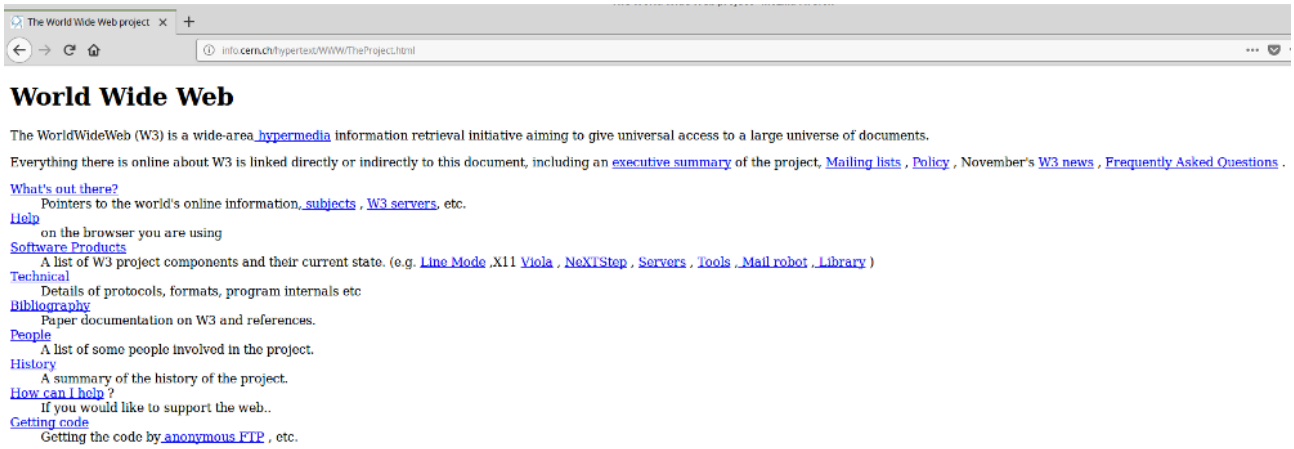
DÉVELOPPEMENT WEB
TP1 : LE WEB
DÉCOUVERTE DES DIFFERENTES TECHNOLOGIES

Le web est la principale application d'Internet. Le web (abréviation de World Wide Web ou toile mondiale) est donc un service d'Internet parmi d'autres. Néanmoins, c'est la création du web par Tim Berners-Lee en 1989 qui a popularisé l'utilisation d'Internet auprès du grand public, d'où la confusion qui existe encore entre ces deux termes.

Sources : Guillaume Fichet, Benjamin Monmège, Didier Müller, David Roche, wikipedia

1. Introduction au Web

Le "World Wide Web", plus communément appelé "Web" a été développé au CERN[1] par le Britannique Sir Timothy John Berners-Lee et le Belge Robert Cailliau au début des années 90. À cette époque les principaux centres de recherche mondiaux étaient déjà connectés les uns aux autres, mais pour faciliter les échanges d'information Tim Berners-Lee met au point le système hypertexte. Le système hypertexte permet, à partir d'un document, de consulter d'autres documents en cliquant sur des mots clés. Ces mots "cliquables" sont appelés hyperliens et sont souvent soulignés et en bleu. Ces hyperliens sont plutôt connus aujourd'hui sous le simple terme de "liens".


The World Wide Web project

info.cern.ch/hypertext/WWW/TheProject.html

World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Figure 1 première page web, les hyperliens sont soulignés et en bleu

Techniquement le web se base sur trois choses : le protocole HTTP[2], les URL[3] et le langage de description HTML[4]. Beaucoup de personnes confondent "web" et "internet". Même si le "web" "s'appuie" sur internet, les deux choses n'ont rien à voir puisqu'"internet" est un "réseau de réseaux" s'appuyant sur le protocole IP alors que le web est la combinaison de trois technologies : HTTP, URL et HTML. D'ailleurs on trouve d'autres services que le "web" sur internet, par exemple, les emails avec le protocole SMTP[5] et les transferts de fichiers avec le protocole FTP[6].

Le web est un des services disponibles sur Internet. Internet est un **réseau de services**. Internet est basée sur l'**architecture client/serveur** : Les services Internet sont tous fournis par des **serveurs**.

Les demandeurs du service sont nommés les **clients**. Les **clients** émettent des **requêtes** en se basant sur un **protocole**. Après traitement des requêtes, les serveurs renvoient des **réponses** en se basant sur un protocole. Chaque service est associé à un protocole (**web = HTTP**).

[1] Conseil Européen pour la Recherche Nucléaire
[2] HyperText Transfert Protocol
[3] Uniform Resource Locator
[4] HyperText Markup Language
[5] Simple Mail Transfert Protocol
[6] File Transfert Protocol

2. Les URL

Dans la barre d'adresse de votre navigateur web vous trouverez, quand vous visitez un site, des choses du genre : " https://openclassrooms.com/fr/courses/1603881-creez-votre-site-web-avec-html5-et-css3". Nous aurons l'occasion de reparler du "http" plus tard. La partie "/fr/courses/1603881-creez-votre-site-web-avec-html5-et-css3" s'appelle une URL.
Une URL[7] permet d'identifier une ressource (par exemple un fichier) sur un réseau.
L'URL indique « l'endroit » où se trouve une ressource sur un ordinateur. Un fichier peut se trouver dans un dossier qui peut lui-même se trouver dans un autre dossier... On parle d'une structure en arborescence, car elle ressemble à un arbre à l'envers :

