Node.js

Étude du code de ce projet

Si des problèmes avec Visual Studio Code ajouter dans le .vcode de votre projet :

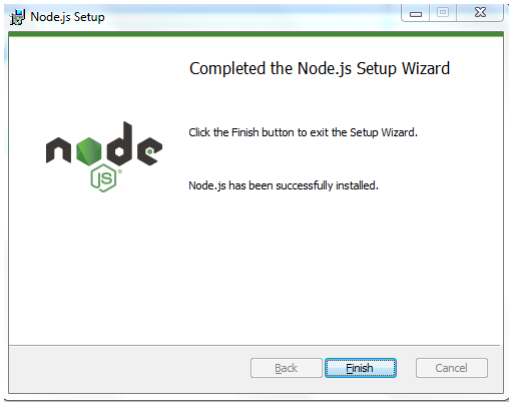
Exemple : module not found

```
{
    "typescript.tsdk": "node_modules/typescript/lib",
    "git.ignoreLimitWarning": true
}
```

Nota : parfois il faudra démarrer vsc par une nouvelle fenêtre pour que le fichier soit pris en compte.

Installation

https://nodejs.org/fr/



Faut avant tout regarder Typescript :

1. Installer node.js puisque on va utiliser npm l'installateur de packages
2. Installer typescript `npm install -g typescript` -g global donc la commande tsc accessible partout
3. Compiler en ligne de commande `tsc source.tc`

Va convertir source.tc en source.js que vous pouvez alors incorporer dans vos pages.

```typescript
class voiture {
  passagers = 4;
  constructor(personnes : number) {
    this.passagers = personnes;
  }

  mamethode(message : string) {
    console.log(message);
  }
}

var mavoiture = new voiture(2);
```

```javascript
var voiture = (function () {
    function voiture(personnes) {
        this.passagers = 4;
        this.passagers = personnes;
    }
    voiture.prototype.mamethode = function (message) {
        console.log(message);
    };
    return voiture;
})();

var mavoiture = new voiture(2);

```

Quelques références :

https://www.tutorialspoint.com/typescript/typescript_tutorial.pdf (un pdf avec tout)

https://www.w3schools.com/js/ (plutôt pour approfondir javascript)

https://www.tutorialspoint.com/typescript/index.htm

https://www.typescriptlang.org/index.html

http://yahiko.developpez.com/tutoriels/introduction-typescript/

DOM

Typescript

```typescript
class Greeter {
    element: HTMLElement;
    span: HTMLElement;
    timerToken: number;

    constructor (element: HTMLElement) {
        this.element = element;
        this.element.innerHTML += "Il est : ";
        this.span = document.createElement('span');
        this.element.appendChild(this.span);
        this.span.innerHTML = new Date().toUTCString();
    }

    start() {
        this.timerToken = setInterval(() => this.span.innerHTML= new Date().toUTCString(), 500);
    }

    stop() {
        clearTimeout(this.timerToken);
    }

}

window.onload = () => {
    var el = document.getElementById('content');
    var greeter = new Greeter(el);
    greeter.start();
};
```

Javascript

```javascript
var Greeter = (function () {
    function Greeter(element) {
        this.element = element;
        this.element.innerHTML += "Il est : ";
        this.span = document.createElement('span');
        this.element.appendChild(this.span);
        this.span.innerHTML = new Date().toUTCString();
    }
    Greeter.prototype.start = function () {
        var _this = this;
        this.timerToken = setInterval(function () {
            return _this.span.innerHTML = new Date().toUTCString();
        }, 500);
    };
    Greeter.prototype.stop = function () {
        clearTimeout(this.timerToken);
    };
    return Greeter;
})();
window.onload = function () {
    var el = document.getElementById('content');
    var greeter = new Greeter(el);
    greeter.start();
};
```

Création d'un serveur node.js

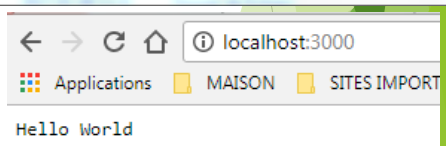
```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

«Arrow functions» en Typescript. En Javascript

