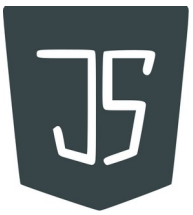
Activité 5 : Le web dynamique avec Javascript



Les chapitres précédent introduisaient deux langages : html et css qui permettent d'éditer des pages web. Mais ces pages sont statiques.
Pour les rendre dynamiques (c'est-a-dire qu'elles puissent se modifier en fonction de l'utilisateur), il faut un langage de Programmation, interprétable par les navigateurs : Le langage Javascript.

Repères historiques

Le langage JavaScript a été crée dans les années 1995 par Brendan Eich, un informaticien Américain alors employé de l'éditeur de logiciels Netscape Communications. L'un des produits phare de cette société était le navigateur Web Netscape Navigator, qui est le premier à intégrer JavaScript. A l'époque, le système Windows de Microsoft et son navigateur Internet explorer sont omniprésents. Netscape Navigator ne peut pas survivre en tant que navigateur commercial. Son code est alors ouvert et il sert de base au navigateur Firefox, toujours en développement actuellement.

Brendan Eich

1. Interactions entre HTML et JavaScript :

Le langage JavaScript sert à ajouter de l'interactivité dans une page Web. Il permet de gérer les entrées de données dans un formulaire. L'activation de bouton par exemple est rendue possible lors du passage du pointeur ou lors d'un clic sur le bouton.

Description d'une page minimale

Comme pour Html et Css, le code JavaScript peut être écrit au sein du code Html ou dans un fichier séparé.

Considérons le code JavaScript et Html suivant :

Code JavaScript contenu dans script.js

```
alert ("JavaScript dans un autre fichier")
```

Code Html contenu dans index.htm

```
<html>
        <head>
                <title> Ma page </title>
                <script language="JavaScript" src="script.js">
                </script>
        </head>
        <body>
                <p> Page avec JavaScript </p>
        </body>
</html>
```

Ce code est disponible dans le répertoire .\code\exemple1

Q1. Ouvrir le fichier index.htm dans un navigateur et **observer** le résultat
Q2. Indiquer dans quels codes sont définis les textes de la page web et de la boîte de dialogue.
Q3. Rechercher et **indiquer** le nom de la balise Html qui référence le script Javascript à exécuter. **Indiquer** la position de cette balise dans la structure de la page Html.
Q4. Décrire le rôle de la fonction alert en JavaScript

Test d'instructions élémentaires

But de l'activité : Modifier en direct le contenu d'une page web en lui appliquant des instructions JavaScript

Q5. Créer un fichier test.html contenant les instructions suivantes :

```
<html>
        <head>
                <title> Page de test </title>
        </head>
        <body>
                <div id='mondiv'> Hello </div>
        </body>
</html>
```

Q6. Ouvrir ce fichier avec le navigateur Firefox. **Ouvrir** la console JavaScript avec la combinaison de touches [Ctrl]+[↑] +[K]
Q7. Ecrire les instructions suivantes dans la console et constater que l'affichage de la page se modifie automatiquement. Il est toujours possible de recharger la page avec la touche [F5], pour se retrouver dans l'état initial (i.e. le fichier stocké en mémoire n'est pas modifié).

➔ Changement de la couleur de fond de la page :

```
document.body.style.background = 'red';
```

➔ Réinitialisation de la couleur par défaut :

```
document.body.style.background = '';
```

➔ Récupération d'un élément. Cette instruction doit être faite une fois pour que d soit définie, avant de tester la suite :

```
let d = document.getElementByld('mondiv');
```

➔ Changement du style de élément d

```
d.style.color = 'blue' ;
```

```
d.style.fontWeight = 'bold' ;
```

```
d.style.border = '1pt dashed green' ;
```

➔ Changement du contenu de l'élément d :

```
d.innerHTML = 'Salut , <i> comment ça va?</i>;
```

2.Évaluation de code JavaScript

Une structure de données pour HTML

L'un des principaux attraits de l'interpréteur Javascript d'un navigateur est la possibilité de modifier dynamiquement et en temps réel le rendu d'une page Web. La figure 1 décrit succinctement le fonctionnement d'un navigateur Web. Le navigateur commence par télécharger le fichier HTML correspondant à l'URL visitée. Une fois récupérée, la page est transformée en une structure de donnée d'arbre appelée DOM (Document Object Model). Une fois l'arbre construit, ses nœuds (les éléments du document HTML) sont parcourus, les règles CSS sont appliquées et le rendu graphique de la page est effectué. Dans le mème temps, les scripts JavaScript présents dans la page sont exécutés.

