

JAVASCRIPT

ASYNCHRONISME





SOMMAIRE

Introduction

Mise en place de l'environnement

Callback

Promesse

Async / Await

Fonction fetch

API

JSON Server

Rapid API

Aller plus loin



INTRODUCTION

JAVASCRIPT

INCONTOURNABLE DU DÉVELOPPEMENT WEB

C'est un langage de programmation de haut niveau

Son typage est dynamique faible

Il s'exécute sur tous les navigateurs (Chrome, Firefox...)

Il est possible de l'exécuter sur un serveur à l'aide de Node.js





ECMAScript

NORME JAVASCRIPT

ECMAScript est un ensemble de normes concernant les langages de programmation de type script et standardisées par Ecma International.

Il s'agit donc d'un standard, dont les spécifications sont mises en œuvre dans différents langages de script, comme JavaScript.



CONTEXTE D'EXÉCUTION

RUNTIME

Les navigateurs disposent d'un moteur d'exécution ou runtime permettant d'exécuter le langage JavaScript.

Ils peuvent être différents en fonction des navigateurs, cela explique dans de rares cas une différence de rendu.

En 2009, Ryan Dahl a créé Node, un programme en C++ permettant d'exécuter du code JavaScript côté serveur. Node a révolutionné le monde du développement web, en donnant à JavaScript la possibilité d'être à la fois un langage frontend et backend.



Moteur V8

Google



SpiderMonkey

Firefox



NodeJS

OpenAI



MISE EN PLACE DE L'ENVIRONNEMENT

CHOISIR UN NAVIGATEUR

OÙ NOTRE CODE VA S'EXÉCUTER ?



Firefox

Mozilla Corp.



Opera

Opera Software



Google Chrome

Google



Microsoft Edge

Microsoft



Veuillez éviter Microsoft Edge...

Les outils développeurs ainsi que le rendu de certains éléments ne sont pas à jour.

CHOISIR UN IDE / ÉDITEUR DE TEXTE

ENVIRONNEMENT DE DÉVELOPPEMENT



WebStorm
JetBrains

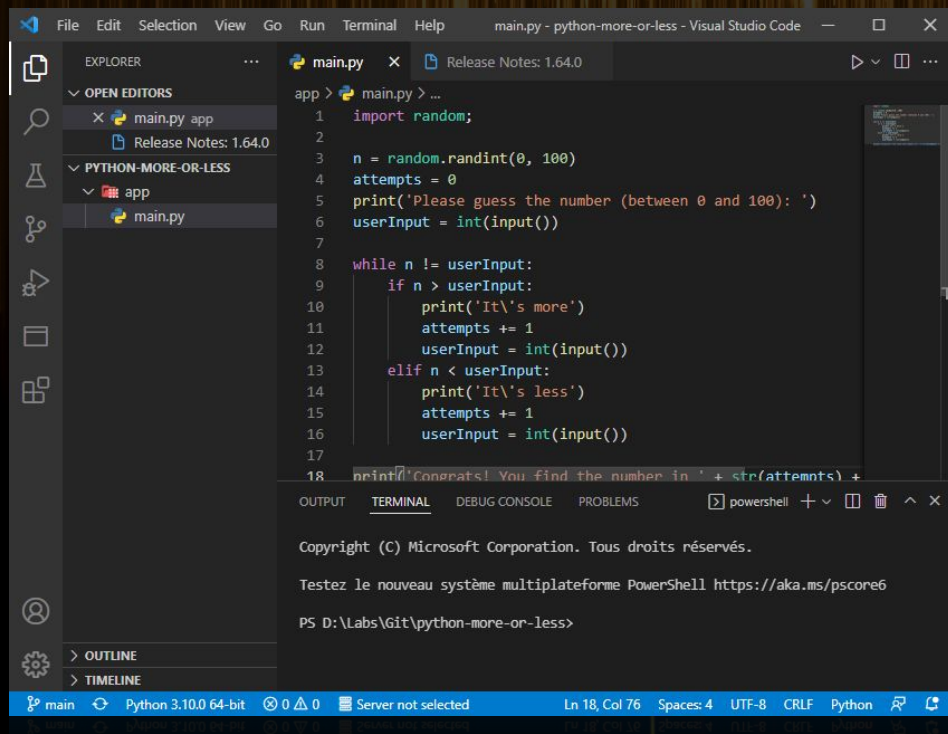


Visual Studio Code
Microsoft

<https://code.visualstudio.com/>

VISUAL STUDIO CODE

INTERFACE & RACCOURCIS



Gratuit

Système d'extensions

Connecté au système de versionning

Le plus populaire

Et bien plus...

