

Module 2: Hands-On - 2

Using a Promise with setTimeout

Step 1: Create a file named callback.js



Step 2: Open it in any text editor

Step 3: Type the following code:

```
function timer(time) {

function timer(time) {

resolve, reject) => {

setTimeout(() => {resolve(time)}, time)

function timer(time) {

timer(2000).then(x => console.log(`Done after ${x}ms`));
}
```

Step 3.1: Create a function named **timer** that takes time in milliseconds and returns a new **promise** that calls the resolve method after a specified time

```
function timer(time) {

return new Promise((resolve, reject) => {

setTimeout(() => {resolve(time)}, time)

};

timer(2000).then(x => console.log(`Done after ${x}ms`));
```



Step 3.2: Call the timer function with 2000 ms as its argument

```
function timer(time) {

function timer(time) {

return new Promise((resolve, reject) => {

setTimeout(() => {resolve(time)}, time)

};

}

timer(2000).then(x => console.log(`Done after ${x}ms`));
```

Step 3.3: Pass in a function in the .then method of the returned promise which takes time taken is passed as argument and is logged to the console

```
function timer(time) {

function timer(time) {

return new Promise((resolve, reject) => {

setTimeout(() => {resolve(time)}, time)

};

}

timer(2000).then(x => console.log(`Done after ${x}ms`));
```

Step 4: Open the command prompt in the same directory as the file

```
C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.18362.476]

(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Intellipaat-Team\Desktop\Module 2\promise>_
```

Step 5: Run the file using the command 'node promise.js'

```
C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.18362.476]

(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Intellipaat-Team\Desktop\Module 2\promise>node promise.js

Done after 2000ms

C:\Users\Intellipaat-Team\Desktop\Module 2\promise>
```