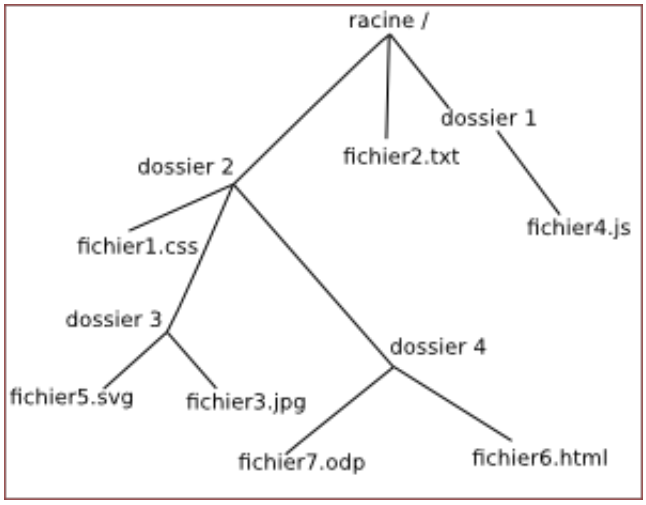


Figure 2 structure en arborescence

Comme vous pouvez le constater, la base de l'arbre s'appelle la racine de l'arborescence et se représente par un /

2.1. Chemin absolu ou chemin relatif ?

Pour indiquer la position d'un fichier (ou d'un dossier) dans l'arborescence, il existe 2 méthodes : indiquer un chemin absolu ou indiquer un chemin relatif. Le chemin absolu doit indiquer « le chemin » depuis la racine. Par exemple l'URL du fichier fichier3.jpg sera : **/dossier2/dossier3/fichier3.jpg**
Remarquez que nous démarrons bien de la racine **/** (attention les symboles de séparation sont aussi des /).

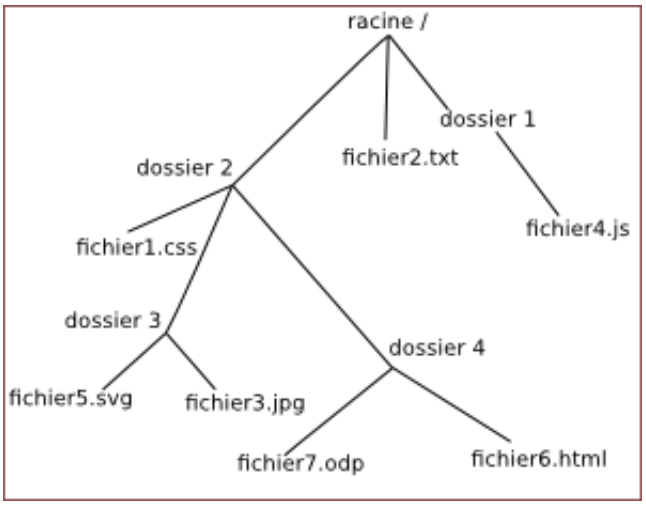
Imaginons maintenant que le fichier fichier1.css fasse appel au fichier fichier3.jpg (comme un fichier HTML peut faire appel à un fichier CSS). Il est possible d'indiquer le chemin non pas depuis la racine, mais depuis le dossier (dossier2) qui accueille le fichier1.css, nous parlerons alors de chemin relatif : **dossier3/fichier3.jpg**
Remarquez l'absence du / au début du chemin (c'est cela qui nous permettra de distinguer un chemin relatif et un chemin absolu).

Imaginons maintenant que nous désirions indiquer le chemin relatif du fichier fichier1.css depuis l'intérieur du dossier dossier4. Il faut « remonter » d'un « niveau » dans l'arborescence pour se retrouver dans le dossier dossier2 et ainsi pouvoir repartir vers la bonne « branche ». Pour ce faire il faut utiliser 2 points :
../dossier2/fichier1.css

Il est tout à fait possible de remonter de plusieurs « niveaux » : **../../ depuis le dossier dossier4** permet de « retourner » à la racine.

[7]Uniform Resource Locator

Exemple : soit la structure en arborescence suivante.



Le contenu du fichier "fichier7.odp" utilise le fichier "fichier5.svg". Donnez le chemin relatif qui devra être renseigner dans le fichier "fichier7.odp" afin d'atteindre le fichier "fichier5.svg".
Remarque : la façon d'écrire les chemins (avec des slash (/) comme séparateurs) est propre aux systèmes dits « UNIX », par exemple GNU/Linux ou encore Mac OS. Sous Windows, ce n'est pas le slash qui est utilisé, mais l'antislash (\). Pour ce qui nous concerne ici, les chemins réseau (et donc le web), pas de problème, c'est le slash qui est utilisé.

