

# Programmation Web – Avancé

JavaScript & Node.js

Partie 15 : Node.js, les modules côté serveur & les packages

Version 2020



*Presentation template  
by [SlidesCarnival](#)*

# Introduction à l'utilisation de JS côté serveur : Node.js, les modules et les packages



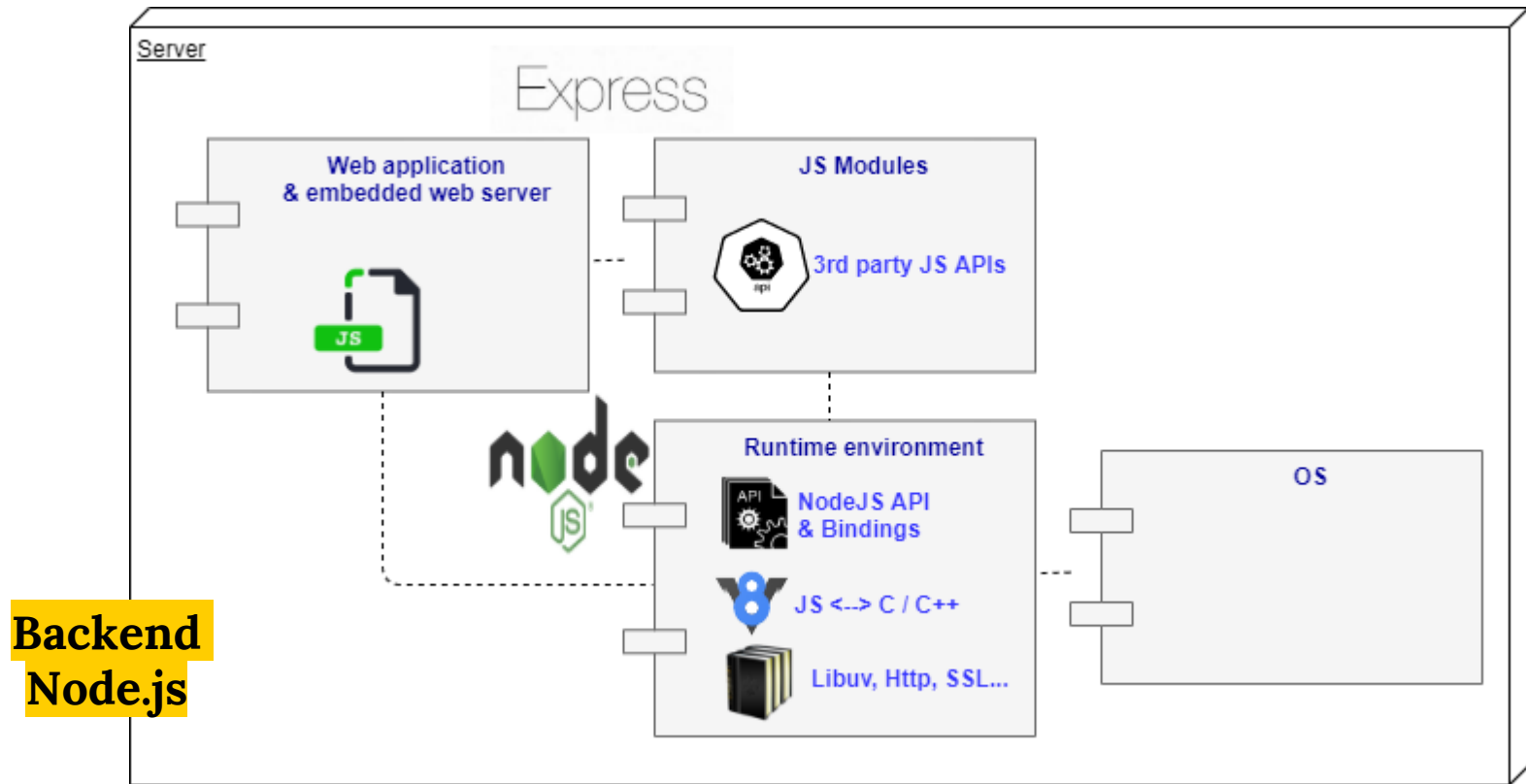
Comment utiliser le JS pour réaliser une application de type serveur ?



## Introduction à Node.js



- Environnement serveur open source
- Création d'outils et applications côté serveur en JS
- Utilisation optimale des ressources des serveurs sans dépendance à un serveur http externe
- Multiplateforme (Windows, Linux, Mac...).





