Meet Up Outline Notes – These are the things I will say

Have power point in one window and codepen, google tab, cat image in another window

**Slide 1: Intro**

* About me
  + A year ago I had no idea what coding is
  + No previous experience at all in any computer science, coding, engineering
  + Learn graduate
  + I want to make the transition better for other people like me
  + Tech is a powerful industry, people creating websites are dictating the way we live our lives, the industries that succeed, the jobs that are created
  + People coming from diverse backgrounds helps the tech community be better, technology is for everyone, to be used by the world, so the people creating the technology need to be just as diverse as the people using it
* About Learn
  + There are tons of online resources that are great but nothing compares to a structured learning environment with amazing instructors and supportive staff
  + Focus on the tech community, personal development as well as professional
  + Career services, community events, meetups
  + Diversity – age ranges, backgrounds, pro-women
  + 3 months of classroom time designed to be as close to a work-like setting as possible
  + Internship is the difference maker from other bootcamps
* Overview of the next two hours
  + I’m going to outline some basic info about programming in general and some basic concepts
  + We’ll go over the three main languages that are used in combination to make a useful website
  + Open Chrome

**Slide 2: What are we doing here anyway**

* Lots of people don’t really know what coding is
  + medical coding
  + 1 and 0
  + Writing languages
  + IT
* Programming or coding
* Creating software – you hear programmers referred to as developers or devs for short, or as software engineers
* Programming is learning a language, or multiple languages and learning how they work and what we can do with them

**Slide 3: Why learn programming**

* What drives you to learn this new skill?
* This list is partially mine and some from my other members in my cohort at Learn
* Challenging - never ending amount of things to learn, which can feel really overwhelming but is also really exciting
* Problem solving - makes you examine your everyday life
* Trade - if you can make a website, you have a skill a lot of people don’t have and a skill a lot of people need, if you have a business you need a website
* So many options for jobs and industries, learning to code doesn’t mean you will sit in the basement
* Organizational – coding requires attention to detail and ability to visualize the big picture
* Universal language – language of logic
* Everyday tech - almost everyone uses websites or webapps everyday but very few have any idea how they work
* Opens opportunities – business, every hear someone say, I have a great idea for an app. Yeah, then what?
* Creating visually appealing interfaces, but coding itself can be an artform
* Jobs are available everywhere, lots of jobs are based out of an office or work space, but one of the amazing parts of programming is it can be done remotely

**Slide 4: Programming languages**

* While we are learning to write code, I like to compare it to learning a new language
* Each of these languages just like any other language have
  + Particular vocabulary words – key words built into each language
  + Sentence structure – just like stringing together words into sentences and sentences into paragraphs, the order matters and meaning (and in code – functionality) can be lost if not done properly
  + Punctuation – all the weird symbols programming languages love, some languages (like Ruby) use less symbols, but they are important to every language
  + Grammar – in language we call the structural rules grammar, in programming we call it syntax – the exact formatting that the computer expected
    - Like any language, programming languages have evolved and grown and have weird quirks that are “just because”
    - The same way cough doesn’t rhyme with though
* Convention – the unwritten rules of programming that help humans read and interpret code
  + Descriptive naming of variables, commenting your code
* Best practice – if you are building a house you should get good blueprints, use solid materials, even if it is more work at the time
* Computers are only as smart at the humans telling them what to do, they aren’t super good at interpreting your meaning or getting the gist of it
  + Make a sandwich example – write instructions with every detail of the process

**Slide 5: Programming languages**

* There is two main parts of a website – the stuff you see and the stuff you don’t
* Frontend – also called the client side
* UX/UI – they are often grouped together but can be very separate entities
  + People with art or graphic design backgrounds are often drawn to UI
    - UI is the eye candy of the website, making the site visually appealing
  + UX is about making the website user friendly, accessible to all people and all devises
    - My parents with the Lyft app
    - Walking him through the app and looking at it from his eyes, I realize there are a few more steps than are intuitive
    - It’s a classic situation of “user error” but if a person’s interaction with a website is frustrating or just outright unsuccessful, your business is going to suffer
  + HTML/CSS/JS – the three musketeers of frontend programming
* Backend – also called server side logic
  + Ruby on rails, Python, JS
    - JS is a full stack language – that can run frontend and backend
  + Making changes to the website like uploading a photo or creating a user
  + What is happening behind the scenes

**Slide 6: How the internet works**

* Before we get into the specific languages I want to cover a bit of how the internet works
* Just a very high level overview
* HTTP – hypertext transfer protocol (s – secure, means all the communications between your browser and the website are encrypted) it’s a certification that is rapidly becoming standard
* When a user clicks on a URL it triggers an http request made up of two parts:
  + An IP(internet protocol) address of a particular server where the information is stored
  + And a CRUD action – create, read, update, delete, the most basic functionality a website
* Each page has multiple pieces or assets that come together to create the page
* The server is responds with data displayed in the browser
* Every request gets a response, even if that response is an error
  + One thing you quickly learn about programming is errors codes are some of the best, most valuable information you can get
* The CRUD actions and the response codes come into play as you learn to build full stacked applications
* Now lets get to the action

**Slide 7: HTML - about**

* Hyper Text Markup Language, written in 1989/1990, so it’s pretty old as far as the world of the internet is concerned
* We’ve gone through several iterations and are currently using HTML – not bad really
* In the early days of the internet there was no standardized way to send information/documents and people using different machines and different platforms couldn’t always access the same information
* Original use for HTML was in academics, so that published work to be saved on a server and read online instead of moving physical paper all across the country
* Once people figured out it was pretty easy to use, HTML became popular outside of academic worlds
* The idea that a person could use a word processor and convert the text to a web document, that could be viewed simple and straightforward with a handful of tags to instruct the computer on how data is to be displayed
  + Tags are the real thing, they are pieces of code instruction for the computer
  + MySpace page changes were done in HTML
* A user could click a link – like in the table of contents, that would then take them to another part of the page
* HTML is a data display language, it’s the skeleton of the page, the nouns
  + Tags and <> open and close, except for when they don’t
  + Semantic tags describe what the tags do

