NSI – 1ere	COURS Séquence 5B : Interactions avec l'utilisateur dans une page WEB. Introduction à la programmation évènementielle	LFV

Les pages créées en HTML / CSS sont statiques c'est-à-dire que l'interaction avec l'utilisateur est réduite à la possibilité de cliquer sur un lien hypertexte présent dans la page. On peut dynamiser une page web de deux manières :

- soit du côté serveur avec PHP (ou Hypertext Preprocessor) qui peut, par exemple, ajouter le résultat d'une requête à une base de données dans la page qui sera fournie au navigateur; ceci sera traité plus tard.
- soit du côté client avec JavaScript qui peut, par exemple, faire apparaître des info-bulles contextuelles ou réaliser des animations.

I: JavaScript.

JavaScript est un langage créé en 1995 par Brendan Eich qui travaille chez Netscape Communication Corporation principal concurrent à l'époque d'Internet Explorer (en terme de navigateur WEB). JavaScript est alors transmis à l'ECMA (European Computer Manufacturers Association) pour standardisation sous la dénomination d'*ECMAScript* abrégé en ES. Les deux versions, 5 sortie en 2009, et 6 sortie en 2014, sont implémentées dans les différents navigateurs présents sur le marché. Les versions suivantes 7, 8 et 9 sont des mises-à-jour "mineures" de l'ES6 et les fonctionnalités ajoutées ne sont pas supportées par tous les navigateurs.

Le fait que l'exécution du code Java s'exécute sur le Client apporte plusieurs avantages :

- Améliorer l'interactivité (temps de réponse plus court),
- Améliorer les débits sur le réseau (éviter des envois erronés au serveur),
- Proposer des pages un peu plus dynamiques,
- Décharger le serveur de tâches que l'on peut faire sur le poste client.

II : Programmation JS : tour d'horizon

JavaScript s'insère dans les documents HTML entre les balises `<script>…</script>`.
Les fonctions sont **déclarées** dans l'en-tête entre <head> et </head> et sont **appelées** entre <body> et </body>. Cela est cohérent : les appels s'effectuent à partir du contenu affiché du document (entre <body> et </body>) en suivant les indications données dans l'en-tête du document (entre <head> et </head>).

Sur l'exemple ci-dessous on constate qu'en termes **d'actions effectuées** :

- une fonction Javascript peut modifier le contenu d'un élément html grâce à `.innerHTML` (fonction_1),
- une fonction Javascript peut ouvrir une fenêtre surgissante grâce à `alert` (fonction_2),
- une fonction Javascript peut modifier le style d'un élément html grâce à `.style` (fonction_3).

On constate que pour les cas 1 et 3 il faut **indiquer à quel élément html** s'applique la modification souhaitée. Cela peut se faire classiquement de deux façons :

- en indiquant l'id de l'élément que l'on souhaite modifier gâce à document.getElementById (fonction_1),
- en passant le mot clef this en argument : il désigne l'élément html appelant la fonction (fonction_3).

On constate que les appels de fonctions ont lieu lors **d'évènements** bien précis : on parle de programmation évènementielle. Ici lors d'un clic sur des boutons (fonction_1 et fonction_2) ou lorsque la souris entre ou ressort d'un élément html (pour la fonction_3).

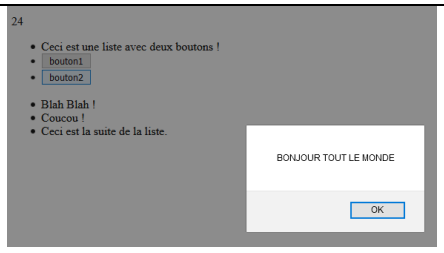
Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle

Enfin on constate que, **comme n'importe quel langage de programmation**, Javascript peut faire des calculs et manipuler des variables (fonction_1). Javascript sait aussi faire des boucles, des instructions conditionnelles etc. mais ce n'est pas au programme.

Remarque : mémorisez bien ce qui précède, il y a quatre principes à constater et comprendre pour commencer en Javascript. Ils sont donnés ci-dessus et il est nécessaire de les intégrer. On peut les résumer ainsi :

évènement --> exécution d'un programme --> actions sur des éléments du document --> qui sont à indiquer

```
<!DOCTYPE html>
<html lang="fr">
<head>
	<title>Cours : exemple 1</title>
	<meta charset="utf-8"/>

	<script>
	function f_1(a, b) {
		document.getElementById('paragraphe_un').innerHTML = (a + b).toString();
	}

	function f_2() {
		alert('BONJOUR TOUT LE MONDE');
	}

	function f_3(couleur) {
		document.getElementById('coucou').style.color = couleur;
	}
	</script>

</head>
<body>
  <p id="paragraphe_un">
	Ceci est le premier paragraphe presque vide.
  </p>

  <ul>
	<li> Ceci est une liste avec deux boutons ! </li>
	<li> <input type="button" value="bouton1" onclick="f_1(13, 11)"/> </li>
	<li> <input type="button" value="bouton2" onclick="f_2()"/> </li>
  </ul>

  <ul>
	<li> Blah Blah ! </li>
	<li id="coucou" onmouseenter="f_3('red')" onmouseleave="f_3('black')"> Coucou ! </li>
	<li> Ceci est la suite de la liste.
	</ul>

</body>
</html>
```