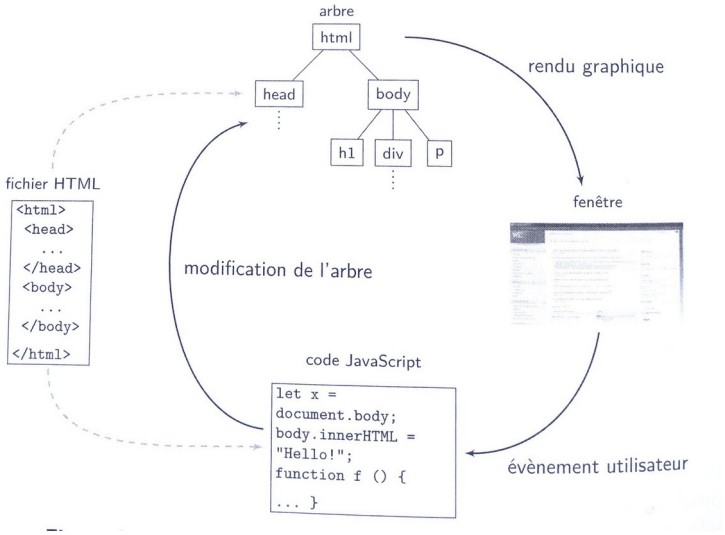


Figure 1: Structure de données en HTML (DOM)

L'une de leur activité principale est la définition de gestionnaires d'événements, ces gestionnaires peuvent modifier librement le DOM, ajouter ou supprimer des balises ou leurs attributs mais encore une fois, le fichier original qu'il soit stocké localement ou sur un serveur Web n'est pas modifié puisque ce script est exécuté sur la machine par le navigateur.

Les évènements

Un événement est par exemple le chargement d'une page, le passage du pointeur sur la page ou un élément de la page, un clic avec la souris sur la page ou un élément de la page... Ces événements peuvent servir à déclencher l'exécution d'une fonction.
Les évènements usuels sont :

Load	Chargement d'une page
Click	Click avec la souris, sélection d'un élément
Dblclick	Double-clic sur l'élément
Mouseover	Entrée du pointeur sur un élément
Mouseout	Sortie du pointeur de l'élément
Keypress	Appui sur une touche produisant un caractère

Considérons le code suivant :

```
<html>
	<head>
		<title>Ma page</title>
		<script language="JavaScript" >
		function rebours(x) {
			while (x >= 0) {
				alert (x) ;
				x-- ;
			}
		}
		n = prompt("Saisir un nombre entier", "nombre") ;
		</script>
	</head>
	<body onLoad="rebours(n)">
		<p> Page avec JavaScript </p>
	</body>
</html>
```

Q8. Indiquer l'évènement déclencheur de l'appel de la fonction rebours du script JavaScript.

Q9. A la lecture de la fonction rebours, **indiquer** les caractères utilisés pour délimiter les instructions associées à cette fonction. **Décrire** le rôle de cette fonction.

Q10. Prédire le comportement du navigateur Firefox après le chargement de ce code html / javascript. **Exécuter** ce code et comparer le résultat obtenu avec celui prédit.

Un jeu en Javascript

But de l'activité : Programmer dans une page HTML le jeu du devine nombre

Pour rappel, ce jeu a déjà été codé en Python. Il consiste à trouver un nombre choisi au hasard entre 1 et 100 par exemple. A chaque proposition du joueur, le programme répond 'trop grand' ou 'trop petit', 'gagné' ou 'perdu'. Le nombre de coups joués est compté et limité à 8.
A la fin de la partie, le message 'gagné' ou 'perdu' sera affiché sur la page Web qui clignotera de façon différente suivant le résultat.

Fonctions Javascript utilisées :
Les interactions avec l'utilisateur seront réalisées avec les boîtes de dialogue `prompt` et `alert`.
Le tirage au sort d'un nombre peut être effectué avec les instructions :

- ✗ `Math.random()` : renvoie un nombre à virgule (flottant) compris entre 0 et 1
- ✗ `Math.random(x)` : arrondit x à l'entier le plus proche

Q11. A partir du document ressource page 5/5, **définir** et **tester** le code de ce jeu

