



# Introduction à Deno 1.30

Présenté par Logan TANN  
Pour OnePanthéon





# Audience

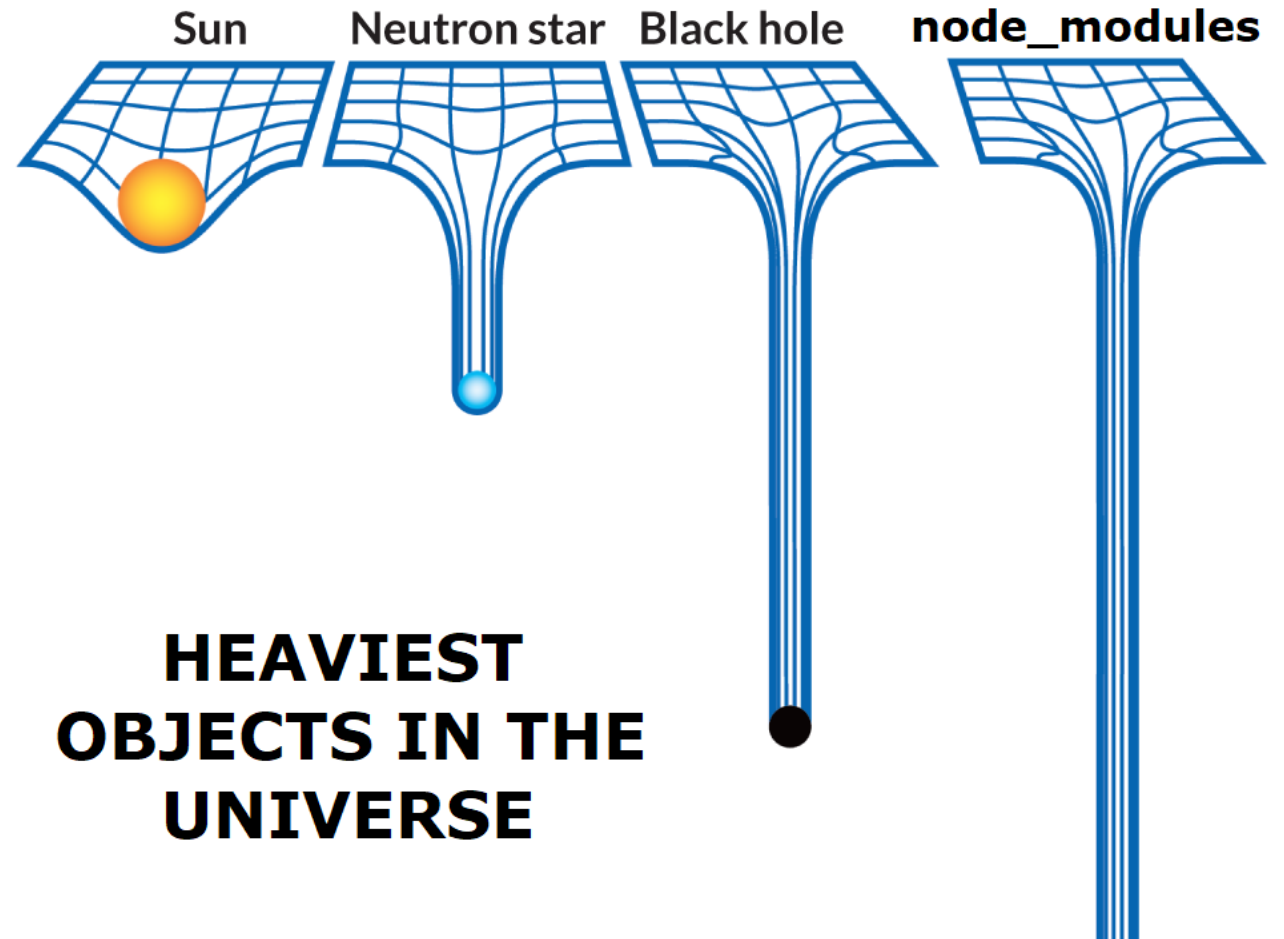
- Pour les curieux du web en général 🧐 -

👉 En particulier : les Typescript Enthousiasts, ou pour ceux qui ont réalisé un projet avec NodeJs.

🕶 Note : Pour comprendre les exemples, une expérience en Javascript moderne est recommandée.

