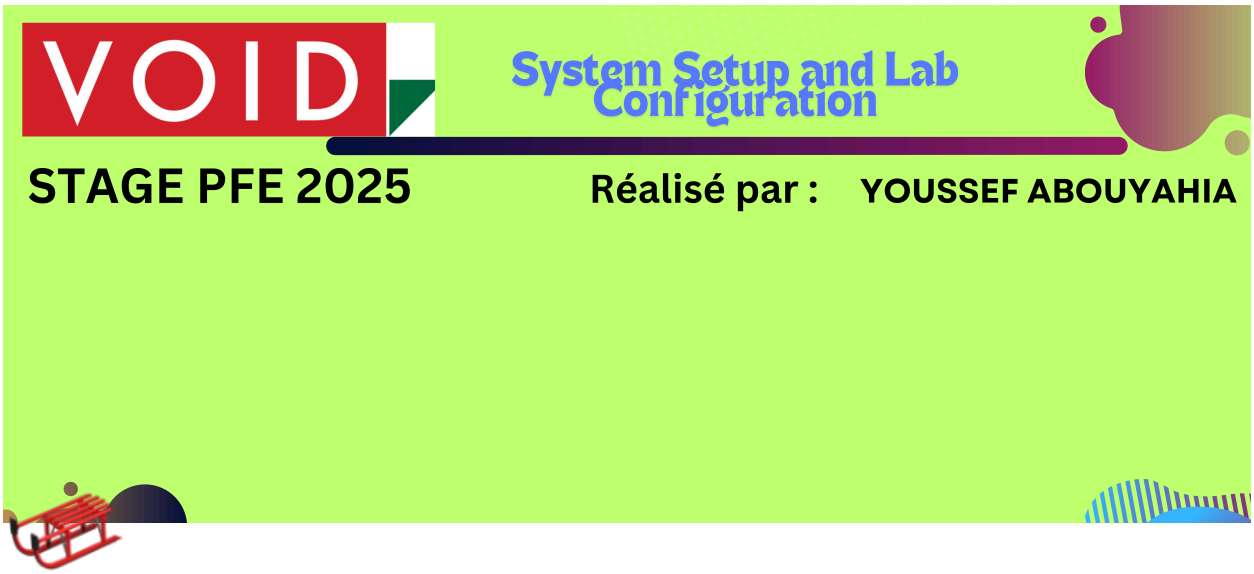


Pop Quiz

I had to install `httpd` , `openssl`

```
~/Doc/V/Sprint 1  brew install openssl                                  1 err | 09:48:37
==> Auto-updating Homebrew...
Adjust how often this is run with HOMEBREW_AUTO_UPDATE_SECS or disable with
HOMEBREW_NO_AUTO_UPDATE. Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man b
rew`).
==> Auto-updated Homebrew!
Updated 4 taps (shivammathur/php, homebrew/services, homebrew/core and homebrew/
cask).
==> New Formulae
algolia         dvr-scan        iguana          kbld            otel-cli
async_simple    evans           jenkins-cli     netfetch        restish
==> New Casks
cloudflare-warp@beta                    font-special-gothic-expanded-one
desktime                                sokim
font-source-han-code-jp                 soundanchor
font-special-gothic-condensed-one

You have 26 outdated formulae installed.

openssl@3 3.4.0 is already installed but outdated (so it will be upgraded).
==> Downloading https://ghcr.io/v2/homebrew/core/openssl/3/manifests/3.4.1
######################################################################## 100.0%
==> Fetching dependencies for openssl@3: ca-certificates
==> Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2025-
######################################################################## 100.0%
```

Then restarting the service `httpd` and it works on localhost

It works!

- After installing `nvm` I'm downloading `node` latest version by default

→ NVM: Node Version Manager (nvm) is a command-line tool that allows developers to manage multiple versions of `Node.js` on a single machine. This is particularly useful because different projects may require different `Node.js` versions, which can lead to compatibility issues if only one version is used.

```
~/Doc/V/S/.nvm | #v0.40.1 . ./nvm.sh                                    ok | 10:03:33
~/Doc/V/S/.nvm | #v0.40.1 nvm install node # "node" is an alias for the latest
 versio
Downloading and installing node v23.9.0...
Downloading https://nodejs.org/dist/v23.9.0/node-v23.9.0-darwin-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v23.9.0 (npm v10.9.2)
Creating default alias: default -> node (-> v23.9.0)
~/Doc/V/S/.nvm | #v0.40.1                                  ok | 12s | 10:06:59
```

