04 – Au delà du HTML

Nous avons vu que le Web est constitué de l'ensemble des pages HTML fournies par les serveurs HTTP du monde entier et qu'il forme ainsi un vaste ensemble de documents hypertextes. On peut consulter cette banque de données en utilisant un client HTTP, le plus souvent un navigateur comme Firefox.

Mais il y a bien plus que cela !
Nous allons voir aujourd'hui comment on peut rendre tout cela plus présentable via le CSS mais également comment les sites peuvent être dynamiques en analysant les données que fournit le client HTTP, en stockant les données sur leurs serveurs ou même chez le client !

A – Gérer l'aspect de la page via le CSS

01 – Ouvrir la page suivante : http://www.csszengarden.com/tr/francais/
Utiliser les choix de style pour voir qu'on ne change que l'apparence, le texte restant identique mais présenté différemment.

Nous obtenons une page contenant du texte et des menus permettant de changer l'apparence du site. C'est cela qui est notable : seule l'apparence et la disposition change, pas le contenu : le texte reste identique. Le code HTML dans le body reste identique. On dit qu'on a changé **le style de présentation**.

02 – Utiliser l'un des visuels, afficher le code source avec un clic-droit. Choisir un nouveau visuel et visualiser à nouveau le code source, sans fermer le précédent. Que constate-t-on ?

03 – Observez attentivement le contenu des deux balises head. Cherchez la différence qui existe dans les balises link, permettant de lier un document externe à la page html. Notez les deux codes correspondants.

La balise link semble associer un fichier d'extension **css** à notre fichier **html**. On donne son adresse à l'aide d'un attribut **href** comme pour créer des liens.

Les fichiers fournis dans les feuilles de style CSS (`"stylesheet"`) : il s'agit de fichiers qui contiennent des instructions permettant au navigateur de savoir comment et où le concepteur désire voir s'afficher le contenu de la page HTML.

Comment ça fonctionne ?
Pour le savoir nous allons aller sur un site moins complexe à comprendre.

04 – Utiliser le lien suivant pour vous connecter sur trinket : https://trinket.io/html/fc2c81f15e
solution via iframe : https://formation.infoforall.fr/structure.html

05 – Observez l'aspect puis rajoutez cette ligne à l'intérieur des balises head des 3 fichiers HTML.

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

Par exemple :

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8" />
		<title>Accueil HTML</title>
		<link rel="stylesheet" href="mesgouts.css">
	</head>

	<body>
		<h1>Accueil</h1>
	  <p>Vers les liens <a href="liens_html.html">HTML</a></p>
		<p>Vers les liens <a href="liens_perso.html">SNT</a></p>
	</body>

</html>
```

Moralité : nous avons réussi à créer une unité de ton sur nos trois pages en utilisant un seul fichier commun de configuration graphique : le fichier CSS(Cascading Style Sheets).

06 – Notre ensemble de fichiers forme un site dit statique. Faire bouger la souris sur la page. Statique veut-il dire que le site ne possède pas d'éléments mouvants ?

Statique fait référence au fait que le codes HTML et CSS sont toujours fournis à l'identique par le serveur HTTP qui les transmet. Sur un site dynamique, le serveur peut recevoir des données, lire des données dans sa propre base de données et renvoyer un code HTML ou CSS différents en fonction de la façon dont il a été programmé. L'une des activités fournies traite du cas du PHP par exemple.

Comment utiliser le CSS ?
C'est relativement simple à comprendre lorsqu'on a le code sous les yeux.

07 – Ouvrir l'onglet **mesgouts.css.** Observer le code pour tenter d'en voir la structure.

En voici un extrait :

```
a {
	color:yellow;
	font-size:x-large; /* ou small, x-small, x-large, medium,   ... */
}
```

On donne le nom d'une balise HYML sur laquelle on doit agir. Ici **a**.
Nous allons changer certaines des propriétés d'affichage de ces balises.
Lesquelles ? Celles fournies entre l'accolade d'ouverture `{` et l'accolade de fermeture `}`.

Les propriétés seront placées dans ce bloc `{-}` et indentées de façon à rendre le code plus clair.

Chaque modification est codifiée de la même façon :

- 1 - on donne le nom de la propriété
- 2 - on place un double-point pour indiquer la nouvelle valeur de la propriété (et pas =)
- 3 - on donne la nouvelle valeur de la propriété
- 4 - on place un point-virgule pour indiquer la fin de la modification sur cette propriété

