Informatique 3OC

A. Ridard

Programmation Web

Gymnase Yverdon

canton de vaud

Table des matières

I. Web statique : HTML + CSS

Le **Web** est un ensemble de documents reliés entre eux par des liens hypertextes.

Par extension, on englobe aussi sous le terme de Web toute l'infrastructure hébergeant cette collection mondiale de documents hypertextes. Elle comprend :

- une architecture client-serveur utilisant le protocole [1] HTTP [2]
- des langages de programmation côté client :
  — **HTML** [3] et **CSS** [4] pour écrire les documents hypertextes
  — **JavaScript** pour offrir une haute interactivité
- des langages de programmation côté serveur :
  — PHP, Python, ... pour générer du HTML

Cette infrastructure s'appuie sur le réseau Internet, HTTP étant l'un des nombreux protocoles de la couche application s'exécutant au dessus des protocoles TCP et IP.

1. HTML

Voici les grandes étapes du développement du Web :

- **1980 - 1984 :** création du langage SGML [5] basé sur le concept de balises et de liens hypertextes, proposant pour la première fois de séparer la structure d'un document et sa présentation (graphique)
- **1986 :** SGML devient la norme ISO 8879 : 1986
- **1991 :** Tim Berners-Lee, informaticien britannique travaillant au CERN, annonce la première version du Web basé sur un format de document inspiré du SGML mais très simplifié : HTML
- **1993 :** les premiers navigateurs Web graphiques apparaissent, le langage HTML évolue pour intégrer de nouvelles balises telles que les tables
- **1994 - 1997 :** création du World Wide Web Consortium (W3C), organisme de normalisation qui publie la spécification HTML 4.0 en 1997
- **2000 - 2007 :** le Web se développe de manière rapide et anarchique, HTML est étendu de manière non standard par les différents navigateurs, les grands groupes développent leurs activités sur le Web
- **2007 - 2014 :** un long effort de modernisation et de standardisation est entrepris pour donner le HTML 5, finalisé en 2014. Ce format permet d'intégrer du contenu multimédia (audio, vidéo) et facilite l'écriture de pages Web adaptatives [6]

Le langage HTML permet de décrire la **structure** et le **contenu** d'un document Web.

Pour créer un nouveau document Web, il suffit de taper «! » dans VS Code :

[1]. ou sa version sécurisée HTTPS
[2]. HyperText Transfer Protocol
[3]. HyperText Markup Language
[4]. Cascading Style Sheets
[5]. Standard Generalized Markup Language
[6]. les pages s'adaptent automatiquement au support

Nouveau document Web

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

Premier exemple

```html
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Titre du document</title>
</head>
<body>
    <h1>Titre de section</h1>
    <p>Premier paragraphe</p>
    <!-- Commentaire -->
</body>
</html>
```

On peut déjà observer :

- une déclaration de type : <!DOCTYPE html>
- des balises ouvrantes <...>
- des balises fermantes </...>
- des balises vides <.../>
- des attributs associés à des valeurs : lang = "fr", charset = "UTF-8", ...
- un commentaire <!–...–>

Premier exemple

Balises de structure : titres de sections

```html
    <h1>Balise h1</h1>
    <h2>Balise h2</h2>
    <h3>Balise h3</h3>
    <h4>Balise h4</h4>
    <h5>Balise h5</h5>
    <h6>Balise h6</h6>
```

Balises de structure : titres de sections

Balise h1

Balise h2

Balise h3

Balise h4

Balise h5

Balise h6

Balises de structure : paragraphe

```
<p>
    Les espaces      et les retours à la ligne
    sont ignorés à l'intérieur d'un paragraphe délimité par
    les balises &#60;p&#62; et &#60;/p&#62;.
</p>

<p>
    Chaque nouvel élément p crée un nouveau paragraphe,
    insérant un retour à la ligne.
</p>
```

Balises de structure : paragraphe

Les espaces et les retours à la ligne sont ignorés à l'intérieur d'un paragraphe délimité par les balises et .

