MÉMO2 - DÉVELOPPEMENT SEM2

Jeudi 28 Janvier / Vendredi 29 Janvier 2016 - Cours Javascript

Frédéric LOSSIGNOL

Rappel

Nous avons vu en semaine 1 :

- les variables et les constantes
- les tableaux
- la concaténation
- Les types de variable (string / number) et les conversions (parseInt, parseFloat)
- la class Date()

La semaine 2 consiste à comprendre 2 notions fondamentales que nous retrouverons dans tous les langages de programmation :

- Les ***Conditions***
- Les ***Boucles***

Nous aborderons également à la fin une autre notion très utile dans vos futurs codes : les ***Fonctions***.

A retenir :

Les conditions *vous permettent d'exécuter une ou plusieurs instruction(s),* ***SI*** *une condition que vous avez déterminée est remplie*.

Nous reverrons ci-dessous la syntaxe d'une condition.

Les Boucles vous permettent de répéter du code X fois, ou tant qu'une condition particulière est remplie. Il existe 3 types de boucles, chacune étant adaptée à des cas particuliers : la boucle **for**, la boucle **while**, la boucle **do while**.

Nous verrons plus bas pourquoi et comment les utiliser.

Note : Tous les bouts de code que vous trouverez dans ce document sont directement exécutables et testables depuis la console de votre navigateur (Firebug ou la console sur Chrome), ou même sur https://jsfiddle.net

SOMMAIRE

1 Les conditions

1.1 Pourquoi utiliser les conditions ?

Pour exécuter du code ***SI*** une condition déterminée est remplie.

Exemple 1 : Nous voulons tester si un utilisateur est majeur ou non.
Exemple 2 : Nous souhaitons réaliser un outil qui renvoie à l'utilisateur le résultat du mélange de 2 couleurs primaires.

1.2 La syntaxe d'une condition

Les mots-clés utilisés pour construire une condition sont ***if***, ***else if***, et ***else*** (respectivement si, sinon si, sinon). L'écriture d'une condition se fait comme ceci :

```
if(conditionAVerifier) {
// Code à exécuter si la condition est vérifiée
}
else {
// Code à exécuter si la condition n'est pas vérifiée
}
```

Exemple 1 : Tester l'âge du visiteur

```
// On crée une variable age qui stocke la saisie de l'utilisateur
var age = parseInt(window.prompt('Quel âge avez-vous ?'));

// si l'utilisateur a 18ans ou plus  alors on lui affiche un message "Vous êtes majeur"
if(age >= 18) {
  alert('Vous êtes majeur');
}
// sinon  on lui affiche un message "Vous êtes mineur"
else {
  alert('Vous êtes mineur');
}
// pour exemple, ci-dessous le même code conditionnel avec la syntaxe ternaire
// (age>=18) ? alert('Vous êtes majeur') : alert('vous êtes mineur');
```

Note : exécutez ce code directement depuis la console firebug ou Chrome ou sur jsfiddle : https://jsfiddle.net/learnlab/7c6p0obt

Dans le cas où vous avez plusieurs conditions à tester, alors utilisez un ou plusieurs ***else if*** à la suite de votre ***if***.

Exemple 2 : un script qui renvoie le résultat du mélange de 2 couleurs primaires.

```
/*
Utilisation de ELSE IF et de son alternative SWITCH en cas de conditions multiples.
*/
// Nous commençons par déclarer 2 variables d'entrée (color1, color2)
// et 1 tableau result contenant les résultats possibles
var color1;
var color2;
var result = ['violet', 'orange', 'vert'];

color1 = window.prompt('Choisissez une couleur primaire (rouge, bleu, ou jaune)');
color2 = window.prompt('Choisissez une couleur de mélange (rouge, bleu, ou jaune)');

// si on mélange les couleurs bleu et rouge
if((color1=='rouge' && color2=='bleu') || (color1=='bleu' && color2=='rouge')) {
    alert('En mélangeant du rouge et du bleu, vous obtenez du ' + result[0]);
}
// sinon si on mélange les couleurs rouge et jaune
else if((color1=='rouge' && color2=='jaune') || (color1=='jaune' && color2=='rouge')) {
   alert('En mélangeant du rouge et du jaune, vous obtenez du ' + result[1]);
}
// sinon si on mélange les couleur bleu et jaune
else if((color1=='bleu' && color2=='jaune') || (color1=='jaune' && color2=='bleu')) {
   alert('En mélangeant du jaune et du bleu, vous obtenez du ' + result[2]);
}
// sinon
else {
   alert('vous n\'avez pas renseigné une couleur acceptée');
}
```

