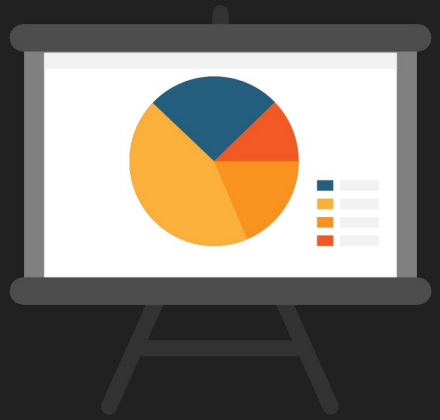
JavaScript Avancé

Rachid EDJEKOUANE (edjek@hotmail.fr)

Prérequis

- Notions d’Algorithmique
- Base HTML / CSS (côté navigateur)
- Maîtrise de son environnement informatique
- Maîtrise de son éditeur de code (Visual Studio Code)

Introduction



Javascript is not Java

Créé en 1995 par Brendan EICH (en 10 jours) pour le compte de la Netscape Communications Corporation, afin d'ajouter de l'interactivité sur les pages web, Microsoft Internet Explorer l'ajoute en 1996.

JavaScript est aujourd'hui l'un des langages de programmation les plus populaires au monde.

A l'origine le langage s'appelait 'LiveScript', il a été renommé JavaScript pour des raisons marketing au moment de sa sortie afin de surfer sur le succès du langage Java.

Javascript : côté client

Le rôle de JavaScript dans le développement web est crucial.

En effet, JavaScript permet d'ajouter de l'interactivité et du dynamisme aux pages web.

Grâce à JavaScript, il est possible de créer des effets visuels, des animations, des menus déroulants, des formulaires interactifs, des applications web complexes, et bien plus encore.

Javascript : côté server

JavaScript est également utilisé pour la création d'applications côté serveur, grâce à des environnements d'exécution tels que Node.js.

De cette façon, il est possible d'écrire du code JavaScript à la fois côté client et côté serveur, permettant une meilleure synchronisation et une expérience utilisateur plus fluide.

Javascript

En somme, JavaScript est un langage de programmation incontournable pour tout développeur web.

Sa popularité, sa flexibilité et son rôle clé dans l'interactivité et la dynamique des pages web en font un outil essentiel pour tout projet de développement web moderne.

Javascript en résumé

Un langage de programmation régulier, facile à démarrer, difficile à maîtriser.

Permet de donner une certaine interactivité aux éléments sur le web.

Syntaxe similaire à C ou Java mais sans typage.

Permet de modifier du contenu HTML ou CSS appliqué à un élément.

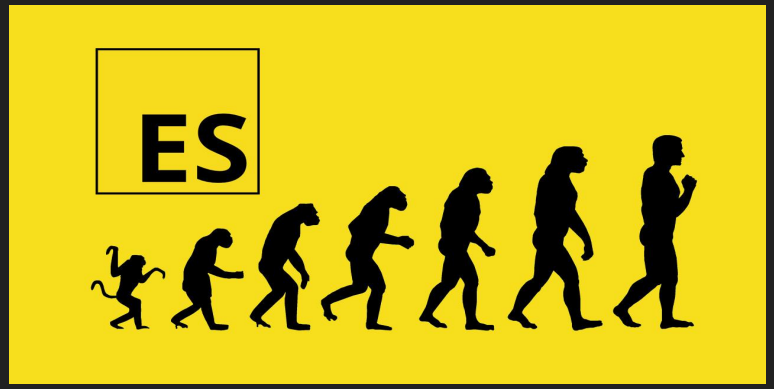
Vous pouvez même envoyer ou récupérer des informations sur Internet pour mettre à jour le contenu du Web sans recharger la page.

Pourquoi apprendre le Javascript ?

- Principal langage de script côté navigateur
- Langage polyvalent
- Ecosystème et communauté

Javascript ou ECMAScript

- JavaScript = Langage de script
- ECMA = Organisme de standardisation
- ES6 ECMAScript 6 (2015), ECMAScript 2022, ES13



Découverte

Javascript : insérer du code

Il existe trois façons d'exécuter du code javascript dans un site Web :

Importez un fichier Javascript à l'aide de la balise <script> :

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

Intégrez le code dans le HTML à l'aide de la balise <script>.

```
<script> /* du code */ </script>
```

Injectez le code sur un événement à l'intérieur d'un tag :

```
<button onclick="javascript: /*code*/">appuyez sur moi</button>
```

Javascript : syntaxe

Très similaire à C++ ou Java mais beaucoup plus simple.

```
let myNumber = 10; //Ceci est un commentaire
let myString = "hello";
let myArray = [10,20,"name",true];
let myObject = { name: "javi", city: "Barcelona" };

function hello(message)
{
    for(let i = 0; i < 10; i++)
        console.log(" dire: " + message);
}
```

Les bases

Javascript : les types de données

En JavaScript, il existe différents types de données et de variables qui peuvent être utilisés dans un programme.

Les types de données primitives sont les types de données les plus élémentaires.

Javascript : les types de données primitives

- Number : représente les nombres de toute nature, qu'ils soient entiers ou à virgule flottante. (limités à ±(2^53-1)).
- String : représente une chaîne de caractères.
- Boolean : représente une valeur de vérité, soit true (vrai) soit false (faux).
- Null : représente une valeur nulle ou inexistante.
- Undefined : représente une variable ou un objet qui n'a pas été défini.

