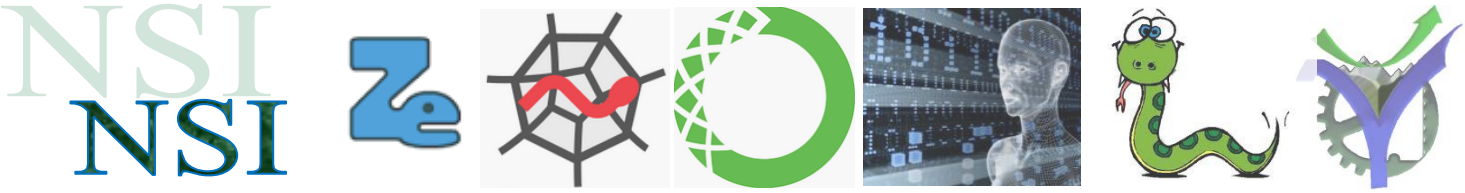


Interactivité, Interface Homme Machine

Résumé :

Nous allons découvrir l'interactivité sur le web réalisée de plusieurs manières. Le but n'est pas de construire un manuel exhaustif d'HTML5, de CSS de JavaScript ni de PHP !! Mais d'articuler, au travers de quelques exemples, comment les uns et les autres fonctionnent et coopèrent ensembles pour apporter de l'interactivité sur le web.

Attachez-vous par-delà ces découvertes à bien comprendre les liens entre ces différentes technologies web.

Pour approfondir vos connaissances, et pourquoi pas pour commencer votre formation de web designer des ressources seront proposées.

Bon travail et bonnes découvertes !

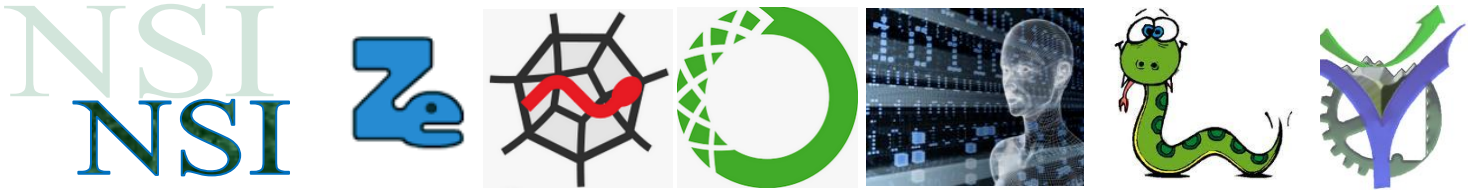
Sommaire :

1 Présentation et mise en place des outils 3
1.1 Un éditeur de texte performant notepad++ 3
1.2 Un serveur Wamp Apache MySQL PHP : uWamp 3
1.3 Un éditeur de fichier hexadécimal : hexedit 3
1.4 Un navigateur Firefox 4
1.5 Quelques aides pour calculer les couleurs 4

2 HTML, CSS en une seule leçon ! 4
2.1 Introduction 4
2.2 Quelques exemples de pages html pour se lancer 5
a) Une page élémentaire 5
b) Vérification automatique de la syntaxe des pages Web 6
c) Un peu plus de balises 7
d) Une page avec du style 8

3 Généralités sur l'interactivité sur le web 11
3.1 L'interactivité coté client - coté serveur 11
3.2 L'éco-conception du Web 12

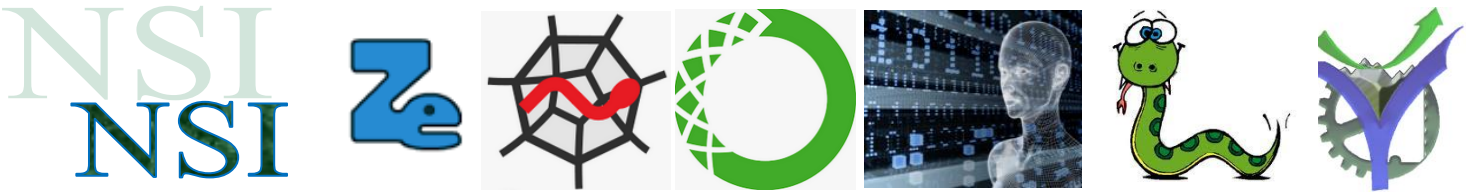
4 Interactivité avec html 14
4.1 Le principe de fonctionnement du formulaire 14
4.2 Exploitation des données avec le serveur 16
a) Envoi des informations avec POST 16
b) Envoi des informations avec GET 17
4.3 Quelques détails dans le fonctionnement des formulaires <form> 18
a) Le fonctionnement du couple <label> <input> 18
b) Les différents types de saisies 18
c) Les boutons input valider et annuler 19
4.4 Un beau formulaire avec une mise en forme css 20
a) Premier exemple 20
b) A faire soi-même 22



5 Javascript 23
5.1 Pourquoi le javascript 23
5.2 Capturer un premier évènement 23
a) Pour débuter 23
b) Un deuxième exemple : 24
c) Quels évènements sont disponibles en HTML5 ? 25
d) Quelques évènements de formulaire 26
e) Avec un exemple d'utilisation 26
5.3 Où placer le code JavaScript 27
a) Code placé dans une balise HTML 27
b) Placer le code JavaScript dans un élément script au sein de la page HTML 28
c) Placer le code JavaScript dans un fichier séparé 28
5.4 Le chargement des pages html vs javascript 30
5.5 Un exemple de formulaire 31
a) Présentation du formulaire 31
b) Application d'une feuille de style 31
c) Pré-traitement de la saisie avec javascript 31

6 Du coté du serveur le PHP 34
6.1 PHP 34
a) Historique 34
b) Pour tester nos exemples en PHP 34
6.2 Un convertisseur °F - °C 35
a) Fonctionnement de l'interface de conversion 35
b) Le programme PHP 36
c) Quelques commentaires 37
6.3 Les variables superglobales de PHP 38
6.4 Le serveur les cookies 38
a) Présentation 38
b) Fonctionnement 39
c) Les cookies et PHP 39
d) Les cookies et la RGPD 40
6.5 Le serveur les sessions 40

7 Ressources, non exhaustives 42
7.1 Sur l'HTML et CSS 42
a) Citons quelques références de sites parmi un grand nombre : 42
b) La gestion des couleurs 42
c) Une activité proposée sur mes sites 42
7.2 Le JavaScript 42
a) Où écrire le code JavaScript 42
b) Les évènements 42
c) Pour tester son code 42
d) Positionnement des éléments et empilement 42
e) Les unités en css px, em, pt … 42
f) Divers 43
7.3 Site Pierre Giraud 43



1 Présentation et mise en place des outils

Afin d'aborder les notions d'interactivité et d'IHM sur le web nous allons présenter quelques outils gratuits nécessaires pour nos travaux. Ces outils seront mis en place avec le premier TP.

1.1 Un éditeur de texte performant notepad++

Cet éditeur de texte est très utile pour l'édition des pages html ou php que nous allons travailler. Il n'ajoute aucune fioriture, permet de terminer chaque ligne de texte par les terminaisons souhaitées.

Il réalise un encodage en UTF8 pour la gestion des caractères accentués et exotiques en conformité avec les normes actuelles.

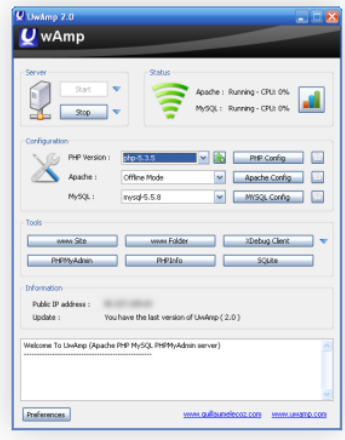
La vignette ci-dessous indiquera l'utilisation de notepad++ pour nos travaux.

1.2 Un serveur Wamp Apache MySQL PHP : uWamp

Pour expérimenter l'interactivité client-serveur il nous faut installer un serveur web. Plusieurs solutions existent.

Nous utiliserons ici une solution complète qui ne nécessite pas d'installation particulière.

On télécharge le fichier archive on le décompresse et on le lance, les processus Apache (serveur, PHP) et MySQL (Base de données) sont opérationnels.

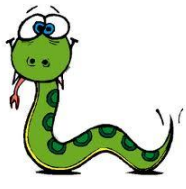


1.3 Un éditeur de fichier hexadécimal : hexedit

Un éditeur de fichier hexadécimal permet une visualisation du binaire d'un fichier quel que soit son type.

Il en existe plusieurs, nous utiliserons le freeware hexedit :

⇦



1.4 Un navigateur Firefox

Pour tester nos pages html, interagir avec les serveurs nous utilisons un programme appelé navigateur.

Pour nous ce sera le navigateur Firefox au célèbre petit logo :

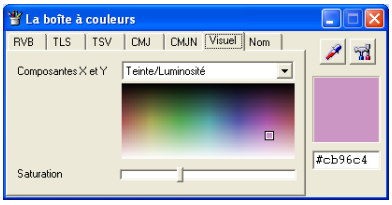
1.5 Quelques aides pour calculer les couleurs

La boite à couleurs :

http://www.commentcamarche.net/download/telecharger-34055480-la-boite-a-couleurs

http://www.colovid.be/ColorBox.htm

https://web-color.aliasdmc.fr/



2 HTML, CSS en une seule leçon !

Rassurez-vous c'est impossible !

2.1 Introduction

Tim Berners-Lee

HTML (**H**yper**T**ext **M**arkup **L**anguage) n'est pas un langage de programmation : **c'est un langage de balisage** qui sert à indiquer au navigateur comment structurer les pages web visitées.

HTML se compose d'une série d'éléments avec lesquels vous pouvez encadrer, envelopper ou baliser différentes parties du contenu pour les faire apparaître ou agir d'une certaine manière. Des balises encadrantes peuvent transformer une petite partie de contenu en un lien vers une autre page sur le Web, mettre des mots en italique, etc……

Sir Tim Berners-Lee[1] est considéré comme l'inventeur du HTML en 1992. Il fut aidé à ses débuts par l'ingénieur et informaticien belge Robert Cailliau qui cosigna notamment avec lui, en novembre 1990, un document désormais entré dans l'Histoire et intitulé « WorldWideWeb : Proposition pour un projet hypertexte ». Depuis 1994, il préside le World Wide Web Consortium (W3C), organisme qu'il a fondé. Il est lauréat du prix Turing 2016.

⇦

[1] Source Wikipédia.

Pour découvrir et approfondir l'emploi des langages html et css il faudra se référer aux cours sur le sujet. Voir quelques références en annexe.

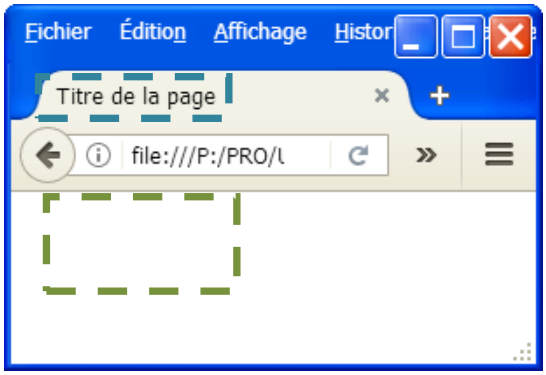
Néanmoins nous allons ici présenter les notions minimales pour pouvoir aborder la suite du travail sur l'interactivité à partir d'une page html de base.

A vous de compléter votre information en fonction de vos besoins et de découvrir par vous-mêmes d'autres aspects, les outils sont gratuits et les ressources foisonnent.

