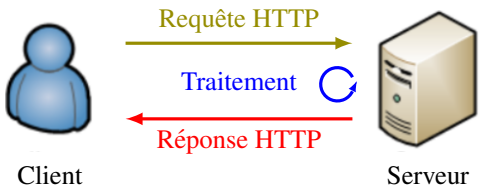
1 Rappels sur le modèle client/serveur

On a vu dans le cours précédent que la consultation de pages sur un site fonctionne sur une architecture client/serveur :

- un internaute connecté au réseau via son ordinateur et un navigateur web est le client ;
- le serveur correspond aux ordinateurs contenant les applications qui fournissent les pages demandées ;
- le client et le serveur communiquent entre eux en utilisant un protocole de communication (`HTTP`).



On distingue deux grandes familles de sites web :

Définition 1.1 (Interactif vs dynamique)

- Les sites web dits « interactifs » sont les sites web pour lesquels les modifications des pages ont lieu *uniquement* du côté client.
- Les sites web dits « dynamiques » sont les sites web pour lesquels des modifications de pages ont lieu du côté serveur.

2 Qu'est-ce que le langage `JavaScript` ?

Le principe général du langage `JavaScript` est de créer des sites web interactifs, c'est-à-dire des sites web qui puissent afficher du contenu en fonction d'événements réalisés côté client. C'est un langage de programmation libre créé en 1995 par Brendan Eich de la famille des langages interprétés orientés objets. La majorité des navigateurs web dispose d'un interpréteur `JavaScript` intégré.

C'est également un langage compatible avec le DOM des pages web, c'est-à-dire qu'il peut agir directement sur les éléments du DOM et les modifier.

3 Insertion d'un code `JavaScript`

Il existe deux possibilités pour insérer un code `JavaScript` dans un document `HTML` :

- En l'écrivant entre les balises `<script>` et `</script>` du conteneur `<head>`. On utilise alors la syntaxe suivante :

  `<script>`*`code JavaScript`*`</script>`

- En utilisant un fichier externe d'extension `.js`. On utilise alors la syntaxe suivante :

  `<script src=`*`nom_fichier`*`></script>`

  Le paramètre `nom_fichier` est obligatoire et est une chaîne de caractères correspondant au chemin relatif depuis le fichier `HTML` source.

4 Syntaxes élémentaires en `JavaScript`

4.1 Instructions et variables

- **Une instruction** se termine toujours par un point-virgule `;`
- **Le nom d'une variable** commence toujours par le mot-clé `let`
- **Affecter une valeur à une variable `x` :** `let x= ...`

Exemple 4.1 L'instruction `let x = 2;` permet de définir une variable `x` et de lui affecter la valeur 2.

4.2 Les tableaux

Les tableaux `JavaScript` sont des données structurées analogues aux listes `Python`. Ils sont définis par des crochets et ses éléments sont séparés par des virgules.

- *`tableau`*`[i]` : renvoie l'élément situé à l'index `i` du tableau (les index commencent à 0).
- *`tableau`*`.length` : renvoie la longueur du tableau.

Exemple 4.2 On considère le tableau `let tab = [5,3,15].`

- `tab.length` renvoie la valeur 3 ;
- `tab[1]` renvoie la valeur 3 ;
- `tab[0]=1` modifie le premier élément du tableau et lui affecte la valeur 1.

4.3 Structure d'embranchement `si ... alors ... sinon`

Syntaxe	Exemple

`if (condition)`
`{`
`  instruction 1;`
`  ...`
`  instruction n;`
`}`
`else`
`{`
`  instruction 1;`
`  ...`
`  instruction m;`
`}`
	Affichage du signe d'une variable `x` :
	`if (x<=0)`
	`{`
	`  alert('négatif');`
	`}`
	`else`
	`{`
	`  alert('positif');`
	`}`

Remarque 4.3 La fonction `alert(v)` permet d'afficher dans une boîte de dialogue le contenu de la variable `v`.

