Les langages du Web

Table des matières

Objectifs

- Comprendre la structure des éléments d'une page web ;
- Comprendre les concepts d'un site dynamique.

Contexte

Durée : 2h

Environnement de travail : Repl.it

Pré-requis : Connaître le fonctionnement d'Internet et ses protocoles.

[cf. 7s0txEuY]

E-commerce, médias sociaux, enseignement, outils collaboratifs, etc., une grande partie des services que nous utilisons aujourd'hui ont une existence numérique. Les applications web sont devenues un élément crucial pour la plupart de nos activités personnelles et professionnelles.

Historiquement le web était d'abord un espace de diffusion de contenu. Mais dans les années 2000 avec le développement des langages de programmation web, comme JavaScript ou PHP, il est devenu un espace applicatif à part entière. On ne lit plus seulement sur le web, on fait.

Ce module a pour objectif de présenter quelques-uns des langages principaux sur lesquels reposent ces applications web.

- Nous présenterons tout d'abord l'architecture d'une application web, à travers les concepts de front-end et de back-end.
- Nous introduirons ensuite les langages de base des pages web : le HTML et le CSS.
- Enfin, nous aborderons le JavaScript et le PHP, deux langages de programmation qui permettent de passer des contenus statiques aux applications interactives.

L'architecture 3 tiers

[cf. XUe143FN]

Objectifs

- Comprendre la communication client-serveur ;
- Parler à une base de données.

Mise en situation

Le fonctionnement d'un site web repose sur l'utilisation de plusieurs logiciels fonctionnant sur des ordinateurs distants, qui jouent chacun leur rôle.

1. Le navigateur s'exécute sur la machine cliente. Il envoie une requête HTTP à un serveur web, qui lui renvoie en retour une page HTML.
2. Mais dans le cas d'un site dynamique, cette page n'existe pas en tant que telle sur le serveur web, elle est construite à la volée par un serveur applicatif. Celui-ci exécute un programme dans un langage tel que PHP pour construire la page web à retourner au moment ou elle est demandée.
3. Souvent, ce serveur applicatif a lui-même recours à un autre serveur, le serveur de données. Il peut lui demander l'état des données à un instant t, comme par exemple le contenu d'un catalogue ; ou lui donner l'ordre de modifier ces données, par exemple en réduisant le stock disponible pour un produit qui vient d'être vendu.

Modèle client-serveur

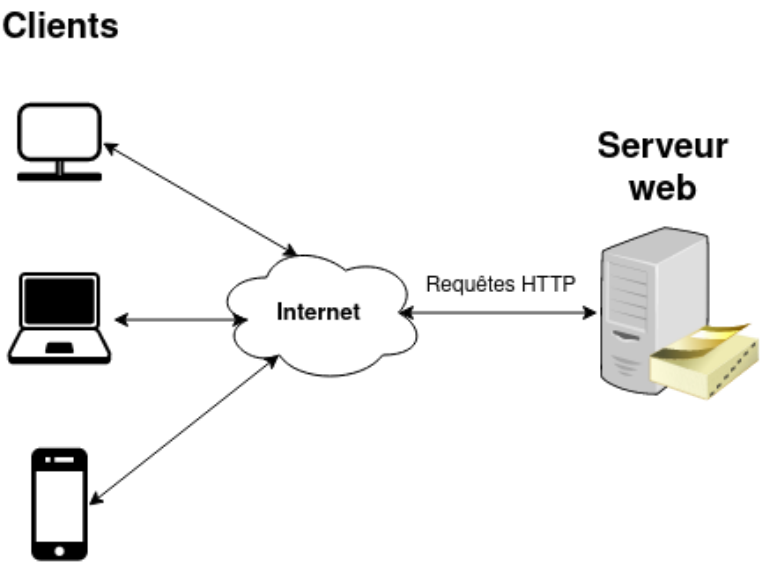
Un utilisateur possède une machine. Il navigue sur le web via son **navigateur** : Mozilla Firefox, Google Chrome, Safari, etc.

Les pages web affichées par le navigateur sont stockées sur des machines distantes : des **serveurs**.

On a donc au moins deux machines :

- le **client**, c'est à dire le navigateur, qui demande une page web,
- le **serveur**, qui héberge des logiciels destinés à **servir** les clients.

Ces deux machines communiquent via un **protocole** appelé HTTP.

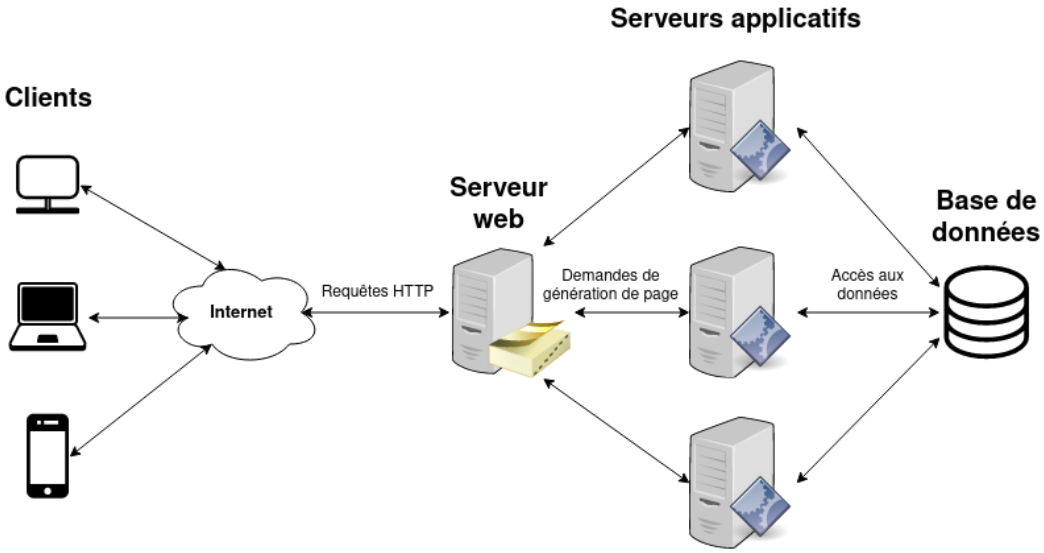


Architecture client-serveur

Plusieurs serveurs

Le terme serveur web est employé de manière assez large. Plus rigoureusement, on sépare un serveur web en plusieurs composants :