```
function (req, res){
  …
}
```

Démarrer le serveur

```
D:\>
D:\>NodeJS_V8.2.0\node ExemplesNodeJS\ex1_node.js
Server running at http://127.0.0.1:3000/
```



Commencer ici : https://www.w3schools.com/nodejs/

Approfondir avec exemples

https://openclassrooms.com/courses/des-applications-ultra-rapides-avec-node-js/node-js-mais-a-quoi-ca-sert

https://nodejs.developpez.com/tutoriels/javascript/debuter-avec-node-js-partie-3/

Pour notre exemple films

Packages à charger

server.js — Exemple _BaseDonnees_CRUD_MYSQLI _AJAX_JSON — Visual Studio Code

Fichier Modifier Sélection Afficher Accéder Déboguer Tâches Aide

EXPLORATEUR

ÉDITEURS OUVERTS
- Bienvenue
- server.js

EXEMPLE _BASEDONNEES_CRUD_...
- .vscode
- BD

Bienvenue | server.js

```
1  var express = require('express');
2  var http = require('http');
3  var session = require("express-session");
4  var bodyParser = require('body-parser');
5  var path    = require("path");
6  var fs    = require("fs");
7  var multer  = require('multer'); //mutipart form data
```

Dans notre projet on va créer un dossier node_modules pour y mettre nos packages.

On va également créer un fichier package.json de suivi des versions de nos packages.

package.json — Exemple _NodeJS_CRUD_MYSQLI _AJAX_JSON — Visual Studio Code

Fichier Modifier Sélection Afficher Accéder Déboguer Tâches Aide

EXPLORATEUR

ÉDITEURS OUVERTS
- Bienvenue
- package.json

EXEMPLE _NODEJS_CRUD_MYSQ...
- .vscode
- BD
- css
- images
- js
- node_modules
- pochettes
- requetes
- ~$le_get_contents_VS_PHP....
- films.html
- package.json
- server.js

Bienvenue | package.json

```
1  {
2    "name": "Exemple _NodeJS_CRUD_MYSQLI _AJAX_JSON",
3    "version": "0.0.0",
4    "private": true,
5    "dependencies": {
6
7    }
8  }
9
```

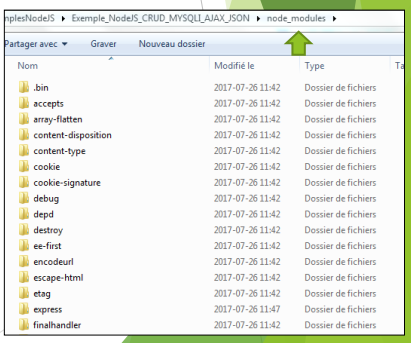
Modules ici

L'option --save indique à NPM d'inclure automatiquement le paquet dans la section des dépendances de votre package.json, ce qui vous permet d'économiser une étape supplémentaire.

Pour installer le framework express la même chose pour tous les autres. Pour enlever un package npm uninstall <nom du package>

```
D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save express
+ express@4.15.3
added 1 package in 1.743s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>
```

```
{
  "name": "Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON",
  "version": "0.0.0",
  "private": true,
  "dependencies": {
    "express": "^4.15.3"
  }
}
```



Les autres modules (packages) aussi

Invite de commandes

```
D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save express-session
+ express-session@1.15.4
added 6 packages in 3.997s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save body-parser
+ body-parser@1.17.2
added 4 packages in 3.405s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save path
+ path@0.12.7
added 4 packages in 3.497s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save fs
+ fs@0.0.1-security
added 1 package in 7.065s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save multer
+ multer@1.3.0
added 21 packages in 6.851s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>
```

package.json — Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON — Visual Studio Code

Fichier Modifier Sélection Afficher Accéder Déboguer Tâches Aide

EXPLORATEUR

ÉDITEURS OUVERTS
- Bienvenue
- package.json

EXEMPLE_NODEJS_CRUD_MYSQL...
- .vscode
- BD
- css
- images
- js
- node_modules
- pochettes
- requetes
- ~$le_get_contents_VS_PHP....
- films.html
- package-lock.json
- package.json
- server.js

Bienvenue | package.json

```
1  {
2    "name": "Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON",
3    "version": "0.0.0",
4    "private": true,
5    "dependencies": {
6      "body-parser": "^1.17.2",
7      "express": "^4.15.3",
8      "express-session": "^1.15.4",
9      "fs": "0.0.1-security",
10     "http": "0.0.0",
11     "multer": "^1.3.0",
12     "path": "^0.12.7"
13   }
14 }
15
```

Tester serveur pour avoir la page films.html

https://expressjs.com/

http://expressjs.com/fr/guide/routing.html

Express en production : https://expressjs.com/en/advanced/best-practice-performance.html#use-try-cat

server.js — Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON — Visual Studio Code

Fichier Modifier Sélection Afficher Accéder Déboguer Tâches Aide

EXPLORATEUR

ÉDITEURS OUVERTS
- Bienvenue
- server.js

EXEMPLE_NODEJS_CRUD_MYSQL...
- .vscode
- BD
- css
- images
- js
- node_modules
- pochettes
- requetes
- ~$le_get_contents_VS_PHP....
- films.html
- package-lock.json
- package.json
- server.js

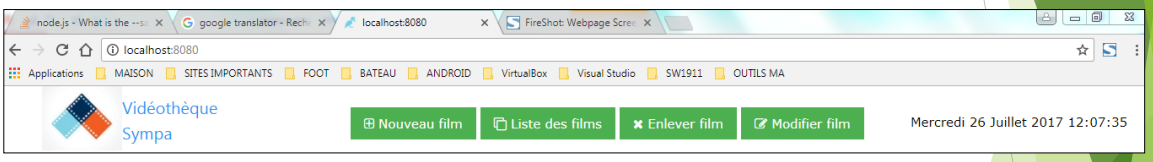
Bienvenue | server.js