Note : exécutez ce code directement depuis la console Firebug ou Chrome ou sur jsfiddle : https://jsfiddle.net/learnlab/3t8ha39e

Dans le code précédent, notez que les opérateurs logiques *ET* et *OU* (***&&*** et **||**) sont très utiles pour éviter de répéter des ***if*** et des ***else if***.

1.3 La syntaxe du switch (remplace les else if)

Les parenthèses qui suivent le mot clé ***switch*** indiquent une expression dont la valeur est testée successivement par chacun des *case*. Lorsque l'expression testée est égale à une des valeurs suivant un *case*, la liste d'instruction qui suit (après les ***:***), celui-ci est exécuté.

Le mot clé ***break*** indique la sortie de la structure conditionnelle.

Le mot clé ***default*** précède la liste d'instructions qui sera exécutée si l'expression n'est jamais égale à une des valeurs.

```
switch (variableATester) {
case 1: // signifie : dans le cas où variableATester == 1
// Code à exécuter si la condition est vérifiée;
break; // le break permet de sortir de du switch

case 2: // signifie : dans le cas où variableATester == 2
// Code à exécuter si la condition est vérifiée;
break;

case 3: // signifie : dans le cas où variableATester == 3
// Code à exécuter si la condition est vérifiée;
break;

default: // (facultatif) signifie : dans tous les autres cas cas
// Code à exécuter si la condition est vérifiée;
break;
}
```

Note : Retrouvez le code précédent (script des couleurs primaires) construit avec un switch et des if imbriqués sur jsfiddle : https://jsfiddle.net/learnlab/nr0fnxsc

2 Les boucles

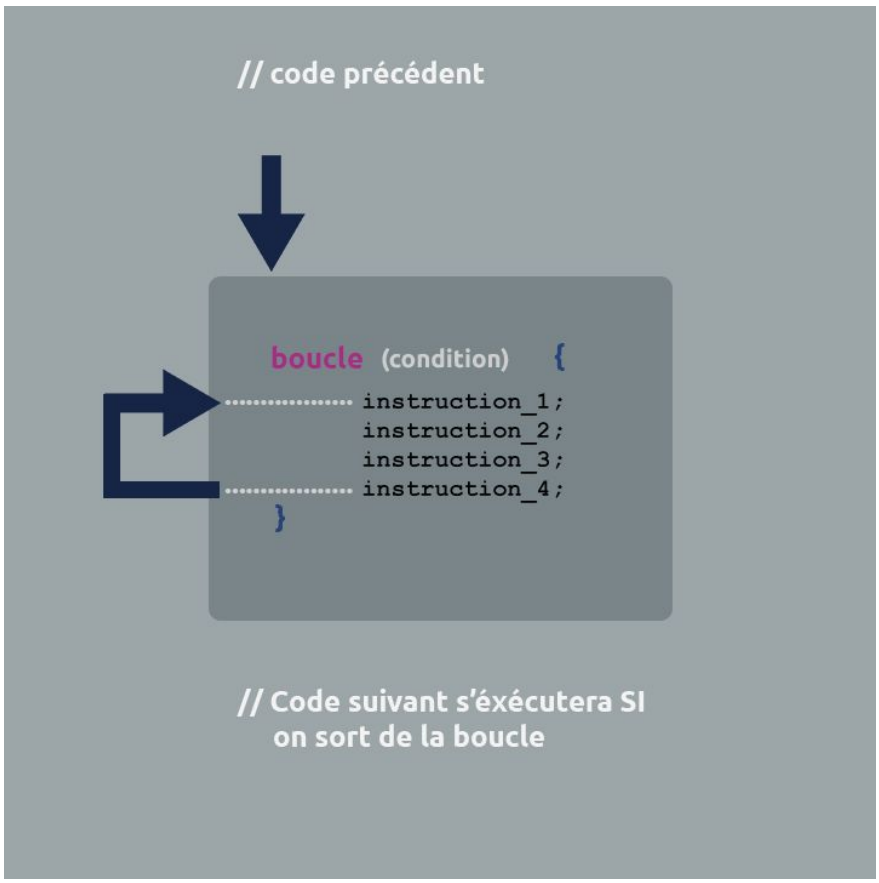
Les Boucles vous permettent de répéter du code X fois, ou tant qu'une condition particulière est remplie. Il existe 3 types de boucles, chacune étant adaptée à des cas particuliers : la boucle ***for***, la boucle ***while***, la boucle ***do while***.