NODE JS

RUNTIME JAVASCRIPT



Node JS
OpenJS Foundation

<https://nodejs.org/fr/>



Lors de l'installation de Node.js, veillez à bien ajouter le chemin parmi les variables d'environnement système.

```
node -v  
v19.0.1
```

```
--
```

TERMINAL / CMD

VÉRIFICATION DE L'INSTALLATION

NodeJS est un runtime JavaScript basé sur le moteur V8 de Google Chrome.
Il va nous être nécessaire pour avoir npm qui sera notre gestionnaire de packages / dépendances.

npm

EXTENSION CODE RUNNER

CÔTÉ PRATIQUE

.run

Code Runner

Jun Han

★★★★★

JavaScript

PHP

Python

TypeScript

Et bien plus...

Une fois l'extension installée, il suffira de sélectionner dans le fichier JavaScript le code à exécuter, et cliquer sur run code.

CRÉATION DU PROJET

KICK START JAVASCRIPT !

js-course



1. Créer et ouvrir le dossier js-course à l'aide de VS. Code.
2. Créer un dossier app.
3. Créer un dossier js dans le dossier app.
4. Créer le fichier main.js dans le dossier js.



CALLBACK



DÉFINITION

C'EST QUOI UN CALLBACK / FONCTION DE RAPPEL ?

En JavaScript, il est important de comprendre que l'on peut stocker une fonction dans une variable.

Si une fonction peut s'apparenter à une variable alors on peut faire passer une fonction en paramètre d'une autre fonction. C'est ce que l'on appelle un callback.

JS

```
const isEven = (n) => {  
  return n % 2 == 0  
}  
  
console.log(typeof isEven)
```

main.js



function

SORTIE CONSOLE



MÉTHODE MAP

FONCTIONNEMENT

JS

```
// Faire partir les ids de zéro
let productIds = [4, 5, 6, 7]
let newProductIds = productIds.map((elem, key) => key)
console.log(newProductIds)

// Application de la racine carrée pour chaque élément
let elements = [1, 36, 9, 25]
let squareRoots = elements.map(Math.sqrt)
console.log(squareRoots)
```

main.js



[0, 1, 2, 3]

[1, 6, 3, 5]

SORTIE CONSOLE

La méthode map prend une fonction en paramètre et pour chaque élément du tableau, elle applique le traitement défini par la fonction.

—

BLOQUANT / NON BLOQUANT

COMPRENDRE L'UTILITÉ DES CALLBACKS

JS

```
import fs, { readFile } from 'fs'

// Bloquant
const content = readFileSync('./data/poem.txt')
console.log('Poem : ', content)

// Non bloquant
readFile('./data/poem.txt', (err, content) => {
  if (err) throw err
  console.log('Poem : ', content)
})

console.log('Next operations...')
```

main.js

Le système de callback permet d'avoir un code non bloquant. Une fois le fichier "poem.txt" lu, un événement sera envoyé et la fonction callback sera opérée.



QUELQUES EXEMPLES

RENCONTRÉS FRÉQUEMMENTS

JS

```
// Lecture d'un fichier
readFile('./data/poem.txt', (err, content) => {
  if (err) throw err
  console.log('Poem : ', content)
})

// Écouteur sur l'événement "click"
let submitBtn = document.getElementById("submit")
submitBtn.addEventListener("click", () => {
  console.log("Vous venez de cliquer sur le bouton [Envoyer]")
})
```

main.js

Le système de callback permet d'avoir un code non bloquant. Une fois le fichier "poem.txt" lu, un événement sera envoyé et la fonction callback sera opérée.

ÉCRIRE LA FONCTION MAP

COMPRENDRE PAR L'EXEMPLE

JS

```
let marks = [6, 18, 15.5, 11]

function map(arr, callback) {
  let result = []
  for(const element of arr) {
    result.push(callback(element))
  }

  return result
}

// Ajout d'un point bonus pour chaque note
let marksWithBonus = map(marks, element => { return element + 1 })
console.log(marksWithBonus)
```

main.js

[7, 19, 16.5, 12]

SORTIE CONSOLE



TRAVAUX PRATIQUES



FONCTION FILTER

En vous inspirant de la méthode sur tableau ".filter", écrire la fonction filter prenant en paramètre un tableau et un callback.

