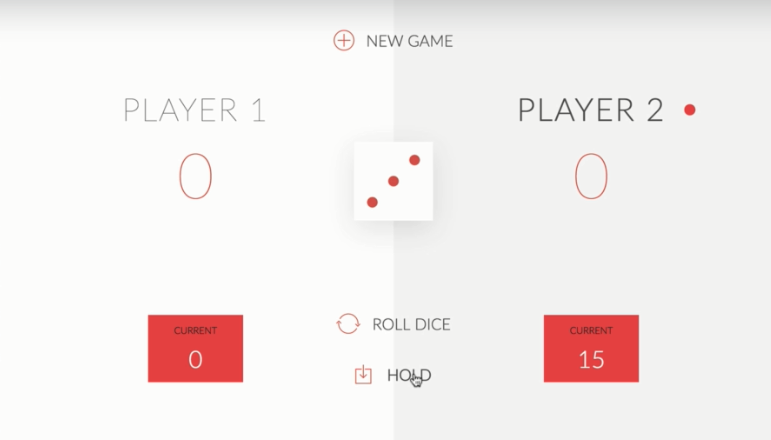
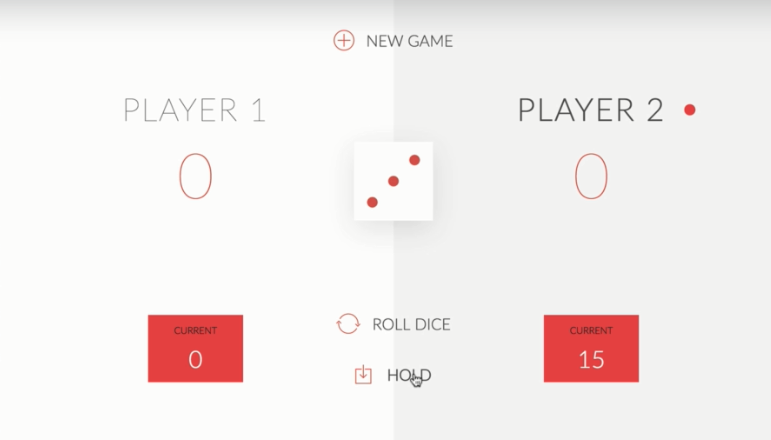
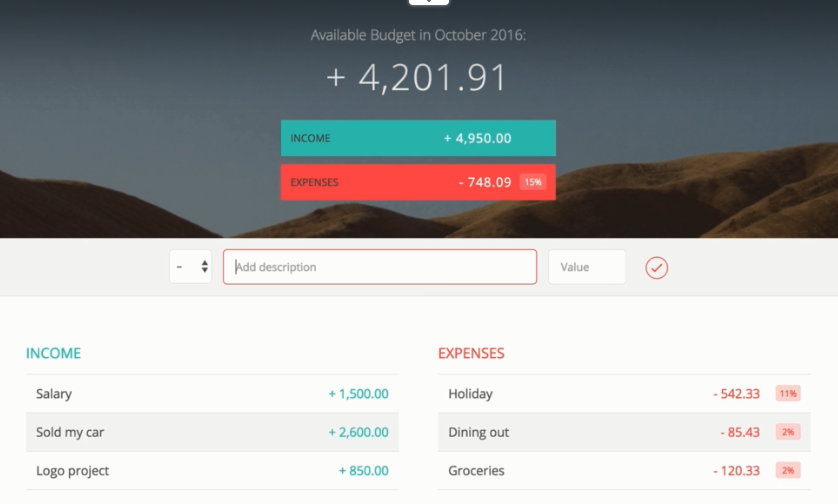
# JavaScript Walk-through

I would like to first thank you for taking your time to dive into this extensive JavaScript walk-through, which will take your understanding and knowledge of JavaScript to fit the needs in your professional environment. It is one of the most powerful developing languages and also powers a lot of the applications that most people interact with on a daily basis. Below you are going to see what the different topics are that we are going to be covering.