2.2 Quelques exemples de pages html pour se lancer

a) Une page élémentaire[2]

Voilà une première page html et son résultat ci-dessous ouverte dans un navigateur :



```
1  <!doctype html>
2  <html lang="fr">
3      <head>
4          <title>
5              Titre de la page
6          </title>
7          <meta charset="utf-8"/>
8      </head>
9      <body>
10     </body>
11 </html>
```

Ce premier fichier pourra servir de base à vos créations : page_web_de_depart.html

Repérons quelques balises :

<html> </html>

<head> </head> En tête de la page (header)

<body> </body> Corps de la page web

Nous notons que la plupart des balises doivent être fermées avec /

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

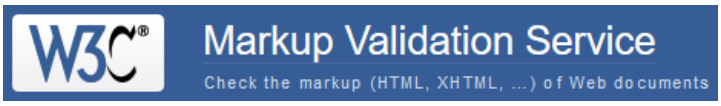
⇦

[2] Réalisé à partir de https://developer.mozilla.org/fr/docs/Apprendre/HTML/ consulté le 13 décembre 2019.

b) Vérification automatique de la syntaxe des pages Web

Nous pouvons faire vérifier la syntaxe de nos pages Web avec des sites spécialisés comme

https://validator.w3.org/



Essayons avec notre fichier :

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for contents of text-input area

Checker Input

Show ☑ source ☐ outline ☐ image report

Options...

Check by text input ☐ CSS

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

1 → **Coller votre Texte**

2 → Check

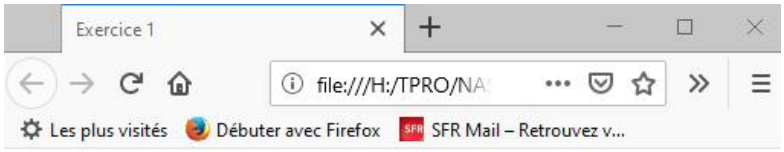
3 → **Document checking completed. No errors or warnings to show.**

⇦

c) Un peu plus de balises

Voilà une page plus complète qui va nous permettre de découvrir un peu plus de mécanismes. Cette page sera analysée en TP.

Les PSoC une technologie d'avenir.html



Les PSoC une technologie d'avenir

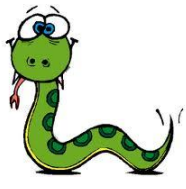
CYPRESS
EMBEDDED IN TOMORROW™

Quelques liens utiles autour des PSoC ...

- Aperçu des applications
- PSoC = Arduino killer
- Cypress PSoC 6: The IoT Problem Solver
- Le véhicule du futur
- Mise en oeuvre des PSoC au lycée Vaucanson

PSoC *are the best*

PSoC6 IoT Winner



d) Une page avec du style

Nous allons suivre en TP les indications données sur le site cité en référence[3] pour obtenir la page html ci-dessous.

Le code contient des instructions html et des instructions css, celles-ci permettent de mettre en forme les pages et de les agrémenter pour obtenir des présentations plus agréables :

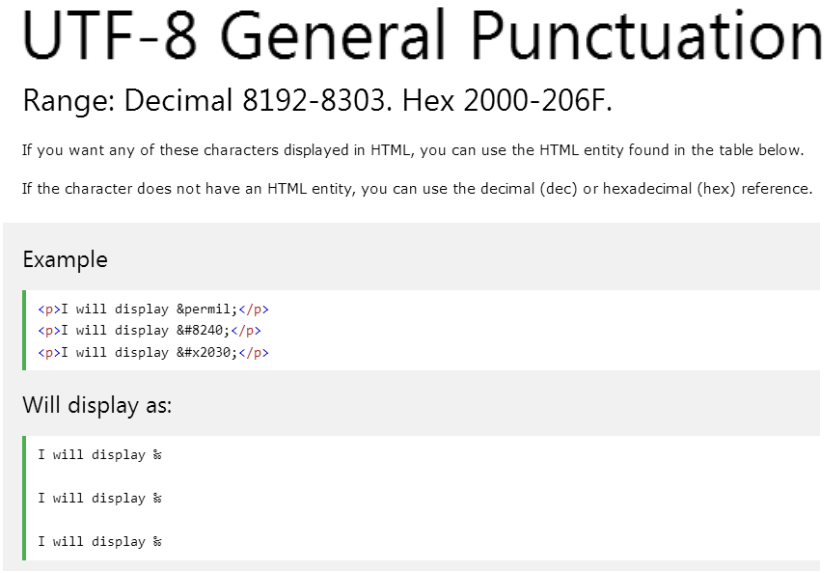
Cascading Style Sheets

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

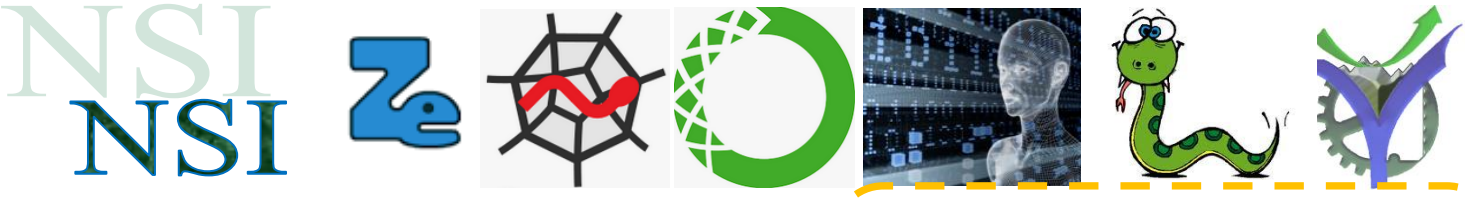
Quelques indications :

> **Note :** Les éléments en HTML sont insensibles à la casse, c'est-à-dire qu'ils peuvent être écrits en majuscules ou en minuscules. Par exemple, un élément `<title>` peut être écrit <title>, <TITLE>, <Title>, <TiTlE>, etc. et il fonctionnera parfaitement. La meilleure pratique, cependant, est d'écrire tous les éléments en minuscules pour des raisons de cohérence, de lisibilité et autres.

Des caractères spéciaux UTF-8 peuvent être ajoutés comme … voir ici.



[3] https://www.w3.org/Style/Examples/011/firstcss.fr.html consulté le 13 décembre 2019.



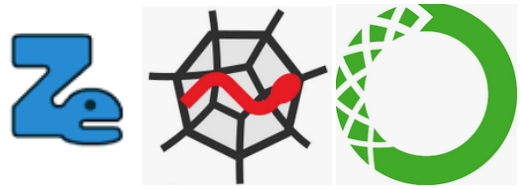
Les fichiers :

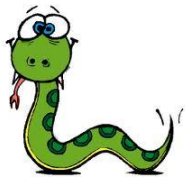
Page_html_css_etape_complet.html
Page_html_css_etape_style_externe.html
monstyle.css

Note : les navigateurs n'interprètent pas les espaces ou saut de lignes. De plus s'ils rencontrent une instruction mal formée, ou bien qu'ils ne reconnaissent pas, elle est simplement ignorée. Contrairement aux langages de programmations qui s'arrêtent en invoquant un ou plusieurs messages d'erreurs.

Le code de la page : Page_html_css_etape_style_externe.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Ma première page avec du style externe</title>
5    <meta charset="utf-8"/>
6    <link rel="stylesheet" href="monstyle.css">
7  </head>
8
9  <body>
10
11 <!-- Menu de navigation du site -->
12 <ul class="navbar">
13   <li><a href="index.html">Home page</a>
14   <li><a href="reflexions.html">Réflexions</a>
15   <li><a href="ville.html">Ma ville</a>
16   <li><a href="liens.html">Liens</a>
17 </ul>
18
19 <!-- Contenu principal -->
20 <h1>Ma première page avec du style externe</h1>
21
22 <p>Bienvenue sur ma page avec du style!
23
24 <p>Il lui manque des images, mais au moins, elle a du style.
25 Et elle a des liens, même s'ils ne mènent nulle part...
26 &hellip;
27
28 <p>Je devrais étayer, mais je ne sais comment encore.
29
30 <!-- Référence au lien d'origine -->
31 <address>D'après https://www.w3.org/Style/Examples/011/firstcss.fr.html<br>
32 </address>
33
34 </body>
35 </html>
```





Et le fichier de style css externe : monstyle.css