```
a {
	color:yellow;
	font-size:x-large; /* ou small, x-small, x-large, medium,   ... */
}
```

Ici, on utilise donc des balises **a** dont

- le texte sera en jaune : `color:yellow;`
- les caractères sont de taille x-large : `font-size:x-large;`

Dernière remarque : on peut placer des commentaires en utilisant /* puis */. Le texte situé entre les deux sera alors uniquement destiné au lecteur humain du code.

08 – Modifier le code pour que les liens apparaissent en orange et en encore plus grand.

```
a {
      color:orange;
      font-size:xx-large; /* ou small, x-small, x-large, medium,    ... */

}
```

Le CSS permet également de gérer des conditions. Par exemple, nous voulons modifier les apparences lorsqu'on survole une balise. On utilise alors : et on utilise le mot-clé hover qui veut dire flotter (au dessus).

```
a:hover {
      background-color:yellow;
      color:black;
}
```

09 – A votre avis, à quoi correspond l'attribut **background-color** ? Arrangez-vous pour créer une unité de couleur avec le lien lorsqu'il n'est pas survolé.

```
a:hover {
      background-color:orange;
      color:black;
}
```

Remarque : vous pouvez voir qu'il existe une autre façon de fournir les couleurs en donnant trois composants : le rouge, le vert et le bleu. Cette méthode sera vue dans la partie photographie.
Ici dans h1, on a : `color:#333333;`
Cela veut dire qu'on a une intensité rouge de 33 (valeur comprise entre 00 et FF en hexadécimal)

Idem pour le vert et le bleu.
On obtient ainsi du gris foncé.
Le noir est codé par #**000000** ou #**000** de façon simplifiée.
Le blanc est codé par #**FFFFFF** ou #**FFF** de façon simplifiée.

De nombreux sites proposent de visualiser les couleurs obtenues par une association RGB en hexadécimal. Un exemple parmi d'autres : https://htmlcolorcodes.com/fr/

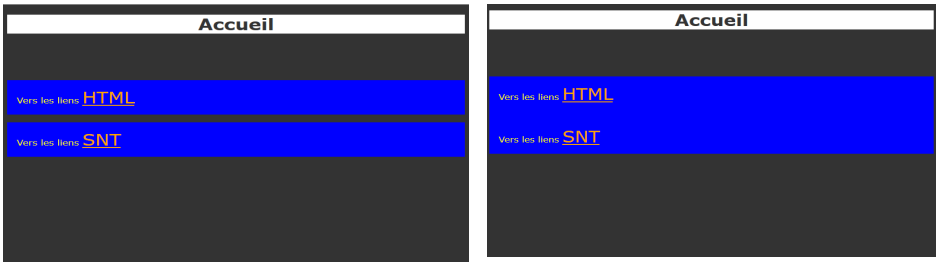
Imaginons maintenant qu'on veuille reproduire sur la page d'accueil le bloc bleu regroupant plusieurs paragraphes :

10 – Modifier la balise p pour que le fond soit bleu (par exemple #0000FF). Regarder la page d'accueil. Ca fonctionne ?

Et ça ne fonctionne pas. Il reste du gris entre les paragraphes !

Pour parvenir à englober tout à la fois, nous allons utiliser une balise générique de type block : la balise **div**. Cette balise n'a aucun sens sémantique, elle n'a pour but que de regrouper certaines balises en vu de leur appliquer un CSS commun.

Ici, l'effet obtenu à gauche et l'effet voulu à droite :



11 – Supprimer la couleur de fond de la balise p. Observer le code de **liens_html.html**. Modifier le code HTML de **index.html** de façon à obtenir l'affichage de droite.