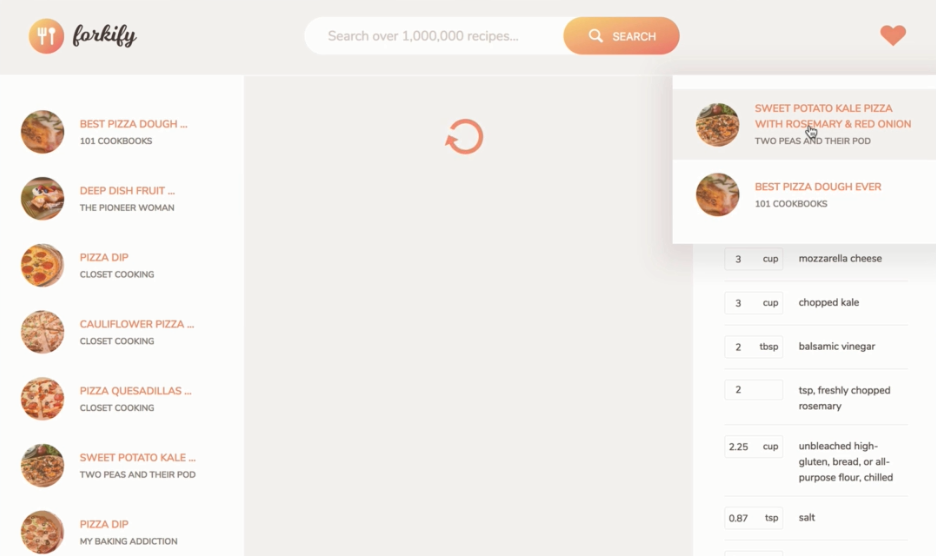
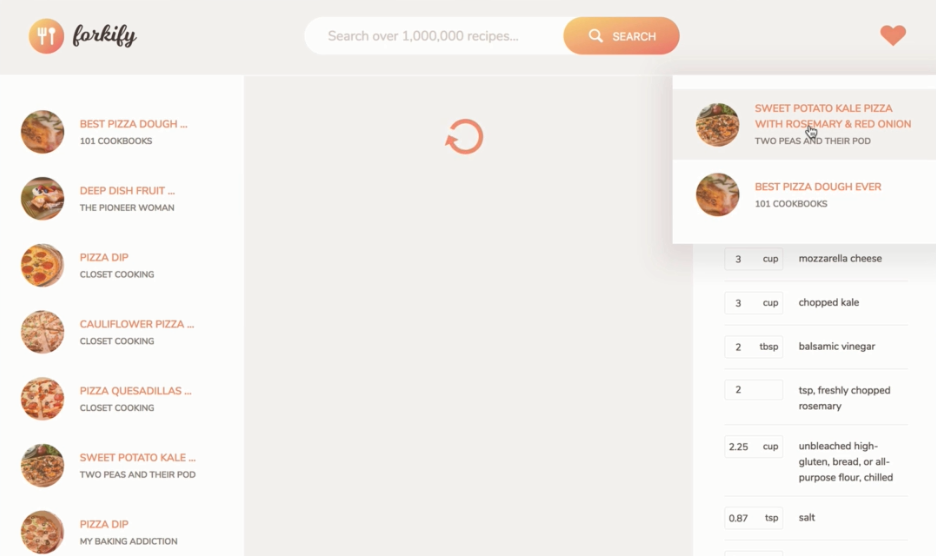
* Walk-through Content
  + JavaScript Language Basics – Will cover the bare basics of JS that you will take with you everywhere as you continue to learn/develop in JS.
  + How JavaScript works behind the scenes – This will cover how the language works, what the language does that we don't see and how everything comes together to create the framework of the future.
  + JavaScript in the Browser: DOM manipulation and events – This is going to bring us to your first little project that you can create if you want to follow along. It is going to be a dice rolling game. The UI will look something like what you see below.



* Advanced JavaScript: Objects and Functions – We will be covering things like prototypes and inheritance, class functions, closures and a few other topics.
* Putting it all together. The Budget Application Project – This section will give you the opportunity to take everything that you have learned up to this point to create a budge application that will allow you to add items that are income and other items that are expenses. See below for an idea of the UI.



* Next Generation JavaScript: Intro to ES6/ES2015 – This will include ES6, ES7 and even some ES8 features that you have at your disposal. First, we will look at what is new in ES6 and then move forward to see what is available with the newer versions syntactically.
* Asynchronous JavaScript: Promises, Async/Await and AJAX – These are used for loading things from a given server with AJAX.
* Modern JavaScript: Using ES6, NPM, Babel and Webpack – This will be the final project that you are going to be able to work on. It is going to combine everything that you have learned so far, plus the addition of learning some more modernized tricks when it comes to ES6. Below is what to expect for the UI of this project.



