Warmup

We will start every class with a warm-up 🏋️‍♀️. Please come early to make sure you have time ⏰ to work on these.

1. Make sure you have everything set up!
2. Talk to the person next to you—what is their name? What do they do? 🤜🤛
3. Take a look around with the person next to you.

**2 Welcome to Foundations**

**Lecture Slides**

# 3 Foundations of Programming

## Lecture Slides

## Exercises

### Basic Requirements

You will only need your Chrome browser with the developer console open for the following activities.

#### Vocabulary

Can you try to define the following in your own words?

* expression
* operator
* type

#### numbers & strings

##### numbers

1. Enter the following expressions, line by line, into your Chrome console. What happens for each?
2. **4 + 10;**
3. **1 \* 3;**
4. **12 \* 4;**
5. **4 % 2;**
6. **5 % 2;**
7. **5 / 1 - 99;**
8. **5000 \* -100 \* (1 + 2) \* (5 \* 6);**
9. **1241 / 9 + 99;**

**Notice how we want to end all of our expressions with semi-colons.**

1. Based on your work above, what does **%** do?
2. Calculate your age in minutes using the console.
3. The console is getting a little full. Use **clear()** or **⌘ K** to clean things up a little.

##### strings

1. In your console, write your name as a **string**!
2. Use the **+** operator to concatenate (join together) two or more  
   **string**s, e.g.:
3. **// Your first and last names as an example**
4. **"Lady " + "Gaga";**
   * Your first and last names (as shown above in the example code snippet)
   * Your favourite singer’s full name
   * Code Chrysalis
5. Try the following in your console, line by line. What happens? Fix the errors:
6. **Where are the quotes?**
7. **'hmm something is not right"**
8. **'Do other ' \* 'operators work with string concatenation?'**

#### Exploration

We want you to get comfortable with actively exploring through code. This often can mean using code that you don’t know very much about yet. But don’t worry! This is part of the process and you’ll learn them in no time.

1. Answer the following questions by coding to test the questions:
   1. What happens when I add a string and a number together?
   2. What if I reverse the order (e.g. number + string)?
   3. What happens if I multiply a number of **5** with a string **5**?
2. Try the following line by line in your console. What do you think adding **.length** does? Try finding the length of other strings.
3. **"hello".length;**
4. **"hello world".length;**
5. **"123".length;**
6. How about the below?
7. **"hello".toUpperCase();**
8. **"HELLO".toLowerCase();**
9. **"hello".endsWith("o");**
10. **"hello".endsWith("e");**
11. **"hello".endsWith("llo");**
12. **"hello".endsWith();**

Can you think of more ways to test it?

**.toUpperCase()**, **.toLowerCase()**, and **.endsWith()** are what we call native methods. They are built-in to the JavaScript language and are available only for strings. We’ll learn more about them later on.

1. Bonus question: Where can you find information about the methods used above?

## Homework

* Complete the exercises for this lesson.

# 4 Developer Environments

## Vocabulary

* **repository**: The central location where your files are stored. You can think of it as a folder.
* **source code editor**: A text editor program designed specifically for editing source code of computer programs. (Think MS Word for code)

## Lecture Slides + Directions

## Basic Requirements

1. Download the zip file from [Foundations](https://monarch.codechrysalis.io/foundations.zip)
2. Extract the files to your desktop
3. Open the repository in Visual Studio Code
4. Edit a JavaScript file, save it, and refresh the html page in the browser to see the change
5. Create a second **console.log** and have it show up in the browser

## Resources

Please read “HOW-TO-CODE” in Foundations: Lesson 1 Details.

## Review

1. What does **console.log** do?

# 5 How To Code

## Objectives

* Understand where to put files
* Understand how to write code into a file
* Understand how to run code in Google Chrome
* Understand how to develop your code

## Where to put files

Download and unzip [this folder](https://monarch.codechrysalis.io/foundations.zip). Put the **foundations** folder somewhere besides your **Downloads** folder. You can put it wherever you want, as long as you remember where it is. For example:

* On your **Desktop**
* In **Documents** or **My Documents**

Throughout the course, we will refer to this folder as “your **foundations** folder”.

Inside your **foundations** folder, there is a **\_lesson-template** folder. **You should copy the \_lesson-template folder for each lesson**. You don’t have to do everything at once. But, at the end of the course, your **foundations** folder should look something like this:

**foundations/**

**\_lesson-template/**

**functions-intro/**

**index.html**

**script.js**

**variables-intro/**

**index.html**

**script.js**

**some-other-lesson/**

**index.html**

**script.js**

## How to write code into a file

### Make a folder for a lesson

When you write new code, you should make a folder for the lesson you are working on. For example, imagine that you are working on a lesson called **functions-intro**.

* Using **Finder** (Mac) or **Windows Explorer** (Windows), make a copy of the **foundations/\_lesson-template** folder.
* Change the name of the copy. It should be something like the lesson name. For example, **functions-intro**.

should look like this:

**foundations/**

**functions-intro/**

**index.html (empty file)**

**script.js (empty file)**

### Open your foundations folder in VS Code

1. Open **VS Code**.
2. From the **File** menu, click **Open**.
3. Find and double-click on your **foundations** folder.
4. Click **Open**.

