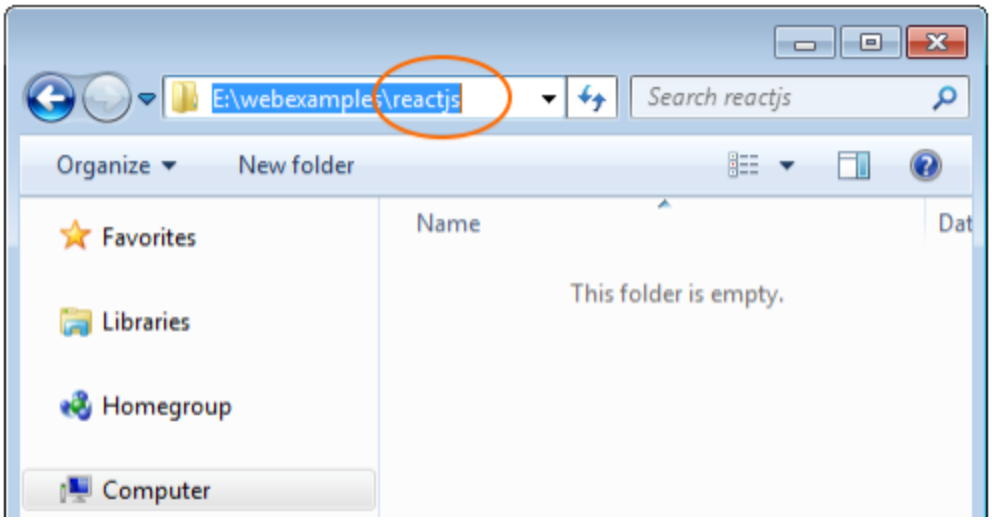
HTTP SERVER AVEC NODEJS

En créer un HTTP Server avec NodeJS, on pourra accéder aux sources des données statiques telles que HTML, Javascript, CSS,... dans le dossier via http, tel que :