4.4 Structure de boucle itérative `pour`

Syntaxe	Exemple

`for(init; condition ; incrémentation)`
`{`
`  instruction 1;`
`  ...`
`  instruction n;`
`}`
	Affichage des éléments d'un tableau en le parcourant sur les index :
	`let tab = [3,2,1,'Partez !!!'];`
	`for(let i=0; i<texte.length; i++)`
	`{`
	`  alert(tab[i]);`
	`}`

4.5 Structure de boucle conditionnelle `tant que`

Syntaxe	Exemple

`while(condition)`
`{`
`  instruction 1;`
`  ...`
`  instruction n;`
`}`
	Affichage des éléments d'un tableau en le parcourant sur les index :
	`let tab = [3,2,1,'Partez !!!'];`
	`let i=0;`
	`while(i<texte.length)`
	`{`
	`  alert(tab[i]);`
	`  i += 1;`
	`}`

4.6 Les fonctions

Les fonctions `JavaScript` se manipulent comme les fonctions `Python`. Leur seule différence réside dans la syntaxe. En `JavaScript`, la syntaxe d'une fonction est la suivante :

```
function nom_de_la_fonction(liste_paramètres )
{
   instruction 1;
   ...
   instruction n;
   return liste_valeurs
}
```

5 Gestion du DOM

5.1 Attributs événementiels

La plupart des éléments `HTML` admettent des attributs événementiels de façon à ce qu'un événement `JavaScript` soit activé :

- `onclick` : exécute un code `JavaScript` lorsque l'on clique gauche avec la souris ;
- `ondblclick` : exécute un code `JavaScript` lorsque l'on double-clique gauche avec la souris ;
- `onmousedown` : exécute un code `JavaScript` lorsque l'on clique sur un bouton de la souris ;
- `onmouseenter` : exécute un code `JavaScript` lorsque le pointeur de la souris entre dans l'élément ;
- `onmouseleave` : exécute un code `JavaScript` lorsque le pointeur de la souris quitte l'élément ;
- `onmousemove` : exécute un code `JavaScript` lorsque le pointeur de la souris est présent dans l'élément et déplacé à l'intéreur de celui-ci ;
- `onmouseout` : exécute un code `JavaScript` lorsque le pointeur de la souris est à l'extérieur de l'élément ;
- `onmouseover` : exécute un code `JavaScript` lorsque le pointeur de la souris entre dans l'élément ou dans l'un de ses fils ;
- `onmouseup` : exécute un code `JavaScript` lorsque l'on relâche un des boutons de la souris.

Remarque 5.1 Tous ces attributs ne sont pas valables pour tous les éléments `HTML`. On consultera internet pour plus de détails.

Pour utiliser ces attributs, on utilise la syntaxe suivante :

```
<nom_balise nom_attribut_even="code_JavaScript">
```

Exemple 5.2

- Le code `HTML`

```
<button onclick="alert('Bonjour !');">Hello World</button>
```

permet de créer un bouton contenant le texte `Hello World` et affichant dans une boîte de dialogue le texte `Bonjour !` lorsque l'on clique dessus.

- Le code `HTML`

```
<span onclick="javascript:location.href='https://www.google.fr'">Google</span>
```

permet de spécifier le texte `Google` pour que l'on puisse accéder au moteur de recherche `www.google.fr` lorsque l'on clique dessus.

5.2 Les sélecteurs `JavaScript`

Pour pouvoir manipuler le DOM, il est nécessaire de récupérer les éléments `HTML` sur lesquels on veut agir. On dispose de plusieurs fonctions suivant le type de sélection que l'on veut faire :

- `document.getElementById(`*`nom_id`*`)` : renvoie l'élément d'identifiant `nom_id` s'il existe et `null` sinon.
- `document.getElementsByClassName(`*`nom_class`*`)` : renvoie un tableau de tous les éléments dont l'attribut `class` est `nom_class` et un tableau vide s'il n'en existe pas.
- `document.getElementsByTagName(`*`nom_balise`*`)` : renvoie un tableau de tous les éléments dont les noms de balises sont égaux à `nom_balise` et un tableau vide s'il n'en existe pas.

