Langage JavaScript

Issam REBAI – Équipe MOTEL – Département informatique – Campus de Brest
issam.rebai@imt-atlantique.fr

Version 0.3 [R.2023.01.19]

AU PROGRAMME

Histoire du langage

- Fiche signalétique
  - Acronyme : JS
  - Nom : JavaScript
  - Naissance : 04 décembre1995
  - Lieu : Netscape Communications Corporation (dissolu en 2003)
  - Fondateur : Brendan Eich
  - Forme : Norme ECMA (ECMA-262) – dernière version : 2021
  - Spécification : https://www.ecma-international.org/ecma-262/10.0/
  - Parents :
    - Syntaxe : Java, C
    - Principe : Scheme, Self et Perl
- Comment ?
  - En 10 jours en simplifiant la syntaxe Java
  - Développer une version client du *script* LiveScript (coté serveur)
- Pourquoi ?
  - Apporter du dynamisme, de l'interactivité dans le navigateur client tout en complétant le HTML
  - Ajouter de la logique métier par des algorithmes légers dans les pages Web

Les éditions ECMAScript

Version	Nom officiel	Description
ES1	ECMAScript 1 (1997)	1ère édition
ES2	ECMAScript 2 (1998)	Changement de rédaction
ES3	ECMAScript 3 (1998)	regex, try/catch, switch, do-while
ES4	ECMAScript 4	Rien de plus
ES5	ECMAScript 5 (2009)	"strict mode", support de JSON, String.trim(), Array.isArray(), méthodes itératives pour les tableaux, …
ES6	ECMAScript 6 (2015)	let, const, valeur par défaut pour les paramètres, Array.find(), Array.findIndex()
	ECMAScript 2016	**, Array.includes()
	ECMAScript 2017	string padding, Object.entries(), Object.values(), async functions, shared memory
	ECMAScript 2018	Paramètres rest et l'opérateur spread, itérations asynchrones, Promise.finally(), enrichissement de RegExp

Caractéristiques du langage

- Paradigmes supportés :
  - Fonctionnel typé dynamiquement
  - Procédural
  - Objet (prototype et non instances de classe)
- Les variables ne sont pas typées
- Les tableaux peuvent être associatifs ➔ dictionnaire
- Le « ; » n'est pas obligatoire pour indiquer la fin d'une instruction ➔ fin de ligne indique la fin d'instruction (être vigilant sur l'interprétation d'une instruction sur plusieurs lignes)
  - Prendre l'habitude de terminer chaque instruction par un « ; »

Exemples de fonctionnalités réalisées avec JS

- Afficher des pop-up
- Faire défiler un texte
- Insérer un menu dynamique (qui se développe au passage de la souris)
- Proposer un diaporama (changement d'image toute les x secondes, mettre en pause, aller à l'image précédente/ suivante, etc.)
- Afficher une horloge analogique
- Faire en sorte que des images suivent le pointeur de la souris
- Créer de petits jeux
- etc.
- **Modifier dynamiquement le contenu du document HTML**

Intégrer du code JS dans une page HTML

- Deux possibilités :
  - Code JS entre les balises `<script> </script>`
    - Traditionnellement dans la partie `head` de HTML
    - N'importe où dans le document HTML
  - Code JS dans des fichiers séparés et référencés dans les pages HTML
    - Traditionnellement dans la partie `head` de HTML
    - Syntaxe : `<script type="text/javascript" src="monscript.js"></script>`
      L'attribut type n'est pas nécessaire en HTML5
- Recommandé de placer le *script* en fin de document juste avant `</body>` ➔ ne pas bloquer le chargement de la page, et exécuter les *scripts* uniquement lorsque le document est prêt

Intégrer du code JS dans une page HTML ▸ Exemple « Hello world »

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello world</title>
</head>
<body>
<script>
 document.write("Hello world");
</script>
</body>
</html>
```

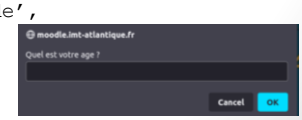
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Hello world</title>
</head>
<body>
 <h1>Title</h1>
 <script src="helloWorld.js"/></script>
</body>
</html>
```

Contenu du fichier `helloWorld.js` situé dans le même dossier

```
document.write("Hello world");
```