3. HTML et au CSS : les bases

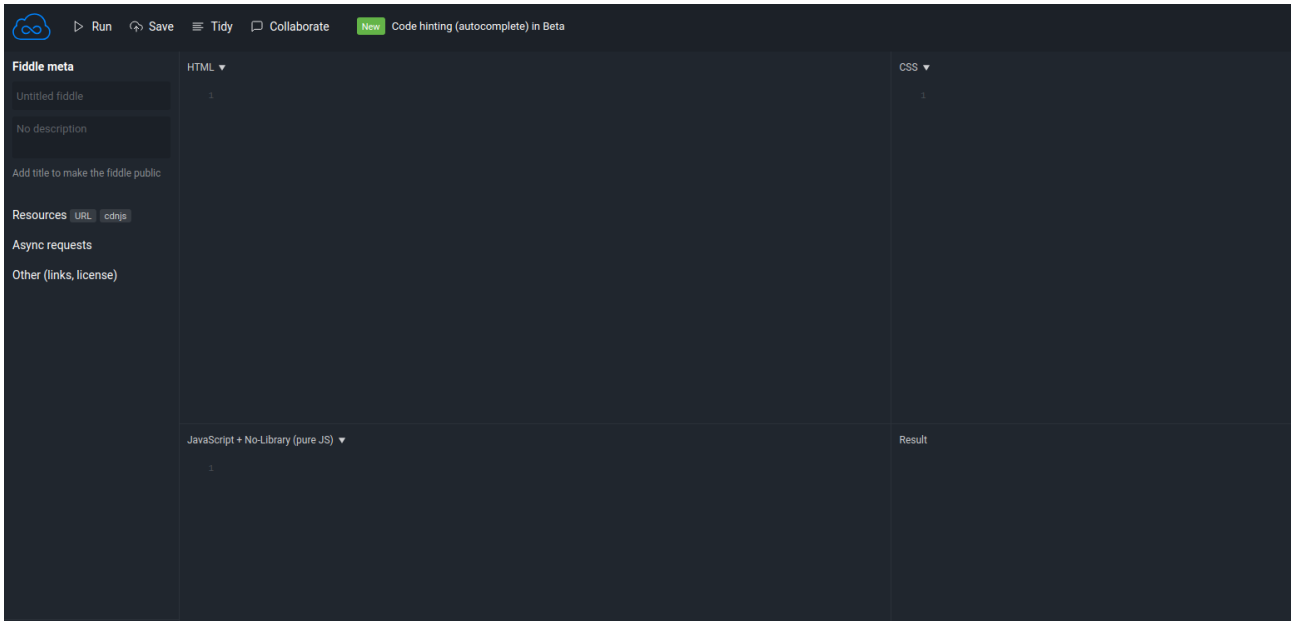
L'Hypertext Markup Language, généralement abrégé HTML, est le format de données conçu pour représenter les pages web. C'est un langage de balisage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet également de structurer sémantiquement et de mettre en forme le contenu des pages, d'inclure des ressources multimédias, dont des images, des formulaires de saisie, et des programmes informatiques. Il permet de créer des documents interopérables avec des équipements très variés de manière conforme aux exigences de l'accessibilité du web. Il est souvent utilisé conjointement avec des langages de programmation (JavaScript) et des formats de présentation (feuilles de style en cascade).
Source : wikipedia

Pour l'instant, nous allons retenir deux éléments de cette définition « conçu pour représenter les pages web » et « un langage de balisage ».
Grâce au HTML vous allez pouvoir, dans votre navigateur (Firefox, Chrome, Opera,....), afficher du texte, afficher des images, proposer des hyperliens (liens vers d'autres pages web), afficher des formulaires et même maintenant afficher des vidéos (grâce à la dernière version du HTML, l'HTML5).
HTML n'est pas un langage de programmation (comme le Python par exemple), ici, pas question de conditions, de boucles… c'est un langage de description.

3.1. JSfiddle.net

Pour aborder le HTML, nous allons, dans un premier temps utiliser le site jsfiddle.net.

⇒ Lancer un navigateur web et taper http://jsfiddle.net/ dans la barre d'adresse et vous devriez voir apparaître ceci :



Nous allons pour l'instant uniquement utiliser la fenêtre « HTML » et la fenêtre « Result ».

Écrivez le code HTML suivant et cliquez sur : ▷ Run

```
<h1>Hello World! Ceci est un titre</h1>
<p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris ?</p>
```

Observez ce qui s'affiche dans la fenêtre.
Comme déjà évoqué ci-dessus, en HTML tout est une histoire de balise que l'on ouvre et que l'on ferme. Une balise ouvrante est de la forme `<nom_de_la_balise>`, les balises fermantes sont de la forme `</nom_de_la_balise>`.

En observant attentivement le code, vous devriez forcément remarquer que toute balise ouverte doit être refermée à un moment ou un autre. La balise ouvrante et la balise fermante peuvent être sur la même ligne ou pas, cela n'a aucune espèce d'importance, la seule question à se poser ici est : ai-je bien refermé toutes les balises que j'ai ouvertes ?

Enfin pour terminer avec les généralités sur les balises, il est important de savoir qu'une structure du type :

```
<balise1>
<balise2>
</balise1>
</balise2>
```

est interdite, la balise2 a été ouverte après la balise1, elle devra donc être refermée avant la balise1.

En revanche, l'enchaînement suivant est correct :

```
<balise1>
<balise2>
</balise2>
</balise1>
```

Notez que dans notre exemple nous respectons bien cette règle « d'imbrication » des balises avec la balise `<p>` et la balise `<strong>`.
Il est important de comprendre que chaque balise a une signification qu'il faut bien respecter (on parle de la sémantique des balises). Par exemple le texte situé entre la balise ouvrante et fermante `<h1>` est obligatoirement un titre important (il existe des balises `<h2>`, `<h3>`.....qui sont aussi des titres, mais des titres moins importants (sous-titre)). La balise `<p>` permet de définir des paragraphes, enfin, la balise `<strong>` permet de mettre en évidence un élément important.

Vous devez aussi savoir qu'il existe des balises qui sont à la fois ouvrantes et fermantes (`<balise/>`) : un exemple, la balise permettant de sauter une ligne, la balise `<br/>` (balise qu'il faut d'ailleurs éviter d'utiliser car elle ne possède pas de signification sémantique propre).
Il est possible d'ajouter des éléments à une balise ouvrante, on parle d'attribut. Une balise peut contenir plusieurs attributs :

```
<ma_balise attribut_1= "valeur_1" attribut_2="valeur_2">
```

Il existe beaucoup d'attributs différents, nous allons nous contenter de 2 exemples avec l'attribut id (id pour identifiant) et class.

⇒ Écrire le code HTML suivant :

```
<h1>Ceci est un titre</h1>
<h2 class="titre_1">Ceci est un sous titre</h2>
<p id="para_1">Ceci est un <strong>paragraphe</strong>. Avez-vous bien
compris ?</p>
```

Observer ce qui s'affiche dans la fenêtre.
Le HTML n'a pas été conçu pour gérer la mise en page. Le HTML s'occupe uniquement du contenu et de la sémantique, pour tout ce qui concerne la mise en page et l'aspect « décoratif » (on parle du « style » de la page), on utilisera le CSS[8]. Dans JSFIDDLE, il est possible d'écrire du CSS dans la fenêtre en haut à droite.

