JavaScript

JavaScript a été créé pour animer les pages web, il peut maintenant être également utilisé pour programmer dans de très nombreux environnements : serveurs, applications mobiles, objets connectés...

Prérequis : disposer d'un navigateur web moderne comme Firefox, Chrome ou Microsoft Edge, mais pas Internet Explorer qui ne gère pas correctement le code JavaScript.

1 JavaScript : les bases

1.1 Structure d'un fichier contenant du JavaScript

Le code JavaScript vient s'insérer au sein du code HTML. La structure d'un fichier HTML qui utilise du code JavaScript est classique :

- le doctype ;
- les balises `<html>` et `</html>` ;
- les balises qui définissent le début et la fin de l'en-tête du fichier : `<head>` et `</head>`. On y insère un lien vers le fichier CSS qui définira la mise en forme de la page web `mef.css` ;
- les balises `<body>` et `</body>` qui définissent le début et la fin du contenu de la page web.

Pour insérer le code JavaScript on utilise les balises `<script>` et `</script>` comme l'illustre le code du listing 1.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                         "http://www.w3.org/TR/xhtml1-strict.dtd">
2
3  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
4      <head>
5          <!-- En tete du fichier HTML -->
6          <title>JavaScript</title>
7          <link href="mef.css" rel="stylesheet" media="all" type="text/css">
8          <meta charset="utf-8" />
9          <meta http-equiv="content-type" content="text/html; charset=UTF-8"/>
10     </head>
11     <body>
12         <script>
13             // Commentaire : c'est ici que le code JavaScript apparaitra
14         </script>
15     </body>
16 </html>
```

FIGURE 1 – Structure d'une page contenant du JavaScript

1.2 Commentaires

Comme on peut le constater sur le listing 1, les commentaires en JavaScript s'écrivent derrière un `//` (commentaire d'une seule ligne). On peut également les insérer entre `/*` et `*/` pour des commentaires de plusieurs lignes par exemple.

1.3 La console

Le JavaScript est un langage de programmation, on peut visualiser des informations dans la console du navigateur. Par exemple pour afficher la console sur Firefox il suffit d'utiliser le raccourci clavier *CTRL+MAJ+K*.

Le listing 2 permet d'afficher `Salut le Monde` dans la console.

```
13          <script>
14              // afficher des information dans la console
15              console.log("Salut le Monde");
16          </script>
```

FIGURE 2 – Affichage de texte dans la console du navigateur

1.4 Affichage de texte sur la page web

Le listing 3 permet d'afficher `Hello World` sur la page web.

```
13          <script>
14              // afficher du texte dans la page
15              document.write("Hello World");
16          </script>
```

FIGURE 3 – Affichage de texte sur la page Web

1.5 Variables

On déclare une variable en écrivant `var` ou `let` devant le nom de la variable. Pour une variable qui n'a pas vocation à changer de valeur au cours du programme, on la déclare avec le mot clé `const`. Une variable en JavaScript peut être de type **nombre**, de type **chaine de caractères** ou de type **booléen**. Les booléens peuvent prendre les valeurs `true` ou `false`. Une variable qui est déclarée mais pas initialisée sera de « type » **undefined**. Un exemple de définition de variables est donnée à la figure 4.

```
13          <script>
14              var chaine = "Hello World"; // string
15              var nombre = 3.14; // number
16              var x; // undefined
17          </script>
```

FIGURE 4 – Déclaration de variables en JavaScript

1.6 Fonctions et portée des variables

Une fonction se déclare grâce au mot clé `function`. Dans l'exemple de la figure 5 on définit une fonction qui retourne le résultat de la multiplication de deux variables. La portée des variables n'est pas la même suivant que la variable est définie hors de la fonction (variable globale) ou dans la fonction (variable locale).

```
13          <script>
14              var a = 3;
15              var b = 4;
16              function mult() {
17                  var ab = a*b;    // les variables a et b sont visibles
18                  return ab;       // la variable ab est visible
19              }
20              var res = mult();    // la variable ab n'est plus visible
21              document.write("a*b = ", a, "*", b, " = ", res, "</br>");
22          </script>
```

FIGURE 5 – Définition d'une fonction et portée des variables

1.7 Convention de nommage

Tout nom de variable doit commencer par une lettre minuscule. Si le nom d'une variable est constitué de plusieurs mots, les mots sont accolés et chaque mot commence par une majusule (sauf le premier mot) : `maVariablePreferee`.

1.8 Utilisation d'un fichier JavaScript

Afin d'organiser sa programmation il est utile de définir les variables et les fonctions dans des fichiers à part. On insère alors le contenu du fichier au sein d'une balise ouvrante `<script>` qu'on referme immédiatement (cf. figure 6 et 7).

```
1  // definition des fonctions
2  function mult(a, b) {
3      return a*b;
4  }
```

FIGURE 6 – Contenu du fichier `fAndVar.js`

```
13          <!-- insertion des fonctions et variables du fichier javascript.js -->
14          <script src="fAndVar.js"> </script>
15          <script>
16              var c = 5;
17              var d = 6;
18              // appel de la fonction mult() definie dans le fichier fAndVar.js
19              document.write("c*d = " + mult(c,d));
20          </script>
```

FIGURE 7 – Utilisation dans le fichier HTML

Tout le programme JavaScript peut être écrit dans un fichier d'extension `.js`. Dans le fichier HTML il n'y a alors plus que les balises `<script>` utilisées pour l'insertion du fichier JavaScript. Il est d'usage d'insérer le fichier JavaScript à la fin du *Body* du fichier HTML lorsque que c'est possible.