- Le **serveur web** à proprement parler. C'est l'interlocuteur direct du client avec qui il dialogue en HTTP. Il est chargé de renvoyer la page web finale. Apache et Ngnix sont les principaux serveurs web.
- Les **serveurs applicatifs** : des programmes qui créent les pages web à la volée à partir de données dynamiques, comme les résultats d'une recherche. Les serveurs applicatifs sont sollicités par le serveur web pour générer la page finale. Ils sont programmés dans des langages de programmation comme PHP, Python, Java, Ruby, etc.
- Le **serveur de base de données** : le programme qui stocke les données utilisées par le serveur applicatif. Les plus connus sont MySQL, PostgreSQL, MariaDB.



Architecture client-serveur avec serveur applicatif et base de données

Front-end *Remarque*

Tous les éléments d'une page web visibles par le navigateur forment le **front-end**. C'est la couche de présentation de la page web.

Les éléments graphiques qui composent le front-end sont en fait la traduction par le navigateur du contenu des pages web écrits dans trois langages : HTML, CSS et JavaScript.

Back-end *Remarque*

Les programmes qui s'exécutent sur le serveur, invisibles pour le navigateur, sont réunis sous le terme de **back-end**: c'est l'infrastructure qui permet de servir les requêtes réalisées par le client et de construire les pages web.

Architecture 3 tiers *Définition*

L'architecture classique d'une application web est en couches :

- la **présentation** est la couche du client,
- le **traitement** est la couche applicative,
- l'accès aux **données**.

Modèle de communication

Le modèle de communication établit une hiérarchie entre les couches :

1. Le client envoie une requête à l'application et attend sa réponse.
2. La couche application reçoit la requête et effectue un traitement particulier. À son tour, elle communique avec la base de données pour lui donner des données à stocker ou lui demander certaines informations nécessaires au traitement.
3. Une fois les données reçues par l'application, le traitement peut continuer et elle va pouvoir répondre à la requête initiale du client.

Échanges entre niveaux *Attention*

Chaque couche rend service à la couche directement supérieure. Ainsi, le client et la base de données ne communiquent pas directement entre eux car ces couches ne sont pas voisines.

Demande d'authentification *Exemple*

Lorsqu'un utilisateur veut se connecter à son compte, il fournit généralement son e-mail et son mot de passe.

Le formulaire de connexion envoie les identifiants dans sa requête au serveur.

Ce dernier reçoit la demande de connexion et exécute un traitement qui consiste à vérifier l'existence de l'utilisateur et le mot de passe associé. Il fait donc appel à la base de données, et si la réponse de celle-ci le confirme, il peut authentifier le client.

L'application finit par répondre à la requête du client en le redirigeant vers la bonne page.

À retenir

- Les applications web se décomposent généralement en trois tiers : le client, l'application, la base de données.
- Le client est chargé de la présentation de la page et l'application génère la page à l'aide des données de la base de données.
- Chaque couche ne communique qu'avec son voisin immédiat.

[cf. 8aVNYswu]

Exercice : Appliquer la notion

On donne ici plusieurs programmes d'une application web 3 tiers. Classer chacun de ces programmes dans la couche qui lui correspond.

Nginx | Apache | PHP | Chrome | PostgreSQL | MySQL | Firefox | Django

Client	Serveur de données	Serveur web	Serveur applicatif

Le langage client HTML

[cf. dZVgl7wY]

Objectif

- Découvrir comment est structuré le contenu d'une page web.

Mise en situation

Lorsque l'on visite un site web, le navigateur affiche un document sous la forme de caractères mis en forme, articulés avec des images, ou encore des liens.

En fait le navigateur récupère un fichier texte contenant du code HTML. Ce code indique au navigateur quels sont les éléments de contenu (une chaîne de caractères, une image), quel est leur type (un titre, un paragraphe) et comment les ordonner (ce paragraphe ci vient avant celui-là).

C'est en interprétant ce code HTML que le navigateur construit une présentation lisible par un être humain.

HTML : le langage de contenu des pages web — *Définition*

Le contenu des pages web est écrit en HTML, pour *HyperText Markup Langage*.

HTML est un langage de **balisage** : sa fonction est de **structurer** le contenu de la page, grâce à des balises.

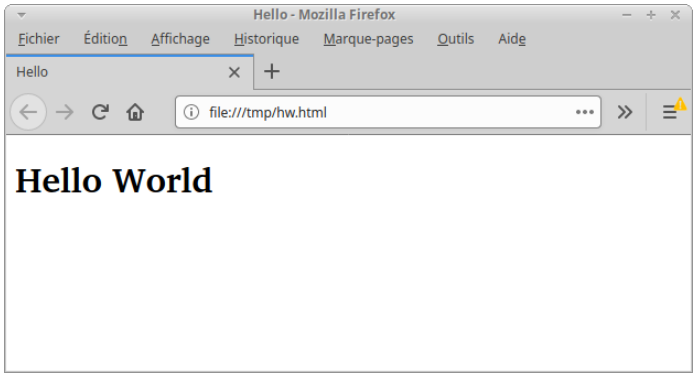
Les balisent servent à décrire la fonction des éléments et à leur donner du **sens**. Elles se composent de trois parties :

- une balise ouvrante : `<balise>`,
- un contenu,
- une balise fermante : `</balise>`,

Le contenu des pages web est stocké dans des fichiers HTML dont l'extension est généralement `html`.

