- La table dans le `FROM` est la table de *gauche*
- La jointure est faite sur la table de *droite*
- `LEFT JOIN` récupère toutes les données de la table de gauche, même sans correspondance dans la table de droite
- `RIGHT JOIN` récupère toutes les données de la table de droite, même sans correspondance dans la table de gauche
- L'absence de correspondance est indiquée par la valeur `NULL`

Pour aller plus loin...

- **Vues** : tables virtuelles qui stockent le résultat d'une requête et peuvent être utilisées dans d'autres requêtes
- **Fonctions** : écrire vos propres fonctions SQL
- **Déclencheurs** (*triggers*) : exécuter des requêtes SQL lors d'événements
  - Exemple : mettre à jour automatiquement un champ dans une table lors de l'ajout d'une entrée dans une autre table
- **Fonctions de fenêtrage** : faire des aggrégations sans fusionner les lignes groupées

Autres types de bases de données

- Bases de données pour séries temporelles
  - InfluxDB
- Bases de données pour graphes
  - Neo4j
- Bases de données orientées documents
  - MongoDB

Ressources

- Le cours de Sylvain Lombardy : https://slombardy.zzz.bordeaux-inp.fr/ens/sgbd/
- https://sql.sh/
- https://gvwilson.github.io/sql-tutorial/
- Grafikart : https://grafikart.fr/formations/apprendre-sql
- La documentation du SGBD que vous utilisez

TP : les bases de données font leur cinéma

À vous de jouer !

PHP et SQL

SQL avec PHP

- API différente selon le SGBD utilisé
- API `PDO` pour avoir une API unique, mais seulement disponible en orienté objet
- Utilisation de MySQL / MariaDB ⇒ utilisation de l'interface `mysqli`

`https://www.php.net/manual/fr/book.mysqli.php`
(regardez les parties *Style procédural*)

Connexion

```php
$db = mysqli_connect($address, $user, $password, $db);
if (mysqli_connect_errno()) {
    echo "Erreur de la connexion à la BDD : ";
    echo mysqli_connect_error();
    die;
}
```

- `$address` : l'adresse du serveur MariaDB
  - `"localhost"`
- `$user` : le nom d'utilisateur pour se connecter
  - sur votre machine : `"root"`
  - sur le serveur de l'ENSEIRB : votre login
- `$password` : le mot de passe pour se connecter
  - sur votre machine : `""`
  - sur le serveur de l'ENSEIRB : le mot de passe que vous avez renseigné lors de l'activation de votre espace web
- `$db` : la base de données que vous allez utiliser
  - sur votre machine : le nom que vous avez choisi pour votre base de données
  - sur le serveur de l'ENSEIRB : votre login

Comment stocker les informations d'authentification ?

Le problème

- Utilisation de deux serveurs MariaDB différents :
  - En local sur votre machine, pour le développement
  - Sur le serveur de production

  $\Rightarrow$ Informations de connexion différentes
- **On ne veut pas qu'un mot de passe soit stocké dans Git !**

Une solution

- Créer un fichier `config.php` qui contient les informations de connexion :

```php
$db_address = "localhost";
$db_user = "root";
$db_password = "";
$db_db = "it103";
```

- Inclure ce fichier avant d'appeler `mysqli_connect()` :

```php
include("config.php");
$db = mysqli_connect($db_address, $db_user, $db_password, $db_db);
```

- Ne pas versionner `config.php` : l'inclure au fichier `.gitignore`
- Versionner un fichier `config.dist.php` qui est un modèle pour le fichier `config.php`

Exécuter une requête sans résultat

```php
$req = "INSERT INTO articles VALUES (NULL, 'Vive
    les pandas', 'Lorem ipsum', '2023-12-25')";
if (mysqli_query($db, $req))
{
    echo "Requête réussie";
}
else
{
    echo "Requête râtée";
}
```