Getting Started

Before we get to diving into things, we should get you started with some resources. For the first resource you are going to have the repository where all these projects you will be working on can be found in addition to the instructions for them. At the bottom of this page you will find an FAQ section that you may find beneficial. Next up for the second resource this is going to be a collection of different resources that you can use to help extend your knowledge. This will contain different access points for VS Code, Emmet Cheat Sheets, Codepen for testing, and the use of brackets if you choose to use that editor over VS Code.

<https://github.com/jonasschmedtmann/complete-javascript-course>

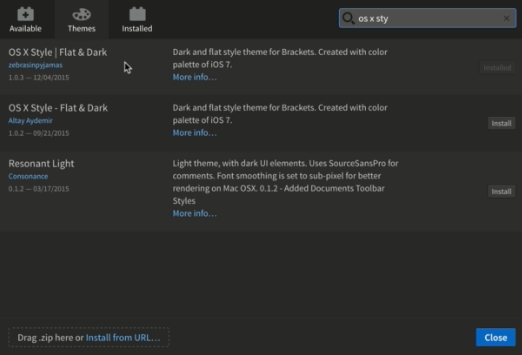
<http://codingheroes.io/resources/>

Setting Up Tools

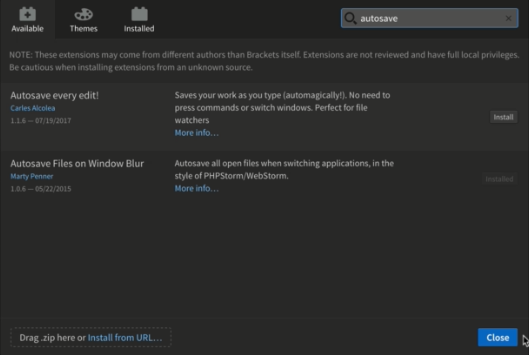
Before we can begin writing JS code, we are going to need to setup our environments so that you can. For this as mentioned above, you will either use Brackets or VS Code to follow through with this. If you do use another editor, know that you may run into issues with the execution of the code. If you already have an editor installed that you are planning to use, that will be fine, just ignore the setup part. It is important to note that the later projects are created within VS Code. Within the repository supplied above there are actually steps there that oyu can follow for setting up both editors.

<https://github.com/jonasschmedtmann/complete-javascript-course/blob/master/editors-setup.md>

In the event you choose to use brackets follow the steps directly below this. We are going to set everything up to where your brackets will look just like the screenshots. First you are going to want to click on the Extension Manager on the right side of the editor.



Once you have that you will need to also setup the extension for autosave. This will automatically save your files as you are making modifications. By using this, it will help to ensure that you don't lose any of your code along the way in the event of a crash. This isn't required for you to proceed with this, but it should help you in the long run. By doing this you will save yourself the frustration and the issues that tend to come with data loss.

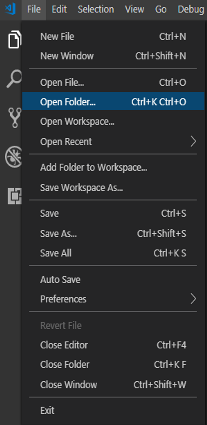


Remember that there is the setup there for VS Code, which we aren't going to go over currently. However, it should have a very similar setup to what you are seeing above for bracket. There is a lot more extensions that you should consider adding to the VS Code that we will make use of later in the walk-through.

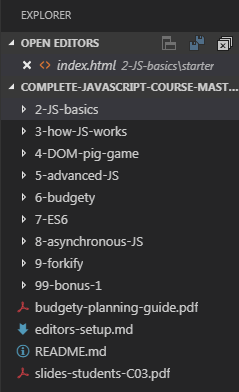
Which will bring us to the next thing that you are going to need. You will need to pick a browser that you are wanting to have your code executed in. For this, I am using Google Chrome as my browser, you can choose whatever you are the most comfortable with. The reason I am using chrome is we have the developer tool at our disposal that already has a console built-in to the browser. This will allow you to perform operations with values and even variables to execute calculations or other tasks.

# JavaScript Language Basics

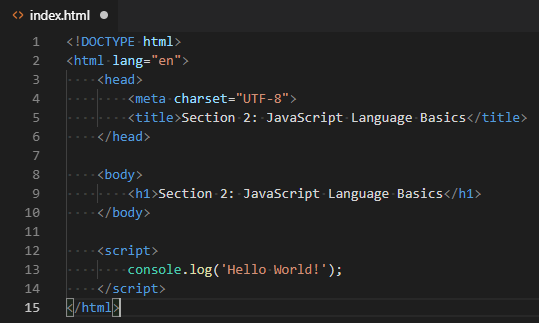
Throughout this section you are going to be learning about the basics of JS. You may be wondering, but what do the basics consist of? This would be variables, data types, functions, objects arrays and many other various topics. Not only are you going to be learning the basics of JS, but you will be learning the fundamentals of programming to make the content more desirable. It is important to note that these are the basics and not necessarily the way that you will be creating applications, but to get a better concrete understanding of JS. Look at the below image.



