Contrôle de la qualité III

Module IV: Javascript

Enseignant: **Adel Ghlamallah**

Institut Grasset

Contenu du module

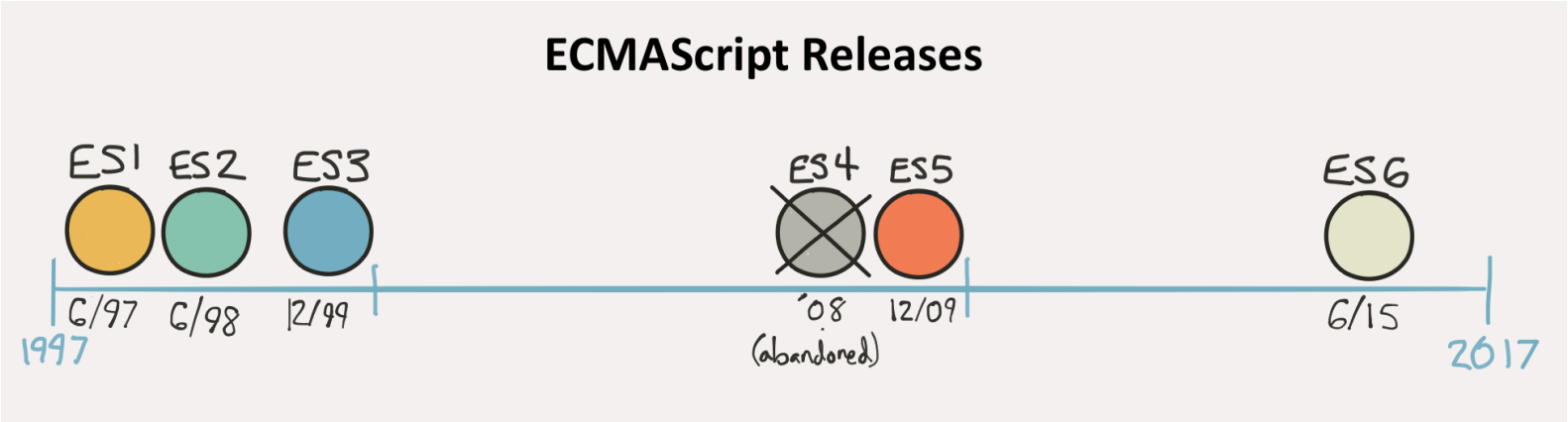
- Le langage HTML/CSS/JavaScript
- Les scripts, les DRM
- Les variables, les fonctions.

Introduction à JavaScript

- JavaScript a été standardisé en 1997 sous le nom d'ECMAScript.
- Depuis, le langage a subi plusieurs séries d'améliorations pour corriger certaines maladresses initiales et supporter de nouvelles fonctionnalités.
- JavaScipt était initialement crée pour être utilisé dans la partie client (navigateur) d'une application Web

Versions

- Depuis, le langage a subi plusieurs séries d'améliorations pour corriger certains problèmes et ajouter de nouvelles fonctionnalités.



- La dernière version de ECMAScript est ES2018 mais elle n'est pas supportée par toutes les versions.
- Dans ce cous, on va se baser sur la version ES2015 (appelée aussi ES6)

Introduction à JavaScript

- JavaScript est un langage de programmation interprété
- Il est donc nécessaire de posséder un interpréteur pour exécuter du code JavaScript
- Chaque navigateur possède un interpréteur JavaScript, qui diffère selon le navigateur.

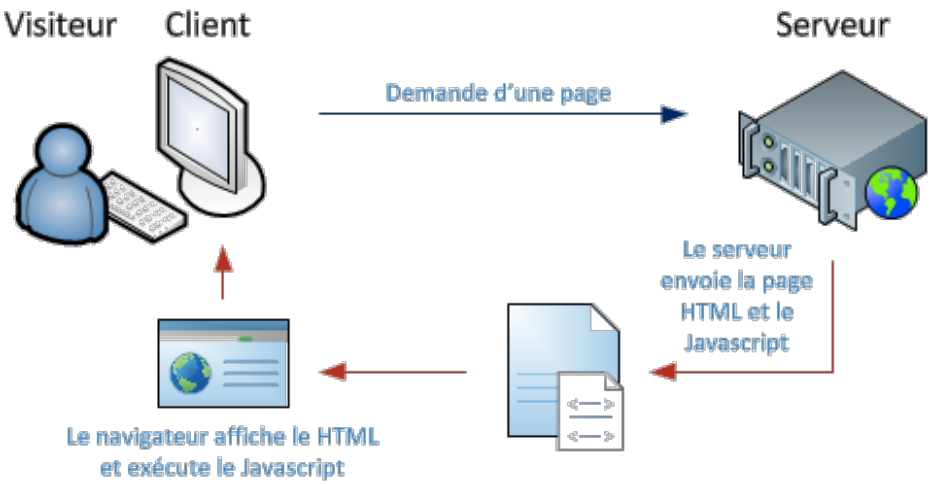
Introduction à JavaScript

- Le JavaScript est souvent utilisé sur Internet, conjointement avec les pages Web HTML.
- JavaScript s'inclut directement dans la page Web (ou dans un fichier externe) et permet de dynamiser une page HTML, en ajoutant des interactions avec l'utilisateur, des animations, de l'aide à la navigation
- N'importe quel navigateur Web est aujourd'hui capable d'exécuter du code écrit en JavaScript.
- Avant, JavaScript était un langage dit client-side, c'est-à-dire que les scripts sont exécutés par le navigateur chez l'internaute (le client).