Exécuter une requête et afficher les résultats

```php
$sql = 'SELECT * FROM articles ORDER BY pubdate DESC';
$articles = mysqli_query($db, $sql);

echo "<ul>";
while ($article = mysqli_fetch_assoc($articles))
{
    echo "<li>";
    echo "#" . $article["id"] . " - ";
    echo $article["title"];
    echo "</li>";
}
echo "</ul>";
mysqli_free_result($articles);
```

Exécuter une requête avec un unique résultat

```php
$sql = "SELECT COUNT(*) AS nb FROM articles";
$query = mysqli_query($db, $sql);
$nb_articles = mysqli_fetch_assoc($query);
echo $nb_articles["nb"];
mysqli_free_result($query);
```

Exécuter une requête paramétrée

Imaginons :

- Les articles ont un attribut booléen `published` pour savoir si l'article est déjà public ou pas
- On a un formulaire qui permet de chercher les articles publiés selon leur titre

```
$sql = 'SELECT * FROM article ';
$sql .= 'WHERE published = 1 AND ';
$sql .= 'title LIKE "%' . $_GET["q"] . '%"';
```

Quel pourrait être le problème ?

Injections SQL

$\Rightarrow$ Injection SQL possible

Quels sont les articles renvoyés si je cherche les articles dont le titre contient `" OR 1=1 --` ?

Injections SQL

⇒ Injection SQL possible

Quels sont les articles renvoyés si je cherche les articles dont le titre contient `" OR 1=1 --` ?

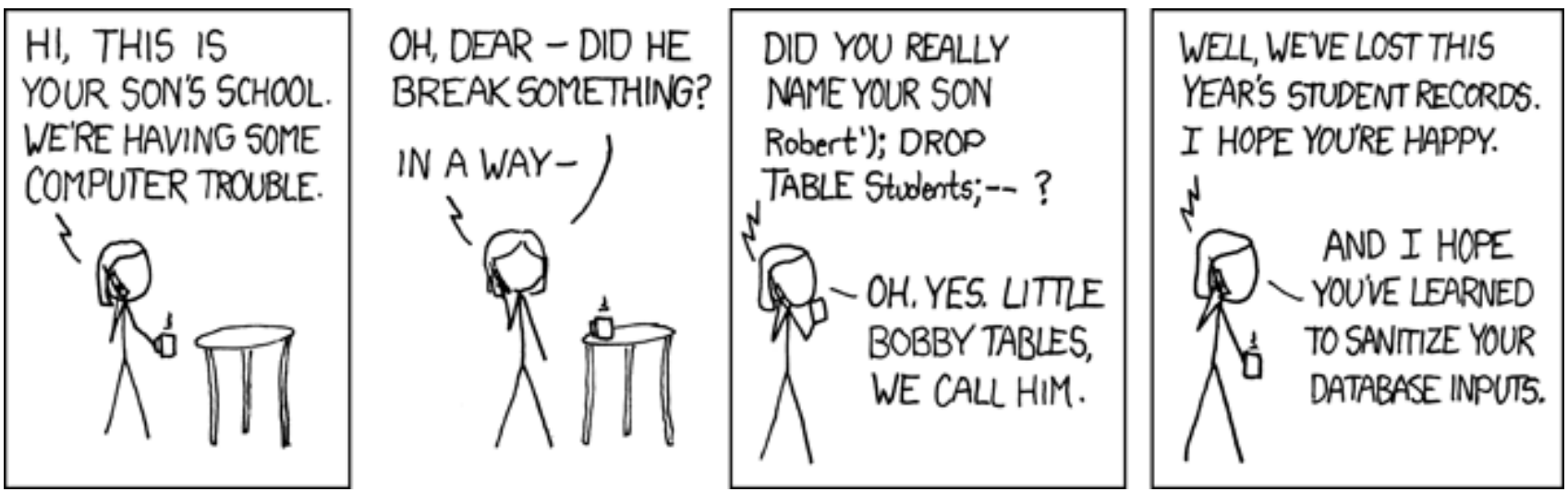
- Tous les articles de la table sont renvoyés
- Pourrait être pire : suppression des tables, ajout d'un utilisateur avec les droits d'administration, *etc*
- **C'est une faille de sécurité !**

Injections SQL

⇒ Injection SQL possible

Quels sont les articles renvoyés si je cherche les articles dont le titre contient `" OR 1=1 --` ?

