Développement Web: Introduction à JavaScript

Abdelraouf Hecham - IUT de Montpellier - AS

hecham@lirmm.fr (www.lirmm.fr/~hecham)

8 Janvier 2018

Outline

Pourquoi Javascript pour le Front-end?

HTML/CSS sont fait pour représenter des information statiques.

Ils ne permettent pas de:

- Rendre la page interactive.
- Vérifier un formulaire en temps réel coté client.
- Afficher/cacher du contenu, zoomer sur des images, etc.
- Récupérer des données serveur sans rafraichir la page (faciliter la recherche, mettre a jour la page sans intervention du client, etc.).
- etc.

Javascript est un langage puissant mais dangereux

Un langage puissant:

- Un langage de script performant.
- Un langage flexible et multi-platforme.
- Un langage Oriente-Objet (héritage).

Un langage dangereux (prône aux mauvaises pratiques):

- Un langage à typage dynamique (les variables non pas de type fixe).
- Un langage interprété et non compilé. (Les erreurs ne serons pas détectées automatiquement avant l'exécution!).
- Un langage Orienté-Objet à prototype (pas de classes, que des objets).

Ou mettre le code Javascript?

Trois options:

1. Mettre le code Javascript directement dans un fichier HTML (.html) dans une balise “script”: <script type="text/javascript"> /* code Javascript */ </script>.
2. Mettre le code Javascript dans un fichier a part (.js) et le referencer dans un fichier HTML: <script type="text/javascript" src="monFichier.js"></script>.
3. Coder directement en utilisant la console du navigateur web (seulement pour faire des testes et pour apprentissage). (Chrome, Firefox: Ctrl + shift + i)

Bases du Langage

- JavaScript est sensible à la casse (nomvariable $\neq$ nomVariable).
- Les instructions sont séparées par des points-virgules*. (Recommandé. Faire très attentions aux sauts-à-la-ligne.)
- Les commentaires sont introduits par: // une seule ligne et /* plusieurs lignes */

Outils de Débogage

Débogage : Suivre l'exécution du code pour détecter des erreurs.

1. **console.log(message)** : permet d'ecrire un message ou la valeur d'une variable dans la console. **(recommandé)**

```
console.log("Breaking Bad est meilleur que Game of Throne");
// Breaking Bad est meilleur que Game of Throne
console.log("L'exécution du code continue");
```

2. **alert(message)** : permet de créer une boite de dialogue qui affiche le message et de pauser l'exécution du code Javascript jusqu'à ce que l'utilisateur clique sur 'Ok'.

```
alert("Breaking Bad est meilleur que Game of Throne");
// boite de dialogue "Breaking Bad est meilleur que Game of Throne"
console.log("L'exécution du code se met en pause, ceci n'est affiché qu'après que l'utilisateur clique sur OK.");
```

3. Mot-clé **debug** : permet de faire une pause dans l'exécution du code. **(recommandé)**

```
console.log("Breaking Bad est meilleur que Game of Throne");
// Breaking Bad est meilleur que Game of Throne
debug;
console.log("L'exécution du code se met en pause");
```

Variables et Constantes

- Pour declarer une variable en utilise le mot-clé var (ou let) e.g.

```
var nom = "snow"; (ou let nom = "snow")
var prenom;
prenom = "john";
var age = 20;
age = "Javascript s'en fout des types!";
```

- Le nom d'une variable (constante/fonction/objet) doit débuter avec une lettre, un tiret bas, ou un symbole dollar et peut contenir des caractères numériques, alphabétiques et des tirets bas voire des caractères Unicode).

- Etat d'une variable:
  1. Erreur: ReferenceError si elle n'a jamais été déclarée avant.
  2. undefined si elle a été déclarée mais aucune valeur ne lui a été affectée.
  3. null ou la valeur affectée si elle a été déclarée et initialisée à null ou à une valeur.

- Une constante ne peut pas changer de valeur, on utilise const pour la déclarer:

```
const PI = 3.14;
```

Exercice: (cours 1 ex 01)

Types des Variables

La dernière version du standard ECMAScript définit sept types de données :

1. Type booléen : true et false.
2. Type null, un mot-clé spécial pour indiquer une valeur nulle (au sens informatique).
3. Un type pour les valeurs indéfinies (undefined).
4. Un type pour les nombres. 42 ou 3.14159. (entiers et réels).
5. Un type pour les chaînes de caractères. "Coucou"
6. Un type pour les symboles, apparus avec ECMAScript 2015 (ES6). Ce type est utilisé pour représenter des données immuables et uniques.
7. un type non primitif pour les objets (Object)