```css
body {
    padding-left: 11em;
    font-family: Georgia, "Times New Roman", Times, serif;
    color: purple;
    background-color: #d8da3d }
ul.navbar {
    list-style-type: none;
    padding: 0;
    margin: 0;
    position: absolute;
    top: 2em;
    left: 1em;
    width: 9em }
h1 {
    font-family: Helvetica, Geneva, Arial, SunSans-Regular, sans-serif }
ul.navbar li {
    background: white;
    margin: 0.5em 0;
    padding: 0.3em;
    border-right: 1em solid black }
ul.navbar a {
    text-decoration: none }
a:link {
    color: blue }
a:visited {
    color: purple }
address {
    margin-top: 1em;
    padding-top: 1em;
    border-top: thin dotted }
```

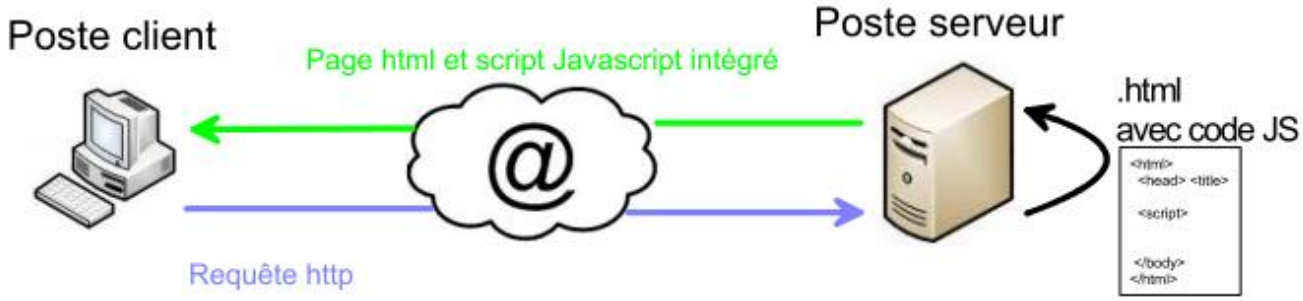
3 Généralités sur l'interactivité sur le web

Nous présentons quelques différentes situations dans lesquelles de l'interactivité est possible dans un site web. Il est important de bien discriminer si l'interaction est du côté client, du côté serveur ou les deux en coopérations.

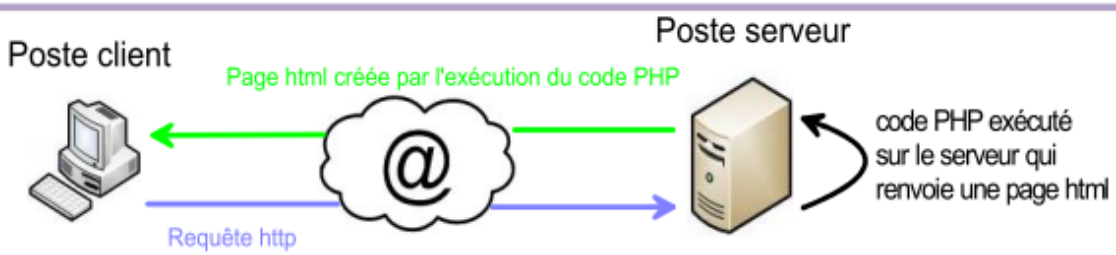
3.1 L'interactivité coté client - coté serveur

Site statique : Le poste client exécute le code html dans son navigateur.

Site dynamique local : Le poste client exécute le code html et JavaScript dans son navigateur.

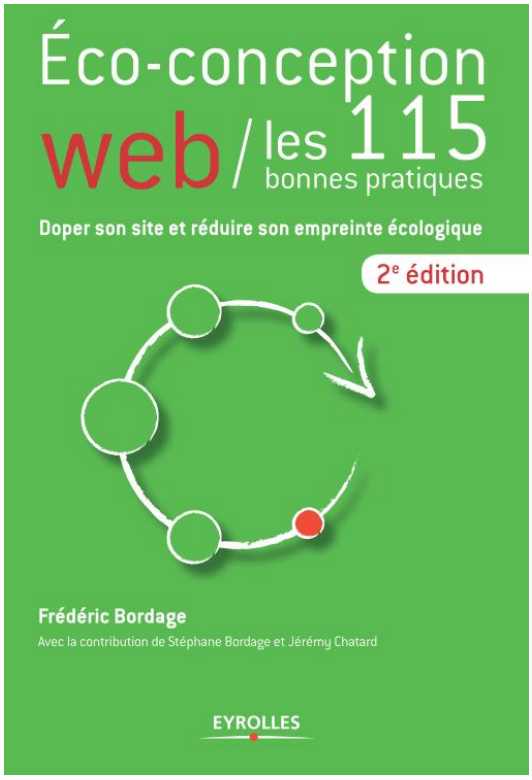


Site dynamique serveur : Le poste client exécute le code html reçu dans son navigateur. Ce code html est issu des instructions exécutées sur le serveur. Ces instructions ne sont pas visibles par le client.



3.2 L'éco-conception du Web

L'éco-conception du Web : et oui l'utilisation du Web n'est pas sans consommer des ressources planétaires. Dans les équipements électroniques consommatrices de matériaux rares, ou bien dans les énergies consommées d'une part dans le refroidissement des datacenters et d'autre part dans le fonctionnement du réseau, l'envoi des messages, les flux de données, etc….. On considère que l'ensemble du 'Cloud' représente la cinquième consommation électrique globale de la planète.
Il est donc urgent d'améliorer ses pratiques, le livre[4] ci-dessous y contribue (12€ format papier, 8,49€ format numérique) voilà le sommaire :



Les 115 bonnes pratiques

Conception 41

FONCTIONNELLE
- Éliminer les fonctionnalités non essentielles 43
- Quantifier précisément le besoin 44
- Fluidifier le processus 45
- Préférer la saisie assistée à l'autocomplétion 46
- Respecter le principe de navigation rapide dans l'historique 47

GRAPHIQUE
- Favoriser un design simple, épuré et adapté au Web 48
- Préférer l'approche « mobile first » ou, à défaut, RESS plutôt que RWD 49

TECHNIQUE
- Proposer un traitement asynchrone lorsque c'est possible 50
- Limiter le nombre de requêtes HTTP 51
- Stocker localement les données statiques 52
- Choisir les technologies les plus adaptées 53
- Utiliser un framework ou développer sur mesure 54
- Limiter le recours aux plug-ins 55
- Limiter l'utilisation de Flash 56

Templating 57

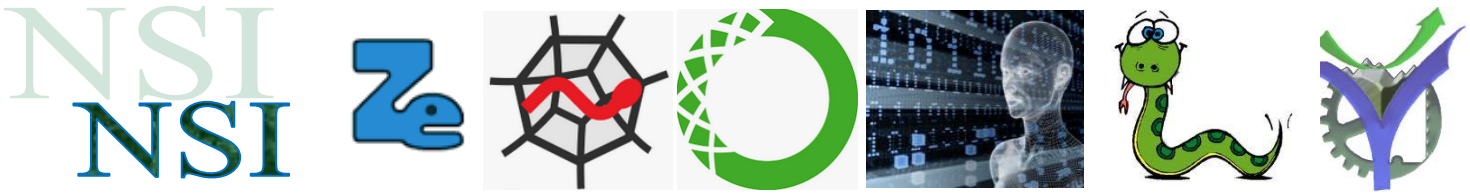
HTML
- Valider les pages auprès du W3C 59
- Externaliser les CSS et JavaScript 60

POLICES
- Favoriser les polices standards 61
- Préférer les glyphes aux images 62

IMAGES
- Supprimer les balises images dont l'attribut SRC est vide 63
- Redimensionner les images en dehors du navigateur 64
- Éviter d'utiliser des images bitmap pour l'interface 65
- Optimiser les images vectorielles 66

CSS
- Générer des spritesheets CSS 67
- Découper les CSS 68
- Limiter le nombre CSS et les compresser 69
- Préférer les CSS aux images 70
- Écrire des sélecteurs CSS efficaces 71
- Grouper les déclarations CSS similaires 72
- Utiliser les notations CSS abrégées 73
- Toujours fournir une CSS print 74
- Utiliser les commentaires conditionnels 75
- Modifier plusieurs propriétés CSS en une seule fois 76

[4] https://www.eyrolles.com/Informatique/Livre/ecoconception-web-les-115-bonnes-pratiques-9782212678062/



Code client 77

JavaScript

Valider le code JavaScript avec JSLint 79
Éviter d'utiliser try...catch...finally 80
Utiliser les opérations primitives 81
Mettre en cache les objets souvent accédés en JavaScript 82
Privilégier les variables locales 83
Privilégier les fonctions anonymes 84
Utiliser le concaténateur de chaînes de façon optimale 85
Préférer les fonctions aux strings, en argument à setTimeout() et setInterval() 86
Éviter les boucles for...in 87

DOM 88

Réduire les accès au DOM via JavaScript 88
Ne pas modifier le DOM lorsqu'on le traverse 89
Rendre les éléments du DOM invisibles lors de leur modification 90
Réduire au maximum le repaint (appearence) et le reflow (layout) 91
Utiliser la délégation d'événements 92

Animations

Privilégier les changements visuels instantanés 93
Éviter les animations Javascript/CSS coûteuses 94

Échanges de données

Utiliser Ajax pour les zones de contenu souvent mises à jour 95
Utiliser la méthode GET pour les requêtes Ajax 96

Hébergement 125

Ressources et contenu

Optimiser les images bitmap 127
N'utiliser que les portions indispensables des librairies JavaScript et CSS 128
Minifier les fichiers CSS 129
Minifier les fichiers JavaScript 130
Compresser les feuilles de style CSS et les bibliothèques JavaScript 131
Combiner les fichiers CSS et les fichiers JavaScript 132
Optimiser la taille des cookies 133
Compresser la sortie HTML 134

Infrastructure physique

Choisir un hébergeur « vert » 135
Utiliser une électricité « verte » 136
Adapter la qualité de service et le niveau de disponibilité 137
Utiliser des serveurs virtualisés 138
Optimiser l'efficacité énergétique des serveurs 139
Installer uniquement les services indispensables sur le serveur 140
Monter les caches entièrement en RAM 141
Privilégier les serveurs équipés de mémoires SSD 142
Stocker les données dans le cloud 143
Désactiver les logs binaires de MySQL ou MariaDB 144

Infrastructure logicielle

Utiliser un serveur asynchrone 145
Limiter le recours aux certificats SSL 146
Héberger les ressources sur un domaine sans cookies 147

Code serveur 97

Conception

Favoriser les pages statiques 99
Créer une architecture applicative modulaire 100
Utiliser certains forks applicatifs orientés « performance » 101
Choisir un format de données adapté 102
Limiter le nombre de domaines servant les ressources 103

CMS

Utiliser un moteur de templating 104
Utiliser tous les niveaux de cache du CMS 105
Générer les PDF en dehors du CMS 106
Redimensionner les images en dehors du CMS 107
Encoder les sons en dehors du CMS 108

Serveur d'applications

Mettre en cache le bytecode 109
Mettre en cache les données calculées souvent utilisées 110
Libérer de la mémoire les variables qui ne sont plus nécessaires 111
Ne pas appeler de fonction dans la déclaration d'une boucle de type for 112
Supprimer tous les warnings et toutes les notices 113
Utiliser des variables statiques 114
Éviter la réécriture des getter/setter natifs 115
Ne pas assigner inutilement de valeurs aux variables 116
Utiliser la simple quote (') au lieu du guillemet («) 117
Remplacer les $i++ par des ++$i 118

Base de données

Éviter d'effectuer des requêtes SQL à l'intérieur d'une boucle 119
Ne se connecter à une base de données que si nécessaire 120
Ne jamais écrire de SELECT * FROM 121
Limiter le nombre de résultats 122
Utiliser les procédures stockées 123
Éviter les redirections 148
Ne pas générer de page 404 149
Désactiver certains logs d'accès du serveur web 150
Désactiver le DNS Lookup d'Apache 151
Désactiver la directive AllowOverride d'Apache 152

Cache

Utiliser un CDN 153
Utiliser un reverse proxy 154
Mettre en cache le favicon.ico 155
Ajouter des en-têtes Expires ou Cache-Control 156
Utiliser les ETags 157
Mettre en cache les réponses Ajax 158

4 Interactivité avec html

4.1 Le principe de fonctionnement du formulaire

Voilà un formulaire très simple organisé avec les trois fichiers ci-dessous :

php-exemple.css	*Le style*
php-exemple.html	*Le code html*
php-exemple.php	*Le code PHP exécuté sur le serveur*

Le résultat obtenu, sans mise en forme sophistiquée :

Dites un mot !! Bonjour

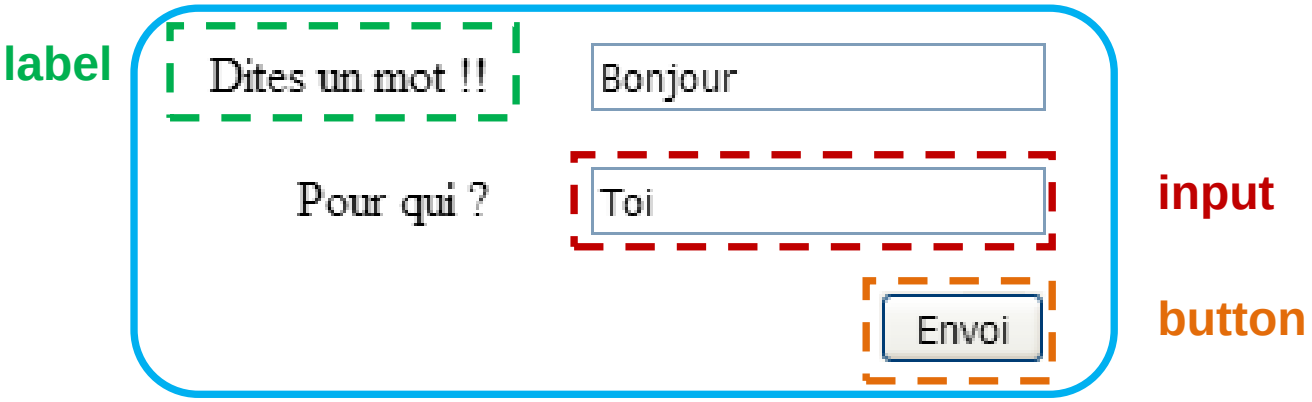
Pour qui ? Toi

Envoi

Voilà le fichier css utilisé :

```
form {
    width: 420px;
}
div {
    margin-bottom: 20px;
}
label {
    display: inline-block;
    width: 240px;
    text-align: right;
    padding-right: 10px;
}
button, input {
    float: right;
}
```

Les différents éléments ou widgets (gadgets) utilisés :



Nous y voyons quelques éléments, ou widgets, d'interface à l'œuvre :

```
1  <!DOCTYPE html>
2  <!-- Fichier php-exemple.html -->
3  <html>
4    <head>
5      <meta charset="utf-8">
6      <title>PHP soumission de données avec Form</title>
7      <link rel="stylesheet" href="php-exemple.css" media="all">
8    </head>
9    <body>
10     <form method="post" action="php-exemple.php">
11       <div>
12         <label for="mot">Dites un mot !!</label>
13         <input name="mot" id="mot" value="Bonjour">
14       </div>
15       <div>
16         <label for="pour">Pour qui ?</label>
17         <input name="pour" value="Toi">
18       </div>
19       <div>
20         <button>Envoi</button>
21       </div>
22     </form>
23   </body>
24 </html>
```

Interface de type champ de saisie texte

Interface de type bouton de validation

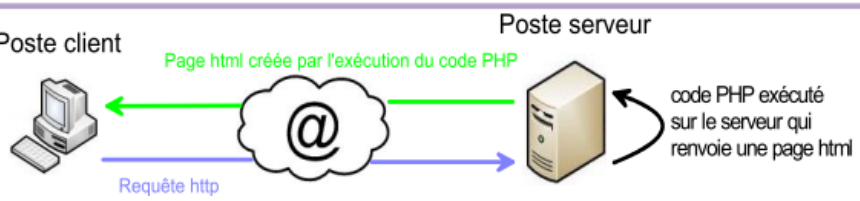
4.2 Exploitation des données avec le serveur

Le principe du fonctionnement est rappelé ci-dessous :

Site dynamique serveur : Le poste client exécute le code html reçu dans son navigateur. Ce code html est issu des instructions exécutées sur le serveur. Ces instructions ne sont pas visibles par le client.

Coté client ou front-end

HTML, CSS, JavaScript



Coté serveur ou back-end

PHP, Python, .NET, Java, SQL

La balise de formulaire précise le mode d'envoi des informations (GET ou POST) au serveur et l'action à réaliser (script PHP) pour exploiter celles-ci :