⇒ Écrire le code HTML suivant :

```
<h1>Ceci est un titre</h1>
<h2>Ceci est un sous titre</h2>
<p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris ?</p>
```

⇒ Écrire le code CSS suivant :

```
h1
{
      text-align: center;
      background-color: red;
}
h2
{
      font-family: Verdana;
      font-style: italic;
      color: green;
}
```

Observez ce qui s'affiche dans la fenêtre.

Dans l'exemple précédent, les propriétés « text-align » et «background-color» seront appliquées au contenu de toutes les balises de type h1 (avec respectivement les valeurs «center» et «red»).....

Écrivez le code HTML suivant :

```
<h1>Ceci est un titre</h1>
<h2>Ceci est un sous titre</h2>
<p id="para_1">Ceci est un <strong>paragraphe</strong>. Avez-vous bien
compris ?</p>
```

[8]Cascading Style Sheets

Écrire le code CSS suivant :

```
#para_1
{
      font-style: italic;
      color: green;
}
```

Observez ce qui s'affiche dans la fenêtre.
Il est donc possible de cibler un paragraphe et pas un autre en utilisant l'id du paragraphe (en CSS l'id se traduisant par le signe #).
Il est aussi possible d'utiliser l'attribut class à la place de l'id. Dans le CSS on utilisera le point . à la place du #.
L'attribut "class" permet de donner le même nom à plusieurs reprises dans une même page alors que l'id est unique.
Si nous avions eu un 3ème paragraphe, nous aurions pu avoir : `<p class="para_1">Voici un 3e paragraphe</p>`, mais nous n'aurions pas pu avoir : `<p id="para_1"> Voici un 3e paragraphe </p>`, car le nom para_1 a déjà été utilisé pour le 1er paragraphe.

3.2. Créer son premier site web

JSFIDDLE est un très bel outil, mais il ne peut pas être utilisé pour la réalisation d'un vrai site internet (ou d'une vraie application web).
A l'aide d'un éditeur de texte, nous allons créer 2 fichiers : un fichier qui contiendra du HTML (index.html) et un fichier qui contiendra du CSS (style.css).

À l'aide d'un éditeur de texte (ex : notepad++), créer un nouveau fichier.

Sauvegardez-le en précisant son nom, par exemple "index.html".

Écrivez le code suivant dans votre éditeur de texte (sans oublier de sauvegarder quand vous avez terminé) :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Voici mon site</title>
      </head>
      <body>
            <h1>Hello World! Ceci est un titre</h1>
            <p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris
?</p>
      </body>
</html>
```

Testez votre code à l'aide d'un navigateur web (ex : Firefox ou Chrome) en “double-cliquant” sur le fichier index.html
D’après l'exemple précédent, vous reconnaissez le code se trouvant entre les balises `<body>` :

```
<body>
...
</body>
```

Tout votre code HTML devra se trouver entre ces 2 balises.

Le reste des balises devraient vous êtes inconnues. Passons-les en revue :

- La première ligne (<!doctype html>) permet d'indiquer au navigateur que nous utiliserons la dernière version du HTML, le fameux HTML5.
- La balise <html> est obligatoire, l'attribut lang="fr" permet d'indiquer au navigateur que nous utiliserons le français pour écrire notre page.
- Les balises <head>...</head> délimitent ce que l'on appelle l'en-tête. L'en-tête contient, dans notre exemple, 2 balises : la balise <meta charset="utf-8"> qui permet de définir l'encodage des caractères (utf-8) et la balise <title> qui définit le titre de la page (attention ce titre ne s'affiche pas dans le navigateur, ne pas confondre avec la balise <h1>).

Toujours à l'aide d'un éditeur de texte, créer un fichier qui va contenir le CSS de notre page (par exemple style.css).

⇒ Complétez ce fichier à l'aide du code suivant :

```
h1
{
      text-align: center;
      background-color: red;
}
p
{
      font-family: Verdana;
      font-style: italic;
      color: green;
}
```

Pour l'instant notre CSS ne sera pas appliqué à notre page, pour ce faire, il faut modifier notre code HTML en ajoutant une ligne qui va permettre d'associer notre code CSS à notre page.

⇒ Modifier le code HTML avec la ligne suivante <link rel="stylesheet" href="style.css"> :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Voici mon site</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Hello World! Ceci est un titre</h1>
            <p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris
?</p>
      </body>
</html>
```

⇒ Tester votre code à l'aide d'un navigateur web en “double-cliquant” sur le fichier index.html

Dans l'exemple que nous venons de voir, les fichiers "index.html" et "style.css" se trouvent dans le même dossier. Il est souvent utile de placer les fichiers CSS dans un dossier “CSS”. Il faudra alors modifier le code HTML en conséquence : <link rel="stylesheet" href="css/style.css">

Pour terminer, voici quelques balises très utilisées :

La balise a

```
<a href="mon_autre_page.html">Cliquez ici pour vous rendre sur mon autre page</a>
```

La balise a permet de créer des liens hypertextes, ce sont ces liens hypertextes qui vous permettent de "voyager" entre les pages d'un site ou entre les sites. Les liens hypertextes sont par défaut soulignés et de couleur bleue (modifiable grâce au CSS). La balise a possède un attribut href qui a pour valeur le chemin du fichier que l'on cherche à atteindre ou l'adresse du site cible (exemple : `<a href="http://www.google.fr">Cliquez ici pour vous rendre sur google.fr</a>`). Entre la balise ouvrante et fermante, on trouve le texte qui s'affichera à l'écran (c'est ce texte qui est souligné et de couleur bleue). La balise peut sans problème se trouver en plein milieu d'un paragraphe.

La balise image

Comme vous devez déjà vous en douter, la balise image sert à insérer des… images :

```
<img src="mon_image.jpg" alt="avion"/>
```

La balise img est à la fois ouvrante et fermante comme la balise br. Elle possède 2 attributs :

- src qui doit être égal au nom du fichier image (ou au chemin si le fichier image se trouve dans un autre dossier).
- alt qui doit être égal à une description de votre image (cet attribut est utilisé notamment par les systèmes de description des pages web utilisées par les non-voyants), il faut donc systématiquement renseigner cet attribut.

