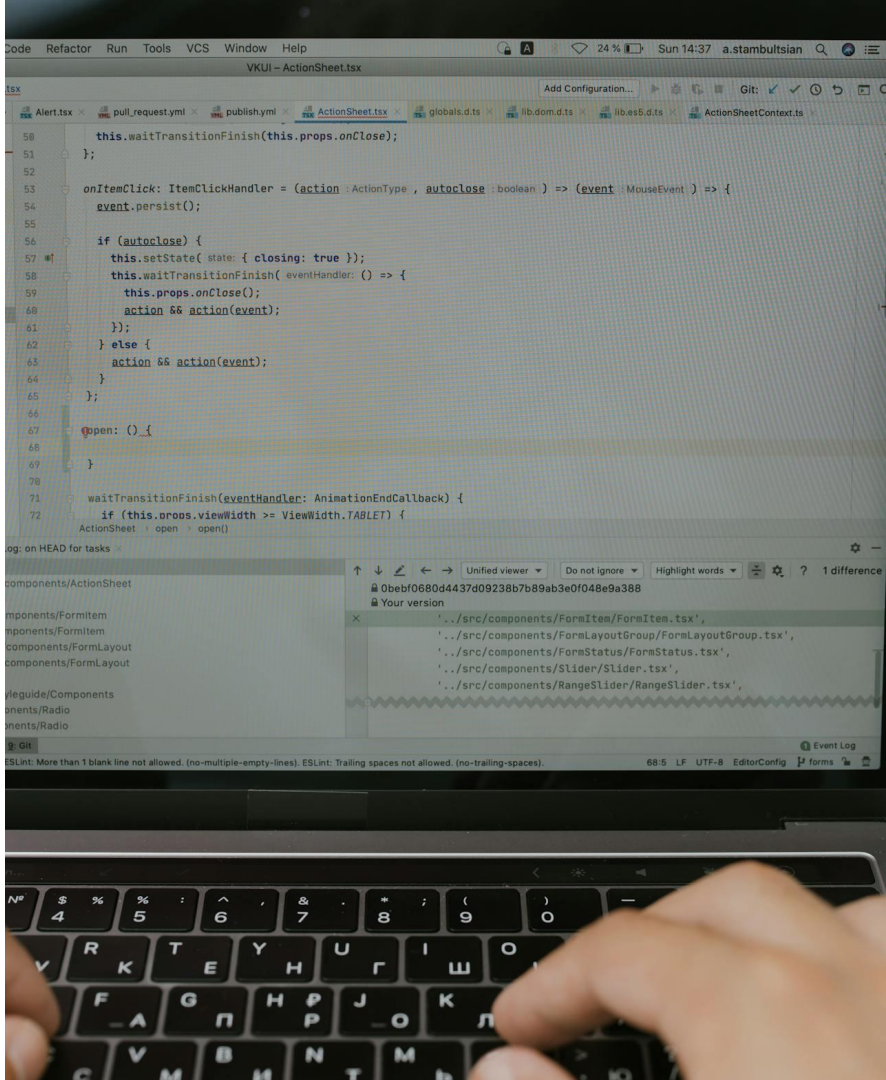
Javascript, Web API et JSON

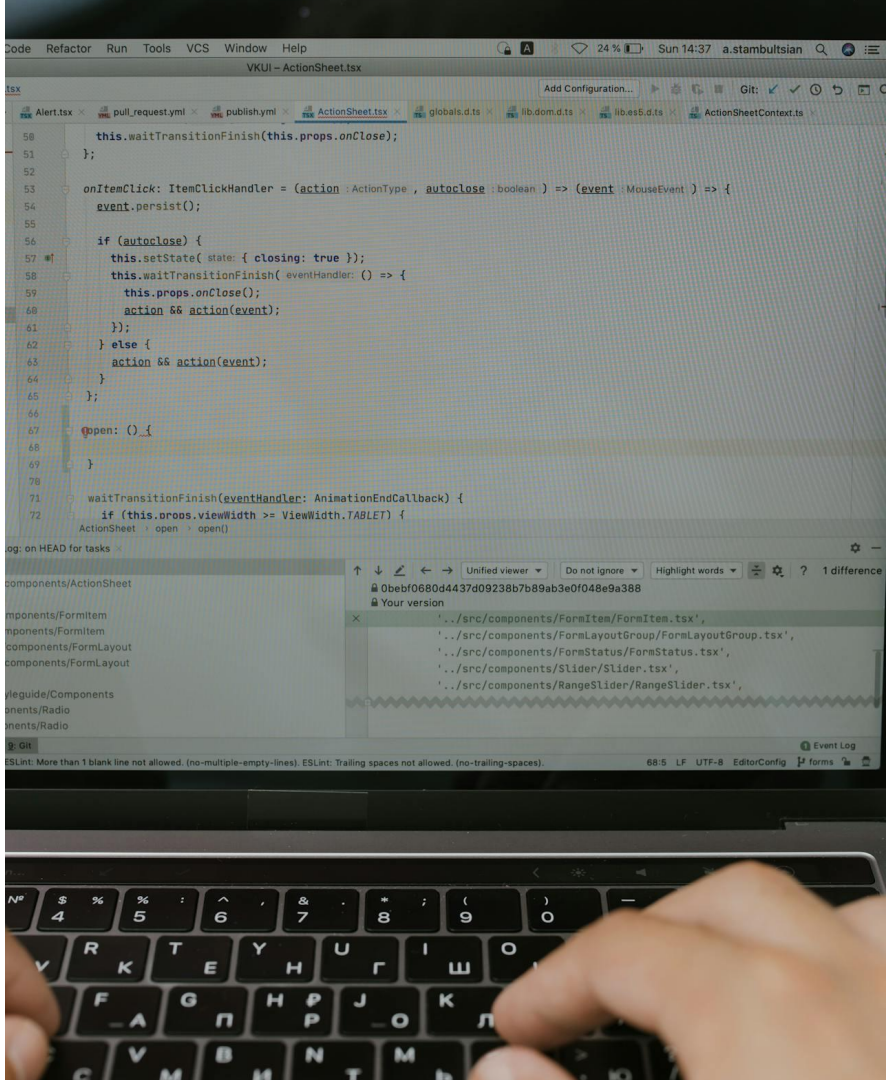
Partie 1 - Javascript

Valentin RIBEZZI



Sommaire

I. Présentation du Javascript
II. Outils pour le développement
III. Syntaxe et structures de base
    A. *Variables, types de données et opérateurs*
    B. *Structures de contrôle*
    C. *Fonctions et modules*
IV. Interaction avec une page web
V. Les événements
VI. POO en Javascript
VII. Frameworks



01

Javascript, Web API et JSON

Partie 1 - Javascript

Présentation du Javascript

Présentation du Javascript

JavaScript est un langage de programmation, initialement côté client, associé au langage de structuration HTML. Javascript est un langage interprété, il n'est pas compilé et doit être exécuté comme un script par un "interprète".
Pour rappel, le langage compilé quant à lui est transformé en code machine lors de sa compilation. Très souvent, le résultat de cette compilation est la création d'un fichier dit "exécutable".

Le Javascript permet d'ajouter des fonctionnalités interactives aux pages webs comme modifier la structure HTML dynamiquement, modifier du code CSS, gérer des événements ou encore faire des redirections.

Cette technologie est intégré nativement dans les navigateurs, tout comme l'HTML ou encore le CSS.

02

Javascript, Web API et JSON

Partie 1 - Javascript

Outils pour le développement

Outils pour le développement

Un éditeur de texte ou IDE

Il servira à écrire les codes Javascript qui viendront s'associer à nos fichiers HTML et CSS