```
<body>
      <h1>Accueil</h1>
      <div>
            <p>Vers les liens <a href="liens_html.html">HTML</a></p>
            <p>Vers les liens <a href="liens_perso.html">SNT</a></p>
      </div>
</body>
```

Pour en apprendre plus sur le CSS et tester les modifications en direct, vous pouvez utiliser le site de W3Schools par exemple.

Remarque : la balise html ** ** est une balise-bloc n'ayant aucun sens sémantique. Elle sert simplement de conteneur. Ici, elle sert à contenir différents paragraphes. En appliquant une propriété à la div, les balises internes (on dit filles) suivent les mêmes règles.

B – Gérer l'aspect dynamique du serveur

Pour l'instant, les pages fournies par le serveur sont statiques : le serveur va fournir le code HTML de la page et le code CSS de la page. Point. A chaque fois que la requête http est demandée sur une page, c'est la même page qui s'affiche.

Remarque : statique ne veut pas dire que rien ne bouge sur la page. On peut créer des animations en CSS. Statique ne fait pas référence au mouvement des choses à l'écran mais au fait que les codes HTML et CSS soient toujours les mêmes.

Impossible avec ce type de code d'enregistrer une commande, de vérifier un mot de passe ou de compter le nombre de fois que vous êtes venu.

Pour faire tout cela, il faut utiliser un programme côté serveur. On peut faire cela avec Python. Mais l'un des langages les plus utilisés reste le PHP.

12 – Allez à l'adresse suivante : **https://formation.infoforall.fr/**

Nous arrivons sur un site de démonstration sur lequel des programmes php récupèrent les requêtes, les analysent et créent des pages html en fonction des données reçues.

Pourquoi ? Simplement car il serait stupide de gérer des mots de passe côté client : il suffirait de regarder le code source.

13 – Choisir un pseudo et un mot de passe. Regardez les paramètres passés au serveur lors de l'envoi. Observez le code source à l'aide d'un clic-droit : le vrai mot de passe apparaît-il ?

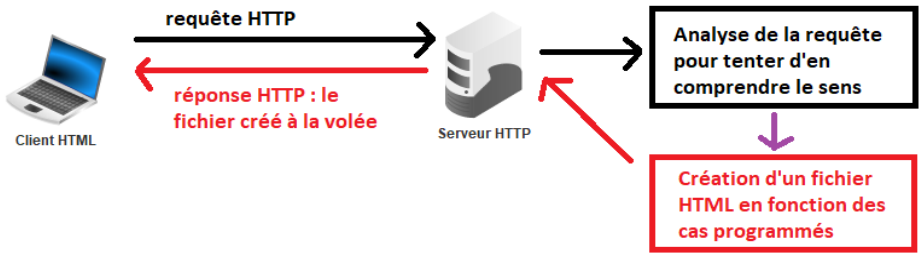
Si on regarde le code source (avec un clic-droit), on voit ceci :

```
<!DOCTYPE html>
<html lang="fr">
        <head>
                <meta charset="utf-8" />
                <title>Accueil</title>
        </head>
        <body>
                <h1>SNT : côté client / côté serveur : VERIFICATION</h1>
                <p>Identication : mauvais mot de passe</p>
                <p>Pour visualiser le reste du site, il faudra vous identifier !</p>
                <form method='get' action='identification.php'>
                        <p>Votre pseudo :
                              <input type='text' name='pseudo' placeholder='tapez votre pseudo' />
                        </p>
                        <p>Mot de passe :
                              <input type='text' name='motpasse' placeholder='tapez votre mdp' />
                        </p>
                        <p><input type='submit' value='Connexion' /></p>
                </form>
        </body>
</html>
```

La balise **form** est une balise html typique si ce n'est que :

- elle permet d'envoyer les données rentrées dans le formulaire (ici les textes nommés **pseudo** et **motpasse**) vers
- la page dont l'URL est indiquée dans action : **identification.php**
- en utilisant ici la méthode **get**, à savoir on passe les données via l'URL.

Pour connaître le mot de passe à fournir, il faudrait donc pouvoir voir le vrai code qui s'exécute lorsque le serveur HTTP reçoit la requête. Or, justement, le client ne peut pas le voir. C'est le but.



A titre d'exemple, voici le code volontairement simple et visible qui tourne sur cette page.

14 – Sans même connaître la moindre chose en PHP, tentez de trouver le mot de passe à utiliser en décodant le fichier php.

