CodingDojo Project Guideline by Michael Choi
(last updated on April 13th 2018)

Having a well built portfolio is one of the most important items you can work while in CodingDojo, especially if you're looking to change your career. Please first do everything you can to get your belt and once that is done, work diligently to work on your project. You should have at least 3 strong projects you can showcase, with a strong resume and cover letter, ***before*** you start applying for jobs. Having a strong portfolio will significantly increase your chance of getting the exposure you need to impress potential employers. Weak portfolio will make it extremely difficult for you to get any interviews.

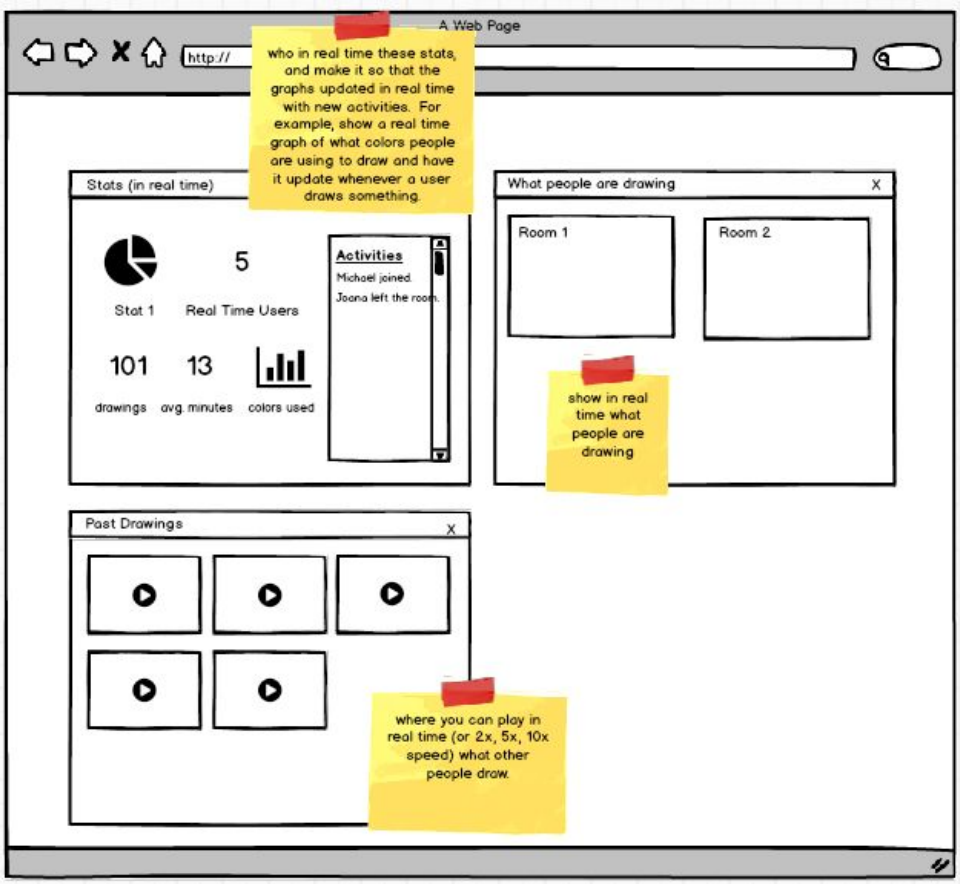
Project Topics
There are technologies that 90% of the developers know how to do extremely well. There are also technologies that we teach at the Bootcamp or which you can explore while you're at the bootcamp, that majority of the developers don't know or haven't experimented with. It's the latter that at least 2 of your projects should focus on. If all of your three projects are within the realm that other developers already know how to do extremely well, your project/portfolio won't stand out.

So what are these topics that dwell in this 10%, where 90% of the web developers don't know and wished they had time to learn? In other words, what should at least 2 of your projects be focused on?