```
13      <body>
14          <!-- Contenu de la page HTML -->
15          <!-- Contenu de la page HTML -->
16          <!-- Contenu de la page HTML -->
17
18          <!-- insertion des fonctions et variables du fichier javascript.js -->
19          <script src="fAndVar.js"> </script>
20      </body>
```

FIGURE 8 – Le code JavaScript dans un fichier et insertion dans le fichier HTML

1.9 Opérateurs

Les opérateurs classiques sont similaire que dans la plupart des langage de programmation haut-niveau. Attention le + a une double fonction, l'addition et la concaténation. Quelques exemples sont donnés à la figure 9.

```
15          <script>
16              var a = 5 + 6; // addition
17              var b = "Hello" + " World" // concatenation
18              var c = 6 - 5; // soustraction
19              var d = 6 * 5; // multiplication
20              var e = 12 / 6; // division
21              var f = 12 % 5; // modulo
22          </script>
```

FIGURE 9 – Les opérateurs en JavaScript

1.10 Tableaux

Les tableaux permettent de stocker plusieurs valeurs dans un seul élément. Les cases du tableau sont repérés par un indice, la première case d'un tableau à pour indice 0. Les tableaux peuvent être déclarés et initialiser de plusieurs manières illustrées à la figure 10.

```
15          <script>
16              var tableau1 = new Array(10); // creates an array of 10 slots
17              var tableau2 = ["blue", "red", 12, 13]; // creates an array of 4 slots
18              document.write(tableau1[0]); // prints the first element of the array
19          </script>
```

FIGURE 10 – Déclaration des tableaux en JavaScript

Comme le montre la figure 11 les tableaux sont des objets qui possèdent des méthodes, les plus utiles sont :

- `pop()` : ôte du tableau le dernier élément du tableau ;
- `push("Saperlipopette")` : ajoute un élément dans un tableau (ici la chaîne de caractères `Saperlipopette`) ;
- `shift()` : ôte le premier élément et glisse le reste du tableau d'un cran vers la gauche ;
- `unshift("Bachibouzouk")` : décale le contenu du tableau d'un cran vers la droite et ajoute la chaîne de caractère `Bachibouzouk` au début du tableau.

Leur utilisation est illustrée à la figure 11.

```
15          <script>
16              <!-- Tableaux -->
17              var tableau = ["blue", "red", 12, 13]; // creates an array of 4 slots
18              tableau.pop(); // remove 13 from the array
19              tableau.push("end"); // append "end" in the array
20              tableau.shift(); // remove "blue" and shift the rest to the left
21              tableau.unshift("begining"); // insert "begining" in the begining of the array
22              // final array : "begining", "red", 12, "end"
23
24              //Affichage du contenu d'un tableau
25              for(const value of tableau){
26                  console.log(value);                 // affichage dans la console
27                  document.write(value + "</br>");    // affichage dans la page web
28              }
29          </script>
```

FIGURE 11 – Les méthodes des tableaux en JavaScript

Exercice n°11 page 18

2 Structures conditions

2.1 Opérateurs de comparaison

Les opérateurs qui permettent d'utiliser les structures conditions, et qui retournent un booléen sont :

- == : égal à ?
- != : différent de ?
- < : plus petit que ?
- <= : plus petit que ou égal à ?
- > : plus grand que ?
- >= : plus grand que ou égal à ?
- === : est égal à et est du même type que ?

2.2 Opérateurs booléens

Les opérateurs correspondant aux fonctions booléennes sont :

- `&&` : opérateur logique ET ;
- `||` : opérateur logique OU.

2.3 Structure condition if... else if... else...

La structure condition permet d'exécuter une partie de programme si la condition est vraie (si le retour du test est True) (cf. figure 12).

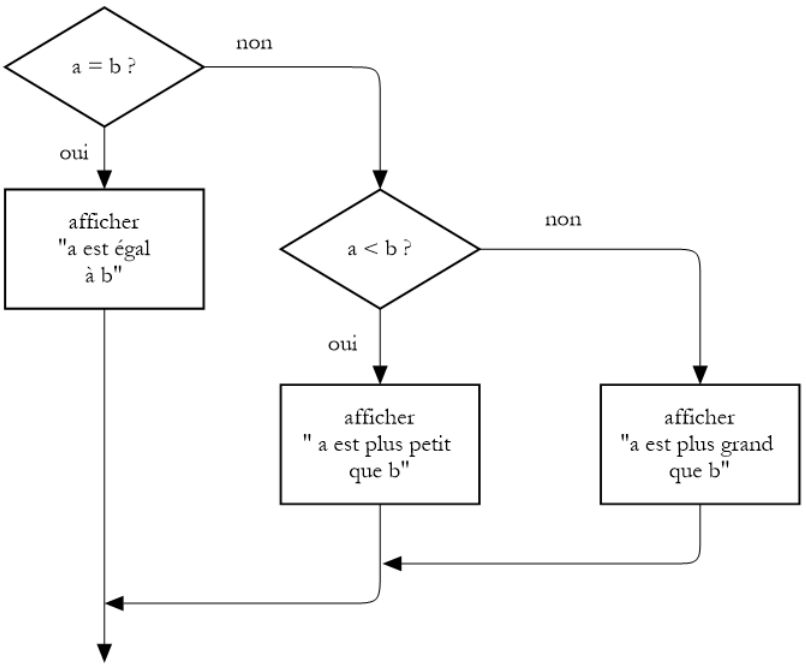

FIGURE 12 – Ordinogramme de la structure condition (comparaison des valeurs de a et b)

Les mots clés de la structure condition en JavaScript sont :

- `if(condition 1)` : si la condition 1 est vérifiée, une partie du programme est exécutée ;
- `else if(condition 2)` : sinon si la condition 2 est vérifiée, une autre partie du programme est exécutée ;
- `else` : et sinon c'est encore une autre partie du programme qui est exécutée.

En algorithmique la structure condition est décrite à la figure 13. Le programme correspondant est donné en JavaScript à la figure 14.