Introduction à JavaScript

- L'explosion du Web puis l'avènement du Web 2.0 ont rendu JavaScript de plus en plus populaire.
- Les concepteurs de navigateurs Web ont optimisé la rapidité d'exécution du code JavaScript, jusqu'à en faire un langage très performant.
- Cela a conduit à l'apparition en 2009 de la plate-forme Node.js, qui permet d'écrire en JavaScript des applications Web très rapides du côté serveur (server-side)

Introduction à JavaScript



- Enfin, l'arrivée des tél. intelligents et tablettes dotés de systèmes différents et (iOS, Android) a conduit à l'apparition d'outils de développement dits multi-plateformes. Ils permettent d'écrire en une seule fois des applications mobiles compatibles avec les différents systèmes
- Certaines de ces plateformes sont basées sur JavaScript comme React Native.

Introduction à JavaScript

- JavaScript est maintenant présent partout. Il est devenu un des langages les plus utilisés au monde.
- Il dispose d'un large écosystème de composants, de plateformes (Jquery, Angular, Reac, NodeJs, …etc) et d'une immense communauté de développeurs.
- Sa connaissance vous ouvrira les portes de la programmation côté navigateur Web (on parle de développement front-end), côté serveur (back-end) ou côté mobile.
- A l'heure actuelle, beaucoup considèrent JavaScript comme la technologie la plus importante dans le monde du développement logiciel.
- Sa dimension universelle conjuguée à sa facilité d'accès font de JavaScript un excellent choix comme premier langage pour débuter la programmation.

Évolution de JavaScript

- Cette vidéo montre l'évolution de l'utilisation des plateformes JavaScript dans GitHub:

https://www.youtube.com/watch?v=l2tZ4V0UHac

- Les langages les plus populaires sur le site Stack Overflow :

https://www.youtube.com/watch?v=cKzP61Gjf00

Ajouter JavaScript dans une page html

- Deux façons d'intégrer Javascript dans une page HTML:
  - Mettre le code Javascript entre les balises :<script></script>

```
<script>
    alert('Mon premier code JavaScript!');
</script>
```

  - Mettre le code Javascript dans un fichier externe et l'importer dans la page :

```
<script src="nom_fichier.js"></script>
```

Les bases de Javascript

- Comme les autres langages, JavaScript inclut:
  - Des variables, tableaux, objets
  - Des boucles, conditions,
  - Des fonctions
  - Et mêmes des classes …
- Toutes les instructions dans JavaScript se terminent par un point virgule « ; »

Les commentaires

- Les commentaires peuvent être sur une seule ligne en les préfixant par une double barre oblique //

```
alert("Premier code JavaScript"); // Ceci est une commentaire
```

- Il existe une autre manière de créer des commentaires en entourant une ou plusieurs lignes par les caractères /* et */

```
/* Un commentaire
sur plusieurs
lignes */
```

Déclaration des variables

- La déclaration de variable doit commencer par le mot clé **var**. À partir de la version ES2015, il est possible aussi d'utiliser **let** à la place de var.
- On a pas besoin de déclarer le type d'une variable. Javascript est une langage dynamiquement typé.
- Exemple:

```
var a;
```

Affectation de valeurs

- Pour assigner une valeur à une variable :

```
var a;
 a = 10;
alert (a);
```

Ou

```
var a = 10;
alert(a);
```

- Si la valeur initiale d'une variable ne changera jamais au cours de l'exécution du programme, cette variable est ce qu'on appelle une constante.

```
const a = 10; // Création d'une variable constante
a = 20;       // Impossible : a ne peut pas changer de valeur !
```

Déclaration et affectation

- On peut déclarer et assigner des variables sur une seule et même ligne :

  ```
  var  a, b = 10, c;
  ```

- On peut aussi écrire:

  ```
  var a, b;
  a = b = 2;
  ```

Types de variables

- En JavaScript, nos variables sont typées dynamiquement, ce qui veut dire que l'on peut y mettre du texte en premier lieu puis l'effacer et y mettre un nombre quel qu'il soit, et ce, sans contraintes.
- Les trois types principaux en JavaScript sont :
  - Le type numérique
  - Les chaînes de caractères (*string*)
  - Les booléens (*boolean*)

Type numérique

- Le type numérique (number) représente tout nombre, que ce soit un entier, un nombre négatif, un réel, etc.
- Les opérateurs numériques peuvent être appliqués aux valeurs numériques: +, -, *, / et % (modulo)
- Exercice:

```
var diviseur = 3, resultat1, resultat2, resultat3;
resultat1 = (16 + 8) / 2 - 2 ; // 10
resultat2 = result1 / diviseur;
resultat3 = resultat1 % diviseur;
alert(resultat2); // Résultat de la division : ???
alert(resultat3); // Reste de la division : ???
```

Opérations numériques

- Il est également possible d'augmenter ou de diminuer la valeur d'un nombre avec les opérateurs+= (-=) et++ (--).
- ++ est appelé opérateur d'incrémentation, car il permet d'incrémenter (augmenter de 1) la valeur d'une variable.

```
Var b = 0;      // b contient 0
b += 1;         // b contient 1
b++;            // b contient 2
b+=5;        //équivalent à: b = b+5;
console.log(b); // 2
```

Opérations numériques

- À noter que ceci ne s'applique pas uniquement aux additions mais fonctionne avec tous les autres opérateurs arithmétiques :
  - +=
  - -=
  - *=
  - /=
  - %=

Chaînes de caractères (String)

- Le type String représente n'importe quel texte. On peut l'assigner de deux façons différentes :
  - var texte1 = "Mon premier texte"; // Avec des guillemets
  - var texte2 = 'Mon deuxième texte'; // Avec des apostrophes

- Il est important de se rappeler que si on:

```
var a= '10';
```

alors le type de cette variable est une chaîne de caractères et non pas un type numérique.

