**Imperative Code**

A lot of JavaScript is **imperative code**. If you don't know what "imperative" means here, then you might be scratching your head a bit. [According to the dictionary](http://www.dictionary.com/browse/imperative), "imperative" means:

*expressing a command; commanding*

When JavaScript code is written *imperatively*, we tell JavaScript exactly **what** to do and **how** to do it. Think of it as if we're giving JavaScript *commands* on exactly what steps it should take. For example, I give you the humble for loop:

**const** people = ['Amanda', 'Farrin', 'Geoff', 'Karen', 'Richard', 'Tyler']

**const** excitedPeople = []

**for** (**let** i = 0; i < people.length; i++) {

excitedPeople[i] = people[i] + '!'

}

If you've worked with JavaScript any length of time, then this should be pretty straightforward. We're looping through each item in the people array, adding an exclamation mark to their name, and storing the new string in the excitedPeople array. Pretty simple, right?

This is *imperative* code, though. We're commanding JavaScript what to do at every single step. We have to give it commands to:

* set an initial value for the iterator - (let i = 0)
* tell the for loop when it needs to stop - (i < people.length)
* get the person at the current position and add an exclamation mark - (people[i] + '!')
* store the data in the ith position in the other array - (excitedPeople[i])
* increment the i variable by one - (i++)

Remember the example of keeping the air temperature at 71º? In my old car, I would turn the knob to get the cold air flowing. But if it got too cold, then I'd turn the knob up higher. Eventually, it would get too warm, and I'd have to turn the knob down a bit, again. I'd have to manage the temperature myself with every little change. Doesn't this sound like an imperative situation to you? I have to manually do multiple steps. It's not ideal, so let's improve things!

**Declarative Code**

In contrast to imperative code, we've got **declarative code**. With declarative code, we don't code up all of the steps to get us to the end result. Instead, we *declare* what we want done, and JavaScript will take care of doing it. This explanation is a bit abstract, so let's look at an example. Let's take the imperative for loop code we were just looking at and refactor it to be more declarative.

With the imperative code we were performing all of the steps to get to the end result. What \_is\_ the end result that we actually want, though? Well, our starting point was just an array of names:

**const** people = ['Amanda', 'Farrin', 'Geoff', 'Karen', 'Richard', 'Tyler']

The end goal that we want is an array of the same names but where each name ends with an exclamation mark:

["Amanda!", "Farrin!", "Geoff!", "Karen!", "Richard!", "Tyler!"]

To get us from the starting point to the end, we'll just use JavaScript's [.map()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map?v=example) function to declare what we want done.

**const** excitedPeople = people.map(name => name + '!')

That's it! Notice that with this code we haven't:

* created an iterator object
* told the code when it should stop running
* used the iterator to access a specific item in the people array
* stored each new string in the excitedPeople array

...all of those steps are taken care of by JavaScript's .map() Array method.

**QUIZ QUESTION**

Is the following code imperative or declarative?

**const** people = ['Amanda', 'Farrin', 'Geoff', 'Karen', 'Richard', 'Tyler']

**const** longNames = people.filter(name => name.length > 6)

* 

Imperative

* Declarative

SUBMIT

***.map() and .filter()***

*A bit rusty on JavaScript's .map() and .filter() Array methods? Or perhaps they're brand new to you. In either case, we'll be diving into them in the React is "just JavaScript" section. Hold tight!*

**React is Declarative**

We'll get to writing React code very soon, but let's take another glimpse at it to show how it's declarative.

<button onClick={activateTeleporter}>Activate Teleporter</button>

It might seem odd, but this is valid React code and should be pretty easy to understand. Notice that there's just an onClick attribute on the button...we aren't using .addEventListener() to set up event handling with all of the steps involved to set it up. Instead, we're just declaring that we want the activateTeleporter function to run when the button is clicked.

**Declarative Code Recap**

*Imperative* code instructs JavaScript on *how* it should perform each step. With *declarative* code, we tell JavaScript *what* we want to be done, and let JavaScript take care of performing the steps.

React is declarative because we write the code that we *want*, and React is in charge of taking our declared code and performing all of the JavaScript/DOM steps to get us to our desired result.

**Further Research**

* Tyler's [Imperative vs Declarative Programming](https://tylermcginnis.com/imperative-vs-declarative-programming/) blog post
* [Difference between declarative and imperative in React.js?](https://stackoverflow.com/questions/33655534/difference-between-declarative-and-imperative-in-react-js) from StackOverflow