Javascript : les variables

number :

```
let myNumber = 10;
```

string :

```
let myString = "hello" | 'hello' | `hello` ;
```

boolean :

```
let vrai = true;
```

undefined

```
let variable;
```

null

```
let variable = null;
```

Javascript : les tableaux et objets

En plus de ces types de données primitifs, JavaScript dispose également de deux types de données complexes :

- Array (tableau) : représente une liste ordonnée de valeurs.
- Object (objet) : représente une collection de propriétés, où chaque propriété est une paire clé: valeur.

Javascript : les variables

Array :

```
let myArray = [10, 20, 'name', true];
```

Object :

```
let person= {name: 'john', city: 'Paris'};
```

Javascript : les types de données

En ce qui concerne les variables en JavaScript est un langage à typage dynamique, ce qui signifie que le type de données d'une variable peut changer au cours du temps.

Les variables peuvent être déclarées en utilisant les mots-clés var, let ou const. Les variables déclarées avec var ont une portée globale ou de fonction, tandis que les variables déclarées avec let et const ont une portée de bloc.

Les variables déclarées avec const ne peuvent pas être réassignées après leur initialisation.

Javascript : assignation

L’opérateur d'assignation : ils permettent d'assigner une valeur à une variable.

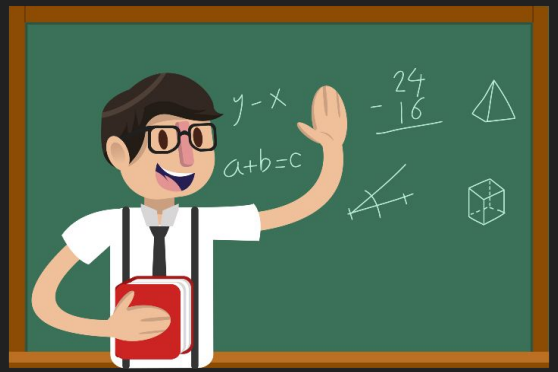
C’est l'opérateur "=" qui permet d'assigner une valeur à une variable.

```
let firstname = 'Rachid'
// la variable firstname contient la valeur Rachid

console.log(firstname);
// affiche : Rachid
```

Javascript : opérateurs arithmétiques

Les opérateurs arithmétiques : ils permettent de réaliser des opérations mathématiques sur des valeurs numériques.

Les opérateurs arithmétiques de base sont "+", "-", "*", "/", "**" et "%"



Javascript : sucre syntaxique

Les opérateurs d'incrémentation et de décrémentation : ils permettent d'ajouter ou de soustraire une unité à une variable.

Les opérateurs d'incrémentation sont "++" et "--"

Javascript : opérateurs de concaténation

L’opérateur de concaténation : ils permettent de joindre des chaînes de caractères.

L'opérateur de concaténation est le signe "+"

```
let firstname = 'Rachid'

console.log('Bonjour, mon nom est ' + firstname + '!');
// affiche : Bonjour, mon nom est Rachid!
```

Conditions

Javascript : opérateurs de comparaison

Les opérateurs de comparaison permettent de comparer des valeurs.

Les opérateurs de comparaison :

"==", "===", "!=", "!==", "<", "<=", ">", ">="

Javascript : opérateurs logiques

Ils permettent de combiner des expressions logiques.

Les opérateurs logiques les plus courants sont :

"&&" (et), "||" (ou) et "!" (non)

Javascript : les structures de contrôle

Les structures de contrôle permettent de diriger le flux d'exécution d'un programme.

Elles se divisent en deux familles :

Les structures de contrôle conditionnelles permettent d'exécuter des instructions en fonction d'une condition. Les plus courantes sont les structures "if" et "switch".

Les structures de contrôle de boucles permettent de répéter des instructions un certain nombre de fois ou jusqu'à ce qu'une condition soit remplie. Les plus courantes sont les boucles "for" et "while".

Javascript : les structures conditionnelles (if/else)

Une structure conditionnelle permet de tester si une condition est vraie ou fausse, et de prendre une décision en conséquence.
Les structures conditionnelles sont utilisées pour donner de l'interactivité aux scripts.

La structure "if" permet d'exécuter une ou plusieurs instructions si une condition est true, ou une autre instruction si la condition est false.
Il est possible d'imbriquer plusieurs instructions "if" pour réaliser des tests plus complexes.

Javascript : les conditions

Si, Sinon Si, Sinon :

```
let age = 18;
if (val > 18) {
       console.log("Vous êtes majeur !");
} else if (val < 18) {
       console.log("Vous êtes mineur!");
} else {
       console.log("Vous n’avez pas donner de nombre !");
}
```

Javascript : les structures conditionnelles (switch)

La structure "switch" est une alternative à la structure "if" qui permet de tester une expression sur plusieurs valeurs possibles.

Elle est souvent utilisée pour réaliser des tests sur une même variable avec plusieurs valeurs possibles, sans avoir à répéter la variable à chaque fois.

Javascript : les conditions (switch/case)

```
switch (expr) {
    case "16" :
        console.log("Vous êtes mineur!");
        break;
    case "20" :
        console.log("Vous êtes majeur!");
        break;
    default :
        console.log("Vous n’avez pas donné de nombre !");
}
```

Les boucles

Javascript : les structures de contrôle de boucles

En programmation, une boucle est une structure de contrôle qui permet de répéter un bloc d'instructions un certain nombre de fois.

En JavaScript, il existe plusieurs types de boucles, chacune ayant des spécificités propres.