Hello World — *Exemple*

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello</title>
  </head>
  <body>
    <h1>Hello World</h1>
  </body>
</html>
```



Balises

Syntaxe

Le code minimal d'un fichier HTML est composé de :

1. Une balise initiale, indiquant le type du document `<!DOCTYPE html>`.
2. Une balise racine `<html>` : elle englobe tout le reste du contenu.
    - La balise `<head>` : elle contient les méta-données de la page et son contenu n'apparaît pas dans le rendu final.
    - La balise `<body>` : elle contient tout le contenu visible de la page.

Exemples de balises de base

Complément

- Titres (du plus au moins important) : `<h1> <h2> <h3> <h4> <h5> <h6>`
- Blocs : `<div> <section> <article> <form> <footer> <header> <menu>`
- Éléments linéaires : `<span> <p>`
- Champs : `<input /> <textarea /> <button> <select>`
- Emphases : `<em> <b> <i>`
- Listes : `<ul> <ol> <li>`
- Images : `<img src=""/>`
- Liens : `<a href="">`
- Commentaires (non-visibles dans le rendu finale) : `<!-- Ceci est un commentaire -->`

HTML et éditeur de texte

Remarque

Les éditeurs de textes comme LibreOffice Writer ou Microsoft Word utilisent un langage de balisage proche du HTML pour représenter les documents. Quand un titre est créé ou une image insérée, ce sont en réalité des balises similaires qui sont enregistrées dans le fichiers, et **rendues** par l'éditeur.

Paragraphe

Exemple

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <!-- Encodage du fichier, ici UTF-8 -->
5     <meta charset="utf-8">
6     <!-- Titre de la page dans le navigateur -->
7     <title>Un paragraphe</title>
8   </head>
```

```
<body>
  <h1>Un titre en HTML </h1>
  <p>Un paragraphe de texte dans lequel on peut écrire le corps d'une page web.
  Ceci est un exemple minimaliste de fichier HTML.
  </p>
</body>
</html>
```

Un titre en HTML

Un paragraphe de texte dans lequel on peut écrire le corps d'une page web. Ceci est un exemple minimaliste de fichier HTML.

Images

Exemple

```
<!DOCTYPE html>
  <head>
    <meta charset="utf-8">
    <title>Une image</title>
  </head>
<html>
 <body>
  <h1>Vacances 2010</h1>
  <h2>À la mer</h2>
  <!-- Image dont l'emplacement sur le serveur se trouve dans le dossier images -->
  <img src="/images/seven-sisters.jpg"/>
 </body>
</html>
```

Vacances 2010

À la mer

HTML à travers les âges

Complément

HTML a été créé en 1992, mais chaque navigateur avait sa propre version de HTML, et sa propre manière d'interpréter les balises. Les sites web avaient un aspect différent sur chaque navigateur, particulièrement pour les éléments complexes.

Depuis sa standardisation par le W3C (World Wide Web Consortium) en 1995, la compatibilité s'est améliorée et aujourd'hui la plupart les sites web ont un rendu identique pour tous les navigateurs standards.

À retenir

- HTML est un langage de balisage permettant de structurer le contenu du document.
- Un fichier HTML contient des en-têtes (métadonnées) et un corps (contenu).
- La structure d'un document peut se faire de manière sémantique : titre, paragraphe, section, image, lien, etc.
- Les navigateurs savent comment afficher ces différents éléments de manière standard.

[cf. z1ujA0L8]

Exercice : Appliquer la notion

Voici le contenu d'un fichier HTML.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Ma page</title>
 </head>
 <body>
   <h3>Liste de courses</h3>
   <ol>
    <li><s>Pommes</s></li>
    <li><s>Farine</s></li>
    <li>Levure</li>
    <li><b>Oeufs</b></li>
   </ol>

   <h3>Choses à faire <br/><i>Très important!</i></h3>
   <ul>
    <li><s>Factures</s></li>
    <li>Appeler Mamie</li>
    <li>Sport</li>
   </ul>
 </body>
</html>
```

Question 1

Testez-le sur Repl.it[1] en créant un repl de type `HTML, CSS, JavaScript`.

Question 2

Quelle est la différence entre les balises `<ol>` et `<ul>` ?

Question 3

Que fait la balise `<br/>` ?

Question 4

Quelle phrase est en italique et quelle est la balise qui permet ça ?

[1] https://repl.it/

Le langage client CSS

[cf. 3fNS86Zj]

Objectif

- Découvrir comment les pages web sont stylisées.

Mise en situation

Une page web n'est pas seulement constituée d'information structurée, elle est également associée à une mise en page : des polices de caractères, des alignements, des espacements, des couleurs, etc.

C'est ce que l'on appelle le **style**.

Le style d'une page HTML est défini par une feuille de style écrite dans le langage CSS.

CSS est un langage qui permet par exemple de dire :

- « La police principale est en Times ».
- « Ajouter un espacement de 6 points avant tous les paragraphes ».
- Ou encore : « les titres de niveau 1 sont à afficher en vert ».

CSS : le langage de style des pages web *Définition*

Le CSS (pour *Cascading Style Sheets*) est le langage qui définit les propriétés de style des éléments des pages web.

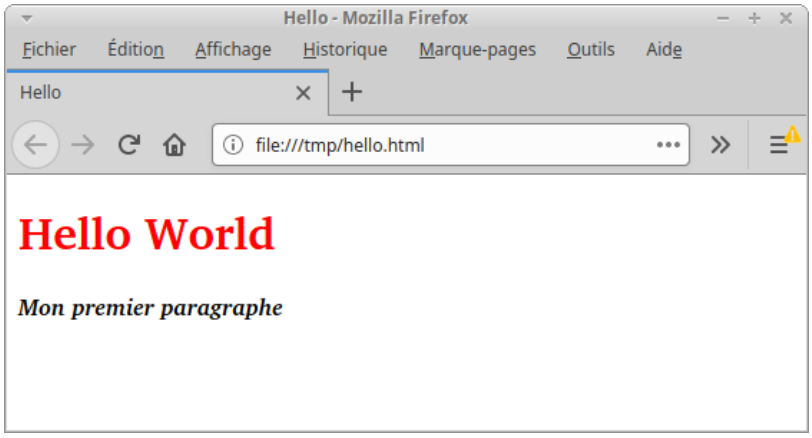
Ces éléments de styles sont par exemple :

- les couleurs,
- les polices de caractères,
- la taille des caractères,
- les alignements et marges,
- etc.

Hello world *Méthode*

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Hello</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Hello World</h1>
    <p>Mon premier paragraphe</p>
    <style>
      h1 {
```

```
        color: red;
      }
      p {
        font-weight: bold;
        font-style: italic;
      }
    </style>
  </body>
</html>
```



CSS et sélecteurs

Syntaxe

Le langage CSS fonctionne en appliquant un style particulier à des éléments de la page. La structure générale est la suivante :

```
type_element {
  propriete: valeur;
}
```

Dans cet exemple, tous les éléments de type `type_element` seront impactés par la valeur de `propriete`.