Conversion de types de données

- JavaScript est un langage à **typage dynamique**. Les types de données sont **convertis automatiquement** durant l'exécution du script.

```
var reponse = "La réponse est: " + 42; // "La réponse est: 42"
reponse = 42 + " est la réponse."; // "42 est la réponse."

// Attention!
reponse = "40" + 2; // "402"
reponse = "40" - 2; // 38
```

- **Bonne pratique:** Parser (faire la conversion) explicitement les nombres qui sont sous forme de chaines de caractères:

```
// Parser des nombres entiers:
reponse = parseInt("40") + 2; // 42

// Parser des nombres décimaux:
reponse = parseFloat("40.0") + 2.0; // 42.0
```

- **Exercice: (cours 1 ex 02)** Déclarez la variable 'somme' qui contient le calcul de la somme des valeurs suivantes: 13, "55.4", "104", 37.6, "0"

Chaines de Caractères

- Encadrée par des doubles-quotes (") ou des simples-quotes ('): e.g. "truc", 'truc', "l'apostrophe".
- Caractère d'échappement (backslash) \ : e.g.
  "Il est fan de \"Westworld\"" // Il est fan de "Westworld"
- Retour a la ligne \n : "une ligne. \n une autre ligne."

```
/* une ligne.
une autre ligne.*/
```

- Écrire une chaine de caractère sur plusieurs lignes de code:
  1. Solution simple:

```
var str = "cette chaîne "
 + "est cassée sur "
 + "plusieurs lignes."; // cette chaîne est cassée sur plusieurs lignes.
```

  2. Utiliser le backslash \ :

```
var str = "cette chaîne \
est cassée sur \
plusieurs lignes."; // cette chaîne est cassée sur plusieurs lignes.
```

  3. Utiliser l'accent grave ` (pas le simple-quote):

```
var str = `cette chaîne
est cassée sur
plusieurs lignes.`; // cette chaîne est cassée sur plusieurs lignes.
```

Tableau de littéraux

- Un tableau est définie par des crochets [ ], e.g:

```
var starks = ["Eddard", "Robb", "Sansa", "Arya"];
starks[0]; // Eddard
starks[1]; // "Robb"
```

Exercice: Sur la console, déclarez le tableau suivant: maListe = ["Potato", , 42, true];
Quelle est la valeur du deuxième élément de maListe?

- Déclaration d'un tableau vide:

```
var starks = []; // tableau vide dont le nombre d'éléments est inconnu
starks[0] = "Eddard"; // affecter une valeur à un élément
```

- La propriété length d'un tableau renvoie son nombre d'éléments.

```
starks.length; // 1
```

Les Objets

- Objets en Javascript sont des tuples (attribut : valeur), e.g.
  var jonSnow = { nom: "Snow", prenom: "Jon", age: 20, surnoms: ["White Wolf", "Savior of the Free Folk", "Commander of the Night's Watch"] }

```
jonSnow["nom"]; // now
jonSnow.nom; // Snow
jonSnow.surnoms[2]; // Commander of the Night's Watch
jonSnow["surnoms"][2]; // Commander of the Night's Watch
```

- Les objets peuvent être imbriqués:
  var robb = { nom: "Stark", prenom: "Robb", pere: {nom: "Stark", prenom: "Eddard"}} }

```
robb["pere"]["prenom"]; // Eddard
robb.pere.prenom; // Eddard
```

Fonctions

- Les fonctions permettent de répéter de mêmes portions de codes (traitement, calcul, etc.). Déclaration:
  function nomFonction(liste éventuelle des arguments){ /* Instructions */ }

- Exemple:

```
function hello(){ alert("Hello!"); }
hello(); /* Boite de dialogue "Hello!" */
```

Exercice: (Cours 1 ex 03) Exécutez le code suivant, que fait-il?

```
function afficher(message){ alert(message); }
afficher("Vous devez cliquer 3 fois sur Ok");
afficher("Vous devez cliquer 2 fois sur Ok");
afficher("Vous devez cliquer 1 fois sur Ok");
afficher("Vous devez cliquer 10 fois sur Ok");
afficher("Hahaha non c'est bon :)");
```

Fonctions

- Une fonction peut **retourner une valeur** :

```
function carre(nbr){ return nbr*nbr; }
var x = 2;
var y = carre(x);
console.log(x + ", " + y); // 2, 4
```

- Une fonction peut aussi être **référencée par une variable** :

```
var carre = function(nbr){ return nbr*nbr; }
```