Javascript : les boucles (for)

Dans la boucle for() la "condition" est vérifiée avant chaque itération, si elle est true, le bloc d'instructions est exécuté.

Ensuite, l'expression d'incrémentation est évaluée, puis la condition est à nouveau vérifiée, et ainsi de suite jusqu'à ce que la condition ne soit plus true.

```
for(let i = 0; i < 10; i++) {
    console.log(i)
}
```

Javascript : les boucles (while)

La boucle "while" permet d'exécuter un bloc d'instructions tant qu'une condition est vraie.

Elle s'utilise de la façon suivante :

```
let i = 0;

while(i < 10) {
    i++;
}
```

Javascript : les boucles (do while)

La boucle "do…while" permet d'exécuter un bloc d'instructions tant qu'une condition est vraie, mais dans tous les cas l'instruction sera exécutée au moins une fois.

Elle s'utilise de la façon suivante :

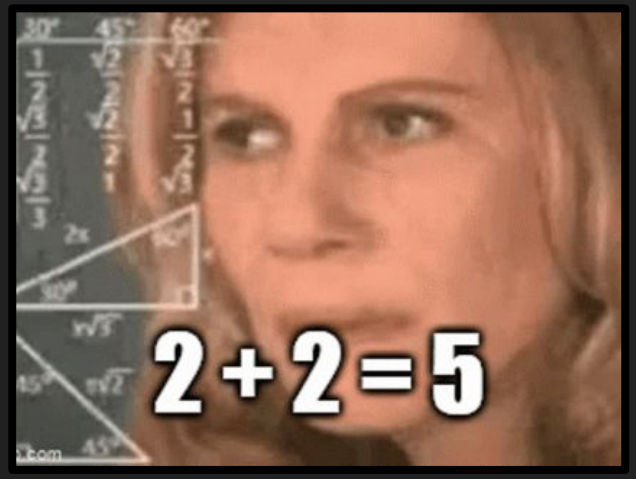
```
let i = 0;

do {
    i++;
} while(i < 10)
```

Javascript : les boucles

Il est important de noter que la boucle "for" est la plus couramment utilisée en JavaScript, mais les autres boucles peuvent s'avérer utiles dans certaines situations.

De plus, il est important de faire attention aux boucles infinies, qui peuvent causer des problèmes de performance et bloquer l'exécution du code.

Les fonctions



Javascript : les fonctions

Les fonctions en JavaScript sont des blocs de code qui sont conçus pour effectuer une tâche spécifique.

Elles permettent de découper un programme en petits morceaux plus faciles à comprendre et à gérer.

Une fonction peut prendre des arguments, qui sont des valeurs utilisées par la fonction lors de son exécution, et peut également renvoyer une valeur en retour.

Javascript : les fonctions

Il existe deux types de fonctions en JavaScript : les fonctions natives ou prédéfinies et les fonctions personnalisées.

Les fonctions natives sont des fonctions qui sont déjà incluses dans le langage JavaScript et qui peuvent être appelées directement.

Les fonctions personnalisées sont créées par les développeurs et peuvent être appelées de la même manière que les fonctions natives.

Pour appeler une fonction, on utilise son nom suivi de parenthèses, qui peuvent contenir des arguments si la fonction en a besoin.

Javascript : fonction en pratique

```
function addition(a, b) {
  return a + b;
}

let result = addition(2, 3);

console.log(result); // affiche 5
```

Javascript : fonction anonyme

En JavaScript, une fonction anonyme est une fonction qui n'a pas de nom explicite. Elle peut être déclarée de deux manières différentes : avec une expression de fonction anonyme ou avec une fonction fléchée.

Les fonctions anonymes sont souvent utilisées pour des fonctions auto-invoquées, des callbacks ou pour des fonctions qui ne seront utilisées qu'une seule fois. Elles permettent également de limiter la portée des variables en créant une closure.

Javascript : fonction fléchée

Les fonctions fléchées, également appelées "arrow functions" en anglais, sont une nouvelle syntaxe introduite en ES6 (ECMAScript 2015) pour déclarer des fonctions en JavaScript.

Les fonctions fléchées ont quelques avantages par rapport aux fonctions traditionnelles.

```
const multiply = (a, b) => { return a * b };
```

La flèche => remplace le mot clé function et le code à exécuter est placé après la flèche.

Javascript : fonctions fléchées

Voici quelques avantages des fonctions fléchées en JavaScript :

- Syntaxe concise : Le code est plus facile à lire et à comprendre.
- Liaison de contexte automatique : Dans les fonctions fléchées, le context this est automatiquement lié au context parent.
- Pas toujours besoin du mot clé return : Si elle contient qu'une seule instruction de retour, il n'est pas nécessaire d'utiliser le mot clé return.
- Fonctions anonymes : Les fonctions fléchées peuvent être utilisées comme des fonctions anonymes, ce qui permet de gagner du temps lors de la déclaration de fonctions simples.

Javascript : portée des variables

La portée de bloc est la portée la plus récente et est introduite avec l'utilisation de let et const. Les variables déclarées avec let ou const sont accessibles uniquement à l'intérieur du bloc où elles ont été déclarées.

Il est important de noter que la portée des variables est déterminée par le lieu où elles sont déclarées et non par l'endroit où elles sont utilisées.

En utilisant le mot-clé var, la portée d'une variable est soit la fonction qui contient la déclaration, soit le contexte global si la variable est déclarée en dehors de toute fonction.