But for the project we were asked to use version 22.0 let's do that:

```
~/Doc/V/S/.nvm | #v0.40.1 nvm install 22.0                  ok | 12s | 10:06:59
Downloading and installing node v22.0.0...
Downloading https://nodejs.org/dist/v22.0.0/node-v22.0.0-darwin-x64.tar.xz...
######################################################################## 100.0%
Computing checksum with sha256sum
Checksums matched!
Now using node v22.0.0 (npm v10.5.1)
~/Doc/V/S/.nvm | #v0.40.1                        ok | 10s | 22.0.0 node | 10:12:30
```

→ You can even use aliases to make it easier to switch between versions. For example: `nvm alias sprint1 v22.0`

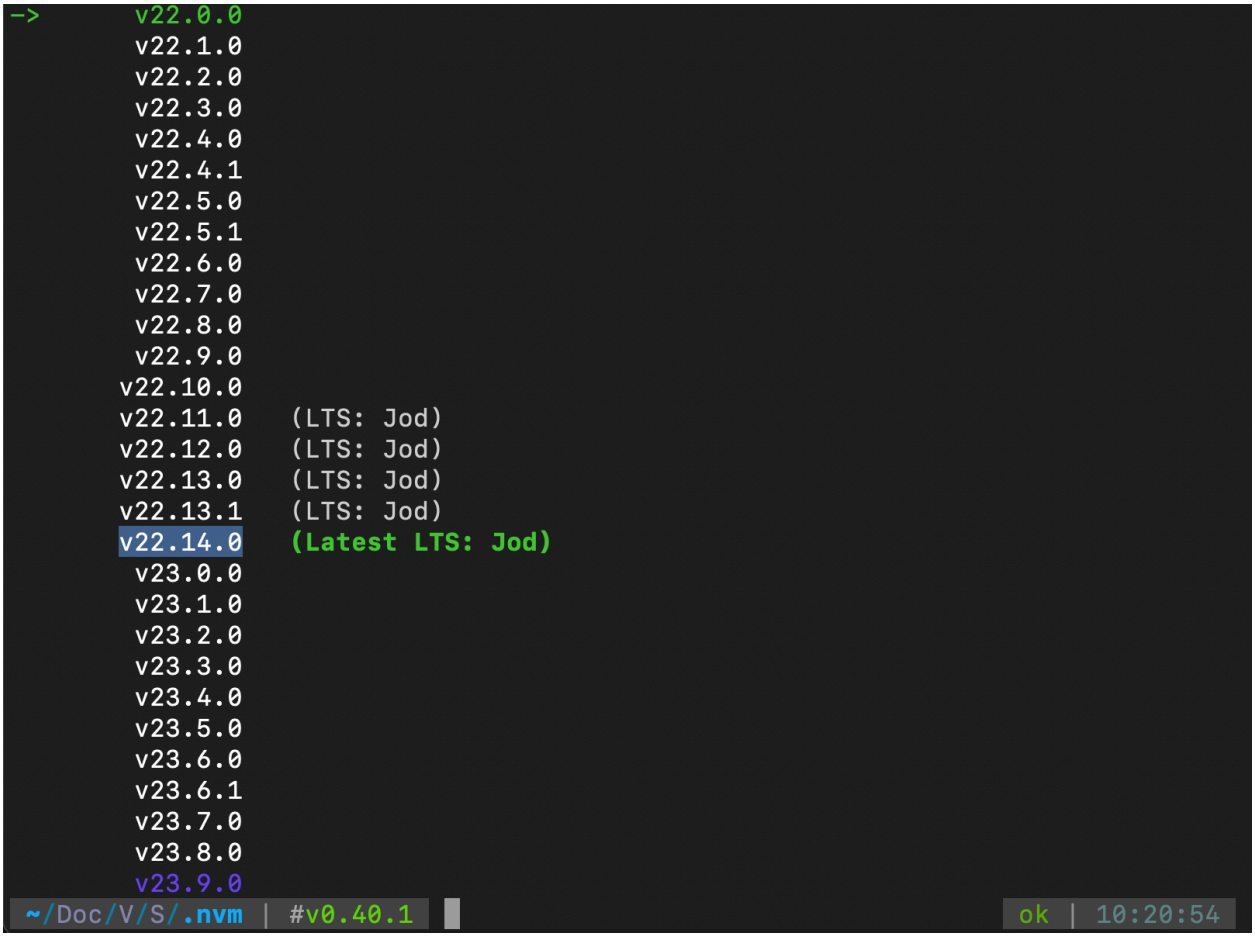
```
 ~/Doc/V/S/.nvm | #v0.40.1  nvm alias sprint1 v22.0                ok | 10:17:01 ]
sprint1 -> v22.0 (-> v22.0.0)
 ~/Doc/V/S/.nvm | #v0.40.1  nvm use sprint1                        ok | 10:17:55 ]
Now using node v22.0.0 (npm v10.5.1)
 ~/Doc/V/S/.nvm | #v0.40.1                                         ok | 10:18:04
```

