Setting Up a Web Development Environment

Introduction

This assignment describes how to install a development environment for creating Web applications. We will be using the same development environment throughout the semester. We will add new content every week, pushing the code to a GitHub source repository, and then deploying the content to a remote server.

Topics

- Integrated Development Environments - Installing IntelliJ
- Web Server - Installing Node.js
- Front end framework - Creating ReactJS Web applications
- Source control - Pushing source code to GitHub.com
- Hosting a Web application - Deploying Web applications to Netlify

Integrated Development Environments (IDEs)

Integrated Development Environments (IDEs) are software applications that help software developers to create software applications. They usually provide file management, editing, compiling, building, execution, and debugging. The following are examples of popular IDEs in the market

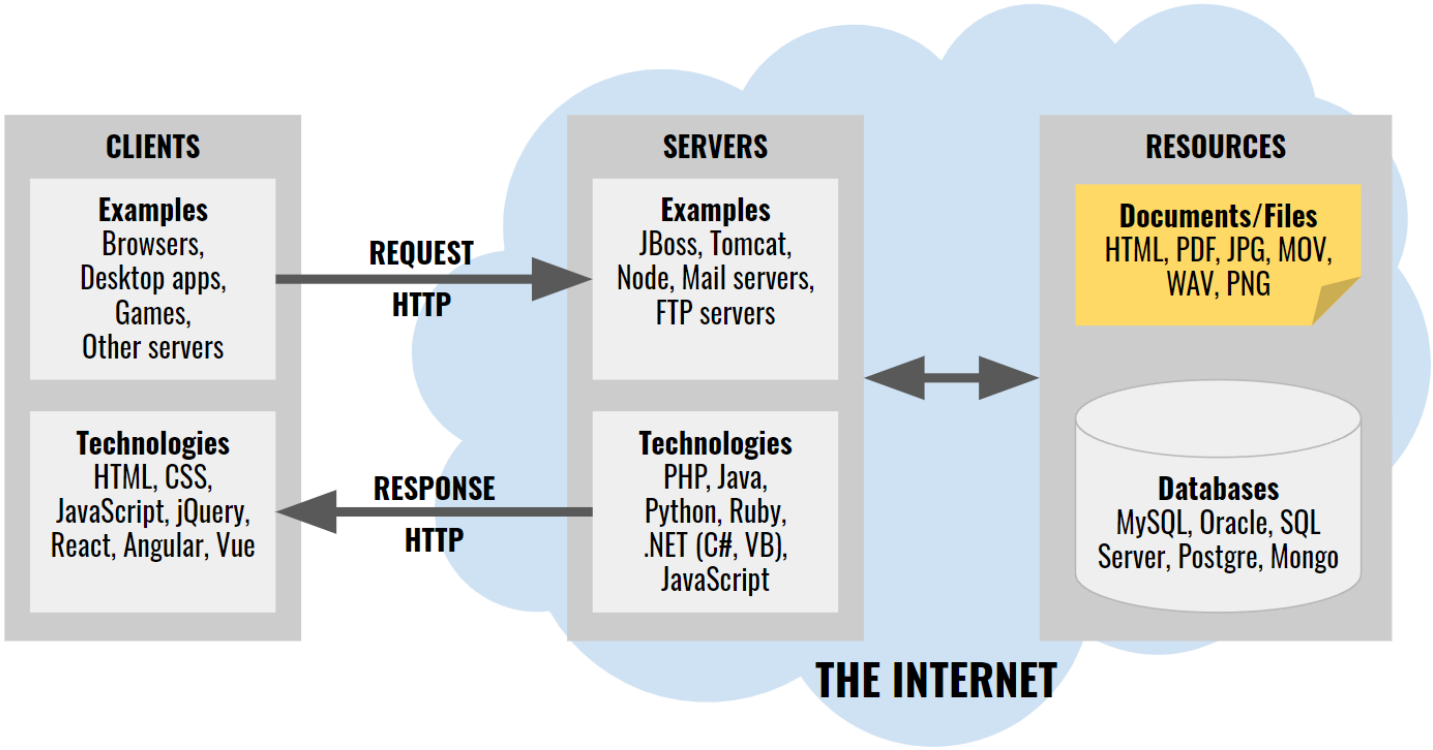
- IntelliJ
- WebStorm
- Visual Studio Code
- Atom
- Eclipse
- Netbeans

You are free to use any IDE, but we strongly suggest using IntelliJ Ultimate. The assignments, slides, and videos, developed for this course assume you are using IntelliJ. The teaching assistants will best be able to help you if you are using IntelliJ. To install IntelliJ create an account at the URL shown below. Use your university EDU email account to get the IDE for free.

https://www.jetbrains.com/

Once you have registered for an account, navigate to the URL shown below to download the IDE for your particular operating system and follow installation instructions

https://www.jetbrains.com/idea/download



Web Servers

Web servers are software applications that listen for network connections from Web browsers and respond with Web content such as HTML Web pages, CSS style format, images, music, videos, and JavaScript source code. There are many HTTP servers including:

- Tomcat
- GlassFish
- Jetty
- Nginx

The servers above were created in and are friendlier to different programming languages including:

- Java
- JavaScript
- Python
- .NET Visual Basic, C#, etc.
- Perl
- PHP

In this course will be creating Web servers from scratch using the JavaScript programming language running on local and remote computers. To run JavaScript we will need Node.js, a popular JavaScript runtime environment. Navigate to the URL below and download Node.js for your operating system

https://nodejs.org/

Download the recommended version and install it on your local computer.

