**Setting Up a Web Development Environment**

# 1 Introduction

This assignment describes how to install the development environment for creating and working with Web applications we will be working on this 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 hosted on Netlify.

## 1.1 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

# 2 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

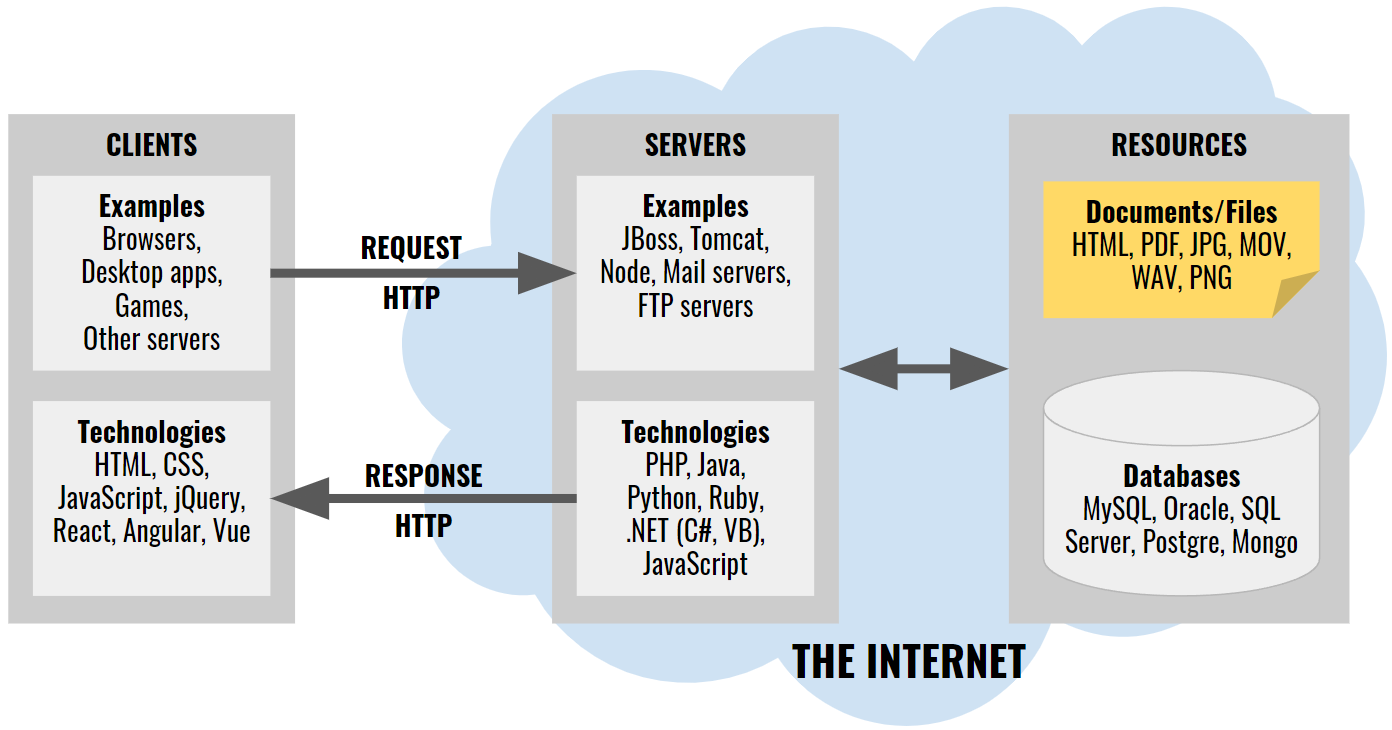
* IntelliJ
* WebStorm
* Visual Studio Code
* Atom
* Eclipse
* Netbeans

You are welcome to use any IDE, but we strongly suggest using [IntelliJ Ultimate](https://www.jetbrains.com/idea/download). 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>



# 3 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.

# 4 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
* Vue.js
* Dojo

This course will focus on ***React.js*** since it has become one of the most popular front end libraries 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 file system using a macOS and then Windows OS