Les balises form, input et button

Les formulaires sont des éléments importants des sites internet, ils permettent à l'utilisateur de transmettre des informations. Un formulaire devra être délimité par une balise `form` (même si ce n'est pas une obligation) :

```
<form>
....
</form>
```

Il existe différentes balises permettant de construire un formulaire, notamment la balise `input`. Cette balise possède un attribut type qui lui permet de jouer des rôles très différents. La balise `button` nous sera aussi d'une grande utilité.

Exercice

⇒ Créer un fichier html contenant le code suivant :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Voici mon site</title>
      </head>
      <body>
            <form>
                  <p>voici un champ de texte : <input type="text"/></p>
                  <p>voici une checkbox <input type="checkbox"/></p>
                  <button>Cliquez ici !</button>
            </form>
      </body>
</html>
```

⇒ Tester le code à l'aide d'un navigateur web en “double-cliquant” sur le fichier html saisi.

Les balises div et span

Ces 2 balises sont très utilisées (surtout la balise `div`) pour organiser la page et regrouper plusieurs balises dans une même entité.

La balise `div` est une balise de type “block” et `span` est une balise de type “inline”.

Sans vouloir trop entrer dans les détails, il faut bien comprendre que l'ordre des balises dans le code HTML a une grande importance. Les balises sont affichées les unes après les autres, on parle du flux normal de la page.

C'est ici qu'entrent en jeu les balises de type "block" et les balises de type "inline".

- Les contenus des balises de type "block" (`p, div,h1,...`) **se placent sur la page web les uns en dessous des autres.**
- Les contenus des balises de type "inline" (`strong, img, span, a`) **se placent sur la page web les uns à côté des autres.**

Cela revient à dire qu'une balise de type “block” prend toute la largeur de la page alors qu'une balise de type “inline” prend juste la largeur qui lui est nécessaire.

Exercice

⇒ Créez un fichier html et testez le code suivant :

```
<!doctype html>
<html lang="fr">
   <head>
      <meta charset="utf-8">
      <title>Voici mon site</title>
   </head>
   <body>
   <div>div est une balise de type "block"</div>
      <p>la balise p est une autre balise de type block</p>
      <span>En revanche, span est une balise de type "inline"</span>
      <a href="www.google.fr">Et voici une autre balise de type "inline"</a>
      <h1>h1 est bien une balise de type "block"</h1>
      <span>la balise span est "obligée" de se placer en dessous</span>
   </body>
</html>
```

4. Interaction avec une page web : utilisation du JavaScript

Nous avons déjà pu nous familiariser avec le couple HTML-CSS qui est en fait un trio, car aujourd'hui un développeur web ne peut pas faire l'impasse sur le JavaScript.

Notre but ici n'est pas d'apprendre un nouveau langage de programmation, mais juste d'étudier quelques exemples d'utilisation du JavaScript, notamment dans le cas des interactions entre un utilisateur et une page web.

Avant d'entrer dans le vif du sujet, un petit rappel historique : JavaScript a été créé en dix jours par Brendan Eich en 1995. Malgré son nom, JavaScript n'a rien à voir avec le langage Java, même si Brendan Eich affirme s'être inspiré de nombreux langage, dont Java, pour mettre au point JavaScript. Après des débuts chaotiques, il est, comme déjà dit plus haut, devenu incontournable dans le développement web.

4.1. Mise en place de l'environnement de travail

⇒ Créer un dossier portant votre nom. À l'aide d'un éditeur de texte (ex : notepad++), créer 3 fichiers dans votre répertoire de travail : index.html, style.css et script.js

Après avoir ouvert le fichier index.html à l'aide d'un éditeur de texte, saisir le code ci-dessous :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <p>Voici une page web qui ne fait pas grand-chose</p>
      </body>
<script src="script.js"></script>
</html>
```

Rien de très nouveau dans ce code, à part :
`<script src="script.js"></script>`
qui permet d'exécuter le programme JavaScript contenu dans le fichier "script.js".

Après avoir ouvert le fichier style.css à l'aide d'un éditeur de texte, saisir le code ci-dessous :

```
h1{
      text-align: center;
}
```

Après avoir ouvert le fichier script.js à l'aide d'un éditeur de texte, saisir le code ci-dessous :

```
alert("Le JavaScript fonctionne !")
```

⇒ Afin d'afficher la page web que nous venons de créer dans un navigateur web, cliquer sur le fichier "index.html"

Comme vous pouvez le constater, la page web s'affiche bien, mais nous avons en plus une fenêtre (souvent qualifiée de surgissante ou pop-up en anglais) qui apparaît. L'apparition de cette fenêtre est bien évidemment due à l'instruction "`alert`" présente dans le JavaScript.
Le but ici n'étant pas d'apprendre à programmer en JavaScript, nous nous contenterons pour le moment de cette simple instruction "`alert`". Évidemment JavaScript permet de faire bien plus de choses. En effet on retrouve en JavaScript les grands classiques des langages de programmation : variable, condition, boucle, fonction,...
Dans l'exemple ci-dessus, l'instruction "`alert`" est exécutée dès l'ouverture de la page web, il est tout à fait possible de lier l'exécution de certaines instructions JavaScript à l'action d'un utilisateur (par exemple un clic sur un bouton).