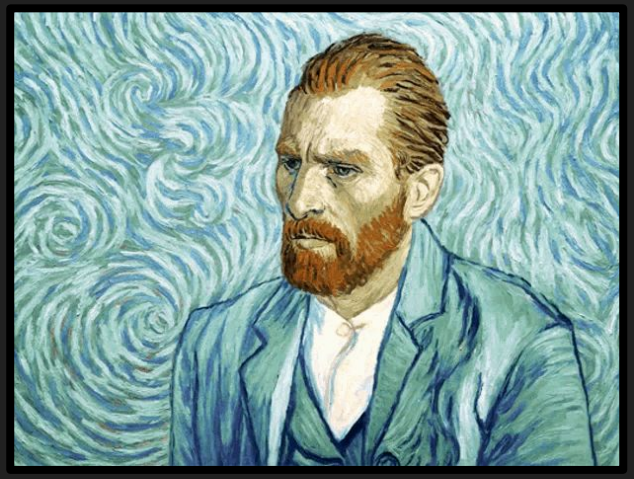
Javascript : scope

La portée d'une variable (scope) détermine où celle-ci peut être accessible dans le script. Il existe trois types de portée de variable en JavaScript : la portée globale, la portée locale et la portée de bloc.

La portée globale est la portée la plus large et les variables définies dans cette portée sont accessibles depuis n'importe quelle partie du code.

La portée locale est définie à l'intérieur d'une fonction et les variables définies dans cette portée ne peuvent être utilisées que dans cette fonction.

Les tableaux



Javascript : les tableaux

Les array (tableaux) en JavaScript sont des structures de données qui permettent de stocker et d'accéder à plusieurs valeurs en utilisant un seul nom.

```
let fruits = ['pomme', 'banane', 'orange', 'kiwi'];
```

Les éléments d'un tableau peuvent être de différents types de données, tels que des chaînes de caractères, des nombres, des booléens, des objets ou même d'autres tableaux.

Javascript : les tableaux

Nous pouvons accéder à un élément spécifique d'un tableau en utilisant son index, qui commence à 0 pour le premier élément.

Par exemple, pour accéder au premier élément du tableau fruits, nous devons utiliser l'index 0:

```
fruits[0] // correspond à 'pomme'
```

Javascript : méthodes de tableau

Les méthodes de tableau sont des fonctions intégrées à JavaScript qui sont utilisées pour effectuer des opérations sur des tableaux.

Les méthodes les plus couramment utilisées sont push(), pop(), shift(), unshift(), splice() et slice(). Elles permettent d'ajouter ou de supprimer des éléments à un tableau, de découper ou de copier des parties d'un tableau, etc.

Javascript : les tableaux

```
console.log(fruits); // affiche ['pomme', 'banane', 'orange', 'kiwi', 'raisin'] dans la console

fruits.push("raisin"); // ajoute 'raisin' à la fin du tableau

fruits.pop(); // supprime le dernier élément du tableau

console.log(fruits); // affiche ['pomme', 'banane', 'orange', 'kiwi'] dans la console
```

Les Objets

Javascript : les objets explication

En JavaScript, les objets sont des structures de données complexes qui permettent de stocker des valeurs et des fonctions ensemble. Les objets sont utilisés pour représenter des entités du monde réel ou des concepts abstraits dans un programme.

La plupart des valeurs en JavaScript sont des objets ou peuvent être converties en objets.

Par exemple, les chaînes de caractères, les nombres et les tableaux sont tous des types d'objets en JavaScript.

Javascript : les objets (déclaration)

```
let person = {
    firstName: 'julien',
    lastName: 'Dupont',
    notes : [13, 16, 19],
    brother: { firstName : 'Stéphane'
    },
    hello: function() {
        return 'Hello, world!'// code de la méthode
    }
};
```

Javascript : les objets (utilisation)

Les méthodes d'un objet sont des fonctions qui peuvent être appelées en utilisant la syntaxe suivante :

```
person.hello(); // retourne Hello, world!
```

Javascript : les objets (exemple de code)

Lorsqu'une méthode est appelée, elle peut accéder aux propriétés de l'objet en utilisant le mot clé `this` :

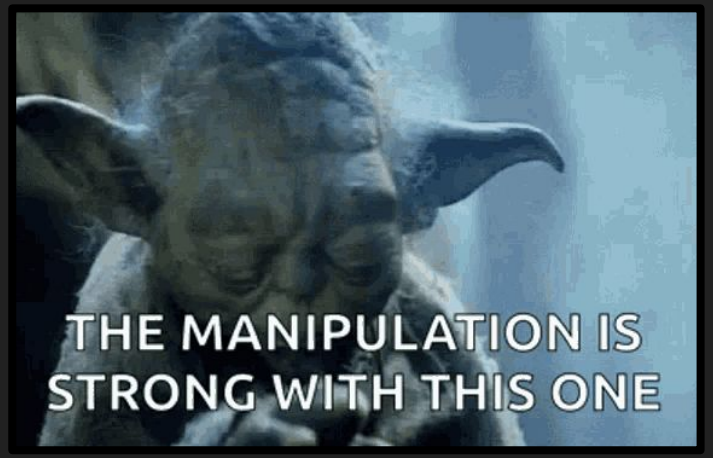
```
let objet = {
    propriete: 'valeur',
    methode: function() {
        console.log(this.propriete);
    }
};

objet.methode(); // affiche 'valeur' dans la console
```

Javascript : les objets (intérêt)

En utilisant les objets en JavaScript, vous pouvez organiser et manipuler des données de manière plus efficace.