Les boîtes de dialogue

- Alerter l'utilisateur de quelque chose :
  - Fonction : `alert();`
  - Boutons : Ok (+ croix de fermeture)
  - Syntaxe :
    - `window.alert('message');`
    - `alert('message');`
- Demander à l'utilisateur de confirmer quelque chose :
  - Fonction : `confirm();`
  - Boutons : Ok et ANNULER (+ croix de fermeture)
  - Syntaxe :
    - `value=confirm('message de confirmation');`
- Demander à l'utilisateur de saisir une donnée :
  - Fonction : `prompt();`
  - Boutons : Ok et ANNULER (+ croix de fermeture)
  - Syntaxe :
    - `value = prompt('message de demande', valeurParDefaut_champSaisie);`



- Point sur la déclaration des variables
- Voir les mémentos disponibles sur l'espace d'enseignement
- Consulter le site https://developer.mozilla.org/fr/docs/Web/JavaScript

LA SYNTAXE JS

Déclaration de variables

- Historiquement, deux façons de déclarer une variable :
  - Explicitement (plus rigoureux) :
    - Exemple : `var maVariable="Hello world";`
    - Portée :
      - En dehors des fonctions : accessible de n'importe où dans le script (en dehors et à partir des fonctions) ➔
        - **variable globale**
        - ajoute une propriété à l'objet **global** (this.maVariable)
      - Dans une fonction : limité à la fonction elle-même ➔ **variable locale**
  - Implicitement :
    - Exemple : `maVariable="Hello world";`
    - Portée : globale quelque soit l'endroit de déclaration

Déclaration de variables

```
<script>
var a = 12;
var b = 4;
function MultipliePar2(b) {
   var a = b * 2;
  return a;
}
console.log("Le double de ",b," est ",MultipliePar2(b));
console.log("La valeur de a est ",a);
</script>
```

Déclaration de variables

```
<script>
var a = 12;
var b = 4;
function MultipliePar2(b) {
   a = b * 2;
  return a;
}
console.log("Le double de ",b," est ",MultipliePar2(b));
console.log("La valeur de a est ",a);
</script>
```


Déclaration de variables

- La déclaration avec `let` restreint la portée de la variable au bloc dans lequel elle est déclarée
  - Dans un bloc de répétition ou de condition ➔ la variable n'existe pas en dehors de ce bloc.
  - Dans une fonction ⇔ var
  - En dehors des fonctions ➔ **variable globale sans son ajout en tant que propriété à l'objet global** (~~this.variable~~)

Remarque : une variable de même nom déclarée dans un sous contexte n'écrase pas la variable du contexte supérieur mais aura la priorité sur cette dernière.

- Exemple:
  - `let maVariable="Hello";`

Déclaration de variables

```
1.  function test() {
2.   let variable_1="v_1";
3.   if(variable_1=="v_1"){
4.      let variable_2="v_2";
5.      alert(variable_2);
6.    }
7.   alert(variable_1);
8.   alert(variable_2);
9.  }
```

```
1.  let variable_1="v_1";
2.   if(variable_1=="v_1"){
3.      let variable_1="v_2";
4.      alert(variable_1);
5.    }
6.   alert(variable_1);
```


