## Javascript Basics – Mosh on Udemy.

Section 1

2-What is JavaScript

* Originally built for client-side processing in a browser. In 2007 Ryan Dahl encapsulated Chrome’s V8 Open-Source JS engine within C++ to create Node. This allows JS to run outside a browser.

3-Setting up the Development

* Installed Live Server. This allows you to see changes in browser (Web page & console) dynamically as you make them. Start it then copy url to google chrome. Right click to “inspect” page to bring up developer tools. This course uses console quite a bit. In top right of developer tools, select view that disconnects the panel so it can be placed side by side w/VS Code like Mosh has on his screen.
* Errors showing up on Console;
  1. First, it could not find favicon.ico. I copied a jpg file from web and renamed to favicon.ico. Placed in same folder as html file. That fixed the error.
  2. Second error was due to Last Pass extension. Google throws error on winID being null.
  3. turned extension off and now all is working!!
  4. On new Dell, the McAfee extension was causing effort. Turned off and now working.
* Installed Node on 5520. Run .js file from command prompt >node 2-basics.js
  1. Can do New Terminal in VS Code to create a terminal in VScode. Then run code right in VS Code.

## Shortcuts

* To comment out html code or JavaScript code, select it then press crtl /.
* Save using crtl s.
* To create a default html file, on first line enter ! followed by tab.
* To clear console: ctrl l (lower case L). To rerun js file just save it again.
* To move line press ALT and up or down arrow.
* Duplicate line – Shift Alt down arrow.

## Best Practices

* Define variables using let not var. var was used before ES6, but there are issues w/this.
  + ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.
* In HTML file, add script section at end of body.
* Use single quotes for strings. More popular the double quotes. Either will work.
* Mosh likes to use if..else rather then switch..case. He thinks if..else is more streamline.
* Use for..of loop to iterate over arrays. This became available in ES6.
* Use for..in loop to iterate over the properties of an object.
* The continue key word jumps you to next iteration. This is old legacy JavaScript and do not recommend using.
* Two methods for creating objects. Mosh says both are equally good, but constructor approach more similar to Java, C#, etc.
  + Factory Function – use camel case naming. Camel case - createOneTwoThree
  + Constructor Function – use pascal naming. Pascal case - CreateOneTwoThree

## Callback functions in Methods

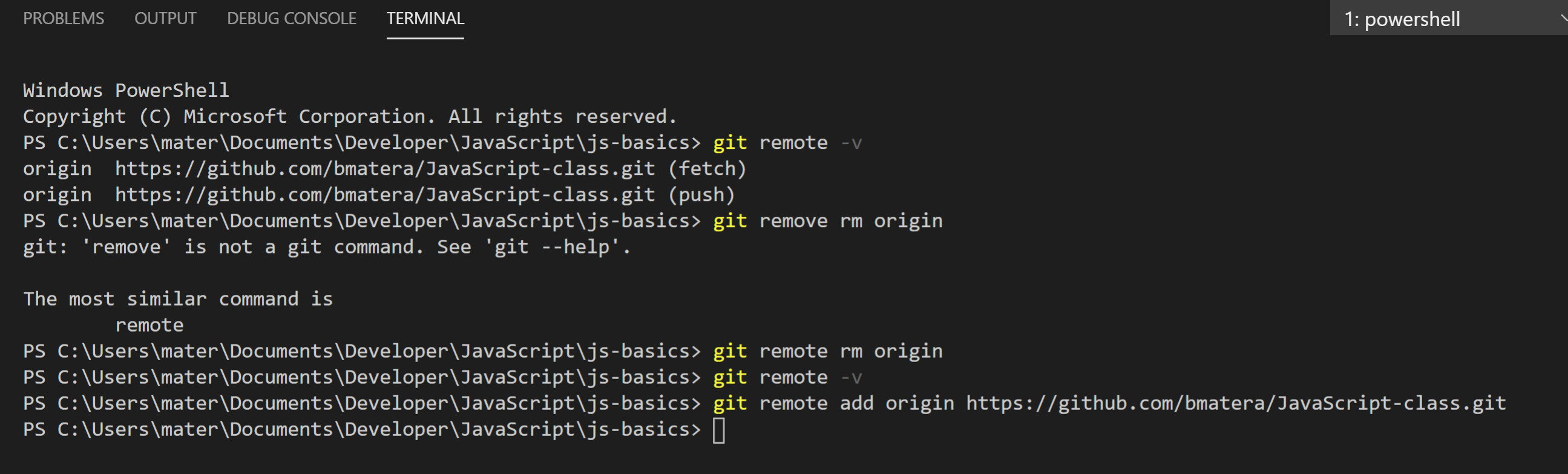
Common methods discussed that have a callback function, typically to process an object’s elements. The callback function is called back for each element in the array.