Les objets sont utilisés dans de nombreux aspects du développement JavaScript, tels que les frameworks et les bibliothèques de développement web, les applications de visualisation de données et les jeux vidéo en ligne.

Manipulation du DOM



Javascript : les objets (déclaration)

```
let person = {
    firstName: 'julien',
    lastName: 'Dupont',
    notes : [13, 16, 19],
    brother: { firstName : 'Stéphane'
    },
    hello: function() {
        return 'Hello, world!'// code de la méthode
    }
};
```

Javascript : les objets (utilisation)

Les méthodes d'un objet sont des fonctions qui peuvent être appelées en utilisant la syntaxe suivante :

```
person.hello(); // retourne Hello, world!
```

Javascript : les objets (exemple de code)

Lorsqu'une méthode est appelée, elle peut accéder aux propriétés de l'objet en utilisant le mot clé `this` :

```
let objet = {
    propriete: 'valeur',
    methode: function() {
        console.log(this.propriete);
    }
};

objet.methode(); // affiche 'valeur' dans la console
```

Javascript : les objets (intérêt)

En utilisant les objets en JavaScript, vous pouvez organiser et manipuler des données de manière plus efficace.

Les objets sont utilisés dans de nombreux aspects du développement JavaScript, tels que les frameworks et les bibliothèques de développement web, les applications de visualisation de données et les jeux vidéo en ligne.

Javascript : les objets explication

En JavaScript, les objets sont des structures de données complexes qui permettent de stocker des valeurs et des fonctions ensemble. Les objets sont utilisés pour représenter des entités du monde réel ou des concepts abstraits dans un programme.

La plupart des valeurs en JavaScript sont des objets ou peuvent être converties en objets.

Par exemple, les chaînes de caractères, les nombres et les tableaux sont tous des types d'objets en JavaScript.

Javascript : manipuler le DOM

```
<button id="myButton">Cliquez sur le bouton</button>

const elem = document.querySelector('#myButton');

elem.addEventListener('click', function(){
    elem.textContent = 'coucou'
})
```

Événements

Javascript : les événements

En JavaScript, les événements sont des actions ou des occurrences qui se produisent dans le navigateur, tels que le chargement d'une page, le clic sur un bouton ou encore la saisie de texte dans un champ de formulaire.

Les événements permettent d'interagir avec l'utilisateur et de rendre les pages Web plus dynamiques.

Javascript : interface event

Les événements sont associés à des éléments du DOM (Document Object Model) tels que des boutons, des liens ou des formulaires, et sont gérés par des fonctions appelées "gestionnaires d'événements".

Lorsqu'un événement se produit sur un élément, le navigateur appelle la fonction de gestionnaire d'événements correspondante pour exécuter une action spécifique en réponse à l'événement.

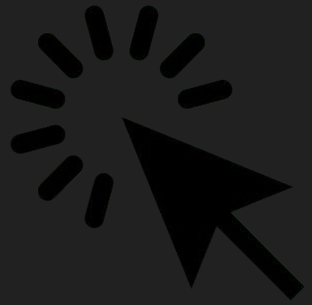
Javascript : interface event

Ces événements sont représentés en JavaScript par des objets basés sur l'interface Event. Certains types d'événements sont définis plus précisément dans des interfaces qui héritent de Event.

Pour ajouter un gestionnaire d'événements à un élément, on utilise la méthode addEventListener() qui permet de spécifier le type d'événement à écouter et la fonction de gestionnaire d'événements à appeler en réponse à cet événement.

Javascript : interface event

En somme, les événements en JavaScript permettent d'interagir avec les utilisateurs en répondant à leurs actions sur les éléments du DOM de la page web.



Gestion des Evénements

Javascript : gestion des évènements

La gestion des événements dans le DOM permet de détecter des actions effectuées par l'utilisateur sur une page web et de réagir à ces actions en déclenchant une fonction ou un traitement en réponse à l'événement détecté.

Pour ajouter un gestionnaire d'événements en JavaScript, on peut utiliser la méthode addEventListener() qui prend en paramètre le nom de l'événement à détecter et la fonction à exécuter en réponse à cet événement.

Javascript : gestion des évènements

Voici un exemple simple de gestion d'événement en JavaScript pour détecter un clic sur un bouton et afficher une alerte :

```
// Récupération du bouton par son ID
    const bouton = document.getElementById('mon-bouton');

// Ajout d'un gestionnaire d'événement au clic
    bouton.addEventListener('click', function() {
        alert('Le bouton a été cliqué !');
    });
```

Javascript : la validation des formulaires

La gestion des formulaires en JavaScript permet de manipuler et d'envoyer des données à partir de formulaires HTML.

Cette manipulation peut être utile pour effectuer des vérifications sur les données saisies, les modifier ou les envoyer à un serveur sans rechargement de la page.