Quelques propriétés CSS

Exemple

- Polices : `font-family`, `font-size`, `color`, `font-weight`, `text-shadow`
- Dimensions : `width`, `height`, `margin`, `padding`
- Dispositions : `position`, `top`, `bottom`, `right`, `left`
- Visibilité : `display`, `opacity`, `visibility`
- Formes : `border-radius`, `transform`, `border`

Ajouter du CSS avec la balise <style>

Méthode

La première façon d'ajouter du CSS à une page HTML consiste à écrire le code entre les balises `<style>` dans le fichier `HTML`.

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>...</title>
 </head>
 <body>
  ...
  <style>
```

```
      ...
  </style>
 </body>
</html>
```

Le rouge et le noir

Exemple

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8" />
  <title>Ma page</title>
 </head>
 <body>
  <h1>Le <span class="rouge">rouge</span> et le <span class="noir">noir</span></h1>
  <style>
    body {
      color: gray;
    }
    h1 .rouge {
      color: red;
    }
    h1 .noir {
      color: black;
    }
  </style>
 </body>
</html>
```

Ajouter du CSS avec un fichier .css

Méthode

La seconde méthode consiste à utiliser un fichier `.css` indépendant, lié à la page dans les en-têtes.

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

Le rouge et le noir v2

Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Ma page</title>
    <link href="style.css" rel="stylesheet" />
  </head>
```

```
  <body>
    <h1>Le <span class="rouge">rouge</span> et le <span class="noir">noir</span>
</h1>
  </body>
</html>
```

```
body {
 background : #FFFEFA;
 color: gray;
}
h1 .rouge {
 color: red;
}
h1 .noir {
 color: black;
}
```

À retenir *Syntaxe*

- Le CSS permet de styliser les éléments structurés d'une page HTML.
- On peut ajouter le style directement dans le fichier HTML avec la balise ou lier le fichier HTML à un fichier CSS indépendant.
- Les propriétés CSS s'appliquent à un ensemble d'éléments HTML grâce à des sélecteurs.

[cf. QO5aWTDu]

Exercice : Appliquer la notion

On dispose du code HTML et CSS suivant.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exercice</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>Palette</h1>
    <table>
      <tr>
        <td class="red">Red</td>
        <td class="blue">Blue</td>
        <td class="green">Green</td>
        <td class="yellow">Yellow</td>
      </tr>
      <tr>
        <td class="black">Black</td>
        <td class="pink">Pink</td>
        <td class="orange">Orange</td>
        <td class="violet">Violet</td>
      </tr>
    </table>
  </body>
</html>
```

```css
h1 {
  color: indigo;
}
table {
  margin: 55px;
  border: black solid 1px;
}
td {
  padding: 10px;
}
.red {
  background: red;
}
.blue {
  background: cornflowerblue;
}
.pink {
  background: hotpink;
}
.yellow {
  background: gold;
}
.orange {
  background: orange;
```

```
}
.green {
  background: greenyellow;
}
.violet {
  background: purple;
}
.black {
  background: black;
  color: white;
}
```

Question 1

Intégrer ces deux fichiers dans Repl.it et visualiser le résultat.

Question 2

Quel élément est entouré d'une bordure ?

Question 3

Quelle propriété CSS modifie la couleur d'un texte ?

Question 4

Changer le style de la case « Black » pour lui mettre un fond blanc et un texte noir.

Le langage client JavaScript

[cf. ihMQo6WP]

Objectifs

- Comprendre comme une page web réagit aux événements ;
- Comprendre le concept de communication asynchrone.

Mise en situation

Aujourd'hui les pages web sont souvent utilisées pour interagir avec les utilisateurs. Par exemple en consultant un blog, on sera amené à trier les articles, faire une recherche, ou encore afficher une image en plein écran.

HTML et CSS sont des langages déclaratifs assez simples qui n'ont pas prévu tous les types de présentation, et encore moins d'interaction.

Aussi il est nécessaire de mobiliser un langage de programmation complet, au sein même du navigateur, pour pouvoir faire tout ce qui n'est pas prévu. Il s'agit du JavaScript.

JavaScript permet par exemple de changer l'ordre d'un résultat de recherche, en le triant par date ou par pertinence.

JavaScript *Définition*

Un troisième langage vient compléter HTML et CSS pour permettre la programmation des interactions avec l'utilisateur : **JavaScript**.

JavaScript est un langage de programmation exécuté directement par le navigateur web (**côté client**).

Hello World *Exemple*

Le code suivant crée un titre interactif, qui affiche le texte « Hello World » dans une fenêtre pop-up lors d'un clic.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exercice</title>
  </head>
  <body>
    <h1><em style="cursor: pointer" onclick="handleClick()">Cliquer</em> pour dire bonjour</h1>
    <script>
      function handleClick(){
        alert("Hello World");
      }
    </script>
  </body>
</html>
```

On peut intégrer du code JavaScript à une page de deux manières, grâce à la balise `<script>` :

- Soit en l'écrivant directement entre les balises.
- Soit en l'important depuis un fichier, dont l'extension est généralement `.js`.

Hello World v2

Exemple

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exercice</title>
  </head>
  <body>
    <h1><em style="cursor: pointer" onclick="handleClick()">Clique</em> pour dire bonjour</h1>
    <script src="script.js"></script>
  </body>
</html>
```

```js
function handleClick(){
    alert("Hello World");
}
```

Événements

Fondamental

Le premier intérêt du JavaScript dans une page HTML est la gestion des **événements** : clic, survol du curseur, déroulement de la page, écriture dans un champ, raccourcis clavier, etc.