```
1  Algorithme Structure condition
2  variable
3      a, b : entier
4  début
5      a <-- 5
6      b <-- 6
7      si a == b alors
8          début
9              afficher("a et b sont égaux.")
10         fin
11     sinon si a < b alors
12         début
13             afficher("a est plus petit que b.")
14         fin
15     sinon
16         début
17             afficher("a est plus grand que b.")
18         fin
19 fin.
```

FIGURE 13 – Structure condition si... sinon si... sinon...

```
15          <script>
16              var a = 5;
17              var b = 6;
18              if(a == b){
19                  document.write("a et b sont égaux");
20              }
21              else if(a < b){
22                  document.write("a est plus petit que b");
23              }
24              else{
25                  document.write("a est plus grand que b");
26              }
27          </script>
```

FIGURE 14 – Structure condition if... else if... else...

2.4 Structure condition switch... case...

Lorsque la partie du programme qui doit être exécutée est déterminée par la valeur d'une variable on va plutôt utiliser une structure de type switch... case... (cf. figure 15).

```
16          <script>
17              let repasCantine = "couscous"; // today's meal
18              switch(repasCantine){
19                  case "couscous":
20                      document.write("Il va y avoir du monde, mais j'y vais quand
                          même");
21                      break;
22                  case "choux fleurs":
23                      console.log("Le dîtes pas à mum's, je mange une pizza !");
24                      break;
25                  default:
26                      documen.write("Surprise !");
27              }
28          </script>
```

FIGURE 15 – Structure condition switch... case...

Exercice n°12 page 19

3 Structures répétitives

3.1 Boucle for

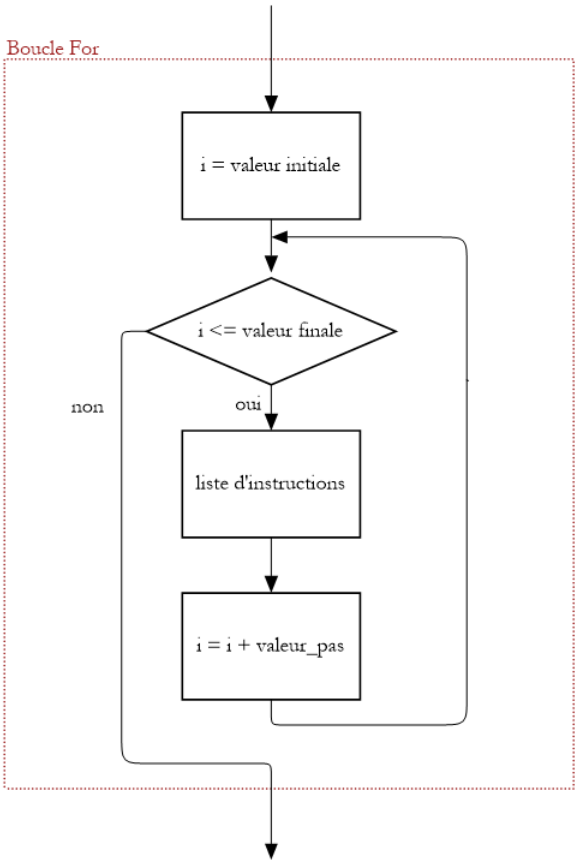

FIGURE 16 – Ordinogramme de la structure répétitive : la boucle `for`

La boucle `for` est utilisée lorsqu'on sait combien de fois on veut répéter la boucle. Elle s'exprime sous la forme `for(exp1; exp2; exp3)` où :

- `exp1` : est l'initialisation de la variable qui permettra de compter les itérations de la boucle (par exemple `var i = 0`);
- `exp2` : est la condition, la boucle s'exécutera tant que l'expression exp2 retournera `true` (par exemple `i < 10`);
- `exp3` : permet d'indiquer l'action à appliquer sur la variable d'itération à la fin de chaque tour de boucle (par exemple l'incrémenter : `i++`).

Dans l'exemple des figures 17 et 18 on répète dix fois la boucle afin d'afficher les valeurs de i (0 à 9).

```
1  Algorithme Structure répétitive (boucle for)
2  variable
3      i : entier
4  début
5      pour i variant de 0 à 9 par pas de 1 faire
6      début
7          afficher('Valeur de i : ', i, alaligne)
8      fin
9  fin.
```

FIGURE 17 – Boucle for en algorithmique

```
15          <script>
16              for(var i = 0; i < 10; i++){
17                  document.write("Valeur de i : ", i, "</br>");
18              }
19          </script>
```

FIGURE 18 – Boucle for en JavaScript

3.2 Boucle while

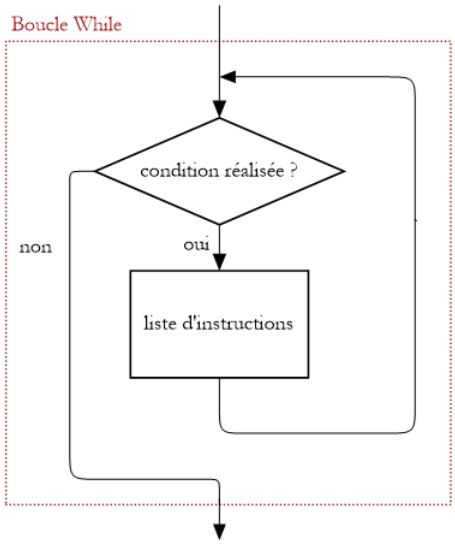

FIGURE 19 – Ordinogramme de la structure répétitive : la boucle `while`

Lorsqu'a priori on ne sait pas combien de fois doit s'exécuter la boucle, on utilise une boucle `while`. La boucle while est construite de la manière suivante `while(condition)`, elle s'exécute tant que l'expression testée renvoie `true`.