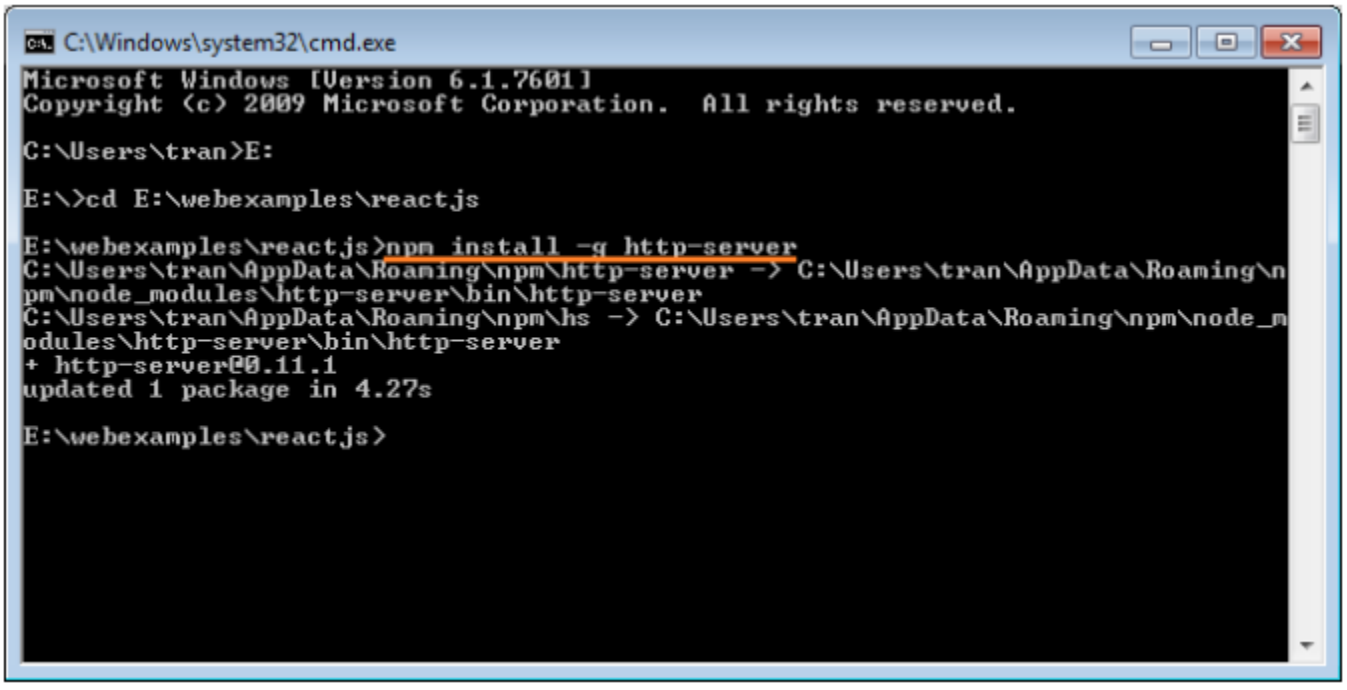
http://localhost:8080/abc.html

http://localhost:8080/abc.js

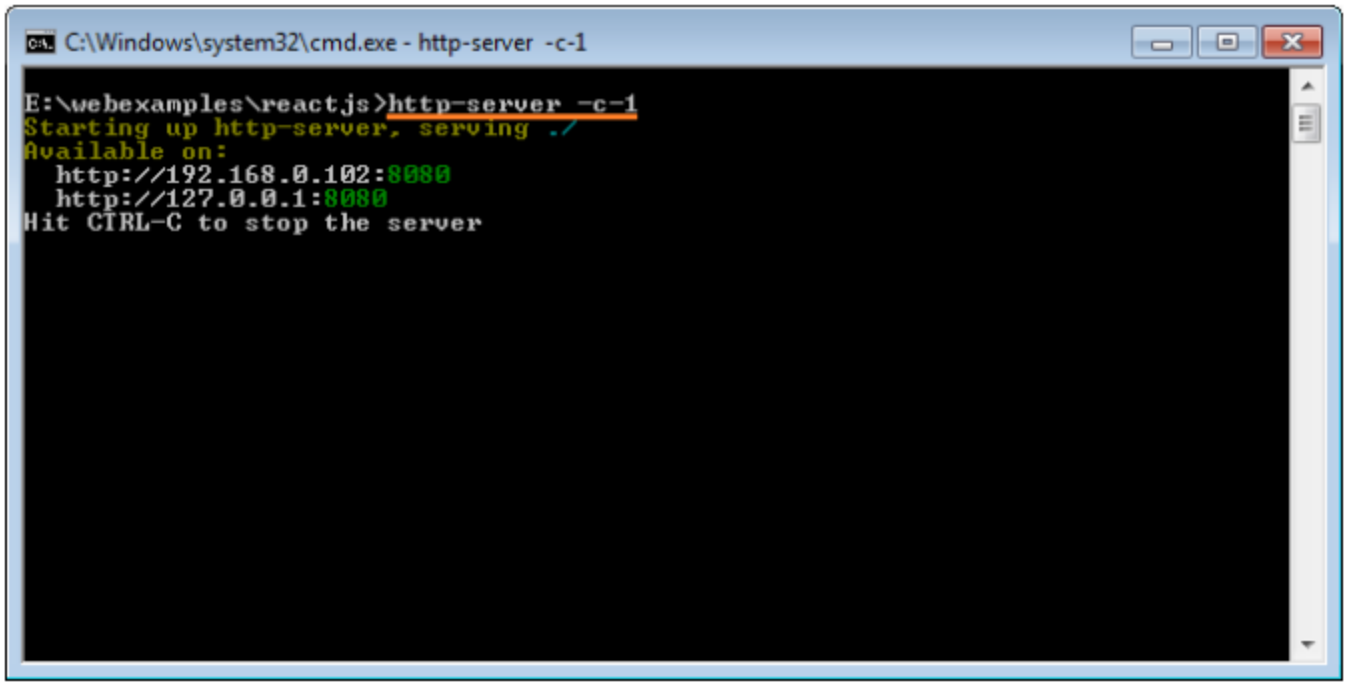
Créons alors un autre dossier, par exemple :