## JS côté serveur : quel Software utiliser pour développer ?

- Installation de l'environnement **Node.js** LTS [\[54.\]](#)
- Pour écrire et exécuter vos scripts :
  - **VS Code**
  - N'importe quel terminal : **Git Bash**
- Pour gérer différentes versions de votre code et différentes machines : **Git, Gitlab / Github** et **VS Code**



## JS côté serveur : quel Software utiliser pour développer ?

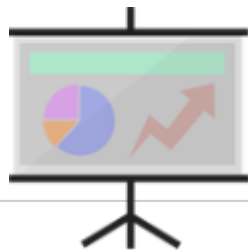
---

- Pour consommer vos web APIs rapidement :  
**REST Client** [\[77.\]](#) comme extension de **VS Code**



## Exécution et déploiement d'applications web sur le cloud : quels services ?

- ⦿ Pour développer & exécuter vos scripts sur le cloud: **CodeSandbox**
- ⦿ Pour déployer vos applications : **Netlify** ou **Heroku**



## JS côté serveur : Où mettre votre code Node.js ?

- A. Directement via l'interpréteur de commandes de Node au sein d'un terminal : **node**
  - **PRESENTATION** : vérifier l'installation de Node.js & opérations mathématiques de base





## JS côté serveur : Où mettre votre code Node.js ?

---

B. Via l'interpréteur de commandes de Node au sein d'un terminal et d'un script externe :

- Start : **node script\_name** (.js optionnel)
- Stop : **CTRL + c**



## Node.js

- **DEMO** : Appel d'une méthode au sein d'un script externe via Node.



## Introduction aux modules sous Node.js

- Un module = librairie JS fournissant des fonctions et / ou des objets
- Inclure un module intégré ou installé :

```
// module integrated to the runtime environment
let http = require("http");
// module following package installation
let shortid = require("shortid");
```



## Introduction aux modules sous Node.js

- Inclure un propre module :

```
// import default export
let vehicle = require("./Car.js") ;

// import Named object
let vehicle = require("./Car.js") ;
// use of external Named object
let dacia = new vehicle.Car("Dacia", "Sandero");
// import Named object & destructuring assignement
let {Car} = require("./Car.js") ;
// use of external Named object following destructuring assignement
dacia = new Car("Dacia", "Sandero");
```



## Introduction aux modules sous Node.js

- Création d'un module :
  - Création d'un fichier **nomModule.js**
  - Au sein du fichier **nomModule.js**, utilisation de **exports** pour rendre les propriétés et méthodes disponibles en dehors du fichier module

```
// default export
module.exports = Car;

// export as named object
exports.Car = Car;
```



## Node.js & utilisation de modules intégrés

- **DEMO** : Serveur web minimaliste avec Node.js fonctionnant sur le port 777 et mettant à disposition les fichiers présents dans le répertoire **/public**



- Lecture des propriétés données dans une URL: **url.parse()**
- Lecture de la propriété « chemin » données dans un objet URL (myURL) : **myURL.pathname**
- Lecture asynchrone d'un fichier : **fs.readFile()**
- Chemin absolu où Node.js est appelé : **\_\_dirname**



## Introduction aux packages sous Node.js

---

- Tous les fichiers nécessaires pour un module sont contenus dans un package
- Installer un package (via terminal) : **npm install package\_name**
- NPM : gestionnaire de package [\[55.\]](#)



## Introduction aux packages sous Node.js

- Packages associés à une app : **package.json**
  - Création sur base de questions ou par défaut de **package.json** : **npm init** ou **npm init -y**

```
{ "name": "node-modules",  
  "dependencies": {  
    "mime": "^2.4.6",  
    "shortid": "^2.2.15"  
  },  
  "devDependencies": {},  
  "scripts": {  
    "start": "node node-modules.js"  
  },  
  "author": "e-Baron"}
```





## Introduction aux packages sous Node.js

- Arbre exact des dépendances installées :  
**package-lock.json**
  - Génération automatique pour chaque opération modifiant **node\_modules** ou **package.json**

```
{  
  "requires": true,  
  "lockfileVersion": 1,  
  "dependencies": {  
    "mime": {  
      "version": "2.4.6",  
      "resolved": "https://registry.npmjs.org/mime/-/mime-2.4.6.tgz",  
      ...  
    }  
  }  
}
```



## Introduction aux packages sous Node.js

---

- Localisation d'un module par Node : recherche parmi tous les chemins spécifiés dans `module.paths` (`node_modules...`)
- Mise à jour des packages vers leur dernière version [\[102.\]](#)



## Introduction aux packages sous Node.js

- **DEMO** : Serveur web minimaliste avec Node.js, des modules et des packages
  - Génération aléatoire d'un ID à l'aide d'un package NPM : **shortid**
  - Gestion du MIME type à l'aide d'un package
  - Export & import du module Car