Lecture 24 | Promises and beyond, Automation using JavaScript

- Reading files serially using Promises

```
const fs = require("fs");
const log = console.log;

log("Before");

let f1p = fs.promises.readFile("f1.txt"); // read file f1.txt (pending)
f1p.then(cb); // callback
function cb(data) {
  log("File data : " + data); // File data : text from f1 (resolved)
  let f2p = fs.promises.readFile("f2.txt"); // read file f2.txt (pending)
  f2p.then(cb2); // callback
}

function cb2(data) {
  log("File data : " + data); // File data : text from f2 (resolved)
  let f3p = fs.promises.readFile("f3.txt"); // read file f3.txt (pending)
  f3p.then(cb3); // callback
}

function cb3(data) {
  log("File data : " + data); // File data : text from f3 (resolved)
}

log("After");
```

> Output :

```
> node serialReadingPromises.js
Before
After
File data : text from f1
File data : text from f2
File data : text from f3
```

- Improvements over the previous example

```
const fs = require("fs");
const log = console.log;

log("Before");
```

```
let f1p = fs.promises.readFile("f1.txt"); // read file f1.txt (pending)

function cb(data) {
  log("File data : " + data); // File data : text from f1 (resolved)
  let f2p = fs.promises.readFile("f2.txt"); // read file f2.txt (pending)
  return f2p; // return the promise
}

function cb2(data) {
  log("File data : " + data); // File data : text from f2 (resolved)
  let f3p = fs.promises.readFile("f3.txt"); // read file f3.txt (pending)
  return f3p; // return the promise
}

function cb3(data) {
  log("File data : " + data); // File data : text from f3 (resolved)
}

f1p
  .then(cb)
  .then(cb2)
  .then(cb3)
  .catch(function (err) {
    log(err);
  });

log("After");
```

> Output :

```
> node serialReadingPromises.js
Before
After
File data : text from f1
File data : text from f2
File data : text from f3
```

Automation using JavaScript | Puppeteer | `npm i puppeteer`

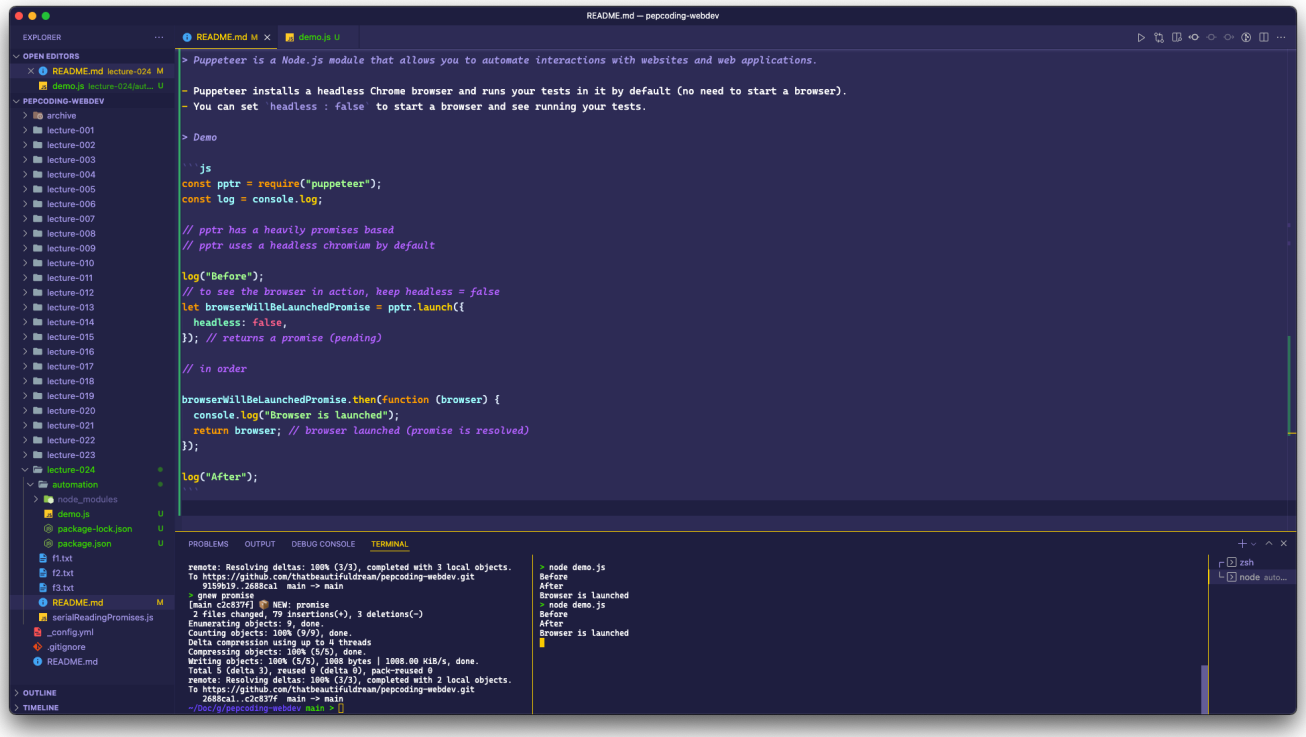
> Puppeteer is a Node.js module that allows you to automate interactions with websites and web applications.