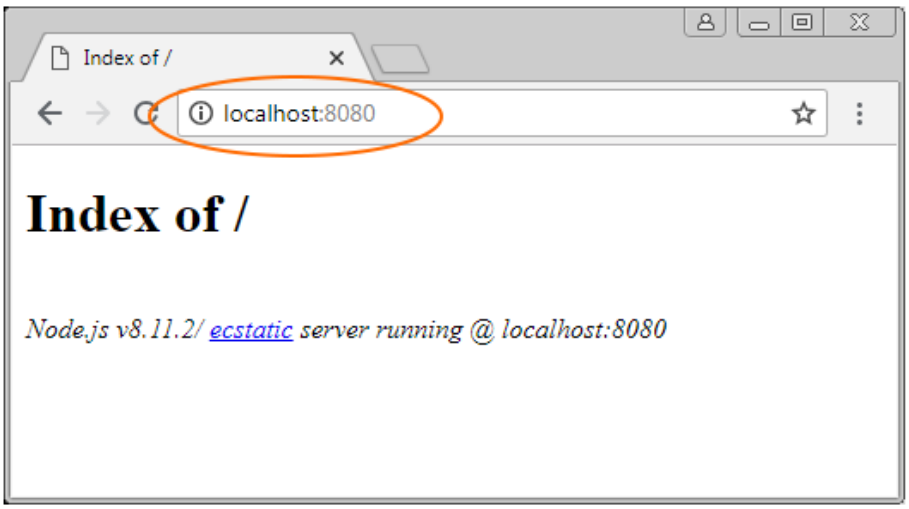
Puis exécutez la commande pour installer http-server :



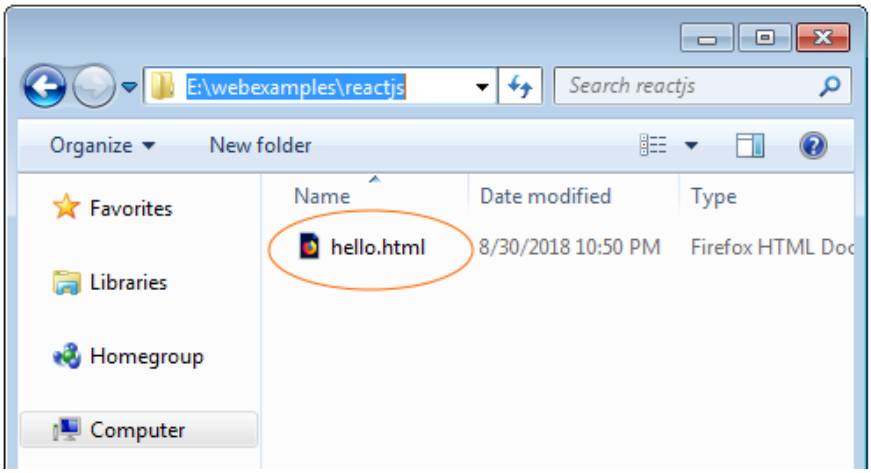
Ensuite, démarrez HTTP-Server :