# Un peu d'histoire

> De Node.JS à Deno





# Les débuts de Node.JS

- ☹ Ryan Dahl est frustré des limitations d'Apache
- Difficultés d'implémenter des applis en temps réel
  - Mécanismes de prog évènementielle et concurrente pas assez efficace.



JSConf 2009 : Il présente un projet perso, qui deviendra Node.JS

👉 Bénéficie aujourd'hui d'une grande popularité



# Un projet devenu trop populaire

10 ans plus tard, il présente à la JSConf 2019 :  
« 10 Choses que je regrette à propos de Node.js »



Il évoque par exemple :

- Node n'est pas assez proche des standards JS des navigateurs
- NPM souffre de plein de défauts

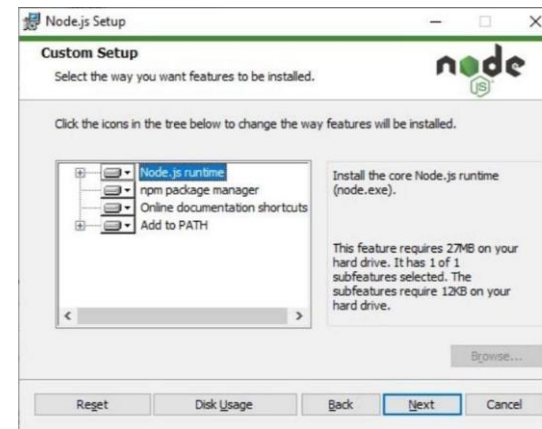
# Prise en main de Deno

Installation et découverte en exemples





# Installation



**1 - Juste le binaire à installer depuis github**



# Installation



1 - Juste le binaire à installer depuis github

## 2 – Installeur automatique en ligne de commande

Shell (Linux & Mac) :

```
$ curl -fsSL https://deno.land/x/install/install.sh | sh
```

Windows :

```
$ irm https://deno.land/install.ps1 | iex
```

Docker :

```
$ docker run -it denoland/deno:1.28.1
```





# Exemple 1 : Hello World

Consigne :

1. Faire un programme qui affiche "Hello World".
2. Faire un programme qui affiche le contenu du fichier "canard.txt"

```
~/p/A/0/solutions main ↑1 $ deno run --allow-read .\01_HelloWorld.js 4ms
Hello World !

>(. )__ <(. )__ =(. )__
(____/ (____/ (____/
```



# Exemple 1 : Hello World

Solution :

```
console.log("Hello World !");

// support du top level await
// ... et aussi un système de permissions vraiment bien foutu.
// avant ça faisait un throw, mais depuis la version 1.28, ça a changé
const texte = await Deno.readFile("canard.txt");

console.log(texte);
```



## Exemple 2 : Appel d'API

1. Voici une API très sympa : <https://catfact.ninja/fact>
2. Faire appel à cette api en utilisant Typescript et fetch
3. Ne rendre l'affichage du message que si l'utilisateur vous autorise

```
~/p/A/0/solutions main ↑1 $ deno run --allow-net .\02_ApiCall.ts  
Voulez vous écouter un fait sur les chats ? [y/N] y
```

```
Voici un fait sur les chats :  
> 70% of your cat's life is spent asleep.
```



# Exemple 2 : Appel d'API

Solution :

```
// typescript et ES Module support out of the box
import {CAT_ENDPOINT, CatResponse} from "../catfact.ts";

// Les APIs web sont supportées :D
const request = await fetch(CAT_ENDPOINT);
const response: CatResponse = await request.json();
const message = response.fact;

const agreed: boolean = confirm("Voulez vous écouter un fait sur les chats ? ");
if (agreed) {
    console.log("\nVoici un fait sur les chats : ");
    console.log(">", message);
} else {
    console.log("\nBon temps pis ... ");
}
```



# Exemple 3 : Serveur web et NPM

1. Importer la librairie Aqua (<https://deno.land/x/aqua@v1.3.5/mod.ts>) et faire un hello world
2. Importer une librairie NPM (npm:qrcode@1.5.0) et afficher un QR Code
3. Utiliser un import map pour rendre l'importation des dépendances plus maintenables

```
~/p/A/0/solutions main ↑1 $ deno run --allow-net .\03_WebServer.ts
Démarrage du serveur...
Serveur web démarré sur http://localhost:3001
[]
```