- Tous les articles de la table sont renvoyés
- Pourrait être pire : suppression des tables, ajout d'un utilisateur avec les droits d'administration, *etc*
- **C'est une faille de sécurité !**



https://xkcd.com/327/

Se protéger des injections SQL : les requêtes préparées

```php
$sql = "SELECT * FROM article WHERE published = 1 AND title LIKE ?";
$req_pre = mysqli_prepare($db, $sql);
$param = '%' . $_GET["q"] . '%';
mysqli_stmt_bind_param($req_pre, "s", $param);
mysqli_stmt_execute($req_pre);
mysqli_stmt_bind_result($req_pre, $id, $title, $published);

echo "<ul>";
while (mysqli_stmt_fetch($req_pre))
{
    echo "<li>#" . $id . " - " . $title . "</li>";
}
echo "</ul>";
mysqli_stmt_close($req_pre);
```

- Dans la requêtes les paramètres sont indiqués par des ?
- Appel à `mysqli_stmt_bind_param()` une fois pour tous les paramètres :
  - `"s"` : le paramètre est une chaîne de caractères
  - `"i"` : le paramètre est un nombre entier
  - `"d"` : le paramètre est un nombre flottant

```php
mysqli_stmt_bind_param($req_pre, "si", $str, $nb);
```

- `mysqli_stmt_bind_result()` pour indiquer dans quelles variables mettre les résultats

Exercice : stocker les commentaires en base de données

1. Créez une table pour les commentaires
   - On souhaite stocker la date et l'heure du commentaire

   $\Rightarrow$ colonne de type `DATETIME`
2. Ajoutez quelques commentaires dans la table avec phpMyAdmin
3. Modifiez la page de votre blog pour qu'elle affiche les commentaires qui sont enregistrés en base de données
4. Modifiez la page de votre blog pour que les commentaires envoyés avec le formulaire soit sauvegardés en base de données
   - Astuce : la fonction SQL `NOW()` donne la date et l'heure actuelle

À vous de jouer !

Stocker des mots de passe en base de données

Comment stocker des mots de passe d'utilisateurs dans une base de données ?

- En *clair* (on peut connaître le mot de passe en regardant le contenu de la table) ?
- Quels sont les risques ?

Stocker des mots de passe en base de données

Comment stocker des mots de passe d'utilisateurs dans une base de données ?

- En *clair* (on peut connaître le mot de passe en regardant le contenu de la table) ?
- Quels sont les risques ?

Solution

⇒ Socker des mots de passe *hachés*

- On stocke le résultat d'une fonction de hachage appliquée au mot de passe en clair lors de la création du mot de passe
- On hache le mot de passe saisi et on compare à ce qui est base de données lors de l'authentification
- `https://www.php.net/manual/fr/faq.passwords.php`
- Est-il encore possible de dérober le mot de passe ?

Conclusion sur PHP : pour aller plus loin

- Programmation orientée objet en PHP
- Organisation du code en suivant l'architecture MVC (**M**odèle **V**ue **C**ontrôleur)
- Installation, gestion et utilisation de bibliothèques PHP
  - Avec `Composer`, par exemple

Conclusion sur PHP : dans la vraie vie

- Faire une application web est difficile et répétitif :
  - Organisation propre du code
  - S'assurer que son application n'a pas de faille de sécurité
  - Éviter de réinventer la roue en permanence
  - ...

⇒ Utilisation de *frameworks* (« *cadriciels* ») :

- Bibliothèque qui inclut beaucoup de fonctionnalités prêtes à l'emploi
- Utilisation généralement d'un ORM (*object-relational mapping*) pour abstraire l'accès à la base de données : vous n'avez plus besoin d'écrire de SQL !
- Gestion des formulaires, des URLs, des fichiers statiques, des erreurs, des sessions, des cookies, de l'inscription et la connexion, ...
- Utilisation généralement d'un moteur de templates : vous n'écrivez plus de PHP au milieu de HTML, mais encore un autre langage
- Exemples : Symfony, Laravel, CakePHP, CodeIgniter, ...