**Slide 8: Coding in HTML**

* Normally when you write code you use a text editor
  + Atom, Sublime text, Vim, and a whole bunch others
  + We use Atom at Learn because it is free and open-sourced, can be used on Mac or Windows machines, user friendly, supports all the languages, developed by github
  + You will need to declare a doctype in HTML
  + Save a new file with the extension .html
    - type html and hit tab and the doctype info will auto populate
  + All the code that appears on the page is written inside the body tags

**Slide 9: Coding in HTML**

* Header tags – options of 1-6
  + A document should only have one H1 for SEO purposes
  + All others can be used as much as needed
* List tags
  + A container for code that will be in a list form
  + Ordered lists have numbers
  + Unordered list have bullets
  + A list item is a nested within the UL/OL tags
* Image tags – the URL lives inside the tag because it is part of the code
  + Add attributes to modify the tag

**Slide 10: CSS**

* Cascading Style Sheets, the file works top to bottom so any conflicting information lower on the page will override previous information
* Makes the page pretty, the skin or the “adjectives” of the page
* Must have HTML to have CSS
* Show the codepen button without CSS
* Like the height of the picture, HTML elements can be modified with inline styling
  + Drawbacks are the tag becomes way too cluttered
  + When styling multiple elements the code is not DRY
  + We want the pages to be separate “called separation of concerns” – best practices
  + So instead CSS lives on separate pages and the style sheet is linked to the HTML with a link tag in the head of the HTML tags in the head metadata
  + Instead of inline styling, we id HTML elements and modify them in a CSS file
  + Element selector – modifies every element of that type

**Slide 11: Coding in CSS - removed**

* Unlike HTML there is no need to declare a doctype because you must have HTML to have CSS and the doctype is already set
* Only need to save the file with the extension .css
* Syntax is the tag followed by curly braces containing the {property: value;}
* Puedo-selectors – a special state of the browser, not fully built into the page, mutable, a popular one is hover – it doesn’t change the way the browser is rendering the page

**Slide 11: JS - about**

* First appeared in 1995, code name of Mocha, Brendan Eich wrote in 10 days
  + Not to be confused with Java, but sorta meant to be confused with Java
* The dynamic and logic part of the page, the part that does stuff
* The muscles or the “verbs” of the page
* Object-oriented programming language – groups of data types that interact with each other
* camelCase – convention for JS naming

**Slide 12: JS Basics – Data Types**

* Data types – in order for the computer to manipulate information we must identify what type of data it is
  + 6 primary types
  + Numbers
    - Integers – whole numbers, fractions/decimals are called floats
    - Can do math, follows the order of operations
  + Strings – single vs. double quotes, using a ‘ inside “”, escape character / to use quotes or other symbols in a string, concatenation, length property
  + Boolean – true or false values, words but not strings
  + Undefined
  + Null – not 0, equates to nothing else
  + Symbols – new data type in the most recent update of JS

**Slide 13: JS Operators**

* What does JS do?
* Math
  + Add, subtract, multiply, divide
  + Modulus, gives the remainder of a division problem
  + Exponents \*\*
* Cocatination – the + with two strings
  + If you want spaces you must include them
* Comparison operators
  + = assigns (save for later)
  + == type coercion, do the two things equate to the same value?
  + === equates exactly, including the same data type, the preferable usage best practice
  + Booleans – && both must be true, || – at least one must be true
* Negate - ! bang operator
* Try these operators in console

**Slide 14: Dev Tools**

* Take a little pause in JS and talk about dev tools (developer tools)
* Inspect element, helpful for debugging and you can see the information behind any page, live changes to see the effect of the code on the browser, not actually changing the code
* Chrome JS console – built in dev tools, interacting with JS using the browser, functionality without making an HTML file, a JS file and linking them together in order to do basic JS
* View page source – great way to learn about coding, find an interesting page and look at the source code
  + Notice it is not terribly readable
  + The computer doesn’t care about white space between lines
  + The reason we follow layout conventions is to make the code readable for humans
  + If we follow best practices it is easier follow the logic of code you didn’t write – which as a developer will be most of it and spot errors in your code
* In Chrome, change the top margin, background color, refresh page

**Slide 15: JS Basics – Variables**

* Variables – just like in math a variable is a placeholder or a container for storing other information
* Keywords are built in reserved words used by the JS language
  + Function, let, if, while, for, return, catch
  + Can image why we wouldn’t want to name a variable “var”
* Variables use a = to indicate assignment
* Can use another variable to save the results of combining other variables

**Slide 16: JS Functions**

* Manipulating JavaScript logic and writing our own pieces of code
* Function is a keyword in JS and the naming convention is camelCase

**Slide 17: JS Functions**

* The way functions can be dynamic and therefore reusable is to pass in arguments
* Running a function without an argument will have the same outcome every time
* Arguments change the outcome of the function

**Slide 18: JS Methods, string properties**

* Properties – not changing the string, just returning information
* Methods – this is where JS actions start to take place
* Methods store prewritten code actions
  + Methods change the state of the user input
  + The () is the syntax that signals the action to occur
* Make two variables
  + toUpperCase, toLowerCase, concat
  + repeat – add number as the argument
  + endsWith – returns a Boolean value
  + replace – enter the string to be replaced, the replacement

**Slide 19: JS Basics - window alerts**

* Alert(), prompt()
* Var username = prompt(“What is your name?”)
* Do string methods to the saved input