

[COMP 426](#)[Home](#)[Assignments](#)[Topics](#)[Calendar](#)[Log out](#)

Assignment 0

Contents

1. [1. Install Required Software](#)
 - [Windows](#)
 - [Mac](#)
2. [2. Set Up Workspace](#)
 - [2.1 Clone the COMP 426 GitHub Repository.](#)
 - [2.2 Download Dependencies](#)
3. [3. Demo](#)
 - [3.1 Create a project directory and initialize with npm](#)
 - [3.2 Install Browsersync](#)
 - [3.3 HTML](#)
 - [3.4 Start the development server](#)
 - [3.5 Edit Some HTML](#)
 - [3.6 Stop the development server](#)
4. [4. Submit](#)
 - [4.1. Create a comp426.com account](#)
 - [4.2. Submit the assignment](#)
 - [4.3. Check your score](#)

Welcome to COMP 426 - *Modern Web Programming*. The purpose of this assignment is to get familiar with the tools that we will be using in COMP 426 for the rest of the semester as well as the assignment submission process.

In total, there will be 11 programming assignments including this one. All assignments will be released on the first day of class, but the due dates for each are spread throughout the semester.

These assignments were specifically designed to augment the material from class. Some assignments will introduce important new material that isn't directly covered in class. To be successful in COMP 426, it is crucial to keep up with submitting assignments on time as well as attending class. Historically, students who fall behind in the assignments tend to struggle in class; similarly, students who miss class tend to struggle when completing assignments.

To ensure that assignments are completed and submitted on time, a significant late penalty will be applied to all late submissions. Furthermore, the COMP 426 LAs have been instructed not to give help on late assignments. The only avenue for getting help on a late assignment is to schedule a special appointment with the graduate TA Aaron Smith or the head LA Chris Burgess. **A 10% late penalty will be applied to assignment submissions for every day late. No credit will be given for submissions more than 10 days late.**

This semester, most of the assignments will be autograded, meaning you will get near-instant feedback about your grade as you are working. You may resubmit autograded assignments as many times as you wish, so feel free to keep trying until you get it right. A few assignments will be manually graded. Manually graded assignments will only be graded one time, after their respective due dates.

Working with other students on assignments is strictly prohibited. In addition to the assignments, COMP 426 also requires a term project which will offer ample opportunities for collaborating with other students to design and build a web app. The flip side of this is that you must work individually on assignments. Additionally, be careful that you do not submit any code that you did not personally write. **Submitting code copied from others or from the internet may result in your submission being flagged by the system for plagiarism.**

Note that this first assignment, a00, is graded only for completion. Future assignments will be much deeper in scope, and you should expect to spend significantly more time on them.

1. Install Required Software

Below are two links that guide you through the process of installing the required software for this course. This software must be installed on your computer not only so you can complete the assignments, but also so you can receive help in office hours. Visual Studio Code is the recommended code editor for COMP 426, but if you are willing to sign up for a (free) student license, then [WebStorm](#) is also a good choice.

Windows

If you have a PC running Windows, follow the [Windows setup guide](#).

Mac

If you have a Mac running OSX, follow the [Mac setup guide](#).

2. Set Up Workspace

Once you have installed the required software, the next step is to download and install a special code package designed specifically for COMP 426. This package will set up the following files on your computer:

- An organized directory structure with subdirectories for every assignment you will complete throughout the semester
- Starter code to give you a head start on some of the assignments
- A command-line tool for automatically submitting assignments to the autograder straight from Visual Studio Code

This section explains how to download and set up the COMP 426 code package onto your computer.

2.1 Clone the COMP 426 GitHub Repository

The course-specific code package is hosted online in a [GitHub repository](#). The first step is to "clone" (aka download) the code to your computer. There are many ways to clone a GitHub repository to your computer---and you may be familiar with this process from other courses. For instance, you may have prior experience cloning repositories using [GitHub Desktop Client](#). If you have one, feel free to use your preferred method to clone the repository. Otherwise, this guide assumes this is your first time cloning a GitHub repository and walks you through the process using Visual Studio Code. Complete the following steps to download the COMP 426 code package using Visual Studio Code.

1. Open Visual Studio Code
2. Select "View" from the top bar and click on "Command Palette..." in the box that pops up.
3. Type "Git: Clone" (which should begin to autofill) and press enter.
4. Copy and paste the following URL into the bar that pops up at the top middle of the window:
<https://github.com/onsmith/comp426-cli>
5. Press enter twice. Select "Open Repository" when the option arises.

2.2 Download Dependencies

The COMP 426 code that you downloaded uses a few "dependencies." Dependencies are another word for additional libraries and packages written by third-party developers that are required and used internally by the COMP 426 code. In the early days of programming, managing dependencies was time consuming and difficult. Luckily, nowadays Node.js comes with a fantastic built-in tool, called Node Package Manager (npm), which is designed to automatically manage JavaScript dependencies for you. Npm is a command line

tool, which means it only works through the terminal (aka command line). We'll learn a lot more about how to use npm later in the semester, but for now we'll use its most basic fundamental feature: automatically selecting and downloading all relevant libraries to your computer.