Ceci est le premier paragraphe presque vide. • Ceci est une liste avec deux boutons ! • bouton1 • bouton2 • Blah Blah ! • Coucou ! • Ceci est la suite de la liste.	24 • Ceci est une liste avec deux boutons ! • bouton1 • bouton2 • Blah Blah ! • Coucou ! • Ceci est la suite de la liste.	
Page au chargement	Après un clic sur le bouton 1	Après un clic sur le bouton 2
Et lorsque la souris survole 'Coucou !' , le 'coucou !' devient rouge.		

Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle

III : Programmation JS : en détail

III.1 : Evènements permettant d'appeler les fonctions

Les évènements sont définis dans le code html car ce sont les éléments html qui doivent réagir à des évènements et envoyer l'information aux fonctions Javascript.

Clic sur un bouton

Pour cela il faut déjà définir un élément bouton en html : c'est une balise input l'attribut `type="button"`.
Puis spécifier l'évènement qu inous intéresse : c'est l'attribut `onclick`.
Enfin préciser le texte que l'on veut afficher sur le bouton : c'est l'attribut `value`.
Ce qui donne dans le code html :

```
<input type="button" value="du texte" onclick="nom_fonction()"/>
```

Survol de souris (pas clairement au programme de NSI)

On l'applique à n'importe quel élément html. Il y a de nombreuses variantes : lorsque la souris entre au-dessus de l'élément html, lorsque la souris sort d'au-dessus l'élément etc.
Pour ces deux exemples cela donne, par exemple pour un paragraphe :

```
<p onmouseenter="fonction_a(5, 42)" onmouseleave="fonction_b()"> Blah Blah Blah </p>
```

On pourrait bien entendu n'en mettre qu'un seul des deux et ne pas passer d'arguments à la première fonction. On pourra enfin trouver facilement sur https://www.w3schools.com/js/default.asp une liste plus complète des évènements pris en charge (par exemple `onmousedown et onmouseup`)

III.2 : Syntaxe pour bien définir les fonctions

On utilise :
- un point virgule à la fin de chaque instruction,
- une apostrophe simple ' pour les chaînes de caractères,
- des accolades ouvrantes et fermantes en plus de l'indentation,
- pour les experts, les variables doivent être déclarées avant d'être définies. Ainsi lors de la première définition d'une variable x on écrira `var x = 50;` au lieu de `x = 50` en Python

Remarque : dans tout ce cours les fonctions JS ne renverront aucune valeur et se contenteront de modifier l'affichage de la page web.

III.3 : Sélection d'éléments HTML

On peut utiliser la méthode `document.getElementById(…)` qui permet de sélectionner un élément html à partir de son `id`. Ainsi si dans le code html on a défini une image `<img id="licorne_bleue" src="licorne.jpg">`, on pourra la sélectionner dans le code javascript en utilisant `document.getElementById('licorne_bleue').`

III.4 : Actions pouvant être effectuées par les fonctions

Un élément html sélectionné `elementSelectionne` dans Javascript peut être modifié grâce aux instructions suivantes :
`elementSelectionne.innerHTML = ……` pour modifier son contenu HTML,
`elementSelectionne.style.xxxxx = ……` pour modifier la propriété xxxxx de son style CSS
`elementSelectionne.src = ……` pour modifier son attribut src (voir `cours_exemple_2.html`)
`elementSelectionne.value ( = ……)` pour lire dans un champ de saisie (ou écrire dans un champ de sortie)
Enfin on peut ouvrir une fenêtre de dialogue grâce à la méthode `alert('message d'alerte')`

III.5 : Autres techniques

Tout ce qui précède n'est qu'un tout petit aperçu des possibilités de JS, mais il serait hors-programme d'aller plus loin.

Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle

IV : externaliser le code JS en dehors du fichier HTML

Il suffit de spécifier dans l'en-tête du fichier html où est le fichier js à utiliser grâce à une balise <script></script> (comme avant) munie de l'attribut src pour spécifier l'emplacement du fichier (comme pour les images). Voici un exemple avec les contenus du fichier html puis le contenu du fichier js :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title>Cours : exemple 2</title>
    <meta charset="utf-8">
    <script src="./cours_exemple_2.js"></script>
</head>
<body>
    Blah Blah </br>
    <img onmousedown="change1()" onmouseup="change2()" src="./yeux_1.jpg" id="image_yeux" />
    </br> Piou Piou Piou
</body>
</html>
```

Le fichier cours_exemple_2.js contenant :

```
function change1() {
      document.getElementById('image_yeux').src = './yeux_2.jpg' ;
}
function change2() {
      document.getElementById('image_yeux').src = './yeux_1.jpg' ;
}
```

	
La page au chargement	La page lorsqu'on garde la souris cliquée sur l'image

V : Capacités attendues (ce qu'on attend de vous)

Identifier les composants graphiques permettant d'agir avec une page web.
Identifier les évènements que les fonctions associées aux différents composants graphiques sont capables de traiter.
Analyser et modifier les méthodes exécutées lors d'un clic sur un bouton.

Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle

VI : Pour anticiper sur le prochain cours : entrées / sorties textuelles

En html, les balises <input> servent à fournir des "entrées" à javascript. Nous avons déjà vu les balises <input> de type "button". Il existe aussi les balises <input> de type "text" qui permettent de saisir du texte.
Pour afficher du texte, on utilisera les balises <output> </output> qui peuvent n'afficher que du texte (pas d'attribut de type)

```
<!DOCTYPE html>
<html lang="fr">
<head>
	<title>Cours : exemple 3</title>
	<meta charset="utf-8">
	<script src="./cours_exemple_3.js"></script>
</head>
<body>
	<label for="saisie_1">Nombre A</label>
	<input type="text" id="saisie_1">

	</br>
	<label for="saisie_2">Nombre B</label>
	<input type="text" id="saisie_2">

	</br>
	<label for="sortie"> A * B : </label>
	<output id="sortie"> </output>

	</br>
	<input type="button" value="Calculer !" onclick="calculer()" />
</body>
</html>
```

Voici un exemple :

Et le code js :

Nombre A Nombre B A * B : Calculer !	Nombre A 7 Nombre B 6 A * B : 42
La page au chargement	La page après avoir rempli les champs de saisie et cliqué sur le bouton

VII : Déboguer et commenter son code

Les commentaires s'effectuent en utilsiant un double slash en début de ligne : `// commentaires blah blah`
On utilisera avec profit les outils de développement web de Firefox ou de Chrome, et en particulier la console, pour déboguer son code si besoin.

```
function calculer(){
	var a = document.getElementById('saisie_1').value ;
	var b = document.getElementById('saisie_2').value ;
	var resultat = a * b ;
	document.getElementById('sortie').value = resultat ;
}
```

Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle

Feuille de synthèse

Évènements appelant les fonctions (depuis HTML)

```
Le bouton (au programme) : <input type="button" value=" texte" onclick="nom_fonction()"/>

Autres éléments html     : <p onmouseenter="nom_fonction()"> Blah Blah Blah </p>
                           On peut aussi utiliser onmouseleave, onmousedown, onmouseup etc.
```

Sélectionner des éléments html (depuis JS)

```
document.getElementById('licorne_bleue')
```

Syntaxe du code JS

```
Utilisation d'accolades, de points virgule, des
mots-clefs function et var etc.

function calc(a, b){
      var c = a * b ;
      document.getElementById('x').value = c ;
}
```

Actions depuis JS pouvant opérer sur un élémént html sélectionné

`elementSelectionne.innerHTML = ……` pour modifier son contenu HTML
`elementSelectionne.style.xxxxx = ……` pour modifier la propriété xxxxx de son style CSS
`elementSelectionne.src = ……` pour modifier son attribut src (voir `cours_exemple_2.html`)
`elementSelectionne.value ( = ……)` pour lire dans un champ de saisie (ou écrire dans un champ de sortie)

On peut aussi ouvrir une fenêtre de dialogue grâce à la méthode `alert('message d'alerte')`

Utilisation de champs de saisie et d'affichage : un cas classique

HTML

```
<label for="saisie">Nombre B</label>
<input type="text" id="saisie">
</br>
<label for="sortie"> 2 * B : </label>
<output id="sortie">
</br>
<input type="button" value="double" onclick="doubler()" />
```

JS

```
function doubler(){
    var a = document.getElementById('saisie').value ;
    var resultat = 2 * a ;
    document.getElementById('sortie').value = resultat ;
}
```

Autres

```
Externalisation du code avec <script src="./nom_du_fichier.js"></script>
Commentaires avec // en début de ligne
Déboguage en utilisant les outils de développement web et la console en particulier pour JS
Utilisez W3Schools pour de nombreux exemples
```

VIII : Partie TP

Reproduire en HTML + JS ce qui est indiqué dans les vidéos fournies en utilisant les fichiers html, css et js fournis à compléter.
Pour le css, vous pouvez le laisser tel quel.
(L'IMC (Indice de Masse Corporelle) se calcule avec la formule Masse / (Taille^2) où la taille est en mètres.)

Licence CC-NC – Version Initiale par C. Lejeune & J. de Villèle