```
<form method="post" action="php-exemple.php">
```

a) Envoi des informations avec POST

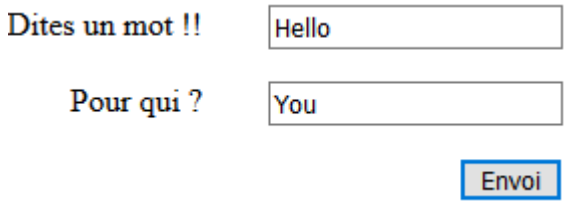
Dans ce cas les informations collectées par le formulaire sont envoyées au serveur dans le corps de la requête HTTP. Elles sont donc invisibles car elles ne sont pas dans l'URL.

Mais attention elles ne sont pas cryptées pour autant.

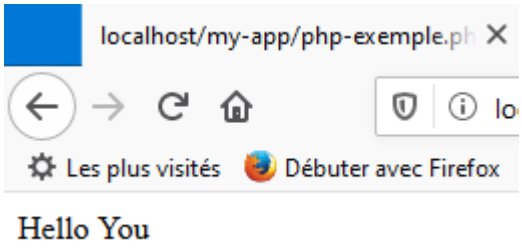
Voilà le contenu de php-exemple.php :

```php
<?php
  // La variable globale $_POST contient les valeurs accessibles avec leurs noms.
  // Dans le cas d'une utilisation de la méthode GET il faut utiliser $_GET

  $variable_mot = htmlspecialchars($_POST['mot']);

  $variable_pour  = htmlspecialchars($_POST['pour']);

  echo  $variable_mot, ' ', $variable_pour;

?>
```

On obtient ce fonctionnement :

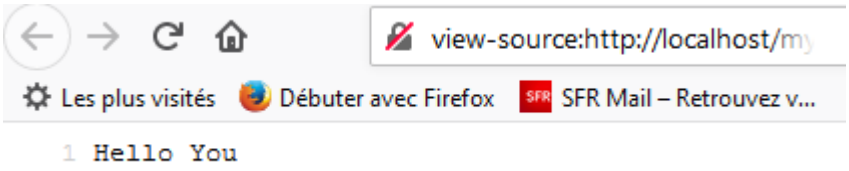


Formulaire html envoi des données



réponse du serveur

A noter que le code PHP exécuté par le serveur uWamp ne renvoi que de l'html. Si on observe le code source de la page web obtenue en réponse nous n'observons que les deux mots saisis dans les champs du formulaire :



b) Envoi des informations avec GET

Deuxième solution pour l'envoi de nos données de formulaire la méthode GET :

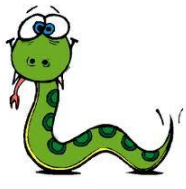
```
<form method="get" action="php-exemple.php">
```

Nous observons alors que les informations sont visibles dans l'URL envoyées au serveur, identifiées par leurs noms attribut name :

```
<label for="mot">Dites un mot !!</label>
<input name="mot" id="mot" value="Bonjour">

<label for="pour">Pour qui ?</label>
<input name="pour" value="Toi">
```

Cette méthode ne devra pas être utilisée pour des informations sensibles.

Néanmoins elle a son utilité pour un site commercial ou d'accès à de la connaissance, type Wikipédia, car dans ce cas on peut enregistrer l'URL correspondant à une recherche particulière pour pouvoir la réutiliser ensuite.



4.3 Quelques détails dans le fonctionnement des formulaires <form>

a) Le fonctionnement du couple <label> <input>[5]

Prenons l'un de nos champs de saisie comme exemple :

```
<label for="mot">Dites un mot !!</label>
<input name="mot" id="mot" value="Bonjour">
```

L'élément HTML `<label>` représente une légende pour un objet d'une interface utilisateur. Il peut être associé à un contrôle en utilisant l'attribut `for` ou en plaçant l'élément du contrôle à l'intérieur de l'élément `<label>`. Un tel contrôle est appelé *contrôle étiqueté* par l'élément `<label>`.

Rattacher un libellé à un élément de saisie (`<input>`) offre différents avantages :

Le texte du libellé n'est pas seulement associé visuellement au champ, il est *techniquement* associé avec le champ. Ainsi, lorsque l'utilisateur a le focus sur le champ, un lecteur d'écran pourra énoncer le contenu du libellé et permettre à l'utilisateur de disposer d'un meilleur contexte.

> Un **lecteur d'écran** (également appelé *revue d'écran*) est un logiciel d'assistance technique destiné aux personnes « empêchées de lire » (aveugles, fortement malvoyantes, dyslexiques, dyspraxiques…) : il retranscrit par synthèse vocale et/ou sur un afficheur braille ce qui est affiché sur l'écran d'un ordinateur tant en termes de contenu que de structure et permet d'interagir avec le système d'exploitation et les logiciels applications. (Source Wikipédia)

Vous pouvez cliquer sur le libellé pour passer le focus voire activer le champ. De cette façon, on dispose d'une meilleure ergonomie car la surface d'utilisation du champ est agrandie, ce qui s'avère utile sur les petits appareils comme les téléphones portables.

Pour associer un élément `<label>` avec un élément `<input>`, il faut fournir un identifiant à l'élément `<input>` sous la forme d'un attribut `id`. Ensuite, on peut renseigner l'attribut `for` de l'élément `<label>` avec la valeur de cet identifiant.

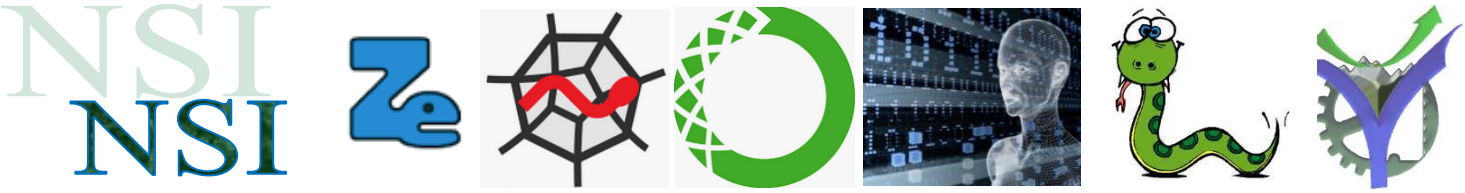
b) Les différents types de saisies[6]

L'élément HTML **`<input>`** est utilisé pour créer un contrôle interactif dans un formulaire web qui permet à l'utilisateur de saisir des données. Les saisies possibles et le comportement de l'élément `<input>` dépend fortement de la valeur indiquée dans son attribut `type`.

La façon dont un élément `<input>` fonctionne dépend grandement de la valeur de son attribut `type`. Aussi, pour chacun de ces types, on aura une page de référence dédiée.

Par défaut, lorsque l'attribut `type` n'est pas présent, il aura la valeur implicite `text`.

[5] https://developer.mozilla.org/fr/docs/Web/HTML/Element/Label consulté le 17 décembre 2019
[6] https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input



Quelques types de champs disponibles :

- `button` : un bouton sans comportement défini.
- `checkbox` : une case à cocher qui permet de sélectionner/désélectionner une valeur
- `color` : HTML5 un contrôle qui permet de définir une couleur.
- `date` : HTML5 un contrôle qui permet de saisir une date (composé d'un jour, d'un mois et d'une année).
- `datetime-local` : HTML5 un contrôle qui permet de saisir une date et une heure (sans fuseau horaire).
- `email` : HTML5 un champ qui permet de saisir une adresse éléctronique.
- `file` : un contrôle qui permet de sélectionner un fichier. L'attribut **`accept`** définit les types de fichiers qui peuvent être sélectionnés.
- `number` : HTML5 un contrôle qui permet de saisir un nombre.
- `password` : un champ texte sur une seule ligne dont la valeur est masquée. Les attributs `maxlength` et `minlength` définissent la taille maximale et minimale de la valeur à saisir dans le champ.

Note : Tout formulaire comportant des données sensibles doit être servi via HTTPS. Les navigateurs alertent les utilisateurs lorsque les formulaires avec de telles données sont uniquement disponibles via HTTP.

- `radio` : un bouton radio qui permet de sélectionner une seule valeur parmi un groupe de différentes valeurs.
- `reset` : un bouton qui réinitialise le contenu du formulaire avec les valeurs par défaut.
- `submit` : un bouton qui envoie le formulaire.
- `tel` : HTML5 un contrôle pour saisir un numéro de téléphone.
- `text` : un champ texte sur une seule ligne. Les sauts de ligne sont automatiquement retirés.

Nous verrons l'usage de quelques-uns de ces champs en TP.

c) Les boutons input valider et annuler

Pour envoyer les données saisies dans un formulaire il suffit d'utiliser le bouton `submit` ou le bouton `reset` pour revenir à l'état initial.

A noter que si la valeur n'est pas précisée c'est le choix `submit` par défaut.

Un exemple d'utilisation sera donné dans le paragraphe suivant.

4.4 Un beau formulaire avec une mise en forme css[7]

a) Premier exemple

Les fichiers sont présentés ci-dessous :

Formulaire_exemple_1.html

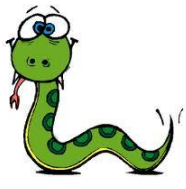
style_exemple_1.css

```
<fieldset>
 <legend> A propos des CSS : </legend>
   <p>Savez-vous ce que veut dire CSS ? : </p>
     <label for="oui" class="inline">oui</label>
     <input type="radio" name="CSS" value="oui" id="oui"
     checked="checked" />
     <label for="non" class="inline">non</label>
     <input type="radio" name="CSS" value="non" id="non" />

  <label for="utilise">Si oui, les utilisez-vous plutôt : </label>
   <select name="utilise" id="utilise">
   <option value="toujours"> toujours</option>
   <option value="parfois"> parfois</option>
   <option value="jamais"> jamais</option>
   </select>
</fieldset>
```

```
<fieldset>
 <legend>Vos coordonnées :</legend>
  <label for="email">Votre email :</label>
   <input type="text" name="email" size="20"
   maxlength="40" value="email" id="email" />

  <label for="comments">Vos commentaires :</label>
   <textarea name="comments" id="comments" cols="20" rows="4">
   </textarea>
</fieldset>
```

```
<p>
<input type="submit" value="Envoyer" />
<input type="reset" value="Annuler" />
</p>
```

[7] D'après http://css.mammouthland.net/formulaire_css.php consulté le 16 décembre 2019.

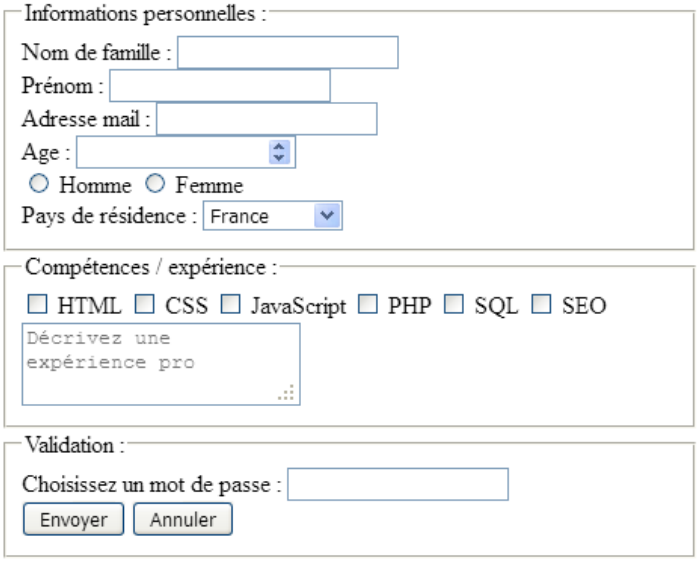


Le fichier css :