```
<!DOCTYPE html>
<html lang="fr">
	<head>
		<meta charset="utf-8" />
		<title>Accueil</title>
	</head>
	<body>
		<h1>SNT : côté client / côté serveur : VERIFICATION</h1>
		<?php
			if ($_GET['motpasse'] == 'snt2019!') {
				?>
				<p>Identification : check</p>
				<p>Cette phrase ne peut apparaitre que si vous avez
				tapé le bon mot de passe</p>
				<?php
			}
			else {
				?>
				<p>Identication : mauvais mot de passe</p>
				<p>Pour visualiser le reste du site, il faudra vous
					identifier !</p>
				<form method='get' action='identification.php'>
					<p>Votre pseudo : <input type='text' name='pseudo'
						placeholder='tapez votre pseudo' /> </p>
					<p>MdP : <input type='text' name='motpasse'
						placeholder='tapez votre mdp' /></p>
					<p><input type='submit' value='Connexion' /></p>
				</form>
				<?php
			}
		?>
	</body>
</html>
```

15 – Testez la réponse sur le site. Une fois passé le cap du mot de passe, observez le code source obtenu côté client. A-t-on affaire à du PHP ou à un HTML pur ?

Comme vous le voyez, l'intérêt des programmes côté serveur est le côté totalement opaque qu'il procure pour le client. De cette façon, on limite les intrusions sur le site. Mais, cela veut également dire que le serveur peut effectuer des opérations sur les données du client sans que le client ne soit au courant …

Remarque : en réalité, les mots de passe n'apparaissent jamais aussi clairement dans les codes. Il sont eux-mêmes cryptés de façon à rester dissimulés aux yeux des gens ayant accès au code.

Voyons maintenant ce que le citoyen que vous êtes peut faire pour protéger ses données.

16 – Lorsque vous faites une recherche sur un moteur de recherche, celui-ci peut-il stocker l'ensemble de vos requêtes ? Quelle est la différence affichée entre Qwant, Duck Duck Go et Google par exemple ?

17 – Avec Firefox, faire une recherche sur un moteur de recherche respectant la vie privée. Lorsque vous faites une recherche sur un moteur de recherche, celui-ci peut-il stocker l'ensemble de vos requêtes sur ses propres serveurs ? Utiliser OUTILS – DEVELOPPEMENT WEB – INSPECTEUR DE STOCKAGE pour visualiser si le site a placé des données permettant de vous identifier sur votre poste.

