Introduction à *JavaScript*

Qu'est-ce que le JavaScript ?

- JavaScript est un langage de programmation web ou d'interprétation
- JavaScript est utilisé pour ajouter de l'interactivité et de la dynamique aux sites web.
- JavaScript peut également être utilisé côté serveur grâce à des environnements comme Node.js pour créer des applications web complètes.

Attention ! Ne pas confondre avec le langage Java

Ou coder en JavaScript ?

- la balise `<script>` délimite code JavaScript
- peut être insérée soit dans l'entête (`<head>`) soit dans le corps (`<body>`) de la page HTML
- les scripts sont exécutés dans leur ordre d'apparition dans la page

Placer les scripts dans l'en tête pour qu'ils soient chargés et exécutés avant la construction et l'affichage de la page

Par défaut, les scripts placés dans le body sont exécutés au fur et à mesure que celui-ci est chargé et que le DOM est construit.
→ met en attente le moteur d'analyse HTML/CSS

Souvent les scripts sont placés à la fin du corps de la page (juste avant </body>) → améliorer la vitesse de chargement de la page.

Images et css en provenance de différents sites peuvent être téléchargées simultanément (en parallèle) mais une fois que le navigateur a rencontré une balise script ce n'est (par défaut) plus possible → les scripts bloquent les téléchargements parallèles

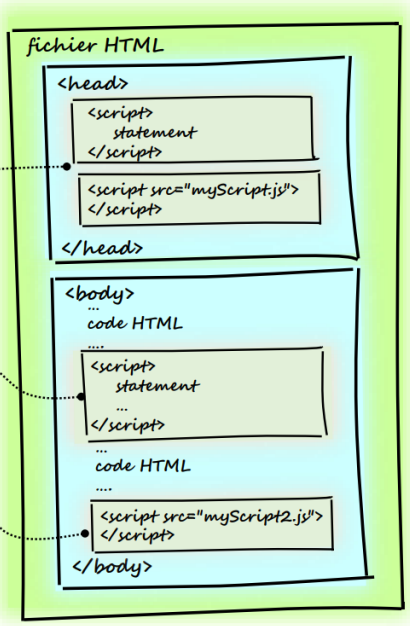

figure d'après "Head First HTML 5 programming"
Eric Freeman, Elisabeth Robson- Ed. O'Reilly, 2011

Fichier *JavaScript* Externe

Le JavaScript est généralement écrit dans des fichiers distincts avec l'extension **`.js`**.

Tout d'abord, créez un fichier séparé avec l'extension **`.js`**, par exemple **script.js**

Dans votre fichier HTML, utilisez la balise **<script>** dans la section **<head>** pour lier votre fichier JS externe :

```
<head>
    <meta charset="UTF-8">
    <title>Site de Voyage</title>
    <link rel="stylesheet" href="styles.css">
    <script src="script.js"></script>
</head>
```

Les commentaires

Tout comme en HTML et en CSS, dans le code JavaScript, vous pouvez écrire des commentaires qui ne seront pas interprétés par le navigateur. Ces commentaires sont destinés à fournir des explications aux autres développeurs sur le fonctionnement du code (et à vous-même, si vous revenez sur votre code après un certain temps et avez du mal à vous souvenir de ce que vous avez fait). Ils sont extrêmement utiles et devraient être utilisés régulièrement, surtout pour des applications de grande envergure. Il existe deux types de commentaires :

1. Commentaire sur une ligne

```
// Ceci est un commentaire
```

2. Commentaires sur plusieurs lignes

```
/*
  Ceci est un commentaire
  sur deux lignes
*/
```

Syntaxe du *JavaScript*

```
// Fonction: créer un nouveau paragraphe et l'ajouter en bas du HTML

function createParagraph() {
    let para = document.createElement("p");
    para.textContent = "Cliquez ICI !";
    document.body.appendChild(para);
  }

  /*
    1. Regrouper les coordonnées de tous les boutons de la page et les organiser en tableau
    2. Faire une boucle dans ce tableau et ajouter un "click event listener" à chaque bouton

    Quand le bouton est cliqué, la fonction "createParagraph()" sera exécutée
  */

  let buttons = document.querySelectorAll("button");

  for (let i = 0; i < buttons.length; i++) {
    buttons[i].addEventListener("click", createParagraph);
  }
```

Dans cet exemple, ce code JavaScript crée une fonction nommée **`createParagraph( )`** et associe cette fonction à tous les boutons présents dans la page HTML.

Les variables en JavaScript

Une variable est un espace alloué qui nous permet de stocker des informations de différents types, de manière temporaire.

Comment Déclarer une variable en Javascript?