```css
body {
  font-family:"trebuchet ms",sans-serif;
  font-size:90%;
  }
form {
  background-color:#FAFAFA;
  padding:10px;
  width:280px;
  }
fieldset {
  padding:0 20px 20px 20px;
  margin-bottom:10px;
  border:1px solid #DF3F3F;
  }
legend {
  color:#DF3F3F;
  font-weight:bold
  }
label {
  margin-top:10px;
  display:block;
  }
label.inline {
  display:inline;
  margin-right:50px;
  }
input, textarea, select, option {
  background-color:#FFF3F3;
  }
input, textarea, select {
   padding:3px;
   border:1px solid #F5C5C5;
   border-radius:5px;
   width:200px;
   box-shadow:1px 1px 2px #C0C0C0 inset;
  }
select {
   margin-top:10px;
  }
input[type=radio] {
  background-color:transparent;
  border:none;
  width:10px;
  }
input[type=submit], input[type=reset] {
  width:100px;
  margin-left:5px;
  box-shadow:1px 1px 1px #D83F3D;
  cursor:pointer;
  }
input:focus, textarea:focus {
  background-color:white;
  }
input[type=submit]:focus, input[type=reset]:focus {
  background-color:#FFF3F3;
}
 input[type=submit]:hover, input[type=reset]:hover
{
 background-color:#FCDEDE;
}
input[type=submit]:active, input[type=reset]:active {
  background-color:#FCDEDE;
  box-shadow:1px 1px 1px #D83F3D inset;
}
textarea {
 /* Empêcher le redimensionnement */
   resize: none;
}
```

b) A faire soi-même

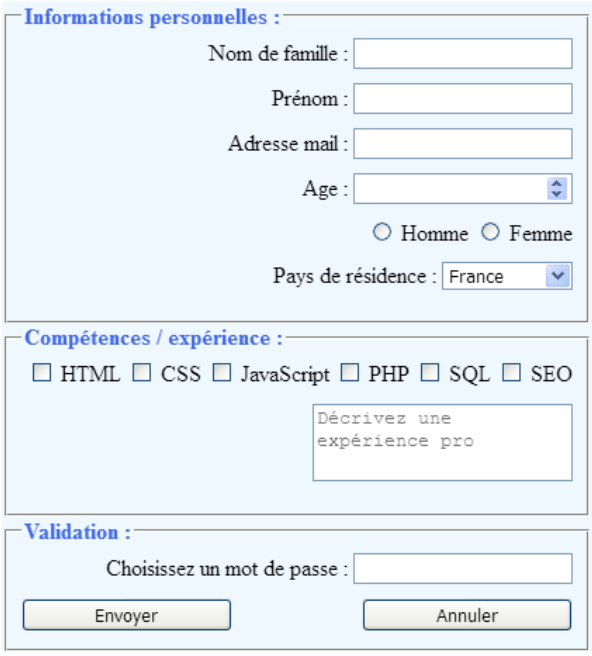
A partir du fichier html : Formulaire_complexe.html



Créer le fichier cours.css pour obtenir le résultat le plus proche de celui présenté ci-dessous :

Quelques propriétés à utiliser :

```
font-weight:
margin-bottom:
text-align:
width:
float:
color:
background-color:
```



Pour vous aider vous pouvez compléter le fichier ci-dessous en le renommant cours.css :

cours_a_completer.css

5 Javascript

5.1 Pourquoi le javascript[8]

Le JavaScript est un langage de programmation créé en 1995 par Brendan Eich[9]. Il a été standardisé sous le nom d'ECMAScript en juin 1997 par ECMA International dans le standard ECMA-262. Le standard ECMA-262 en est actuellement à sa 8e édition. Le groupe ECMA International se charge de publier les standards de ce langage.

Le JavaScript est aujourd'hui l'un des langages de programmation les plus populaires et il fait partie des langages web dits « standards » avec le HTML et le CSS.

On dit que le HTML, le CSS et le JavaScript sont des standards du web car les principaux navigateurs web (Google Chrome, Safari, Firefox, etc.) savent tous « lire » (ou « comprendre » ou « interpréter ») ces langages et les interprètent généralement de la même façon ce qui signifie qu'un même code va généralement produire le même résultat dans chaque navigateur.

Pour définir ce qu'est le JavaScript et le situer par rapport aux autres langages, et donc pour comprendre les intérêts et usages du JavaScript il faut savoir que :

- Le JavaScript est un langage dynamique ;
- Le JavaScript est un langage (principalement) côté client ;
- Le JavaScript est un langage interprété ;
- Le JavaScript est un langage orienté objet.

Brendan Eich

Brendan Eich en 2012.

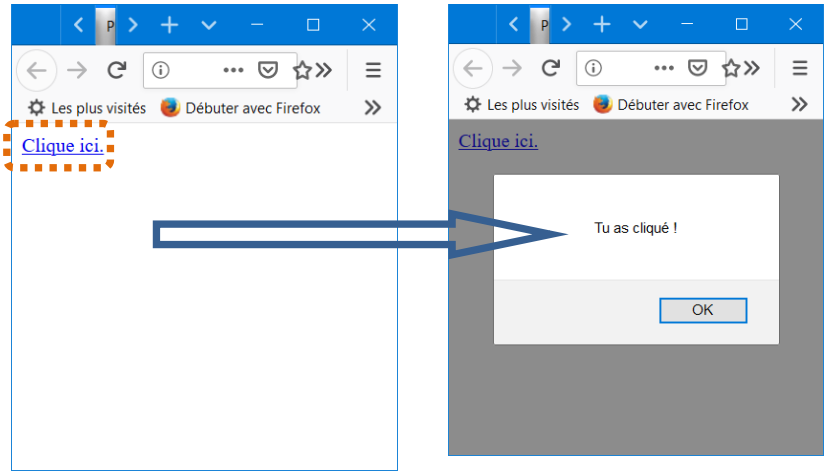
Fonctions

Président-directeur général (en)
Brave Software (d)
depuis le 28 mai 2015

Président-directeur général (en)
Mozilla Corporation
24 mars - 3 avril 2014

5.2 Capturer un premier évènement

a) Pour débuter



La notion d'évènement est très importante. La programmation javascript consiste à mettre en place le comportement attendu de notre interface en réponse à des interactions avec la souris ou bien le clavier. Par exemple :

onclick : clic de souris

ondblclick : double clic

onmouseover : la souris est sur l'élément

[8] Merci au site https://www.pierre-giraud.com/javascript-apprendre-coder-cours/

[9] Source Wikipédia.

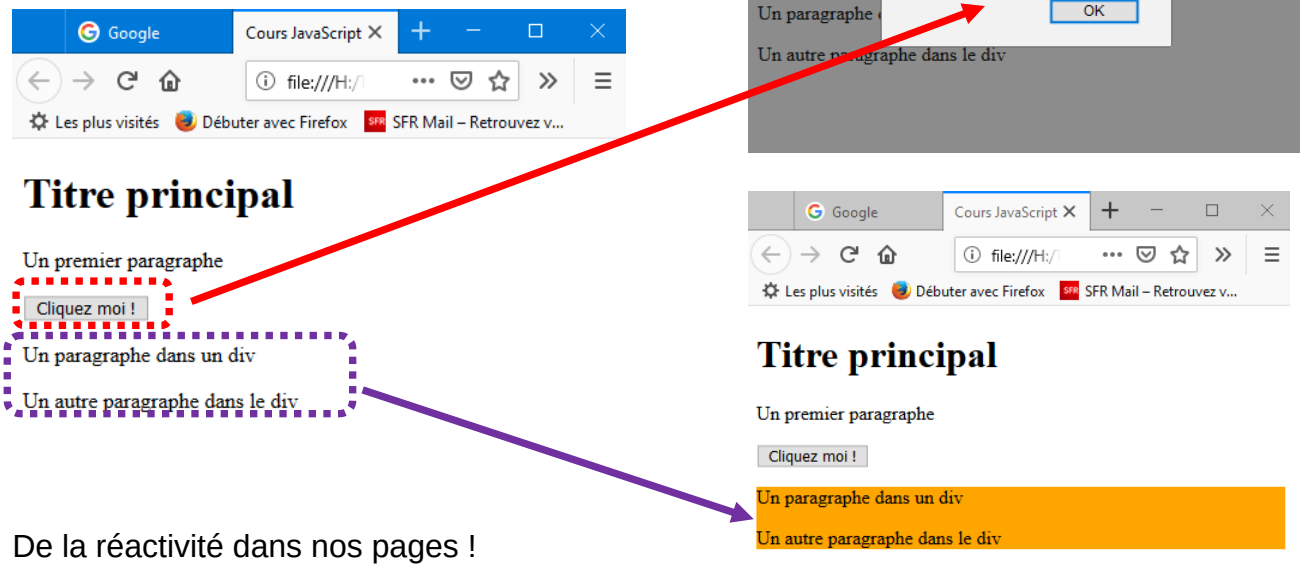
Le code : exemple_1_javascript.html

```
<!doctype html>
<html lang="fr">
    <head>
        <title>
            Premier essai JavaScript
        </title>
        <script>
            function popup(){ alert('Tu as cliqué !') }
        </script>
        <meta charset="utf-8"/>
    </head>
    <body>
        <a href="javascript:popup()">Clique ici.</a>
    </body>
</html>
```

La balise html <a> créé un lien hypertexte donc ici appelle la fonction javascript:popup() qui affiche le message en invoquant la fonction alert.

b) <u>Un deuxième exemple[10] :</u>

exemple_2_javascript.html



De la réactivité dans nos pages !

Le code utilisé :

[10] https://www.pierre-giraud.com/javascript-apprendre-coder-cours/addeventlistener-gestion-evenement/

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
    </head>

    <body>
        <h1>Titre principal</h1>
        <p>Un premier paragraphe</p>
        <button onclick="alert('Bouton cliqué')">Cliquez moi !</button>
        <div onmouseover="this.style.backgroundColor='orange'"
            onmouseout="this.style.backgroundColor='white'">
            <p>Un paragraphe dans un div</p>
            <p>Un autre paragraphe dans le div</p>
        </div>
    </body>
</html>
```

c) Quels évènements sont disponibles en HTML5 ?

Il existe beaucoup d'évènements disponibles en particulier avec la version 5 d'HTML. Ils sont répartis en plusieurs catégories :

- Évènement de fenêtre – Balise <body>
  Événements déclenchés pour l'objet window – fenêtre. Ils s'appliquent généralement à la balise <body>
- Évènement de curseur / souris
  Événements déclenchés par la souris, son curseur, ses boutons ou sa molette – ou action similaire. Ils s'appliquent à tous les éléments HTML5.
- Évènement de formulaire
  Événements déclenchés par des actions dans un formulaire. Ils s'appliquent à tous les éléments HTML5, mais plus particulièrement dans un formulaire.
- Évènement multimédia
  Événements déclenchés par les médias tels que les vidéos, les images ou les fichiers audio. Ils s'appliquent à tous les éléments HTML5, mais plus particulièrement aux éléments des médias, comme audio, intégrer, img, objet, vidéo.
- Évènement du clavier
  Événements déclenchés par une touche du clavier ou une action similaire de l'utilisateur. Ils s'appliquent à tous les éléments HTML5.