```
1   Algorithme Factorielle (boucle while)
2   variable
3       i, factorielle, factDe : entier
4   début
5       factDe ← 5          co valeur dont on veut calculer la factorielle fco
6       i ← factDe
7       factorielle ← 1     co stocke la valeur de la factorielle au fur et a mesure de
        son calcul fco
8       tant que i > 1 faire
9       début
10          factorielle ← factorielle * i
11          i ← i-1
12      fin
13      afficher('Factorielle de ', factDe, ' : ', factorielle, alaligne)
14  fin.
```

FIGURE 20 – Boucle while en algorithmique

```
15          <script>
16              var factorielle = 1
17              var i = 4//variable d'itération
18              while(i != 1){
19                  factorielle *= i--;
20              }
21              document.write("Factorielle : ", factorielle, "</br>");
22          </script>
```

FIGURE 21 – Boucle while en JavaScript

3.3 Break et continue

L'instruction `break` interrompt la boucle ; L'instruction continue interrompt l'itération en cours et passe à la suivante. Ces deux instructions sont illustrées à la figure 22.

```
15                  // illustration de la commande break
16                   for(var i = 0; i < 6; i++){
17                       if(i == 3) break;
18                       document.write(i);
19                  } // affiche 0 1 2
20                  // illustration de la commande continue
21                   for(var i = 0; i < 6; i++){
22                       if(i == 3) continue;
23                       document.write(i);
24                  } // affiche 0 1 2 4 5
```

FIGURE 22 – Instructions `break` et `continue`

Exercice n°?? page ??

4 Interactions avec l'utilisateur

4.1 Boites de dialogue

Pour ouvrir une boite de dialogue il faut utiliser l'instruction `prompt("texte à afficher")` qui retourne une chaine de caractères qu'on peut stocker dans une variable. Dans l'exemple de la figure 23 :

- j'ouvre une boite de dialogue (instruction `prompt()`) dans laquelle je demande à l'utilisateur de saisir son prénom ;
- lorsque l'utilisateur valide, la valeur saisie est stockée dans la variable `prenom` ;
- dans une deuxième boite de dialogue je lui demande de saisir son âge ;
- et pour finir j'affiche dans une troisième boite de dialogue le prénom et l'âge de l'utilisateur gâce à l'instruction `alert()`.

```
15              <script>
16                  const prenom = prompt("Quel est ton prénom ?");
17                  const age = prompt("quel est ton âge ?");
18                  alert("Bonjour " + prenom + " tu as " + age + " ans.");
19              </script>
```

FIGURE 23 – Boites de dialogues

Si on veut que l'utilisateur saisisse un nombre qui soit utilisable dans un calcul, il faudra convertir la variable qui stocke la saisie en nombre, grâce à l'instruction `Number()`, car l'instruction `prompt()` retourne une chaîne de caractères. C'est illustré dans la figure 15. La conversion en chaîne de caractère se fait avec l'instruction `String`.

Exercice n°3 et 6 page 19

4.2 Bouton cliquable

La figure 24 illustre l'insertion d'un bouton cliquable dans une page du site web.

```
9   <body>
10      <!-- Creation of a button identified by the tag "bouton" -->
11      <button id="bouton">Clic, clic, clic...</button>
12
13      <!-- Insertion of the code JavaScript inside the page of the website -->
14      <script src="js/03_bc.js"></script>
15  </body>
```

FIGURE 24 – Insertion d'un bouton cliquable

La figure 25 est le contenu du fichier JavaScript. On commence par y définir une fonction `clic()` qui affiche du texte dans la console : *"Tu as cliqué !"* . La méthode `addEventListener` prend deux paramètres : le type de l'événement

et la fonction qui gère l'événement. Il est utilisé sur le corps de la page web (entre les balises `body`, c'est à dire sur la variable globale `document` du JavaScript). Dans notre exemple le clic sur le bouton (événement `click`) déclenche l'appel de la fonction `clic()`.

```
function clic() {
    console.log("Tu as cliqué !");
}

// Recuperation de l'action sur le bouton qui se trouve
// dans le body (document.) de la page.
var boutonElt = document.getElementById("bouton");
// Ajout d'un gestionnaire pour l'evenement click
boutonElt.addEventListener("click", clic);
```

FIGURE 25 – Utilisation du bouton cliquable

Pour que le clic sur le bouton *Clic, clic, clic...* ne soit plus géré par le gestionnaire d'événement il faut appeler la méthode `removeEventListener()` (cf. figure 26).

```
// Suppression du gestionnaire pour l'evenement click
boutonElt.removeEventListener("click", clic);
```

FIGURE 26 – Ne plus géré le bouton cliquable avec la fonction `clic()`

D'autres exemples d'événement utilisables sont : `"mouseover"` et `"mouseout"`. Tous les événements scrutables sont listés sur la page web :
`https://developer.mozilla.org/en-US/docs/Web/Events#Mouse_events`.

4.3 Fonction anonyme

Une fonction anonyme est une fonction qui n'a pas de nom, elle se définit directement dans le gestionnaire d'événement comme le montre la figure 27. Attention, l'utilisation de fonction anonyme ne permet pas la suppression des gestionnaires d'événement.

```
var boutonElt = document.getElementById("bouton");
// Utilisation d'une fonction anonyme
boutonElt.addEventListener("click", function () {
    console.log("Tu as cliqué !");
});
```

FIGURE 27 – Utilisation du bouton cliquable

4.4 Appui sur une touche du clavier

Le code de la figure 28 permet d'afficher dans la console la touche appuyée au clavier : le type d'événement à utiliser est `keypress`. L'information sur le caractère est fournie par la propriété `charCode` de l'objet `evenement`. La méthode `String.fromCharCode` permet de traduire la valeur du caractère un une chaîne représentant le caractère.