This course requires basic familiarity with the terminal, especially for using tools like npm. Don't worry if you don't feel comfortable with the terminal just yet---the assignment guides will walk you through the most important commands. In the long run, however, having a working knowledge of the command line is a very valuable skill, especially for aspiring software engineers. With this in mind, it may benefit you to make sure you fully understand every command we use. If you're looking to learn more about using the terminal, the [linux commands cheat sheet](#) is a great resource.

The following steps guide you through opening a terminal window from Visual Studio Code and using npm to automatically install dependencies.

1. To open a terminal from Visual Studio Code, open the "View" menu and select "Integrated Terminal" (or just "Terminal" on Mac).
2. A new terminal window should have opened in the bottom panel of Visual Studio Code. Type the following command into the terminal and then press enter:

```
$ npm install
```

3. Wait for this process to complete. If warnings appear in the terminal, it's safe to ignore them.

The command `npm install` tells npm to look in the file named `package.json`, which should contain a list of dependencies to install, and install these dependencies. By default, the downloaded dependencies are placed into a special directory named `node_modules`. After the command completes, you should be able to find both `package.json` and `node_modules` in the left side panel of Visual Studio Code. Try clicking on them to see what's inside.

Again, we'll learn a lot more about how to use npm in a later assignment (a02, to be exact).

3. Demo

After completing the last section, your computer should be set up for developing websites in COMP 426! Let's run through a quick demo to make sure everything works.

For this demo, we're going to pretend we're starting a new web dev project. We'll set up a new project directory using npm and Visual Studio Code. **Pay close attention to this process, because you'll want to repeat it every time you start working on a new COMP 426 assignment.** In fact, this process can even be used outside of class in the real world to get set up working on new web-based projects.

3.1 Create a project directory and initialize with npm

The first step for starting a new project is usually to create a new folder that will store your project code. However, for COMP 426, folders have already been created for you for all assignments (a00 - a10). You can see these folders in the left side panel of Visual Studio Code, which shows the file explorer.

The file explorer can be toggled open or closed by clicking on the pictured icon.



The explorer icon in Visual Studio Code

If you were going to be creating a new project folder, you could do it by right clicking in the empty space at the bottom of the explorer and clicking "New Folder." However, for this demo we will just use the existing folder "a00".

Next, right click on the folder "a00" and click "Open in Terminal." This will open a new terminal window *pointing at the a00 folder*. **This raises an important fact about terminals---they always point to a specific folder location in your computer.** Usually, the terminal prompt will give some indication of which folder it is currently pointing at. For instance, you might see something like this:

```
your-computer-name:a00 yourname$
```

The important thing here is a00, which indicates that you are currently in the a00 folder.

There is a special terminal command, `pwd`, which tells you exactly which folder the terminal is currently in. `pwd` stands for "print working directory." Try running `pwd` in your terminal window now, and verify that it is pointing to the a00 folder.

```
$ pwd
/c/apps/comp426-cli/a00
```

Whenever you are working on a particular assignment, it is absolutely crucial that you **run any related terminal commands from inside the assignment's folder**.

Once you have confirmed that your terminal is pointing to the a00 folder, run the command `npm init`. This command tells npm to create a brand new web project in the current folder. As part of the initialization process, it'll ask you a long list of questions about your new web project. It's okay to just press enter through all the prompts.

```
$ npm init
```

What does running `npm init` actually do? It's actually pretty simple: it just creates a new `package.json` file in the current directory. The `package.json` file is npm's version of a configuration file for your entire web project. It includes lots of valuable information about your project including the project's name, version, description, and author. It also includes a list of the dependencies that your project relies on, and a list of custom scripts, or actions, that may be a part of your web application. The answers you type to the questions asked during the `npm init` process are used to add more information to the `package.json` file that is created for your project.

At this point, the a00 folder should now have a brand new `package.json` file with the following contents:

```
{
  "name": "a00",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

3.2 Install Browsersync

Next, we're going to use npm to add our first dependency to the new a00 web project. The package we'll add is called [Browsersync](#), and it will help make web development much easier! Browsersync is a "dev server," and its job is to show you what your website looks like in real time as you actively make changes to your

code. Browsersync works by pretending to be a real web server, running in the background on your computer and monitoring changes to the files in your project directory. Although it's optional, you may wish to install and use Browsersync when working on all assignments for COMP 426.

To install Browsersync as a dependency for a00, run the following command inside the a00 terminal:

```
$ npm install browser-sync
```

Remember back in Section 2.2 when we ran `npm install` to install dependencies for the COMP 426 code? This time we're adding the extra word (called a command line argument) `browser-sync` to the command. This tells npm to search online for the package named `browser-sync`, add it as a dependency to our new project a00, and download it. One simple command automatically added a powerful third-party program to your new app---without even writing a single line of code! We'll be using `npm install` a lot this semester to add other third-party packages to our code.

Note: To get credit for a00, you must have Browsersync added as a dependency to your project.

3.3 HTML

Next, let's add some html. Right click on the a00 folder, select "New File," and name the new file `index.html`.

Copy and paste the following code into the new file and save:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Assignment 0 test!</title>
  </head>
  <body>
    <div>
      <h1>Looking good!</h1>
      <p>Go ahead and edit me.</p>
    </div>
  </body>
</html>
```

3.4 Start the development server

With the simple `index.html` file in place, its time to see if things actually work. Using the terminal inside the a00 folder, run the following command to start the development server:

```
$ npx browser-sync start -sw
```

A browser window should have been automatically opened, showing the HTML page we created in the prior step. If not, open <http://localhost:3000/> in Google Chrome. You should see the text inside the `<h1>` `</h1>` and `<p>` `</p>` now being rendered on the website.