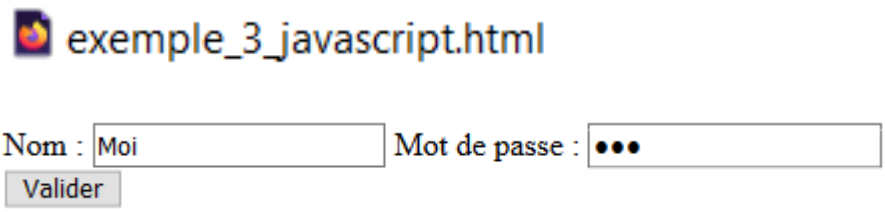
Une excellente synthèse est disponible ici :

http://41mag.fr/liste-des-balises-html5/liste-des-evenements-html5

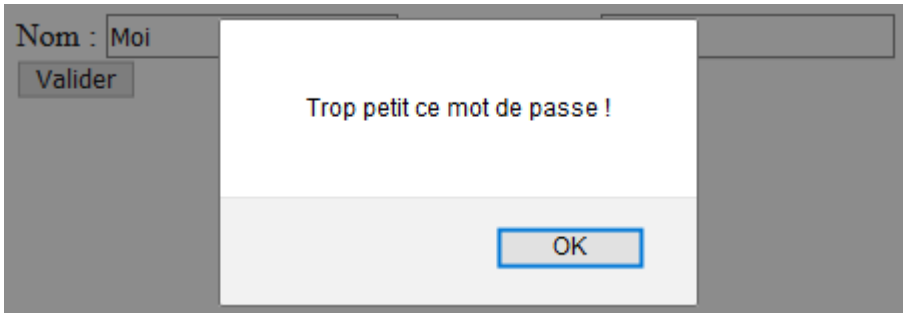
d) Quelques évènements de formulaire

Attribut	Description
onblur	Le script démarre lorsque l'élément, le champ du formulaire, perd le focus
onchange	Script à exécuter lorsque un élément est modifié
oncontextmenu New	Script à exécuter quand un menu contextuel est déclenché
onfocus	Script à exécuter quand un élément gagne le focus
onformchange New	Script à exécuter lorsqu'un formulaire est modifié
onforminput New	Script à exécuter lors de la saisie d'un formulaire
oninput New	Script à exécuter lors de la saisie d'un élément
oninvalid New	Script à exécuter quand un élément n'est pas valide
onselect	Script à exécuter quand un élément est sélectionné
onsubmit	Script à exécuter lors de la soumission d'un formulaire

e) Avec un exemple d'utilisation



Dès que l'on quitte le formulaire : on perd le focus et l'analyse du mot de passe est déclenchée.



Le code utilisé, notez l'évènement onblur qui invoque la fonction qui vérifie que la chaine val contient au moins 9 caractères.

```
<!-- D'après interro des lycées 1ère NSI Nathan -->
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
    <script>
        function controle(val) {
            if (val.length<8)
            alert('Trop petit ce mot de passe !')
            }
    </script>
    </head>

    <body>
        <form>
            <label for='nom'>Nom :</label>
            <input type='text' id='nom'/>
            <label for='mdp'>Mot de passe :</label>
            <input type='password' id='mdp' onblur='controle(this.value)' />
            <input type='submit' value='Valider' />
        </form>
    </body>
</html>
```

Avant d'expérimenter d'autres fonctions et évènements nous allons examiner où mettre le code javascript dans nos pages en corrélation avec le chargement de celles-ci dans les navigateurs.

5.3 Où placer le code JavaScript[11]

On va pouvoir placer du code JavaScript à trois endroits différents :

Directement dans la balise ouvrante d'un élément HTML ;

Dans un élément script, au sein d'une page HTML ;

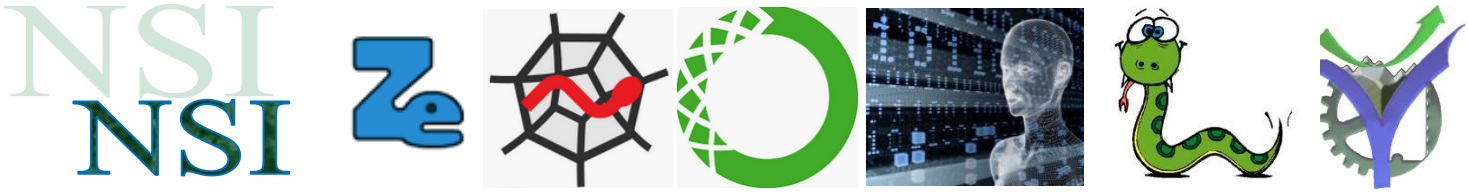
Dans un fichier séparé contenant exclusivement du JavaScript et portant l'extension .js.

a) Code placé dans une balise HTML

Ici, on crée deux boutons en HTML et on place nos codes JavaScript à l'intérieur d'attributs onclick. Le code à l'intérieur des attributs va s'exécuter dès qu'on va cliquer sur le bouton correspondant. Dans le cas présent, cliquer sur le premier bouton a pour effet d'ouvrir une fenêtre d'alerte qui affiche « Bonjour ! ». Cliquer sur le deuxième bouton rajoute un élément p qui contient le texte « Paragraphe ajouté » à la suite des boutons.

⇦

[11] https://www.pierre-giraud.com/javascript-apprendre-coder-cours/ou-ecrire-code-javascript/ reproduit avec autorisation de l'auteur.



exemple_4_javascript.html

Titre principal

Cliquez moi | Ajouter un paragraphe

Paragraphe ajouté

Paragraphe ajouté

Paragraphe ajouté

Paragraphe ajouté

```
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
        <meta name="viewport"
          content="width=device-width, initial-scale=1, user-scalable=no">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <button onclick="alert('Bonjour !')">
            Cliquez moi
        </button>

        <button onclick="(function(){
            let para = document.createElement('p');
            para.textContent = 'Paragraphe ajouté';
            document.body.appendChild(para);
            })();">
            Ajouter un paragraphe
        </button>
```

Aujourd'hui, de nouvelles techniques nous permettent de ne plus utiliser ce genre de syntaxe et il est généralement déconseillé et considéré comme une mauvaise pratique d'écrire du code JavaScript dans des balises ouvrantes d'éléments HTML.

b) Placer le code JavaScript dans un élément script au sein de la page HTML

Nous avons déjà utilisé cette technique dans les exemples précédents.

On va également pouvoir placer notre code JavaScript dans un élément script qui est l'élément utilisé pour indiquer qu'on code en JavaScript.

On va pouvoir placer notre élément script n'importe où dans notre page HTML, aussi bien dans l'élément `head` qu'au sein de l'élément `body`.

De plus, on va pouvoir indiquer plusieurs éléments script dans une page HTML pour placer plusieurs bouts de code JavaScript à différents endroits de la page.

exemple_5_javascript.html

Tout d'abord, comme précédemment, la séparation des codes n'est pas optimale ici puisqu'on mélange du JavaScript et du HTML ce qui peut rendre l'ensemble confus et complexe à comprendre dans le cadre d'un gros projet.

De plus, si on souhaite utiliser les mêmes codes sur plusieurs pages, il faudra les copier-coller à chaque fois ce qui n'est vraiment pas efficient et ce qui est très mauvais pour la maintenabilité d'un site puisque si on doit changer une chose dans un code copié-collé dans 100 pages de notre site un jour, il faudra effectuer la modification dans chacune des pages.

c) Placer le code JavaScript dans un fichier séparé

Le code JavaScript peut se placer dans un fichier externe.

exemple_6.js

exemple_6_javascript.html

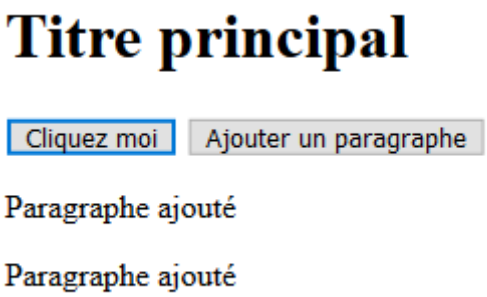
Voilà le code :

exemple_6_javascript.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours JavaScript</title>
        <meta charset="utf-8">
        <meta name="viewport"
          content="width=device-width, initial-scale=1, user-scalable=no">
        <script src='exemple_6.js' async></script>
    </head>

    <body>
        <h1>Titre principal</h1>
        <button id='b1'>Cliquez moi</button>
        <button id='b2'>Ajouter un paragraphe</button>
    </body>
</html>
```

exemple_6.js

```
let bonjour = document.getElementById('b1');
let ajouter = document.getElementById('b2');

bonjour.addEventListener('click', alerte);
ajouter.addEventListener('click', ajout);

function alerte(){
    alert('Bonjour');
    }

function ajout(){
    let para = document.createElement('p');
    para.textContent = 'Paragraphe ajouté';
    document.body.appendChild(para);
    }
```



Cette méthode est la meilleure puisqu'elle permet une excellente séparation du code et une maintenabilité optimale de celui-ci. En effet, si on veut insérer le code JavaScript contenu dans notre fichier dans 100 pages différentes, il suffira ici d'appeler ce fichier JavaScript dans les 100 pages. En cas de modification du code, il suffira alors de le modifier une fois dans le fichier JavaScript.

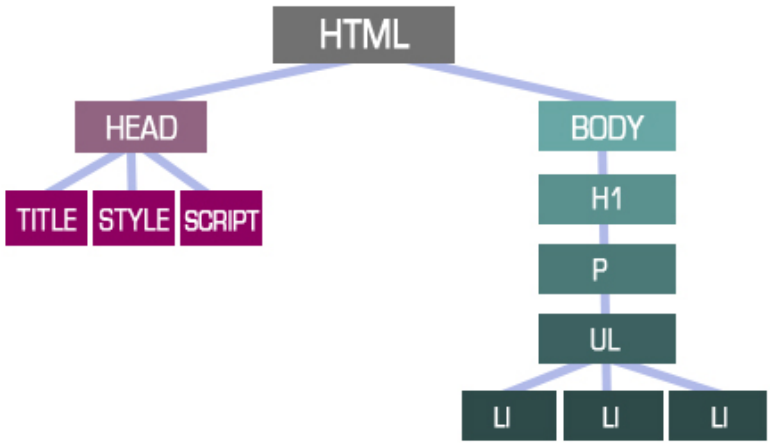
5.4 Le chargement des pages html vs javascript

Il faut faire attention à la manière dont les pages HTML sont chargées et interprétées dans le navigateur par défaut, un navigateur va lire et exécuter le code dans l'ordre de son écriture[12].

Plus précisément, lorsque le navigateur arrive à un élément script, il va stopper le traitement du reste du HTML jusqu'à ce que le code JavaScript soit chargé dans la page et exécuté.

Par ailleurs lors du chargement d'une page HTML une description de celle-ci est constituée au fur et à mesure. Cette représentation est une interface indépendante de tout langage de programmation le DOM pour ***D***ocument ***O***bject ***M***odel.

Ce DOM[13] peut ensuite être analysé et traité, comme par exemple l'ajout de paragraphes dans nos exemples précédents.



Le code JavaScript accède au DOM avec la fonction `getElementById('b1')` qui permet d'identifier un élément dans l'arbre du DOM grâce à son `id`.

Le problème dans le chargement des pages consiste à ne pas tenter d'exécuter des fonctions JavaScript alors que le contenu de la page n'est pas totalement chargé d'une part et de permettre un chargement en 'temps masqué' du code javascript sans bloquer la lecture du code HTML par une analyse prématurée si celui-ci est placé en début du code d'autre part.

C'est le rôle du téléchargement asynchrone, donc non bloquant demandé dans notre exemple :