- Une autre précision importante, si on utilise des apostrophes pour « encadrer » untexte et qu'on a besoin utiliser des apostrophes dans ce même texte, il vous faudra alors « échapper » les apostrophes comme suit:

```
var a= 'Ça c\'est quelque chose !';
```

Chaînes de caractères (String)

- JavaScript offre plusieurs méthodes pour manipulés des chaînes de caractères (String): +, concat(), toUpperCase(), toLowerCase(), length
- La concaténation consiste à ajouter une chaîne de caractères à la fin d'une autre, comme dans cet exemple :
  - var a = 'Bonjour', nom = ' la vie', resultat;
  - resultat = a+ nom;
  - alert(resultat); // Affiche : « Bonjour la vie»
- On peut aussi faire :
  - var texte = 'Bonjour ';
  - texte += ' la vie';

Chaînes de caractères (String)

- Exemples pour: , concat(), toUpperCase(), toLowerCase(), length
-

Booléen

- Les booléens c'est des valeurs logiques qui peuvent être soient True(vrai) ou False(faux)
- un booléen est un type à deux états qui sont les suivants : vrai ou faux. Ces deux états s'écrivent de la façon suivante :
- var isTrue = true;
- var isFalse = false;

Interagir avec l'utilisateur

- Une des façon d'interagit avec l'utilisateur est d'utiliser la fonction prompt():
  - var prenom = prompt('Entrez votre prénom :');
  - alert(prenom); // Affiche le prénom entré par l'utilisateur
- Exercice:

```
var a = 'Bonjour ', nom, fin = ' !', resultat;
nom   = prompt('Quel est votre prénom ?');
resultat = debut + nom + fin;
alert(resultat);
```

Conversion entre types

- Convertir une chaîne de caractères en nombre

```
var a, b, resultat;
a  = prompt('Entrez le premier chiffre :');
b = prompt('Entrez le second chiffre :');
resultat = a + b;
alert(resultat);
```

- Le champ de texte de prompt() est récupéré sous forme d'une chaîne de caractères, que ce soit un chiffre ou non. Du coup, si on utilise l'opérateur +, on ne fera pas une addition mais une concaténation !

- C'est là que la conversion des types intervient. Il suffit de convertir la chaîne de caractères en nombre.

- Pour cela, on utilise la fonction **parseInt()** pour convertir en entier ou **Number()** pour convertir en nombre (réel ou entier) :

```
resultat = parseInt(a) + parseInt(b);
```

Conversion entre types

- Pour convertir un nombre en chaîne de caractères:
  - Si on concatène un nombre et une chaîne la conversion de fait automatiquement en chaîne de caractère
  - Mais si on concatène deux nombres, ceux-ci s'ajouteraient à cause de l'emploi du +. D'où le besoin de convertir un nombre en chaîne. Voici comment faire :

```
var texte, nombre1 = 4, nombre = 2;
texte = nombre1 + '' + nombre2;
alert(texte);
```

- Nous avons juste ajouté une chaîne de caractères vide entre les deux nombres, ce qui aura eu pour effet de les convertir en chaînes de caractères.

Nommage des variables

- Le nom choisi pour une variable n'a pour la machine aucune importance, et le programme fonctionnera de manière identique.
- La manière dont sont nommées les variables affecte grandement la facilité de compréhension d'un programme: choisir des noms significatifs.
- Avant:

```
const nb1 = 10;
const nb2 = 3.14;
const nb3 = 2 * nb2 * nb1;
console.log(nb3);
```

- Après:

```
const rayon = 10;
const pi = 3.14;
const perimetre = 2 * pi * rayon;
console.log(perimetre);
```

Exercice

- Écrire un programme JavaScript qui permet :
  - L'utilisateur fait entrer une température en degré Celsius
  - Le programme converti la température en Fahrenheit et l'affiche

Exercice

```
<script type="text/javascript">

    var tempCel = parseInt(prompt("Entrez temperature en
degre Celsius"));

    var tempFar = tempCel * 9 / 5 + 32;

    alert(tempCel +' Celsuis  = '+tempFar+' Fahrenheit');

  </script>
```

Exercice: permutation de nombres

- Écrire un programme JavaScript qui permet de permuter deux nombres sans utiliser de variable temporaire

Exercice: permutation de nombres

```
<script type="text/javascript »>
    var nombre1 = 10;
    var nombre2 = 20;

    nombre1 += nombre2;
    nombre2 = nombre1 - nombre2;
    nombre1 -= nombre2;

    console.log(nombre1);
    console.log(nombre2);
</script>
```

Instructions de conditions

- Pour une exprimer une condition dans JavaScript, on utilise l'instruction **if**.

```
if (condition) {
  // instructions exécutées si la condition est vraie
}
```

- Une condition est une expression dont l'évaluation produit une valeur soit vraie, soit fausse : on parle de valeur booléenne. Ce type n'a que deux valeurs possibles :true (vrai) et false (faux).