Pages dynamiques côté client avec JavaScript

Présentation de JavaScript

- **Permet l'exécution d'instructions dans le navigateur**
- Naissance en 1995, intégration à Netscape (ancêtre de Firefox)
- Standardisation par l'Ecma sous le nom d'ECMAScript
  - EC6 : version 6 du standard du langage
- Langage impératif, orienté objet, interprété, faiblement typé
- Utilisation possible aussi côté serveur
  - Avec Node.js qui joue alors le même rôle que PHP
  - Pas vu dans ce cours
- Souvent abbrégé JS, extension de fichier `.js`

Rien à voir avec le langage Java !

Que peut-on faire avec le JavaScript ?

Côté navigateur

- Interagir avec le HTML et le CSS
  - Afficher / masquer des éléments HTML
  - Faire des animations
  - Changer des propriétés CSS
  - Réagir à des événements
  - ...
- Faire des requêtes HTTP sans recharger la page
  - Valider un formulaire
  - Auto-complétion
  - Recherche instantanée
  - Défilement infini
  - ...

Où mettre du code JavaScript

- Exécution par le navigateur

$\Rightarrow$ Le navigateur doit récupérer le code JavaScript

Où mettre du code JavaScript

- Exécution par le navigateur

$\Rightarrow$ Le navigateur doit récupérer le code JavaScript

Option 1 : Directement dans la page HTML

```
<body>
    <!-- votre code HTML ... -->

    <script>
        alert("Boum !");
    </script>
</body>
```

Où mettre du code JavaScript

Option 2 : Dans un fichier dédié

```
<body>
    <!-- votre code HTML ... -->

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

Fichier `script.js` (enregistré à côté de vos fichiers HTML, CSS, ...) :

```
alert("Boum !");
```

- Permet de factoriser le code (le même fichier JS peut facilement être inclus à différentes pages)
- Le navigateur peut mettre ce fichier en cache : évite de tout le temps télécharger le même fichier

Où mettre du code JavaScript

Où placer la balise <script> ?

- Si le code JavaScript est directement dans la page, le moteur JavaScript peut commencer à exécuter le code avant que le reste de la page soit reçu
  - Problème si le code JS manipule des éléments HTML pas encore chargés
- Si la balise est lien vers un fichier JavaScript, le navigateur va tout de suite charger le fichier JS, ce qui peut bloquer le chargement du reste de la page HTML

⇒ **À la fin du code HTML** (juste avant </body>)

Question

A-t-on besoin d'un serveur dynamqique (comme PHP) pour faire du JS ?
Ou bien peut-on aussi faire du JavaScript avec un site statique ?

L'incontournable *Hello world*

1. Créez une page HTML qui contient le code JS suivant (option 1) :

```
alert("Hello World !");
```

2. Affichez la page avec votre navigateur. Qu'observez-vous ?
3. Déplacez le code JS de la page HTML vers un fichier dédié (option 2). Assurez-vous que vous observez toujours la même chose.

À vous de jouer !

La syntaxe JavaScript : Bases

- Point-virugle après chaque instruction optionnel, mais nécessaire pour séparer deux instructions sur la même ligne :

```
inst1; inst2
inst3
inst4;
```

- JS n'est pas sensible aux espaces et indentations (comme le C)
- Commentaires : même syntaxe qu'en C

La syntaxe JavaScript : Variables

```
let verite = true; // booléen
let age = 24; // entier
let temperature = 15.3; // flottant
// portée des variables avec let : bloc

// ancienne syntaxe : var au lieu de let
var truc = 32;
// portée des variables avec var : fonction

/* Opérations arithmétiques comme en C : */
let resultat = age * temperature - 13;
age++;
age *= 2;

/* Constantes: */
const FOO = 67;
resultat = 67 * FOO;

let valeur_nulle = null;
let valeur_absente; // <=> = undefined;
```

La syntaxe JavaScript : Chaînes de caractères