Now, VS Code should open your **foundations** folder. On the **Explorer** on the left side, find and click on the folder for the current week and lesson.

### What is inside the template folder

We will give you code to start with for most lessons. Each lesson has 2 important files, **index.html** and **script.js**. **index.html** is a file that connects your web browser to JavaScript code. **script.js** is the JavaScript code itself. All you have to do is copy the code snippets into your files.

#### index.html

**index.html** is the same for most lessons. It looks like this:

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="utf-8" />**

**<meta**

**name="viewport"**

**content="width=device-width, initial-scale=1, shrink-to-fit=no"**

**/>**

**<meta name="theme-color" content="#000000" />**

**<script type="text/javascript" src="script.js"></script>**

**</head>**

**<body>**

**<h1>Code Chrysalis Foundations Classwork</h1>**

**<h2>**

**Hit <font color="blue"> Cmd + Option + J</font> to open the dev console.**

**</h2>**

**<h2>But first we give you this gallant galloping unicorn.</h2>**

**<img**

**src="https://cdn.dribbble.com/users/1281272/screenshots/4515441/unicorn.gif"**

**/>**

**<h3>**

**No one should ever have to look at a blank HTML page with no unicorns.<br />That's**

**just a waste of pixels...**

**</h3>**

**</body>**

**</html>**

Don’t worry if you don’t understand everything about this code. You don’t need to know, but you can ask your teacher, if you are interested. You probably don’t need to ever make changes to the **index.html** file.

#### script.js

**script.js** will be different for every lesson. It contains JavaScript code. You will have to write your own code in order to make things work as you expect.

Here is an example of some JavaScript code that you might copy and paste into **script.js**.

**console.log("Welcome to Code Chrysalis Foundations!");**

**// <-- This is a comment. This line is not executed.**

**// console.log("You can remove the // symbols to 'un-comment' the line.");**

Save **script.js** after you have pasted the code.

## How to run code in Google Chrome

Usually, we use Google Chrome to browse the Internet. You also use it to open files on your computer. Pretty cool, huh?

As with many things with computer programming, there is more than one way to do something. Do whichever is fastest for you.

### Open a file from Google Chrome.

As usual, start Google Chrome. Instead of typing a web address, like “[www.example.com](http://www.example.com/)”, click on the **File** menu and then click **Open**. You can also use **Command+O** or **Ctrl+O**. Find the **index.html** file that you want to open and open it.

If you open your developer console, you can see what happens when you run your code. You can open the developer console by using **Command+Option+J** or **Control+Shift+J**.

### Open a file from VS Code

In VS Code’s Explorer (on the left side), find the **index.html** file that you want to open. Right-click on the file and choose **Reveal in Finder**. Then you can double-click on the **index.html** file in your Finder window.

## How to develop your code

Programming involves a lot of trial and error. When you are writing code, you should have 2 windows open.

* VS Code - Where you edit and save your code.
* Google Chrome - Where you refresh and run your code.

When you want to make some changes to your code, look at your VS Code window and make the changes. **Be sure to save the file.** Then you can look at Google Chrome and refresh (**Command+R** or **Ctrl+R**). Google Chrome will execute the latest version of your code. Awesome!!!

1

# 6 Intro to Variables

## Objectives

* Be able to declare variables and assign values
* Know when to use **var**, **let**, or **const**

## Lecture Slides

## Paired Activity

For each of the following code blocks, use a whiteboard or pen and paper to reason about what the value of **x** is supposed to be on the last line with your partner.

Once you have arrived at a conclusion that you are comfortable with, enter the lines into a console and check your answer. Was your hypothesis correct? If not, understand why (talk with a classmate, or ask for help).

**let a = 5;**

**a + 10;**

**a; // => ???**

**let b = 17;**

**b = (b + 1) / 2;**

**b \* 4;**

**b; // => ???**

**let c = 5;**

**let d = 20;**

**c = d;**

**d = d + 7;**

**c; // => ???**

**let e = 10;**

**let f = 5;**

**e = e \* 4 - 3;**

**e + 17;**

**e = e + f;**

**e; // => ???**

## Exercises

Create a copy of the **\_lesson-templates** folder called **variables-intro**.

Inside of **script.js** complete the following exercises:

### Basic Requirements

#### Focused Practice

All of these exercises use the techniques from the this lecture.

