

Javascript - Introduction

1WEBD – Javascript Web Development



Sommaire

1. Histoire et Evolution.
2. Implémentation.
3. Variables et Déclarations.
4. Types de Variables.



1. Histoire et Evolution

1. Histoire et Evolution

Histoire

- Développé en 1995 en 10 jours chez Netscape par Brendan Eich
- Conçu pour rendre le web plus interactif
- Devrait être exécuté côté client
- Devrait être facile à apprendre

1. Histoire et Evolution

Evolution

- Initialement nommé Livescript
- Changé en Javascript pour se rapprocher de Java
- Présenté comme un complément à Java
- Publié en mars 1996 dans Netscape 2.0
- Microsoft crée JScript en août 1996 pour concurrencer Netscape

1. Histoire et Evolution

Evolution - ECMAScript

- Javascript est standardisé en 1997 (ECMAScript)
- Nouveaux standards tous les ans

1. Histoire et Evolution

Evolution - ECMAScript - ES5

- Publié en 2009
- Manipulation de tableaux
- Support pour JSON
- Mode strict

1. Histoire et Evolution

Evolution - ECMAScript – ES6

- Publié en 2015
- Classes
- Modules
- Promesses
- Fonctions fléchées

1. Histoire et Evolution



2. Implémentation

2. Implémentation

Balise

- Javascript s'integre dans une page avec une balise script
- Peut etre placée dans le **<head>** ou en fin de **<body>**
- Peut bloquer la page (possible de eviter cette situation avec l'attribut **defer**)

2. Implémentation

Balise



```
<script>  
  alert("hello world")  
</script>
```

2. Implémentation

Balise



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

2. Implémentation



3. Variables et Déclarations

3. Variables et Déclarations

Conventions

- JS s'écrit en camelCase (uneVariable)
- Case-Sensitive (sensible à la casse): mavariable != maVariable
- Utiliser des noms de variables descriptifs (n'est pas seulement valide en JS)

3. Variables et Déclarations

Déclaration de Variables

- Peuvent être déclarée avec **const**, **let** ou **var**



```
const|let|var toto = 42
```

3. Variables et Déclarations

Déclaration de Variables – Const

- Introduit dans ES6
- Une variable **const** ne peut être réassigné
- Un tableau ou un objet peuvent être modifiée même en étant **const**

3. Variables et Déclarations

Déclaration de Variables – Let

- Introduit dans ES6
- Peut être réassigné
- Est déclaré pour son bloc (cf slide sur le **hoisting**)

3. Variables et Déclarations

Déclaration de Variables – Var

- Première façon (historiquement) de déclarer des variables en JS
- Susceptible au **hoisting**
- A éviter

3. Variables et Déclarations

Déclaration de Variables – Hoisting

- Décrit la façon dont Javascript traite les déclarations de variables et de fonction
- Quand un script est exécuté:
 - Les déclarations de variables déclarées avec **var** sont hissées (**hoisted**) au sommet de leur scope (condition, fonction etc)
- Ce n'est pas forcément le cas de leur initialisation
- Une fonction est hissée au complet (définition et fonctionnalité)
- La variable peut donc être utilisée avant son initialisation (sa valeur sera donc **undefined**)

3. Variables et Déclarations

Déclaration de Variables – Hoisting

- Les variables déclarées avec **let** et **const** ne subissent pas le "**hoisting**" (elles sont en "**temporal dead zone**")
- Les expressions de fonction sont hissées selon qu'elles soient déclarées en **var**, **let** ou **const**

3. Variables et Déclarations

Déclaration de Variables – Hoisting

`console.log(monVar);` // Utilisation de la variable avant sa déclaration
`var monVar = 5;` // Déclaration et initialisation de la variable

hoisting

`var monVar;` // Seule la déclaration est hissée
`console.log(monVar);` // Ici, monVar est undefined
`monVar = 5;` // Initialisation de la variable

`maFonction();` // Appel de la fonction avant sa déclaration

`function maFonction() {`
 `console.log('Hello!');`
`}`

`function maFonction() {` // La déclaration et la définition de la fonction sont hissées
 `console.log('Hello!');`
`}`

`maFonction();` // Appel de la fonction est valide

3. Variables et Déclarations



4. Types de Variables

4. Types de Variables

Important

- En Javascript, tout est un Objet (même les fonctions)

4. Types de Variables

Primitives - Nombres

- Javascript ne fait pas de distinction entre types de nombres (entiers, décimaux etc)

4. Types de Variables

Primitives - Nombres

```
// Déclaration de variables numériques
const entier = 42;           // Un nombre entier
const decimal = 3.14;       // Un nombre décimal
const negatif = -7;         // Un nombre négatif
const scientifique = 1.23e5; // Notation scientifique, équivalent à 123000

// Conversion de chaînes de caractères en nombres
const chaine = "123.45";
const nombreDepuisChaine = Number(chaine); // Convertit la chaîne en nombre

// Utilisation de Number pour la conversion et le calcul
const resultat = Number("10") + Number("20"); // Convertit les chaînes en nombres et les additionne
```

4. Types de Variables

Primitives – Chaines de Caractères

- Représentation des textes, entourées de guillemets simples, doubles, ou de backticks pour les **template literals**

4. Types de Variables

Primitives – Chaines de Caractères

```
// Déclaration de chaînes de caractères
const salutation = "Bonjour";
const nom = "Alice";

// Concaténation de chaînes
const message = salutation + " " + nom + "!";
console.log(message); // Affiche "Bonjour Alice!"

// Utilisation de Template Literals (backticks ` `)
const messageTemplate = `${salutation} ${nom}!`;
console.log(messageTemplate); // Affiche également "Bonjour Alice!"

// Longueur d'une chaîne
const longueur = nom.length;
console.log("Longueur du nom :", longueur); // Affiche la longueur du nom
```


4. Types de Variables

Primitives – Booléens

- Peuvent avoir une valeur de **true** ou **false**

4. Types de Variables

Primitives – Booléens



```
const valid = true  
const invalid = false
```


4. Types de Variables

Primitives – Symbols

- Genere une nouvelle valeur à chaque appel (sauf quand on utilise Symbol.for)
- Utile pour avoir des clés univoques dans des objets
- Attention: ne sont pas gardé pendant une conversion JSON (JSON.stringify)

4. Types de Variables

Primitives – Symbols

```

// Création d'un nouveau Symbol
const sym1 = Symbol();

// Création d'un Symbol avec une description
const sym2 = Symbol("description");

// Chaque Symbol est unique
console.log(sym1 === sym2); // false

```

4. Types de Variables

Primitives – Symbols – Alternatives pour Transférer des Données

- Utiliser la librairie Crypto du navigateur (Crypto.getRandomValues())
- Utiliser d'autres libraires externes (uuid par exemple)


4. Types de Variables

Primitives - Undefined ou Null

- Représentent respectivement une variable non initialisée et une valeur absente
- Null est souvent utilisé par le développeur pour indiquer quelque chose de vide
- Undefined est la valeur par default d'une variable non initialisée

4. Types de Variables

Primitives - Undefined ou Null



```
let valeurVide;  
let valeurNulle = null
```

4. Types de Variables