```
// Trois délimiteurs possibles : ", ', ` :
let texte1 = "Texte A";
let texte2 = 'Texte B';
let texte3 = `Texte B`;

// ` permet aussi les retours à la ligne :
let long_texte = `Lorem
Ipsum`;

// Concaténations :
let phrase = texte1 + " " + texte2;
// Substitutions uniquement avec ` :
let phrase2 = `${texte1} ${texte2}`;
```

Déverminage

- `alert(variable);`
  - Pas pratique pour afficher les objets (`[Object]`)
  - Pas pratique s'il faut y faire appel de nombreuses fois

- La console JavaScript du navigateur
  - Permet d'exécuter du JavaScript
  - Permet d'afficher le contenu de variables
- `console.log(variable);`
  - Affiche le contenu de la variable dans la console JavaScript du navigateur
- `console.table(tab);`
  - Affiche joliment un tableau dans la console JavaScript du navigateur

La syntaxe JavaScript : Conditions

- Comparaison : ==, === (en considérant aussi le type),<, <=, >, >=, !=, !==
- Logique : !var, &&, ||

```
if (a === b)
{
    // ...
}
else if (a === c)
{
    // ...
}
else
{
    // ...
}
```

```
// Condition ternaire :
let truc = machin ? "vrai" : "faux";

// Raccourci pour x ? x : "faux" :
truc = x || "faux";

switch (x) {
    case 0:
        // ...
        break;
    case "un":
        // ...
        break;
    default:
        // ...
}
```

Un langage faiblement typé

Qui donne des résultats parfois suprenants :

```
12 + 1 // 13
12 + "1" // "121"
12 * "2" // 24
12 * 'B' // NaN (Not a Number)

null == 0 // false
null > 0 // false
null >= 0 // true
```

La syntaxe JavaScript : Boucles

```
let i = 0;
let s = 0;
while (i < 12)
{
    s += i;
}

for (let i = 0; i < 12; i++)
{
    s += i;
}

do {
    s += i;
} while (i < 12);
```

- `break` et `continue` fonctionnent comme en C

La syntaxe JavaScript : Tableaux

```
let tableau = ["pg109", "pg110"];
// peut contenir différents types d'éléments
console.log(tableau[0]);
tableau[1] = "PG110";
tableau.push("it103");

for (let i = 0; i < tableau.length; i++)
{
    console.log(tableau[i]);
}

for (let cours of tableau)
{
    console.log(cours);
}
```

La syntaxe JavaScript : Objets

Équivalent des structures en C :

```
let personne = { prenom: "Toto", age: 17 };
console.log(personne.prenom);
console.log(personne["prenom"]);
personne.age++;
```

Utilisation possible comme tableau associatif (voir aussi `Map()` depuis ES6) :

```
let cours = {
    "pg109": "C",
    "pg110": "Projet"
};
cours["it103"] = "Web";

for (let c in cours)
{
    console.log(cours[c]);
}
```

La syntaxe JavaScript : Fonctions

```
function truc(param1, param2)
{
    // fait des trucs...

    // pas obligatoire si rien à renvoyer
    return result;
}

let r = truc(a, b);
```

La syntaxe JavaScript : Fonctions anonymes

```
function si_pair(nb, f)
{
    if (nb % 2 == 0)
    {
        f(nb);
    }
}

si_pair(6, function(n) {
    // Ceci est une fonction anonyme
    console.log(`${n} est pair`);
});

// Autre syntaxe :
si_pair(6, (n) => {
    // Ceci est une fonction anonyme
    console.log(`${n} est pair`);
});
```

- Les fonctions anonymes existent aussi en PHP

La syntaxe JavaScript : Fonctions d'objet

- Une introduction cachée à la programmation objet...
- Une fonction définie dans un objet est une *méthode*