Some of these include

1. Anything done in real time using sockets. All web developers know how to do Ajax but still a vast majority haven't learned how to use Node and how to build something real time with sockets.
2. Everyone can claim to have used someone else's library, but how many can say that they've created their own gem, npm module, their own MVC framework (either front end or backend)? How more impressive would someone be if you said you built and deployed a new ruby gem, a new npm module, javascript library (e.g. that generates amazing charts, etc), your own new MVC framework, etc, and explain in the resume details of these gems, modules, libraries, or frameworks? It will sure make you look better than someone who's only used to using other people's libraries, gems, frameworks, etc.
3. Responsive Web Design. Learning tools like LESS, SASS, etc. Everyone claims to know CSS but how many web developers can say they know responsive design, LESS, and SASS. To take this further, how cool is it if you could say you built your own css library like Bootstrap but leveraging LESS and also your own design philosophies (e.g. 16 grid system vs 12 grid system, different css reset techniques, etc)?
4. There are great visualization tools such as D3JS (and other tools you can ask your instructors). What about using these in a creative way and even better integrate socket so that any changes that are made can be updated in real time using d3JS? Alternatively, what if you were to use D3JS to demonstrate/visualize how different sorting algorithms or different data structures work? How about using data sets such as what's available in Google BigQuery or in Data.gov and using d3JS to show some really interesting insights? Even better, make it so that these graphs/visualizations are moving/updating so that when someone clicks on your project, they see the graphs moving/updating, etc. For example, maybe your projects shows how some of these data/trends have changed in the last 3-5 years and there's a slide on the top where you can adjust the speed of these changes and the scope of the data you're updating. This way, without someone even adjusting the slider or the time span, your project automatically shows how some of these data have changed over 3-5 years. For example something like https://www.census.gov/dataviz/visualizations/010/ but where using D3JS, you have the map automatically update from 1940 to current time in 5 seconds (without requiring someone to click, etc).

5. Anything with Java/Android or iOS development. Most web developers do not know how to build things natively. If you had the opportunity to learn iOS or Android, make sure you build a project using these tools and see if you can use sockets to integrate your app with something on the web. For example, maybe create a multiplayer drawing app (on either the phone or through a web app) that shows in real time what other people are drawing (or even better have it replay how others draw things in 1x speed, 2x speed, 5x speed, etc). If you built something using Swift, see if you can deploy that in the App store, as not many developers can say that they built an app and launched it in the App store. Please note that 80-90% of the web developers do know how to build things with iOS and that iOS development really requires someone with a lot more experience/knowledge than someone who's a true new beginner.
6. Knowing Angular or React is a huge advantage but do make sure that your project is really showing a nice single page application. For example, don't build the wall assignment using Angular or React as it's not impressive, but what if you could build a simplified version of Google Doc (with multiple people editing the doc) or Google Slides with simple menu bar on the top? Again, integrating sockets with this would be super impressive. Or if you were thinking about the drawing app, what if you had multiple windows that you could move around that showed different activities? For example, something like below



7. You all learned Python and you know that Python has some great libraries for data scientists. Why not pick up face recognition tools, other cool libraries that do something interesting, and build something? For example, we had one student build a simple web app that detects which way you're looking and depending on this rotate the image on the web screen (e.g. apple, picture of a car, etc) based on how much you've rotated your face. For example you see this image rotate to the left when you look to the left, rorate to the right, when you're looking at the right, etc.

Other Tips

1. If you have only used MySQL say up to this point, project week is a great time for you to get exposed to say Postgres, SQLlite, Redis, MongoDB, and other database types. They are all very easy to learn, but it looks much more impressive to say that you've played with Redis, Postgres, MySQL, MongoDB than to say you've only done things with MySQL. Also note that SQLite is great for total beginners but you'll probably never use SQLite in a production setting. This means that it won't be impressive enough to say that you know SQLite and MongoDB for example.
2. If you have already graduated, please remember that the Dojo Academy has live instructor led classes to help alumni with algorithms. Take advantage of this and attend all algorithm sessions while you're working on your portfolio. All of these classes are for free for our alumni, for now. Reach out to your instructor if you don't know how to attend these sessions.
3. Note that starting this month, CodingDojo will host its own hackathon where alumni can participate and win recognitions. If you're needing more projects, you should really participate in these hackathons so that you can get a project and perhaps even win some of these hackathons (which will make you look good on your resume)
4. If you haven't already done so, make sure your resume, cover letter, and 3 projects are all approved and reviewed by your instructor before you start applying for jobs.
5. ***CRITICAL***: Before you build out your project, **ALWAYS** build out the wireframe and make sure your wireframe is no more than 3-4 pages. Your project should not be too complex and should be very simple. Remember that most people will look at each of your project for no more than 5 seconds so your first page must be the most impressive. Once your wireframe is done, **ALWAYS** build out the project in just pure HTML, CSS, and Javascript. This is often called clickable prototype. Don't build out the functionality until AFTER you've built out the front end first. This process alone often will save you 3-5 times the work (compared to you building this out without a wireframe or a clickable prototype). This process will also make you look better when you have to explain the steps you took to build out your project (to potential employers or if you're demoing this project to your cohort members).
6. Only amateurs deploy things using Heroku. Do not ever just deploy things on Heroku and think you're done. Most employers will not interview you as they may think you're a true beginner who doesn't know how to deploy things for real.

