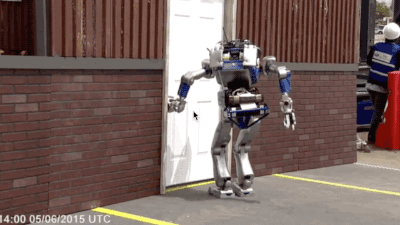
**Module #6 - Sort of Code with Pseudocode (Required)**

In this module, you'll be undertaking an introductory dive into the world of programming logic.

Be warned! This module is one of the most important in the Pre-Work Curriculum, and it is also one of the most challenging. Push yourself to put in the time and effort necessary to complete it fully.

**Why is Programming So Hard?**

One of the biggest reasons that programming is such a challenging profession is because fundamentally, computers are very, *very* dumb. Much of the complex logic and layered thinking that we humans take for granted would be completely bewildering to even the most complex of computers.



Along this vein of thought, the art of *coding* and *programming* is principally focused on taking complex ideas and on breaking them down into simple instructions that a computer, a machine, or a browser can interpret. Learning to code, thus requires proficiency in the two components necessary to translate ideas into *machine-language*.

1. The first is that every coder must learn the *syntax* that a computer understands. Syntax is sort of like the grammar and punctuation of machine-speak. It's like knowing that every sentence has a subject and a predicate or like knowing that every sentence begins with a capital letter and ends with a period. In the same way, programming languages come with their own *rule-sets*. These rule-sets may specify a special meaning for brackets or for symbols—or for a particular method for utilizing code across different files. Throughout the course, you'll be learning a *ton* of syntax. It will seem tricky at first, but over time, it will feel like second nature. In truth, syntax will quickly become the least of your worries.
2. Instead, your real challenge will be associated with the second proficiency: converting your ideas into *computerthink*. You see, unlike people, computers are limited in their ability to work with abstract, with vague, or with general ideas. Instead, complex problems need to be broken down into small, discrete blocks of logic.

**Think Like a Computer**

Take for instance the following challenge:

*How do you make a peanut butter and jelly sandwich?*



Perhaps you might make a list that looks like the following:

1. *Get bread, peanut butter, and jelly from fridge.*
2. *Get utensils and a plate.*
3. *Slather peanut butter and jelly.*
4. *Combine smeared breads.*

This is all fine and dandy... but really, there's more to it than this.

Take Item #3, for instance: *Slather peanut butter and jelly*. A human might understand this instruction, but a computer or a machine would also need to be told the following:

* *How many times do you slather?*
* *What direction do you slather?*
* *What amount do you slather?*
* *What utensil do you use to slather?*
* *Which do you slather first?*

As you will find out, learning to code often requires one to think far more slowly than we are often accustomed. Instead of rushing through minor details, your computer will force you to slow down and to reason out your thoughts in a sometimes nauseating number of steps.

**But What About the Future?**

Unfortunately, until a breakthrough in artificial intelligence emerges, you and I are stuck with our dumb computers. This is good news for us as developers, however, because it guarantees that the skills we possess will be in demand for a good time to come.

In the remainder of this module, you will be shown a video exposing you to the logical building blocks used in almost every programming language (definitely watch this!). You will then take this newfound understanding to create a simple game using the visual programming language Scratch. Don't let the cartoony interface and silly cat logo deceive you. You can build some powerful things with Scratch, and this activity will throw you for a doozy.

**Video (Required)**

* [Introduction to Programming Logic](https://youtu.be/GNyvRP6HzhA)

**Assignment (Required):**

* [Scratch That!](https://the-coding-bootcamp.gitbooks.io/pre-work-book/content/assignment6.html)

**Additional Reading:**

* [How to Tackle Something You Have No Idea How to Do](https://www.themuse.com/advice/how-to-tackle-something-you-have-no-idea-how-to-do)

**Supplemental Resources:**

* [Getting Started with Scratch](http://d.umn.edu/%7Esivelab/project/learningToCode/Getting-Started-Guide-Scratch2.pdf)
* [Scratch Help Guide](https://scratch.mit.edu/help/)

**Copyright**

Coding Boot Camp © 2016. All Rights Reserved.

**Assignment #6: Scratch That!**

**Overview:**

In this assignment, you'll be channeling your inner child and creating a simple game using [Scratch](https://scratch.mit.edu/), a free and simplified programming language.