```
1  var express = require('express');
2  var http = require('http');
3  var session = require("express-session");
4  var bodyParser = require('body-parser');
5  var path    = require("path");
6  var fs   = require("fs");
7  var multer  = require('multer'); //mutipart form data
8  var upload = multer({ dest: 'pochettes/' })
9  //var gconf = require('./config/gconf');
10 //var uc = require('./user/users_controler');
11 //var dbconcg=require('./groupschat/groupschat_model');
12
13 var app = express();
14 var server = http.createServer(app);
15 server.listen(8080);
16
17
18 app.use(express.static(__dirname));//to get also css, js,...
19 app.use(bodyParser.json()); // support json encoded bodies
20 app.use(bodyParser.text()); // support json encoded bodies
21 app.use(bodyParser.urlencoded({ extended: true })); // support encoded bodies
22
23
24     app.get('/', function(req, res) {
25         res.sendFile(path.join(__dirname + '/films.html'));
26     });
```

Soumettre à node.js notre «serveur» server.js

```
D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\node ser
ver.js
```

Notre serveur est maintenant à l'écoute du port 8080



Pour utiliser mysql

Installer le package

```
d:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>D:\NodeJS_V8.2.0\npm inst
all --save mysql2
+ mysql2@1.3.6
added 20 packages in 10.425s

d:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON>
```

Reste sera analysé via le code de l'application :

Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_CALLBACK	2017-07-27 11:11	Dossier de fichiers

Le même exemple mais avec des Promises en Typescript

Préalables pour lecture

https://basarat.gitbooks.io/typescript/content/docs/promise.html
https://basarat.gitbooks.io/typescript/content/docs/quick/nodejs.html
https://code.visualstudio.com/docs/languages/typescript#_using-newer-typescript-versions

Étude du projet

Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES	2017-08-02 17:32	Dossier de fichiers

ts-node (compiler à la volée du Typescript)

TypeScript Node

NPM version | downloads 1M/month | build failing | coverage 86% | Greenkeeper enabled

> TypeScript execution environment and REPL for node. **Works with** `typescript@>=1.5`.

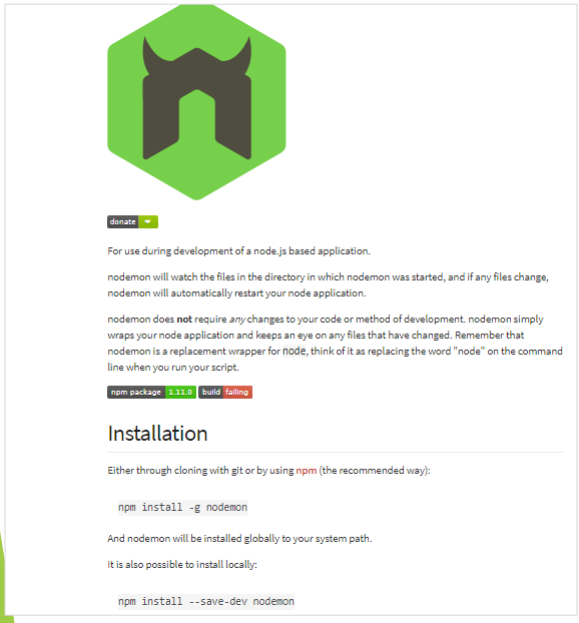
Installation