Front End Frameworks

When you visit a Webpage on your browser, you are viewing a document formatted using the ***HyperText Markup Language*** (HTML). ***HTML*** documents are plain text documents that can be edited with any text editor. We refer to the software layer that interacts directly with a human as the ***front end***. As we demanded more from our Webpages, the industry evolved techniques, tools and frameworks to deal with the growing complexity. Many techniques were borrowed from other engineering fields. Today's front end frameworks implement advanced software engineering techniques and design patterns such as:

- Object oriented programming
- Component technology
- Singletons
- Services
- Model view controller

There are many front end frameworks to choose from and they all implement some or all of the techniques above:

- Angular
- Ember
- Sail
- Meteor
- React.js
- Dojo

This course will focus on ***React.js*** since it has become one of the most popular front end frameworks in the industry. Using ***npx***, a ***Node.js*** tool part of the Node.js installation, create a React.js project where we will be learning all about Web development. From the ***desktop***, or the ***root*** of your hard drive, or your ***home*** directory, create a directory for this semester, and then another directory under that for this course. Below are examples of creating directories from your home directory of your fille system using a macOS and then Windows OS

On macOS, start the terminal application and type the following to create a directory in ***~/2022/spring/webdev***

```
cd  ~                          # navigates to your home directory in your file system
mkdir  2022                    # creates a directory called 2022
mkdir  2022/spring             # creates a directory called spring inside 2022
mkdir  2022/spring/webdev      # creates a directory called webdev inside spring/2022
cd  2022/spring/webdev         # navigates to the directory webdev inside spring/2022
```

On Windows OS, start the console application and navigate to your home directory with the same command as for macOS. The only difference from macOS is that instead of a forward slash (/), Windows OS uses a backlash (\ )

```
cd  ~                          # navigates to my home directory on my file system
mkdir  2022                    # creates a directory called 2022
mkdir  2022\spring             # creates a directory called spring inside 2022
mkdir  2022\spring\webdev      # creates a directory called webdev inside spring under 2022
cd  2022\spring\webdev         # navigates to the directory webdev inside of spring inside of 2022
```

You are free to choose other places in your file system, but if you do, please make sure all directory names:

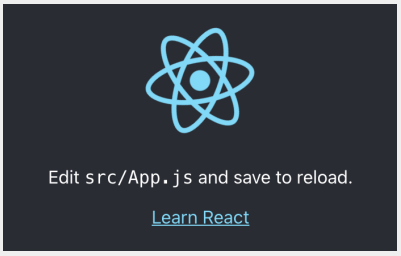
- are all lowercase
- do not have spaces in them
- are inside directories that also meet these criteria

Still from the terminal or console, navigate to the directory you created for this course and type the following to create a brand new React.js project called ***web-dev*** using the ***create-react-app*** Node.js module.

```
npx   create-react-app   web-dev
```

A new directory called ***web-dev*** will be created and lots of libraries and code will be downloaded into the new directory. Wait for the process to complete and then navigate into the new directory and run the project.

```
cd   web-dev
npm  start
```



Source Control

Install a git client

On macOS you already get a command line git client. You can fully interact with github.com from the terminal. On Windows OS, you'll need to install a git client from where you will be able to issue the same commands from a console. Download git for windows from https://git-scm.com/download/win, run the installer and follow the instructions. At then end of the installation you should be able to execute git commands from new console instances. All examples in this course assumes you have git installed in your OS.

You can also install a graphical git client if you prefer. There are a lot of alternatives, but the author prefers Sourcetree since it works well and consistently in both macOS and Windows. To install Sourcetree download from https://www.sourcetreeapp.com/, install, and follow instructions. We will not be covering how to use Sourcetree, but you are free to use it if you wish. All examples in this course will be using the command line git client.

