

Javascript Avancé

1WEBD – Javascript Web Development



Sommaire

1. Patterns Avancés.
2. Typescript.
3. Webpack.
4. Quatrième partie.



1. Patterns Avancés

1. Patterns Avancés

Déstructuration

- **Définition:** Un moyen expressif et compact de 'décomposer' les objets et les tableaux
- **Objectif:** Simplifier l'extraction de données de structures complexes

1. Patterns Avancés

Déstructuration sur Objet

- Comment extraire les propriétés **a** et **b** de **objet**

```
const objet = { a: 1, b: 2, c: 3 };  
const { a, b } = objet;  
console.log(a, b); // Affiche 1 2
```

1. Patterns Avancés

Déstructuration sur Tableau

- Sélection des deux premiers éléments du tableau

```
const tableau = [1, 2, 3, 4];  
const [premier, deuxieme] = tableau;  
console.log(premier, deuxieme); // Affiche 1 2
```

1. Patterns Avancés

Paramètre Rest

- **Définition:** Permet de représenter un nombre indéterminé d'arguments sous forme d'un tableau
- **Utilisation:** Principalement dans les fonctions pour gérer des arguments variables

1. Patterns Avancés

Paramètre Rest

- **...nombres** regroupe tous les arguments en un tableau **nombres**

```
function somme(...nombres) {  
  return nombres.reduce((acc, actuel) => acc + actuel, 0);  
}  
console.log(somme(1, 2, 3)); // Affiche 6
```


1. Patterns Avancés

Spread Operator

- **Définition:** Permet de 'dispenser' les éléments d'un tableau ou d'un objet
- **Utilisation:** Utile pour la copie, la concaténation de tableaux, ou la combinaison d'objets

1. Patterns Avancés

Spread Operator sur Objet

- Création d'un nouvel objet **combineObj** en combinant **obj1** et **obj2**
- Si une clé existe sur tous les objet, c'est la valeur du dernier dans la chaine qui est utilisé

```
const obj1 = { a: 1, b: 2 };
const obj2 = { c: 3, d: 4 };
const combineObj = {...obj1, ...obj2};
console.log(combineObj); // Affiche { a: 1, b: 2, c: 3, d: 4 }
```

1. Patterns Avancés

Spread Operator sur Tableau

- Fusion de **arr1** et **arr2** en un nouveau tableau **combine**

```
const arr1 = [1, 2];  
const arr2 = [3, 4];  
const combine = [...arr1, ...arr2];  
console.log(combine); // Affiche [1, 2, 3, 4]
```

1. Patterns Avancés

Null Coalescing Operator (??)

- **Définition:** Un opérateur logique qui renvoie son opérande de droite lorsque celui de gauche est **null** ou **undefined**, sinon renvoie l'opérande de gauche
- **Utilité:** Permet de définir des valeurs par défaut de manière concise

1. Patterns Avancés

Null Coalescing Operator (??)

- Utilisation de ?? pour assigner une valeur par défaut

```
const valeur = null;  
const parDefaut = valeur ?? "valeur par défaut";  
console.log(parDefaut); // Affiche "valeur par défaut"
```

1. Patterns Avancés

Null Coalescing Operator (??) - Comparaison avec l'operator OR (||)

- `||` considère **0** comme **falsy** (qui peut être considéré **false**), tandis que `??` ne réagit qu'à **null** ou **undefined**

```
const valeurFausse = 0;
const parDefautOR = valeurFausse || "valeur par défaut";
console.log(parDefautOR); // Affiche "valeur par défaut"
```

1. Patterns Avancés

Optional Chaining Operator (?.)

- **Définition:** Permet d'accéder à des propriétés profondément imbriquées d'un objet sans avoir besoin de valider chaque référence intermédiaire
- **Utilité:** Réduit le risque d'erreurs en accédant à des propriétés d'objets potentiellement nuls ou indéfinis

1. Patterns Avancés

Optional Chaining Operator (?.)

- Accès sécurisé à **objet.a.b.c** même si **a** ou **b** est **null** ou **undefined**



```
const objet = { a: { b: { c: 1 } } };  
const valeur = objet.a?.b?.c;  
console.log(valeur); // Affiche 1
```


1. Patterns Avancés

Optional Chaining Operator (?.)