Chaque nouvel élément p crée un nouveau paragraphe, insérant un retour à la ligne.

Balises de structure : listes

```
On peut organiser le texte en liste non-énumérée :
<ul>
    <li>item 1</li>
    <li>item 2</li>
</ul>

ou en liste énumérée :
<ol>
    <li>item 1</li>
    <li>item 2</li>
</ol>
```

Balises de structure : listes

On peut organiser le texte en liste non-énumérée :

- item 1
- item 2

ou en liste énumérée :

1. item 1
2. item 2

Balises de structure : table

```
On peut aussi présenter du texte dans une table :
<table>
    <tr>
        <td>1</td>
        <td colspan="2">2</td>
        <td>3</td>
    </tr>
    <tr>
        <td rowspan="2">4</td>
        <td >5</td>
        <td rowspan="2" colspan="2">6</td>
    </tr>
    <tr>
        <td>7</td>
    </tr>
    <tr>
        <td>8</td>
        <td>9</td>
        <td>10</td>
        <td>11</td>
    </tr>
</table>
```

Balises de structure : table

On peut aussi présenter du texte dans une table :

1	2		3
4	5 7		6
8	9	10	11

Balises de texte

```
<p>
    On peut mettre du texte en <b>gras</b>, <i>italique</i>,
    <u>souligné</u>, <mark>surligné</mark>, mais aussi en
    <small>petit</small> ou <code>code source</code>.
    <br/>
    On peut aussi le mettre en indice comme dans
    1000101<sub>2</sub> ou en exposant comme dans
    2<sup>3</sup> = 8.
    <br/>
    Les balises peuvent évidemment être
    <i><b><u>combinées</u></b></i>.
    <br/>
    Pour insérer un retour à ligne dans le texte,
    on a utilisé la balise vide &#60;br/&#62;
</p>
```

Balises de texte

On peut mettre du texte en **gras**, *italique*, souligné, surligné, mais aussi en petit ou `code source`.
On peut aussi le mettre en indice comme dans 1000101_2 ou en exposant comme dans $2^3 = 8$.
Les balises peuvent évidemment être ***combinées***.
Pour insérer un retour à ligne dans le texte, on a utilisé la balise vide

Liens hypertextes et images

```
<p>
    Pour accéder à un document Web depuis notre page, on peut utiliser un <a href="https://gymnase-
 yverdon.ch/"> lien hypertexte</a>
</p>

<p>
    On peut aussi insérer une image dans notre page
    <img src="html.png" height="100"/>
</p>
```

Liens hypertextes et images

Pour accéder à un document Web depuis notre page, on peut utiliser un lien hypertexte

On peut aussi insérer une image dans notre page 

Balises neutres, identifiants et classes

```
<div>
    La balise &#60;div&#62; est une <span>balise neutre</span> de structure qui définit un bloc de texte
 .
    <br/>
    La balise &#60;span&#62; est une balise neutre de texte qui laisse le texte inchangé.
</div>

<div style="color:blue">
    Associées aux attributs <code>id</code> et <code>class
    </code>, ces balises participent à la définition des propriétés graphiques des éléments constituant
 une page Web.
    <span style="color:red">Ce point sera détaillé dans la section CSS...</span>
</div>
```

Balises neutres, identifiants et classes

La balise est une balise neutre de structure qui définit un bloc de texte.
La balise est une balise neutre de texte qui laisse le texte inchangé.
Associées aux attributs `id` et `class` , ces balises participent à la définition des propriétés graphiques des éléments constituant une page Web. Ce point sera détaillé dans la section CSS...