Pour déclarer une variable en JavaScript on commence par le mot clef **var/let/const** suivi du **nom** de la variable. Une variable est constituée de deux parties : son **nom** et sa **valeur**.

```
const nom = 'jean'
let prenom = 'Herard'
var age = 12
```

*Il n'est pas nécessaire de préciser les fins de lignes en JavaScript c'est-à-dire de mettre des points virgules **(;)** à la fin de chaque ligne, mais il est recommandé de le faire malgré que les fins de lignes soient automatiquement détectées par l'interpréteur de JavaScript.*

Mot clef

Les variables sont des conteneurs pour stocker des données et peuvent être déclarées de 3 manières :

- En utilisant **var**
- En utilisant **let**
- En utilisant **const**

Quand utiliser var, let ou const ?

1. Déclarez toujours les variables

2. Toujours utiliser **const** si la valeur ne doit pas être modifiée

3. Toujours utiliser **const** si le type ne doit pas être modifié (Tableaux et Objets)

4. N'utilisez que **let** si vous ne pouvez pas l'utiliser **const**

5. À utiliser uniquement **var** si vous DEVEZ prendre en charge les anciens navigateurs.

Type de variable en JavaScript

On peut stocker différents types d'informations dans une variable comme:

- Les **nombres** comme par exemple : **42**
- Les **chaînes de caractères** comme par exemple : **'Une nouvelle chaîne'**
- Les **booléens** comme par exemple : **true**
- Les **tableaux** comme par exemple : **[' Paris ', ' Lille ']**
- Les **objets** comme par exemple : **{ age : 43 }**
- Il existe aussi le type **undefined**, un peu particulier, il permet de déterminer si une variable est définie ou non.
- On peut utiliser l'opérateur **typeof** pour savoir le type de données

Les nombres

```
let nb = -34
let nb1= 3.14
let nb2c = 65
```

- Des entiers **(int)** / **65 || -34**
- Des nombre réels **(float)** / **3.14**

Lorsqu' on manipule des réels, pour marquer la partie décimale on utilise un point.

Les chaînes de caractère

On utilise les chaînes de caractères pour stocker des mots ou des phrases. Pour indiquer à l'interpréteur qu'on est en train de travailler avec une chaîne de caractères, on entoure le texte par des guillemets simples **‘ ‘** ou des guillemets doubles **” “**.

```
var nom = "Denzel Washington"
let phrase = 'bonjour comment ca va?'
```

Si votre chaîne de caractère contient des **‘ ou “** , il faut les écharper.

```
let chaine1 = "C'est bon"
let chaine2 = 'C\'est bon aussi'
```

Les booléens

Un booléen est un type de variable à deux états qui nous permet de stocker une information qui peut être vraie **(true en anglais)** ou fausse **(false en anglais)**

```
let vrai = true
let faux = false
```

Les Tableaux [‘ ’]

Les tableaux permettent de stocker une liste d'informations.

```
var employe = ['Jacques', 'Herard', 'Pierre']
var test= [true, 34, 'Melo']
```

Les Objets { }

```
let eleve = {
  nom: 'Jacques',
  age: 22,
  notes: [12, 34, 20]
}
```

Un objet est une collection de données et/ou de fonctionnalités associées. Il contient habituellement différentes parties comme des variables et des actions (qu'on appelle propriétés et méthodes quand elles sont à l'intérieur de ces objets).

Les Opérateurs

Les opérateurs sont des symboles spéciaux utilisés pour effectuer des opérations sur des variables et des valeurs. En JavaScript, ils sont largement utilisés et comprennent plusieurs catégories.

Opérateurs Arithmétiques

Ces opérateurs sont utilisés pour effectuer des opérations mathématiques courantes.

- Addition (+)
- Soustraction (-)

Opérateur	Description	Exemple	Résultat
+	Addition	`5 + 2`	7
-	Soustraction	`5 - 2`	3
*	Multiplication	`5 * 2`	10
/	Division	`5 / 2`	2.5
%	Modulo (reste de division)	`5 % 2`	1
++	Incrémentation	`let a = 5; a++`	6
--	Décrémentation	`let b = 2; b--`	1

Opérateurs d'Affectation

Ils sont utilisés pour attribuer des valeurs à des variables JavaScript.

Opérateur	Description	Exemple	Résultat
=	Affectation	`let a = 10`	10
+=	Addition puis affectation	`a += 5`	15
-=	Soustraction puis affectation	`a -= 2`	13

Opérateurs de Comparaison

Ces opérateurs comparent deux valeurs et renvoient un booléen.