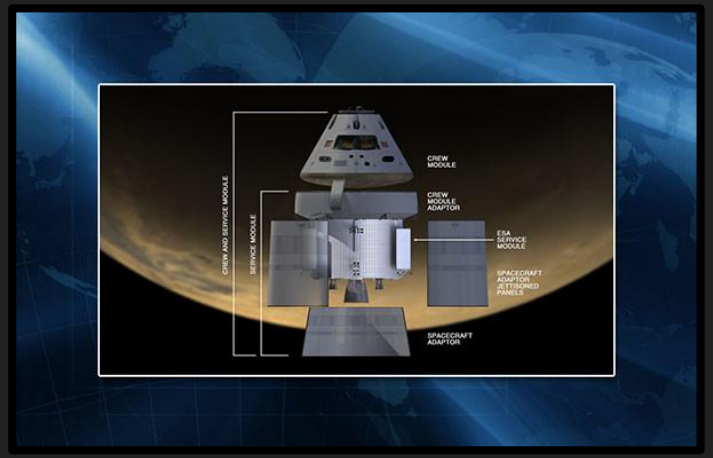
Javascript : bonnes pratiques

En ce qui concerne les bonnes pratiques, il est important de garder à l'esprit la simplicité et la lisibilité du code.

Utilisez des noms de variables descriptifs et évitez les noms de variables génériques tels que 'a', 'b', 'x' ou 'y'.

Enfin, il est important de documenter le code, en incluant des commentaires pour expliquer le but et le fonctionnement des fonctions, des méthodes et des variables.

Modules


CREW AND SERVICE MODULE
SERVICE MODULE
CREW MODULE
CREW MODULE ADAPTOR
ESA SERVICE MODULE
SPACECRAFT ADAPTOR JETTISONED PANELS
SPACECRAFT ADAPTOR

Javascript : les modules

En JavaScript, les modules sont des fichiers JavaScript qui exportent des fonctionnalités pour les rendre utilisables dans d'autres fichiers JavaScript.

Les modules permettent d'organiser et de découper votre code en morceaux réutilisables, ce qui améliore la maintenance et la lisibilité de votre code.

Javascript : les modules

```
<script type="module" src="main.js"></script>

// math.mjs
export const add = (a, b) => a + b;
export const subtract = (a, b) => a - b;

// main.mjs
import { add, subtract } from './math.js';
console.log(add(5, 3)); // Output: 8
console.log(subtract(5, 3)); // Output: 2
```

Déstructuration

Javascript : destructuring

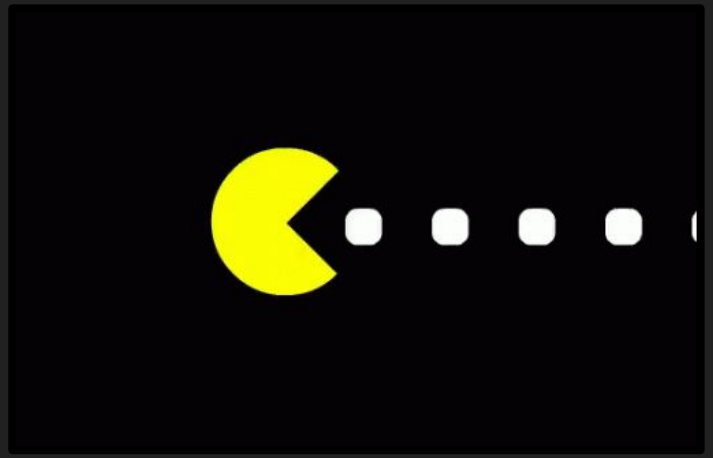
La déstructuration de l'objet est une caractéristique JavaScript utile pour extraire les propriétés des objets et les lier aux variables.

Mieux encore, la déstructuration de l'objet peut extraire plusieurs propriétés dans une seule déclaration, peut accéder aux propriétés à partir d'objets imbriqués, et peut définir une valeur par défaut si la propriété n'existe pas

Javascript : destructuring

```
const person = {
    firstName: 'John',
    lastName: 'Doe',
    age: 30,
};

const { firstName, lastName } = person;
```

Spread Operator



Javascript : spread operator

L'opérateur de propagation, souvent appelé "spread operator" en anglais, est une fonctionnalité de JavaScript introduite dans ECMAScript 6 (ES6).

Il est utilisé pour étendre une expression dans des endroits où de multiples éléments (comme des arguments de fonctions ou des éléments de tableaux) sont attendus.

Javascript : spread operator

copier un tableau :

```
const tableauOriginal = [1, 2, 3];
const copieTableau = [...tableauOriginal];
console.log(copieTableau); // [1, 2, 3]
```

concaténer 2 tableaux :

```
const tableau1 = [1, 2, 3];
const tableau2 = [4, 5, 6];
const tableauConcatene = [...tableau1, ...tableau2];
console.log(tableauConcatene); // [1, 2, 3, 4, 5, 6]
```

array

Javascript : ES6 fonctions sur les tableaux

Les méthodes de tableau en JavaScript sont des fonctions intégrées qui peuvent être utilisées pour effectuer des opérations sur des tableaux.

Elles permettent de manipuler, de filtrer, de trier et de transformer les éléments d'un tableau de différentes manières.

Javascript : ES6 fonctions sur les tableaux

map() : Crée un nouveau tableau avec les résultats de l'appel d'une fonction fournie sur chaque élément du tableau d'origine.

filter() : Crée un nouveau tableau contenant tous les éléments du tableau d'origine qui remplissent une condition donnée par une fonction de test.

reduce() : Applique une fonction à un accumulateur et à chaque élément du tableau (de gauche à droite) pour réduire le tableau à une seule valeur.

find() : Renvoie la première valeur d'un tableau qui satisfait une condition donnée par une fonction de test.

HTTP

Javascript : requêtes HTTP

En informatique, HTTP (Hypertext Transfer Protocol) est le protocole utilisé pour transférer des données sur le Web.