* Array of objects: courses.find() - find a course object.
* Iterate through an array: numbers.forEach() - iterate and return array element and index (if needed).
* Sort an array: courses2.sort(function(a, b))
* Test elements: nums6.every() and nums6.some() (e.g. positive or negative numbers)
* Filter array elements: nums6.filter() (returns an array, eg. Only positive numbers)
* Map array elements: filtered.map() (map array element to primitive or object).
* Reduce array elements: nums7.reduce() (add all elements of an array).

Look these up on google: javascript reduce. The Mozilla references are detailed w/examples.

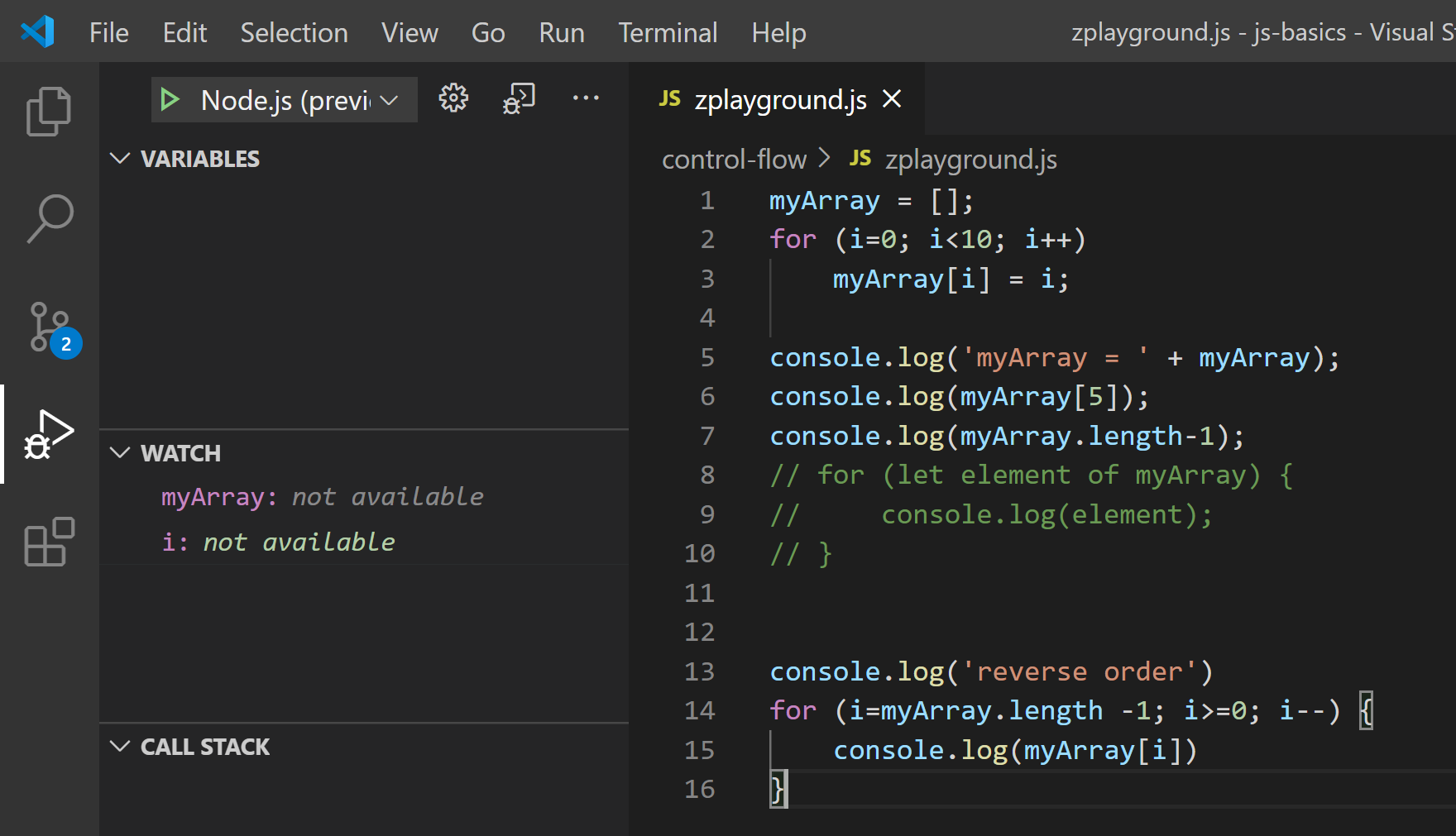
Good reference for filter, map, reduce: <https://www.freecodecamp.org/news/javascript-map-reduce-and-filter-explained-with-examples/>

