DLA PIPER

DLA Piper

Jour 1

Simple page

DLA PIPER

Introduction

Bienvenue!
Nous allons aujourd'hui vous montrer un exemple de site web très simple. Le squelette qui vous est donné ne contient que du HTML (HyperText Markup Language), le langage clé dans le monde du web. Nous vous proposons d'aggrémenter cette simple page de deux autres langages, le CSS (Cascading StyleSheet), dont le rôle sera de customizer le style de notre page (couleurs, formattage…), et le JS (JavaScript), qui permet de rendre le site dynamique ("responsive").
Au moindre doute, merci de visiter ce site, très complet. Si, après avoir cerché, le doute persiste, nous sommes à votre disposition.

Composer vos requêtes Google de la forme '*sujet mal compris* W3Schools' peut vous faire gagner beaucoup de temps.

Les recherches en anglais donnent en général bien plus de résultats.

Les éléments fournis contiendront des commentaires ou des FIXME pour vous indiquer ou placer vos 'réponses'

Etape 1 - HTML simple

Pour cette étape, il vous est demandé de compléter la page existante en ajoutant **dans une balise texte** le message suivant:

```
Hello World! The date is:
```

Toujours a l'intérieur de cette balise texte, ajoutez une balise **span** vide, avec l'id `clock`:

```
<span id="clock"></span>
```

Bien évidemment, nous ajouterons le code qui permet d'afficher la date dans une prochaine étape.
Just en dessous de ce texte, ajoutez **deux balises 'bouton'**:

- Le premier contenant le texte 'Turn on the red light!'
- Le deuxième contenant le texte 'Paint it black!'

De la même façon, nous ajouterons les actions de ces boutons prochainement.

Etape 2 - HTML - Formulaire et table

+ Formulaire

La majeure partie des sites web finissent à un moment ou à un autre par demander des informations a l'utilisateur. Ceci est représente par un **formulaire**, permettant à l'utilisateur d'entrer du texte, des images, des documents … directement sur la page web. Ces informations seront à terme récupérées par un **serveur** pour être stockées et sauvegardées.
Aujourd'hui, nous allons simplement stocker ces informations **en local**, c'est à dire de manière éphémère.
Ajoutez **un formulaire contenant trois champs**:

- Un champ texte pour entrer un nom. Ce champ se nommera `nameInput`
- Un champ numérique pour entrer un âge. Ce champ se nommera `ageInput`
- Un champ numérique pour entrer un montant de participation. Ce champ se nommera `participationInput`

Pensez à associer un **label** à chacun de ces champs avec le texte correspondant, ainsi qu'un **bouton submit** à la fin du formulaire.

+ Table

Maintenant que l'utilisateur peut renseigner ses informations, nous pouvons les afficher ! Nous verrons comment les afficher dynamiquement dans les prochaines étape. Pour le moment, ajoutez **une table HTML**:

- La première ligne sera le **header** de cette table, composée de trois colonnes (une colonne par champ, dans le même ordre)
- Les lignes suivantes seront ajoutées dynamiquement par la suite.

Etape 3 - Javascript et CSS simples

+ CSS

Le CSS est un langage très simple, mais très flexible, permettant assez facilement de formatter des pages web. Par exemple, le site que nous avons ici n'est pas du tout **responsive**: il ne d'adapte pas a la taille de l'écran, ni à l'appareil sur lequel il est affiché (iPhone, tablette, pc ...).
Pour remédier à celà, le monde du web met à disposition des **directives CSS** simples et efficaces:

- Ajoutez la balise permettant de définir le **viewport** de votre page web.
- Ajoutez au fichier CSS fourni un selecteur prenant en compte tout le **body** de votre page.
- Ajoutez-y ces directives:

```
display: flex;
flex-direction: column;
align-content: center;
align-items: center;
```

Nous ne perdrons pas trop de temps à expliquer chacune d'elles; sachez simplement que la 'technologie' utilisée ici se nomme flexbox, et qu'elle permet d'organiser une page web de manière très simple et très flexible. Normalement, elles sont assez explicites pour avoir une idee de ce qu'elles font. Libre à vous d'aller lire le lien donné ci-dessus pour plus d'explication. Maîtriser cette technologie peut néanmoins s'avérer extrèmement chronophage.