Les requêtes HTTP sont des messages envoyés par le navigateur Web ou l'application cliente à un serveur Web pour demander des données, telles que des pages Web ou des ressources comme des images ou des fichiers.

Javascript : API et leur manipulation

Une API (Application Programming Interface) est une interface logicielle qui permet à deux applications de communiquer entre elles.

En d'autres termes, une API permet à une application de demander des données ou des services à une autre application.

XMLHttpRequest

Javascript : requêtes HTTP et XMLHttpRequest

En JavaScript, nous pouvons utiliser l'objet XMLHttpRequest (XHR) pour envoyer des requêtes HTTP depuis le navigateur Web. Voici un exemple de code qui envoie une requête HTTP GET à l'aide de l'objet XHR :

Javascript : XMLHttpRequest

```
let xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data');
xhr.onload = function() {
  if (xhr.status === 200) {
    console.log(xhr.responseText);
  } else {
    console.log('Erreur de requête');
  }
};
xhr.send();
```

Javascript : XMLHttpRequest

Dans cet exemple, nous avons créé un nouvel objet XHR en utilisant new XMLHttpRequest(). Nous avons ensuite ouvert une connexion HTTP en utilisant la méthode open(), spécifiant la méthode HTTP (GET dans ce cas) et l'URL de la ressource à récupérer.

Nous avons également défini une fonction de rappel onload qui est appelée lorsque la réponse est reçue. Si la réponse est réussie (statut HTTP 200), nous afficherons la réponse dans la console. Sinon, nous affichons un message d'erreur.

Javascript : method XMLHttpRequest

En plus de la méthode GET, nous pouvons utiliser les méthodes HTTP POST, PUT et DELETE pour envoyer des données à un serveur.

Voici un exemple de code qui envoie une requête HTTP POST avec des données JSON à l'aide de l'objet XHR :

Javascript : XMLHttpRequest en pratique

```
let xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.example.com/data');
xhr.setRequestHeader('Content-Type', 'application/json');
xhr.onload = function() {
  if (xhr.status === 200) {
    console.log(xhr.responseText);
  } else {
    console.log('Erreur de requête');
  }
};
let data = { nom: "John", age: 30 };
xhr.send(JSON.stringify(data));
```

Javascript : requêtes HTTP et leur manipulation

Dans cet exemple, nous avons utilisé la méthode HTTP POST en spécifiant l'URL de la ressource et le type de contenu des données à l'aide de setRequestHeader(). Nous avons ensuite défini une fonction de rappel onload qui est appelée lorsque la réponse est reçue.

Nous avons également créé un objet JavaScript data contenant les données que nous voulons envoyer au serveur, puis nous avons utilisé JSON.stringify() pour les convertir en format JSON et les envoyer avec la méthode send().

Fetch()

Javascript : API et leur manipulation

En JavaScript, nous pouvons utiliser l'objet Fetch pour interagir avec des API en envoyant des requêtes HTTP pour récupérer des données. Voici un exemple de code qui utilise l'objet Fetch pour récupérer des données JSON depuis une API :

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

Javascript : API et leur manipulation

Dans cet exemple, nous avons utilisé la méthode fetch() pour envoyer une requête HTTP GET à l'URL de l'API.

La méthode fetch() renvoie une promesse qui résoudra en une réponse Response à la requête.

Nous avons ensuite appelé la méthode json() sur la réponse pour extraire les données JSON de la réponse.

La méthode json() renvoie également une promesse qui résoudra en l'objet JavaScript correspondant aux données JSON.

Javascript : API et leur manipulation

Enfin, nous avons utilisé la méthode then() pour gérer la promesse renvoyée par la méthode json(). Dans cette méthode, nous avons utilisé console.log() pour afficher les données dans la console. Si une erreur se produit, nous l’affichons dans la console à l'aide de la méthode catch().

Javascript : API et leur manipulation

En plus de la méthode GET, nous pouvons utiliser les méthodes HTTP POST, PUT et DELETE pour envoyer des données à une API. Voici un exemple de code qui envoie des données JSON à une API en utilisant la méthode HTTP POST:

Javascript : API et leur manipulation

```
fetch('https://api.example.com/data', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json'
  },
  body: JSON.stringify({ nom: "John", age: 30 })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error(error));
```

Javascript : API et leur manipulation

Dans cet exemple, nous avons utilisé l'option method pour spécifier que nous voulons utiliser la méthode HTTP POST. Nous avons également utilisé l'option headers pour spécifier que nous envoyons des données JSON.

Enfin, nous avons utilisé l'option body pour spécifier les données à envoyer à l'API. Nous avons créé un objet JavaScript data contenant les données que nous voulons envoyer au serveur, puis nous avons utilisé JSON.stringify() pour les convertir en format JSON.