```
// Gestion de l'appuie sur les touches du clavier
var x = "a";
document.addEventListener("keypress", function (evenement) {
    console.log("Touche appuyée : " + String.fromCharCode(evenement.charCode));
});
```

FIGURE 28 – Gestion de l'appui sur une touche du clavier

Il est également possible de détecter l'appuie ou le relâchement d'une touche (cf. figure 29).

```
// Affiche des informations sur un evenement clavier
function infosClavier(e) {
    console.log("Evènement clavier : " + e.type + ", touche : " + e.keyCode);
}
// Gestion de l'appui et du relachement d'une touche du clavier
```

FIGURE 29 – Appui ou relâchement d'une touche du clavier

4.5 Clic sur la page du site web

L'objet `Event` associé à un événement de type `click` contient les propriété suivantes :

- button : renvoie le bouton de la souris utilisé ;
- clientX : renvoie la coordonnée horizontale du clic ;
- clientY : renvoie la coordonnée verticale du clic.

```
// prints the informations of the events
function infosSouris(event){
    console.log("Evènement souris : " + event.type + ", bouton " +
        event.button + ", X : " + event.clientX + ", Y : " + event.clientY);
}

// Management of the mouse
document.addEventListener("click", infosSouris);
```

FIGURE 30 – Appui ou relâchement d'une touche du clavier

Exercice n°4 page 19

4.6 Fermeture de la page web

Par exemple pour afficher une demande de confirmation de fermeture de la page web on peut utiliser le code de la figure 31.

```
// Gestion de la fermeture de la page web
window.addEventListener("beforeunload", function (event) {
    var message = "Ne me quitte pas...";
    event.returnValue = message; // Provoque une demande de confirmation (standard)
    return message; // Provoque une demande de confirmation (certains navigateurs)
});
```

FIGURE 31 – Événement fermeture de la page web

4.7 Autres événements...

Il existe de nombreux événements dont la gestion n'a pas été abordé dans le cours, pour les plus intéressés... *Qwant is your friend !*

5 Animer une page web

5.1 Autodestruction de la page

Le code de la page 32 permet de réaliser un compteur qui se décrémente automatiquement grâce au code JavaScript. Les nouveautés dans ce code sont :

- l'utilisation de la fonction `setInterval(fonction, délai)` permet d'exécuter *fonction()* de manière répétitive toutes les *délai* millisecondes ;
- l'utilisation de la fonction `setTimeOut(fonction, délai)` qui permet d'exécuter la *fonction* une seule fois, une fois que *délai* millisecondes est écoulé.

```javascript
// compteurElt designe l'element compris entre les balises <span>
var compteurElt = document.getElementById("compteur");

// Diminue le compteur jusqu'a 0
function diminuerCompteur() {
    // Conversion en nombre du texte du compteur
    var compteur = Number(compteurElt.textContent);
    if      compteurElt.textContent = compteur - 1;
    }
    else {
        // Annule l'execution repetee
        clearInterval(intervalId);
        // Modifie le titre de la page
        var titre = document.getElementById("titre");
        titre.textContent = "BOUM !!!";
        // Modification du titre au bout de 2 secondes
        setTimeout(function () {
            titre.textContent = "Tout est cassé :(";
        }, 2000);
    }
}

// Appelle la fonction diminuerCompteur toutes 1000 millisecondes
var intervalId = setInterval(diminuerCompteur, 1000);
```

FIGURE 32 – JavaScript : Ca va sauter !

```html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
    <head>
        <title>Ca va sauter !</title>
        <meta charset="utf-8" />
    </head>
    <body>
        <h1 id="titre">Cette page web s'autodétruira dans
            <span id="compteur">10</span> seconde(s)...</h1>

        <script src="27.js"> </script>
    </body>
</html>
```

FIGURE 33 – HTML : Ca va sauter !

5.2 Formulaires

5.2.1 Zones de texte

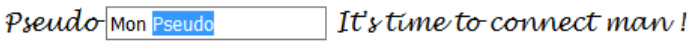

FIGURE 34 – Saisie du pseudo

Le code de la figure 35 présente la création d'un formulaire ne comprenant qu'un seul champ texte.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
    <head>
        <title> Formulaire </title>
        <meta charset="utf-8" />
    </head>
    <body>
        <form>
            <h1>Zones de texte</h1>
            <p>
                <label for="pseudo">Pseudo</label>
                <input type="text" name="pseudo" id="pseudo" required />
                <span id="aidePseudo"></span>
            </p>
        </form>
        <form>
            <h1>Cases à cocher</h1>
            <p>
                <label for="confirmation">Confirmation</label>
                <input type="checkbox" name="confirmation" id="confirmation"/>
            </p>
        </form>

        <script src="28.js"> </script>
    </body>
</html>
```

FIGURE 35 – Une zone de texte

L'entrée est de type `"text"`. D'autres types d'entrées sont possibles, par exemples :

- `"password"`;
- `"email"`;
- `"textarea"`.