- Exemple:

```
var nombre = Number(prompt("Entrez un nombre :"));
if (nombre > 0) {
  console.log(nombre + " est positif");
}
```

Les opérateurs de comparaison

- Les commandes de condition dans JavaScript, comme dans les autres langages de programmation, sont formés d'opérateur de logique et de comparaison.
  - Les opérateurs de comparaison

Opérateur	Signification
==	égal à
!=	différent de
===	contenu et type égal à
!==	contenu ou type différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Exercice:

```
var nbre = Number(prompt("Entrez un nombre:"));
if (nbre > 0) {
  console.log(nbre + " est positif");
}
else {
  console.log(nbre + " est négatif ou nul");
}
```

Les opérateurs de logiques

- En plus des opérateurs logiques, JavaScript utilise les opérateurs logiques pour exprimer une condition.

Opérateur	Type de logique	Utilisation
&&	ET	valeur1 && valeur2
||	OU	valeur1 || valeur2
!	NON	!valeur

- Exercice: Écrire un programme qui permet de vérifier si un nombre entré par un utilisateur est entre 0 et 1000

Exercice

```
var nombre = Number(prompt("Entrez un nombre :"));
if ((nombre < 0)  ||  (nombre > 1000)) {
  console.log(nombre + " est en dehors de l'intervalle [0, 100]");
}
```

Conditions avec choix

- En plus de l'instruction **if**, on peut aussi utilise l'instruction **else**:

```
if ( condition  ) {
    // Du code…
} else { //condition non satisfaite
    // Un autre code …
}
```

- On peut aussi ajouter un **else if** si on en a besoin:

```
if (  condition 1 ) {
    // Du code…
} else if (condition 2) {
    // Autre code
}  else { //Ni condition 1, ni condition 2 ne sont satisfaites
    // Un autre code …
}
```

Exercice

- Exercice: écrire le code qui selon qu'il fait (soleil, pluie ou neige), il vous propose le vêtement qu'il faut mettre:

```
var meteo = prompt("Quel temps fait-il dehors ?");
if (meteo === "soleil") {
  console.log("Portez un t-shirt.");
} else if (meteo === "pluie") {
  console.log("Ramenez un parapluie.");
} else if (meteo === "neige") {
  console.log("Restez au chaud à la maison.");
} else {
  console.log("Erreur: choisir soit: soleil, pluie ou neige!");
}
```

La condition « switch »

- La condition **if else** n'est pas toujours pratique pour faire du cas par cas ; c'est là que l'instruction **switch** devient utile.

- La syntaxe est la suivante:

```
switch (expression) {
  case valeur1:
    // instructions exécutées quand expression vaut valeur1
    break;
  case valeur2:
    // instructions exécutées quand expression vaut valeur2
    break;
  ...
  default:
    // instructions exécutées quand aucune des valeurs ne correspond
}
```

- Les instructions **break;** dans les blocs **case** sont indispensables pour sortir du **switch** et éviter de passer d'un bloc à un autre.

Exercice

- Réécrire l'exemple précédent en utilisant l'instruction **switch**:

```
var meteo = prompt("Quel temps fait-il dehors ?");
if (meteo === "soleil") {
  console.log("Portez un t-shirt.");
} else if (meteo === "pluie") {
  console.log("Ramenez un parapluie.");
} else if (meteo === "neige") {
  console.log("Restez au chaud à la maison.");
} else {
  console.log("Erreur: choisir soit: soleil, pluie ou neige!");
}
```

Exercice (solution)

```
var meteo = prompt("Quel temps fait-il dehors ?");
switch (meteo) {
  case "soleil":
    console.log("Sortez en t-shirt.");
    break;
case "pluie":
    console.log("Ramenez un parapluie.");
    break;
  case "neige":
    console.log("Restez au chaud à la maison.");
    break;
  default:
    console.log("Choisir soit: soleil, pluie ou neige!");
}
```

Tester l'existence de contenu d'une variable

```
var test = 'vrai ?';
if (test) {
    alert('Vraie !');
} else {
    alert('Faux !');
}

var test = '';
if (test) {
    alert('Vraie !');
} else {
    alert('Faux !');
}
```

- Comment définir si le contenu d'une variable est vrai ou faux ?
- Le contenu est converti à faux si : un nombre qui vaut zéro ou bien une chaîne de caractères vide. Sinon c'est considéré comme vrai.

Exercice

Écrire le programme qui demande à un utilisateur de faire entrer son âge. Selon l'âge entré, il affiche un commentaire

- Age entre :72 ans - 54 ans
  Les baby-boomers

- Age : 53 ans - 38 ans
  La génération X

- Age entre 38 ans - 18 ans
  La génération Y ou génération des milléniaux

- Age entre 18 ans - 8 ans
  La génération Z

- Age moins de 8 ans
  La génération alpha

Les boucles

- JavaScript propose trois principales formes pour exprimer des boucles:
  1. for (condition) {….}
  2. while (condition) {…}
  3. do { ……..} while (condition)

La boucle avec l'instruction «while»

- La boucle **while** permet de répéter des instructions tant qu'une condition est vérifiée.
- La syntaxe de l'instruction **while** est la suivante.

```
while (condition) {
  // instructions exécutées tant que la condition est vérifiée
}
```

- Exercice:

```
 alert ("Démarrer !")
  var nombre = 1;
  console.log("Au début du programme, la valeur est:"+nombre);

  while (nombre <= 5) {
   console.log(nombre);
   nombre++;
 }
console.log("Fin du programme, la valeur est:"+nombre);
```

Attention à la boucle infinie

- Une des erreurs fréquente liée à la boucle while est la "boucle infinie".
- Modifiez l'exemple suivant en oubliant volontairement la ligne qui incrémente la variable nombre.

```
Var nombre = 1;
while (nombre <= 5) {
    console.log(nombre);
    // La variable nombre n'est plus modifiée : la condition sera toujours vraie
}
```

- Lors de l'exécution de ce programme, on effectue un premier tour de boucle puisque la condition nombre <= 5 est initialement vérifiée. Mais comme on ne modifie plus la variable nombre dans le corps de la boucle, la condition est indéfiniment vraie : il s'agit d'une boucle infinie.