```
let personne = {
    prenom: "Toto",
    age: 17,
    se_presenter: function() {
        console.log(‘Je m’appelle ${this.prenom}
            et j’ai ${this.age} ans‘);
    }
};
personne.se_presenter();
```

Manipulation du DOM

- *Document Object Model*
- Toute la structure HTML de votre page

Récupérer des éléments HTML

- Utilise les sélecteurs CSS

```
// Renvoie *un* élément (ici celui avec id="foo") :
let el = document.querySelector("#foo");
console.log(el.parentNode);
console.log(el.childNodes);
console.log(el.firstChild);
console.log(el.nextSibling);

// Tous les éléments avec la classe CSS animate :
let elements = document.querySelectorAll(".animate");
elements.forEach(e => {
    console.log(e);
});
```

Manipulation du DOM

Propriétés des éléments HTML

```
console.log(element.innerHTML);
console.log(element.innerText);

console.log(element.getAttribute("href"));
console.log(element.style); // règles CSS
console.log(element.classList); // classes CSS
    appliquées à l'élément

element.setAttribute("href", "http://ph-sw.fr");
element.style.color = "red";
element.classList.add("menu-link");
element.innerHTML = "<p>Foo</p>";
element.innerText = "Bar";
```

Manipulation du DOM

Création d'éléments HTML

```
const newEl = document.createElement("li");
newEl.innerText = "Un dernier élément";

// Ajoute le <li> comme dernier élément du <ul>:
const liste = document.querySelector("ul");
liste.appendChild(liste);

// Directement en écrivant le code HTML :
liste.insertAdjacentHTML(
    "beforebegin",
    "<h3>Une liste</h3>"
);

// Suppression :
liste.firstChild.remove()
```

- D'autres fonctions d'ajout possible (à différents endroits) : `insertBefore()`, `prepend()`, ...

Événements

```
let el = document.querySelector("#foo");
el.addEventListener("click", (event) => {
    console.log(event.target);
    console.log(event.currentTarget);
});
```

- `event.target` : élément sur lequel on a réellement cliqué
- `event.currentTarget` : élément sur lequel on écoute l'événement
- `event.preventDefault()` : empêche le comportement par défaut lors de cet événement
  - Exemple : un clic sur un lien ne doit pas nous mener à la cible du lien

Exemple : événements et formulaires

```
const form = document.querySelector("form");
form.addEventListener("submit", (e) => {
    e.preventDefault();
    const data = new FormData(e.currentTarget);
    console.log(data.get("pseudo"));
});
```

Autres événements :

- Sur les champs de fomulaire :
  - `change` : la valeur dans le champ a changé
  - `focus` : le champ vient de recevoir le focus
  - `blur` : le champ vient de perdre le focus
- Sur n'importe quel élément :
  - `keydown` : une touche est appuyée
- https://developer.mozilla.org/fr/docs/Web/Events

Chaque type d'événement possède des attributs spécifiques (nouvelle valeur, quelle touche est appuyée, ...)

Exercice : afficher ou masquer les commentaires en JavaScript

Reprenez l'exercice avec le lien pour afficher ou masquer les commentaires (repartez de la page sans PHP), mais faites-le en JavaScript : sans recharger la page, le clic sur un bouton affiche ou masque les commentaires et adapte le texte du bouton.

Indication pour masquer ou afficher un élément :

```
el.setAttribute("hidden", "hidden");
el.removeAttribute("hidden");
```

À vous de jouer !

Requêtes HTTP en JavaScript

- Aussi appelé AJAX : *Asynchronous JavaScript And XML*
- Ancienne API : `XMLHttpRequest`
- API actuelle : `fetch`
  - Utilise des *promesses* (qu'on ne verra pas en détails)

Exemple d'envoi de formulaire en JavaScript :

```
form.addEventListener("submit", (e) => {
    e.preventDefault();
    const data = new FormData(e.currentTarget);
    fetch("adresse.php", {
        method: "POST",
        body: data
    }).then(r => {
        if (!r.ok)
            alert("Erreur");
        else
            return r.text();
    }).then(content => {
        console.log("Réponse :", content);
    });
});
```

JSON

- *JavaScript Object Notation*
- Format textuel pour représenter des données, comme XML

JavaScript

- Conversion entre JSON textuel et objets JavaScript :