You can eve set up a version to be used by default like this: `nvm alias default v22.0`

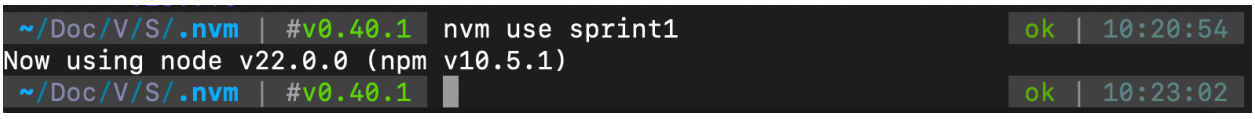
In this command we specify the version 22.0 to be by default for example if we open a new terminal we'll be using `v22`

```
 ~/Doc/V/S/.nvm | #v0.40.1  nvm alias default v22.0                ok | 10:18:04
default -> v22.0 (-> v22.0.0)
 ~/Doc/V/S/.nvm | #v0.40.1  nvm install v22.0                      ok | 10:19:26
v22.0.0 is already installed.
Now using node v22.0.0 (npm v10.5.1)
```

You can also list the remote version with `nvm ls-remote`



Then switch back easily with the alias we just created and this is how we can use nvm in a nutshell:



Set Up the Project:

▼ BACKEND

Let's start by creating a directory name it: `Pop-Quiz` then `frontend` and `backend` as usually in production software we need to respect modularity of the code, hence it's good practice to decouple the code which has a lot of rules but let's leave for another day! Now let's just structure our code!

```
[ ~/Doc/V/Sprint 1  mkdir pop-quiz                                  ok | 10:26:39
[ ~/Doc/V/Sprint 1  d pop-quiz                                      ok | 10:26:46
[ ~/Doc/V/Sprint 1  cd pop-quiz                                     ok | 10:26:48
[ ~/Doc/V/S/pop-quiz  mkdir frontend backend                        ok | 10:26:51
[ ~/Doc/V/S/pop-quiz  ls                                            ok | 10:27:00
backend  frontend
 ~/Doc/V/S/pop-quiz                                                 ok | 10:27:01
```

→ Now let's set up the backend part of our project:

```
[ ~/Doc/V/S/p/backend  composer init                                ok | 10:31:50 ]

  Welcome to the Composer config generator


This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [void/backend]: void/pop-quiz
Description []: This is my first package for a project entitled: Pop-Quiz for th
e first sprint.
Author [welmoubarik <w.elmoubarik@void.fr>, n to skip]: Youssef Abouyahia
Minimum Stability []:
Package Type (e.g. library, project, metapackage, composer-plugin) []:
License []:

Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]?
Search for a package:
Would you like to define your dev dependencies (require-dev) interactively [yes]
? no
Add PSR-4 autoload mapping? Maps namespace "Void\PopQuiz" to the entered relativ
e path. [src/, n to skip]: skip
The src folder name "skip" is invalid. Please add a relative path with tailing f
orward slash. [A-Za-z0-9_-/]+/
Add PSR-4 autoload mapping? Maps namespace "Void\PopQuiz" to the entered relativ
```

composer to generate `composer.json` file

```
Add PSR-4 autoload mapping? Maps namespace "Void\PopQuiz" to the entered relativ
e path. [src/, n to skip]: no

{
    "name": "void/pop-quiz",
    "description": "This is my first package for a project entitled: Pop-Quiz fo
r the first sprint.",
    "authors": [
        {
            "name": "Youssef Abouyahia"
        }
    ],
    "require": {}
}

Do you confirm generation [yes]? yes
 ~/Doc/V/S/p/backend                                    ok | 4m 26s | 10:36:18
```

this is the structure of my package

→ Now time to install using `composer install`

```
 ~/Doc/V/S/p/backend  composer install                  ok | 4m 26s | 10:36:18
Composer could not detect the root package (void/pop-quiz) version, defaulting t
o '1.0.0'. See https://getcomposer.org/root-version
No composer.lock file present. Updating dependencies to latest instead of instal
ling from lock file. See https://getcomposer.org/install for more information.
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
 ~/Doc/V/S/p/backend                                    ok | 10:39:09
```