Un navigateur Web

Il servira à exécuter et à débugger notre code JavaScript. Tous les navigateurs permettent de le faire.

03

Javascript, Web API et JSON

Partie 1 - Javascript

Syntaxe et structures de base

Syntaxe et structures de base

Variables, types de données et opérateurs

Avant toute chose, comment utilisons-nous le langage Javascript pour l'associer à des fichiers HTML ?

- **Utilisation d'une balise `<head>` à mettre dans le `<head>` de votre page HTML. Mais de nos jours, il est plus correct de mettre cette balise à la fin du fichier afin de charger vos scripts après le chargement de votre page HTML.**
- **Création d'un fichier indépendant à importer dans votre fichier HTML pour que les deux soient "connectés" :**

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

Syntaxe et structures de base

Variables, types de données et opérateurs

Voyons maintenant comment écrire du Javascript !

Syntaxe et structures de base

Variables, types de données et opérateurs

Il existe trois types de variables en Javascript.

```
var uneVariable = 30;

let uneAutreVariable = "Boujour";

const UNE_DERNIERE_VARIABLE = new Array(8, 4, 5);
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Il existe trois types de variables en Javascript.

```
var uneVariable = 30;
```

La portée de cette variable sera limitée à la fonction ou la classe dans laquelle elle est déclarée.

```
let uneAutreVariable = "Boujour";
```

La portée de cette variable sera limitée au "bloc" laquelle elle est déclarée.