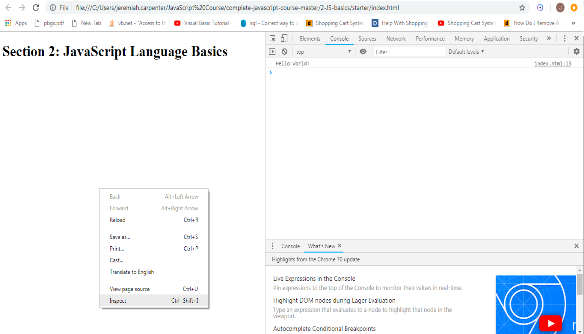
If you are following along and decided to use VS Code, I have supplied a screen shot above of how you can get the whole course pulled up within the environment. Simply go to File → Open Folder and then select the folder where the content is all contained. Your tree for your project should look something like you see below.



You will notice that this is nothing more than an HTML file that is bare to begin with. So, let's get to adding some JS to this. There are two major ways that you can accomplish this. The first is the use of in-line JS, which is going to be directly within the file using <script> tags. The other will be the use of a JS file instead that is imported into the file. We are going to begin with the use of the script tag which you will see below. In this example we are using the console in the developer tools to display Hello World.

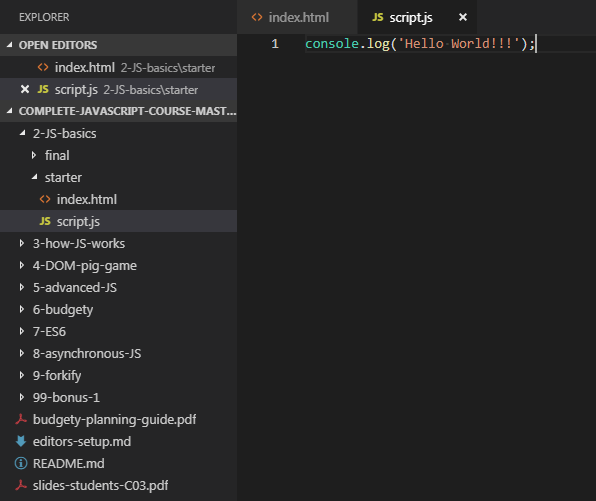


So how do we see this in action? You are going to need to go to the folder location where this HTML file is being saved(index.html). When you find the file you just simply need to double click or open the file. This is going to load up the browser, but where is our log message? This is going to be within the Developer tools which you can access with F12 or by right-clicking and selecting Inspect. Once you have the tools open navigate to the Console tab and you should see your message like below.

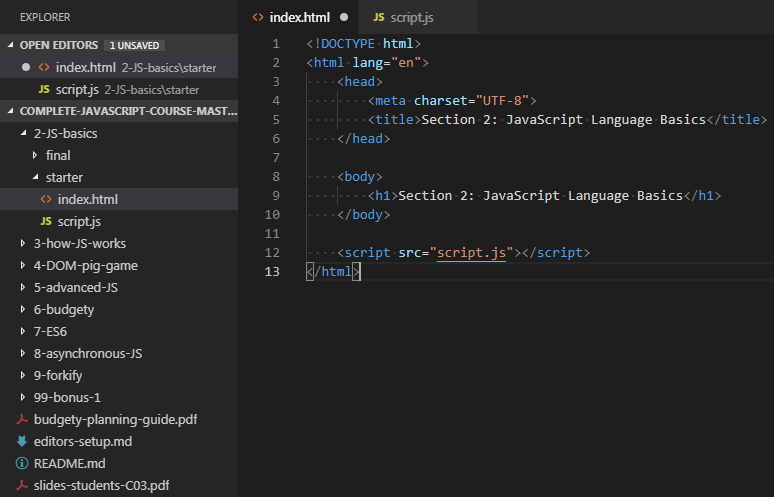


Sure, we are only sending a message to the console and not actually using JS to its full potential. However, this is going to be the format that we are going to follow while we traverse through basics and fundamentals until a later section. One important thing to note about the console is that you will notice on the right there is an index.html.13 which tells you what file executes the log, as well as the line the code originates from.