1. Create variables called **name**, **age**, **currentTask** and **isProgrammer** and assign them values so your test cases print what you expect! Use the keywords **let** and **const** instead of **var** (you will need to think about which variables should NOT be reassigned).
2. **// Examples**
3. **const school = "Code Chrysalis";**
4. **console.log(school); // should print "Code Chrysalis"**
5. **let week = 1;**
6. **console.log(week); // should print "1"**
7. **// Your code**
8. **console.log(name); // should print your name**
9. **console.log(age); // should print your age**
10. **console.log(isProgrammer); // should print "true"**
11. **console.log(currentTask); // should print "1"**
12. Re-assign **currentTask** to 2.
13. **console.log(currentTask); // should print "2"**
14. Compute some basic geometry. You may need to use Google to remember the formulas.
15. **// Compute and store the area of a square**
16. **const squareSideLength = 2;**
17. **// Your code here.**
18. **console.log(squareArea); // should print "4"**
19. **// Compute and store the area of a rectangle**
20. **const rectangleBaseLength = 3;**
21. **const rectangleHeightLength = 4;**
22. **// Your code here.**
23. **console.log(rectangleArea); // should print "12"**
24. **// Compute and store the area of a triangle**
25. **const triangleBaseLength = 4;**
26. **const triangleHeightLength = 5;**
27. **// Your code here.**
28. **console.log(triangleArea); // should print "10"**

### Medium Requirements

#### Challenging Practice

These exercises may use techniques that were not covered in this lecture.

1. More basic geometry. Some of your answers might be a little bit different than the expected values. Don’t worry. They should just be close to the expected values.
2. **// Compute and store the circumference and area of a circle**
3. **const circleDiameter = 10;**
4. **// Your code here.**
5. **console.log(circleCircumference); // should print something close to "31.41592653589793"**
6. **console.log(circleArea); // should print something close to "78.53981633974483"**
7. Without directly reassigning the values, swap the values of **a** & **b** so that your test cases print what you expect.

Hint: you will need to create another variable to store data temporarily.

**let a = "B";**

**let b = "A";**

**console.log(a); // should print "A"**

**console.log(b); // should print "B"**

1. Exponential growth. Change the value of a single variable to contain the powers of two.
2. **let value = 1;**
3. **// you do something to value**
4. **console.log(value); // should print "1"**
5. **// you do something to value**
6. **console.log(value); // should print "2"**
7. **// you do something to value**
8. **console.log(value); // should print "4"**
9. **// you do something to value**
10. **console.log(value); // should print "8"**
11. **// you do something to value**
12. **console.log(value); // should print "16"**
13. **// you do something to value**
14. **console.log(value); // should print "32"**
15. **// you do something to value**
16. **console.log(value); // should print "64"**
17. String concatenation. Concatenate different variables to display the desired result.
18. **const firstName = 'your first name';**
19. **const lastName = '???';**
20. **const city = '???';**
21. **console.log(???); // should print something like "Hello, my name is Yan Fan. I live in Tokyo."**

### Advanced Requirements

For these exercises, you may need to use techniques that we haven’t covered in class.

1. Declare a function called **counter** that, when invoked, always **console.log**s a number that is one more than the previous invocation.
2. **// hint: you may need to do something here**
3. **function counter() {**
4. **// your code here**
5. **}**
6. **counter(); // => 1**
7. **counter(); // => 2**
8. **counter(); // => 3**
9. **// etc.**
10. Declare a function called **isOdd** that, when invoked, **console.log**s whether or not the given number is odd.
11. **function isOdd(givenNumber) {**
12. **// your code here**
13. **// should print "Yes, it's odd" or "No, it's even", depending on the given number.**
14. **}**

## Homework

* Complete the **Basic Requirements** for this lesson.
* Complete the **Additional Reading**.

### Additional Reading 📖

1. [DevTools](http://javascript.info/devtools)
2. [Eloquent JavaScript Chapter 1 - Values](http://eloquentjavascript.net/01_values.html)
3. [JavaScript for Cats: The Console](http://jsforcats.com/#basics)
4. [JavaScript for Cats: Strings](http://jsforcats.com/#strings)
5. [JavaScript for Cats: Values](http://jsforcats.com/#values)

Course Evaluation - コース評価

We really try hard to make sure our course is the best it can be. Please give us constructive feedback on how we can improve. - Code Chrysalisではこのコースを常に改善することに取り組んでいます。どのように改善できるか、建設的なフィードバックをいただけるととてもありがたいです。

\* Required

Email address \*

Your email

Name - 氏名 \*

Your answer

How would you evaluate your understanding of the material today? - 今日のクラス内容または教材についての理解度を教えてください。 \*

0 - Lost: I was so lost, I didn't even know where to start. - 方向性を見失い、どこから始めれば良いかわからない状況。

1 - Struggling: I struggled through this task and I do not think I did it correctly. - 苦戦しており、正しく理解できていないと思われる。

2 - Getting There: I think I did ok. It was difficult, but I managed and I am feeling progress. - なんとか理解できており、難しくはあるが追いつけている。

3 - Acceptable: I was able to complete the basic requirements and I am feeling ok. - 基礎の課題は全てできており、理解できているとおも

4 - Strong: I was able to complete the basic requirements and keep going. I feel good. - 基礎の課題は完璧で、これらの内容は理解できた。

Any comments about the score you gave yourself above? - 上記のスコア付けをしていただいた理由等のコメントをください。

Your answer

What went well today? - 今日はどこまで理解できたかのコメントをください。 \*

Your answer

What did NOT go well today? What could have been better? - 今日はどこで理解できなかったか、より良く理解するためには何が必要かを教えてください。 \*

Your answer

Any other comments? - 他に何かコメントがあればコメントをいただきたいです。

Your answer

Submit