En JavaScript on utilise la propriété `value` de l'élément du DOM correspondant pour modifier la valeur affichée dans la zone de texte (cf. figure 36).

```
var pseudoElt = document.getElementById("pseudo");
pseudoElt.value = "Mon Pseudo";

// Affichage d'un message au moment de la saisie du pseudo
pseudoElt.addEventListener("focus", function () {
    document.getElementById("aidePseudo").textContent = "It's time to connect man
                    !";
});
// Suppression du message precedent
pseudoElt.addEventListener("blur", function () {
    document.getElementById("aidePseudo").textContent = "";
});

// Affichage de l'etat de confirmation
document.getElementById("confirmation").addEventListener("change", function
                    (event) {
    console.log("Demande de confirmation : " + event.target.checked);
});
```

FIGURE 36 – Modification de la valeur de la zone de texte

Lorsque l'utilisateur clic sur la zone de texte pour y écrire, la zone de texte déclenche un événement de type **focus**. Lorsqu'il clique sur une autre zone de la page la zone de texte déclenche un événement de type**blur**. Ces événements sont exploitables dans le code JavaScript. Avec le code de la figure 37 on affiche au moment de la saisie *it's time to connect!* (cf. figure 34).

```
var pseudoElt = document.getElementById("pseudo");
pseudoElt.value = "Mon Pseudo";

// Affichage d'un message au moment de la saisie du pseudo
pseudoElt.addEventListener("focus", function () {
    document.getElementById("aidePseudo").textContent = "It's time to connect man
                    !";
});
// Suppression du message precedent
pseudoElt.addEventListener("blur", function () {
    document.getElementById("aidePseudo").textContent = "";
});

// Affichage de l'etat de confirmation
document.getElementById("confirmation").addEventListener("change", function
                    (event) {
    console.log("Demande de confirmation : " + event.target.checked);
});
```

FIGURE 37 – Modification de la valeur de la zone de texte

5.2.2 Cases à cocher

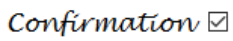

FIGURE 38 – Case à cocher

La figure 39 illustre la création d'un formulaire de type « case à cocher » en HTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                    "http://www.w3.org/TR/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
    <head>
        <title> Formulaire </title>
        <meta charset="utf-8" />
    </head>
    <body>
        <form>
            <h1>Zones de texte</h1>
            <p>
                <label for="pseudo">Pseudo</label>
                <input type="text" name="pseudo" id="pseudo" required />
                <span id="aidePseudo"></span>
            </p>
        </form>
        <form>
            <h1>Cases à cocher</h1>
            <p>
                <label for="confirmation">Confirmation</label>
                <input type="checkbox" name="confirmation" id="confirmation"/>
            </p>
        </form>

        <script src="28.js"> </script>
    </body>
</html>
```

FIGURE 39 – Une case à cocher

L'événement `change` se produit lorsque la propriété booléenne `checked` change de valeur (lorsque la case est cochée), c'est utilisé dans l'exemple de la figure 40 pour afficher dans la console le choix effectué par l'utilisateur (cf. figure 41).

```
var pseudoElt = document.getElementById("pseudo");
pseudoElt.value = "Mon Pseudo";

// Affichage d'un message au moment de la saisie du pseudo
pseudoElt.addEventListener("focus", function () {
    document.getElementById("aidePseudo").textContent = "It's time to connect man
                !";
});
// Suppression du message precedent
pseudoElt.addEventListener("blur", function () {
    document.getElementById("aidePseudo").textContent = "";
});

// Affichage de l'etat de confirmation
document.getElementById("confirmation").addEventListener("change", function
                (event) {
    console.log("Demande de confirmation : " + event.target.checked);
});
```

FIGURE 40 – Modification de la valeur de la zone de texte

```
Demande de confirmation : false
Demande de confirmation : true
Demande de confirmation : false
Demande de confirmation : true
Demande de confirmation : false
```

FIGURE 41 – Affichage des changements dans la console

5.2.3 Boutons radio

Un groupe de boutons radio permet à l'utilisateur de faire un seul choix parmi plusieurs possibilités.

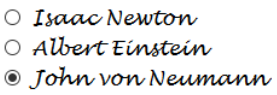

FIGURE 42 – Les boutons radio

Le code qui permet de créer des boutons radio est donné à la figure 43, les attributs `"name"` des boutons qui sont liés doivent avoir le même nom.

```
        <form>
            <p>
                Lequel de ces physiciens est à l'origine de l'élaboration d'un
                    modèle d'architecture sur lequel repose le fonctionnement
                    des processeurs actuels ?
                </br>
                <input type="radio" name="physiciens" id="phy1" value="Newton">
                <label for="Newton">Isaac Newton</label>
                </br>
                <input type="radio" name="physiciens" id="phy2" value="Einstein">
                <label for="Einstein">Albert Einstein</label>
                </br>
                <input type="radio" name="physiciens" id="phy3" value="von
                    Neumann" checked>
                <label for="von Neumann">John von Neumann</label>
                </br>
            </p>
        </form>
```

FIGURE 43 – Boutons radio

Afin de récupérer l'information « quel bouton est choisi » et l'afficher dans la console on peut utiliser le code de la figure 44. Le résultat obtenu est donné à la figure 45.

```
// Affichage du physicien choisi
var phyElts = document.getElementsByName("physiciens");
for (var i = 0; i < phyElts.length; i++) {
    phyElts[i].addEventListener("change", function (event) {
        console.log("Physicien choisi : " + event.target.value);
    });
}
```

FIGURE 44 – Boutons radio

```
Physicien choisi : Einstein
Physicien choisi : von Neumann
Physicien choisi : Newton
Physicien choisi : Einstein
Physicien choisi : von Neumann
```

FIGURE 45 – Affichage dans la console

5.2.4 Liste déroulante

Une liste déroulante permet également à l'utilisateur de faire un seul choix parmi plusieurs possibilités.

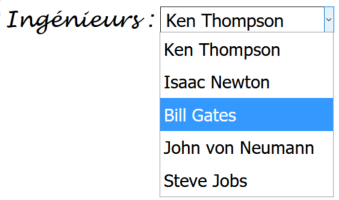

FIGURE 46 – Liste déroulante

Le code qui permet de créer une liste déroulante est donné à la figure 47.