Un élément interactif : formulaire

```
<form action="https://math-ridard.fr/formulaire.php">
    Nom : <input type="text" name="nom"/>
    <br/>
    Prénom : <input type="text" name="prenom"/>
    <br/>
    Année de naissance : <input type="number" name="annee-naiss"/>
    <br/>
    <button type="submit">Envoyer</button>
</form>
```

Un élément interactif : formulaire

Nom :
Prénom :
Année de naissance :
Envoyer

Un élément interactif : formulaire

Nom : Ridard
Prénom : Anthony
Année de naissance : 1979
Envoyer

Un élément interactif : formulaire

```
<?php
    $nom = $_GET["nom"];
    $prenom = $_GET["prenom"];
    $age = date("Y") - $_GET["annee-naiss"];
?>

<?php echo $prenom . " " . $nom; ?>, vous avez
<?php echo $age; ?> ans cette année.
```

Un élément interactif : formulaire

Anthony Ridard, vous avez 45 ans cette année.

L'interprète PHP, installé sur un serveur (celui qui héberge mon site), exécute le code se trouvant entre les balises <?php et ?>.

Le langage PHP se comporte naturellement bien avec le HTML, et souvent associé au SGBD MySQL il permet aussi de gérer le stockage des données sur le serveur.

Nous ne détaillerons pas ces points qui sortent du cadre de ce cours.

Générer du HTML avec Python

Écrire un programme Python qui crée un fichier nommé **table.html** contenant les tables de multiplication de 1 à 10. Chaque table doit être précédée d'un titre de niveau 1, et suivie de la table dans une liste non énumérée. Par exemple, la portion du document correspondant à la table de 1 sera :

```
<h1>Table de 1</h1>
<ul>
    <li>1 * 1 = 1</li>
    <li>1 * 2 = 2</li>
    ...
</ul>
```

2. CSS

Le langage CSS permet de définir les propriétés graphiques des éléments HTML.

Attribut style

```
<p style="font-family: serif">
    On écrit un <span style="border:1pt solid black">
    paragraphe</span> de texte, mais cette fois on
    <span style="color:red">modifie</span> le style
    graphique des <span style="border:1pt solid black">
    éléments</span>.
</p>
```

Attribut style

On écrit un paragraphe de texte, mais cette fois on modifie le style graphique des éléments.

On peut factoriser le code [7] à l'aide des identifiants et des classes.

Balise style

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS 2</title>
    <style>
        p {font-family: serif}
        .bord {border:1pt solid black}
        #n42 {color:red}
    </style>
</head>
<body>
    <p>
        On écrit un <span class="bord">paragraphe</span> de texte,
        mais cette fois on <span id="n42">modifie</span> le style
        graphique des <span class="bord">éléments</span>.
    </p>
</body>
```

De cette manière, il est aisé de modifier les valeurs des propriétés graphiques.

Balise style

```
    <style>
        p {font-family: sans-serif}
        .bord {border:2pt dashed orange}
        #n42 {color:blue}
    </style>
```

Attribut style

On écrit un paragraphe de texte, mais cette fois on modifie le style graphique des éléments.

Si plusieurs fichiers HTML doivent respecter le même style graphique, la **balise style** ne suffit plus.

Fichier style.css

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS 4</title>
    <link href="style.css" rel="stylesheet"/>
</head>
```

Fichier style.css

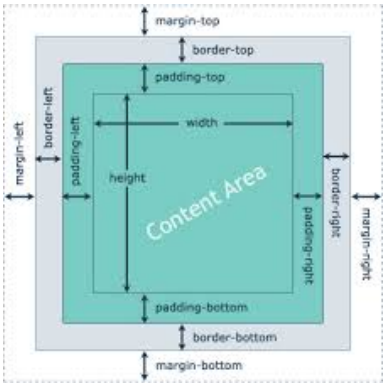
```
p {font-family:sans-serif;}
.bord {border:2pt dashed orange;}
#n42 {color:blue;}
```

[7]. pour améliorer sa lisibilité et sa maintenance

Le format CSS est standardisé par le W3C, en voici quelques propriétés :