```
const UNE_DERNIERE_VARIABLE = new Array(8, 4, 5);
```

La portée de cette variable est équivalente à "let" mais sera une constante, plutôt qu'une variable. Ainsi, son contenu ne pourra pas être altéré lors de l'exécution du code.

Syntaxe et structures de base

Variables, types de données et opérateurs

Javascript a la particularité de ne pas être un langage dit “typé”, c’est-à-dire qu’une variable initialisé comme un entier

```
var uneVariable = 30;
```

peut totalement devenir une chaîne de caractères si on choisit de la faire ainsi.

```
var uneVariable = “Boujour”;
```

Syntaxe et structures de base

Variables, types de données et opérateurs

Pour connaître le type d'une variable, il est possible d'utiliser la méthode **typeof**().

Syntaxe et structures de base

Variables, types de données et opérateurs

Les opérateurs

Opérateurs	Descriptions
+, -, *, /, %, =	Opérateurs de base
==, <, >, <=, >=, !==	Opérateurs de comparaison (hors type)
===, !===	Opérateurs de comparaison (avec type)
+=, -=, *=, \=, %=	Opérateurs associatifs
&&, ||, !	Opérateurs logiques (AND, OR et NOT)
[VARIABLE]++, [VARIABLE]--	Opérateurs d'incrémentation et de décrémentation
+, .	Opérateurs de concaténation

Syntaxe et structures de base

Structures de contrôle

Les conditions

En effet, il existe des structures en Javascript qui permettent de tester et d'exécuter du code en fonction d'une ou plusieurs conditions spécifiques.

```
if (true) titi = "Bonjour";
```

- Pour une instruction

```
if (true) {
    ...;
}
```

- Pour une instruction mais avec plus de "lisibilité"

```
if (true) {
    ...;
} else if {
    ...;
} else {
    ...;
}
```

- Pour plusieurs instructions

Syntaxe et structures de base

Structures de contrôle

Les conditions (Opérateur ternaire)

Il existe également une structure qui permet de réaliser une condition de manière moins “verbeuses” qu’un “IF” mais qui est peut être plus dur à maintenir si utilisé trop souvent.

```
(42) ? console.log("42 est vrai") : console.log("42 est faux");
```

Syntaxe et structures de base

Structures de contrôle

Les conditions (Switch)

Parfois, il est trop compliqué d'utiliser des "IF" de part le nombre et la complexité des tests à réaliser. Dans ces cas là, nous pourrons utiliser l'expression "SWITCH" qui permet de réaliser de multiples conditions en partant d'une expressions de base.

```
const age = 20;
switch (age) {
   case 10:
      console.log("Tu as 10 ans");
      break;
   case 20:
      console.log("Tu as 20 ans");
      break;
   default:
      console.log("Tu as un autre âge");
}
```

Pour le **break**, nous en reparlons dans les boucles !

Syntaxe et structures de base

Structures de contrôle

Les boucles

Comme leurs noms l'indiquent, cela servira à faire des traitement de base par rapport à une condition ou une itération. Il existe plusieurs types de boucles.

La boucle While

```
let i = 0;
while (i < 5) {
   console.log(i);
   i++;
}
```

La boucle Do While

```
let j = 20;
do {
   console.log(j);
   j--;
} while (j > 15);
```

Syntaxe et structures de base

Structures de contrôle

Les boucles

La boucle For

```
for (let k = 0; k < 5; k++) {
   console.log(k);
}
```

La boucle For (Avec In)

```
let tableau = [1, 2, 3, 4, 5];
for (let element of tableau) {
   console.log(element);
}
```

Si vous voulez altérer avec le déroulement d'une boucle, il existe deux instructions pour cela :

- **break** : Pour sortir de la boucle lorsque l'instruction est atteinte
- **continue** : Pour passer à l'itération suivante sans finir d'exécuter l'instruction courante

Syntaxe et structures de base

Fonctions et modules

Les fonctions

En JavaScript, les fonctions sont des blocs de code conçus pour effectuer une tâche particulière. Elles sont essentielles pour structurer le code, le rendre réutilisable et facile à maintenir.

Il existe plusieurs types de fonctions :

- Fonctions de base (Instruction de fonctions)
- Expressions de fonctions
- Fonctions anonymes
- Fonctions fléchées
- Fonctions “callback”
- Fonctions auto-exécutables

Syntaxe et structures de base

Fonctions et modules

Les fonctions

Fonctions de base (Instruction de fonctions)

```
function maFontion(){
   console.log("Ma fonction");
}

function maFonctionAvecParametres(param1, param2){
   console.log(param1, param2);
}

maFontion();
maFonctionAvecParametres("Bonjour", 42);
```

Syntaxe et structures de base

Fonctions et modules

Les fonctions

Expressions de fonctions

