

# تقنية NodeJS وإطار العمل Express

اليوم الثاني

# مفاهيم أساسية في تقنية NodeJS

شرح طريقة تركيب الإضافات وإدارتها باستخدام NPM

شرح مفهوم Asynchronous Code

شرح بناء خادم HTTP بسيط

اطار العمل Express

كيفية استخدام pug

كيفية استخدام الإضافة lodash

كيفية معالجة النماذج

كيفية إرسال البريد الإلكتروني

كيفية بناء برنامج متعدد اللغات

# NPM

- في البداية يجب علينا اعداد ملف package.json من خلال الأمر التالي على سطر الأوامر داخل مجلد المشروع الخاص بنا:

```
$ npm init
```

- بعد إدخال الأمر قم بتعبئة البيانات المطلوبة أو قبول الإعدادات المدخلة مسبقاً
- الآن أصبح المشروع جاهز للاستفادة من الإضافات
- يمكنك إضافة إضافة جديدة للمشروع من خلال الأمر:

```
$ npm install -s [packageName]
```

- يجب الحرص دائماً على إضافة الوسم -s لضمان حفظ الإضافة في ملف الإعدادات package.json
- يمكن اختصار كلمة install بالحرف i وسيقوم بنفس العملية

# العمليات غير المتزامنة Asynchronous

- في هذا الجزء سنشرح مفهوم العمليات غير المتزامنة في لغة JavaScript
- يوجد طريقتين لعمل العمليات غير المتزامنة وهي:
  - Promise
  - Callback

# Callback

```
// callback function usually has two arguments the first one is for
// the errors, which you should set to null incase the execution was
// successful or you should fill it with an error in case of errors
var asyncFunction = function(argA, callback){
  setTimeout(function(){
    // "returning" result
    callback(null, {tasty: 'sandwich'})

    // or
    // "throwing" errors
    callback(new Error('Error message'))
  }, 50)
}

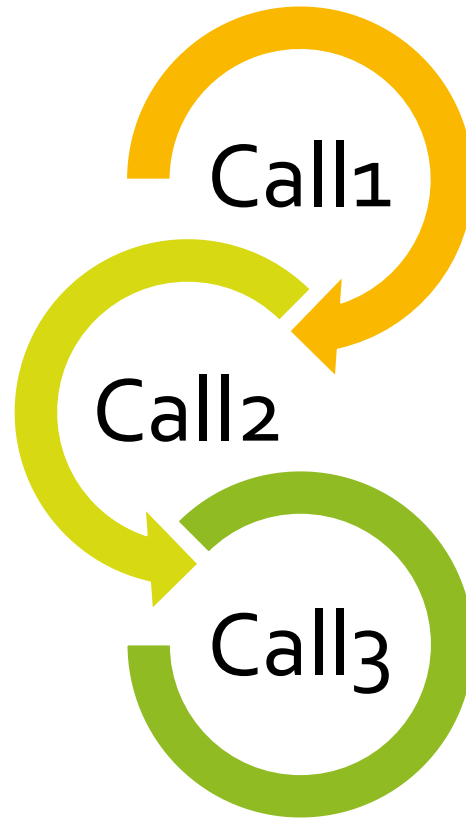
asyncFunction("hello", function(error, result){
  if(error){
    console.error("oops! error happend")
    return; // don't forget about this!
  }
  // handle result
  console.log("nice every thing works as expected")
})
```

# Promise

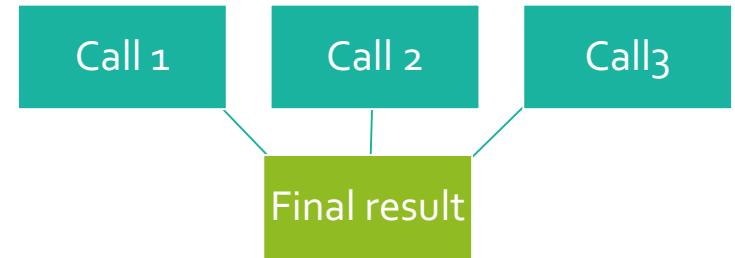
```
function asyncFunction(makeItResolve) {  
  // Promise Object takes two function parameters  
  // the first one we use it when the function is successfully  
  // executed which will trigger the callback of the then function  
  // the second one we may use it when the function has some failures  
  // which will trigger the callback of the catch function  
  return new Promise ((resolve, reject) => {  
    setTimeout (() => {  
      if(makeItResolve)  
        resolve('resolved OK')  
      else  
        reject ('Oops! rejected')  
    }, 2000)  
  })  
}  
  
// if you want the promise to get rejected change the true value to false  
function asyncCall () {  
  console.log ('calling')  
  var promise = resolveAfter2Seconds (true)  
  promise  
    .then(function(result){  
      console.log (result)  
    })  
    .catch(function(error){  
      console.error(error)  
    })  
}  
  
asyncCall ();
```

## أنواع العمليات غير المتزامنة

### Waterfall



### Parallel



## تركيب الإضافة async

- يمكن تركيب الإضافة من خلال الأمر التالي

```
$ npm install -s async
```

- بعد تركيب الإضافة يمكنك استدعاء الإضافة في أي من ملفاتك من خلال الأمر التالي

```
> var async = require("async")
```