```
<script src='exemple_6.js' async></script>
```

⇦

[12] Voir https://www.pierre-giraud.com/javascript-apprendre-coder-cours/ou-ecrire-code-javascript/

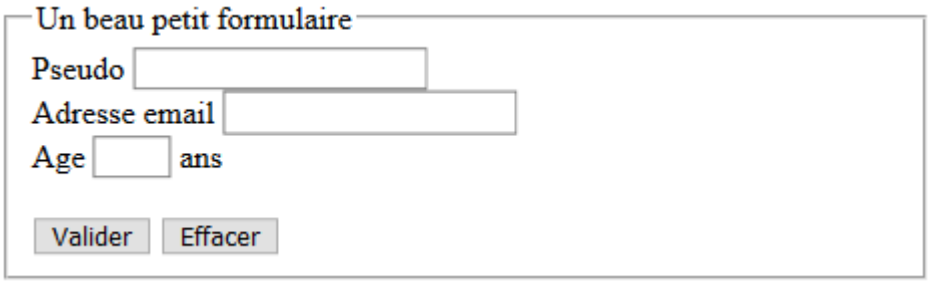
[13] Voir http://41mag.fr/quest-ce-que-le-dom-dune-page-web.html
https://www.pierre-giraud.com/javascript-apprendre-coder-cours/presentation-dom/

5.5 Un exemple de formulaire

Pour conclure nous présentons dans cette section un exemple de formulaire complet[14]. L'objectif est de proposer une interface de saisie d'informations pour l'inscription sur un site par exemple.

a) Présentation du formulaire

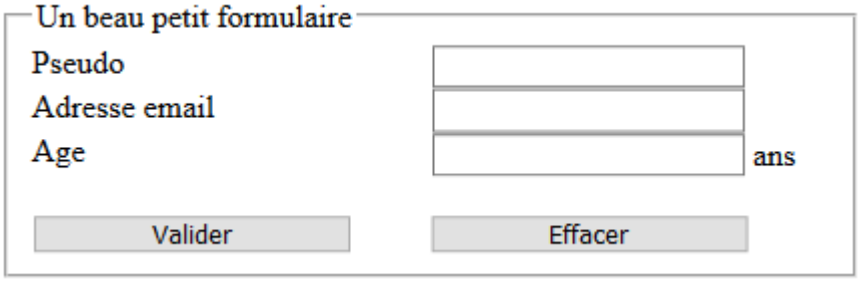
Le formulaire est présenté ci-dessous :



exercice_eleve_formulaire_1_javascript.html

b) Application d'une feuille de style

Nous allons améliorer la présentation avec un peu de CSS :



code_css_eleve_formulaire_1.css

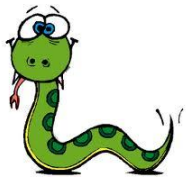
c) Pré-traitement de la saisie avec javascript

Il nous reste à effectuer les vérifications de saisies d'informations. Pour rappel les formulaires de saisie sont destinés à envoyer les informations au serveur, celui-ci les traitera de manière définitive. Par contre effectuer une vérification sur le poste client avant la soumission au serveur économise de la bande passante sur le réseau et augmente la réactivité.

Par contre le serveur devra toujours considérer les informations reçues comme corrompues voir suspectes et les traiter comme tel. Ce point sera abordé au chapitre suivant PHP.

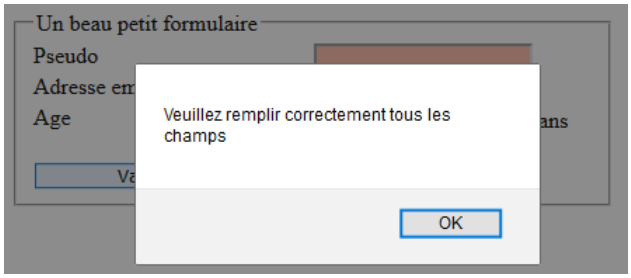
⇦

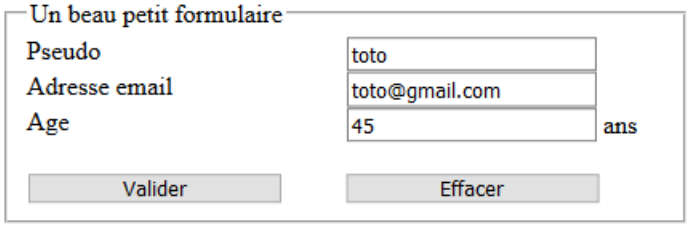
[14] A partir de l'exemple https://openclassrooms.com/fr/courses/146276-tout-sur-le-javascript/144576-td-verification-dun-formulaire

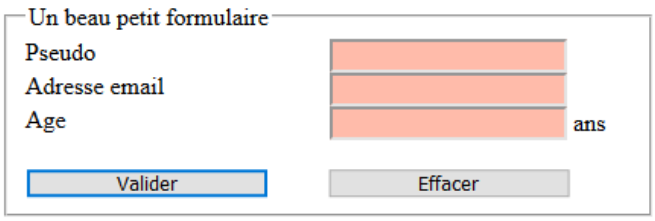


Le comportement souhaité de notre interface consiste à indiquer en rouge les champs correspondants à des saisies ne respectant pas le format attendu.

Lorsque tous les champs sont corrects le formulaire est soumis au serveur par une requête POST.







Nous allons examiner ce principe de fonctionnement pour le champ pseudo l'ensemble sera étudié en TP.

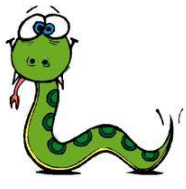
Pour le champ de saisie du pseudo :

```
<label for="pseudo">Pseudo</label>
<input type="text" id="pseudo" name="pseudo" value="" size="20" maxlength="20" onblur="verifPseudo(this)" />
```

L'évènement onblur provoque l'appel de la fonction javascript verifPseudo

```
function verifPseudo(champ)
{
    if(champ.value.length < 2 || champ.value.length > 21)
    {
        surligne(champ, true);
        return false;
    }
    else
    {
        surligne(champ, false);
        return true;
    }
}
```

L'appel de la fonction surligne permet de colorer le champ en cas d'erreur ou bien de laisser la couleur par défaut dans le cas contraire.



```
function surligne(champ, erreur)
{
    if(erreur)
        champ.style.backgroundColor = "#fba";
    else
        champ.style.backgroundColor = "";
}
```

Lors de la soumission du formulaire l'appel à la fonction de contrôle général est réalisé :

```
<form method="POST" action="page.php" onsubmit="return verifForm(this)">
```

```
function verifForm(f)
{
    var pseudoOk = verifPseudo(f.pseudo);
    var mailOk = verifMail(f.email);
    var ageOk = verifAge(f.age);

    if(pseudoOk && mailOk && ageOk)
        return true;
    else
    {
        alert("Veuillez remplir correctement tous les champs");
        return false;
    }
}
```

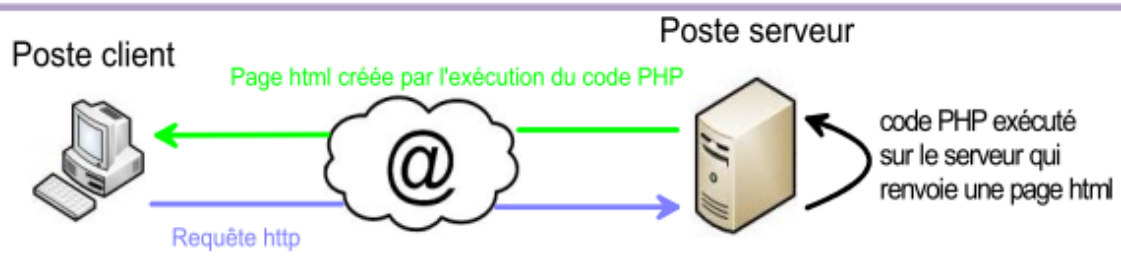
6 Du coté du serveur le PHP

6.1 PHP

a) Historique

Hypertext Preprocessor, plus connu sous son sigle PHP est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur http.

Site dynamique serveur : Le poste client exécute le code html reçu dans son navigateur. Ce code html est issu des instructions exécutées sur le serveur. Ces instructions ne sont pas visibles par le client.



Le langage PHP a été créé en 1994 par Rasmus Lerdorf pour son site web. C'était à l'origine une bibliothèque logicielle en C11 dont il se servait pour conserver une trace des visiteurs qui venaient consulter son CV.

Au fur et à mesure qu'il ajoutait de nouvelles fonctionnalités, Rasmus a transformé la bibliothèque en une implémentation capable de communiquer avec des bases de données et de créer des applications dynamiques et simples pour le Web. Rasmus a alors décidé, en 1995, de publier son code, pour que tout le monde puisse l'utiliser et en profiter[15]. C'est la naissance de PHP.

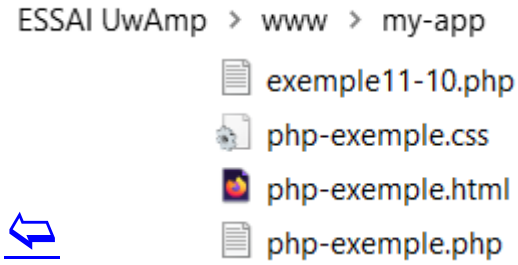
Rasmus Lerdorf

Rasmus Lerdorf en 2007

b) Pour tester nos exemples en PHP

Pour tester nos exemples il faut utiliser un serveur exécutant du code PHP. Nous utiliserons uWamp.

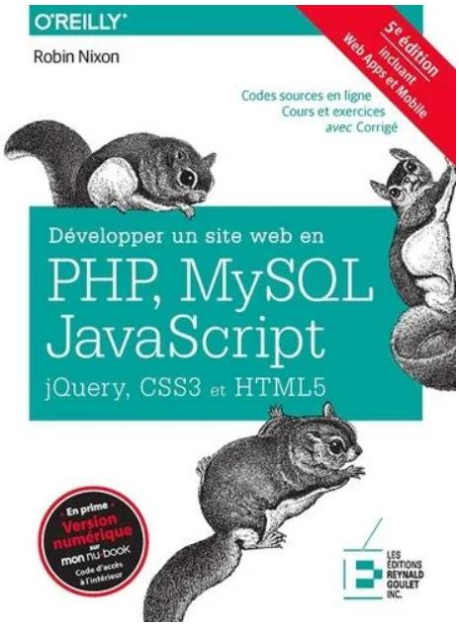
Il faut positionner nos fichiers dans le dossier comme ci-dessous, on lance le navigateur d'uWamp pour sélectionner le fichier à exécuter.



[15] Source Wikipédia

6.2 Un convertisseur °F - °C[16]

Nous empruntons cet exemple à l'excellent livre : Développer un site web en PHP, MySQL JavaScript. Robin Nixon.



Histoire du degré Fahrenheit

Cette unité de mesure est le fruit du spécialiste en physique Daniel Gabriel Fahrenheit. L'échelle de Fahrenheit fut inventée en 1724 en considérant le point de congélation à 32 degrés tandis que le point d'ébullition à 212 degrés.

Histoire du degré Celsius

L'unité degré Celsius fut admise en 1948, jusqu'à lors utilisée sous la forme d'échelle de température centigrade depuis 1742. Le physicien et astronome de la suède Anders Celsius fut l'inventeur de cette dernière échelle qui considérait le 0 comme point de congélation et 100 comme température d'ébullition de l'eau. Une différence de seulement 0,025 degrés Celsius en ce qui concerne le point d'ébullition.

Convertir les Celsius en Fahrenheit : **Fahrenheit = Celsius * 9 / 5 + 32**

Convertir des Fahrenheit en Celsius : **Celsius = (Fahrenheit - 32) * 5 / 9**

a) Fonctionnement de l'interface de conversion

Ce programme est contenu dans un seul fichier .PHP et est composé de deux parties : la page web qui fait la saisie des valeurs à convertir et le code PHP qui effectue la conversion.

exemple11-10.php

Lancement de la page html avec

```
Entrez soit les °F, soit les °C et
cliquez sur Convertir
```

Fahrenheit
Celsius
Convertir

Le fichier doit être placé dans un serveur pour être exécuté.

[16] http://www.calculconversion.com/conversion-celcius-farenheit.html

Après avoir entré 100 °C on obtient la réponse :

Entrez soit les °F, soit les °C et
cliquez sur Convertir

100 °C équivaut à 212 °F

Fahrenheit
Celsius
Convertir

b) Le programme PHP

La partie PHP :