2.1 D'accord, mais pourquoi utiliser les boucles ?

Il existe bien des cas pour lesquels les boucles sont indispensables, en voici 2 exemples :

- Parcourir un tableau de données et les afficher de façon formatée sur une page web (par exemple afficher une liste d'articles de blog, ou une liste de produits sur un e-commerce).
- Parcourir un tableau de données pour rechercher un terme ou un mot (par exemple, pour rechercher un mot sur une "application dictionnaire").

Schéma de fonctionnement d'une boucle



On lit le code de haut en bas. Le code à l'intérieur de la boucle se répétera tant que la condition est remplie. Si la condition n'est plus remplie, alors on sort de la boucle.

2.2 La boucle *for*

La boucle ***for*** est particulièrement adaptée pour les boucles où l'on sait à l'avance combien de fois nous devrons répéter les instructions dans la boucle. Son fonctionnement prend en compte une initialisation, une condition de fin et un pas d'incrémentation.

La syntaxe de la boucle *for* :

```
for (initialisation; conditionDeFin; incrémentation) {
// Code à éxécuter
}
```

Ouvrez la console Firebug ou Chrome (F12>onglet Console), copiez et exécutez ce code :

```
// Écrire100 fois "je kiff le javascript" et afficher les numéros de lignes /

for(var i = 1;  i <= 100;  i++){
  console.log(i + ' je kiff le javascript');
}

// L'initialisation : i vaut 1 (je commence à 1)
// La condition : le code s'exécutera tant que la valeur de i ne sera pas égale à 100
// L'incrémentation : ici i++ signifie i=i+1 (on incrémente de 1 la valeur de i à chaque tour de boucle)
```

Le code est bien répété 100 fois en démarrant de la valeur 1 et en incrémentant de 1.

Un autre exemple à retenir : parcourir un tableau indexé

Déclarons une variable de type tableau et assignons lui des valeurs.

```
var blogCategories = ['javascript', 'php', 'mySql','html', 'css'];
// On souhaite afficher les données contenues dans ce tableau, en lettres majuscules.
for(var i=0;  i<5;  i++) {
 console.log(blogCategories[i].toUpperCase());
 }
/*
Affichera :
JAVASCRIPT
PHP
MYSQL
HTML
CSS
*/
```

Note : La propriété ***.length***, qui permet d'obtenir la longueur d'une chaîne de caractère, retournera le nombre d'entrées d'un tableau lorsqu'elle est appliquée à un tableau.

```
var blogCategories = ['javascript', 'php', 'mySql','html', 'css'];
var nbIndex = blogCategories.length;

console.log(blogCategories.length);
// affichera 5
```

> Ce qui vous permet de déterminer dynamiquement la condition de fin de votre boucle :

```
var blogCategories = ['javascript', 'php', 'mySql','html', 'css'];
var nombreIndex = blogCategories.length ;
// nombreIndex vaut 5 (puisqu'il y a 5 entrées dans le tabelau)

for(var i=0;  i<nombreIndex;  i++) {
console.log(blogCategories[i].toUpperCase());
// On se sert ici de la méthode .toUpperCase() pour transformer les données en majuscules.
 }
/*
Affichera dans la console :
JAVASCRIPT
PHP
MYSQL
HTML
CSS
*/
```

A retenir :

La boucle *for* répète des instructions ***jusqu'à ce qu'une condition donnée ne soit plus vérifiée***, en incluant

- une instruction d'initialisation (ici var i=0),
- un condition de sortie de la boucle (ici i<100)
- et une instruction d'incrémentation. (ici i++)

2.3 Les boucles ***while*** et ***do while***

La boucle *while* répète une ou plusieurs instructions ***tant qu'une condition donnée est vérifiée***.