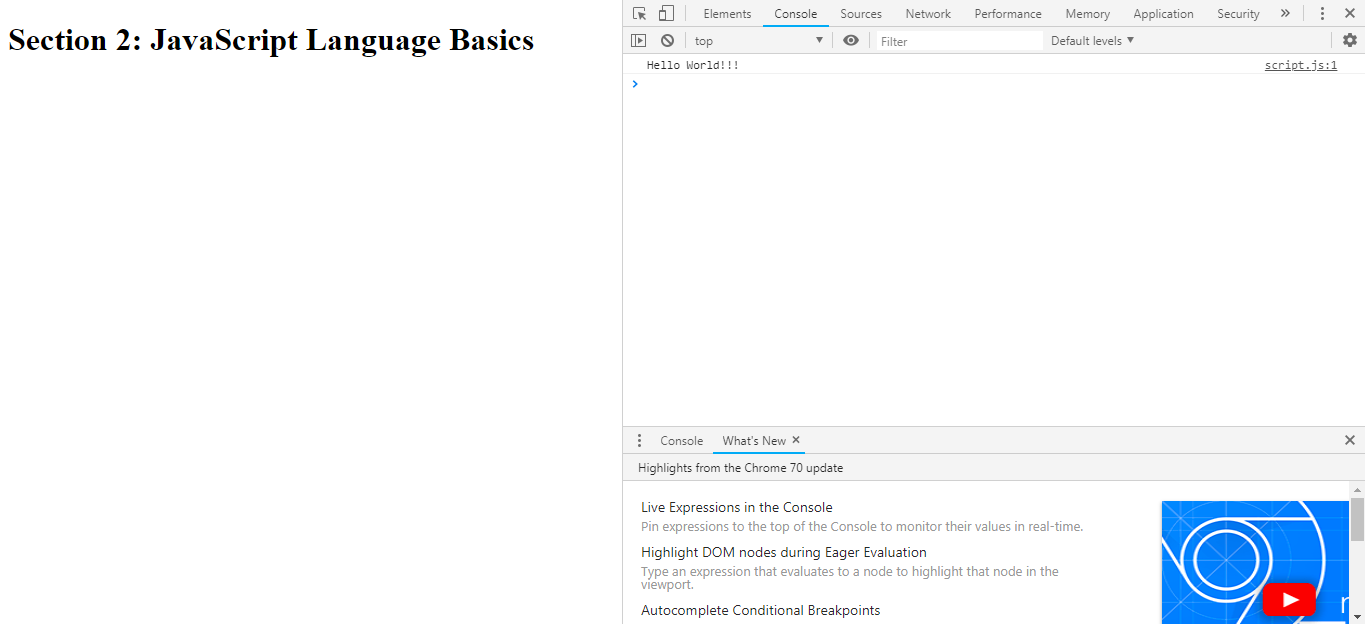
We are going to create a new file that is in the same directory as our index.html. This is going to be called 'script.js'. In here we are going to store our console.log we previously created. You should have something like below.



However, we are not quite done yet. You need to remove the old console.log that we originally had in the body of the script tag. Instead we need to access 'src' as a member of script to the file where we stored this console.log. In the event this file isn't located in the directory, you will have to append the location in the project correctly, but in this instance it is.