En vous inspirant de la méthode sur tableau ".reduce", écrire la fonction reduce prenant en paramètre un tableau et un callback. La valeur initiale sera le premier élément du tableau.

JS

```
let sentence = ["The", "Red", "Horse"]  
console.log(sentence.reduce((a, i) => `${a} ${i}`))  
// The red horse
```

main.js



FONCTION REDUCE



PROMESSE

POSER DES QUESTIONS

PART. I - TROIS NIVEAUX DE CALLBACK

JS

```
const ReadLine = require('readline')

const rl = ReadLine.createInterface({
  input: process.stdin,
  output: process.stdout
})

const main = () => {
  rl.question('What\'s your first name? ', response => {
    console.log(`Your first name is ${response}`)

    rl.question('What\'s your last name? ', response => {
      console.log(`Your last name is ${response}`)

      rl.question('How old are you? ', response => {
        console.log(`You are ${response} yo`)
        rl.close()
      })
    })
  })
}

main()
```

main.js

..

```
What's your first name? Aunim
Your first name is Aunim
What's your last name? Hassan
Your last name is Hassan
How old are you? 30
You are 30 yo
```

SORTIE CONSOLE

POSER DES QUESTIONS

PART. II – UN NIVEAU DE CALLBACK, MAIS UN SENS DE LECTURE INVERSÉ

JS

```
const main = () => {  
  const readAge = age => {  
    console.log(`Your age is ${age}`)  
    rl.close()  
  }  
  
  const readLastName = lastName => {  
    console.log(`Your last name is ${response}`)  
    rl.question('How old are you?', readAge)  
  }  
  
  const readFirstName = firstName => {  
    console.log(`Your first name is ${response}`)  
    rl.question('What\'s your last name? ', readLastName)  
  }  
  
  rl.question('What\'s your first name? ', readFirstName)  
}  
  
main()
```

main.js

SENS DE LECTURE

```
What's your first name? Aunim  
Your first name is Aunim  
What's your last name? Hassan  
Your last name is Hassan  
How old are you? 30  
You are 30 yo
```

SORTIE CONSOLE

CALLBACK HELL

UNE CHOSE QU'AUCUN DÉVELOPPEUR NE SOUHAITE RENCONTRER



OBJET PROMISE

AVANTAGES

Une promesse est un objet [Promise] qui représente la complétion ou l'échec d'une opération asynchrone.

Les promesses sont aujourd'hui utilisées par la plupart des API modernes. Il est donc important de comprendre comment elles fonctionnent et de savoir les utiliser pour optimiser son code.

PROMESSE

Un objet pratique pour la gestion de l'asynchrone

- ✓ Encapsule l'asynchrone
- ✓ Système de gestion des erreurs
- ✓ Peut se chaîner

```
let p = new Promise((resolve, reject) => {  
  let sum = 12 + 8  
  
  if (sum === 20) resolve('Success')  
  else reject('Failed')  
})  
  
p.then(message => { console.log(message) })  
  .catch(message => { console.log(message) })
```

EXEMPLE DE PROMESSE



POSER DES QUESTIONS

UTILISER LES PROMESSES

Grâce à l'utilisation des promesses, nous évitons le "callback hell" en plus de conserver le sens de lecture du haut vers le bas.

```
const askFirstName = () => {
  return new Promise(resolve => {
    rl.question('Quel est ton prénom ? ', resolve)
  })
}

const askLastName = () => {
  return new Promise(resolve => {
    rl.question('Quel est ton nom ? ', resolve)
  })
}

const askAge = () => {
  return new Promise(resolve => {
    rl.question('Quel âge as-tu ? ', resolve)
  })
}

const main = () => {
  askFirstName()
    .then(firstName => { console.log(`Ton prénom est ${firstName}`) })
    .then(() => askLastName())
    .then(lastName => { console.log(`Ton nom est ${lastName}`) })
    .then(() => askAge())
    .then(age => {
      console.log(`Tu as ${age}`)
      rl.close()
    })
}

main()
```