Gestion des erreurs et débogage

Javascript : try … catch

En JavaScript, la structure try...catch est utilisée pour gérer les erreurs dans le code. Elle permet d'entourer un bloc de code qui pourrait générer une exception, puis de spécifier comment réagir en cas d'erreur.

```
try {
  // Bloc de code pouvant générer une exception
  } catch (error) {
  console.error("Une erreur s'est produite :",
error.message);
}
```

Javascript : try … catch

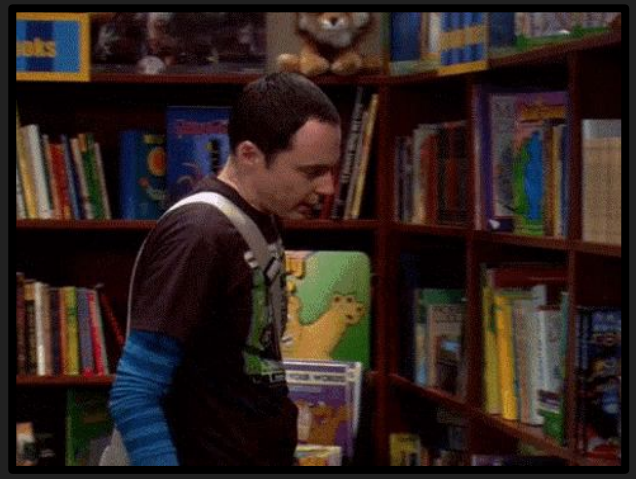
Explications :

- Le code à risque est placé dans le bloc try.
- Si une exception est levée, le bloc catch est exécuté, avec l'objet d'erreur passé comme paramètre.

Javascript : bonnes pratiques

- Éviter les erreurs de syntaxe : Assurez-vous que votre code est correctement indenté et que toutes les accolades et parenthèses sont correctement fermées
- Éviter les variables globales : Les variables globales peuvent entraîner des conflits et des erreurs difficiles à identifier.
- Utiliser console.log() pour le débogage
- Utiliser des blocs try-catch pour gérer les erreurs
- Utiliser des outils de débogage du navigateur

Frameworks bibliothèques



Javascript : frameworks et bibliothèques

La différence entre un framework et une bibliothèque en JavaScript. Une bibliothèque JavaScript est un ensemble de fonctions prêtes à l'emploi qui peuvent être utilisées dans votre code pour effectuer des tâches spécifiques, telles que la manipulation du DOM ou l'animation.

En revanche, un framework JavaScript est un ensemble plus vaste d'outils et de bibliothèques qui fournissent une structure pour votre application et un ensemble de règles à suivre pour organiser votre code.

Javascript : librairies (jQuery, Lodash, Three.js)

- jQuery est une bibliothèque JavaScript populaire qui facilite la manipulation du DOM, la gestion des événements et la création d'effets visuels.
- Lodash est une bibliothèque JavaScript qui fournit des fonctions utilitaires pour la manipulation de tableaux, de chaînes de caractères, d'objets et de nombres.
- Three.js est une bibliothèque JavaScript qui est utilisée pour la création de graphiques en 3D dans un navigateur web. Elle fournit un ensemble d'outils pour créer des objets en 3D, des animations et des effets visuels.

Javascript : librairies, les avantages

Avantages des bibliothèques JavaScript :

- Elles permettent de gagner du temps en fournissant des fonctions prêtes à l'emploi pour effectuer des tâches courantes.
- Elles facilitent l'écriture de code en fournissant des abstractions plus simples et plus claires pour des tâches complexes.
- Elles sont plus faciles à apprendre et à utiliser pour les débutants.
- Elles sont souvent légères, ce qui signifie qu'elles n’alourdissent pas le poids de votre application.

Javascript : librairies, les inconvénients

Inconvénients des bibliothèques JavaScript :

- Elles peuvent entraîner des conflits si plusieurs bibliothèques sont utilisées dans une même application.
- Elles peuvent parfois ralentir les performances si elles sont mal utilisées ou si elles sont trop lourdes.
- Elles peuvent masquer la complexité du code sous-jacent, ce qui peut rendre le débogage plus difficile.

Javascript : frameworks

Un framework JavaScript est un ensemble d'outils et de bibliothèques qui permettent de faciliter le développement d'applications web.

Les frameworks sont conçus pour fournir des fonctionnalités communes et des méthodologies pour organiser et structurer le code JavaScript.

Javascript : frameworks (React, Angular, Vue)

- React est une bibliothèque JavaScript développée par Facebook. Elle est utilisée pour la création d'interfaces utilisateur et utilise une approche basée sur les composants pour organiser le code.
- Angular est un framework JavaScript développé par Google. Il est conçu pour le développement d'applications web monopages (SPA) et utilise une architecture de composants pour organiser le code.
- Vue.js est un framework JavaScript open source qui est facile à apprendre et à utiliser. Il est conçu pour la création d'applications web interactives et utilise une approche basée sur les composants pour organiser le code.

Javascript : frameworks , les avantages

Avantages des frameworks JavaScript :

- Ils fournissent une structure solide pour votre application, ce qui facilite l'organisation et la maintenabilité du code.
- Ils sont riches en fonctionnalités, ce qui permet de gagner du temps en fournissant des outils prêts à l'emploi pour les tâches courantes.
- Ils ont une communauté de développeurs active, ce qui permet de bénéficier de support et de documentation de qualité.
- Ils peuvent être utilisés pour des projets de grande envergure, ce qui facilite la gestion des données et la manipulation du code.

Javascript : frameworks , les inconvénients

Inconvénients des frameworks JavaScript :

- Ils peuvent être lourds, ce qui peut ralentir les performances de votre application.
- Ils ont une courbe d'apprentissage plus raide, ce qui peut rendre l'utilisation difficile pour les débutants.
- Ils peuvent parfois être trop rigides dans leur structure, ce qui peut rendre l'adaptation à des besoins spécifiques plus difficiles.
- Ils peuvent entraîner une dépendance à un framework particulier, ce qui rend la transition vers d'autres technologies plus difficile.

```
console.log('MERCI')
```