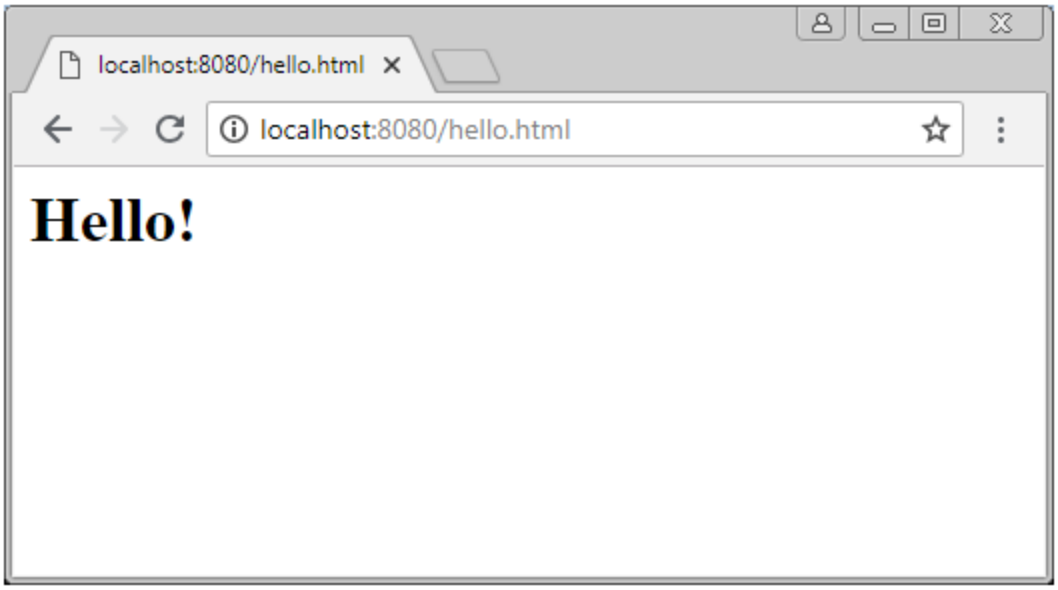


Le HTTP Server a été démarré, il écoute au port 8080. Vous pouvez le vérifier en accédant au lien ci-dessous :



Créez un fichier HTML dans le dossier, et on pourrait l'accéder via HTTP :





NODEJS MODULE

Le module NodeJS est une bibliothèque Javascript. C'est un ensemble de fonctions (function), d'objets et de variables que vous pouvez utiliser dans vos applications. L'utilisation Module permet de simplifier l'écriture de code et de le gérer dans votre application. Normalement, chaque module sera écrit dans un fichier séparé.

Le NodeJS intègre de nombreux Modules. Il a des bibliothèques standards pour que vous puissiez développer des applications.

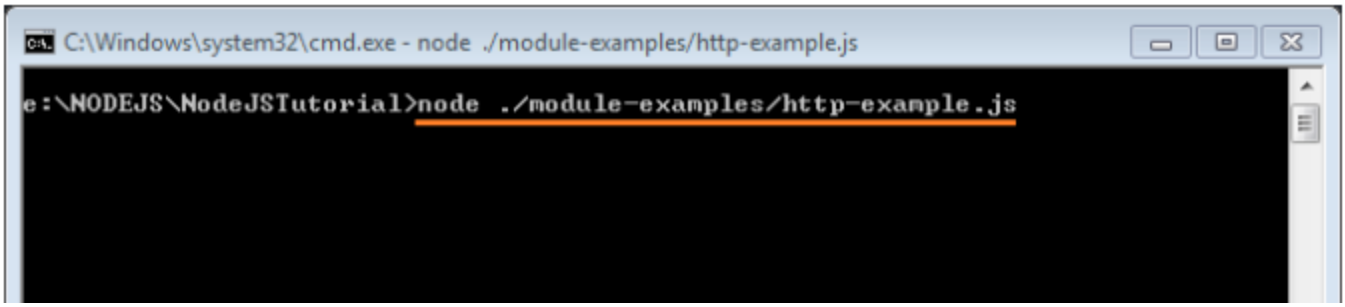
Module	Description
assert	ournit un ensemble des **assertion tests** (tests d'assertion)
buffer	Pour gérer les données binaires (binary data).
child_process	Pour exécuter un processus enfant (child process)
cluster	Pour diviser un seul processus (process) Node en plusieurs processus
crypto	Pour gérer les fonctions cryptographiques **OpenSSL** (OpenSSL cryptographic functions)
dgram	Fournit l'implémentation de **UDP sockets**
dns	Effectuer des recherches (lookups) et des fonctions de résolution (resolution) **DNS**
events	Pour gérer les événements (events)
fs	Pour gérer le système de fichiers
http	Pour faire **Node.js** agir comme un **HTTP server**.

https	Pour faire **Node.js** agir comme un **HTTPS server**.
net	Pour créer des **server** et des **client**
os	Fournit des informations sur le système d'exploitation
path	Pour gérer des chemins de fichiers (file paths).
querystring	Pour gérer les chaînes de requête de URL
readline	Pour gérer les flux des données (data streams) lisibles une ligne (line) à la fois
stream	Pour gérer les flux des données (streaming data)
string_decoder	Décoder (decode) les objets du tampon (buffer objects) en chaînes
timers	Pour exécuter une fonction **Javascript** après un nombre donné de temps
tls	Pour implémenter des protocoles **TLS & SSL**.
tty	Fournit des classes utilisées par **text terminal**.
url	Pour analyser (parse) les chaînes de **URL** (URL strings)
util	Pour accéder aux fonctions utilitaires (Utility functions).
v8	Pour accéder aux informations sur le moteur **V8 JavaScript engine**.
vm	Pour compiler (compile) du code **JavaScript** dans la machine virtuelle (Virtual machine)
zlib	Pour compresser ou décompresser des fichiers.

