

Getting Started with Live Coding!

In this lesson, we will commence our journey of coding live web pages together.
Let's begin!

WE'LL COVER THE FOLLOWING



- Key software components for web development
 - A code editor:
 - A web server
- Installing and using Node.js
- A concise overview of Node Package Manager (NPM)
- Preparing for live coding

Key software components for web development

#

We usually need **three software components** to get started with web development.

These components are:

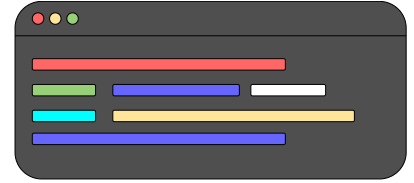
1. a code editor
2. a web server
3. a browser

Luckily, we at Educative provide our learners with embedded coding environments to make learning a breeze, without the hassle of setting up your coding environment.

We will, however, go over the basics of why the above software components are necessary.

A code editor: #

To edit the code, you can use an integrated development environment such as Webstorm, Visual Studio, Eclipse, or even utilize a simple code editor such as Sublime Text, TextMate, Notepad++, or whatever you prefer.



In our case, we provide you two coding editors:

1. **RunJS widget:** this allows for static coding of apps that run on HTML, CSS, and JavaScript.
2. **Live Coding Widget:** this will allow for live coding with browser support. Our app will be hosted on Educative's preassigned host, this should look something like the following under the Live Coding widget:

Your app can be found at: <https://x6jr4kg.educative.run>

A web server

You need a webserver to *host* sample pages. You can choose from a broad range of web servers, including **Apache**, **Node.js**, and **IIS**.

Apache and Node.js are supported on all major platforms, while IIS (and its little brother, IIS Express) runs only on Windows.



In this course, we use Node.js because it is very easy to install, configure, and use. With a very simple preparation step it allows live coding; *as you modify the source files, the changes are immediately reflected in the browser.*

 **Helpful Tip**

In this lesson, you will learn how to get started with Node.js, our live web server.


Installing and using Node.js

As mentioned, we will use Node.js as a web server. Node.js can be obtained via the appropriate setup kit for your platform from <http://nodejs.org/download>.



Installing **Node.js** is very easy and takes only a few minutes. But first, let's see what this is all about.

A concise overview of Node Package Manager (NPM)

 **Note:** If you already have experiences with Node.js and npm, you can skip this short section.

Node.js installs its own **package management tool, npm (Node Package Manager)**.



The main role of this utility is to manage those Node.js packages and components your app or development process depend on.

The npm tool can install packages from a repository to your machine. These packages can be installed *globally*, where all Node apps can access them, or *locally*, to only your project. You can save the references to the installed packages into a **package.json** file that can be used by your npm later to update or install missing components.

Each exercise folder contains two important files.

The **package.json** defines the **start** command that, as its name suggests,

launches the web server and displays your page in the browser. The **index.html** file is the page that is rendered in the browser.

All exercises use a copy of the very same **package.json** file. In the root folder, you can find a **package.json** file to copy and use in your own exercise projects.

Preparing for live coding

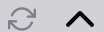
In this course, you will be able to edit the sample code and observe the changes in the browser immediately without restarting the web server or manually refreshing the browser page.

This magic is brought to you by the **live-server** package. In the terminal prompt below, run this command line:

```
npm install live-server -g
```

This instruction starts the Node Package Manager that downloads the live-server package and all its dependencies from the web and installs it globally.

Terminal



A successful install should look as follows:

```
Connecting to Terminal \
root@educative:/#
root@educative:/# npm install live-server
npm WARN deprecated opn@6.0.0: The package has been renamed to `open`
npm WARN saveError ENOENT: no such file or directory, open '/package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open '/package.json'
npm WARN !invalid#1 No description
npm WARN !invalid#1 No repository field.
npm WARN !invalid#1 No README data
npm WARN !invalid#1 No license field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":
"darwin", "arch": "any"} (current: {"os": "linux", "arch": "x64"})

+ live-server@1.2.1
added 191 packages from 149 contributors and audited 2240 packages in 10.09s
found 0 vulnerabilities
```

Great! Now that this is done, we're all set and ready to start learning HTML in the next lesson!

See you there! :)