```
npm install -g ts-node

# Install a TypeScript compiler (requires `typescript` by default).
npm install -g typescript
```

Étude du projet

Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE	2017-08-03 14:18	Dossier de fichiers

Aussi en mode développement



donate

For use during development of a node.js based application.

nodemon will watch the files in the directory in which nodemon was started, and if any files change, nodemon will automatically restart your node application.

nodemon does **not** require *any* changes to your code or method of development. nodemon simply wraps your node application and keeps an eye on any files that have changed. Remember that nodemon is a replacement wrapper for node, think of it as replacing the word "node" on the command line when you run your script.

npm package 1.11.0 | build failing

Installation

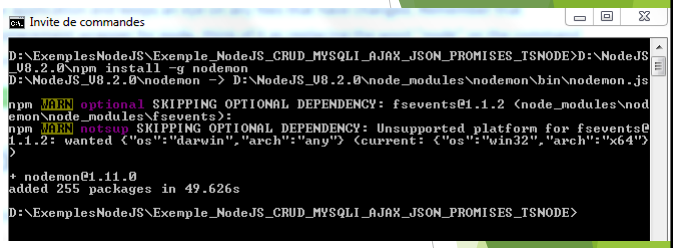
Either through cloning with git or by using npm (the recommended way):

```
npm install -g nodemon
```

And nodemon will be installed globally to your system path.

It is also possible to install locally:

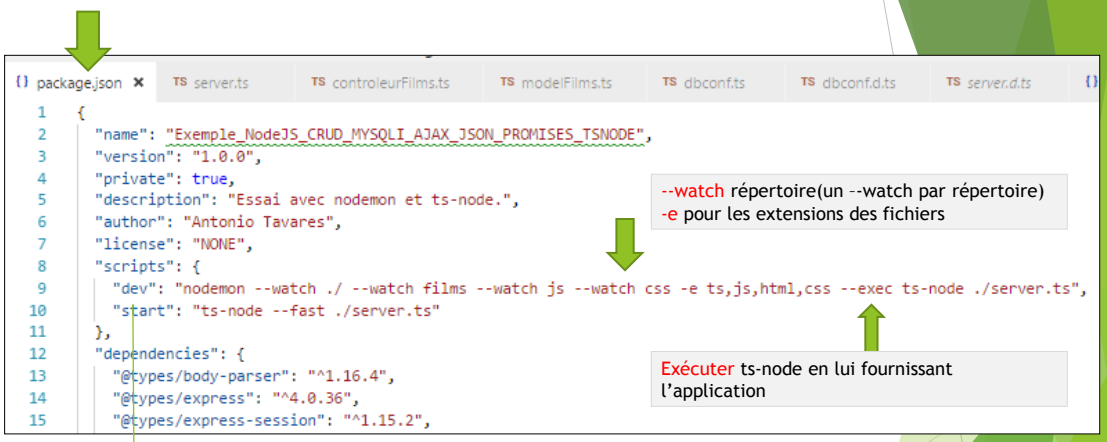
```
npm install --save-dev nodemon
```



Invite de commandes

```
D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE>D:\NodeJS
_V8.2.0\npm install -g nodemon
D:\NodeJS_V8.2.0\nodemon -> D:\NodeJS_V8.2.0\node_modules\nodemon\bin\nodemon.js

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.2 (node_modules\nod
emon\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@
1.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"}
)

+ nodemon@1.11.0
added 255 packages in 49.626s

D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE>
```

Remplacer le module mysql2 par mysql puisque @types/mysql2 n'existe pas. Il y a maintenant en mysql des requêtes paramétrées.
https://github.com/mysqljs/mysql



{} package.json × | TS server.ts | TS controleurFilms.ts | TS modelFilms.ts | TS dbconf.ts | TS dbconf.d.ts | TS server.d.ts

```
1  {
2    "name": "Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE",
3    "version": "1.0.0",
4    "private": true,
5    "description": "Essai avec nodemon et ts-node.",
6    "author": "Antonio Tavares",
7    "license": "NONE",
8    "scripts": {
9      "dev": "nodemon --watch ./ --watch films --watch js --watch css -e ts,js,html,css --exec ts-node ./server.ts",
10     "start": "ts-node --fast ./server.ts"
11   },
12   "dependencies": {
13     "@types/body-parser": "^1.16.4",
14     "@types/express": "^4.0.36",
15     "@types/express-session": "^1.15.2",
```

--watch répertoire(un --watch par répertoire)
-e pour les extensions des fichiers

Exécuter ts-node en lui fournissant l'application

« dev» nom donné pour exécuter en mode développement
« start» en mode production

Démarrage : <ctrl+c> arrêter

npm

```
D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE>npm run d
ev

> Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE@1.0.0 dev D:\ExemplesNode
JS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_TSNODE
> nodemon --watch ./ --watch films --watch js --watch css -e ts,js,html,css --ex
ec ts-node ./server.ts

