




# Lesson 2: Variables

Welcome back!

Download and unzip the class file if you haven't already


$$\underbrace{x}_{\text{Variable}} = \underbrace{1}_{\text{Value}}$$



# What is a variable?

- Variables are a way to keep track of some value
  - Reference it in an easy way
- You may remember these from math: e.g.  $x, y, z$ 
  - They can change value, hence the name *variable*
- Variables are how the computer stores pieces of information
- We often perform operations on our variables to find answers we want



# Variable Naming

- Can we call variables anything?
  - Depends on the language
- In Javascript, there are naming rules we have to follow, or else the computer gets confused.
- Take a look at the names to the right, we'll come back to them in a bit.

```
let X;  
let AREA;  
let height;  
let Width;  
let currentScore;  
let %totalPoints;  
let playerOne;  
let player-Two;  
let $sumOfCredits  
let x&y;  
let _lastPlay  
let isPlayerAlive  
let let;
```



# Javascript Variable Name Rules

- All Variables must be named.
- Names can contain letters, digits, underscores, and dollar signs
- Names must begin with a letter
- Names can also begin with \$ and \_ but are often used in special cases
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names
- All variable names must be unique (no two alike)

So, what do we think of these? Ok or not?

```
let X;  
let AREA;  
let height;  
let Width;  
let currentScore;  
let %totalPoints;  
let playerOne;  
let player-Two;  
let $sumOfCredits  
let x&y;  
let _lastPlay  
let isPlayerAlive  
let let;
```



# Javascript Naming Conventions

- Conventions are not rules! More like guidelines
- Generally best to use \_ between words, e.g. scavenger\_hunt
  - Camelcase also common, but more traditionally used in Python
    - E.g. scavengerHunt
- Don't begin variable names with capital letters.
- Stray away from using \$ or \_ to begin variable names
- Make your variable names informative!
  - They should provide some idea of the information they contain

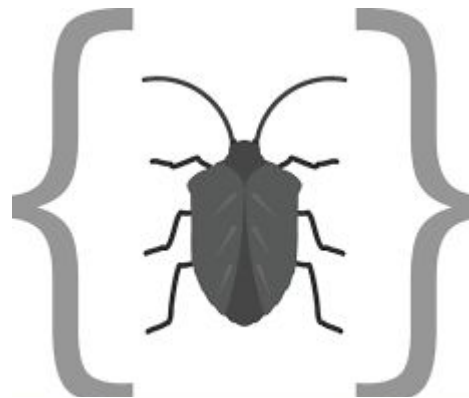


# Task #1

- What happens when we break the rules?
- Open up main.js in Atom
- Follow the instructions for task #1
- What message do you get on the console when you run the code as directed?
  - What does this mean?
  - How is this useful?

## Case Matters

- Names with different capitalization won't map to the same variable value
  - `current_speed`
  - `Current_Speed`
  - `CURRENT_SPEED`
- Helpful thing to check for when debugging!





# Variable Declaration

- Need to tell the computer what we're calling a variable before anything else
- In Javascript, we use the word “let”, as in “let x be the variable name”
  - Keeps the variable **local** - more on that later

```
let x;
```

- Always end a line with a semicolon;
- What does x represent?
  - We need to give it a value!





# Variable Assignment

- Now we give a value to our variable (we can declare and assign in a single step)

```
let cats = 3;  
console.log(cats); // -> 3
```

- Now cats will always be 3... unless we change it!
  - Variable will always evaluate to the most recent value we give it
- Take 5 minutes, and give task #2 a try in main.js



## A Simple Program

```
let age = 19;  
let name = "James";  
console.log(name, "is", age, "years old"); // ->  
James is 19 years old  
age = 21;  
name = "Gina";  
console.log(name, "is", age, "years old"); // ->  
Gina is 21 years old
```



# Variable Reassignment

- This is where the “variable” part comes in!
- After we declare a variable, we can reassign it as much as we want

```
let x = "five";  
console.log(x); // -> five  
x = "nineteen";  
console.log(x); // -> nineteen
```

- Variable always contains the most recently assigned value
- We only need to declare once! No “let” when we reassign



## (Brief) Intro to Data Types

- Data Type: which kind of information is stored in the variable
  - Numbers - couple different types, we'll learn next time
  - Strings - words! Always denoted by “ ” or ‘ ’.
- Unfortunately, we can also re-assign data types (unique to Javascript, can't do this in Java).
- `typeof` returns the data type

```
let height = 62.0; // inches maybe?  
console.log(height); // -> 62
```

```
height = "very tall";  
console.log(height); // -> very tall  
// yep, first height is a number  
// and then it's a string.
```

Later, we'll explain why this is bad



## Last New Thing: Constants

- Constants don't change, almost the opposite of variables.
- Start declaration with “const”, use ALL CAPS in the name
  - First part is a **rule**, second part is a **convention**

```
const DATE_OF_BIRTH = "04-02-2005";  
DATE_OF_BIRTH = "04-10-2005"; // <-error
```

- Errors occur when you break the language's rules, think of it like bad grammar in English.



# Combining Variables

- Often, we'll want to combine multiple variables
- This is easy when they're the same type
  - Number + Number → Number
    - Just normal math
  - String + String → String
    - This is called concatenation (see right)
- Becomes less clear when we mix types
  - Best way to discover how something works? Experiment!

```
let fish = 'tuna';  
let bird = 'robin';  
console.log(fish+bird);  
//prints "tunarobin"
```



## Let's See What You've Learned

- Get started on Task #3 in main.js
- Follow the instructions in the comments
- Take 10 minutes to get as far as you can
  - The second half is mainly guesswork, use your intuition
  - We'll review and explain the outcomes afterward
- Send questions into the chat!

# Review

Today we learned:

- Variable naming rules and conventions
- Declaring and assigning variables
  - `let variable_name = 'variable_value';`
- Changing variable values
- String and Number data types
  - How these two types combine







# Next Time

- Expressions
  - Snippets of code that let us manipulate variables
- Statements
  - Like the full sentences of code
- This is where we'll start to be able to build on what we've learned!



# Wrap Up - See you tomorrow!

Any questions?

[info.codedelaware@gmail.com](mailto:info.codedelaware@gmail.com)