- Mode d'affichage de la boite [8] : **display** (block, inline ou none)
- Taille de la marge : **margin** (longueur)
- Bordure : **border** (taille motif couleur ou none)
- Taille de l'ajustement : **padding** (longueur)
- Couleur du fond de la boîte : **background** (couleur)
- Couleur du texte : **color** (couleur)
- Décoration du texte : **text-decoration** (underline, overline, line-through ou none)
- Justification du texte : **text-align** (left, right, center ou justify)
- Nom de la police : **font-family** (fixed, serif ou sans-serif)
- Graisse de la police : **font-weight** (normal, light, bold ou bolder)
- Style de la police : **font-style** (normal ou italic)
- Taille de la police : **font-size** (longueur ou bien small, normal ou large)

Concernant la boite, il est possible de préciser les propriétés :



Dans une feuille de style (fichier CSS), une règle CSS est de la forme suivante :

Règle CSS

```
p1 p2 ... pn { prop1 : v1 ; prop2 : v2 ; ... ; propk : vk ; }
```

La partie p1 p2 ... pn est le **sélecteur** constitué de plusieurs pas [9] pouvant être :

- une balise
- une classe préfixé par un point
- un identifiant préfixé par un hashtag
- * désignant n'importe quelle balise
- certaines combinaisons écrites en concaténant sans espace

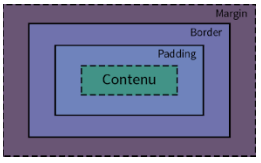
Principe de cascade

En cas de conflit, c'est le principe de cascade qui résout le problème :

- L'attribut style est prioritaire par rapport à la balise style
- La balise style est prioritaire par rapport au fichier style s'il est inclus avant
- Les autres conflits [a] sortent du cadre de ce cours

a. au sein d'un même fichier style par exemple

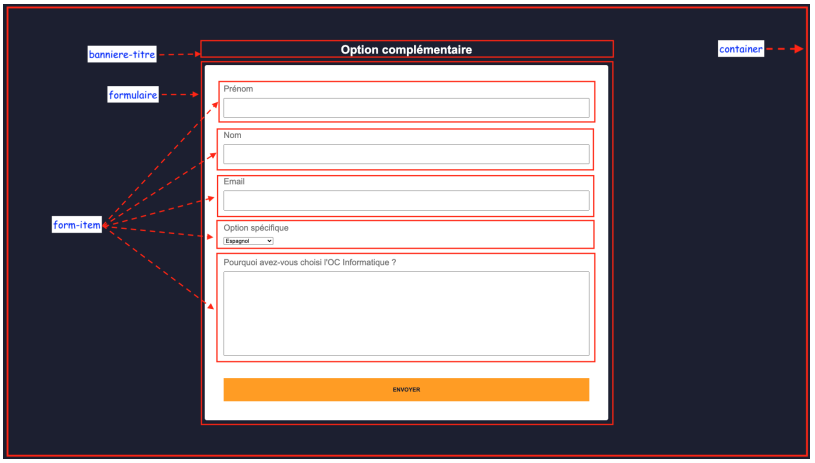
[8]. Chaque élément html est contenu dans une boîte :



[9]. Au fur et à mesure des pas, les éléments concernés sont définis par inclusion

Formulaire 2

Écrire un script html permettant de générer, à l'aide du fichier **style.css** (disponible sur Moodle), le formulaire Web suivant :



Formulaire 2

Pour associer un label à un élément du formulaire, on peut utiliser l'attribut **for** :

```
<div class="form-item">
    <label for="prenom">Prénom</label>
    <input type="text" id="prenom"/>
</div>
```

Pour inclure la liste des options spécifiques, on pourra utiliser le code suivant :

```
<select name="os" id="os">
    <option value="espagnol">Espagnol</option>
    <option value="italien">Italien</option>
    <option value="grec">Grec</option>
    <option value="latin">Latin</option>
    <option value="maths-physique">Maths-Physique</option>
    <option value="bio-chimie">Bio-Chimie</option>
    <option value="eco-droit">Eco-Droit</option>
    <option value="philo-psycho">Philo-Psycho</option>
    <option value="arts_visuels">Arts visuels</option>
    <option value="musique">Musique</option>
</select>
```

II. Web dynamique : JavaScript

Le langage PHP qui s'exécute côté serveur permet de créer des pages Web dynamiques interagissant avec l'utilisateur, mais cette interactivité est limitée pour deux raisons :

- L'envoi de paramètres se fait par une requête HTTP, et donc implique une communication en réseau avec le serveur.
- A chaque nouvelle requête, le serveur recalcule entièrement la page de destination, et donc l'utilisateur voit s'afficher une nouvelle page.