- Si **maFonction** existe, elle est appelée, sinon **resultat** est **undefined**



```
const objet = { maFonction: () => "Hello World" };  
const resultat = objet.maFonction?.();  
console.log(resultat); // Affiche "Hello World"
```

1. Patterns Avancés



2. Typescript

2. Typescript

Introduction

- **Définition:** Un sur-ensemble typé de JavaScript développé initialement par Microsoft
- **Objectif:** Ajouter des types statiques pour améliorer la lisibilité et la robustesse du code

2. Typescript

Avantages

- Amélioration de la maintenabilité et de la lisibilité du code
- Prévention des erreurs courantes à l'exécution
- Assistance dans le développement avec des outils d'auto-complétion et de refactoring

2. Typescript

Inconvénients

- Nécessite une étape de compilation (les navigateurs ne comprennent pas le Typescript)
- Rajoute des regles qui pourraient ralentir le développement (c'est néanmoins un gain de temps sur le moyen/long terme)

2. Typescript

Types

- **Types Primitifs:** *string, number, boolean*
- **Type *any*:** Pour les valeurs de type inconnu (à éviter le plus possible, sinon autant ne pas utiliser Typescript)
- **Arrays et Tuples:** *number[], [string, number]*
- **Enum:** Amélioration de la lisibilité des ensembles de valeurs constantes

2. Typescript

Types

- Déclaration de variables avec des types spécifiques

```
let nom: string = "Alice";  
let age: number = 30;  
let estActif: boolean = true;
```


2. Typescript

Interfaces

- **Définition:** Un moyen de définir la forme d'un objet.
- - ****Utilité:**** Permet de décrire des contrats de structure pour les objets

2. Typescript

Interfaces

- Création et utilisation d'une interface **Personne**

```
interface Personne {  
    nom: string;  
    age: number;  
}  
const employe: Personne = { nom: "Bob", age: 25 };
```

2. Typescript

Depuis Javascript

- **Approche:** Commencer avec du code JavaScript existant
- **Étapes:**
 1. Ajouter des types aux variables et fonctions.
 2. Utiliser des interfaces pour structurer les objets.
 3. Compiler et résoudre les erreurs TypeScript.

2. Typescript

Depuis Javascript

```
function saluer(personne) {  
    return "Bonjour, " + personne.nom;  
}
```

```
function saluer(personne: Personne): string {  
    return "Bonjour, " + personne.nom;  
}
```

2. Typescript



3. Webpack

3. Webpack

Introduction

- **Définition:** Un module bundler rassemble les fichiers de l'application et les combine en un seul (généralement)
- **Utilité:** Un temps de chargement plus rapide et limité au code nécessaire

3. Webpack

Fichier de configuration

- Le fichier **webpack.config.js** est généralement stocké à la racine du projet (à côté du package.json)
- il est composé de deux parties obligatoires:
 - entry: définit le point d'entrée de l'application pour webpack
 - output: définit le chemin et le nom du fichier « compilé » (donc où webpack va écrire son fichier)
- Pour plus d'information, le site webpack.js.org

3. Webpack

Fichier de configuration

```
const path = require('path');

module.exports = {
  entry: './path/to/my/entry/file.js',
  output: {
    path: path.resolve(__dirname, 'dist'),
    filename: 'my-first-webpack.bundle.js',
  },
};
```

3. Webpack

Webpack et Typescript

- Webpack ne supporte pas Typescript nativement
- Faut rajouter explicitement les fichiers **.ts** et **.tsx** dans les extensions autorisées
- Ensuite on rajoute une regle pour dire a webpack d'utiliser **ts-loader** (a installer séparément de webpack) traiter les fichiers Typescript

3. Webpack

Webpack et Typescript

```
module.exports = {
  entry: './src/index.ts',
  module: {
    rules: [
      {
        test: /\.tsx?$/,
        use: 'ts-loader',
        exclude: /node_modules/
      }
    ]
  },
  resolve: {
    extensions: ['.tsx', '.ts', '.js']
  },
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  }
};
```

3. Webpack

Etapes dans un Projet

- Initialiser un nouveau projet
- Créer un dossier pour son Javascript/Typescript (généralement appelé src)
- Créer un dossier de sortie pour le projet (généralement appelé dist ou build)
- Installer webpack et webpack-cli en dépendance dev (donc pas rajouté dans le bundle final)
- Créer son fichier index.html dans le dossier d'output
- Créer un fichier js/ts dans le dossier src



```
npm install webpack webpack-cli --save-dev
```

3. Webpack

- **Etapes dans un Projet**
 - Créer un fichier de config webpack (webpack.config.js)
 - Ajouter un script build dans le package.json qui exécutera webpack

```
{
  "name": "webpack_demo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "webpack": "^5.89.0",
    "webpack-cli": "^5.1.4"
  }
}
```

3. Webpack

Etapas dans un Projet

- Inclure son script dans les fichiers html

3. Webpack

Etapas supplémentaires

- Générer un .js différent pour chaque page pour que chacune ne contienne que le code nécessaire à son fonctionnement

3. Webpack



