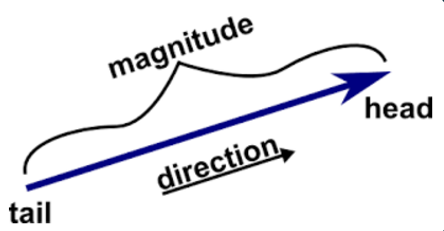
**Vectors Project Guide**



### **0. Guidance**

#### **Learning Javascript**

The basic features of the Javascript language are relatively simple and sufficient for assignments in this course. To get a feel for the syntax, see [learnxiny](https://learnxinyminutes.com/docs/javascript/). You may also find [Javascript for People Who Know Python](https://www.youtube.com/watch?v=GAoheEUiwwY) useful.

#### **Debugging**

I highly recommend spending a little time gaining proficiency with your tools (i.e. Chrome or Firefox). Investment in debugging skills pays huge dividends… Make sure you

1. **Understand how to set breakpoints in your code**
2. **Step through code execution line by line, and**
3. **See the values of your variables at any point in time** (without “print” statements).

See [**Debugging in chrome**](https://developers.google.com/web/tools/chrome-devtools/javascript/).

#### **Getting Help**

Should you find yourself spending an excessive amount of time without progress, please utilize the following:

1. Your partner
2. The discussion boards
3. Email ([micguerrero@csumb.edu](mailto:micguerrero@csumb.edu) or the TA).

#### **Clarifying / Improving this Guide**

If something printed here is unclear to you or seemingly wrong, please leave a comment on the side of page using the commenting feature of Google Docs. It should be as simple has highlighting something, right clicking, and selecting comment. You can see a demonstration of this feature [**here**](https://www.youtube.com/watch?v=RC6LyeigAFs).

### **1. Overview**

In this project you will be implementing the API for a Vector3 object. The function names and signatures are already defined but it is up to you to provide the implementation.

### **2. Todos**

In this course, programming project guides (like this) will come with a set of *to-dos*. These are the tasks that you are responsible for.

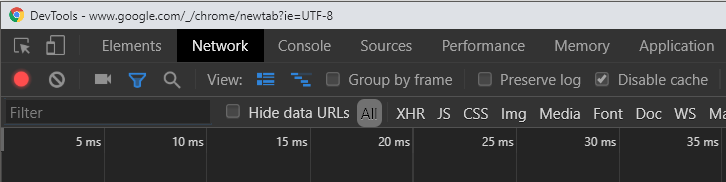
### **3. Common Mistakes**

As you proceed through this assignment, take note of the following commonly made mistakes:

* Not using the “var” keyword before declaring a variable. Javascript will allow you to get away with many things that other languages won’t. In this case, the language will treat your variable as a global which could be very confusing if not intended. For example, declaring variable foo in a function will make foo available everywhere and any change made to it will be permanent.
* Browser caching previous code. **This will happen to you!**

Sometimes you will make changes to your code and the web page will not reflect those changes on refresh due to caching. Here are a couple ways to get around this.

* Hold shift and click the refresh button (chrome, firefox?)
* Make sure the dev tools are open and “disable cache” is selected



* Other Javascript-isms. See [here](https://docs.google.com/document/d/1Ql7-U9bKUf94Wl26uXbGij0MfYvwUy_GxS9_3kWmTvw/edit?usp=sharing).

### **4. Project Files**

You should begin this project with the following files:

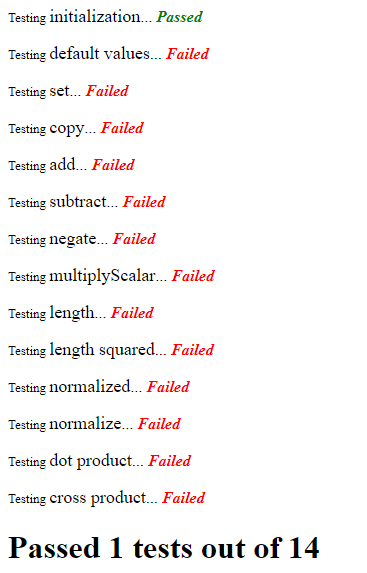
**math/vector3.js**

**test/test.js**

**test/test-vector3.html**

### **5. Setup**

To start, run test-vector3.html and you should see the following:



This is the result of automated testing of your code. Initialization is the only test that currently passes and it does so by making sure that the variables passed into the constructor of Vector3 are applied to variables stored as members.

|  |
| --- |
| var Vector3 = function(x, y, z) {  this.x = x; this.y = y; this.z = z; |

### **6. Implementation**

More detailed information about how to code these will be found inside of the vector3.js file itself. For more of a conceptual understanding of the code and its conventions, read the todos below.

#### **Todo #1 Default Values**

Take a look at the code for *testDefaultValues()* in *test-vector3.html*. It creates a Vector3 without passing any parameters. Make sure you handle this case by setting x, y, or z to 0.

#### **Todo #2 Set**

With this Vector3 object, when you want to overwrite the value of our objects (x, y, or z), you will do so with a “set” function. This will be the convention for functions that start with “set” in the future.

#### **Todo #3 Clone**

One of the most common mistakes made when using this code in later projects is overwriting values that you were trying to save for later. This happens because we have multiple variables referring to the same object (not what we wanted). To avoid this, you can make a new **instance** which is a copy of an existing variable but stored completely separately in memory. This is known as cloning.

#### **Todo #4 Copy**

Unlike clone, this does not make a new instance. It simply copies the values from one existing object to another existing object (the one you called “copy” on).

#### **Todo #5 Everything Else**

Each function has information inside detailing what should happen. Inspect the individual test code functions to get a better idea if you are unsure.

### **6. Grading**

You will be given credit for each function that passes its test for a total of 50 points.