Ces actions de l'utilisateur sont des événements auxquels il est possible d'associer des actions.

Événements

Exemple

Ce code gère deux événements : le clic de la souris sur le paragraphe et l'appui d'une touche sur le clavier. Pour chacun de ces événements, un message différent est affiché.

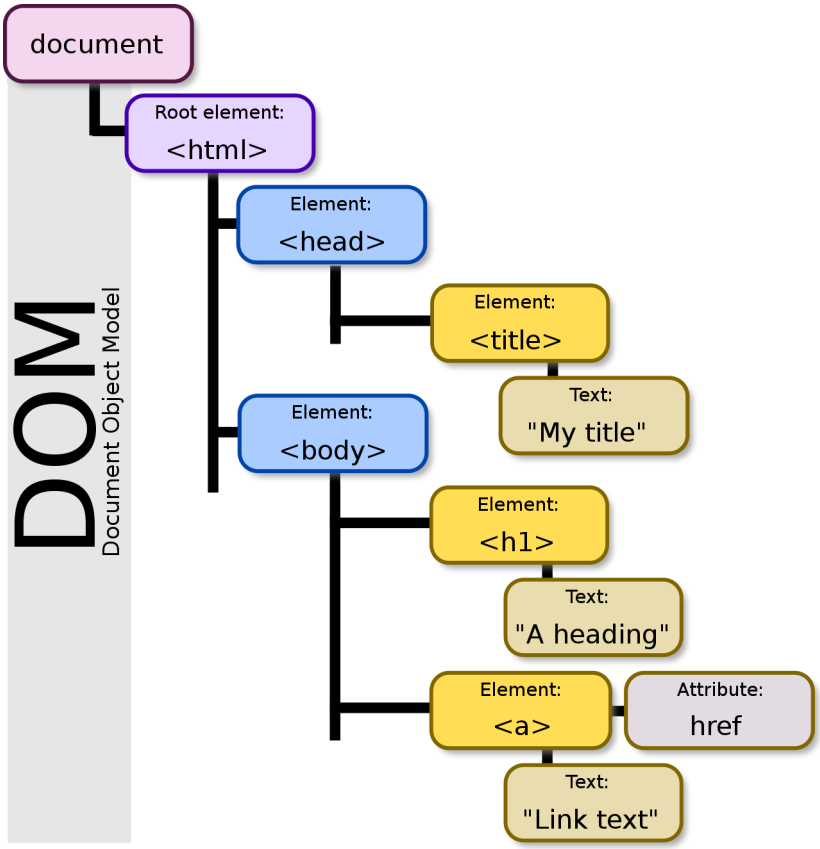
```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exercice</title>
  </head>
  <body onkeypress="handleKey()">
    <p onmouseover="handleMouse()">
      Je souhaite interagir avec toi.
    <p>
    <script src="script.js"></script>
  </body>
</html>
```

```js
function handleMouse(){
    alert("Je suis là...");
}
function handleKey(){
    alert("Hey ! pas avec le clavier !");
}
```

Manipuler la page HTML avec le DOM

Méthode

Avec JavaScript, la structure de la page est représentée par un arbre appelé DOM (pour *Document Object Model*), qui décrit la hiérarchie des balises au sein du document.



Représentation d'un document HTML sous forme d'arbre

JavaScript fournit des utilitaires pour manipuler les éléments du DOM. Par exemple, la méthode `document.getElementsByTagName("h1")` renvoie l'ensemble des titres de type `<h1>`, que l'on peut lire ou modifier.

Manipulation d'éléments avec JavaScript

Exemple

Le code suivant opère deux traitement sur les éléments de titre `<h1>` initialement présents dans la page.

- Le premier traitement change la couleur du premier titre et le centre.
- Le deuxième traitement supprime le deuxième titre.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Transformation</title>
  </head>
  <body>
    <h1>Phrase mise en couleur avec du javascript</h1>
    <h1>Phrase inutile</h1>
    <script>
      // récupère la première balise h1 de la page
      const h1a = document.getElementsByTagName("h1")[0];
      h1a.style.color = "red";
      h1a.style.textAlign = "center";
```

```
15      // récupère la seconde balise h1 de la page
16      const h1b = document.getElementsByTagName("h1")[1];
17      document.body.removeChild(h1b);
18    </script>
19  </body>
20 </html>
```

Phrase transformée avec du javascript

Requêtes et asynchronisme

Complément

Un autre intérêt de JavaScript est de savoir effectuer des requêtes, par exemple pour récupérer et afficher les ingrédients d'une recette de cuisine sur le serveur lors du clic sur un bouton « Voir plus ».

Le concept de requête est lié avec celui d'asynchronisme : au lieu de bloquer le rendu de l'ensemble de la page en attendant que les ingrédients soient récupérés depuis le serveur, le rendu continue et l'utilisateur peut toujours interagir avec la page.

Lorsque la requête termine, une action définie à l'avance appelée ***callback*** est déclenchée.

On retrouve le concept d'événement : une requête qui termine est un événement au même titre que l'appui d'une touche sur le clavier.

Frameworks JavaScript

Complément

Les applications web sont aujourd'hui souvent développées à l'aide de ***frameworks***, ou *cadriciels*. Ils mettent à disposition un ensemble d'outils et de ressources accélérant le développement web.

On pourra citer :

- Vue.js
- React
- Angular

À retenir

Le JavaScript permet de modifier des éléments d'une page web et de réagir à des événements, tels que le clic sur un bouton ou la réception de données demandées au serveur.

[cf. pPy4SMGx]

Exercice : Appliquer la notion

On dispose des codes HTML, CSS et JavaScript suivants.

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Exercice</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <section class="cards">
      <div class="green">0</div>
      <div class="red">7</div>
      <div class="blue">9</div>
      <div class="blue">6</div>
      <div class="yellow">6</div>
      <div class="red">0</div>
      <div class="green">1</div>
      <div class="blue">3</div>
      <div class="green">8</div>
      <div class="red">2</div>
    </section>
    <button onclick="add()">Ajouter</button>
    <script src="script.js"></script>
  </body>
</html>
```

```css
.cards {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  padding: 0 20px;
}
div {
  padding: 70px 50px;
  margin: 20px;
  border-radius: 5%;
  color: white;
  font-weight: bold;
}
.green {
  background: green;
}
.blue {
  background: blue;
}
.red {
  background: red;
}
.yellow {
  background: yellow;
}
```

```css
button {
  background: white;
  border: solid 1px black;
  padding: 10px;
  border-radius: 10px;
}
```

```js
const colors = ["green", "red", "yellow", "blue"]

function add() {
    const cards = document.querySelector(".cards");
    const num = Math.trunc(Math.random() * 9);
    const randomColor = Math.trunc(Math.random() * 4);
    console.log(num, randomColor);

    const newCard = document.createElement("div");
    newCard.classList = colors[randomColor];
    newCard.innerText = num;
    cards.appendChild(newCard);
}
```