As you can notice that we get nothing to install as the package we created an an empty package and not dependent on any other package

Then to run the backend we use the command: `php -S localhost:8888 2>/dev/null`

it's basically used to start a simple web server using PHP's built-in web server functionality while redirecting any error messages to `/dev/null`

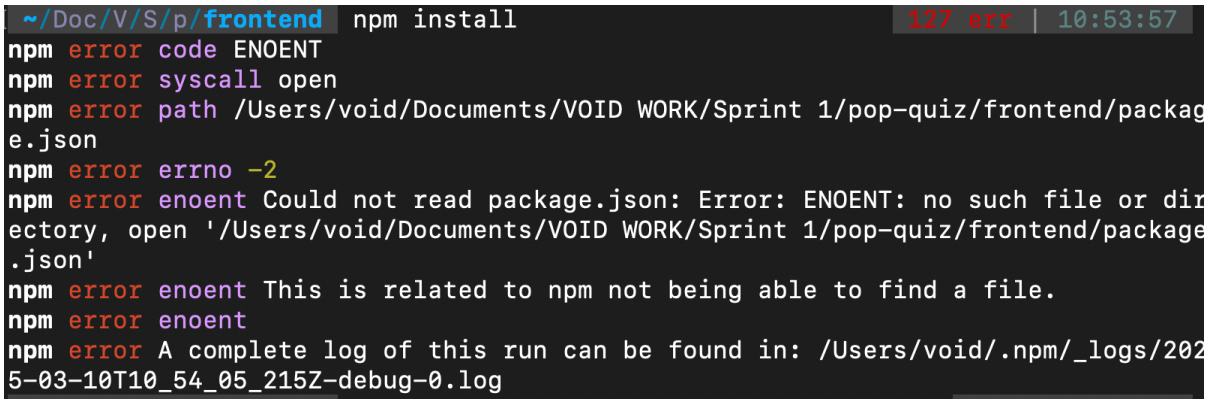
```
 ~/Doc/V/S/p/backend  php -S localhost:8888 2>/dev/null          ok | 10:39:09
```

▼ FRONTEND

`.env.example` file typically contains *dummy values* or placeholders, and it's usually included in the project's repository (e.g., Git).

→ Basically the message here is to use .env file to store any sensitive information such as API keys, database credentials... and do not ever ever commit this file to a public repository the procedure is to usually add it to `.gitignore` file to not be committed !

→ Now we'll need to `npm install` but wait we'll get 2 errors here, first npm not recognized by the shell we only need to source `.zshrc` to apply the changes by nvm, now we'll be here!



- Now we need to initialize the package just like we did with php back with the backend we'll do the same using `npm init`

```
~/Doc/V/S/p/frontend  npm init                                    254 | 10:54:05
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (frontend) frontend
version: (1.0.0)
description: first project for pop-quiz frontend
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/void/Documents/VOID WORK/Sprint 1/pop-quiz/frontend/pa
kage.json:

{
  "name": "frontend",
  "version": "1.0.0",
  "description": "first project for pop-quiz frontend",
  "main": "index.js",
  "scripts": {
```

→ Now we can run npm install successfully

```
~/Doc/V/S/p/frontend  npm install

up to date, audited 1 package in 621ms

found 0 vulnerabilities
~/Doc/V/S/p/frontend                                              ok | 10:58:04
```

→ Next we'll try to run `npm run dev` the command but it'll throw errors

```
~/Doc/V/S/p/frontend  npm run dev                                 ok | 10:58:04
npm error Missing script: "dev"
npm error
npm error To see a list of scripts, run:
npm error   npm run
npm error A complete log of this run can be found in: /Users/void/.npm/_logs/202
5-03-10T10_58_38_145Z-debug-0.log
```

The error is simply because we didn't specify the script dev as it's not recognized by our frontend server, so I went back and modified the file `package.json` adding manually the script to run my application with `npm run dev`

NOTE: It's actually not advised to modify on the package.json file as the simple modification might break the configuration of the project, just for this simple demo and I forgot the `,` ended up with a bigger error than I had before.