Ignoring files and directories

Git can keep track of all the files in your project, but there are some files or directories that you don't want to keep track of, for instance, compiled classes, libraries, etc. You can configure which files and directories you want git to ignore by listing them in a file called ***.gitignore*** at the root of your project, which should already

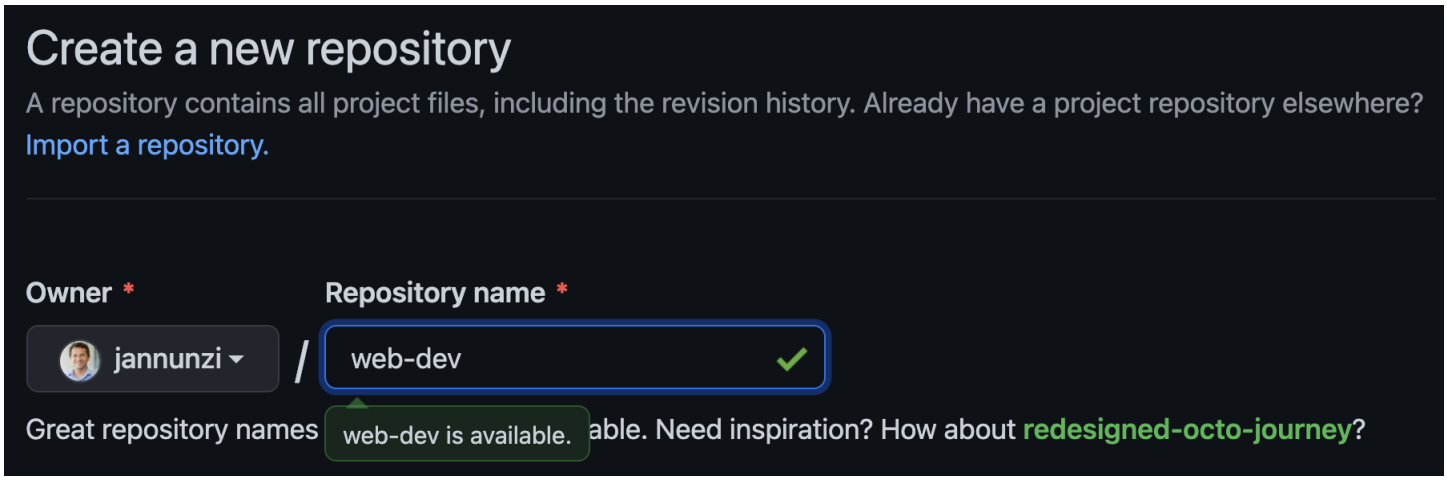
exist. With a text editor or from IntelliJ, edit the file ***.gitignore***. Towards the top of the file, in a blank line, type the following

```
.idea
node_modules
```

This tells git that the ***.idea*** and ***node_modules*** directory should be ignored because this is a directory specific to IntelliJ and not pertinent to React.js project itself. If you are using other IDEs, then you might want to add other files or directories here relevant to your specific IDE. The ***node_modules*** folder should also be ignored since it contains library files that should not be added to source control.

Create a remote source repository

If you don't already have an account on ***github.com***, create a new account or use an account you already have at ***github.com***. Do not use the university's github if you have one. Create a public repository called ***web-dev***, just like the name of the React.js project you created earlier. Here's an example on how it looks on my laptop. Your username "***jannunzi***" will obviously be different.



Once you create the remote repository on github, it will display commands on how to commit and push your code from your computer up to the remote repository. The commands will be similar to the ones shown below. The username "jannunzi" will obviously be different.

```
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/jannunzi/web-dev.git
git push -u origin master
```

Adding and commiting code

Now that we have a remote source repository, let's push our code. From the terminal or console, make sure you are in the web-dev directory and then type the following commands which are based on the commands git

suggested. Ignore the commentary on the right. Also, your actual URL below will differ for you, so don't blindly copy the example below. Use the commands git suggested for you.

```
git init                                                  # initializes local repository
git add .                                                 # adds all files to local repository
git commit -m "first commit"                              # commits files with messag (-m)
git remote add origin https://github.com/jannunzi/web-dev.git  # adds remote repository
git push -u origin master                                 # copies (pushes) local files to remote
```

Refresh the remote repository on github.com and confirm that the files are now available there

Deploying to a remote server

Create an account at https://www.netlify.com/ if you don't already have one. Follow the instructions to create a team or organization and then navigate to the ***Team overview*** screen. In the Team overview screen click on ***New site from Git***. In the ***Create a new site*** screen in the ***Connect to Git provider*** tab, select ***Github***. If you are asked to login to Git, do so as instructed. Give ***Netlify*** all the authorizations it asks for. In the ***Pick a repository*** tab, in the ***Search repos*** field, type ***web-dev*** to lookup the web-dev repository you created earlier. Select the ***web-dev*** repository. In the ***Site settings, and deploy*** tab, select the branch to deploy from, e.g., ***master*** or ***main***, and click ***Deploy site***. The deployment process might take a few minutes. Netlify will pick a random silly name for your application, e.g., dreamy-mcclintock-e0a4ee. That's fine for now, we can setup custom domain names later. While your project deploys it will show the main steps it's going through and will eventually say Published in green half way down the screen. You can then click on the URL towards the top, e.g., https://loving-torvalds-effde8.netlify.app and see your project deployed.

Deliverables

As a deliverable, provide the URLs to the application running on Netlify.com and the URL of your repository. Copy and paste them in Canvas or Blackboard.

- URL to your remote Website
- URL to your remote GitHub site