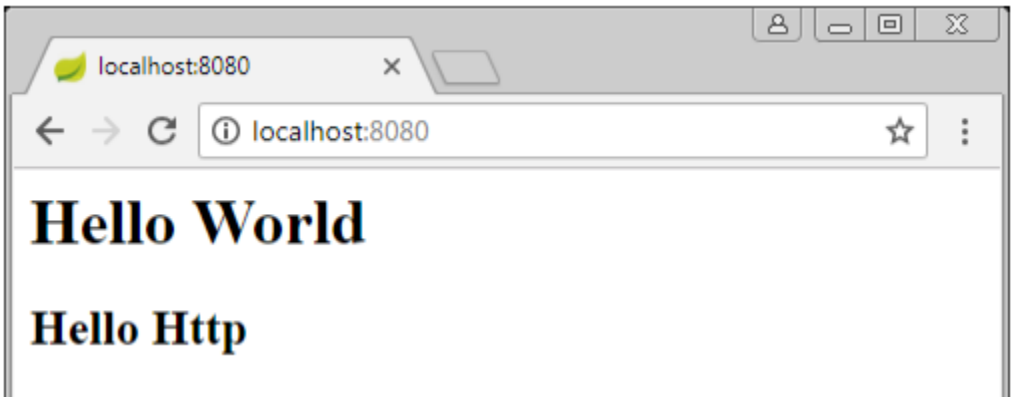
Afin d'utiliser un module, utilisez la fonction require(moduleName). Par exemple, http est un module qui oblige NodeJS à fonctionner comme un HTTP Server.

Exemple : Sur le projet NodeJS, créez un dossier module-examples pour contenir les fichiers Javascript

- Dans la fenêtre CMD



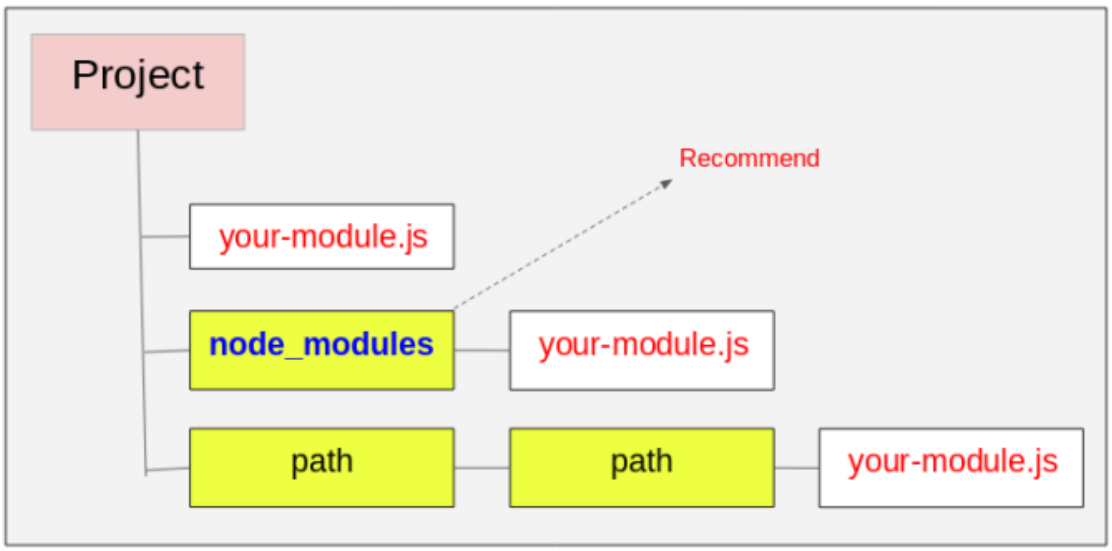
- Dans le navigateur :



MODULE PERSONNALISE

Le Module est un fichier Javascript. Merci à noter la location de ce fichier parce qu'elle affecte à la manière que vous l'utiliser. Deux choix de sélection la location de ce fichier proposés :

- Le fichier Module est mis dans le répertoire nommé node_modules (Le sous répertoire du projet). Ceci est la location recommandée.
- Le fichier Module est mis dans un de vos répertoires, pas dans le node_modules



Remarques :

- Le NodeJS vous recommande de mettre vos fichiers Module dans le dossier node_modules du projet.