La boucle avec l'instruction «for»

- L'instruction **for** est une autre manière d'exécuter yne boucle.
- Voici la syntaxe:

```
for (initialisation; condition; étape) {
    // instructions exécutées tant que la condition est vérifiée
}
```

- **L'initialisation** se produit une seule fois, au début de l'exécution.
- La **condition** est évaluée avant chaque tour de boucle. Si elle est vraie, un nouveau tour de boucle est effectué. Sinon, la boucle est terminée.
- **L'étape** est exécutée après chaque tour de boucle.

Exercice

- Écrire le même exemple vu pour l'instruction **while** avec l'instruction **for**:

```
alert ("Démarrer !")
 for (var nombre = 1; nombre <= 5; nombre++) {
   console.log(nombre);
 }
 console.log("Fin du programme, la valeur est:"+nombre);
```

Attention à la manipulation du compteur

- Une des erreurs fréquente pour la boucle avec l'instruction **for** est la manipulation du compteur dans le corps de la boucle
- Dans l'exemple précédent, on modifie exprès la valeur du compteur nombre dans le corps de la boucle:

```
for (var nombre = 1; nombre <= 5; nombre++) {
   console.log(nombre);
   nombre++;
 }
```

- Dans l'exemple en haut, à chaque tour de boucle, la variable nombre est incrémentée deux fois : dans le corps de la boucle, puis dans l'étape exécutée à la fin de chaque itération.

Exercice

- Écrire un programme qui afficher la table de multiplication de 1 à 10 d'un nombre entré par un utilisateur

Exercice: solution

```
var nombre =  prompt("Entrez un nombre:");
  nombre = Number(nombre);
  for (var i = 1; i <= 10; i++) {
    console.log(nombre+"*"+i+" = " + nombre*i);
  }
```

Quiz

Combien de fois le message “Il fait beau!” sera-t-il répété avec le programme suivant ?

```
var i = 0;

while (i <= 4) {
   console.log("Il fait beau!");
   i++;
}
```

- 1 fois
- 4 fois
- 5 fois
- 6 fois

Quiz

Combien de fois le message “Il fait beau!” sera-t-il répété avec le programme suivant ?

```
for (var i = 1; i < 5; i++) {
   console.log("Il fait beau !");
}
```

- 1 fois
- 4 fois
- 5 fois
- 6 fois

Fonctions

- Une fonction est un groupement d'instructions qui réalise une tâche déterminée.
- Nous avons déja utilisé plusieurs fonctions de JavaScript:alert(),prompt() et parseInt().
- Comme les autres langages, on peut aussi créer nos propres fonctions en JavaScript
- Voici un exemple simple utilisant une fonction:

```
function afficherDate() {
  console.log("La date aujourd'hui est: "+Date);
}
afficherDate();
```

Déclaration d'une fonction

- L'opération de création d'une fonction s'appelle la déclaration. Voici sa syntaxe:

```
// Déclaration d'une fonction nommée maFonction
function maFonction(arguments) {
  // Instructions de la fonction
}
```

- Le mot-clé **function** est présent à chaque déclaration de fonction. C'est lui qui permet de créer une fonction
- Ensuite, c'est le nom de la fonction (maFonction)
- Puis, un couple de parenthèses contenant ce que l'on appelle des arguments. Ces arguments servent à fournir des informations à la fonction lors de son exécution. Les paramètres sont optionnels
- Enfin, un couple d'accolades contenant le code que votre fonction devra exécuter.

Appel d’une fonction

- Il est important de préciser que tout code écrit dans une fonction ne s'exécutera que si on appelle cette dernière (« appeler une fonction » signifie « exécuter »). Sinon, le code qu'elle contient ne s'exécutera jamais.

Exercice

- Réécrire le code suivant pour qu'il utilise une fonction:

```
var resultat;

resultat = parseInt(prompt('Donnez le nombre à multiplier par 5 :'));
alert(resultat * 5);

alert("Assurez-vous d'entrer un nombre différent pour la suite !");

resultat = parseInt(prompt('Donnez le nombre à multiplier par 5 :'));
alert(resultat * 5);
```

Exercice: solution

```
function multiplierParCinq() {
  var resultat = parseInt(prompt('Donnez le nombre à multiplier par 5 :'));
  alert(resultat * 5);
}

multiplierParCinq();
alert("Assurez-vous d'entrer un nombre différent pour la suite !");
multiplierParCinq();
```

Portée des variables

- Toute variable déclarée dans une fonction n'est utilisable que dans cette même fonction ! Ces variables spécifiques à une seule fonction ont un nom : les **variables locales**.

- Exemple:

```
function afficherMessage () {
  var message = "Bonjour !";
  alert (message);
}
afficherMessage();
```

Portée des variables

- À l'inverse des variables locales, celles déclarées en-dehors d'une fonction sont nommées les **variables globales** car elles sont accessibles partout dans le code, y compris à l'intérieur de vos fonctions.