Next Steps

1. If you haven't already, work on your project idea and wireframe your thoughts using Balsamiq, InVision, or other wireframe tools. Make it simple. Get your instructor to review, give you feedback, and give you permission to move forward before you build out the project.
2. Develop a clickable prototype in HTML, CSS, and Javascript first. Show your instructor and get feedback before implementing the features.
3. Build the features. Make sure that your first page is the most impressive (as most people won't go past your first page of your project).

Project Ideas

1. (Idea from MC) An app that converts keyboard activities into music. Imagine you also use WebRTC or something similar to capture in real time where your users are doing and where you can replay how these musics are composed. For the main project page, you can create a new music or watch in real time what others are doing as well as see past re-creations of how others users have created their own music.
2. (Idea from MC) Using D3js, have a rotating globe that shows where people in the world are currently tweeting and what they are tweeting about. Store the activities of the last 24 hours so that you can go back to what people were tweeting about say 10 hours ago. Use data visualizations so that in the first 5 seconds of your project showing, it leaves a very strong impression.

3. (Idea from MC) E-commerce site with real time dashboard that shows all the current users, their geo-locations, which items were added, removed, etc (basically that anything that you think the website owner would be interested in knowing). When an item is added to the cart or removed or whatever, it updates the dashboard.
4. (Idea from MC) Creating a live video chat with WebRTC that automatically converts the language you speak to another language (using Google Speech API). For example, what if you were talking to a friend that doesn't speak your language but if they used your app, they could each speak in their own native language but your app translates the text. Have your app also record these activities where in the dashboard, you can watch past conversations of others using your app and where it shows snapshots of the videos as well as in real time (or 2x, 5x, 10x speed) the conversation that folded through your app.
5. (Idea from MC) Say you wanted to build an app where you can type solutions to some algorithms (e.g. sorting algorithms, simple string questions, etc) and where your app records not only whether the algorithm passed certain tests but also records in real time what you're doing. On your dashboard page, imagine you can see all the different people who took your algorithm app and you can playback to see how they did it in real time (e.g. or 2x, 5x, or 10x speed). If you know Angular, imagine that the app allows you to click on the new user's name (that's currently using the site) that would open a new dialog box where you can see in real time what they're typing in real time (and where you can see what they had typed a few minutes ago by adjusting a slider bar).
6. (Idea from MC) Build a javascript library that others can use, which records all the keyboard strokes as well as mouse movements from the users. On your project dashboard, show video replay of what users did on the website (for those that had this javascript library). You can also show all the websites that are using your javascript library and show the current number of active users on each of the site. Use D3js and other charts to make your project dashboard look super impressive (where it shows where the user put their mouse, where they clicked, etc). You could also have something where it displays the heatmap of where people clicked, dwelled, etc.
7. (Idea from MC) Build an iPhone app that shows some sensor data (e.g. accelerations, location coordinates, x, y, z coordinate info, etc) on the iphone screen. Have each app send this data to your central node server that shows in real time what all the users of your app are doing.
8. (Idea from MC) Using Google Speech API and WebRTC, whenever you say certain words, have those objects appear somewhere in your camera screen! For example, if you say "Apple", have a small "Apple" appear on your video. Then when you say things like "move to the left", have the apple move to the left of the screen. If you say "get bigger", have the object get bigger. You can go crazy and if you say things like "explode now", have the object then explode with some cool animation. Put safe objects and safe words...
9. (Idea from MC) Using Google Speech API, build a small voice assistant that understands commands like "bring up cnn.com", "move that window to the left", "open a new window for espn.com", "show me weather data for me", where it retrieves these information/sites and opens up a new dialogue/module that you can move around with the mouse (or tell the voice assistant to move). You could on the right show all the voice commands you gave. You could show other voice commands others users on your site are giving.
10. (Idea from MC) Build multi-player ninja snowball fighting game. Have each of your cohort login to fight with each other (by throwing a small cut snowball). Show who's winning (by displaying names and scores on the right). Capture these activities so that in your project dashboard, you can see other snowball fight sessions and replay that session. Make it even cooler by allowing video feed of these users to show up.
11. (Idea from MC) Get a drone where you can program it using Node or Python or a language you know. Using socket, build something where you can command the drone using your iphone or using your computer. Record a demo of this and add to your project site.
12. (Idea from MC) Get an Arduino with bunch of cool sensors (e.g. temperature sensor, radar, etc). Have these sensors data be sent to your web app where your web app visualizes in real time some of these activities. This could be a great opportunity to build something hardware and integrate with your software skills.
13. (Idea from MC) Treasure Hunt. Using your web app, allow you the admin to set up different treasures at different sites. Then with the phone, have it tell you how close you are to the nearest treasure. Once you get within x feet of the treasure, have the phone tell you how much you recovered (e.g. 50 ninja gold). On your web app, show in real time, where the current users are, locations of your treasures as well as the values for each treasure. Show a scoreboard of the top users which would be updated whenever a new treasure is found.
14. (Idea from MC) Create a web app where two people can create a book together. The app gives a one sentence intro of the story (e.g. pulled from random variations that is stored in your database). One person would start by having say 10 seconds to record something (which transcribes the voice to the text). Then the next person has 10 seconds to ponder and then 10 seconds to speak. They alternate. The app could for example ask how many back and forth you would