Your goal is to build a game that functions like any of the following examples:

* [A Game of Cat and Mouse (Challenge)](https://www.youtube.com/watch?v=z4iD-559NJ4)
* [Get the Cheesy Puffs! (Recommended)](https://youtu.be/ljQX6YG8eVA)
* [Button Click Exercise (Too easy...)](https://youtu.be/5BymTpAKsi4)

While you have three options, we *strongly* encourage you to complete the *Recommended* or *Challenge* activities.

And yes... we know you haven't been taught Scratch. But if fifth graders can learn it on their own, then so can you. This is your first *real* challenge as a developer. Take the time to independently learn what it takes to build these applications. The logic that underlies these games is *very* similar to the logic you'll be dealing with in Javascript so treat this as a serious endeavor. As a baseline, you can expect the recommended or the challenge assignment to take up to 15 hours of time if you are completely new to programming.

**Before You Begin:**

Go to <https://scratch.mit.edu/> and create a new account. Then take a few minutes to watch the provided tutorials and to browse a few of the projects others have built. As you will quickly discover, you can build some amazing things with Scratch.

**Instructions:**

1. Watch the demos again, and choose an assignment. Again, we *strongly* suggest you complete the *Recommended* or *Challenge* activities.
   * [A Game of Cat and Mouse (Challenge)](https://www.youtube.com/watch?v=z4iD-559NJ4)
   * [Get the Cheesy Puffs! (Recommended)](https://youtu.be/ljQX6YG8eVA)
   * [Button Click Exercise (Too Easy...)](https://youtu.be/5BymTpAKsi4)
2. Make a rough outline that pseudocodes all aspects of the game. In your outline, consider answering the following:
   * What are the rules of the game?
   * How does that translate into code?
   * Will we need to use loops? if/else statements?
   * What variables will we need?
   * Under what circumstances will our variables change?
3. Once you have a rough outline, begin the process of *coding* it out in Scratch. Your final game must include the following functionality:
   * **A Game of Cat and Mouse**
     + Users can move the cat left and right with the arrow keys.
     + Users can press the spacebar to shoot a projectile.
     + Mice fall from the sky at random locations.
     + If a mouse reaches the bottom, the player loses a life.
     + If the player shoots a mouse, the mouse is deleted, and the player gets a point.
     + Once a player reaches 0 life, they lose.
   * **Get the Cheesy Puffs!**
     + Users can move the cat up, down, left, and right with the arrow keys.
     + If the player touches the cheesy puffs, their score goes up by one, and the puffs move to a random location.
     + When the player reaches a score of 10, a victory image is shown, and the game ends.
   * **Button Click Exercise**
     + A variable that starts the game at 0.
     + A button that increases the value of the variable by 1.
     + A button that decreases the value of the variable by 1.
     + A button that resets the value of the button back to 0.
4. Once you finish, create a new folder titled "Scratch Activity" in your Google Drive Pre-Work folder. Then, to that folder, upload a text file with a link to your functioning game. Be sure to make your project publicly sharable! You can test this by opening your game in Chrome's Incognito Window.

**Bonus:**

Find ways to add your own creative touch to things! As a suggestion, consider adding any of the following features:

* **A Game of Cat and Mouse**
  + Random *power-ups* that drop from the sky and change the cat's projectile.
  + Alternative enemies with different properties from the mouse (e.g. dogs take two hits to defeat).
  + Music that changes speed as the player advances to further rounds.
  + Etc...
* **Get the Cheesy Puffs!**
  + Make the cheesy puffs a moving target.
  + Add in obstacles that the user has to avoid.
* **Button Click Exercise**
  + Add more buttons that change your variable in different ways!

**Helpful Hints:**

* Be sure to check out the supplemental resources listed in the reading chapter to get a primer on Scratch programming fundamentals.
* Try to look at other games found on the website. Then *view* the code to get inspiration for your own project. (Borrowing from the work of other developers in the open-source community is a great habit to get into, as you'll be doing the same repeatedly as a web developer.)
* Don't get discouraged if you can't complete this activity all the way. Submit what you were able to accomplish, and be proud of what you scrapped together. What you will learn, time and again, in this bootcamp, is that perfection doesn't come easy and that sometimes it doesn't come at all.
* Stay motivated! This assignment is very much in line with the sorts of challenges you will face in class. At times, this will feel frustrating and pointless, but don't cheat yourself out of a good learning opportunity.

Good luck!

**Copyright**

Coding Boot Camp © 2016. All Rights Reserved.