```
<form>
    <p>
        Qui a mis au point le premier système UNIX ?
        <label for="Ingénieurs">Ingénieurs :</label>
        <select name="Ingénieurs" id="Ingé">
            <option value="KThomson" selected>Ken Thompson</option>
            <option value="INewton">Isaac Newton</option>
            <option value="BGates">Bill Gates</option>
            <option value="JvNeumann">John von Neumann</option>
            <option value="SJobs">Steve Jobs</option>
        </select>
    </p>
</form>
```

FIGURE 47 – Liste déroulante : code HTML

Afin de récupérer l'information « quel choix est fait dans la liste déroulante » et l'afficher dans la console on peut utiliser le code de la figure 48. Le résultat obtenu est donné à la figure 49.

```
// Choix effectué par le joueur
document.getElementById("Ingé").addEventListener("change", function (event) {
    console.log("Ingénieur choisi : " + event.target.value);
});
```

FIGURE 48 – Liste déroulante : code JavaScript

```
Ingénieur choisi : INewton
Ingénieur choisi : BGates
Ingénieur choisi : SJobs
Ingénieur choisi : KThomson
```

FIGURE 49 – Affichage dans la console

5.2.5 Soumettre le questionnaire

Dans l'exemple de la figure 50, une question est posée à l'utilisateur. Il peut y répondre en cochant la bonne case. Sa réponse n'est soumise que lorsqu'il appuie sur le bouton *Envoyer*, elle est alors affichée en dessous du questionnaire. S'il appuie sur le bouton *Annuler* le questionnaire est remis dans son état initial.

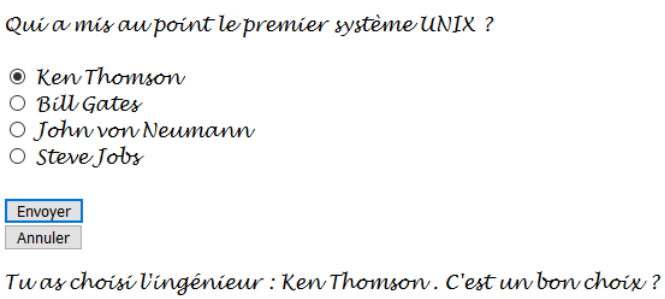

FIGURE 50 – Le formulaire

Le code HTML de la page est donné à la figure 51, le code JavaScript, commenté avec les explications nécessaires à sa compréhension est donné à la figure 52.

```
8           <form>
9               <p>
10                  Qui a mis au point le premier systeme UNIX ? <br/><br/>
11                  <input type="radio" name="response" id="KT" value="KT">
12                  <label for="KT">Ken Thomson</label> <br/>
13                  <input type="radio" name="response" id="BG" value="BG">
14                  <label for="BG">Bill Gates</label> <br/>
15                  <input type="radio" name="response" id="JN" value="JN" checked>
16                  <label for="JN">John von Neumann</label> <br/>
17                  <input type="radio" name="response" id="SJ" value="SJ">
18                  <label for="SJ">Steve Jobs</label> <br/>
19                  <br/>
20                  <input type="submit" value="Envoyer"> <br/>
21                  <input type="reset" value="Annuler">
22                  <br/>
23              </p>
24          </form>
25          <p>
26              Tu as choisi l'ingénieur : <span id="affichage"> physicien </span> .
                    C'est un bon choix ?
27          </p>
```

FIGURE 51 – Soumettre le formulaire : code HTML

```javascript
// La variable aAfficher stocke la valeur comprise
// entre les balises <span id="affichage"> et </span>
var aAfficher = document.getElementById("affichage");
// console.log(aAfficher.textContent);

// fonction qui change le texte afficher entre les balises
// <span> d'identifiant "affichage"
function changerAffichage(texteAAfficher){
    document.getElementById("affichage").innerHTML = texteAAfficher;
}

// form contient un lien vers le formulaire (noeud form du DOM)
var form = document.querySelector("form");

// lorque l'utilisateur clique sur le bouton "soumettre"
form.addEventListener("submit", function (event) {
    // choix stocke la valeur de l'attribut option de l'element
    // selectionnee dans le formulaire de name = "response"
    var choix = form.elements.response.value;
    // suivant le choix on change l'affichage sur la page web
    // thanks to the function changerAffichage()
    switch(choix){
        case "KT":
            changerAffichage("Ken Thomson");
            break;
        case "BG":
            changerAffichage("Bill Gates");
            break;
        case "JN":
            changerAffichage("John von Neumann");
            break;
        case "SJ":
            changerAffichage("Steve Jobs");
            break;
        default:
            changerAffichage("Non ! non ! non ! et non !");
    }
    // Annulation du changement d'etat des boutons radio
    event.preventDefault();
});
```

FIGURE 52 – Soumettre le formulaire : code JavaScript

Exercice n°5 page 19 Activité : Site Web et Questionnaire

Bibliographie

- cours de Anne Jeannin-Girardon - Introduction to web programming - UNISTRA ;
- cours de Yann Gaudeau - algorithme, programmation - UNISTRA
- https://openclassrooms.com/fr/courses/2984401-apprenez-a-coder-avec-javascript
- https://openclassrooms.com/fr/courses/3306901-creez-des-pages-web-interactives-avec-ja
  3545746-reagissez-a-des-evenements
- https://www.pierre-giraud.com/javascript/cours-complet/javascript-validation-formulair
  php

Exercices

Exercice n°1 :

Réaliser un programme qui affiche le résultat de la division euclidienne de deux valeurs saisies par l'utilisateur. Par exemple l'utilisateur saisit :

- dividende : 13 ;
- diviseur : 5.

Le programme affichera le résultat sous la forme : `13 = 2 x 5 + 3`.

Exercice n°2 :

Écrire un programme qui indique si un nombre stocké dans une variable est un nombre positif, négatif ou nul. S'il est positif ou négatif, le programme affichera si le nombre est pair ou impair.

Exercice n°3 :

1. Réaliser une page qui demande dans une boite de dialogue à l'utilisateur s'il veut :
   - 1 : calculer le volume d'un cube ?
   - 2 : calculer le volume d'un cylindre ?
2. Suivant le choix effectué, faire en sorte qu'une boite de dialogue s'ouvre et demande à l'utilisateur de saisir soit le coté du cube, soit le rayon de la section du cylindre et sa hauteur.
3. Toujours suivant le choix de l'utilisateur, afficher le résultat correspondant en rappelant les valeurs saisies soit pour le coté du cube, soit pour le rayon de la section et la hauteur du cylindre.
4. Gérer les saisies erronées lors du choix du volume à déterminer et la sortie du programme après le calcul si la saisie est correcte.

Exercice n°4 :

Reprendre le code de la figure 30 page 11 et le modifier (définir une fonction) pour afficher quel bouton de la souris est cliqué, l'objectif souhaité est illustré à la figure 53.