Déclaration de variables

- La déclaration d'une variable avec `const` la rend accessible uniquement en lecture. Il s'agit d'une variable constante :
  - Elle ne peut pas être déclarée à nouveau
  - Les règles de sa portée se comportent comme un let
  - Son identifiant ne peut pas être réaffecté
    - si le contenu de la constante est un objet ou un tableau, l'objet ou le tableau lui-même pourra toujours être modifié.
    - Si le contenu est une chaîne de caractères ou une valeur, il ne peut pas être modifié
- Exemple:
  - `const MA_CONSATNTE_PI=3.141592;`
  - ```
    const tableau=["un","deux","trois"];
    tableau.push("quatre");
    console.log(tableau);
    ```
    ```

HTML et DOM

- Acronyme : DOM
- Nom : Document Object Model
- Traduction : modèle d'objets de document
- Signification : représentation de la page Web affichée par le navigateur sous forme d'un objet comportant des propriétés et des méthodes
- Caractéristiques :
  - Interface de programmation normalisée par le W3C et implémentée dans différents langages
  - Représentation d'un document HTML sous forme d'un arbre ayant des nœuds et des feuilles
- Rôle : permet le parcours et la modification du contenu du document HTML à partir d'un *script* de programmation

Accéder à un document HTML à partir de JS

- Le document HTML est accessible via l'objet DOM nommé **document** en JS
- Les méthodes d'accès aux éléments de l'objet document :
  - getElementById('identity') : cibler un élément ayant un attribut `id` ayant la valeur `identity`
  - getElementsByTagName('tagName') : cibler tous les éléments du document HTML nommé `tagName`
    Le résultat est un tableau d'éléments
  - getElementsByClassName('className') : cibler tous les éléments ayant un attribut `class` avec la valeur `className`
    Le résultat est un tableau d'éléments
  - querySelector('selector') : cibler le 1er élément trouvé correspondant à un certain sélecteur CSS (`id`, `class`, attribut, …)
  - querySelectorAll('selector') : idem que `querySelector` mais retourne un tableau contenant tous les éléments trouvés

Accéder à un document HTML à partir de JS

- Méthodes d'accès au contenu des éléments de l'objet document :
  - `innerHTML` : récupérer tout le contenu d'un élément HTML et pas seulement le texte contenu dans celui-ci (les descendants qui y sont aussi)
  - `textContent` : récupérer le contenu textuel présent à l'intérieur d'un élément et de ses descendants
- Méthodes d'accès à des types d'éléments
  - `title` : titre de la page
  - `body` : l'élément `body` du document HTML
  - `links` : liste des éléments `area` et `a` possédant un attribut `href`

Accéder à un document HTML à partir de JS ▸ Exemples

```
<!DOCTYPE html>
<html>
    <head>
        <title>Le DOM HTML</title>
        <meta charset="utf-8">
    </head>
    <body>
        <h1 id="gros_titre">Le DOM</h1>
        <p>Du texte et <a href="http://w3.org">un
 lien</a></p>
        <script>
            var titre =
 document.getElementById('gros_titre').innerHTML;
            alert(titre);
        </script>
    </body>
</html>
```

Le concept d'événement

- Capturer des actions de l'utilisateur sur des éléments de la page Web et associer des traitements ➔ la page réagit aux comportement de l'utilisateur
- Principe
  - Créer une fonction contenant le traitement à réaliser lors de l'action de l'utilisateur
  - Déterminer l'élément (les éléments) réactif(s) à l'action
  - Déterminer le type d'événement auquel va réagir l'élément
  - Ajouter à la balise de l'élément (des éléments) l'attribut correspondant à l'événement choisi et invoquer la méthode en question
- Exemple
  - `<p onclick="alert('Bravo !');">Cliquez-moi, cliquez-moi !</p>`

Extrait de la liste des événements

onAbort : le chargement de la page a été interrompu, soit par l'appui sur 'Cancel' ou par le clique sur un lien.
onError : lors de la survenu d'une erreur
onResize : la page a été redimensionnée
onScroll : lors du scrolle dans la page
onLoad : l'objet (une image par exemple) est complètement chargée
onUnload : l'explorateur quitte la page
onBlur : un champ de formulaire perds le focus
onChange : un champ du formulaire est édité et quitté
onFocus : un champ de formulaire prend le focus
onKeyDown : une touche est pressée (durant l'édition d'un champ de formulaire)
onKeyPress : une touche est pressée (durant l'édition d'un champ de formulaire)
onKeyUp : une touche est relâchée (durant l'édition d'un champ de formulaire)
onSelect : une partie du contenu du champ est sélectionnée
onClick : sur un clic
onDblClick : sur un double click
onMouseOut : lorsque le pointeur de la souris sort de l'objet
onMouseOver : lorsque le pointeur de souris passe sur l'objet

ACTIVITÉ PRATIQUE

Étudier la syntaxe JS et réaliser les pages Web suivantes :

1. Calcul le factoriel d'un nombre introduit par l'utilisateur
2. Gestion (ajouter, lister) d'un annuaire de personnes ayant un nom, prénom, un téléphone et une adresse mail
3. Ajouter, à la solution de la question 2, les fonctions de modification et de suppression de personnes.