TRAVAUX PRATIQUES



LE QUESTIONNAIRE

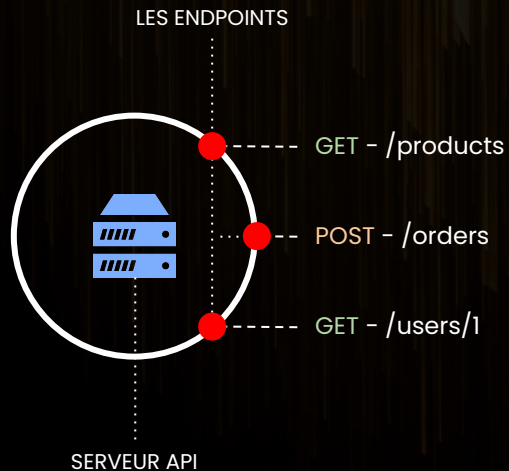
Reprendre l'exemple du questionnaire et ajouter une quatrième question en utilisant les promesses.



API

FONCTIONNEMENT D'UNE API REST

API : APPLICATION PROGRAMMING INTERFACE



L'API est une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

Nous pouvons faire l'analogie avec une télécommande [outil de requête normalisé] et un téléviseur proposant différentes chaînes, l'API et ses endpoints.



JSON PLACEHOLDER

COMPRENDRE LE FONCTIONNEMENT D'UNE API / FAUSSE API REST

ENDPOINTS

NOMBRE DE RESSOURCES

<https://jsonplaceholder.typicode.com/posts>

100 articles

<https://jsonplaceholder.typicode.com/comments>

500 commentaires

<https://jsonplaceholder.typicode.com/albums>

100 albums

<https://jsonplaceholder.typicode.com/photos>

5000 photos

<https://jsonplaceholder.typicode.com/todos>

200 todos

<https://jsonplaceholder.typicode.com/users>

10 utilisateurs

A noter que les ressources ont des relations, les articles ont des commentaires et les albums ont des photos.

JSON PLACEHOLDER



ROUTES REST

CAS DE LA RESSOURCE ARTICLE

VERBE HTTP	URL - ROUTE	ACTION	DESC.
GET	/articles	Consulter	Liste tous les articles
GET	/articles/create	Consulter (<i>Créer</i>)	Affiche le formulaire de création d'articles
POST	/articles	Ajouter	Ajout de l'article après soumission du formulaire
GET	/articles/:id	Consulter	Affiche l'article spécifiée par l'id
GET	/articles/:id/edit	Éditer	Affiche le formulaire d'édition de l'article spécifiée par l'id
PATCH	/articles/:id	Modifier	Modifie l'article spécifiée par l'id
PUT	/articles/:id	Modifier	Remplace l'article spécifiée par l'id
DELETE	/articles/:id	Supprimer	Supprime l'article spécifiée par l'id





FONCTION FETCH

FONCTION FETCH

LIRE UN FICHIER JSON DANS LE CLOUD

JS

```
let url = 'https://...'  
  
fetch(url)  
  .then((response) => { return response.json() })  
  .then((data) => {  
    // Affichage des données  
    console.log(data)  
  })
```

js/main.js

La fonction fetch est comprise par NodeJS depuis la version 17.5.

En précisant une url valide dans l'exemple ci-contre, on peut accéder à l'aide de "fetch" à un fichier dans le cloud.

—



FONCTION FETCH

TRAVAILLER AVEC UNE API

SIGNATURE DE LA FONCTION `fetch(...)`

`fetch(resource, options)`

LISTE DES OPTIONS

`method`
`headers`
`body`
`mode`
`credentials`
`cache`
`redirect`
`referrer`
`referrerPolicy`
`integrity`
`keepalive`
`signal`

```
// POST request using fetch()
fetch("https://jsonplaceholder.typicode.com/posts", {

  // Adding method type
  method: "POST",

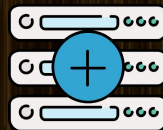
  // Adding body or contents to send
  body: JSON.stringify({
    title: "foo",
    body: "bar",
    userId: 1
  }),

  // Adding headers to the request
  headers: {
    "Content-type": "application/json; charset=UTF-8"
  }
})

// Converting to JSON
.then(response => response.json())

// Displaying results to console
.then(json => console.log(json));
```