3.Gestion des évènements

Résumons l'interaction entre l'utilisateur et la machine : l'utilisateur déclenche un évènement qui réagit en appelant un code JavaScript. Le code JavaScript s'exécute et provoque la modification du DOM de la page WEB.
Remarquons que même si la page provient d'une machine distante, le fichier HTML interprété dans le navigateur a été enregistré sur la machine utilisateur, **le client**, et le

code JavaScript est exécuté sur cette machine. On peut en déduire immédiatement les éventuels problèmes de sécurité que cela soulève.
Dans les exemples qui suivent, un attribut, formé du mot on et du nom de l'évènement est ajouté dans l'élément défini par une balise.
La valeur de l'attribut est le code JavaScript qui appelle une fonction message.

```
<html>
	<head>
	<title>Ma page</title>
	<script language="JavaScript">
		function message(){
			alert("Un événement s'est produit") ;
		}
	</script>
	</head>
	<body>
		<p onClick="message()" >Cliquer</p>
		<p onDblclick="message()" >Double-cliquer</p>
		<h3 onMouseover="message()" > Approcher le curseur </h3>
		<h3 onMouseout="message()" > Eloigner le curseur </h3>
		Taper sur une touche <input type="text" onKeypress="message()" >
	</body>
</html>
```

Code est disponible dans le répertoire .\code\exemple4

Les attributs `onClick`, `onDblclick`, `onMouseover`, `onMouseout` et `onKeypress`, sont présents dans ce code.

Q12. Prédire le comportement du navigateur Firefox après le chargement de ce code html / javascript. **Exécuter** ce code et **comparer** le résultat obtenu avec celui prédit.

Plusieurs gestionnaires d'événements peuvent être associés à un même élément. Dans le code qui suit, les deux attributs `onMouseover` et `onClick` sont dans la balise <a>. Ils permettent d'appeler les deux fonctions distinctes `site` et `message`.

```
<html>
	<head>
		<title>Ma page</title>
		<script language="JavaScript">
		function message(){
			alert("Un événement s'est produit");
			}
		function attention(){
			alert("Vous ne pouvez pas cliquer sur le lien !");
			}
		function site(){
			alert ('Education Nationale') ;
			}
		</script>
	</head>
	<body>
		<input type="button" value ="A lire" onClick="attention()" ></br>
		<a href="http://www.education.gouv.fr/" onMouseover="site()"
			onClick="message()">Cliquer pour aller sur le site<a/>
	</body>
</html>
```

Code est disponible dans le répertoire .\code\exemple5

4.Synthèse : Développement d'une page WEB interactive

But : Développer les une page WEB interactive en complétant
- les fonctions gérant les interactivités en Javascript
- la définition des évènement déclencheur dans le fichier HTML

Q13. Télécharger le fichier .\code\synthese\code.zip. Décompresser ce fichier dans un répertoire accessible en écriture.

Le répertoire décompressé contient un fichier html et un autre Javascript à compléter.

Q14. Ouvrir les fichiers `activite1.js` et `activite1.html` avec un éditeur de texte (Vscode ou notepad++ par exemple). Ouvrir en parallèle le fichier `activite1.html` sur un navigateur WEB pour contrôler les résultats.

Q15. Modifier ces fichiers afin de suivre les consignes affichées sur le navigateur WEB. En cas de besoin un fichier d'aide est disponible depuis le menu.

Conclusion

Q16. JavaScript favorise la programmation fonctionnelle et évènementielle. **Expliquer** le principe de ce type de programmation.

5.Pour aller plus loin

Un développeur doit créer une page web qui illustre la synthèse additive des couleurs. Pour cela, il voudrait changer la couleur du fond d'écran de la page en agissant sur trois glissières. Chacune doit modifier l'intensité d'une couleur (rouge, vert et bleu) de 0 à 255. Les valeurs décimales de chaque couleur doivent s'afficher au dessous de chacune des glissières.Il veut également afficher les codes couleur CSS au format rgb() et hexadécimal. Il n'arrive pas à venir à bout de son travail, pouvez-vous l'aider à le terminer ?

Pour cela compléter le code contenu dans le fichier `PourAllerPlusLoin.zip`

Eléments du langage Javascript

1. Les variables

Deux méthodes permettent de déclarer une variable : **`let`** et **`const`**

- ✗ **`const`** permet de déclarer une variable qui doit avoir une valeur initiale et ne peut être réassignée
- ✗ **`let`** permet de déclarer une variable qui peut ne pas avoir de valeur initiale et peut être réassignée.

Exemple

```
const a = 'bonjour' ;
let b ;
b =3 ;
```

2.Les conditions

La structure est :

```
if (condition) {
instructions ;
}
else {
instructions ;
}
```

remarque : on peut utiliser if sans else

3.Les boucles

La boucle for

Sa structure est :

```
for (expression initiale, ; condition ; incrémentation){
instructions ;
}
```

Exemple

```
for(let i=0 ; i<5 ; i=i+1){
instructions ;
}
```

Dans cet exemple, les instructions sont répétées 5 fois

La boucle while

Sa structure est :

```
while (condition) {
instructions ;
}
```

Les instructions sont répétées tant que la condition est vraie