The screenshot shows a web browser window with the address bar set to 'localhost:3001'. The page content displays 'Hello world' in a large, bold, serif font. Below this, the text 'Un super site :' is followed by a square QR code.

```

import Aqua from "aqua";
import QRCode from "qrcode";

console.log("Démarrage du serveur...");

const aqua = new Aqua(3001);
aqua.get("/", async () => {
  const url = await QRCode.toDataURL('https://www.onepantheon.fr/');
  return `
    <H1>Hello world</H1>
    <p>Un super site : <br>
    |   
    </p>
  `;
});

console.log("Serveur web démarré sur http://localhost:3001");

```

## Exemple 3 : Appel d'API (Solution)

01 - Introduction > solutions > {} deno.json > ...

```

1  {
2    "imports": {
3      "aqua": "https://deno.land/x/aqua@v1.3.5/mod.ts",
4      "qrcode": "npm:qrcode@1.5.1"
5    }
6  }

```



# Récap



## Avantages :

1. Expérience développeur très satisfaisante.
2. Un binaire standalone avec plein d'outils built-in
3. Veut faire changer les habitudes.



## Inconvénients :

1. Difficile de faire changer les habitudes
2. Librairies souvent peu documentées ou supportées
3. VsCode est le seul éditeur fiable

=> Génial pour des scripts d'automatisation ou des petits projets



# Pour aller plus loin...

- **Documentation** : <https://deno.land/manual@v1.28.1/introduction>
- **Nouveautés Deno** : <https://deno.com/blog/>
- **DenoDeploy** : plateforme de déploiement on the edge (gratuit)
- **Fresh** : framework web fullstack concurrent à React
- **Deno News** : Veille numérique officielle sur Deno
  
- **Discordeno** : Lib pour faire des bots discord
- **Caviar** : Game Engine basé sur le WebGPU
- **Awesome deno** sur github





# Prochaine séance

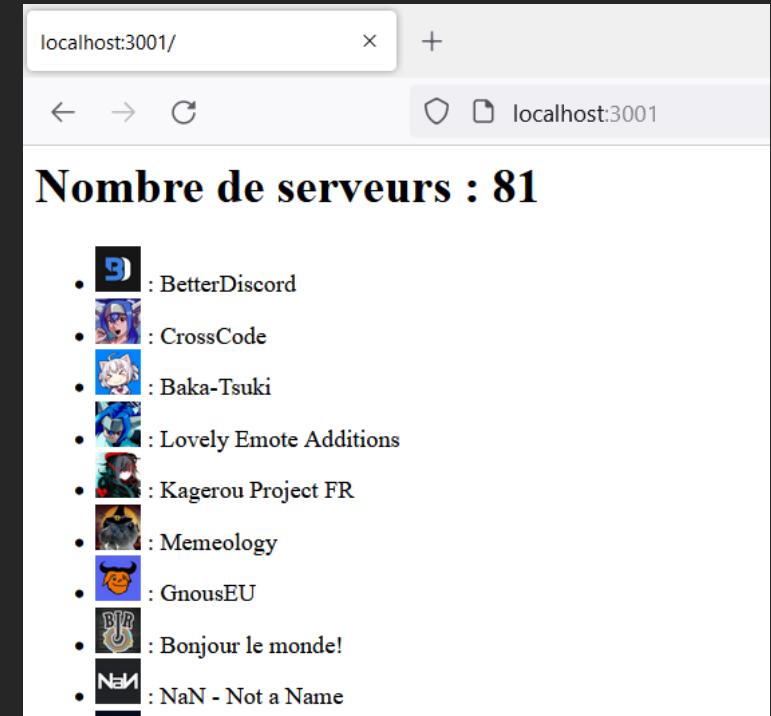
## - Pratique avec le protocole Oauth 2 -

Sujet : Faire une application listant les différents serveurs discord d'une personne, et la déployer en edge functions.

Principe (selon la faisabilité) :

- Un mélange entre cours et TP (prepared kata)
- Une correction distribuée en PDF

Date : 10 février (dans une semaine) si tout va bien





# Des questions ?

Merci pour votre écoute !

Lien github vers les slides et le code :  
<https://github.com/OnePantheon/Ateliers-deno/tree/main/01 - Introduction>