Question 1

Créez les fichiers `index.html`, `style.css` et `script.js` dans `Repl.it` et afficher le résultat.

Question 2

Quel événement est écouté sur la page ? Sur quel élément HTML ?

Question 3

Que se passe-t-il lorsque l'on clique sur le bouton ?

Question 4

Ajouter un second bouton sur la page HTML avec le même comportement et `Adjoindre` comme nom.

Les langages serveur

[cf. vU0EkBsW]

Objectifs

- Comprendre la différence entre site statique et site dynamique ;
- Découvrir les méthodes pour générer des pages web dynamiquement.

Mise en situation

Certaines pages web évoluent très régulièrement : sites d'actualités, comptes en banque, résultats de recherche, catalogue de produits, etc.

L'information qu'elles présentent dépend de données qui sont modifiées par d'autres utilisateurs en temps réel. Ces pages HTML sont dites dynamiques. En fait elle n'existent pas en tant que fichier HTML sur un serveur web, elle sont construites à chaque demande d'un utilisateur sur le serveur.

La construction de ces pages est réalisée par un programme qui s'exécute sur le serveur. Aujourd'hui il existe plusieurs langages pour faire cela, comme PHP, Java ou Python, et même JavaScript (qui peut donc être utilisé aussi bien côté client que serveur).

Par exemple : lorsque vous demandez une page avec des informations sur l'état de votre compte en banque, en fait vous donnez l'ordre d'exécuter un programme sur le serveur. Ce programme va consulter la base de données qui contient vos informations bancaires et produire une page HTML qui sera renvoyée à votre navigateur.

Site web dynamique *Définition*

Un site web dynamique est un site dont les pages varient en fonction du temps et du contexte, sans modification des fichiers de code.

Les pages d'un site dynamique sont générées « à la volée » sur le serveur au moment où un client demande à les afficher.

Moteur de recherche *Exemple*

En l'absence de pages dynamiques, un moteur de recherche devrait créer un fichier HTML pour toutes les recherches possibles, et mettre à jour régulièrement le contenu de ces pages.

Il est évident que c'est impossible. La solution trouvée consiste à garder le même squelette (une page de résultats de recherche), mais de récupérer dynamiquement le contenu depuis une base de données.

Ainsi, un seul fichier est utilisé et la gestion est beaucoup plus simple.

Page d'accueil d'un blog *Exemple*

Un blog est assez répétitif : il s'agit d'une suite d'articles.

Un site statique demanderait de copier manuellement les blocs HTML permettant d'afficher un résumé d'article autant de fois qu'il y a d'articles.

Un site dynamique connaît le squelette d'un résumé et injecte le résumé des articles autant de fois qu'il y a d'articles de manière automatisée, au moment de la génération.

Langages de programmation côté serveur

Fondamental

Le langage HTML est un langage **descriptif** : il permet de structurer un document mais n'autorise pas l'exécution d'**instructions**.

Pour apporter du dynamisme à une page, il est nécessaire d'exécuter des instructions (récupération de données, envoi de mail, requêtes à d'autres applications, etc.).

Les **langages de programmation côté serveur** répondent à cette problématique : ils construisent la page demandée, qui sera renvoyée au client par le serveur web.

Du code bien caché

Attention

Le code permettant de générer les pages dynamiques n'est pas accessible par le client, qui ne voit que le résultat final.

Un écosystème fourni

Chaque langage de programmation côté serveur possède ses spécificités et sera choisi en fonction des besoins du site web. Chaque langage est associé à un grand nombre de **bibliothèques logicielles**, qui gère des fonctionnalités courantes.

Parmi les langages côté serveur les plus répandus, on compte :

- Python
- PHP
- JAVA

PHP

Exemple

PHP a la spécificité de pouvoir être intégré directement dans du code HTML, que l'on peut interpréter comme le **squelette** de la page.

Le code suivant affiche un nombre aléatoire à chaque chargement de la page.

```php
<html>
  <head>
    <title>Génération d'un nombre aléatoire entre 1 et 10</title>
  </head>
  <body>
    <?php echo '<p>' . rand(1, 10) . '</p>'; ?>
  </body>
</html>
```

Le code HTML reçu par le client lors d'un chargement pourra par exemple être :

```html
<html>
  <head>
    <title>Génération d'un nombre aléatoire</title>
  </head>
  <body>
    <p>5</p>
  </body>
</html>
```

On constate qu'il s'agit de code HTML simple, sans référence à PHP.

La pointe de l'iceberg *Complément*

La génération de pages dynamiques n'est qu'un des aspects de la programmation côté serveur. Ils permettent de réaliser les traitements nécessaires au fonctionnement de nombreux sites :

- Vérification des informations de connexion (utilisateur, mot de passe),
- Traitement d'un ordre d'achat,
- Enregistrement de préférences,
- etc.