- Puppeteer installs a headless Chrome browser and runs your tests in it by default (no need to start a browser).
- You can set `headless : false` to start a browser and see running your tests.

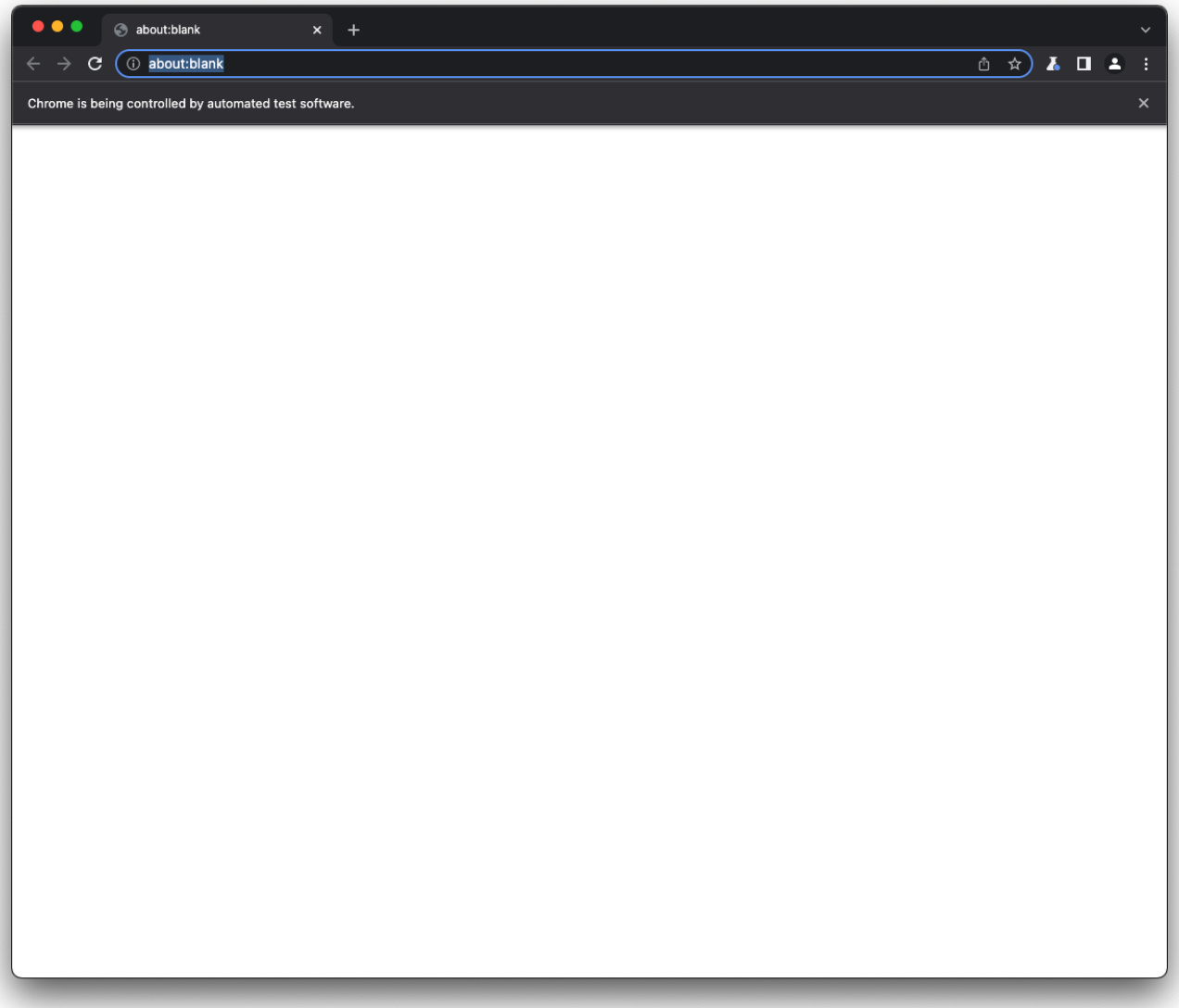
> Demo

```
const pptr = require("puppeteer");
const log = console.log;

// pptr has a heavily promises based
// pptr uses a headless chromium by default

// to see the browser in action, keep headless = false
let browserWillBeLaunchedPromise = pptr.launch({
  headless: false,
  defaultViewport: null,
}); // returns a promise (pending)
// defaultViewport is the default viewport size and setting it to null
helps normal view

// in order

browserWillBeLaunchedPromise.then(function (browser) {
  console.log("Browser is launched");
  return browser; // browser launched (promise is resolved)
});

log("After");
```

Code :