18 – Faire de même avec Google par exemple. Conclusion ?

C – La vérité sur les requêtes HTTP !

Nous avons vu que le navigateur est un programme qui est capable d'interpréter un document hypertexte extrêmement codifié. Ce langage codifié ce nomme le **HTML**.

Pour obtenir cette ressource HTLM, le navigateur communique avec un **serveur HTTP** distant. Pour trouver le serveur, notre navigateur a besoin d'une **URL** qui contient notamment le moyen de localiser le serveur et la ressource voulue.

Dans quasiment tous les cas, les pages à afficher contiennent des images, et un fichier **CSS** à trouver. Et parfois, on doit envoyer des **paramètres** au serveur....

Mais comment notre client HTTP (notre navigateur) et le serveur HTTP font-ils pour se comprendre ? Ce ne sont que deux programmes ...

La solution : on les a conçu pour :

- le serveur comprenne à coup sur la demande du client
- le client comprenne à coup sur la réponse du serveur.

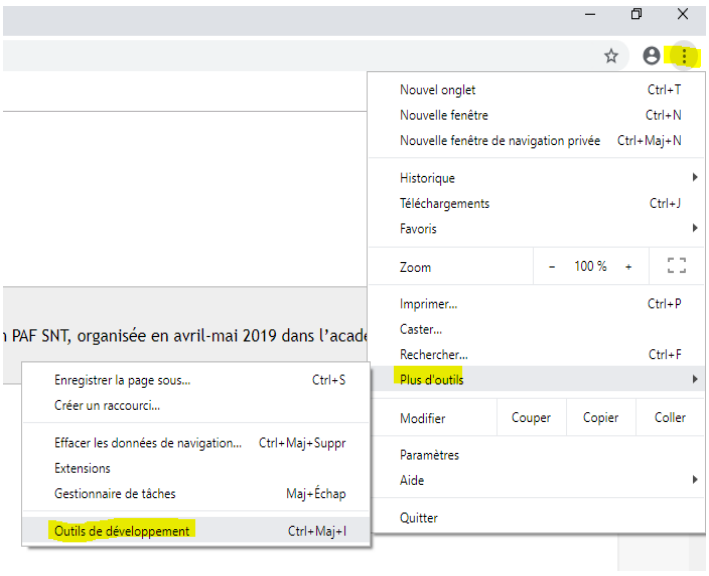
Cette méthode est le fameux **protocole HTTP** .

Imaginons qu'on veuille accéder à la première page Web de l'histoire.
Nous avons que son URL est : http://info.cern.ch/hypertext/WWW/TheProject.html

Regardons comment cela fonctionne. Nous allons utiliser Chrome car il permet d'afficher clairement ce qu'on envoie vers le serveur voulu.

19 – Dans Chrome, ouvrir menu, choisir Plus d'outils puis Outils de développement. Sélectionner **Network** ou **Réseau**.

20 – Choisir réseau/network **PUIS** placer ceci dans la barre d'adresse :
http://info.cern.ch/hypertext/WWW/TheProject.html

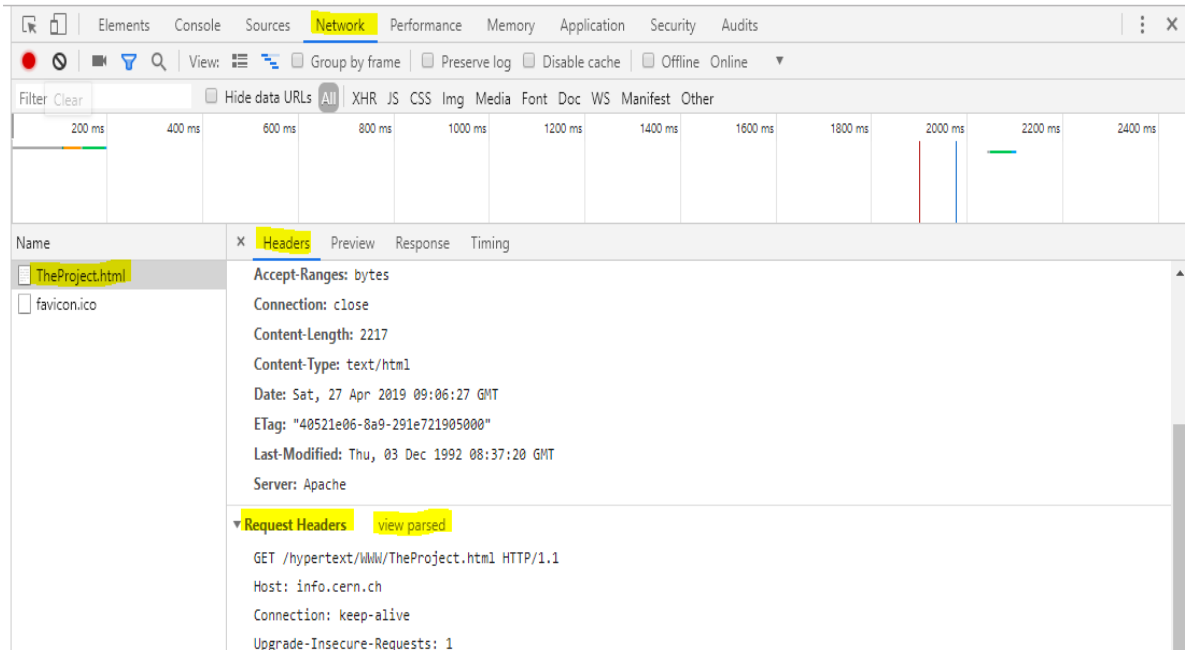


Un ensemble de sous-fenêtres en partie vides va apparaître.

21 – Dans la barre d'adresses, lancez **http://fe.fil.univ-lille1.fr/sntweb/**

22 – Cliquer sur la première ressource (la première ligne), prendre l'onglet HEADERS, descendre jusqu'à trouver REQUEST HEADERS et appuyer sur le bouton VIEW SOURCE.

Et on obtient ceci par exemple :



Vous avez sous les yeux la requête qu'envoie le client et que le serveur reçoit : (je ne place qu'une partie du message, la plus « compréhensible »)