À retenir

- Les sites dynamiques sont des sites dont les pages changent souvent.
- Les langages de programmation côté serveur permettent de générer des pages web.
- Le client ne reçoit que des pages web classiques et n'a pas accès au code de génération.

[cf. fYrff6jQ]

Exercice : Appliquer la notion

Voici ici un exemple simplifié de traitement d'une requête d'un serveur applicatif Python.

```
# Code du serveur applicatif
def traitement_serveur(name):
  html = "<h1>Hello {}!</h1>\n".format(name)
  html += "<p>How are you doing {}?</p>".format(name)
  return html

# Requête réalisée par l'utilisateur
if __name__ == '__main__':
  html = traitement_serveur("Gandalf")
  print(html)
```

Question 1

Créer un repl Python pour exécuter ce code. Quel est le code HTML retourné par le programme ?

Question 2

Modifier le repl pour que le texte renvoyé à l'utilisateur soit en français.

Un langage serveur : PHP

[cf. lIoVYLmK]

Objectif

- Savoir générer une page HTML dynamiquement grâce à PHP.

Mise en situation

Parmi les langages web côté serveur, PHP a été, et reste encore, très utilisé. C'est un langage impératif assez simple à appréhender pour démarrer, un peu moins exigeant que Java par exemple.

La syntaxe de PHP est assez proche d'autres langages comme C ou JavaScript.

Une de ses spécificités tient au fait qu'il est assez facile d'intégrer directement du code HTML dans un script PHP, ce qui allège la programmation des portions statiques de la page retournée.

C'est un langage libre dont l'écosystème est très développé dans le domaine des applications web.

PHP : le langage serveur de référence *Définition*

« PHP: Hypertext Preprocessor, plus connu sous son sigle PHP (sigle auto-référentiel), est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet. PHP a permis de créer un grand nombre de sites web célèbres, comme Facebook et Wikipédia. »

Article Wikipédia PHP : https://fr.wikipedia.org/wiki/PHP.

Écrire du PHP *Méthode*

Les fichiers PHP portent l'extension `.php` et peuvent combiner du code HTML et du code PHP à proprement parler.

Le PHP s'écrit entre des balises :

```php
<?php
/*
  Ceci est un commentaire.
  Tout ce qui est écrit entre ces symboles
  n'est pas pris en compte et est destiné
  aux développeurs.
*/
?>
```

Elles peuvent être insérées n'importe où dans le code HTML.

L'instruction `echo` insère le résultat de commandes PHP en lieu et place de la balise.

Le fichier final est interprété comme du HTML.

Les fichiers PHP sont réutilisables et peuvent être insérés dans un autre fichier PHP, par la directive suivante :

```
include("source.php")
```

Une variable est un espace de stockage dont le contenu peut changer au fil du temps. Les variables servent à stocker des nombres, des mots, des listes, etc.

Les **variables** en PHP sont précédées d'un symbole `$`.

```
/* La variable n est une chaîne de caractère valant "PHP" */
$n = "PHP"
/* La variable n est maintenant un nombre valant 1312 */
$n = 1312
/* La variable n vaut maintenant 1313 */
$n = $n + 1
```

Exemple de script PHP dans HTML ***Exemple***

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>PHP</title>
</head>
<body>
 <div>
  <?php
    $langage = "PHP";
    $version = 7.4;
    echo "<h1>Apprendre le " . $langage . " version " . $version . "</h1>";
  ?>
 </div>
 <div>
  <?php
    echo "Liste en PHP";
    echo "<ul>";
    echo "<li>0</li>";
    echo "<li>1</li>";
    echo "<li>2</li>";
    echo "<li>3</li>";
    echo "<li>4</li>";
    echo "</ul>";
  ?>
 </div>

</body>
</html>
```

Le fichier obtenu après **interprétation** du code PHP est un pur code HTML :

- Les directives `echo` produisent du code HTML.
- Les symboles `$langage` et `$version` ont été remplacés par leur valeurs respectives.

```
<html>
  <head>
    <meta charset="utf-8">
    <title>PHP</title>
  </head>
  <body>
    <div>
    <h1>Apprendre le PHP version 7.4</h1> </div>
    <div>
```

```
      Liste en PHP
      <ul>
        <li>0</li>
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
      </ul>
    </div>
  </body>
</html>
```

Ce qui donne le résultat suivant :

Apprendre le PHP version 7.4

Liste en PHP

- 0
- 1
- 2
- 3
- 4

Structures *Complément*

Les structures de contrôle en PHP ressemblent à la plupart des langages :

- la structure de condition utilise **if**(condition){...}.
- les boucles s'écrivent sous la forme **while**(condition){...} ou **for**(initialisation; condition; incrémentation){...}.

De même, on déclare une fonction avec le mot **function.**

GET et POST

La consultation d'une page web se fait grâce à un **type de requête** et de **paramètres**. PHP est capable de lire ces paramètres.

Dans le cas d'une requête `GET`, les paramètres sont stockés dans un **tableau** appelé `$_GET`, et pour une requête `POST` dans un tableau appelé `$_POST`.

L'accès à un paramètre `GET` se fait avec la syntaxe `$_GET["nom_parametre"]`, similairement pour les requêtes `POST`.

Quel type de requête choisir ? *Complément*

Les requêtes `GET` utilisent l'URL de la page pour passer des paramètres tandis que les requêtes `POST` utilisent le corps de la requête.

La différence essentielle réside dans le fait qu'une URL est partageable tandis que le corps d'une requête ne l'est pas.

De plus, les paramètres `GET` ont une taille limitée et ne peuvent contenir que des caractères ASCII.

Les requêtes `GET` par exemple utilisées pour les recherches, la consultation du profil d'un utilisateur, etc.

Les requêtes POST sont souvent utilisées pour l'envoi des données d'un formulaire, le téléchargement d'une image, etc.

Une bonne règle informelle est de considérer l'accès à une ressource peut se faire via une requête `GET` tandis que l'envoi de données peut se faire via une requête POST.

À retenir

- PHP est un langage très utilisé pour générer des pages web dynamiques.
- HTML et PHP peuvent être dans le même fichier : le code HTML forme le squelette et le code PHP fournit le contenu.
- Les paramètres des requêtes HTTP sont accessibles depuis PHP.

[cf. pZmAwgFN]

Exercice : Appliquer la notion

On dispose des deux fichiers PHP suivants :