Le langage JavaScript, dont un interpréteur est intégré dans tous les navigateurs grand public, permet d'offrir une haute interactivité en calculant côté client!

Console JavaScript

1. Créer un nouveau fichier, *js-console.html*, contenant le code suivant :

```
<body>
    <button>Bouton</button>
    <span id="valeur">0</span>
</body>
```

2. Ouvrir la console Javascript[a] et exécuter les instructions suivantes :

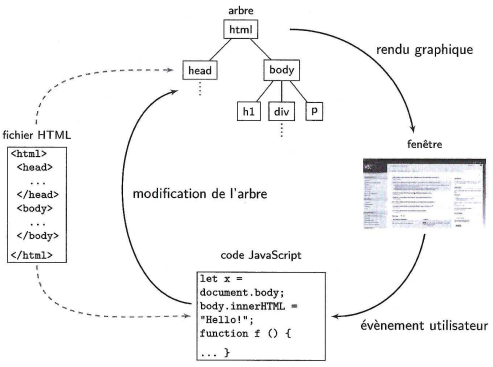
```
document.body.style.background = "blue";

let v = document.getElementById("valeur");
v.innerHTML = "1";
```

3. Recharger la page.

a. Dans Chrome –> ⋮ / Plus d'outils / Outils de développement

Après avoir lu le code HTML, le navigateur crée un arbre, appelé DOM [10], dont les noeuds représentent les éléments HTML. Cet arbre est ensuite parcouru de manière à appliquer les règles CSS, permettant à la page de s'afficher dans la fenêtre.

Le code JavaScript, déclenché par un événement [11] utilisateur, a la capacité de modifier cet arbre et donc l'affichage de la page, sans tout recalculer et sans modifier le fichier HTML initial.



Pour afficher le nombre de clics, nous aurons besoin de :

- définir un gestionnaire d'événement (fonction exécutée suite à un événement)
- d'associer ce dernier à un événement utilisateur

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS - Clic</title>
    <script type="text/javascript">
        let compteur = 0;
        // définition du gestionnaire d'événement
        function suivant() {
            compteur = compteur + 1;
            let v = document.getElementById("valeur");
            v.innerHTML = compteur;
        }
    </script>
</head>
<body>
    <!-- association du gestionnaire à l'événement -->
    <button onclick="suivant();">Clic</button>
    <span id="valeur">0</span>
</body>
```

[10]. Document Object Model
[11]. On parle de programmation événementielle, en opposition à la programmation séquentielle

Comme pour la balise style, les bonnes pratiques actuelles prônent une séparation stricte du code JavaScript, du HTML et du CSS.

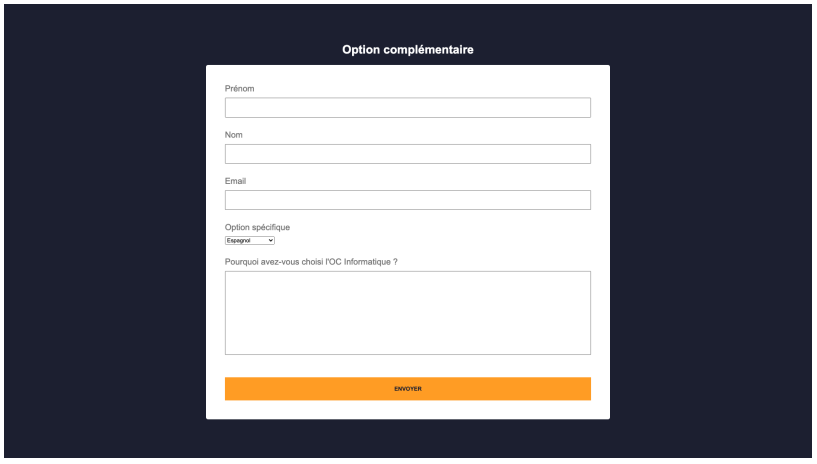
Code HTML :