```
GET /hypertext/WWW/TheProject.html HTTP/1.1
Host: info.cern.ch
```

Principe d'une requête :

Les trois premières informations à transmettre au serveur (de façon codifiée!) sont :
- l'action qu'on lui demande : ici, c'est codifié par GET pour **récupérer** une ressource.
- un caractère d'espacement
- l'**adresse de la ressource sur le serveur** : ici c'est donc /hypertext/WWW/TheProject.html
- un caractère d'espacement
- la version du protocole HTTP utilisé : la version actuelle est **`HTTP/1.1`**

Sous cette première partie, on trouve ensuite beaucoup beaucoup d'informations. Il s'agit de ce qu'on appelle l'**en-tête** : on donne plein d'informations au serveur pour qu'il puisse gérer au mieux notre demande. On transmet également toujours ces informations de façon très codifiée :
- nom codifié de l'information apportée
- deux point en tant que caractère de séparation
- **valeur** qu'on veut transmettre.

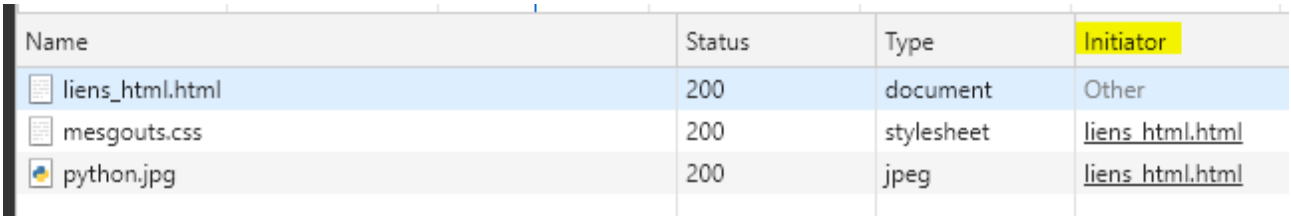
23 – Où le navigateur précise-t-il le nom de domaine par lequel il tente de joindre le serveur ?

Dernière petite manipulation pour les plus curieux : comment le serveur comprend-il qu'il doit envoyer le css, les images ...

24 – Ouvrir la page suivante en utilisant la barre d'adresse de Chrome :
https://formation.infoforall.fr/structure.html

25 – Observer l'ordre des documents demandés par le client. Pourquoi le client commence-t-il par demander le fichier html puis les autres ? Où-trouve-il l'information qui lui précise qu'il a besoin d'un css et d'une image ?

Vous devriez obtenir ceci :



Name	Status	Type	Initiator
liens_html.html	200	document	Other
mesgouts.css	200	stylesheet	liens_html.html
python.jpg	200	jpeg	liens_html.html

Rien de magique :

- le client fait la demande de la page HTML.
- Le client parcourt le HTML qu'il a reçu et commence par le HEAD : il voit qu'on y fait référence à une feuille de style. Du coup, il la demande au serveur. En regardant Initiator, on voit que c'est la « page » elle-même qui en fait la demande de façon autonome.
- Il passe ensuite au BODY. Il voit qu'on y fait référence à une image : il la demande au demande.

Et sur un vrai site, avec un vrai code compliqué ?

26 – Observer les demandes en tapant l'adresse d'un site courant de type moteur de recherche, wikipédia ou autres. Vous risquez d'être surpris par le quantité et le type de requêtes provoquées par votre première requête.

Merci à Hervé Owsinski pour son travail (très peu modifié par JNB)