⇒ Modifier le code HTML comme suit :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <p>Voici une page web qui ne fait pas grand chose</p>
            <button onclick="maFonction()">Cliquer ici</button>
      </body>
      <script src="script.js"></script>
</html>
```

⇒ Modifier le code JavaScript comme suit :

```
function maFonction() {
      alert("Le JavaScript fonctionne !")
}
```

⇒ Tester cette nouvelle page en cliquant sur le fichier index.html

Comme vous pouvez le constater, l'instruction "`alert`" n'est plus exécutée à l'ouverture de la page web, mais uniquement dans le cas où l'utilisateur clique sur le bouton.
On a associé au bouton un événement "`onclick`", en cas de clic sur la souris, la fonction JavaScript "`maFonction()`" est exécutée. Si on s'intéresse au code JavaScript, on retrouve bien une fonction "`maFonction()`" ("`function maFonction(){...}`" en JavaScript est équivalent à un "`def maFonction() :`" en Python). Entre l'accolade ouvrante et l'accolade fermante (qui délimite la fonction), on retrouve uniquement notre instruction "`alert`". À l'ouverture de la page web cette instruction "`alert`" n'est pas exécutée, car elle se trouve dans une fonction. Le clic sur le bouton entraîne l'exécution de la fonction "`maFonction()`" et donc de l'instruction "`alert`".

Il est évidemment possible de faire des choses bien plus complexes que l'affichage d'un simple pop-up avec JavaScript. Il est possible de modifier le style d'une balise, de modifier la classe (CSS) d'une balise ou encore de modifier le contenu d'une balise, voici quelques exemples :

⇒ Modifier le code HTML comme suit :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <p id="monPara">Voici une page web qui ne fait pas grand chose</p>
            <button onclick="maFonction()">Cliquer ici</button>
      </body>
      <script src="script.js"></script>
</html>
```

⇒ Modifier le code JavaScript comme suit :

```
function maFonction() {
      document.querySelector("#monPara").style.color="red";
}
```

⇒ Tester cette nouvelle page en cliquant sur le fichier index.html

Dans l'exemple ci-dessous, nous avons déjà ajouté un id ("monPara") à la balise "p" dans notre code HTML. Dans le code JavaScript, la ligne :
`document.querySelector("#monPara").style.color="red";`
permet de modifier le style de la balise ayant pour id "monPara" : la couleur du texte devient rouge. Comme cette modification du style se trouve dans la fonction "maFonction()", cette modification sera effective uniquement si l'utilisateur appuie sur le bouton.

Il est possible de travailler plus "proprement" en utilisant les classes CSS :

⇒ Modifier le code HTML comme suit :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <p id="monPara">Voici une page web qui ne fait pas grand-chose</p>
            <button onclick="foncRouge()">Rouge</button>
            <button onclick="foncVert()">Vert</button>
      </body>
      <script src="script.js"></script>
</html>
```

⇒ Modifiez le code JavaScript comme suit :

```
function foncRouge() {
      document.querySelector("#monPara").classList.remove("vert");
      document.querySelector("#monPara").classList.add("rouge");
}
function foncVert() {
      document.querySelector("#monPara").classList.remove("rouge");
      document.querySelector("#monPara").classList.add("vert");
}
```

⇒ Modifiez le code CSS comme suit :

```
h1{
      text-align: center;
}
.rouge {
      color:red;
      font-size:20px;
}
.vert {
      color:green;
      font-size:30px;
}
```

⇒ Analyser le code ci-dessus et tester cette nouvelle page en cliquant sur le fichier index.html.

Dans l'exemple page précédente, nous avons maintenant 2 boutons, un clic sur le bouton "vert", permet d'exécuter la fonction "`foncVert()`", un clic sur le bouton "rouge", permet d'exécuter la fonction "`foncRouge()`", jusque-là, rien de vraiment nouveau. La fonction JavaScript "`foncVert()`" permet de modifier la classe CSS de la balise ayant pour id "monPara".
Dans un premier temps, la ligne :
`document.querySelector("#monPara").classList.remove("rouge");`
permet de supprimer l'association entre la balise d'id "`monPara`" et la classe CSS "`rouge`" (si cette association n'existe pas, cette ligne n'a aucun effet).
Dans un deuxième temps, on associe la classe CSS "`vert`" avec la balise d'id "`monPara`" avec la ligne :
`document.querySelector("#monPara").classList.add("vert");`
Le principe est identique avec la fonction "`foncRouge()`".

Il est également possible de modifier le contenu d'une balise HTML :

Modifier le code HTML comme suit :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <p id="monPara">Voici une page web qui ne fait pas grand chose</p>
            <button onclick="modifMessage()">Cliquez ici</button>
      </body>
      <script src="script.js"></script>
</html>
```

Modifier le code JavaScript comme suit :

```
function modifMessage() {
      document.querySelector("#monPara").innerHTML = "Bravo, vous avez cliqué
sur le bouton !"
}
```

Analyser et tester cette nouvelle page en cliquant sur le fichier index.html.

Le contenu de la balise ayant pour id "monPara" est modifié grâce à la ligne :
`document.querySelector("#monPara").innerHTML = "Bravo, vous avez cliqué sur le bouton !"`

Il existe d'autres événements que "`onclick`", par exemple, il est possible de détecter le "survol" par le curseur de la souris d'un élément HTML.

Modifier le code HTML comme suit :

```
<!doctype html>
<html lang="fr">
      <head>
            <meta charset="utf-8">
            <title>Le trio</title>
            <link rel="stylesheet" href="style.css">
      </head>
      <body>
            <h1>Le trio : HTML, CSS et JavaScript</h1>
            <div onmouseover="foncEntre()" onmouseout="foncQuitte()"
id="maDiv">
```

```
                <p>Survolez-moi</p><
            /div>
      </body>
      <script src="script.js"></script>
</html>
```

⇒ Modifier le code JavaScript comme suit :

```
function foncEntre(){
      document.querySelector("#maDiv").classList.remove("blanc");
      document.querySelector("#maDiv").classList.add("rouge");
}
function foncQuitte() {
      document.querySelector("#maDiv").classList.remove("rouge");
      document.querySelector("#maDiv").classList.add("blanc");
}
```

⇒ Modifier le code CSS comme suit :

```
h1{
      text-align: center;
}
p{
      text-align : center;
}
#maDiv{
      width : 200px;
      height : 100px;
      margin : 0 auto;
      border : 2px solid black;
}
.rouge {
      background-color:red;
}
.blanc {
      background-color : white;
}
```

⇒ Analyser le code ci-dessus et tester cette nouvelle page en cliquant sur le fichier index.html.

