# Working with Git

## Installation:

You can download Git for free from the following website: <https://www.git-scm.com/>

Git Bash is included in Git for Windows

git --version

## Configure:

git config --global user.name "amugisha7"

git config --global user.email "amugisha.ug@gmail.com"

Use same details as registered with github

## Initialize:

git init

Done inside folder that will hold the project

## Staging:

git add index.html

(use "git rm --cached <file>..." to unstage)

(use git add –all to stage all the files in the folder)

## Committing:

git commit -m "we have made changes that should be in this comment"

committing without staging, use -a

git commit -a -m "The commit has been made without staging"

## Git Log

To view the history of commits for a repository, you can use the log command:

git log

## Fetching from Github

git fetch origin

to check the difference, use git diff origin/main

if satisfied with the changes, use git merge origin/main to update your local repository

git pull bypasses all this

## Clone from Github

1. Fork link
2. Git clone link (on terminal)
3. Cd to new folder
4. Git remote rename origin upstream
5. Git remote add origin forked-like
6. Make changes, add and commit
7. Git push origin
8. Go to git and create pull request

# Resources link

<https://github.com/zero-to-mastery/complete-web-developer-manual>

# Bootstrap Mobile Responsiveness:

<https://getbootstrap.com/docs/5.2/getting-started/introduction/#quick-start>

# Responsive Full Size Background Image:

html {

background: url(images/bg.jpg) no-repeat center center fixed;

-webkit-background-size: cover;

-moz-background-size: cover;

-o-background-size: cover;

background-size: cover;

}

# Using CSS Grid to create responsive columns

Set your content to have class container.

.container{

display: grid;

grid-gap: 20px; /\*optional\*/

grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));

grid-template-rows: 1fr;

}

# New Javascript Syntax

## Objects (Destructuring):

var person = {

age: 60,

name: “Andrew”,

saved: true,

}

var age = person.age;

var name = person.name;

var saved = person.saved;

//becomes

var {age, name, saved} = person;

# Object Properties:

let a = 30;

let b = “hi”;

const c = true;

let newObject = {a, b, c};

d = 30;

e = “great”;

f = false;

newObject = {d, e, f};

# Strings

let message = `Hello ${firstName} have I met you before? I think we me in ${city} last summer no???`;

# Functions

function isAgeValid(age) {

return age;

}

//becomes

let isAgeValid = (age = 10) =>age;

function whereAmI(username, location){

if (username&&location){

return “I am not lost”;

} else {

Return "I am lost”;

}

}

//becomes

const whereAmI = (username, location) => {

if (username && location) {

return "I am not lost";

} else {

return "I am totally lost!";

}

# Lesson 203 – Find Recruiters/Job by LinkedIn

# Lesson 207 – Subscribe for Industry Updates

# Using the Terminal

|  |  |
| --- | --- |
| **Command** | **Use** |
| ls and dir | List all contents of folder |
| pwd | Check which folder you are in |
| cd .. (note space btn cd and ..) | Go 1 directory up |
| Clear | Clear screen |
| cd / (note space) | Go to top most directory |
| cd ~ (note space) | Go to user directory |
| start . | Opens the current folder |
| Mkdir myfolder | Create a new folder called myfolder |
| Touch index.html | Create a file called index.html |
| Start index.html | Open the file |
| Use “ “ when the name has a space in it |  |
| Mv index.html newindex.html | Rename index.html to newindex.html |
| Rm index.html | Delete index.html |
| Rm -r myfolder | Delete myfolder |

# Git Hub (add – commit – push)

|  |  |
| --- | --- |
| **Command** | **Use** |
| Git clone https://github.com/amugisha7/data1.git | Copy files in data1 project to current directory |
| Git status | Shows status of git directory. Must be run inside folder that is an active git project |
| Git commit -m “my comment” | Commit changes with a comment |
| Git branch | Check which branch you are in |
| Git branch littlefeature | Create a branch called littlefeature |
| Git checkout littlefeature | Go to littlefeature branch |
| Git checkout -b newleaf | Creates a new branch called newleaf and goes to it |
| Git push origin newleaf | Pushes to the master of newleaf |
| Git diff | Checks what changes have been made |
| git remote add upstream https://github.com/zero-to-mastery/start-here-guidelines.git  git pull upstream master | Syncing a Fork with the main repository |

# NPM

* Use sudo npm command to run commands an an administrator
* Npm init creates a package.json file

# React

## Getting started:

Npx create-react-app my-app

npm install tachyons to install tachyons. In Index.js, import ‘tachyons’;

npx kill-port 3000 to free up port 3000

How to Style in line and with tachyons:

<div className="br2 shadow-2 center" style={{ height: '300px', width: '300px', background: "#fff"}}>

## Importing:

import "./Logo.css";

import cclogo from "./cclogo.jpg";

## Inserting images in div:

<img src={cclogo} alt='logo'/>

## Changing Function to Class:

class myFunction extends React.Component {

render () {

return (

<div>

</div>

)

}

}

export default myFunction;

## Giving a Class Props

<myFunction greeting={`Praise the Lord!`} id={1}/>

Inside myFunction’s return:

return (

<div>

<p>this.props.greeting</p>

<img src={`https://picsum.photos/200/300?random=${this.props.id}`}/>

</div>

);

When it is a Function, remove the “this” before .props and pass props as an argument.

Format of a JS object with data:

export const MyData = [

    {

        id: 1,

        name: 'Andrew',

        username: 'andrew1',

        email: 'andrew1@abc.com'

    },

    {

        id: 2,

        name: 'Mugisha',

        username: 'mugisha1',

        email: 'mugisha1@abc.com'

    },

    {

        id: 3,

        name: 'Blessed',

        username: 'blessed1',

        email: 'blessed1@abc.com'

    }

]

## Adding State to a Class:

class App extends Component {

    constructor(){

        super()

        this.state = {

            myData: MyData,

            searchField: ""

        }

    }

    onSearchChange(event){

        console.log(event.target.value);

    }

    render(){

        return(

            <div className="tc">

                <h1>My Testing App</h1>

                <SearchBox searchChange={this.onSearchChange}/>

                <Group1 newData={this.state.myData}/>

            </div>

        );

    }

}

export default App;

## Setting the State

onSearchChange = (event) =>{

        this.setState({searchField: event.target.value});

## Destructuring state properties in render()

const{ newData, searchField} = this.state;

## Creating Final Folder

Npm run build