```html
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JS - Clic</title>
    <script src="script-clic.js" defer></script>
</head>
<body>
    <button id="btn">Clic</button>
    <span id="valeur">0</span>
</body>
```

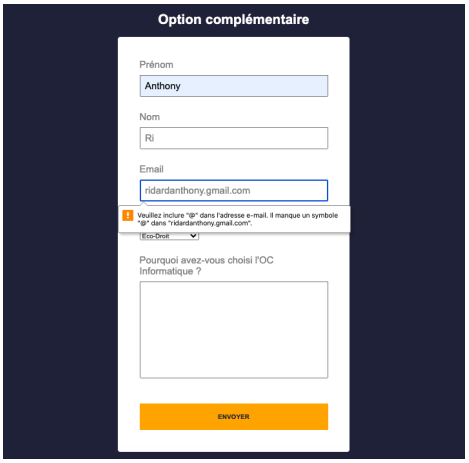
Code JavaScript :

```javascript
let compteur = 0; // variable globale

function suivant() {
    compteur = compteur + 1;
    let v = document.getElementById("valeur"); // variable locale
    v.innerHTML = compteur;
}

let b = document.getElementById("btn");
b.addEventListener("click", suivant);
```

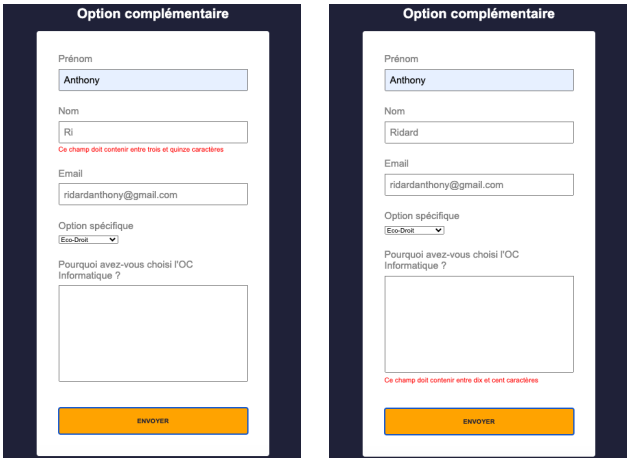
Pour aller plus loin, reprenons notre formulaire :



Notons que le navigateur commence par valider le type (text ou email) des inputs.
En particulier, il affiche un message d'erreur lorsque l'email n'est pas conforme :



Nous souhaitons étendre la validation des données aux autres inputs du formulaire :



Pour cela, nous allons décomposer le code de la manière suivante :

```
// on sélectionne et on stocke tous les éléments HTML nécessaires
const form = document.getElementById('formulaire');
const prenom = document.getElementById('prenom');
const nom = document.getElementById('nom');
const email = document.getElementById('email');
const motivation = document.getElementById('motivation');

const msgError  = document.querySelectorAll('.error');

const titre = document.getElementById('titre');

// on définit le gestionnaire de l'événement "soumission" du formulaire
function validation_stockage(){
}

// on associe ce gestionnaire à son événement
form.addEventListener('submit', function(e){
    // on évite la suppression, par défaut, du formulaire après soumission
    e.preventDefault();
    validation_stockage();
});
```

A la soumission du formulaire, le gestionnaire d'événement doit réaliser :

- la validation des données
- le stockage des données dans la mémoire du navigateur

Nous avons besoin d'ajouter les règles CSS suivantes :