Opérateur	Description	Exemple	Résultat
==	Égalité	`5 == "5"`	true
===	Égalité stricte	`5 === "5"`	false
!=	Inégalité	`5 != "5"`	false
!==	Inégalité stricte	`5 !== "5"`	true
>	Supérieur	`5 > 2`	true
<	Inférieur	`5 < 2`	false

Opérateurs Logiques

Utilisés pour déterminer la logique entre des variables ou des valeurs.

Opérateur	Description	Exemple	Résultat
&&	ET logique	`(5 > 0 && 2 < 10)`	true
||	OU logique	`(5 > 10 || 2 < 10)`	true
!	NON logique	`!(5 > 10)`	true

Exercices Pratiques :

Exercice 1 : Manipulation des variables

Consigne :

- Déclarez une variable **age** et initialisez la avec votre âge.
- Ensuite, créez une autre variable **ageFutur** qui stockera votre âge dans 10 ans et affichez les deux variables dans la console.

```
let age = 25; // Remplacez par votre âge
let ageFutur = age + 10;

console.log("Âge actuel : " + age);
console.log("Âge dans 10 ans : " + ageFutur);
```

Exercice 2 : Calcul de l'âge en jours

Consigne :

- Déclarez une variable **age** avec votre âge actuel. -
- Ensuite, calculez et stockez dans une variable **ageEnJours** votre âge en jours (en supposant une année de 365 jours) et affichez-le dans la console.

Exercice 3 : Concaténation de chaînes de caractères

Consigne :

- Déclarez deux variables **prenom** et nom avec vos prénoms et nom.
- Ensuite, combinez-les dans une variable **nomComplet** pour former votre nom complet et affichez-le dans la console.

Exercice 4 : Utilisation de variables pour stocker des booléens

Consigne :

- Déclarez une variable **estFavori** et attribuez-lui la valeur booléenne **true** si vous aimez le chocolat, sinon attribuez-lui **false**.
- Ensuite, affichez dans la console un message disant si le chocolat est votre favori ou non.

(L'utilisation du if/else est nécessaire)

Les conditions en JavaScript

Contexte : Dans le développement web, la logique conditionnelle est un pilier fondamental. En JavaScript, les structures conditionnelles permettent à nos programmes de prendre des décisions basées sur des critères spécifiques.Que ce soit pour valider des données utilisateur, gérer des flux de navigation, ou exécuter des calculs complexes, les conditions sont omniprésentes et essentielles.

Les structures conditionnelles permettent à votre programme de prendre des décisions en fonction de certaines conditions. Voici les principales structures conditionnelles en JavaScript :

Instruction { if }

L'instruction **if** est la plus simple des structures conditionnelles. Elle exécute un bloc de code si une condition spécifiée est vraie.

```
if (condition) {
    // bloc de code à exécuter si la condition est vraie
}
```

Exemple :

Supposons que l'on souhaite afficher la catégorie d'une personne à un utilisateur en fonction de son âge, en s'assurant que la variable **"age"** est déclarée.

```
let age = 18
if (age == 18) {
    "Il est Majeur!"
}
```

Instruction { else }

```
if (condition) {
    // bloc de code si la condition est vraie
} else {
    // bloc de code si la condition est fausse
}
```

L'instruction **else** est utilisée avec **if** pour exécuter un bloc de code lorsque la condition if est fausse.

Exemple :

Maintenant, on souhaite afficher un autre message à l'utilisateur dans le cas où l'âge de la personne n'est pas **égal à 18**, c'est-à-dire lorsque l'expression est fausse.

```
let age = 15

if (age == 18 ) {
    "Il est Majeur!"
} else {
    "Il est Mineur"
}
```

Instruction { else if }

L'instruction **else if** permet de tester plusieurs conditions.

```
if (condition1) {
    // bloc de code si condition1 est vraie
} else if (condition2) {
    // bloc de code si condition2 est vraie
} else {
    // bloc de code si aucune des conditions précédentes n'est vraie
}
```

Exemple :

En reprenant le même exemple, si l'on souhaite afficher la catégorie d'une personne pour chaque tranche d'âge.

```
let age = 18

if (age == 18) {
    "Il est Majeur!"
} else if (age > 18) {
    "Il est Adulte!"
} else {
   "Il est mineur!"
}
```

Opérateur de comparaison

Lorsqu'on souhaite créer des conditions en JavaScript ou tester la valeur d'une variable, on utilise ce qu'on appelle les opérateurs de comparaison.