Exemple 5.3

- L'instruction `document.getElementById('Rouge')` renvoie l'élément d'identifiant `Rouge` et `null` s'il n'existe pas.
- L'instruction `document.getElementsByTagName('h1')` renvoie un tableau contenant tous les éléments de balises `<h1>` et un tableau vide s'il n'en existe pas.

Remarque 5.4 Un tableau vide est un tableau de longueur 0.

Remarque 5.5 On peut aussi utiliser le mot-clé `this` pour pouvoir manipuler directement l'élément admettant l'attribut événementiel (on le passe en général en paramètre de la fonction à exécuter).

Une fois le ou les éléments sélectionnés, on peut agir dessus : soit en modifiant le contenu, soit en modifiant un attribut existant (style, border, width, etc.).

5.3 Modification du contenu d'un élément

- *elem*.innerHTML : renvoie une chaîne de caractères contenant le contenu textuel de l'élément elem défini préalablement par un sélecteur.
- *elem*.innerHTML = *nouveau_texte* : remplace le contenu textuel de l'élément elem défini préalablement par un sélecteur par la chaîne de caractères nouveau_texte.

  Exemple 5.6 L'instruction E.innerHTML = 'Texte modifié' remplace sur la page web le texte initial de l'élément E par le texte Texte modifié.

- *elem*.insertAdjacentHTML(*pos*,*texte_ajout*) : ajoute au contenu textuel de l'élément elem défini préalablement par un sélecteur le texte texte_ajout à la position pos. Cette position peut-être 'beforebegin', 'afterbegin', 'beforeend' ou 'afterend' comme sur le schéma suivant :

```
<!-- beforebegin -->
<nom_balise>
   <!-- afterbegin -->
   texte existant
   <!-- beforeend -->
</nom_balise>
<!-- afterend -->
```

5.4 Modification du style d'un élément

- window.getComputedStyle(*elem*).getPropertyValue(*attribut_css*) : renvoie la valeur CSS de l'attribut attribut_css de l'élément elem préalablement défini à l'aide d'un sélecteur.
- *elem*.style.*attribut_css* = *val* : affecte la valeur val à l'attribut attribut_css de l'élément elem préalablement défini à l'aide d'un sélecteur.

Exemple 5.7

- L'instruction window.getComputedStyle(element).getPropertyValue('color') renvoie la valeur de l'attribut color de l'élément element.
- L'instruction E.style.display='block' permet d'attribuer le style block à l'élément E.
- L'instruction E.style.color='red' permet d'attribuer la couleur rouge à l'élément E.
- L'instruction E.style.backgroundColor='blue' permet d'attribuer la couleur bleu à l'arrière-plan de l'élément E.

Remarque 5.8

- Les attributs CSS s'écrivant avec des traits d'union sont en général écrits tout attaché avec des majuscules pour différencier les différents mots. Par exemple, background-color est l'attribut CSS, mais on le recherche sous la forme backgroundColor.
- Les couleurs sont toujours renvoyées sous la forme 'rgb(r, v, b)' (ne pas oublier les espaces).

5.5 Modification d'un attribut quelconque d'un élément

- *elem*.*nom_attribut* : renvoie une chaîne de caractères contenant la valeur de l'attribut nom_attribut de l'élément elem défini préalablement par un sélecteur.
- *elem*.*nom_attribut* = *val* : affecte la valeur val (donnée sous forme d'une chaîne de caractères) à l'attribut nom_attribut de l'élément elem défini préalablement par un sélecteur.

Exemple 5.9 On considère l'élément HTML suivant (barre de progression) :

```
<progress id='barre' value='0' max='100'>0</progress>
```

On récupère cet élément par un sélecteur sur l'identifiant : var barre = document.getElementById('barre');
On peut agir sur l'attribut value de la façon suivante :

- barre.value : renvoie la valeur de l'attribut value, c'est-à-dire '0' ;
- barre.value = '25' : affecte à l'attribut value la valeur '25'.