```php
<?php // exemple11-10.php  Livre Robin Nixon p. 280
  $f = $c = '';

  if (isset($_POST['f'])) $f = sanitizeString($_POST['f']);
  if (isset($_POST['c'])) $c = sanitizeString($_POST['c']);

  if (is_numeric($f))
  {
    $c = intval((5 / 9) * ($f - 32));
    $out = "$f &deg;F équivaut à $c &deg;C";
  }
  elseif(is_numeric($c))
  {
    $f = intval((9 / 5) * $c + 32);
    $out = "$c &deg;C équivaut à $f &deg;F";
  }
  else $out = "";

  echo <<<_END
```

Partie HTML

```php
  function sanitizeString($var)
  {
    if (get_magic_quotes_gpc())
      $var = stripslashes($var);
    $var = strip_tags($var);
    $var = htmlentities($var);
    return $var;
  }
?>
```

c) Quelques commentaires

Entrez soit les °F, soit les °C et cliquez sur Convertir

100 °C équivaut à 212 °F

Fahrenheit

Celsius

Convertir

- Le résultat envoyé par le serveur ne contient que du code html :

```
1  <html>
2    <head>
3      <title>Convertisseur de températures</title>
4    </head>
5    <body>
6      <pre>
7        Entrez soit les &deg;F, soit les &deg;C et <br />      cliquez sur Convertir
8
9        <b>100 &deg;C équivaut à 212 &deg;F</b>
10       <form method="post" action="">
11         Fahrenheit <input type="text" name="f" size="7">
12            Celsius <input type="text" name="c" size="7">
13                    <input type="submit" value="Convertir">
14       </form>
15     </pre>
16   </body>
17 </html>
```

Le code PHP n'est donc jamais visible. Le PHP est exécuté sur le serveur uniquement.

- Les valeurs à convertir sont envoyées au serveur par la méthode POST

```
<form method="post" action="">
```

Les noms utilisés font références aux champs *name* donnés dans la page html

Dans le HTML :

```
Fahrenheit <input type="text" name="f" size="7">
   Celsius <input type="text" name="c" size="7">
```

Dans PHP sur le serveur :

```
$_POST['f'] $_POST['c']
```

- Il n'y a pas d'action précisée : action=□□ puisque le code PHP est contenu dans le fichier lui-même.

- Les noms de variables en PHP commencent par $

- La fonction `function sanitizeString($var)`
Nettoie les données envoyées au serveur empêchant de ce fait toute tentative de corruption. Il est très important de toujours considérer les données reçues comme étant corrompues, potentiellement dangereuses et donc les vérifier en conséquence avant de les utiliser.

Nous approfondirons un peu plus le PHP en TP.

6.3 Les variables superglobales de PHP[17]

Les variables superglobales sont des variables internes au PHP, ce qui signifie que ce sont des variables créées automatiquement par le PHP.

Ces variables vont être accessibles n'importe où dans le script et quel que soit le contexte, qu'il soit local ou global. C'est d'ailleurs la raison pour laquelle on appelle ces variables « superglobales ».

Il existe 9 superglobales en PHP. Ces variables vont toutes être des variables tableaux qui vont contenir des groupes de variables très différentes. La plupart des scripts PHP vont utiliser les variables superglobales car ces dernières vont s'avérer très souvent très utiles. Il est donc indispensable de bien les connaitre et de savoir les manipuler.

Les variables superglobales PHP sont les suivantes :

$GLOBALS ; $_SERVER ; $_REQUEST ; $_GET ;

$_POST ; $_FILES ; $_ENV ; $_COOKIE ; $_SESSION.

On écrira toujours les superglobales en majuscules. Cela est une convention qui permet de les différencier des variables « classiques » que nous créons nous-mêmes. Par ailleurs, vous pouvez remarquer que toutes les superglobales à l'exception de $GLOBALS commencent par un underscore _.

Nous avons déjà vu l'utilité des variables $_GET et $_POST.

6.4 Le serveur les cookies

a) Présentation

Un cookie est un petit fichier texte qui ne peut contenir qu'une quantité limitée de données.

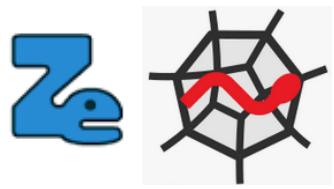
Les cookies vont être stockés sur les ordinateurs de vos visiteurs. Ainsi, à tout moment, un utilisateur peut lui-même supprimer les cookies de son ordinateur. De plus, les cookies vont toujours avoir une durée de vie limitée. On pourra définir la date d'expiration d'un cookie.

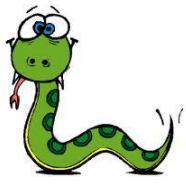
Généralement, nous allons utiliser les cookies pour faciliter la vie des utilisateurs en préenregistrant des données les concernant comme un nom d'utilisateur par exemple.

Ainsi, dès qu'un utilisateur connu demande à accéder à une page de notre site, les cookies vont également automatiquement être envoyées dans la requête de l'utilisateur. Cela va nous permettre de l'identifier et de lui proposer une page personnalisée.

Les cookies ne sont donc pas dangereux en soi même s'ils continuent d'avoir mauvaise réputation. En revanche, on évitera toujours de stocker des informations sensibles dans les cookies comme des mots de passe par exemple car les cookies sont stockés sur l'ordinateur des visiteurs et nous n'avons donc aucune maitrise ni aucun moyen de les sécuriser après le stockage.

[17] D'après https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/variable-superglobale/





b) Fonctionnement[18]

Les cookies font partie des spécifications du protocole HTTP, soit le protocole qui permet de surfer sur des pages web. Le protocole HTTP permet un échange de messages entre le client et le serveur grâce à des requêtes et des réponses HTTP.

Les requêtes et les réponses HTTP contiennent des en-têtes qui permettent d'envoyer des informations spécifiques de façon bilatérale. Un de ces en-têtes est dédié à l'écriture de fichiers sur le disque dur : les cookies.

L'en-tête HTTP qui est réservé à l'usage des cookies s'appelle Set-Cookie. Il s'agit d'une ligne de texte simple de la forme suivante :

```
Set-Cookie : NOM=VALEUR; domain=NOM_DE_DOMAINE; expires=DATE
```

Il s'agit donc d'une chaîne de caractères qui commence par « Set-Cookie : » et suivie de paires clés-valeur, sous la forme CLE=VALEUR, séparées par des points-virgules.

Quelques contraintes :

- Un cookie ne peut pas dépasser 4 Ko
- Un client ne peut pas avoir plus de 300 cookies sur son disque
- Un serveur ne peut créer que 20 cookies maximum chez le client

c) Les cookies et PHP

L'utilisation des cookies est prévue et directement implémentée dans PHP. C'est la fonction setcookie qui s'en charge. Un exemple ici :

exemple_essai_cookie.php

```php
<?php
    setcookie('user_id', '1234');
    setcookie('user_pref', 'dark_theme', time()+3600*24, '/', '', true, true);
?>
<!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Essai cookie</h1>
        <?php
            if(isset($_COOKIE['user_id'])){
                echo 'Votre ID de session est le ' .$_COOKIE['user_id'];
            }
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

[18] D'après https://www.commentcamarche.net/contents/1041-cookies-internet

Le résultat envoyé par le serveur est, comme nous le savons maintenant, une page html :

```
    <!DOCTYPE html>
<html lang="fr">
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Essai cookie</h1>
        Votre ID de session est le 1234
    </body>
</html>
```

Essai cookie

Votre ID de session est le 1234

Un paragraphe

```
<p>Un paragraphe</p>
```

Pour supprimer ou modifier un cookie c'est cette même fonction qui est utilisée. Voir le cours de M. Pierre Giraud :

https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/cookie-creation-gestion/

d) Les cookies et la RGPD

L'utilisation de cookie est soumise à des obligations réglementaire. Le **R**èglement **G**énéral sur la **P**rotection des **D**onnées impose l'autorisation explicite de l'utilisateur avant de déposer des cookies sur son poste informatique, sauf quelques cas précis techniques. Pour plus d'informations consultez le site :

https://www.economie.gouv.fr/entreprises/reglement-general-sur-protection-des-donnees-rgpd

6.5 Le serveur les sessions[19]

Une session en PHP correspond à une façon de stocker des données différentes pour chaque utilisateur en utilisant un identifiant de session unique.

Les identifiants de session vont généralement être envoyés au navigateur via des cookies de session et vont être utilisés pour récupérer les données existantes de la session.

Un des grands intérêts des sessions est qu'on va pouvoir conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre. De plus, les informations de session ne vont cette fois-ci pas être stockées sur les ordinateurs de vos visiteurs à la différence des cookies mais plutôt côté serveur ce qui fait que les sessions vont pouvoir être beaucoup plus sûres que les cookies.

[19] D'après https://www.pierre-giraud.com/php-mysql-apprendre-coder-cours/session-definition-utilisation/

Notez toutefois que le but des sessions n'est pas de conserver des informations indéfiniment mais simplement durant une « session ». Une session démarre dès que la fonction session_start() est appelée et se termine en général dès que la fenêtre courante du navigateur est fermée (à moins qu'on appelle une fonction pour terminer la session de manière anticipée ou qu'un cookie de session avec une durée de vie plus longues ait été défini).

La superglobale $_SESSION est un tableau associatif qui va contenir toutes les données de session une fois la session démarrée.

Pour l'utilisation des sessions voir le site cité en référence.

7 Ressources, non exhaustives

7.1 Sur l'HTML et CSS

a) Citons quelques références de sites parmi un grand nombre :

https://developer.mozilla.org/fr/docs/Apprendre/HTML/Introduction_%C3%A0_HTML

https://www.w3.org/Style/Examples/011/firstcss.fr.html

https://www.pierre-giraud.com/

https://www.w3schools.com/default.asp

b) La gestion des couleurs

Un excellent site décrivant les différentes façons de coder les couleurs avec des nuanciers intelligemment présentés :

https://web-color.aliasdmc.fr/

https://www.w3schools.com/colors/colors_picker.asp

c) Une activité proposée sur mes sites

Création d'un petit site web autour de Jules Vernes

http://sti2dvox.patgue.com/Ressources_7.htm

7.2 Le JavaScript

a) Où écrire le code JavaScript

https://www.pierre-giraud.com/javascript-apprendre-coder-cours/ou-ecrire-code-javascript/

b) Les évènements

https://www.pierre-giraud.com/javascript-apprendre-coder-cours/addeventlistener-gestion-evenement/

c) Pour tester son code

https://jsfiddle.net/

https://validator.w3.org/

d) Positionnement des éléments et empilement

https://developer.mozilla.org/fr/docs/Web/CSS/Comprendre_z-index/Empilement_sans_z-index

https://openweb.eu.org/articles/initiation_flux/

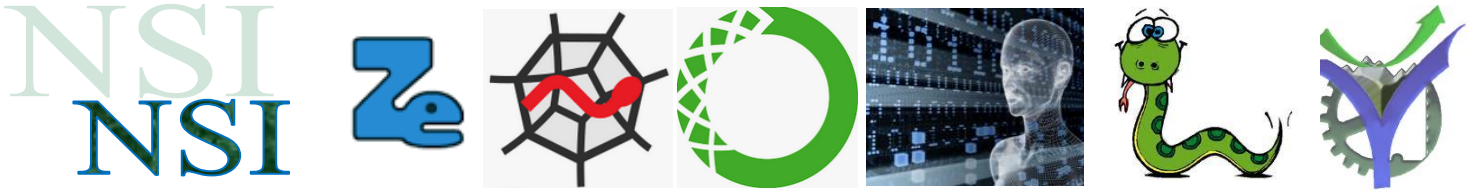
https://openweb.eu.org/articles/initiation_float/

http://fr.learnlayout.com/position.html

e) Les unités en css px, em, pt …

https://www.w3.org/Style/Examples/007/units.fr.html

https://www.w3schools.com/CSSref/css_units.asp



f) Divers

https://developer.mozilla.org/fr/docs/Web/HTML/Element/Input/button
https://developer.mozilla.org/fr/docs/Web/Guide/HTML/Formulaires/Validation_donnees_formulaire
https://www.codingame.com/playgrounds/3777/exercices-de-javascript-pour-debutants-en-informatique/javascript---les-variables

7.3 Site Pierre Giraud

Extraits reproduit avec autorisation de l'auteur, usage gratuit dans le cadre de l'éducation uniquement.

https://www.pierre-giraud.com/
https://www.pierre-giraud.com/html-css-apprendre-coder-cours/