Note: The command `npx browser-sync start -sw` is worth memorizing, especially if you plan to use Browsersync for assignments in COMP 426 (which is highly recommended). Browsersync is actually quite a powerful tool, although we won't get into the more advanced features. If you're interested in learning more, [here's a list of the supported command line options](#). The `-sw` part of the command tells Browsersync to serve your files locally and actively watch for changes that

you may be making to the code. Because we didn't install the browser-sync globally we need to use the **npm** command to run it locally.

3.5 Edit Some HTML

The beauty of using a local web server as you develop is that every time you save your changes, the updated version of the website will be immediately shown in the browser window.

To demonstrate this, we're about to make a change to the `index.html` file created in Section 3.3. The instant you save the changes you make, you should see the webpage automatically reload to reflect the change. This makes development much easier and faster. Get used to this workflow because it is an industry standard and it should make A1 much easier!

Change line 14 from:

```
<p>Go ahead and edit me.</p>
```

to:


```
<strong>Almost done I promise.</strong>
```

Save the changes you made, and watch the webpage instantly reload in the browser!

- If you're on Windows, save by pressing `Ctrl + S` on your keyboard.
- If you're on a Mac, save by pressing `Cmd + S`.

3.6 Stop the development server

You may have noticed that back in Visual Studio Code, the terminal panel still looks something like this:

 browser-sync is running in Visual Studio Code"

There's no dollar-sign prompt (\$), and the terminal won't let you type text into it. This is because *Browsersync is still running* and is using up that whole terminal. If you ever needed access to a second terminal while Browsersync was running, you could always start up another one with `View -> Terminal`.

Visual Studio Code understands that you may need to use multiple terminals at once, and it provides an easy way to switch between them: the small drop-down to the right of the word "TERMINAL" in the screenshot above. Clicking that drop-down displays a list of all the active terminals running in Visual Studio Code. You can kill, or delete, a terminal with the trash can icon. This immediately terminates whatever process is running in the terminal and shuts that terminal down. Other terminals in Visual Studio Code will be unaffected by this action.

Whenever you finish working, you may wish to stop the Browsersync development server. One way to do this is by clicking the trash can icon, thereby killing the whole terminal window. Another way is to click back on the terminal that has Browsersync running (see screenshot above) and press the `Ctrl + C` keys on your keyboard. This should exit Browsersync and return the terminal back to a dollar-sign prompt (\$).

Pressing the `Ctrl + C` keys while working in a terminal sends a special interrupt (SIGTSTP or "terminal stop") to the process currently running on that terminal. That interrupt forces the process to close.

4. Submit

If you've made it to this point, you've finished a00. The final step is to submit the code you created.

All assignments in COMP 426 will be submitted for grading using the same process. First, you'll run a special command in the terminal which will upload your code for (auto) grading. Once your assignment is graded, you can see your score by logging in to your comp426.com account.

4.1. Create a comp426.com account

Before you can submit assignments in COMP 426, you'll first need to create an account on our course website. If you have not done so already, visit comp426.com/signup to create a COMP 426 account now. You'll have to confirm your UNC email address before using your account, so double-check that you typed your email correctly!

Make sure you have created your account and confirmed your email address before continuing. At this point, you should be able to successfully log in to comp426.com using your UNC email address and password.

4.2. Submit the assignment

Now we're going to upload your work for a00.

1. Open a new terminal in Visual Studio Code (view -> Terminal).
2. Make sure the terminal is pointing to the root folder for COMP 426 (Hint: Check what folder you're in with `pwd`).
3. Run the following command in the terminal:

```
$ npm run submit a00
```

4. The grading script will ask you for your onyen. Type it and press enter.
5. The grading script will ask you for your COMP 426 password. Type it and press enter.
6. If you typed the correct onyen and password and are connected to the internet, the grading script should automatically upload your assignment to the server. You will see a green success message if it worked correctly.
7. If everything was successful, the system will ask you if you want to save your onyen and password for future submissions. Select yes or no.

That's it! **Again, you will use this same process to submit future assignments as well.** Simply replace "a00" with "a01" or whatever assignment you're currently working on when you're ready to submit.

If you choose to save your onyen and password, the system will store them in a special file named ".env", stored in the root folder for COMP 426. If you're interested, you can open this file and check it out.

4.3. Check your score

Finally, to check your score you must log on to comp426.com. Navigate to the "assignments" page to see your results. Feel free to resubmit as many times as you'd like, but remember that the system will **always** consider your most recent score as your real grade for the assignment---even if you had a higher score before.