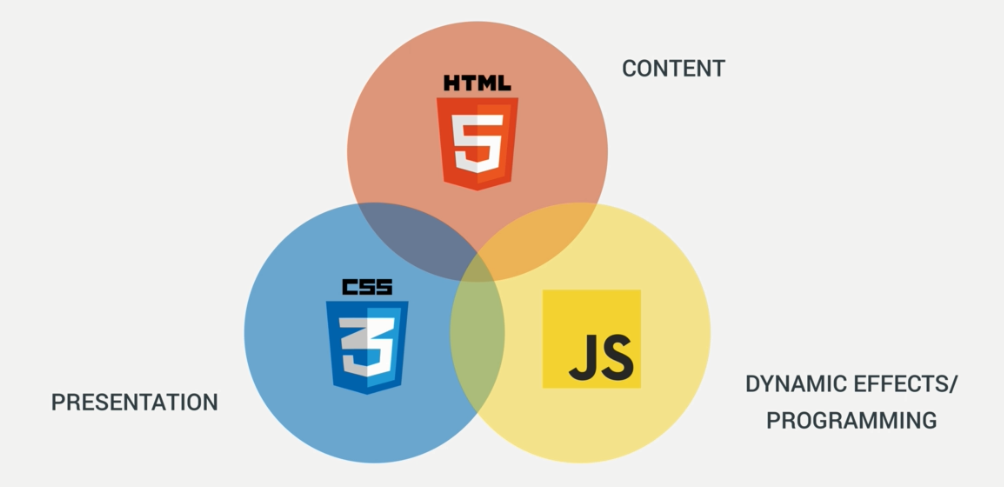


Now that you have that, all you need to do is save the file and then either refresh the browser or access the file again like in the previous steps we took to have it in the browser. NOTE: we will use the script src for the rest of this, so try to become familiar with this process.



Great! You have now created your very first application using JS. So, then what is JS exactly? It is a lightweight, cross-platform, object-oriented computer programming language. I know this sounds complicated, but it really isn't. When we say lightweight, we mean that it doesn't eat up the resources during run-time/compile-time. Cross-platform means that we can use in multiple systems, environments without there being a negative impact on the framework. Lastly, object-oriented means that the programming style is handled through objects, which we will touch base on later in-depth. It is important to note that JS is one of the three core technologies of web development in the modern world.

So where can JS be used? It can be used in different places but there are two major categories they fall within. The first is client-side which is what JS was originally designed for was to be used only in the browser. Secondly, we have server-side which thanks to node.js we can use JS on the server side, but this is a very broad subject, so we won't dive into. JavaScript is what made modern web development possible through dynamic effects and interactivity, as well as modern applications that we can interact with on a user-friendly level. NOTE: there are frameworks/libraries like React and Angular are 100% based on JS. To be able to fully understand languages like those, you will need to have a concrete grasp on JS in order to use them. There are three major languages out there which are HTML, CSS and JS.



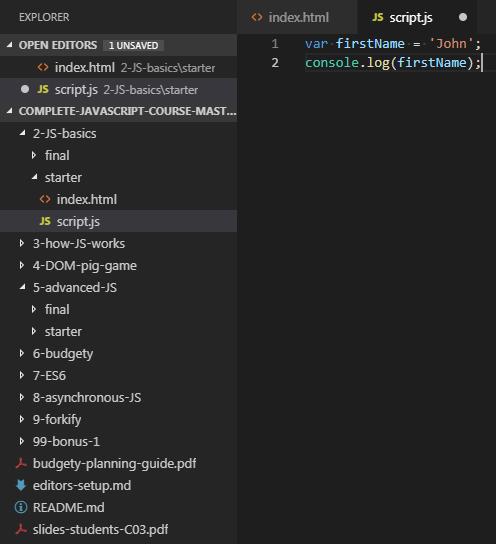
All three work together to create beautiful, function and user-friendly browser experience for the users. So, let's dive into this a bit with how JS ties into this. First you have the HTML which is going to decide which elements are going to be displayed to the browser. Next you have the CSS which is going to be the way that you want the elements to be displayed including sizes, colors, effects, etc. Lastly, you have the JS which is going to be all of the dynamic effects/behaviors that the browser is going to render. We use helpers to be able to make use of these specific effects/behaviors and specific functions/procedures.



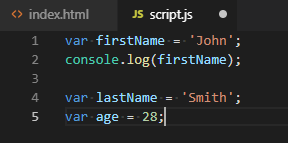
Another way for you to be able to determine the difference between the three consider the describing words above. HTML pretty much determines which nouns to use within the browser or the specific items like a paragraph, header, footer, tables, etc. The CSS portion of these is adjectives about the given elements(noun) or the objective; whether it is color, size, where to place, margin between, etc. Which then brings us to JS which is the verbs or actions that your application is going to make. This is all of the functionality that you are wanting your applications to have through the use of helpers. Great, so you kind of know how these three works together. Below, you will see the flow in which we are going to start touching the content. We will begin with ES5, move into ES6/ES2015, then into ES7/ES2016 and finally into ES8/ES2017.