```php
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Exercice</title>
</head>
<body>
 <h1>
    <?php
      echo "Welcome";
    ?>
  </h1>
  <?php
    echo '<div><a href="/page2.php?message=Hello World">Dire Bonjour</a></div>';
    echo '<div><a href="/page2.php">Rien à dire</a></div>';
  ?>
</body>
</html>
```

```php
<?php
    if (isset($_GET['message'])){
      echo $_GET['message'];
    }else{
      echo 'Pas de message';
    }
?>
```

Question 1

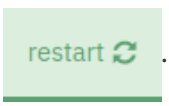
Créez un repl *PHP Web Server*. Intégrer le code de la première page dans le fichier `index.php`. Ajouter un fichier `page2.php` et intégrer le code de la seconde page.

Qu'est ce qui est affiché lorsqu'on clique sur « Dire Bonjour » ?

Question 2

Qu'est ce qui est affiché lorsqu'on clique sur « Rien à dire » ?

Indice :

Rechargez la page avec le bouton 

.

Question 3

Modifier la page `index.php` pour afficher « Au revoir » lorsque l'on clique sur « Rien à dire ».

Essentiel

[cf. DSfq0rYu]

Une application web repose sur une partie cliente, ou *front-end*, dont les langages principaux sont : HTML, CSS et JavaScript.

Elle repose également sur une partie serveur, ou *back-end*. On dispose aujourd'hui de nombreux langages pour implémenter les parties *back-end* des applications web. Un des langages les plus répandus est PHP.

Ainsi, réaliser une application web implique de :

- créer des pages HTML statiques, celles qui ne changent pas d'un utilisateur à l'autre,
- créer des feuilles de style CSS qui définissent la mise en forme du site,
- créer des programmes côté client en JavaScript qui permettent l'interaction directe avec l'utilisateur,
- créer des programmes dans un langage côté serveur, comme PHP, pour créer des pages dynamiquement en fonction des utilisateurs.

Quiz

Exercice 1 : Quiz - Culture

Exercice

Parmi ces types de site web, lesquels ont besoin d'utiliser une génération de page dynamique ?

- ☐ Un moteur de recherche
- ☐ Un site vitrine (CV, réalisations, etc.)
- ☐ Un site avec des animations
- ☐ Un réseau social
- ☐ Un blog

Exercice

Quels termes désignent la même chose ?

- ☐ Traitement et Couche applicative
- ☐ Front-end et back-end
- ☐ Langage de balisage et Langage de script
- ☐ Front-end et Client

Exercice 4 : Quiz - Méthode

Exercice

On souhaite choisir un langage pour développer le serveur d'une application. Comment choisit-on le langage pour le serveur applicatif ?

- ☐ On choisit forcément le PHP : c'est le plus pratique.
- ☐ On choisit le HTML et le CSS car on préfère ne pas mélanger trop de langages.
- ☐ On choisit le langage le plus utilisé du moment.
- ☐ On se dirige vers un langage avec lequel on est à l'aise.
- ☐ On s'intéresse aux outils disponibles dans ce langage pour notre usage.

Exercice

Comment fonctionnent les événements JavaScript ?

- [ ] Une action peut être associée à un événement pour un élément du DOM.
- [ ] On peut renseigner une action directement depuis la balise de l'élément.
- [ ] On peut associer une action à la réponse d'une requête asynchrone.
- [ ] Chaque événement ne peut être associé qu'à une seule action.
- [ ] On peut associer une action à un mouvement de souris.
- [ ] On peut associer une action à un changement de fichier sur le poste de l'utilisateur.

Exercice 7 : Quiz - Code

Exercice

Quel est le langage utilisé dans ce code ?

```
<!DOCTYPE html>
<html>
<body>
  <button>Click me</button>
</body>
</html>
```

- ○ HTML
- ○ PHP
- ○ CSS

Exercice

Quel est le langage utilisé dans ce code ?

```
<?php
  echo "Hello World!";
?>
```

- ○ HTML
- ○ PHP
- ○ CSS

Exercice

Quel est le langage utilisé dans ce code ?

```
p {
  color: red;
  text-align: center;
}
```

- ○ HTML
- ○ PHP
- ○ CSS

Exercice

On se donne le code HTML suivant :

```
 <!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

De quelle couleur est le titre ?

- ○ Marron
- ○ Noir
- ○ Blanc

Exercice

On se donne le code PHP suivant :

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>PHP</title>
</head>
<body>
  <?php
  if(isset($_GET["prenom"])) {
    echo "Bonjour " . $_GET("prenom");
  }
  if(isset($_GET["nom"])) {
    echo "Bonjour M/Mme" . $_GET("nom");
  }
  ?>
</body>
</html>
```

Qu'affiche la page si on la visite avec les paramètres `?prenom=Ada&nom=Lovelace` ?

- ○ Bonjour Ada
- ○ Bonjour M/Mme Lovelace
- ○ Les deux

- ○ Aucun des deux

Crédits des ressources

Architecture client-serveur p. 6

http://creativecommons.org/licenses/by/4.0/fr/, Quentin Duchemin, logos par Aybige, Loxaan Oxyde, Jean Yashu, Hea Poh Lin, ujmoser (https://search.creativecommons.org)

Architecture client-serveur avec serveur applicatif et base de données p. 6

http://creativecommons.org/licenses/by/4.0/fr/, Quentin Duchemin, logos par Aybige, Loxaan Oxyde, Jean Yashu, Hea Poh Lin, ujmoser (https://search.creativecommons.org)

Représentation d'un document HTML sous forme d'arbre p. 22

http://creativecommons.org/licenses/by-sa/3.0/fr/, Birger Eriksson — https://commons.wikimedia.org/w/index.php?curid=18034500