"`onemouseover`" correspond bien au survol par le curseur de la souris d'un élément HTML. L'événement "`onemouseout`" est lui déclenché quand le curseur de la souris quitte un élément HTML donné.
Il existe beaucoup d'autres événements que nous n'aborderons pas ici. Si vous voulez en savoir plus, vous pouvez **consulter ce site**.

En résumé, le code HTML permet de générer des éléments graphiques qui seront affichés par un navigateur web, mais pas seulement : il est aussi possible de mettre en place dans le code HTML des événements. Un événement donné pourra déclencher l'exécution d'instructions JavaScript.

5. Modèle client/serveur

Deux ordinateurs en réseau peuvent s'échanger des données. Dans la plupart des cas ces échanges ne sont pas "symétriques" : en effet un ordinateur A va souvent se contenter de demander des ressources (fichiers contenant du texte, photos, vidéos, sons...) à un ordinateur B. L'ordinateur B va lui se contenter de fournir des ressources à tous les ordinateurs qui lui en feront la demande. On dira alors que l'ordinateur A (celui qui demande des ressources) est un client alors que l'ordinateur B (celui qui fournit les ressources) sera qualifié de serveur.

En tapant «http://www.google.fr», votre machine va chercher à entrer en communication avec le serveur portant le nom «www.google.fr» (en faite c'est plus compliqué, pour les puristes nous dirons donc que la communication va être établie avec le serveur www du domaine google.fr, mais pour la suite nous pourrons nous contenter de l'explication « simplifiée »).

Une fois la liaison établie, le client et le serveur vont échanger des informations en dialoguant :

1. client : bonjour www.google.fr (ou bonjour www se trouvant dans le domaine google.fr), pourrais-tu m'envoyer le fichier index.html
2. serveur : OK client, voici le fichier index.html
3. client : je constate que des images, du code css sont utilisés, peux-tu me les envoyer
4. serveur : OK, les voici

Évidemment ce dialogue est très imagé, mais il porte tout de même une part de « vérité ».

Sur internet, ce modèle client/serveur domine largement car tous les ordinateurs jouent tour à tour le rôle de client et le rôle de serveur. Par exemple, comme expliqué dans l'exemple ci-dessus, on retrouve cet échange client/serveur à chaque fois que l'on visite une page web. Il y a de fortes chances pour que votre ordinateur personnel joue quasi exclusivement le rôle de client (sauf si vous êtes un adepte du "peer to peer").

N'importe quel type d'ordinateur peut jouer le rôle de serveur, mais dans le monde professionnel les serveurs sont des machines spécialisées conçues pour fonctionner 24h sur 24h. Ils peuvent aussi avoir une grosse capacité de stockage afin de stocker un grand nombre de ressources (vidéos, sons,...).

Figure 3 serveur

Afin assurer une continuité de service, dans les sociétés, plusieurs serveurs assurent exactement le même rôle (on parle de redondance). Vous vous doutez bien que Google ne possède pas qu'un seul serveur, en effet, en moyenne, chaque seconde, c'est environ 65000 clients qui se connectent aux serveurs du moteur de recherche de Google. Aucun serveur, même extrêmement performant, ne serait capable de répondre à toutes ces requêtes. Google, Amazon ou encore Facebook possèdent un très grand nombre de serveurs afin de pouvoir satisfaire les demandes des utilisateurs en permanence. Ces entreprises possèdent d'immenses salles contenant chacune des centaines ou des milliers de serveurs (ces serveurs sont rangés dans des armoires appelées "baie serveur").

Figure 4 Salle serveurs

Souvent les serveurs sont spécialisés dans certaines tâches, par exemple, les serveurs qui envoient aux clients des pages au format HTML sont appelés "serveur web".

Il y a quelques années, le web était dit « statique » : le concepteur de site web écrivait son code HTML et ce code était simplement envoyé par le serveur web au client. Les personnes qui consultaient le site avaient toutes le droit à la même page, le web était purement « consultatif ».
Les choses ont ensuite évoluées : les serveurs sont aujourd'hui capables de générer eux-mêmes du code HTML. Les résultats qui s'afficheront à l'écran dépendront donc des demandes effectuées par l'utilisateur du site : le web est devenu dynamique.
Différents langages de programmation peuvent être utilisés « côté serveur » afin de permettre au serveur de générer lui-même le code HTML à envoyer. Le plus utilisé encore aujourd'hui se nomme PHP. D'autres langages sont utilisables côté serveur (pour permettre la génération dynamique de code HTML) : Java, Python...

Voici un exemple très simple de code en PHP :

```
<?php
      $heure = date("H:i");
      echo '<h1>Bienvenue sur mon site</h1>
      <p>Il est '.$heure.'</p>';
?>
```

Sans entrer dans les détails, si un client se connecte à un serveur web qui exécute ce code à 18h23, le serveur enverra au client le code HTML ci-dessous :

```
<h1>Bienvenue sur mon site</h1>
<p>Il est 18h23</p>
```

En revanche si un client se connecte à ce même serveur à 9h12, le serveur enverra au client le code HTML ci-dessous :

```
<h1>Bienvenue sur mon site</h1>
<p>Il est 9h12</p>
```

Comme vous pouvez le constater, le PHP permet de générer des pages HTML dynamiquement. Inutile de préciser que cet exemple est volontairement très simple, le PHP est capable de générer des pages HTML bien plus complexes.

6. Protocole http

Revenons sur l'adresse qui s'affiche dans la barre d'adresse d'un navigateur web et plus précisément sur le début de cette adresse c'est-à-dire le "http" (selon les cas cette adresse commencera par http ou https).
Le protocole (un protocole est ensemble de règles qui permettent à 2 ordinateurs de communiquer ensemble) HTTP va permettre au client d'effectuer des requêtes à destination d'un serveur web. En retour, le serveur web va envoyer une réponse.