[nodemon] 1.11.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMI
SES_TSNODE/**/* D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES_
TSNODE\films/**/* D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISE
S_TSNODE\js/**/* D:\ExemplesNodeJS\Exemple_NodeJS_CRUD_MYSQLI_AJAX_JSON_PROMISES
_TSNODE\css/**/*
[nodemon] starting `ts-node ./server.ts`
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `ts-node ./server.ts`
ALLO
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `ts-node ./server.ts`
```

Types

http://www.typescriptlang.org/docs/handbook/declaration-files/do-s-and-don-ts.html

number , string, object, boolean

Promises avec Bluebird

http://bluebirdjs.com/docs/getting-started.html

```
npm install bluebird --save
```

Then:

```
var Promise = require("bluebird");
```

Quelques raisons sans traduction pour utiliser bluebird

So, here are my top 6 reasons to use a more capable Promise library

1. **Non-Promisified async interfaces** - `.promisify()` and `.promisifyAll()` are incredibly useful to handle all those async interfaces that still require plain callbacks and don't yet return promises - one line of code creates a promisified version of an entire interface.
2. **Faster** - Bluebird is significantly faster than native promises in most environments.
3. **Sequencing of async array iteration** - `Promise.mapSeries()` or `Promise.reduce()` allow you to iterate through an array, calling an async operation on each element, but sequencing the async operations so they happen one after another, not all at the same time. You can do this either because the destination server requires it or because you need to pass one result to the next.
4. **Polyfill** - If you want to use promises in older versions of browser clients, you will need a polyfill anyway. May as well get a capable polyfill. Since node.js has ES6 promises, you don't need a polyfill in node.js, but you may in a browser. If you're coding both node.js server and client, it may be very useful to have the same promise library and features in both (easier to share code, context switch between environments, use common coding techniques for async code, etc...).
5. **Other Useful Features** - Bluebird has `Promise.map()`, `Promise.some()`, `Promise.any()`, `Promise.filter()`, `Promise.each()` and `Promise.props()` all of which are occasionally handy. While these operations can be performed with ES6 promises and additional code, Bluebird comes with these operations already pre-built and pre-tested so it's simpler and less code to use them.
6. **Built in Warnings and Full Stack Traces** - Bluebird has a number of built in warnings that alert you to issues that are probably wrong code or a bug. For example, if you call a function that creates a new promise inside a `.then()` handler without returning that promise (to link it into the current promise chain), then in most cases, that is an accidental bug and Bluebird will give you a warning to that effect. Other built-in Bluebird warnings are described here.

Un exemple justificatif par rapport à notre projet précédent

```
const fs = require('fs');

function readFileAsync(file, options) {
    return new Promise(function(resolve, reject) {
        fs.readFile(file, options, function(err, data) {
            if (err) {
                reject(err);
            } else {
                resolve(data);
            }
        });
    });
}

readFileAsync('somefile.text').then(function(data) {
   // do something with data here
});
```

`:Promise<any>{` ⟵ Syntaxe Typescript

Comme dans le projet précédent.

```
const Promise = require('bluebird');
const fs = Promise.promisifyAll(require('fs'));

fs.readFileAsync('somefile.text').then(function(data) {
   // do something with data here
});
```

Ce que nous allons faire.

async/await

https://hackernoon.com/6-reasons-why-javascripts-async-await-blows-promises-away-tutorial-c7ec10518dd9

The following code demonstrates how that works:

```
async function asyncFunc() {
    console.log('asyncFunc()'); // (A)
    return 'abc';
}
asyncFunc().
then(x => console.log(`Resolved: ${x}`)); // (B)
console.log('main'); // (C)

// Output:
// asyncFunc()
// main
// Resolved: abc
```

You can rely on the following order:

1. Line (A): the async function is started synchronously. The async function's Promise is resolved via `return`.
2. Line (C): execution continues.
3. Line (B): Notification of Promise resolution happens asynchronously.

6. Parallelism

The following code make two asynchronous function calls, `asyncFunc1()` and `asyncFunc2()`.

```
async function foo() {
    const result1 = await asyncFunc1();
    const result2 = await asyncFunc2();
}
```

However, these two function calls are executed sequentially. Executing them in parallel tends to speed things up. You can use `Promise.all()` to do so:

```
async function foo() {
    const [result1, result2] = await Promise.all([
        asyncFunc1(),
        asyncFunc2(),
    ]);
}
```

Instead of awaiting two Promises, we are now awaiting a Promise for an Array with two elements.

Using Async Await in Express with Node 8

https://medium.com/@Abazhenov/using-async-await-in-express-with-node-8-b8af872c0016

Angular 4 – node.js

https://scotch.io/tutorials/mean-app-with-angular-2-and-the-angular-cli