## Git

* Installed Git on my PC.
* Video to get started <https://www.youtube.com/watch?v=6n1G45kpU2o>
* Video on push git to github <https://www.youtube.com/watch?v=I7WfxhF2wEg>
* Within VS Code I selected Source Control and then create repository. Selected JavaScript folder, Staged All, then committed. It then created a .git folder in my JavaScript folder. This is a hidden folder, so need to be sure they are showing or >ls -hidden.
* After making changes to these notes, VS Code immediately showed that file has changed!
* On bottom left it shows Git master is checked out.
* Remember – stage and commit to git, then sync changes to github (bottom left). You can do all this from Source Control menu in VS Code.
* After installing Git on my new Dell XPS, I restarted visual code and it automatically found Git and the existing .git repository that was in the folder.
* I staged new changes but got error on commit. Needed to do the following
  + git config --global user.email "you@example.com"
  + git config --global user.name "Your Name"
  + to set your account's default identity.
  + Omit --global to set the identity only in this repository.
* For Notes.docx I set it to include in .gitignore.
* Syncing Github with new Dell XPS in VS Code.
  + It would have worked straight away if I would have just logged into Github when asked. However, I did not do that but instead deleted repo on Github to try and start over. Once I did that my Git account on local machine needed to remove the repo.
  + Captured terminal below. I first listed remote url using: git remote -v.
  + Then removed the remote (origin): git remote rm origin
  + Then added new origin from Github, which is taken from Github when create new repo.
  + 

## Debug in VS Code

* Open your js file to debug. Then select the Run Debugger on left (red). To run debugger, need to select green circled play button. This is where is gets tricky as there are options when you select drop-down. You want to select “Node.js (preview)” but there are options for this as well. To be able to select those options you need to first select a different option (eg. Attach to Chrome), do not run, now select the “Node.js (preview)” option and you will see two Launch Configurations to select from: Create JavaScript Debug Terminal, Run Current File. Select Run Current File, then press Play button.
* This will run Debugger and open a terminal window and set it to the Debug Console.
* If you did not set a Break Point it will run to completion. Probably not what you want.
* Instead set a Break Point then hit play button again and it will bring up the second screen below. If the Terminal window is not showing go to View ~ Terminal.
* Step Over button executes next line of code. The Step into seems to take you into the Javascript libraries but you can easily exit with Step Out.
* If the debugger runs to end of code it will jump into JavaScript libraries. Need to set a breakpoint before to capture what is going on in code. Keep in mind to set break points within functions otherwise that code may be executed unknowingly.



Red bullet is break point, yellow triangular arrow is current line in execution.

As you step through the code, output is shown in Terminal ~ Debug Console. You can add things to watch in the Watch window.