  Exemple:

```
 var message = "Bonjour ! ";
function afficherMessage () {
  alert (message);
}
afficherMessage();
```

Exercice

- Que va t-il se passer dans ses 2 exemples de code:

```
var message= 'Salut!';
function afficherMessage() {
    alert(message);
}
afficherMessage();
```

```
function afficherMessage() {
    var message= 'Salut!';
}
afficherMessage();
alert(message);
```

Exercice

- Qu'est-ce qui se produirait si on crée une variable globale nommée message et une variable locale du même nom comme dans l'exemple suivant ?

```
var message = 'variable globale !';
function afficherMsg() {
    var message = 'variable locale !';
    alert(message);
}
afficherMsg();
alert(message);
```

Les arguments d'une fonction

- Les arguments sont des informations envoyées à une fonction
- Voici la syntaxe générale de la déclaration d'une fonction acceptant des paramètres. Leur nombre n'est pas limité, mais il est rarement nécessaire de dépasser 3 ou 4 paramètres

```
// Déclaration de la fonction maFonction
function maFonction(param1, param2, ...) {
  // Instructions pouvant utiliser param1, param2, ...
}
// Appel de la fonction maFonction
// param1 reçoit la valeur de arg1, param2 la valeur de arg2, ...
maFonction(arg1, arg2, ...);
```

Les arguments d'une fonction

- Exemple:

```
function afficherMessage(nom) {
  alert ('Bonjour '+nom);
}
afficherMessage("Jean");
afficherMessage("Lolita");
```

- Exercice: Écrire une fonction qui reçoit deux arguments (largeur et longueur) qui calculer la superficie d'un rectangle et l'affiche.

Exercice: solution

```
function calculerSurface (longueur, largeur) {
  var surface = longueur * largeur;
  alert("Surface est:"+longueur*largeur);
}
calculerSurface(10,5);
```

Retour de valeur

- Une fonction peut retourner une valeur comme c'est le cas des fonctions JavaScript suivantes: prompt() et parseInt();

- Pour retourner une valeur une fonction utilise l'instruction **return** suivie de la valeur à retourner:

```
function calculerSurface (longueur, largeur) {
  var surface = longueur * largeur;
  return surface;
}
alert(calculerSurface(10,5));
```

- L'exécution de l'instruction **return** renvoie immédiatement vers le programme appelant. Il ne faut jamais ajouter d'instructions après un **return** dans une fonction : elles ne seraient jamais exécutées.

Fonction anonyme

- Comme leur nom l'indique, ces fonctions spéciales sont anonymes car elles ne possèdent pas de nom.
- Pour déclarer une fonction anonyme, il suffit de faire comme pour une fonction classique mais sans indiquer de nom :

```
function (arguments) {
    // Le code de votre fonction anonyme
}
```

- Comment exécuter cette fonction, si elle n'a pas de nom ?

Fonction anonyme

- Il y a plusieurs façons d'exécuter ce type de fonction. Une des façons est de l'assigner à une variable, puis appeler cette variable.

- Exemple:

```
var surface = function(longueur, largeur) {
   return longueur*largeur;
};
alert(surface(10, 5));
```

Fonction anonyme

- Les dernières évolutions du langage JavaScript ont introduit une syntaxe plus concise pour créer des fonctions anonymes.
- L'exemple suivant est strictement équivalent au précédent.

```
var surface= (longueur, largeur) => {
  return longueur*largeur;
}
alert(surface(10, 5));
```

- Cette syntaxe est appelée fonction fléchée (fat arrow function).

Bonnes pratiques

- Créer des fonctions ayant chacune un rôle bien défini.
- Ne pas réinventer la roue et il est important de privilégier l'utilisation de ces fonctions existantes.
- Le corps d'une fonction ne doit pas être long. En général il est recommandé de ne pas dépasser 15 à 10 lignes par fonction.
- Le nommage des fonctions et des paramètres joue un rôle important dans la lisibilité du programme comme pour les variables.

Sommaire sur les fonctions

- Il existe des fonctions natives JavaScript (comme alert(), prompt()) mais il est aussi possible d'en créer, avec le mot-clé **function**.
- Les variables **locales** sont déclarées avec **var** dans une fonction et ne sont accessibles que dans cette fonction.
- Les variables globales sont déclarées à l'extérieur des fonctions et accessibles partout dans le code.
- Une fonction peut recevoir aucun, un ou plusieurs paramètres. Elle peut aussi retourner une valeur ou ne rien retourner du tout.
- Des fonctions qui ne portent pas de nom sont appelées fonctions anonymes et servent à isoler une partie du code.

Objets

- JavaScript est un langage orienté objet.
- JavaScript met à notre disposition des objets natifs, c'est-à-dire des objets directement utilisables comme par exemples: **String** et **Number**.
- Les objets contiennent trois choses distinctes :
  - un constructeur ;
  - des propriétés (variables);
  - des méthodes (fonctions).

Constructeur

- Le constructeur est un code qui est exécuté quand on crée un nouvel objet.
- Il permet d'effectuer des actions comme par exemple initialiser les variables au sein même de l'objet
- Exemple:

```
var chaineCar = 'Ceci est une chaîne de caractères'; // On crée un objet String
```

Propriétés d'objet

- Une propriété est une variable contenue dans l'objet, elle contient des informations nécessaires au fonctionnement de l'objet.
- Exemple:

```
alert(chaineCar.length); // affiche le nombre de caractères, au moyen de la propriété « length »
```

Méthodes d'objet

- Les méthodes sont des fonctions contenues dans l'objet, et qui permettent de réaliser des opérations sur le contenu de l'objet.
- Exemples:

```
alert(myString.toUpperCase()); // On récupère la chaîne en majuscules, avec la méthode toUpperCase()
```