want and then start this. Record what each person have done and show recordings of the stories others have created through this app. For example, in the beginning when you create a new book, the app could for example say “You’re together eating an ice cream when all the sudden,” where it’s the next person’s turn to finish the sentence.

15. (Idea from MC) Create a simple app where you can direct massive traffic to any site you choose.. Be careful though and don’t attempt this at the Dojo... If the site selected doesn’t have your javascript installed, then only send 10 traffic. If the site has your javascript (with some authorizations that allows you to identify who installed it), then you can have the user identify how many http requests to send. Then, display a graph that shows the number of requests that came back within x seconds and how long these requests took. Display some interesting graphs to illustrate how you are able to build simple tools to test how scalable a site is. Be creative and think of other ways to not only learn how to scale things but also how to generate other cool graphs that you can showcase.
16. (Idea from MC) A lot of organization have hard time creating beautiful organization charts. Create a simple app where users can put the company’s organization information via forms and where your app generates beautiful PDF with the organization chart (maybe for up to 3 levels deep). You could alternatively create a node library where you put the organization information and where it sends you a json data back with a link where you can download the PDF. On the dashboard, you can show all the different PDF generated through your library/app. For example, you could have this be simply be: var json_info = YourAppName({ organization_name: “_____”, positions: [____], names: [____]); or something like this to send all the information you need to generate the PDF simply by passing all of this information as a javascript object through your library.
17. (Idea from MC) Show a messy room with multiple objects. Have a timer that shows how long it takes for each user to clean the room (e.g. cleaning defined as moving the object to the trash, closet, drawers, or bookshelves). Have a 2nd player who can move the objects to other places. Record how these cleaning activities occur (bonus point if you can show the video feed of both people and what they’re saying transcribed as text). Record these cleaning sessions and post metrics on the dashboard that shows who is the best distractor, cleaner, etc. Also allow people to see past battles. Allow people to see current battles! If you’re a good artists, maybe you can even draw some of these objects yourself!
18. (DO) Build a programmable Voice, Video or SMS text app using Twilio.
19. (DO) Create any multiplayer game using sockets.
20. (DO) Use D3.js or any other data viz JS lib and create a dashboard that you can connect to a GPS device like, your own homemade sensor payload using a Raspberry Pi or some such other.