+ JavaScript

Pour cette étape, il est primordial que vous ayiez **identifié vos balises HTML correctement**.

L'attribut `id` est votre ami.

Maintenant que nous avons le 'squelette statique' de notre page web, nous allons le dynamiser. Nous commencerons par ajouter les actions de nos deux boutons. A des fins initiatiques, nous vous donnons le code du bouton 'Paint it black!', à inserer entre les deux balises `<script>` à la fin du fichier.

```
document.getElementById('toBlackButton').onclick = function dateToBlack() {
    document.getElementById('clock').classList.add("black");
    document.getElementById('clock').classList.remove("red");
};
```

Le JavaScript est un langage dynamique et flexible (parfois trop ...) permettant de manipuler les éléments de votre page web **en temps réel**; le HTML étant un simple langage de description, interprété 'une seule fois' par votre navigateur au moment du chargement de la page, il ne permet pas en lui même de dynamiser

une page. Il sert simplement de 'liste d'ingrédients' de votre page, qui reste statique. Le JavaScript vient apporter la partie 'recette', comment s'enchainent les actions, quelles sont leurs effets, etc…

`document` represente notre page. Nous appelons `getElementById`, qui fait exactement ce qui est ecrit: elle récupère l'élément avec l'id passé en paramètre.
`on-click` est un **évènement**: dans la majorité des cas, il représente une action de l'utilisateur. De manière générale, un 'event' est une action venant de l'exterieur, d'une autre partie du programme, d'une autre source … Nous assignons (remplissons) cet évènement à une fonction, nommée `dateToBlack`, qui elle aussi récupère un élément dans la page pour manipuler ses `classes`. Voyez ici une **classe CSS**, un moyen d'identifier nos eléments pour les styliser plus facilement. Référez vous à la fois a W3Schools et au code fourni pour comprendre le lien entre cet **attibut HTML** et le **sélecteur CSS** associé.

Traditionnellement, nous ne mettrions pas le code JavaScript directement dans la page HTML, mais plutôt dans ses propres fichiers, référencés via des balises spécifiques. Aujourd'hui, pour des raisons de clarté, nous ferons tenir toute la page dans un seul fichier HTML et CSS.

Suivant le même modèle que la fonction fournie, ajoutez la fonction au `on-click` de l'autre bouton, qui colore la date en rouge.

N'oubliez pas d'enlever la classe 'black' !

Etape 4 - Formulaire Javascript

Notre formulaire n'a pour l'instant aucun effet. Regardez le code de l'etape 4: la fonction `validateForm` vérifie que l'utilisateur n'a pas laissé de champ vide avant de cliquer sur 'Submit'. Si tout va bien, elle retourne la liste de ce que l'utilisateur a rentré, sinon, elle retourne une liste vide.
Ensuite, regardez la fonction `addToTable`. Celle ci est appelée lorsque l'utilisateur clique sur Submit, vérifie le contenu du formulaire avec la fonction mentionnée juste avant, puis rempli la table avec les nouvelles données.
Suivant l'exemple donné, remplissez la fin de la fonction afin de remplir les deux derniers champs (age et participation) avec les valeurs du formulaire, puis ajoutez la ligne nouvellement créee à la table.

Etape 5 - Framework CSS

Cette étape est avant tout un exemple du résultat que vous pourriez avoir en utilisant un **framework CSS**, une (énorme) list de sélecteurs CSS préfaits vous permettant de styliser votre site simplement en ajoutant des classes a vos éléments.
Celui donné ici se nomme MDL, et implémente le Material Design de Google. Prenez le temps de regarder comment les elements ont été modifiés par rapport a votre version: de simples ajouts dans l'attribut `class` et votre page devient Material !
D'autres frameworks existent, nottament Bootstrap ou encore Foundation.