Syntaxe de ***while*** et de ***do while***

while (conditionAVerifier) { *// Code à éxécuter* *}*	*do {* *// Code à éxécuter* *}* *while (conditionAVerifier)*
Tant que la condition est vérifiée, le code entre accolades est éxécuté (sinon on sort de la boucle).	*Le code est exécuté au moins une fois quoiqu'il arrive, puis tant que la condition est vérifiée.*

Exemple : Demander à une couleur déterminée à l'utilisateur

Ouvrez la console Firebug ou Chrome (F12>onglet Console), copiez et exécutez ce code :

```
// Tant que l'utilisateur ne rentre pas la couleur rouge OU 'bleu OU jaune, on lui redemande son choix
var color;
var color = window.prompt('Veuillez choisir une couleur primaire (rouge, bleu, jaune)');

while(! ((color == 'bleu') || (color == 'rouge') || (color == 'jaune')) ) {
color=window.prompt('Veuillez choisir une couleur primaire (rouge, bleu, jaune)');
}
//  pour vérifier la condition, on utilise ici les opérateur !(différent de), || (ou) et == (opérateur d'égalité)

alert('vous avez choisi la couleur : ' + color);
//  Si l'utilisateur a bien renseignée la couleur rouge OU bleu OU jaune, on lui affiche un message
```

*Ce code fonctionne mais nous voyons immédiatement qu'une boucle **do while** est bien plus appropriée dans ce cas . C'est ce que nous allons faire tout de suite.*

La boucle *do while* est similaire à la boucle ***while*****,** si ce n'est que le code à exécuter s'exécutera d'abord une fois avant que la boucle teste la condition donnée.

Ouvrez la console Firebug ou Chrome (F12>onglet Console), copiez et exécutez ce code :

```
// Tant que l'utilisateur ne rentre pas la couleur rouge OU 'bleu OU jaune, on lui redemande son choix
var color;

do {
color=window.prompt('Veuillez choisir une couleur primaire (rouge, bleu, jaune)');
}
while(! ((color == 'bleu') || (color == 'rouge') || (color == 'jaune')) ) ;

/*
le widow.prompt() est exécuté une 1ère fois, et seulement ensuite la boucle vérifie si la condition est remplie.
*/
```

*La boucle **do while** est plus logique dans ce cas précis, et nous permet d'économiser une répétition du **window.prompt()** par rapport au code précédent, bien que celui-ci fonctionne également.*

A retenir :

Les Boucles vous permettent de répéter du code plusieurs fois, ou tant qu'une condition particulière est remplie. Il existe 3 types de boucles, chacune étant adaptée à des cas particuliers : la boucle ***for***, la boucle ***while***, la boucle ***do while***.

Quelques cas où l'utilisation des boucles est indispensable dans la vie du web :

- Afficher une liste des derniers articles sur un blog dans une page HTML
- Afficher la liste d'un panier d'achat dans une table HTML.
- Créer une fonction de recherche par mots-clés

3 Ajouter dynamiquement des entrées dans un tableau

Vous vous souvenez ? un tableau est une variable capable de stocker plusieurs données.
Vous pouvez vous représenter un tableau sous cette forme.

Index	0	1	2
Valeur	Javascript	Php	Mysql

*Ici la variable que nous nommerons **blogCategories** contient plusieurs valeurs associées à un index.*