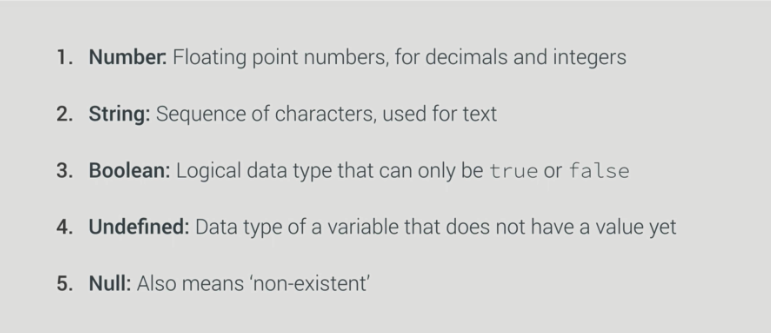
Moving on we are going to begin looking into variables and data types. It is important to know that these are used in all modern programming languages, so this will help give you some more concrete fundamentals for you to build upon. You could look at a variable as a container that we store a given value within, that can be used repeatedly. Below you will see an example of how to create a variable in JS.



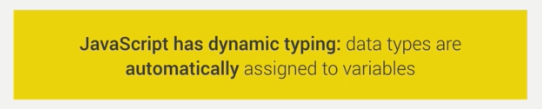
So, what do you think this is going to do? If you said that you think this is going to print to the console 'John' then you have guessed correctly! So, let's create two more variables. The first is going to be lastName and the other age. LastName you need to assign 'Smith' and for the age just 28. You will notice that these both look different like below.



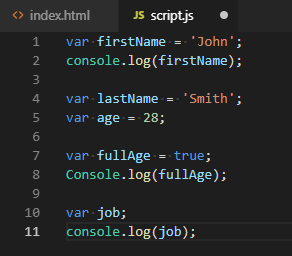
This is the perfect time to bring up data types, as therefore they both appear to have different ways of visualizing their value. In JavaScript there are five distinct data types. Notice these are native primitive types and doesn't cover the variation of other types available. The first is going to be a Number which is a floating point, that is used for decimals and integer values. Up next you have the string which we would use for a literal value like John or Smith in this case. After that, you have the Boolean which is a logical data type that can only be true or false. It is important to note that Booleans are the heart and soul of programming due to them being used to check multiple conditions, which will execute code depending on the condition outcome. Now we have the Undefined, which means it is a variable that doesn't have a value yet. Lastly, we have null, which may be one of the most important ones as it isn't a true value to be null. Null means that something is nothing, which ends up not being an object, which can cause null references throughout your code. Refer to below for the different data types.



One of the most important things to note in JS is that it makes use of Dynamic Typing which means data types are automatically assigned to variables on their own. NOTE: if you use var in C# it will also be able to determine the data type if there is something on the right-hand side going into the variable.



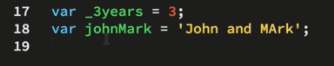
First thing you should always remember is to use meaningful names for your variables like you have already seen us use with first and last name, otherwise no one knows what the variable is for and could be confused. This becomes a problem because you won't always be maintaining your own code, someone else could wind up with that task later. For the second rule you need to follow make sure when it comes to variables you use camel case which is lowercase first word in variable and every word after begins capitalized. In the following example you will notice we have a variable job, but it is without a value being assigned to it and then logged to the console. So, what do you think is going to happen?



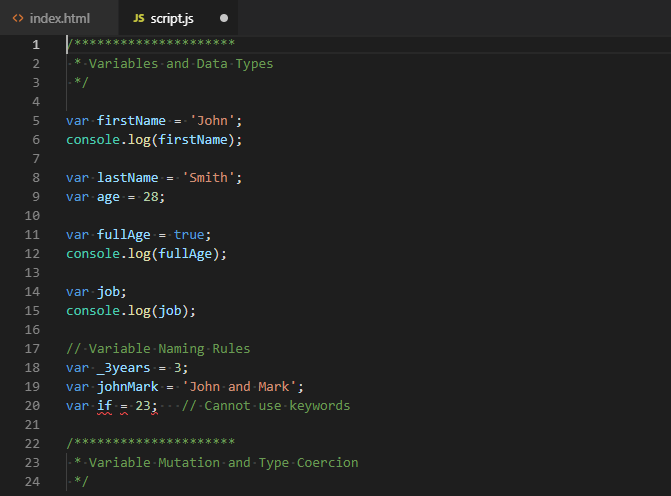
The problem you are going to run into with this is that it hasn't been defined yet, as it has no initial value.