```
let maNouvelleFonction = function laFonctionDeMaNouvelleFonction(){
   console.log("Ma nouvelle fonction");
}
let resultat = maNouvelleFonction();
```

Fonctions anonymes

```
let isNombrePair = function(nombre){
   return nombre % 2 === 0;
}
console.log(isNombrePair(42));
```

Syntaxe et structures de base

Fonctions et modules

Les fonctions

Fonctions fléchées

```
const addition = (a, b) => a + b;
console.log(addition(2, 3));
```

Fonctions “callback”

```
function traitement(callback){
   console.log("Je suis un traitement");
   callback();
}

traitement(function(){
   console.log("Je suis un callback");
});
```

Les fonctions de rappel sont des fonctions passées en argument à une autre fonction et exécutées après que cette dernière a terminé son exécution.

Syntaxe et structures de base

Fonctions et modules

Les fonctions

Fonctions auto-exécutables

```
(function(){
    console.log("Cette fonction s'exécute immédiatement !");
})();
```

04

Javascript, Web API et JSON

Partie 1 - Javascript

Interaction avec une page web

Interaction avec une page web

Le Javascript permet de faire de l'algorithme mais surtout de pouvoir, comme nous en avons parlé, de pouvoir interagir avec les éléments d’une page web, ce qu’on appelle le **DOM** !

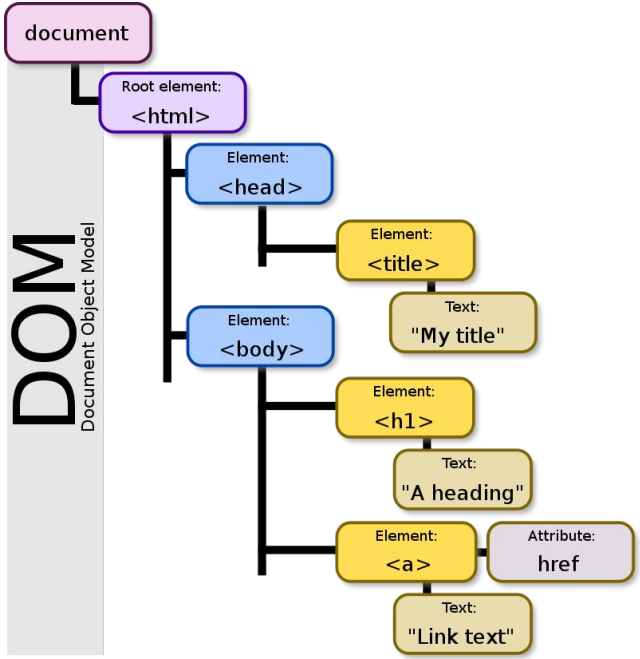
Interaction avec une page web

Qu’est ce que le **DOM** ?

Le DOM est une interface de programmation pour les documents HTML et XML. Il représente la structure de la page comme un arbre de nœuds, où chaque nœud est un objet représentant une partie du document.

Interaction avec une page web

Qu'est ce que le **DOM** ?



Interaction avec une page web

Voyons maintenant comment accéder aux différents éléments du DOM.

```
// Accéder à un élément par son ID
let element = document.getElementById("monElement");

// Accéder à plusieurs éléments par leur classe
let elements1 = document.getElementsByClassName("maClasse");

// Accéder à plusieurs éléments par leur tag (balise)
let elements2 = document.getElementsByTagName("div");

// Accéder à un ou plusieurs éléments en utilisant un sélecteur CSS
let element2 = document.querySelector(".maClasse"); // Premier élément correspondant
let elements3 = document.querySelectorAll(".maClasse"); // Tous les éléments correspondants
```

Interaction avec une page web

Une fois les éléments ciblés, il est possible de modifier le contenu et les attributs des éléments.

```
// Modfier le contenu HTML d'un élément
element.textContent = "Bonjour";
element.innerText += " !"; // On ajoute un point d'exclamation

// Modifier le HTML d'un élément
element.innerHTML = "<strong>Bonjour !</strong>";

// Modifier le style d'un élément
element.style.color = "red";

// Modifier les attributs d'un élément
element.setAttribute("data-id", 42);
```

Interaction avec une page web

Et pour finir, il est, bien entendu, possible de créer et ajouter des éléments qui ne sont pas présent directement le DOM !

