**W01 Creating a Portfolio**

*Doing Stuff with Web Things (The textbook for this course)*

<https://github.com/yenrab/doing_stuff_with_web_things>

*LocalStorage*

(Pg. 35-44 of Doing Stuff with Web Things)

* Local storage stores information, even if the user leaves or refreshes the page
* It uses key value pares. You can use the key to get the value

Text

Description automatically generated

Here is a link to this example: <https://glitch.com/edit/#!/jadens-epic-story-editor?path=script.js%3A15%3A84>

# Portfolio

One of the requirements for this course is to develop a portfolio of the work that you complete for the course. You will turn this in at the end of the semester along with a self evaluation. These items will be used to determine your grade.

This activity is to give you some ideas for managing this portfolio.



### Code Management

Most of your portfolio for this course will be made up of code. It would make sense then, to use a tool specifically designed to help manage code to manage your portfolio. Create a repository for your work for this class in [Github](https://github.com/" \t "_blank) >, a free hosted code repository with the core ability to track changes.

If you aren't sure how to get a version control repository setup, here is a cheatsheet using Github:

* 1. Create an account with [Github](https://github.com/" \t "_blank) if you don't already have one.
  2. Once logged in find and click on the "New Repository" button.
  3. It will ask for a name, description, and whether to include a "README". Fill out the name, description, and check the Readme box.
  4. You should now see a screen listing out the files in the repository (it's a very short list) There should also be a big green button labeled "Clone or download" Click on it and copy the URL in the box. (should look something like **https://github.com/matkat99/port.git**
  5. At this point you need to decide if you want to manage your repository with the command line or with a GUI tool.
     + GUI Tool
       1. Download and install the [Github Desktop](https://desktop.github.com/" \t "_blank) app. Launch it and sign in with your Github credentials.
       2. Click on "add a new project" (usually the +) and select "Clone" If you have authenticated to your Github account correctly then you should see your repository listed. Select it and click "Clone Repository". It will ask you where it should store it. Select a directory on your harddrive.
     + Command line

Make sure your computer has git installed. Open your command line prompt (Terminal.app on the Mac, CMD on Windows), then type

**git --version**

If you get an error go [install git](https://git-scm.com/download).



Navigate to the folder where you want the project stored on your harddrive. If you are unsure how to do that then lets just put it on your desktop. You will automatically start in your User directory when you open a new command prompt...and it's easy to get back there. Type:

**cd ~**

on the Mac or for Windows type:

**cd %user\_profile%**

then simply type:

**cd Desktop**



Enter **git clone <URL copied from github> <name of directory you want created>** Mine for example would look like this:

**git clone https://github.com/matkat99/port.git portfolio**

* 1. Now you can open that folder up in your editor and begin making edits. You should probably also run through something like [try.github.io](https://try.github.io/levels/1/challenges/1) to start learning how to use git.



### Index page

At the root of your repository you should create an index.html file. In this file you should keep a directory of all your notes, exercises, challenges, etc that you add to the portfolio.



### Turn on Github pages

Log into Github and enter the repository for this class. Goto Settings and on the Options page scroll down until you find the section on Github Pages. In the dropdown under Source, choose **master branch**. You should now be able to browse to your repository at the URL Github shows you after the page refreshes. It will look like this: **https://username.github.io/repositoryname/**



### Dynamic Table of Contents

Each week you will be adding to the table of contents of your portfolio. Let's create that list of links dynamically with some Javascript as a review of what you learned in WDD 230 (CIT 230/CS 213). Create a directory called **js** in your portfolio, and add a file called **main.js**. Add the appropriate **script** tag to your html and write a JS function that will do the following:

* 1. Select an **ol** element from your index.html page (This means you need to add the html for this element to your page. You can't select something that doesn't exist.)
  2. Read a list of links from an array. (You will also need to create this array)
  3. For each of the items in the array of links you should create a **li** element, add an **a** element with the label and url from the list of links, and add the new **li** element into the **ol** element you grabbed above.

You can keep the structure of your list of links simple, you will need a label that will be shown to the user, and the URL of the page that should open when the link is clicked. It might look something like this:

**const links = [**

**{**

**label: "Week1 notes",**

**url: "week1/index.html"**

**}**

**]**

Each week you will simply add new item(s) into your array to have them automatically show up in the table of contents.



### Record your work

As you read, take notes and complete exercises record your work in this portfolio, making sure to keep the index up to date. If you do then turning in the link to your portfolio will be the best way to turn in your work each week, and at the mid and final checkpoints.

## Example

Let's look at an example of how this might look in practice. Visit this [WDD 330 repository](https://matkat99.github.io/wdd330-portfolio-example/) on Github. Notice that I created an index file to list out all of my work. I included questions from each week, and an indication of how I spent my time that week

If you look at the [github source code](https://github.com/matkat99/wdd330-portfolio-example" \t "_blank) you will see that there are also folders to hold my work from each week.