JSON SERVER

JSON-SERVER

AVOIR UNE FAUSSE API REST RAPIDEMENT

```
npm install -g json-server
```

1. Installer globalement json-server.

```
https://github.com/aunimhsn/json-server/blob/main/app/db.json
```

2. Créer le fichier db.json et suivre le lien pour avoir le contenu.

```
json-server --watch db/db.json
```

3. Lancer le serveur local de notre fausse API REST.

```
http://localhost:3000/cars
```

4. Ouvrir votre navigateur et se rendre sur le lien du serveur local.

CLIENT REST

OUTIL INDISPENSABLE POUR TRAVAILLER AVEC LES APIs



Insomnia

Kong Inc.



Postman

Postman Inc.

Un client REST est un outil de développement et de débogage.

Il permet de personnaliser et garder en mémoire les requêtes envoyées à une API REST.



INSOMNIA

EXEMPLE D'UTILISATION

GET http://localhost:3000/cars/1

Send

Preview

```
--  
  
{  
  "id": 1,  
  "brand": "Kia",  
  "model": "Rio III",  
  "color": "white",  
  "year": 2014  
}
```



INSOMNIA




RAPID API

UNE COLLECTION D'APIs GRATUITE

AVOIR DES IDÉES DE PROJETS


Recommended APIs

APIs curated by RapidAPI and recommended based on functionality offered, performance, and support!




API-FOOTBALL

+950 football leagues & cups. Livescore (15s), live & pre-match odds, events, line-ups, coaches, players.


Verified 

🔒 10 ⌚ 354 ms ✓ 100%




VACCOCOVID - coronavirus, vaccine and treatment tracker

VACCOCOVID.LIVE is a comprehensive up-to-date Vaccine tracker, COVID-19


Verified  Official

🔒 9.8 ⌚ 580 ms ✓ 100%




SendGrid

Welcome to SendGrid's Web API v3! This API is RESTful, fully featured, and easy to integrate with.


Verified  Official

🔒 9.8 ⌚ 305 ms ✓ 100%



Referential


The fastest API to access countries, states, cities, continents, dial and zip codes in up to 20

Verified 

🔒 9.8 ⌚ 39 ms ✓ 94%


Popular APIs

APIs that are popular and frequently used on RapidAPI!




Recipe - Food - Nutrition

The spoonacular Recipe - Food - Nutrition API gives you to access to thousands


Verified 

🔒 9.9 ⌚ 681 ms ✓ 100%




uNoGS

uNoGS (unofficial Netflix online Global Search) allows anyone to search the global Netflix catalog.


Verified 

🔒 9.8 ⌚ 8591 ms ✓ 93%




Bionic Reading

Bionic Reading® is a new method facilitating the reading process by guiding the eyes through text with


Verified  Official

🔒 9.5 ⌚ 1233 ms ✓ 94%



Trip Purpose Prediction


Understand the reason for a trip with Amadeus AI APIs. The Trip Purpose Prediction API uses AI

Verified 

🔒 8.9 ⌚ 193 ms ✓ 100%

Grâce à Rapid API, des centaines d'APIs sont mises à disposition, avec une indication sur les endpoints fournis.

Il devient alors aisé de trouver des idées de mini-projets en travaillant avec une API qui nous plaît.

 RAPID API

43

CHOIX D'UNE API

LYRICS FINDER : OBTENIR LES PAROLES D'UNE CHANSON

ENDPOINTS

- | | |
|-----|-------------------------|
| GET | Get artist's song lyric |
| GET | Get all artists |
| GET | Get all artist's musics |

```
{  
  "songName": "Aerials",  
  "songLyric": "Life is a waterfall We're one ..."  
}
```

EXEMPLE DE RÉPONSE API DU ENDPOINT "Get artist's song lyric"



TRAVAUX PRATIQUES

Choisir parmi les APIs proposées par Rapid API, une API avec laquelle vous allez travailler. Obtenir des données et les afficher sur une page HTML.

N'hésitez pas à aller plus loin en créant un formulaire permettant d'interagir avec l'API.



RAPID API



ALLER PLUS LOIN

ALLER PLUS LOIN

CONTINUER D'APPRENDRE

JS

Manipuler le DOM en JavaScript

JS

La POO en JavaScript



NodeJS

TS

TypeScript



MERCI DE VOTRE ATTENTION