```
Bouton de la souris cliqué : gauche
Bouton de la souris cliqué : droit
Bouton de la souris cliqué : gauche
Bouton de la souris cliqué : droit
```

FIGURE 53 – Quel bouton de la souris est cliqué ?

Exercice n°5 :

Reprendre le code de la figure 51 et 52 pour :

- que le texte « C'est un bon choix ? » soit remplacer dans la page web par « Wrong... you desappointed me ! » ou « True... Congratulations ! » suivant que la réponse soumise soit vraie ou fausse ;
- qu'au lieu de proposer des boutons radio, l'utilisateur puisse choisir sa réponse dans une liste déroulante.

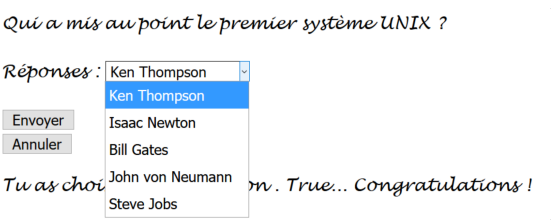

FIGURE 54 – Objectif de l'exercice n°5

Exercice n°6 :

Le cadre de l'exercice est la réalisation d'un RPG « Battle in the Poney Fringant » dans lequel s'affrontent deux personnages : Legolas qui a les oreilles pointues et Gimli qui est plutôt teigneux. Ils peuvent gagner des points de vie en buvant une potion de vie (du ch'ti coco-colo par exemple) et perdre des points de vie (lorsqu'ils sont percutés par une choppe de bière par exemple). Le diagramme de classe correspondant est donné à la figure 55.

1. Créer la classe *Personnage* et implémenter les méthodes de la classe.
2. Créer deux instances de la classe *Personnage*, par exemple :
   - perso1 : Gimli, nain, 150 points de vie, perd 23 points de vie à chaque attaque de l'elfe, et gagne 10 points de vie lorsqu'il boit une choppe de bière ;
   - perso2 : Legolas, elfe, 200 points de vie, perd 37 points de vie à chaque attaque du nain, et gagne 17 points de vie lorsqu'il boit un verre de ch'ti coco-colo de la Lothlorien.
3. Permettre aux deux joueurs de gagner ou perdre des points de vie à tour de rôle via une boite de dialogue. Chaque étape élémentaire du programme sera définie dans une fonction. La partie du code JavaScript d'exécution du jeu ne fera pas plus de quelques lignes.
4. A chaque tour de jeu les caractéristiques (nom, type de personnage et points de vie) des deux protagonistes s'affichent dans une boite de dialogue.
5. La partie se termine dès qu'un des deux joueurs n'a plus de points de vie. Le vainqueur s'affiche alors dans une dernière boite de dialogue.

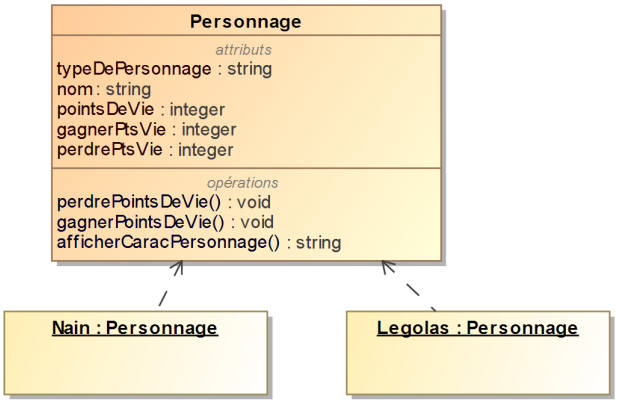

FIGURE 55 – Classe et instances

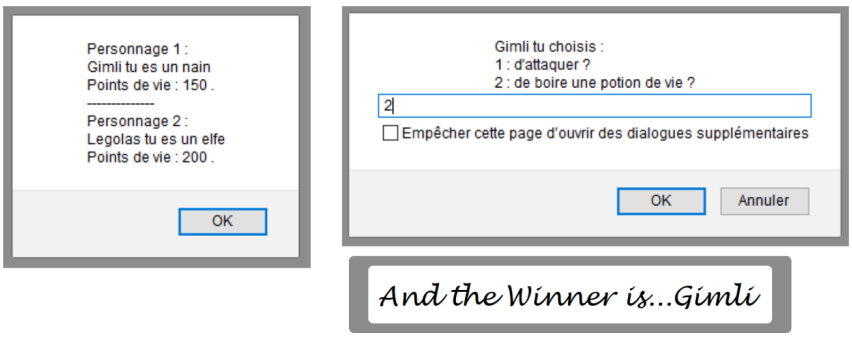

FIGURE 56 – Déroulement de la partie