- la déclaration d'une fonction **peut être faite après l'appel** (elle doit appartenir à la **portée** dans laquelle elle est appelée)

```
console.log(cube(2)); // 8
function cube(nbr){ return carre(nbr)*nbr; }
```

Protées des Variables

- La portée d'une variable (ou fonction) désigne l'ensemble du code dans lequel elle peut être utilisée .
- Lorsqu'une variable est déclarée avec var en dehors des fonctions, elle est appelée variable globale car elle est disponible pour tout le code contenu dans le document.
- Lorsqu'une variable est déclarée dans une fonction, elle est appelée variable locale car elle n'est disponible qu'au sein de cette fonction.

```
var num1 = 10, num2 = 20; // num1 et num2 sont des variables globales
function multiplier(){ return num1*num2; }
multiplier(); // 200

function additionner() {
    var num3 = 30; // num3 est une variable locale
    return num1+num3;
}
additionner(); // 40
console.log(num3) // undefined
```

Boutons et Javascript

- On peut mettre du code Javascript dans les attributs événements des balise: e.g. onclick de la balise button :

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

- On peut aussi appeler des fonctions:

```
<button onclick="afficher('Bonjour!');">Bonjour</button>
```

Fonctions et Tableaux

Exercice: (Cours 1 ex 04)

- Créez un tableau nommé ‘tab’ dont le premier élément est -2, le deuxième 1 et le troisième 4.
- Créez une fonction ‘additionne’ prenant un argument x et renvoyant le résultat de l’addition de x à 2.
- Créez une fonction ‘affiche’, appelée au clic sur le bouton, qui affiche dans des boîtes d’alerte successivement le résultat de additionne appliqué au premier élément, puis au dernier élément du tableau (en utilisant la propriété length).

Testes et Contrôle du Flux des Instructions

- L'instruction conditionnelle **if...else**

```
if (condition1) {
    instruction1;
}else if (condition2) {
    instruction2;
} else {
    dernièreInstruction;
}
```

- L'instruction conditionnelle **switch**

```
switch (expression) {
    case label1:
        instructions1;
        [break;]
    case label2:
        instructions2;
        [break;]
        ...
    default:
        instructionsParDefaut;
}
```

Opérateurs de Comparaison

- `==` vrai si les opérandes sont égaux (après conversion) ("3" == 3 est vrai).
- `!=` vrai si les opérandes sont différents (après conversion) ("3" != 3 est faux).
- `===` vrai si les opérandes sont égaux et de même type ("3" === 3 est faux).
- `!==` vrai si les opérandes sont différents ou pas du même type ("3" !== 3 est vrai).
- `<` (inférieur), `>` (supérieur), `<=` (inférieur ou égale), `>=` (supérieur ou égale).

If...else

Exercice: (Cours 1 ex 05)

- Créez un tableau nommé ‘tab’ dont le premier élément est -2, le deuxième 1 et le troisième 4.
- Créez une fonction ‘soustrait’ prenant un argument x et renvoyant le résultat de la soustraction de x-2 si x est positif ou nul, la chaîne de caractères "Nombre négatif!" sinon.
- Créez une fonction ‘affiche’, appelée au clic sur le bouton, qui affiche dans des boîtes d’alerte successivement le résultat de ‘soustrait’ appliqué au premier élément, puis au dernier élément du tableau (en utilisant la propriété length).

Switch

Exercice: (Cours 1 ex 06)

- Pour avoir le jour (Samedi, etc.) d'aujourd'hui on utilise la fonction `getDay()` d'un objet Date qui retourne un entier entre 0 et 6:

```
var aujourdui = new Date();
var jour = aujourdui.getDay(); Dimanche (0), Lundi (1), ..., Samedi(6)
```

- Au click sur un bouton, lancer une fonction 'jourDeLaSemaine()'. Cette fonction détermine le jour de la semaine et affiche selon le cas dimanche, lundi, mardi... etc. jusqu'à samedi. (utiliser un switch)

Boucles

- ```
  for(var i = 0; i < N; i++) {
      instructions;
  }
  ```

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

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

Boucles

Exercice 1: (Cours 1 ex 07)

- Au clique sur le bouton, lancez la fonction boucle() qui crée un tableau de 3 éléments et utilise une boucle 'for' pour le remplir de sorte que l'élément i du tableau contienne i^2. Affichez le tableau dans une boîte d'alerte.

Exercice 2: (Cours 1 ex 08)

- On veut créer une page de spam.
- Au click sur le bouton "Télécharger le Film", 30 boites d'alerte s'affichent successivement, chaque boite d'alerte contient le nombre de boites qui reste à cliquer.