Voici une version simplifiée de la composition d'une requête HTTP (client vers serveur) :

- la méthode employée pour effectuer la requête
- l'URL de la ressource
- la version du protocole utilisé par le client (souvent HTTP 1.1)
- le navigateur employé (Firefox, Chrome) et sa version
- le type du document demandé (par exemple HTML)
- ...

Certaines de ces lignes sont optionnelles.
Voici un exemple de requête HTTP :

```
GET /mondossier/monFichier.html HTTP/1.1
User-Agent : Mozilla/5.0
Accept : text/html
```

Nous avons ici plusieurs informations :

- "GET" est la méthode employée (voir ci-dessous)
- "/mondossier/monFichier.html" correspond l'URL de la ressource demandée
- "HTTP/1.1" : la version du protocole est la 1.1
- "Mozilla/5.0" : le navigateur web employé est Firefox de la société Mozilla
- "text/html" : le client s'attend à recevoir du HTML

Une requête HTTP utilise une méthode (c'est une commande qui demande au serveur d'effectuer une certaine action). Voici la liste des méthodes disponibles :
GET, HEAD, POST, OPTIONS, CONNECT, TRACE, PUT, PATCH, DELETE

Détaillons 4 de ces méthodes :

- GET : C'est la méthode la plus courante pour demander une ressource. Elle est sans effet sur la ressource.
- POST : Cette méthode est utilisée pour soumettre des données en vue d'un traitement (côté serveur). Typiquement c'est la méthode employée lorsque l'on envoie au serveur les données issues d'un formulaire.
- DELETE : Cette méthode permet de supprimer une ressource sur le serveur.
- PUT : Cette méthode permet de modifier une ressource sur le serveur

6.1. Réponse du serveur à une requête HTTP

Une fois la requête reçue, le serveur va renvoyer une réponse, voici un exemple de réponse du serveur :

```
HTTP/1.1 200 OK
Date: Thu, 15 feb 2019 12:02:32 GMT
Server: Apache/2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=ISO-8859-1
<!doctype html>
<html lang="fr">
<head>
<meta charset="utf-8">
<title>Voici mon site</title>
</head>
<body>
 <h1>Hello World! Ceci est un titre</h1>
<p>Ceci est un <strong>paragraphe</strong>. Avez-vous bien compris ?</p>
</body>
</html>
```

Nous n'allons pas détailler cette réponse, voici quelques explications sur les éléments qui nous seront indispensables par la suite.
Commençons par la fin : le serveur renvoie du code HTML, une fois ce code reçu par le client, il est interprété par le navigateur qui affiche le résultat à l'écran. Cette partie correspond au corps de la réponse.

La 1ère ligne se nomme la ligne de statut :

- HTTP/1.1 : version de HTTP utilisé par le serveur
- 200 : code indiquant que le document recherché par le client a bien été trouvé par le serveur. Il existe d'autres codes dont un que vous connaissez peut-être déjà : le code 404 (qui signifie «Le document recherché n'a pu être trouvé»).

Les 5 lignes suivantes constituent l'en-tête de la réponse, une ligne nous intéresse plus particulièrement :
Server: Apache/2.0.54 (Debian GNU/Linux) DAV/2 SVN/1.1.4
Le serveur web qui a fourni la réponse http ci-dessus a comme système d'exploitation une distribution GNU/Linux nommée "Debian" (pour en savoir plus sur GNU/Linux, n'hésitez pas à faire vos propres recherches). "Apache" est le cœur du serveur web puisque c'est ce logiciel qui va gérer les requêtes http (recevoir les requêtes http en provenance des clients et renvoyer les réponses http). Il existe d'autres logiciels capables de gérer les requêtes

http (nginx, lighttpd...) mais, aux dernières nouvelles, Apache est toujours le plus populaire puisqu'il est installé sur environ la moitié des serveurs web mondiaux !

Le "HTTPS" est la version "sécurisée" du protocole HTTP. Par "sécurisé" en entend que les données sont chiffrées avant d'être transmises sur le réseau.
Voici les différentes étapes d'une communication client - serveur utilisant le protocole HTTPS :

1. le client demande au serveur une connexion sécurisée (en utilisant "https" à la place de "http" dans la barre d'adresse du navigateur web)
2. le serveur répond au client qu'il est OK pour l'établissement d'une connexion sécurisée. Afin de prouver au client qu'il est bien celui qu'il prétend être, le serveur fournit au client un certificat prouvant son "identité". En effet, il existe des attaques dites "man in the middle", où un serveur "pirate" essaye de se faire passer, par exemple, pour le serveur d'une banque : le client, pensant être en communication avec le serveur de sa banque, va saisir son identifiant et son mot de passe, identifiant et mot de passe qui seront récupérés par le serveur pirate. Afin d'éviter ce genre d'attaque, des organismes délivrent donc des certificats prouvant l'identité des sites qui proposent des connexions "https".
3. à partir de ce moment-là, les échanges entre le client et le serveur seront chiffrés grâce à un système de "clé publique - clé privée" (nous n'aborderons pas ici le principe du chiffrement par "clé publique - clé privée"). Même si un pirate arrivait à intercepter les données circulant entre le client et le serveur, ces dernières ne lui seraient d'aucune utilité, car totalement incompréhensible à cause du chiffrement (seuls le client et le serveur sont aptes à déchiffrer ces données)

D'un point vu strictement pratique il est nécessaire de bien vérifier que le protocole est bien utilisé (l'adresse commence par "https") avant de transmettre des données sensibles (coordonnées bancaires...). Si ce n'est pas le cas, passez votre chemin, car toute personne qui interceptera les paquets de données sera en mesure de lire vos données sensibles.