3.1 Comment **Ajouter des éléments** à la fin du tableau avec ***push()***

A lire : http://devdocs.io/javascript/global_objects/array/push

La méthode push() ajouter un ou plusieurs éléments à la fin d'un tableau, et retourne le nombre d'entrée du nouveau tableau.

```
// On déclare le tableau avec les crochets puis on assigne les valeurs à l'intérieur du tableau
var blogCategories = ['Javascript', 'Php', 'Mysql'];

// on ajoute des valeurs à la fin du tableau avec .push()
blogCategories.push('HTML');
blogCategories.push('CSS');

console.log(blogCategories);
// retourne ['Javascript', 'Php', 'Mysql', 'HTML', 'CSS']
/*
Nous pouvons aussi ajouter plusieurs valeurs avec un push()
par exemple :
*/
console.log(blogCategories.push('Symfony','Laravel','Cakphp'));
// retourne 8, soit le nombre d'entrées du nouveau tableau
console.log(blogCategories);
// retourne ['Javascript', 'Php', 'Mysql', 'HTML', 'CSS', 'Symfony', 'Laravel', 'CakePhp']
```

Note : Ouvrez la console du navigateur (Firebug ou Chrome) et lancez le code Jsfiddle suivant : https://jsfiddle.net/learnlab/w3L3ypqp

3.2 Comment **supprimer des éléments *pop()*** et ***shift()***

A lire : http://devdocs.io/javascript/global_objects/array/pop
http://devdocs.io/javascript/global_objects/array/shift

La méthode ***pop()*** retire le dernier élément à la fin d'un tableau, et retourne cet élément.

De la même façon, La méthode ***shift()*** retire le premier élément au début d'un tableau, et retourne cet élément.

Note : Ouvrez la console du navigateur (Firebug ou Chrome), copiez et exécutez ce code

```
// On déclare le tableau avec les crochets puis on assigne les valeurs à l'intérieur du tableau

var blogCategories = ['Javascript', 'Php', 'Mysql'];

blogCategories.pop(); // on supprime le dernier élément du tableau avec .pop()
console.log(blogCategories); // retourne ['Javascript', 'Php']

blogCategories.shift(); // on supprime le premier élément du tableau avec .pop()
console.log(blogCategories); // retourne ['Php']
```

A retenir :

Vous pouvez ajouter ou supprimer dynamiquement des valeurs dans un tableau :

- la méthode push() ajoute une entrée à la fin du tableau
- la méthode unshift() ajouter une entrée en début de tableau
- la méthode pop() supprime la dernière entrée et retourne sa valeur
- la méthode shif() supprime la première entrée et retourne sa valeur

4 Les fonctions

Javascript fournit déjà des fonctions que vous connaissez. Appliquée à des objets on les appelle aussi ***méthodes. parseInt(), parseFloat(), getFullYear(), prompt(), toUpperCase()*** *sont par exemple des fonctions fournies par Javascript.*

par exemple ***new Date().getFullYear()*** *est une fonction qui nous retourne l'année en cours à 4 chiffres.*

D'accord, mais concrètement c'est quoi une fonction ?
Une ***fonction est élément auquel vous donnez un nom, elle*** qui contient votre code, c'est à dire des instructions à exécuter, et elle peut retourner ou non une valeur. Vous pouvez alors exécuter ce code en appelant simplement la fonction où vous voulez dans votre programme général.

Et vous allez maintenant pouvoir créer VOS propres fonctions. Cool non ?

4.1 Pourquoi créer vos propres fonctions ?

Son utilité réside dans le fait qu'elle est réutilisable partout dans votre programme dès que vous en avez avez besoin, en appelant simplement la fonction.

Les fonctions permettent ainsi de simplifier énormément votre code et aussi de l'organiser de façon claire.

4.2 Comment utiliser les fonctions ?

Syntaxe d'une fonction :

```
function nomDeMaFonction() {
// Code à exécuter
}
// Votre fonction peut contenir 1 ou plusieurs paramètres. Ils se placent dans les parenthèses // et sont séparés par des virgules
function nomDeMaFonction(paramètre1, paramètres2, … ) {
// Code à exécuter
}
```

Voyons tout de suite 2 exemples de fonctions avec et sans paramètres …

Exemple : une fonction qui dit bonjour et donne la date et l'heure

```
// Nous déclarons une fonction que l'on décide de nommer direBonjour. Ici elle ne prend aucun paramètre.
function direBonjour() {
    var date = new Date();
    alert('HELLO ! Nous sommes le ' + date.getDay() + '/' + date.getMonth() + '/' +
date.getFullYear());
}

// Puis on peut maintenant appeler simplement la fonction direBonjour() quand on le souhaite
direBonjour(); // affichera la boîte alert avec le message
```