```
// Créer un nouvel élément
let nouveauElement = document.createElement("div");
nouveauElement.textContent = "Je suis un nouveau div";

// Ajouter un élément à un parent existant
let parent = document.getElementById("parent");
parent.appendChild(nouveauElement);
```

05

Javascript, Web API et JSON

Partie 1 - Javascript

Les événements

Les événements

Les événements sont des actions déclenchées par l'utilisateur ou le navigateur, comme des clics de souris, des soumissions de formulaires, ou des chargements de pages. Vous pouvez utiliser JavaScript pour écouter et répondre à ces événements.

Pour réagir aux événements, vous devez ajouter des écouteurs d'événements aux éléments du DOM.

```
// Ajouter un écouteur d'événement pour un clic
element.addEventListener("click", function() {
  alert("Élément cliqué !");
});

// Ajouter un écouteur d'événement pour le chargement de la page
window.addEventListener("load", function() {
  console.log("Page chargée !");
});
```

Les événements

Lors de la gestion d'un événement, on reçoit un "objet" événement qui contient des informations sur celui-ci. Il est possible d'utiliser cet objet pour effectuer des actions en fonction de son état.

```
element.addEventListener("click", function(event) {
    console.log("Position de la souris : ", event.clientX, event.clientY);
});
```

06

Javascript, Web API et JSON

Partie 1 - Javascript

POO en Javascript

POO en Javascript

Comme pour la plupart des langages de programmation, le Javascript permet également de réaliser de la programmation orientée objet (POO).

Cela dit, comme nous allons le voir, le Javascript est assez permissif sur la POO et ce n'est pas le langage le plus adapté pour faire cela, même si c'est tout à fait possible.

POO en Javascript

```
class Personne {
    constructor(nom, age, ville){
        this.nom = nom;
        this.age = age;
        this.ville = ville;
    }
    saluer(){
        console.log(`Bonjour, je m'appelle ${this.nom} et j'ai ${this.age} ans.`);
    }
}

let jean = new Personne("Jean", 30, "Paris");
let marc = new Personne("Marc", 25, "Lyon");
console.log(jean.nom); // Jean
console.log(marc.nom); // Marc
jean.saluer(); // Bonjour, je m'appelle Jean et j'ai 30 ans.
marc.saluer(); // Bonjour, je m'appelle Marc et j'ai 25 ans.
```

POO en Javascript

```
class Developpeur extends Personne {
    constructor(nom, age, ville, langage){
        super(nom, age, ville);
        this.langage = langage;
    }
    coder(){
        console.log(`Je code en ${this.langage}`);
    }
}
jean = new Developpeur("Jean", 30, "Paris", "JavaScript");
jean.saluer(); // Bonjour, je m'appelle Jean et j'ai 30 ans.
jean.coder(); // Je code en JavaScript
```

07

Javascript, Web API et JSON

Partie 1 - Javascript

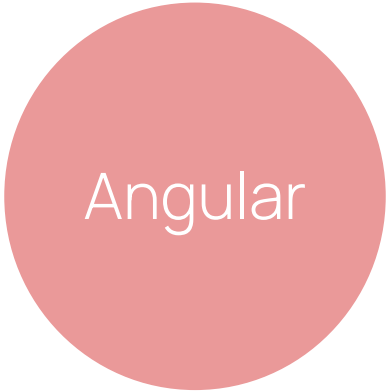
Frameworks

Frameworks

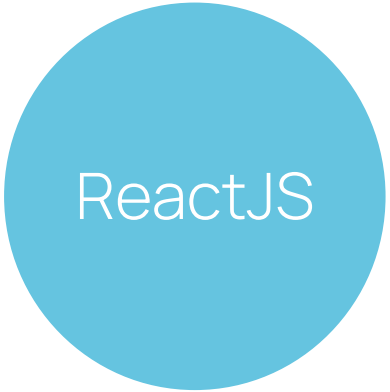
Les frameworks JavaScript facilitent le développement d'applications web en fournissant des structures et des outils préconstruits.

Frameworks

Vue.js est un framework JavaScript simple et flexible qui permet de construire des interfaces utilisateur interactives en utilisant des composants réutilisables. Il est idéal pour démarrer rapidement avec des petits projets et peut évoluer pour des applications plus complexes.



Angular est un framework complet pour construire des applications web dynamiques. Utilisant TypeScript, il suit une architecture bien structurée et est parfait pour les grandes applications d'entreprise grâce à ses fonctionnalités robustes et à son support intégré.



React est une bibliothèque JavaScript créée par Facebook pour construire des interfaces utilisateur. Elle se concentre sur la création de composants réutilisables et rend le développement de grandes applications web plus facile en gérant efficacement la mise à jour de l'interface utilisateur lorsque les données changent.