On macOS, start the ***Terminal*** application. On Windows OS, start the console application ***PowerShell***. On either operating system, type the following to create the directory ***~/2022/fall/webdev***

| **$ cd ~**  **$ mkdir 2022**  **$ mkdir 2022/fall**  **$ mkdir 2022/fall/webdev**  **$ cd 2022/fall/webdev** | # navigates to your home directory in your file system  # creates a directory called 2022  # creates a directory called fall inside 2022  # creates a directory called webdev inside fall/2022  # navigates to the directory webdev inside fall/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 ***tuiter-react-web-app*** using the ***create-react-app*** Node.js module.

| **$ npx create-react-app tuiter-react-web-app** |  |
| --- | --- |

A new directory called ***tuiter-react-web-app*** 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 tuiter-react-web-app**  **$ npm start** |  |
| --- | --- |

# 5 Source Control

## 5.1 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 the 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.

## 5.2 Ignoring files and directories

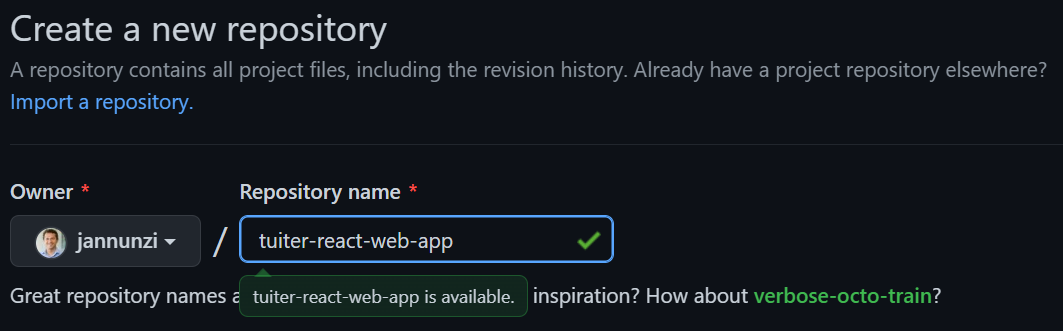
Git can keep track of all the files in your project, but there are some files or directories that you don't want to keep track of, for instance, compiled classes, libraries, etc. You can configure which files and directories you want git to ignore by listing them in a file called ***.gitignore*** at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

| ***.gitignore*** | |
| --- | --- |
| .idea  node\_modules | |

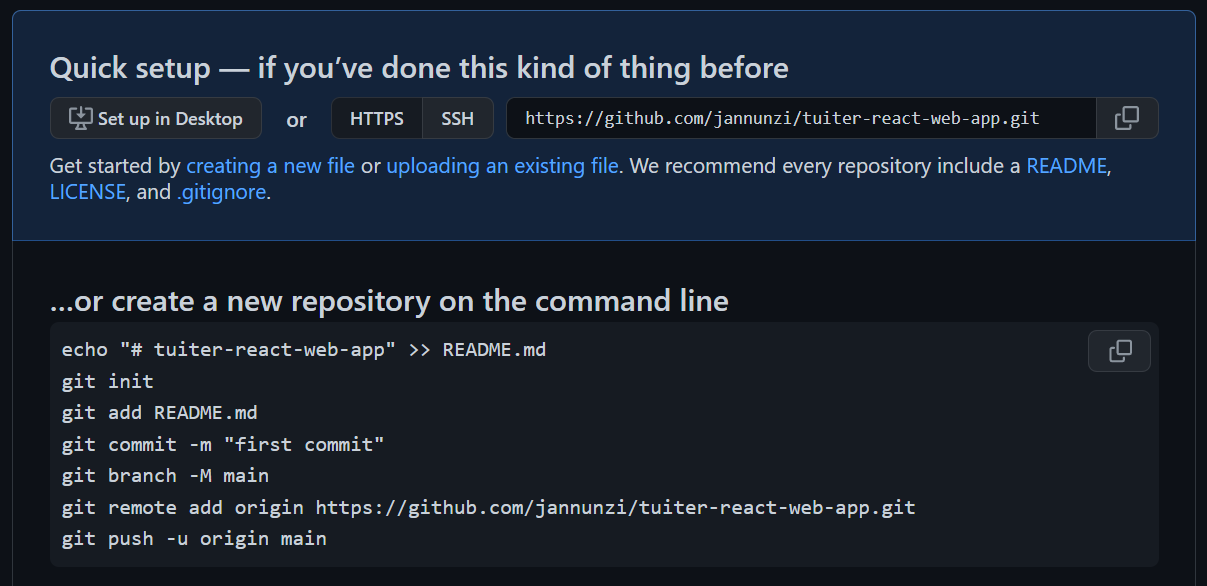
This tells git that the ***.idea*** folder should be ignored because it is a directory specific to IntelliJ and not relevant to React.js project itself. If you are using other IDEs, then you might want to add other files or directories here relevant to your specific IDE. The ***node\_modules*** folder should also be ignored since it contains library files that should not be added to source control. ***Note***: make sure that IDE specific folders and the ***node\_modules*** folders don't make it into the repository!!