Exemple : la même fonction, mais qui prend ici une variable ***prenom*** en paramètre

```
// On déclare  la même fonction à laquelle nous passons en paramètre une variable prénom.
function direBonjour(prenom) {
     var date = new Date();
     alert('HELLO ' + prenom + ' ! Nous sommes le ' + date.getDay() + '/' +
date.getMonth() + '/' + date.getFullYear());
}

direBonjour('Fred');
// affichera la boîte alert avec le message contenant également le prénom passé ici en paramètre
```

Autre exemple : une fonction qui calcule et retourne le produit de 2 nombres

```
// Nous passons en paramètre 2 variables a et b.
function multiplier(a,b) {
    var resultat = a*b;
    return resultat;
}
multiplier(3,5); // retournera alors la valeur 15 dans la console

// vous pouvez alors utiliser cette valeur en la récupérant dans une variable pour l'afficher par exemple
// var monResultat = multiplier(3,5);
// alert (monResultat);
```

Note : Ouvrez la console du navigateur (Firebug ou Chrome), copiez et exécutez ce code. Puis amusez-vous à modifier les paramètres de ***multiplier(3,5)*** *pour tester.*

L'opportunité de créer une fonction vous permet surtout d'y insérer un code long et plus complexe. Et vous pourrez alors utiliser votre fonction dans votre programme général de façon très simple.

Vous vous rappelez du code qui recherchait la phrase la plus longue dans un tableau.
Si nous en faisions une fonction ?

```
function rechercherLePrenomLePlusLong (phrases) {
    var index;
    var indexPhraseLaPlusLongue;
   indexPhraseLaPlusLongue = 0;
   for(index = 0; index < phrases.length; index++)
  {
        if(phrases[index].length > phrases[indexPhraseLaPlusLongue].length)
        {
                indexPhraseLaPlusLongue = index;
        }
  }
   // Affichage du résultat dans un alert
   alert (
        "Le prénom le plus long de la promo60 est : '" +
        phrases[indexPhraseLaPlusLongue] + "' " +
        'et il fait ' +
        phrases[indexPhraseLaPlusLongue].length + ' ' +
        'caractère(s) !'
        );
}
```

Maintenant nous pouvons utiliser la fonction

```
var promo60;
promo60 = ['Marie', 'Elodie', 'Zakaria', 'David', 'Constant', 'Christian',  'Aurélien', 'Bertrand'];

rechercherLePrenomLePlusLong(promo60);
// affiche le prénom le plus long et le nombre de caractères
```

Note : Ouvrez ce code sur https://jsfiddle.net/learnlab/rdngqhed

5 Les outils du développeur / liens utiles

Les outils du développeur :

- Un éditeur de texte ou un IDE (Sublime Text, Netbeans, PHPStorm,...)
- L'outil développement sur les navigateurs (Firebug sur FireFox et l'outil de développement sur Chrome, tout deux accessibles avec le raccourci F12)
- DevDocs (www.devdocs.io) [Le dictionnaire du développeur]

Liens utiles :

Cours sur OpenClassRoom

- https://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript/

Application mobile pour apprendre les bases avec des petits exercices progressifs (sympa dans le métro :))

- (AppStore et PlayStore et windowsStore).
- => https://play.google.com/store/apps/details?id=com.sololearn.javascript&hl=fr
- => https://itunes.apple.com/us/app/learn-javascript/id952738987?mt=8
- => https://www.microsoft.com/fr-fr/store/apps/learn-javascript/9wzdncrdj6b4
- => http://www.sololearn.com/Course/JavaScript

Conseil : révisez les notions apprises en relisant document et le précédent (SEM1 SEM2)

et PRATIQUEZ , c'est la meilleure voie.
Si vous n'êtes pas inspiré(e), pratiquez avec des exemples trouvés sur le web, sur le lien d'OpenclassRoom par exemple.

Si vous avez des questions ou des remarques, je suis joignable sur le https://3wa18.slack.com/ ou par Email : frederic.lossignol@gmail.com

Frédéric Lossignol, votre prof de Developpement