Méthodes d'objets

- Les méthodes sont des fonctions contenues dans l'objet, et qui permettent de réaliser des opérations sur le contenu de l'objet.
- Par exemple, dans le cas d'une chaîne de caractères, il existe les méthodes suivantes:

```
var chaineCar1 = "Ma première chaîne ";
var chaineCar2 = "Ma deuxième chaîne ";
alert(chaineCar1.toUpperCase());
alert(chaineCar2.toLowerCase());
alert (chaineCar1.concat(chaineCar2.toLowerCase()));
alert (chaineCar1.indexOf('ch'));
```

Syntaxe d'un objet

- Pour créer un objet, on utilise la syntaxe suivante:

```
var monObjet= {};
```

- Si on veut ajouter des propriétés à un objet:

```
var myObject = {
    prop1: 'Texte 1',
    prop2: 'Texte 2'
};
```

- Pour accéder à une propriété d'un objet:

```
myObject.prop1;
```

Exercice

- Créer un objet qui définit un livre avec les propriétés suivantes: titre, auteur, année de publication.
- Puis, afficher avec alert (ou Console.log) les valeurs des 3 propriétés.

Solution

```
var livre = {
        titre: "Révolution Politique",
        auteur: "Jean Lula",
        anneePub: 2009,
        prix: 25$
};
alert(livre.titre);
alert(livre.auteur);
alert(livre.anneePub);
```

Modifier un objet

- Une fois un objet créé, on peut modifier les valeurs de ses propriétés avec la syntaxe:

```
monObjet.maPropriete = nouvelleValeur;
```

- Exercice: Modifier l'année de publication à 2019:

```
livre.anneePub = 2019;
```

Ajout d'une méthode à un objet

- On peut ajouter une méthode (fonction) à un objet.
- Exemple:

```
var livre = {
        titre: "Révolution Politique",
        auteur: "Jean Lula",
        anneePub: 2009,
        prix: 25,

        calculerPrix() {
                return this.prix*1.15;
        }
 };
alert( livre.calculerPrix());
```

Le mot-clé this

- Le mot-clé this est utilisé à l'intérieur d'une pour représente l'objet sur lequel la méthode a été appelée.
- La méthode calculerPrix() utilise this pour accéder aux propriétés de l'objet sur lequel elle a été appelée.

L'objet Math

- Le langage JavaScript met à la disposition des programmeurs un certain nombre d'objets standards qui peuvent rendre de multiples services.
- Un des objets les plus utilisés est Math.
- Voici quelques méthodes de Math:
  - Math.round() : arrondit un nombre réel au nombre entier
  - Math.round(4.7); // retourne 5
  - Math.pow(x, y) retourne x à la puissance de y
  - Math.pow(8, 2); // retourne 64

L'objet de Math

Math.sqrt(x) retourne la racine carée de x:
Math.sqrt(64); // retourne 8

Math.abs(x) retourne la valeur absolue de x:
Math.abs(-4.7); // returns 4.7

Math.sin()
Math.cos()
Math.min()
Math.max()

Exercice: Modélisation d'un compte bancaire

- Complétez ce programme pour créer un objet compte ayant les propriétés suivantes :
  - Une propriété titulaire valant "Jean".
  - Une propriété solde valant initialement 0.
  - Une méthode crediter() ajoutant le montant passé en paramètre (éventuellement négatif) au solde du compte.
  - Une méthode decrire() renvoyant la description du compte.
- Utilisez cet objet pour afficher sa description, le créditer de 250, puis le débiter de 80, et enfin afficher de nouveau sa description.

Solution

```
var compte = {
  titulaire: "Jean",
  solde: 0,
  // Ajoute un montant au solde
  crediter(montant) {
    this.solde += montant;
  }
};
// solde: 0
console.log(compte.solde);
compte.crediter(250);
compte.crediter(-80);
// solde: 170
console.log(compte.solde);
```

Sommaires des objets

- En plus des objets natifs (Math, String …etc.), on peut créer nos propres objets.
- Un objet contient un constructeur, des propriétés et des méthodes
- On peut créer un objet en définissant ses propriétés à l'intérieur d'une paire d'accolades.
- A l'intérieur d'une méthode, le mot-clé **this** représente l'objet sur lequel la méthode s'applique.

Les tableaux

- En JavaScript, un tableau est un objet disposant de propriétés particulières.
- Un tableau est un type de donnée qui permet de stocker un ensemble d'éléments.
- Chaque élément est accessible au moyen d'un indice (index en anglais) et dont la numérotation commence à partir de 0.

Indice	0	1	2	3	4
Donnée	Valeur 1	Valeur 2	Valeur 3	Valeur 4	Valeur 5

Création de tableau

- On crée un tableau à l'aide d'une paire de crochets [] .
- Tout ce qui se trouve entre les crochets correspond au contenu du tableau.
- Les différents éléments stockés sont séparés par des virgules et peuvent être de types différents.
- Exemple:

```
var langages = ["JavaScript","Java","C#", "Python", "Swift", "R"];
```

Accès à un tableau

- Pour accéder à un élément d'un tableau, il suffit de spécifier l'index voulu, entre crochets.
- Pour accéder:

```
var langages = ["JavaScript","Java","C#", "Python", "Swift", "R"];
alert (langages[0]);
```

- Pour modifier:

```
langages[2] = "C++";
alert(langages[2]);
```

- Utiliser un indice invalide pour accéder à un élément d'un tableau renvoie la valeur spéciale **undefined**.