```
let json_txt = JSON.stringify({foo: "bar"});
// '{"foo":"bar"}'
let data = JSON.parse(json_txt);
```

- Réponse JSON dans un appel `fetch()` :

```
r.json(); // au lieu de r.text()
```

PHP

- Conversion entre JSON textuel et tableaux associatifs avec `json_encode()` et `json_decode()`

Exercice : envoyer un commentaire depuis JS

Reprenez votre code qui stocke les commentaires en base de données.

Réalisez l'envoi du formulaire en JavaScript :

- `fetch()` enverra les données vers une page PHP `publish_comment.php`
- `publish_comment.php` insère le commentaire en base de données et renvoie le code HTML pour afficher le commentaire
  - On n'utilisera pas `NOW()` pour la date du commentaire, mais on la définira en PHP :

```php
$d = date("Y-m-d H:i:s");
```

- Lors de la réception de la réponse côté JavaScript, on insérera dans la page le code HTML reçu avec `insertAdjacentHTML()`.

À vous de jouer !

Quelques réflexions sur l'utilisation du JavaScript

- Il est possible de désactiver JavaScript dans son navigateur !
- Votre site continue-t-il de fonctionner sans JavaScript ?
- Attention à l'accessibilité !
- Certaines choses sont faisables sans JavaScript !
  - Exemple : https://www.htmhell.dev/adventcalendar/2023/2/
- https://www.kryogenix.org/code/browser/everyonehasjs.html

Un passé compliqué pour JavaScript

- Exécution côté navigateur ⇒ le support de JavaScript dépend du bon vouloir des navigateurs
- Pour faire la même chose, code différent selon le navigateur utilisé
  - Rendait le développement JavaScript (très) compliqué !

⇒ Développement de bibliothèques JavaScript qui cachent cette complexité

  - Exemple : jQuery

- Les différences entre navigateurs tendent à se réduire, mais gardez quand même un œil sur `https://caniuse.com`

L'éco-système JavaScript

- Énormément de bibliothèques : `https://www.npmjs.com`
- Installation et gestion des bibliothèques avec `npm` ou `yarn`

- Projet avec beaucoup de fichiers JavaScript :
  - Outils pour concaténer tous les fichiers en un (fait un gros échange HTTP, plutôt que de nombreux petits échanges)
  - Outils pour minimiser le code JS
  - D'autres outils pour gérer tous ces outils : Webpack, Vite, ...

TypeScript

- Dialecte de JavaScript développé par Microsoft
- Est *transpilé* en JavaScript
- Fonctionnalités supplémentaires :
  - Typage plus fort
  - Types génériques
  - Notions orientées objet plus développées
  - Énumérations
  - ...
- Utilisé par certaines bibliothèques

Single Page Application

- Abrégé *SPA*
- Toute votre application web est sur **une** page HTML
- JavaScript s'occupe de changer le code HTML de la page
- Exemples de bibliothèques : React, Vue.js, Angular
- Accessibilité ? Référencement ?

Conclusion sur JavaScript : pour aller plus loin

- Organisation du code en modules
- Programmation orientée objet
- Gestion des erreurs avec les exceptions
- Les promesses
- Stockage local dans le navigateur
- La problèmatique CORS (*Cross-Origin Resource Sharing*) avec `fetch()`
- Utilisation de bibliothèques
- Utilisation côté serveur avec Node.js

Ressources

- https://developer.mozilla.org/fr/docs/Web/JavaScript
- Une synthèse du langage : https://learnxinyminutes.com/docs/fr-fr/javascript-fr/
- Grafikart : https://grafikart.fr/formations/formation-javascript

Conclusion

Félicitations !

- Vous connaissez maintenant tous les concepts pour développer des applications web
- Les concepts généraux sont les mêmes quels que soient les langages utilisés
- Dans la vraie vie : utilisation de *frameworks*
- Peut aussi permettre de développer des applications mobiles