```
.error{
    margin-top: 5px;
    color: red;
}
.invisible{
    display: none;
}
```

Commençons par écrire la partie validation des données :

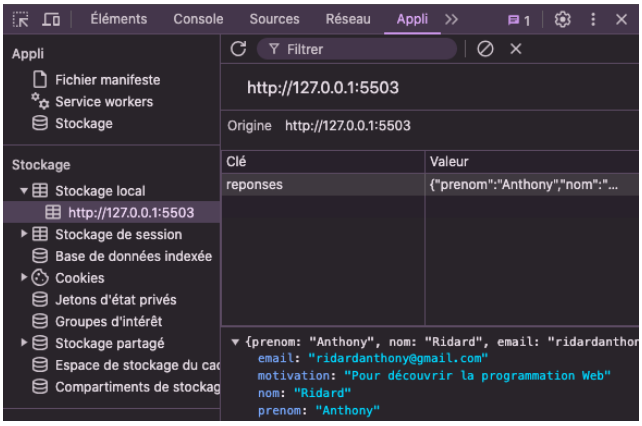
```
// on récupère les valeurs des inputs sans les espaces en début et en fin de saisie
const prenomValue = prenom.value.trim();
const nomValue = nom.value.trim();
const emailValue = email.value.trim();
const motivationValue = motivation.value.trim();

// on "supprime" les messages d'erreurs pour n'afficher que les erreurs restantes
msgError.forEach(error =>{
    error.classList.add('invisible');
});

// on vérifie la "validité" des inputs
// le type (text ou email) est d'abord vérifié par le navigateur
if(prenomValue.length < 2 || prenomValue.length > 10){
    // si la saisie est trop courte ou trop longue, on rend visible le message d'erreur
    // cette DIV est le noeud frère qui suit l'élément "prenom"
    prenom.nextElementSibling.classList.remove('invisible');
} else if(nomValue.length < 3 || nomValue.length > 15){
    nom.nextElementSibling.classList.remove('invisible');
} else if(motivationValue.length < 10 || motivationValue.length > 100){
    motivation.nextElementSibling.classList.remove('invisible');
} else{
    console.log('Inputs validés');
    form.remove();
    titre.innerText = "Formulaire envoyé !";
}
```

Lorsque tous les inputs sont validés, nous obtenons l'affichage suivant :

Poursuivons avec la partie stockage des données :

```
// on stocke les inputs validés dans un objet JavaScript (dictionnaire)
reponses = {
    prenom : prenomValue,
    nom : nomValue,
    email : emailValue,
    motivation : motivationValue
};
console.log(reponses);

// on transforme cet objet JavaScript en objet JSON
reponsesString = JSON.stringify(reponses);
console.log(reponsesString);

// on peut alors enregistrer cet objet JSON dans le navigateur (local storage)
localStorage.setItem('reponses', reponsesString);
```

On retrouve bien ces données sur une ligne sous la forme clé/valeur :



La valeur est elle même ici un objet JavaScript contenant 4 couples de la forme clé/valeur, ou plus exactement l'objet JSON associé (chaîne de caractères).

On peut évidemment récupérer cet objet JSON, le transformer en objet JavaScript, et le manipuler comme un dictionnaire en Python :

```
// inversement, on peut aussi récupérer un objet JSON mémorisé dans le navigateur
repString = localStorage.getItem('reponses');

// puis le transformer en objet JavaScript
repJS = JSON.parse(repString);
console.log(repJS);
console.log(repJS["prenom"]);
```

Pour bien comprendre ces transformations, analysons la console :

Cookies - HTML/CSS

Écrire un script html permettant de générer, à l'aide du fichier **style.css** (disponible sur Moodle), la page Web suivante :

Cookies - JavaScript

Écrire un programme en JavaScript permettant de :

- Rendre invisible la bannière après avoir cliqué sur le bouton.

- Enregistrer l'acceptation des cookies dans la mémoire du navigateur.

Clé	Valeur
cookiesOk	oui

- Supprimer la bannière à l'ouverture de la page en cas d'acceptation passée.

Des compléments seront fournis, au besoin, pour le TP 8 et le projet...