→ Again forgot to install `react-scripts` dependency run this before running the project: `npm install react-scripts --save-dev` (or add it in the dependency part of the file)

```
{
  "name": "frontend",
  "version": "1.0.0",
  "description": "first project for pop-quiz frontend",
  "main": "index.js",
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject",
    "dev": "react-scripts start"
  },
  "author": "",
  "license": "ISC"
}
```

→ Yeah, another point if you run the command at this point you'll still get an error as we didn't specify what file to run on the server by default it's `index.html` but in this case we still haven't created it, so let's go and do just that

```
 ~/Doc/V/S/p/frontend  ls                                    ok | 14s | 11:11:25
index.html        node_modules      package-lock.json package.json
 ~/Doc/V/S/p/frontend                                        ok | 11:11:26
```

→ Again you'll have to respect the structure and create a dire `public` and put your `index.html` in it

```
 ~/Doc/V/S/pop-quiz/frontend  mkdir public                                  ok | 11:12:31
 ~/Doc/V/S/p/frontend  ls                                                   ok | 11:13:09
index.html        package-lock.json public
node_modules      package.json
 ~/Doc/V/S/p/frontend  ls -lah                                              ok | 11:13:22
total 1344
drwxr-xr-x    8 void  staff   256B Mar 10 11:13 .
drwxr-xr-x    4 void  staff   128B Mar 10 10:27 ..
-rw-r--r--    1 void  staff     0B Mar 10 10:50 .env
-rw-r--r--    1 void  staff   1.6K Mar 10 11:11 index.html
drwxr-xr-x  865 void  staff    27K Mar 10 11:08 node_modules
-rw-r--r--    1 void  staff   664K Mar 10 11:08 package-lock.json
-rw-r--r--    1 void  staff   414B Mar 10 11:08 package.json
drwxr-xr-x    2 void  staff    64B Mar 10 11:13 public
 ~/Doc/V/S/p/frontend  mv index.html public                                 ok | 11:13:28
 ~/Doc/V/S/p/frontend  ls                                                   ok | 11:13:42
node_modules      package-lock.json package.json      public
 ~/Doc/V/S/p/frontend                                                       ok | 11:13:43
```

- Okey, I promise now we can fire the frontend server and rest assured it's gonna work like a charm !

Methode 2:

→ This is the simplest as we're not gonna install manually but rather I'll create a react project, first let's start by checking the versions of `npm` and `node`

```
 ~/Doc/V/Sprint 1/pop-quiz  node -v                                         ok | 11:29:34
v23.4.0
 ~/Doc/V/Sprint 1/pop-quiz  npm -v                                          ok | 11:30:40
10.9.2
 ~/Doc/V/Sprint 1/pop-quiz                                                  ok | 11:30:43
```

- Then **Install Create React App globally with:**

```
 ~/Doc/V/Sprint 1/pop-quiz  npm install -g create-react-app                 ok | 11:30:43
npm warn deprecated inflight@1.0.6: This module is not supported, and leaks memory. Do no
t use it. Check out lru-cache if you want a good and tested way to coalesce async request
s by a key value, which is much more comprehensive and powerful.
npm warn deprecated rimraf@2.7.1: Rimraf versions prior to v4 are no longer supported
npm warn deprecated glob@7.2.3: Glob versions prior to v9 are no longer supported
npm warn deprecated fstream-ignore@1.0.5: This package is no longer supported.
npm warn deprecated uid-number@0.0.6: This package is no longer supported.
npm warn deprecated fstream@1.0.12: This package is no longer supported.
npm warn deprecated tar@2.2.2: This version of tar is no longer supported, and will not r
eceive security updates. Please upgrade asap.

added 64 packages in 3s

4 packages are looking for funding
  run `npm fund` for details
 ~/Doc/V/S/pop-quiz                                                  ok | 4s | 11:33:38
```