# Simple HTTP Server

```
const http = require('http')
const port = 3000

// callback function has two parameters as following:
// request: contains the content of the request
// response: is the utility to be used to send the response
//           once the application is ready to reply
const requestHandler = (request, response) => {
  console.log(request.url)
  response.writeHead(200, {'Content-type': 'text/html'})
  response.write(`
    <html>
      <head><title>Welcome to my website</title></head>
      <body><h1>This is the first page</h1></body>
    </html>
  `)
  response.end()
}

// setting up the callback that are going to be used
// when a request event happens
const server = http.createServer(requestHandler)

// this function uses EventEmitter concept to receive
// requests and execute instructions based on the
// callback that was set early
server.listen(port, (err) => {
  if (err) {
    return console.log('something bad happened', err)
  }

  console.log(`server is listening on ${port}`)
})
```

# Express

- <http://expressjs.com/>
- Fast
- Unopinionated (لا يوجد طريقة واحدة فقط للعمل معه)
- Minimalist

```
$ npm install -s express
```

# Express Cont.

```
const express = require ('express');
const app = express ();
const port = 3000;

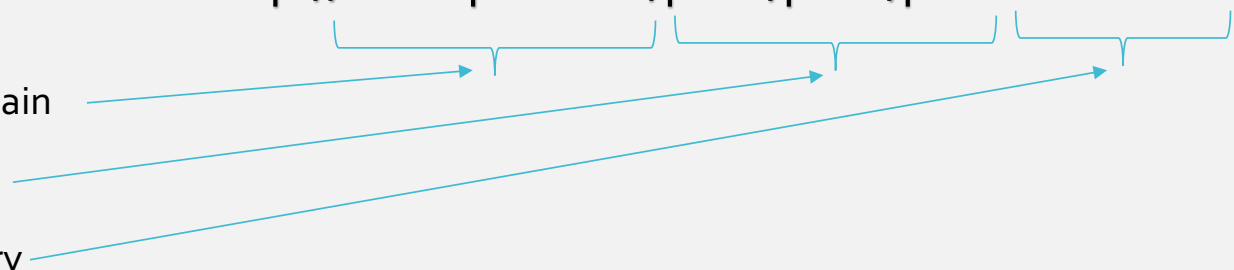
app.get ('/', (request, response) => {
  response.setHeader('Content-type', 'text/html')
  response.statusCode = 200
  response.send (`
    <html>
      <head><title>Welcome to my website</title></head>
      <body><h1>This is the first page</h1></body>
    </html>
  `);
});

app.listen (port, err => {
  if (err) {
    return console.log ('something bad happened', err);
  }

  console.log (`server is listening on ${port}`);
});
```

# أجزاء طلب HTTP

## HTTP Request Parts

- URL
    - Domain
    - Path
    - Query
  - Request Headers
  - Request Content
- 
- The diagram shows the URL `http://example.com/path/path/path?a=1&b=2` with three blue brackets underneath it. The first bracket is under `example.com`, the second is under `/path/path/path`, and the third is under `?a=1&b=2`. Three blue arrows originate from the labels 'Domain', 'Path', and 'Query' in the list to the right and point to the first, second, and third brackets respectively.

# HTTP Request Types

- GET: تستعمل بشكل أساسي لطلب الصفحات ولا يوجد بها Request Content
- POST: تستعمل بشكل أساسي لإرسال طلب إضافة سجلات في الموقع مثل إرسال نموذج التسجيل
- PUT: تستعمل كذلك لإرسال طلبات الإضافة والتعديل على سجلات الموقع
- PATCH: تستعمل لتحديث جزء محدد فقط من السجل المطلوب
- DELETE: تستعمل لإرسال طلب حذف لسجل أو سجلات مسجلة
- جميع هذه الأنواع موجودة ولا يوجد معيار يحكمها لكن ما تم ذكره هو الذي تم التعرف عليه والذي عادةً ما يتم تطبيقه

# إنشاء مشروع Express

```
// this code you should do it once in your machine
```

```
$ npm install -g express-generator
```

```
// create a new empty folder and go to that folder using your Terminal/CMD
```

```
$ express --view=pug
```

# Express Project Structure

- bin
  - www (entry point)
- public
  - Images
  - Javascript
  - stylesheets
- routes
  - index.js
  - users.js
- views
  - error.pug
  - index.pug
  - layout.pug
- package.json
- app.js

## إعداد ملف package.json

```
$ npm init
```

```
$ npm install -s express pug lodash knex
```

- كما تلاحظون قمنا بإضافة أربع إضافات دفعة واحدة وهي كالتالي:
1. Express: وهو إطار العمل الذي نستخدمه لإدارة طلبات HTTP
  2. Pug: إضافة لإنشاء قوالب لصفحات الويب
  3. Lodash: أداة تساعد على التعامل مع المتغيرات وتحريرها
  4. Knex: أداة تستخدم للاتصال بقواعد البيانات وعمل الاستعلامات



# إعداد الملفات

# أسئلة عامة