If you drill into those folders you will see additional directories for organization. Drill into **"objects"**. You will see an **index.html** file and a couple of javascript files.

If you look at the **index.html** you will see that is is basically the template file from the first Doing Stuff chapter, with a couple of extra **<script>** elements added to the bottom to include our javascript. The index is really just to make it easier to view our examples in the browser console.

Take a moment to look through the **superObjects.js** file. Notice that I have many of the code examples from the chapter in there...I used a lot of **console.log()** commands to run many of the examples. I also used this as an opportunity to record my notes from the reading and my thoughts about the examples as well.

My workflow would be to open the **index.html** file in my browser, then open up the browser console, then open the chapter I am currently working on. If you were to pull these files down and do that you would see bunch of output. You could also try additional commands right there as well.

Note that you will not see **all** of the examples from the chapter in the .js files!...many of the examples were exploratory in nature so I simply ran them in the console of the browser once I had my **index.html** page open there. It is much faster to try lots of variations of a command there than it is to code it all up in a JS file. Your goal is to get enough of the example code into the file that you can experiment in the browser console and record your findings as comments.

If you follow a similar system while working the examples yourself...you will find your portfolio ready to show at any time. When it is time to start the first Challenge phase, create another directory in your repository for your projects and they will be added as well.

## Resources

Tools such as Github are powerful, but do require a bit of work to learn. I haven't mentioned anything about using them in this outline. There are many good free resources to get started with them if you don't have any previous experience. A search will bring many of them up easily. If you use Github, it has a very good getting started tutorial right on their site to get you going.

It will be very worth your time to gain some mastery of at least one of these tools.

*How to use Git*

<https://docs.github.com/en/get-started/quickstart/set-up-git>

Graphical user interface, text, application

Description automatically generated

*Link to my GitHub portfolio*

<https://jadenmounteer.github.io/Web-Frontend-Development-II-Portfolio/>

*Example Portfolio*

Source Code: <https://github.com/matkat99/wdd330-portfolio-example>

Portfolio: <https://matkat99.github.io/wdd330-portfolio-example/>

**Master Mobile UX Course**

<https://www.sitepoint.com/premium/courses/master-mobile-ux-2952>

*Master Mobile UX*

* Design for mobile first. You will focus on functionality. This helps you to focus on your content and your functionality. When you start small and build up, it makes everything better.
* Just because your website it responsive, does not mean it is mobile first.

*Improving UX by understanding users*

“It is really hard to design products by focus groups. A lot of times, people don’t know what they want until you show it to them.” – Steve Jobs

* 1. Be there for the user.
  2. Be Useful. Provide the users for what they came for as quickly as possible.
  3. Be quick. A single tap or click makes a difference.
  4. Reachability matters. Think about what fingers the users will be using. A lot of people use their thumbs at the bottom of the device. Make key items in easy to reach places.
  5. Make things load quickly. Reduce image usage. Use SVG/webfonts are lower bandwidth. Optimize and minify CSS and JS files. Remove whitespace and unused code. GZIP files to eliminate unnecessary data.

*Better Mobile UX with Strategic Menu Design*

Be empathetic to user’s needs.

Move the primary controls to the bottom of the screen.

Graphical user interface, application

Description automatically generated

Don’t settle for “good enough”. Always strive for amazing.

For a menu button, have a menu with a border. Hamburger menus are confusing to users.

*Graphical user interface, text, application, chat or text message

Description automatically generated*

Evolve with the tech and do research on how to improve the user experience.

*Designing Intuitive and Mobile Friendly Forms*

Ditch the labels and have them display inside the fields instead.

Make the inputs taller.

A picture containing text, monitor, screenshot

Description automatically generated

Use float labels for when the user clicks on the field.

Background pattern

Description automatically generated with low confidence

Make the fields to select larger! Only ask for the information you need. Make fields appear/disappear based on user input.

Break the form into sections. “Pagination”.

Enable autocomplete.

When a user enters a password, show passwords by default, but allow them to mask the password if they need to do so.

For credit card info, space the numbers for them, auto advance to the next field.

Present the users with the correct keyboard that you need. For numbers, bring up the number pad.

For phone numbers, make them an anchor tag so they can make the call just by tapping on the number. Make a popup that asks them if they want to call.

*Designing a better mobile advertising user experience*

Before you advertise, research. Know how people are using a device.

Don’t use popups. People hate them.

Make ads inside content small and simple. You can make a larger ad appear when the smaller one is clicked.

You can use video ads.

Make the small ad a window that shows the full ad as the user scrolls.   
People did not come for ads; they came for content. Give them content first.