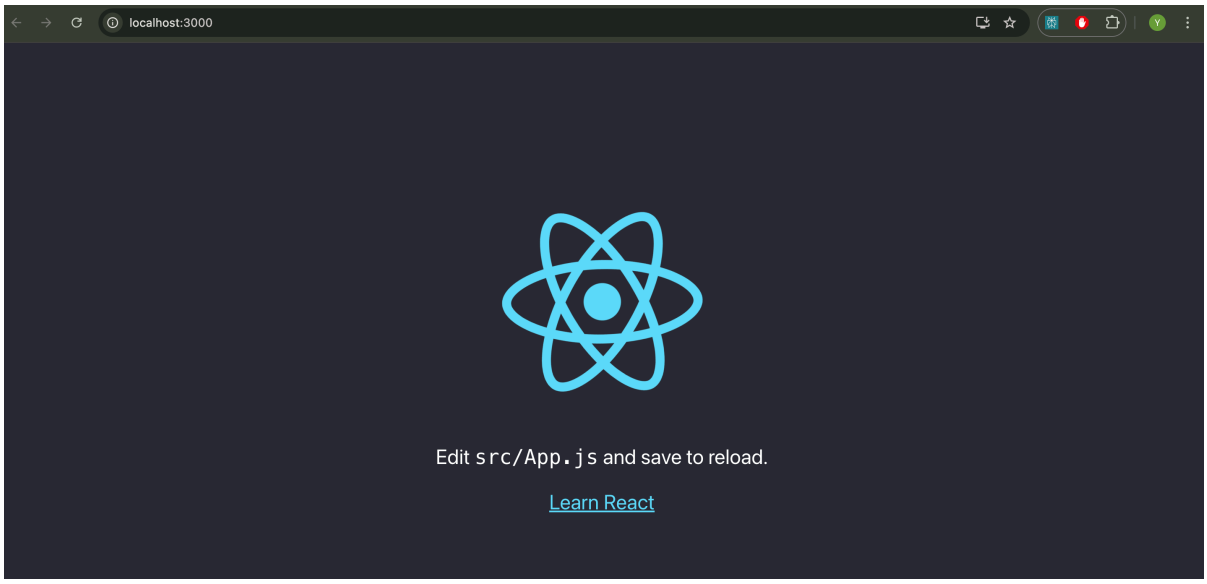
- Next create the react project and specify the name we'll name it `frontend`

```
~/Doc/V/Sprint 1/pop-quiz  npx create-react-app frontend                    ok | 11:34:41
create-react-app is deprecated.

You can find a list of up-to-date React frameworks on react.dev
For more info see:https://react.dev/link/cra

This error message will only be shown once per install.

Creating a new React app in /Users/void/Documents/VOID WORK/Sprint 1/pop-quiz/frontend.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

:
```

- Then cd to the project (frontend) and then run the development server with: `npm start`

```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.100.93:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

- Finally the served webpage by default of react



Note: Now please note when I set up the project this is just a demo for me to install & use my own configuration for the frontend and the backend but don't forget we have preconfigured prjoect on the Pop-Quiz-Repo

→ We only need to run it locally on our machine, and this is what I'm gonna do next, after I kill both the server I just spun

Chapitre 1:

- Let's go to: http://localhost:8888/broken

If you didn't get a result just head back to the logs of the server!

- Let's run a benchmark on the `/crash` using :

```
ab -n 200 -c 10 http://localhost:8888/crash
```

```
~/Doc/V/S/p/frontend  ab -n 200 -c 10 http://localhost:8888/crash     1 err | 11:15:17
This is ApacheBench, Version 2.3 <$Revision: 1923142 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 100 requests
Completed 200 requests
Finished 200 requests


Server Software:
Server Hostname:        localhost
Server Port:            8888

Document Path:          /crash
Document Length:        538 bytes

Concurrency Level:      10
Time taken for tests:   0.015 seconds
Complete requests:      200
Failed requests:        0
Non-2xx responses:      200
Total transferred:      140600 bytes
HTML transferred:       107600 bytes
Requests per second:    12981.96 [#/sec] (mean)
Time per request:       0.770 [ms] (mean)
Time per request:       0.077 [ms] (mean, across all concurrent requests)
Transfer rate:          8912.42 [Kbytes/sec] received
```

→ First let's understand the command:

- **ab -n 200 -c 10 http://localhost:8888/crash:**
  - `n 200:` This tells ab to send a total of 200 requests to the specified URL.
  - `c 10:` This specifies the concurrency level. ab will send 10 requests simultaneously. This simulates 10 users making requests at the same time.
  - `http://localhost:8888/crash:` The URL you are testing.