## 5.3 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 or your work's source control if you already have one. Login to your ***github.com*** account and click the .***New***  button to create a new ***public repository*** called ***tuiter-react-web-app***, 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.



## 5.4 Adding and commiting code

Now that we have a remote source repository, let's add, commit and push our code to the repository. From the terminal or console, make sure you are in the ***tuiter-react-web-app*** 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**  **$ git add .**  **$ git commit -m "first commit"**  **$ git remote add origin https://github.com/jannunzi/tuiter-react-web-app.git**  **$ git push -u origin master** | # initializes local repository  # adds all files to repository  # commits files with message  # adds remote repository  # copies local files to remote | |
| --- | --- | --- |

Refresh the remote repository on [github.com](https://github.com/) and confirm that the files are now available there

## 5.5 Using personal access tokens

While pushing to GitHub you might encounter an error stating that ***password authentication was removed on August 13, 2021***. One way to fix this is to generate a ***personal access token*** and use that instead of your password. To generate a personal access token go to your GitHub ***Settings***, and then ***Developer Settings***. Click on ***Personal access tokens*** and then on ***Generate new token***. Enter a short description in the ***Note*** field, select ***No expiration*** for ***Expiration***, and grant all access privileges by selecting all the checkboxes under ***Select scopes***. You are welcome to be more restrictive it you want. Click on ***Generate token*** and copy the long unique access token to your clipboard. Note that this will be the only opportunity to copy the token and you fail to do so you'll have to delete this token, create a brand new one, and try again. With the token copied to the clipboard, try pushing again to GitHub, but this time paste the token when asked for a password.

If you are not being asked for a password then GitHub might be using a cached authentication. To clear cached GitHub authentications on Windows go to the ***Credential Manager***, click on ***Windows Credentials***, click the GitHub credentials, and click ***Remove***. This time github should ask for your username and password again when trying to push. Paste the access token when asked for the password. To clear cached authentications on macOS, go to your ***Key Chain*** and search for github. Remove the GitHub key chain.

# 6 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. On the ***Team overview*** or ***Sites*** page, click on ***Create/Add new site***.Select **"Import an existing project".**

* Connect to **Git provider**, select **Github.** Give **Netlify** all the authorizations it asks for. **Pick a repository**, in the **Search repos** field. **Lookup and pick the** **tuiter-react-web-app** repository you created earlier
* 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., **loving-torvalds-effde8.** 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.**

# 7 Deliverables

1. Download and install the latest ***Node.js*** as described in [section 3](#_p6zqmqj5j8d)
2. Create a React.js application called ***tuiter-react-web-app*** as described in [section 4](#_olkpu99ezb4a)
3. Push the source code of the React.js application ***tuiter-react-web-app*** to a remote source repository in ***GitHub.com*** as described in [section 5](#_8imqgifl8kn)
4. Deploy the ***tuiter-react-web-app*** React.js application to ***Netlify*** as described in [section 6](#_vt5n8l1nk96v)
5. As a deliverable in ***Canvas***, submit the following URLs
   1. ***(50pts)*** The source repository in GitHub.com. It should look something like  
      [**https://github.com/jannunzi/tuiter-react-web-app-5610-fa22**](https://github.com/jannunzi/tuiter-react-web-app-5610-fa22)
   2. ***(50pts)*** The React.js application running on Netlify. It should look something like  
      [**https://roaring-hummingbird-1d48d4.netlify.app/**](https://roaring-hummingbird-1d48d4.netlify.app/)