Exemple :

Parfois, on souhaite combiner des conditions pour obtenir un résultat spécifique, par exemple : autoriser uniquement les filles de 20 ans dans un restaurant. Cela requiert l'utilisation des opérateurs booléens.

```
let age = 18
var sexe = 'F'

if (age == 18 && sexe == 'F') {
    "Vous avez accès!"
} else if (age > 18) {
    "Accès refuse!"
}
```

```
a == b // a égale à b
a === b // a == b et a est de même "type" que b
a >= b // a supérieur OU égal à b
a > b   // a strictement supérieur à b
a <= b // a inférieur OU égal à b
a < b   // a strictement inférieur à b
```

Opérateur Ternaire

L'opérateur ternaire est une forme abrégée de l'instruction **if-else**.

```
// condition ? <instruction a executer si vrai> : <instruction a executer si faux>

sexe= 'M'
"Je suis " + (sexe == 'M'? "Homme" : "Femme")
```

Instruction { switch }

Le **switch case** permet d'exécuter une opération en fonction de la valeur de l'expression en paramètre. Il peut parfois remplacer une série de **if** et **else**. On y a recours lorsque de nombreux cas doivent être gérés.

L'instruction **switch** est donc utilisée pour effectuer différentes actions en fonction de différentes conditions.

```
switch (expression) {
    case valeur1:
        // Instruction à exécuter si le résultat de l'expression correspond à la valeur1
        break;
    case valeur2:
        // Instruction à exécuter si le résultat de l'expression correspond à la valeur2
        break;
    default:
        // Instruction par défaut si les valeurs ne correspondent à aucune des valeurs attendues
        break;
}
```

Exemple :

Pour un petit calendrier qui afficherait un mois si la valeur d'une case correspond à la valeur de la variable **"mois"** passée en paramètre.

```
let mois = 2

        switch (mois) {
            case 1:
                "Janvier"
                break
            case 2:
                "Fevrier"
                break
            default:
                "Ce Mois n'existe pas"
                break
        }
```

Conclusion : Les structures conditionnelles sont essentielles pour contrôler le flux d'exécution de votre programme en JavaScript. Elles permettent à votre code de réagir différemment selon les données et les situations, rendant vos applications plus dynamiques et interactives.

Quelques opérateurs booléens

```
// || OU

true || true // true
true || false // true
false || true // true
false || false // false
```

```
// && ET

true && true // true
true && false // false
false && true // false
false && false // false
```

Exercices sur les Structures Conditionnelles en JavaScript

Exercice 1: Vérification de l'Âge

Écrivez un programme JavaScript qui demande l'âge de l'utilisateur.
- Si l'utilisateur a 18 ans ou plus, affichez "Accès autorisé"
- sinon affichez "Accès refusé".

(L'utilisation de la fonction prompt() est nécessaire)

Exercice 2: Classification des Notes

Écrivez un programme qui classe une note en "Excellent", "Bien", "Suffisant", ou "Insuffisant" selon les critères suivants:
- 0-50 Insuffisant
- 51-70 Suffisant
- 71-90 Bien
- 91-100 Excellent.

(L'utilisation du if/else - else if est nécessaire)

Exercice 3: Jour de la Semaine

Énoncé: Écrivez un programme qui, en fonction d'un numéro (1-7), affiche le jour de la semaine correspondant (1 pour Lundi, 7 pour Dimanche).

Exercice 4: Calculatrice Simple

Créez une calculatrice simple qui prend deux nombres et une opération (addition, soustraction, multiplication, division) et affiche le résultat.

Exercice 5: Gestion d'un Menu de Restaurant

Vous développez une application pour un restaurant qui propose différents plats. Le menu change en fonction du jour de la semaine. De plus, le restaurant offre une réduction spéciale pour les étudiants. Votre tâche est de créer un programme qui, en fonction du jour de la semaine et du statut de l'utilisateur (étudiant ou non), affiche le plat du jour et le prix après réduction si applicable.

Instructions
- Utilisez une structure switch pour sélectionner le plat du jour en fonction du jour de la semaine.
- Utilisez une structure if pour appliquer une réduction de 20% sur le prix du plat si l'utilisateur est un étudiant.
- Affichez le plat du jour et le prix final.

Données :
Jours de la semaine: Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche.

Introduction à JavaScript

Plats du jour:

Lundi: Pizza - 10€
Mardi: Pasta - 8€
Mercredi: Burger - 9€
Jeudi: Salade - 7€
Vendredi: Poisson - 12€
Samedi et Dimanche: Spécial Week-end - 15€

Réduction étudiante: 20%

Tâche pour l'Utilisateur :

Modifiez les variables **jour** et **estEtudiant** pour tester différents scénarios. Vérifiez si le programme affiche correctement le plat du jour et le prix après réduction pour les étudiants.

```
let jour = "Mercredi"; // Jour actuel
let estEtudiant = true; // Statut de l'utilisateur
let plat, prix;
```