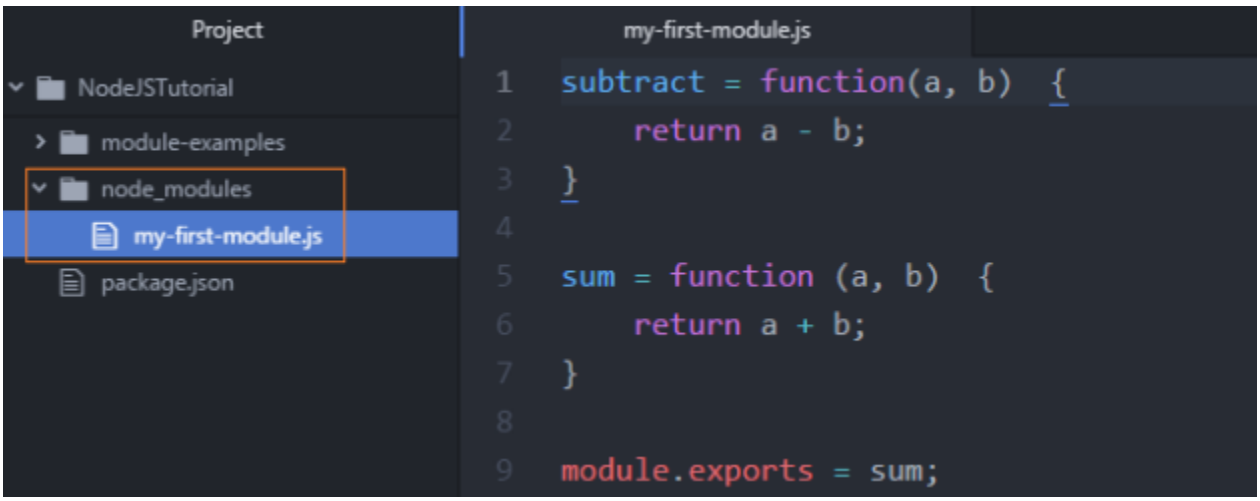
- La syntaxe de export (exportation) quelque chose dans le fichier Javascript :

```
exports = something;

// or:

module.exports = something;
```

Exemple :




```
//module-examples/test-my-first-module.js

var sum = require("my-first-module.js");

console.log("Sum 100 and 150");

var c = sum(100, 150);

console.log("Value = " + c);
```

Exécutez l'exemple : > node ./module-examples/test-my-first-module.js

EXPORTATIONS PLUSIEURS CHOSES :

Un fichier module peut contenir plusieurs fonctions (function), plusieurs variables (variable) et plusieurs classes (class). Cet exemple indique comment export (exporter) plusieurs choses dans ce fichier.

Exemple :

```
//node_modules/my-second-module.js

var COPY_RIGHT = "Export";

// function
var multiple = function(a, b)  {
   return a * b;
};

// classe
var Person = class {
   constructor(name, age) {
     this.name = name;
     this.age = age;
   }

   showInfo()  {
     console.log("name="+ this.name+", age="+ this.age);
   }
};

// Exportation
var somethingToExport = {
  Person: Person,
  multiple : multiple,
  COPY_RIGHT : COPY_RIGHT
};

module.exports = somethingToExport;
```

```js
// module-examples/test-my-second-module.js

var module2 = require("my-second-module.js");

console.log("Utilisation module class Person :");

var p = new module2.Person("Smith", 20);

p.showInfo();

console.log(" -- ");
console.log("Utilisation module fonction multiple :");

var c = module2.multiple(20, 5);

console.log("Value= "+ c);

console.log(" -- ");
console.log("Utilisation module Variable :");

var copyRight = module2.COPY_RIGHT;

console.log("Copy right: " + copyRight);
```