Chromium Instance

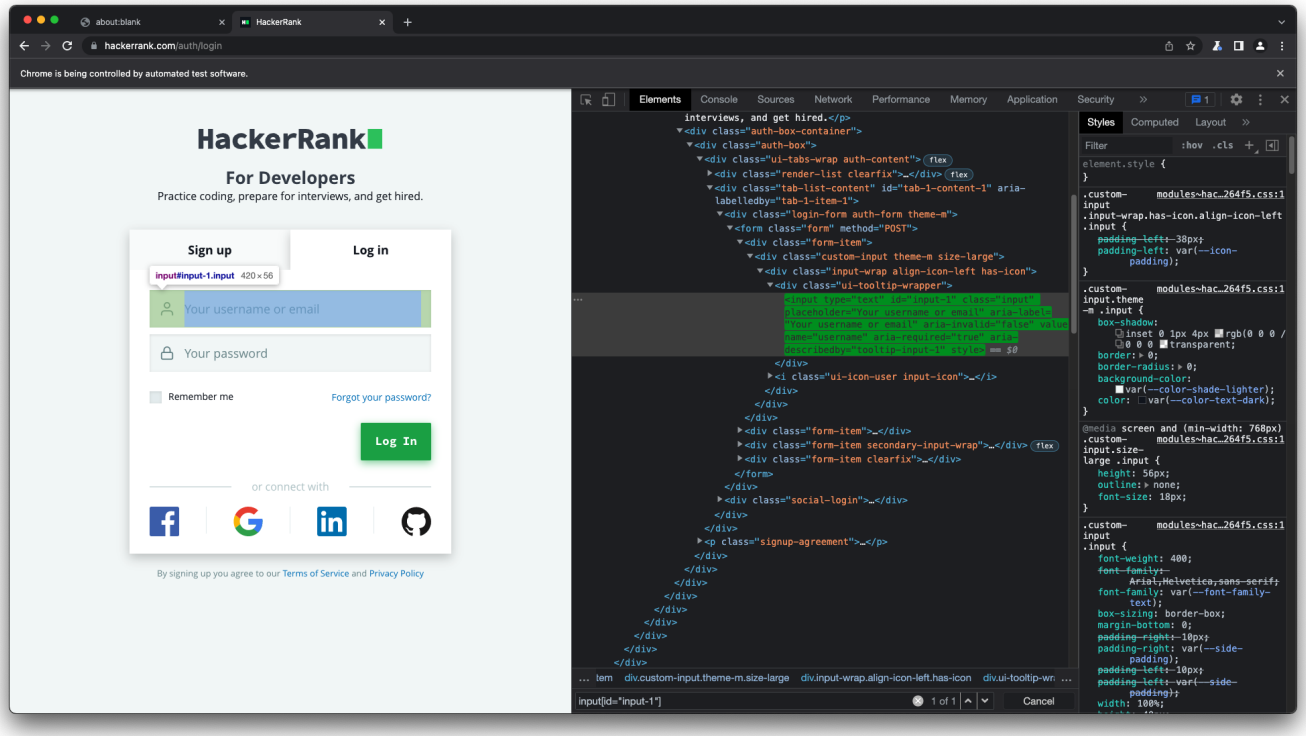


- Opening a new tab and navigating to a page

```
browserWillBeLaunchedPromise
  .then(function (browser) {
    //   console.log("Browser is launched");
    //   return browser; // browser launched (promise is resolved)
    let newTabPromise = browser.newPage(); // returns a promise (pending)
    return newTabPromise; // new tab is created (promise is resolved)
  })
  .then(function (newTab) {
    console.log("New tab is created");
    return newTab.goto("https://milindmishra.me"); // new tab is loaded
(promise is resolved)
  })
  .then(function () {
    console.log("New tab is loaded");
  });
```

HackerRank Web Automation ⚡

Grabbing the relivant id's




```
const loginLink = "https://www.hackerrank.com/auth/login";
const log = console.log;
const email = "xiyale4432@goonby.com";
const password = "xiyale4432";
const name = "Mike Tyson";

let puppeteer = require("puppeteer");

console.log("Before");

let page;

// Puppeteer works on promises

let browserWillbeLauncedPromise = puppeteer.launch({
  headless: false,
  defaultViewport: null,
  rgs: ["--start-maximized"],
});
// we used puppeteer launch method to return an instance of broswer

browserWillbeLauncedPromise
  .then(function (browserInstance) {
    let newTabPromise = browserInstance.newPage();
    return newTabPromise;
  })
  .then(function (newTab) {
    console.log("New Tab opened");
    page = newTab;
```

```
    let pageWillbeOpenedPromise = newTab.goto(loginLink);
    return pageWillbeOpenedPromise;
  })
  .then(function () {
    let typedEmailPromise = page.type("input[id='input-1']", email, {
      delay: 100,
    });
    return typedEmailPromise;
  })
  .then(function () {
    let typePasswordPromise = page.type("input[id='input-2']", password, {
      delay: 100,
    });
    return typePasswordPromise;
  })
  .then(function () {
    let loginPromise = page.click('button[data-
analytics="LoginPassword"]', {
      delay: 100,
    });
    return loginPromise;
  });

console.log("After");
```

> Preview :