**Functional Programming Lecture**

poll everywhere

slides

get lecture repo

terminal to the lecture directory

live-server

bring up the developer tools in chrome, console

---------------------------------

**Slide 2**

**PLAN FOR TODAY**

Feedback from you guys on the course

Functional programming lecture

Worktime

----------------------------

**Slide 3**

**THE COURSE SO FAR**

What we've done so far

We’re about 1/3rd of the way through the class…

--------------------------

**Slide 4**

**Poll everywhere**

-------------------------

**SLIDE 5**

**QUESTIONS**

Issues from other class session:

- Will we be learning how to use an API to get real time info for our website? Yes

- How much will we be using jQuery? We'll discuss it, but it's not really necessary anymore

We’ll use it some and talk about it because it’s important to know. But you don’t really need to do it anymore

- Can you discuss efficiency of loops?

There are some things you do to optimize efficiency, but we don’t really care about making it the most efficient.

We don't care as much about optimizing for efficiency in this class when weighed against readability and maintainability. Saving a nano-second isn't worth it.

We want others to be able to understand our code.

The most efficient for loop is the classic for (let I = 0; I < len…)

You could “unfold” loop where you increase by more than 1, and do more stuff in each iteration… but not worth it here for sure.

-----------------------------

Slide 5

Do you recognize and understand how we declare functions and call them?

-----------------------------

**SLIDE 6**

**Functions**

Recall this is how we declare functions...

**Slide 7**

**Functions are values**

A function is like any other value you declare.

They are “things”, objects. They are “recipes” like a recipe card that you can use and store and move around in different ways.

------------------------------------

**Slide 8**

**FUNCTIONS ARE VALUES**

It’s like a string, a number, an array.

The most similar data type is an “array”. See the code

You can use a function anywhere you can use an array

For arrays the '[]' is the super power

Functions work the same way

For functions the parenthesis '()' is the super power

Example - we create an array and can assign that array to another variable

Side by side with an array, and its essentially the same process but with a function.

-------------------------------------

**Slide 9**

**Anonymous Variables**

We can make functions anonymous.

Here’s an example of an anonymous array. (CODE EXAMPLE IN VISUAL STUDIO)

Passing a literal.

If we don't include a name, it's an anonymous function.

Example of an array first being passed by variable, then an anonymous (literal) array being passed to console.log

Underneath the same example with first a function "sayHello()" being created anonymously, then showing how an anonymous function can be assigned.

The first example is not really valid. There's no way to access that function because there's no name. It'd be the same as writing

[1,2,3]; as a line of code.

There is one difference when declaring functions vs assigning an anonymous function to a variable.

Hoisting.

Declared functions get hoisted before code progresses. Assigned functions don't until they are assigned. So you must assign before calling

otherwise the runtime environment won't know what the function is and will error out.

Use the example from the lecture notes

It's possible to call a named function even if it's declared later in the file because of hoisting. You can't do that with anonymous functions that are then assigned

Best practice use named functions when you can.

We're going to use Anonymous Functions a lot

-----------------------------------------

**Slide 10**

**FUNCTIONS ARE VALUES**

Functions can be assigned to properties in Objects

If you apply static keyword with any method, it is known as static method. A static method belongs to the class rather than object of a class.

A static method invoked without the need for creating an instance of a class. static method can access static data member and can change the

value of it.

non-static here means, functions that belong to objects

------------------------------------------

**Slide 11**

**Functions are Objects**

Which means they can be passed as parameters to other functions

Side by side - on left shows passing in an array

on right shows passing a function

Things of note:

- between the '()' note that we still don't declare that the parameter is an array or function in JavaScript. the code within the function

determines the type

- the more common way to do this is to pass an anonymous function in JavaScript.

This is how we end up with the });

the '}' is the closing bracket to the anonymous function. the ')' is the parenthesis for the calling function. then the ;

We'll be using this in a number of places...

- when looping, we pass the function we want to be run through each iteration\

- when doing interactive programing, where we have listeners and we register a callback function that will run when a button is pushed for example

---------------------------------------

**Slide 12**

**Passing vs calling**

Don't put the '()' when passing function as a parameters

This will instead pass the value of the function rather than the function. You'll get an error if the function expects a function.

You want to pass the recipe, not the cake. So don't put parens when its expecting a function

-------------------------------------

**Slide 13**

**Callback Functions**

We call these things "callback" functions.

Walk through the examples. Maybe show how you can have the two functions passed in as anonymous functions

remember to delete the function name when you paste in the anonymous code

So where would we use this idea of a callback?

In array iteration

-------------------------------------

**Slide 14**

**Array Iteration**

array.forEach(callback)

More common to use an anonymous callback in the ForEach()

On es6 you can use:

for (item of array){

console.log(item);

}

this doesn't work as far back for old browsers while forEach does

of course that gets fixed in react where things get transpiled

But the recommendation here is to use forEach()

--------------------------------------

**Slide 15**

**Mapping**

First slide shows how to do it manually

The second slide shows how the map function does it in one line of code

Another cool one is "mapping"\

It takes in one array and we will transform each of those elements in some way

Each element gets transformed in some way

returns new array

It's super useful

It's often more common to do this using an anonymous function as the parameter rather than passing in the named \function

--------------------------------------

**Slide 16**

**Filtering**

Companion to map

returns new array

What elements do I want to keep?

The callback function returns a Boolean (true or false). If it returns true the element is kept, false the item is tossed

(like a bouncer)

----------------------------------

**Slide 17**

**Reducing**

Aggregation function

Take a bunch of elements and smash it down (sum, medium, average, etc.) and return that value.