Opérations sur les tableaux

- La méthode **push()** permet d'ajouter un ou plusieurs éléments **à la fin** d'un tableau.
- Exemple:

```
langages.push("PHP");
langages.push("Scala","Golang");
alert(langages[8]);
```

- La méthode **unshift()** fonctionne comme push(), excepté que les items sont ajoutés au début du tableau.
- Exemple:

```
langages.unshift("Perl");
alert(langages[0]);
```

Opérations sur les tableaux

- Les méthodes shift() et pop() retirent respectivement le premier et le dernier élément du tableau.
- Exemples:

```
langages.shift();
langages.pop();
alert(langages);
```

Chaînes de caractères et tableaux

- Les chaînes de caractères possèdent une méthodesplit()qui permet de les découper en un tableau, en fonction d'un séparateur.
- Exemple:

```
var chainesLangages = "JavaScript Java C# Python Swift R";
var langages = chainesLangages.split(" ");
alert (langages[0]);
alert (langages[1]);
```

Chaînes de caractères et tableaux

- L'inverse de split(), c'est-à-dire créer une chaîne de caractères depuis un tableau, se nomme **join()**:
- Exemple:

```
var langages = ["JavaScript","Java","C#", "Python", "Swift", "R"];
var nouvLangages = langages.join(' ');
alert (nouvLangages);
```

Parcourir un tableau

- En général, on utilise la commande **for** pour parcourir les éléments d'un tableau.
- Exemple:

```
var langages = ["JavaScript","Java","C#", "Python", "Swift", "R"];
for (var i = 0; i < langages.length; i++) {
    console.log(langages[i]);
}
```

Une autre façon de parcourir le tableau avec for:

```
for (var lang of langages) {
  console.log(lang);
}
```

Exercice1: Somme d'un tableau

- Écrire un programme qui permet de parcourir un tableau d'entiers et de calculer la somme de tous ses éléments du tableau suivant:

  var notes = [12, 8, 10, 20, 15, 5];

Solution: Somme d'un tableau

```
var notes = [12, 8, 10, 20, 15, 5];
var somme = 0;
for (var i = 0; i < notes.length; i++) {
    somme+=notes[i];
}
alert("Somme: "+somme)
```

Exercice: Maximum d'un tableau

- Écrire le programme qui calcule et affiche ensuite la plus grande valeur présente dans le tableau suivant:

```
var notes = [12, 8, 10, 20, 15, 5];
```

Solution: Maximum d'un tableau

```
var notes = [12, 8, 10, 20, 15, 5];
var max = 0;
for (var i = 0; i < notes.length; i++) {
    if (max < notes[i]) {
            max = notes[i];
    }
}
alert("Maximum: "+max);
```

Sommaire des tableaux

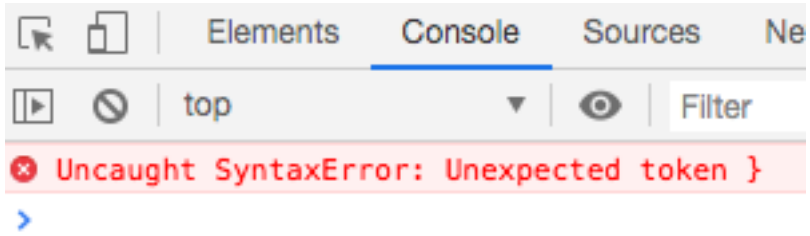
- Un tableau permet de stocker un ensemble d'éléments.
- Un tableau est un objet ayant plusieurs propriétés spécifiques, telles que length qui renvoie sa taille (nombre d'éléments).
- Chacun élément d'un tableau est identifié par un numéro appelé son indice qui commence à 0 et non pas 1.
- On accède à un élément particulier d'un tableau en plaçant son indice entre une paire de crochets [] .
- Des opérations peuvent être appliquées sur les tableaux, comme ajouter des éléments ou en supprimer.
- Pour parcourir un tableau élément par élément on utilise généralement la boucle **for**.

Débogage du code JavaScript

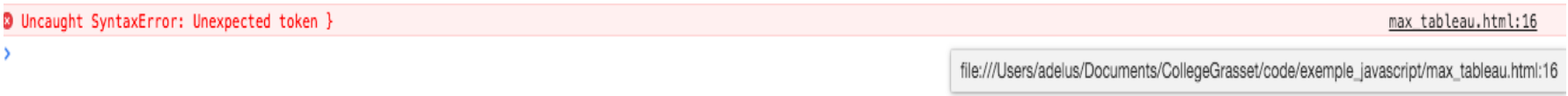
- Le débogage consiste à identifier et supprimer les bugs qui existent dans du code.
- Les bugs syntaxiques sont les plus simples à résoudre, car l'interpréteur JavaScript signale généralement l'endroit où l'erreur est apparue, cette signalisation peut être consultée dans la console de votre navigateur (avec Inspecter) .
- En ce qui concerne les bugs algorithmiques, là il va falloir chercher par vous-mêmes où vous avez bien pu vous tromper. Une des méthodes est d'utiliser des logs dans votre code (console.log) pour afficher les valeurs

Débogage

- En général, quand il y a une erreur de syntaxe on va voir dans la console une erreur en rouge **SyntaxError**:



- En plus, en général on peut voir dans la console à droite la ligne ou l'erreur s'est produite comme suit:



Débogage

- Exemple:

```
functin afficher() {

    alert("Bonjour ");
```

- Dans cet exemple, il y a deux erreurs mais juste une erreur est affichée. Pourquoi ?
  - Corriger la première erreur qui est indiquée dans la console
  - Puis, essayer de nouveau.
  - Qu'elle est l'erreur qui s'affichée maintenant ?
- Conseil: consultez toujours la console avant de demander de l'aide à qui que ce soit, vous trouverez généralement la source de votre problème indiquée dans la console.