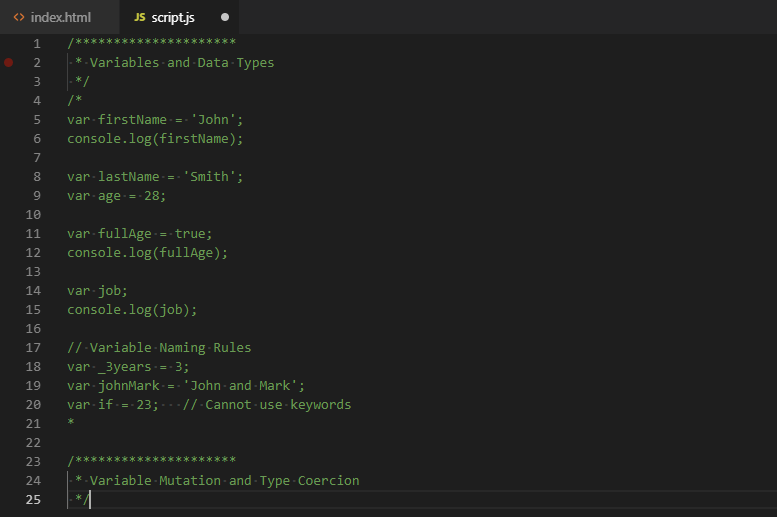
So now let's go ahead and give a value of Teacher and then attempt to log this to the console again. You will notice that there is the undefined issue, but below that you will notice that it is able to log the string. A rule to remember is that no variable can start with a number, if you try you will receive an invalid or unexpected token error. The only characters besides alpha values a variable can start with is an \_ or $. You also cannot add extra characters like the / in the middle of your variable. In addition to these rules, you also cannot use keywords like 'function' or 'delete'. The following are ways you can use valid variables.



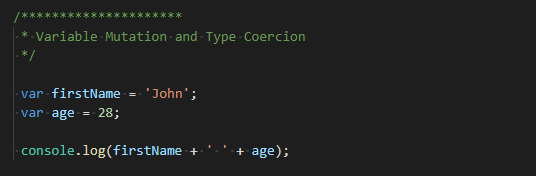
Now that we have some of the basic rules out of the way for variables, we are going to start checking out variable mutation and type coercion. First though, let's go over comments, which is used in most modern-day languages. The first is the single-line comment which is '//', allows you to leave notes about the functionality of something or even what something is, and this is ignored by the compiler. After that, you have the multi-line which can span over numerous lines in the code. This is accessed by using '/\* \*/'. Refer to below.

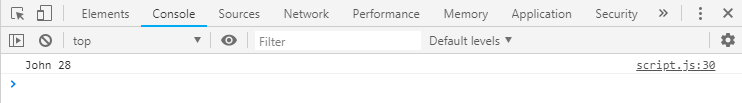


Another reason that you may want to consider using comments is if you find a block of code isn't optimal and it could be done better. In this case, you would select all the code that you are wanting to not be executed by the compiler like you see below.

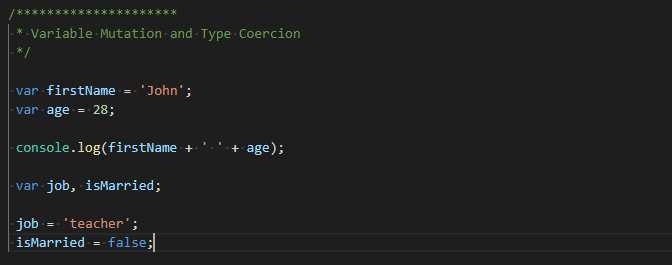


Great, so you learned about comments! Now let's start moving towards mutations and coercion. Let's begin with something simple, you are going to need a firstName(John) and an age(28). You are going to want to write this to the console but first you need to use the firstName and then add ' ' followed by adding the age. This is known as concatenation, which allows you to append more than just one thing together to write to the console. Look below to see how this would be accomplished and the output the console should write out.

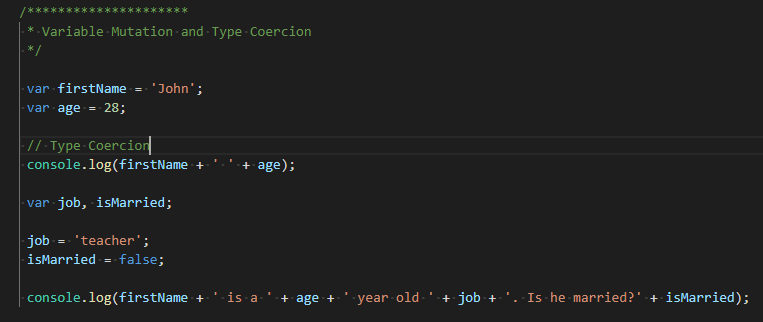




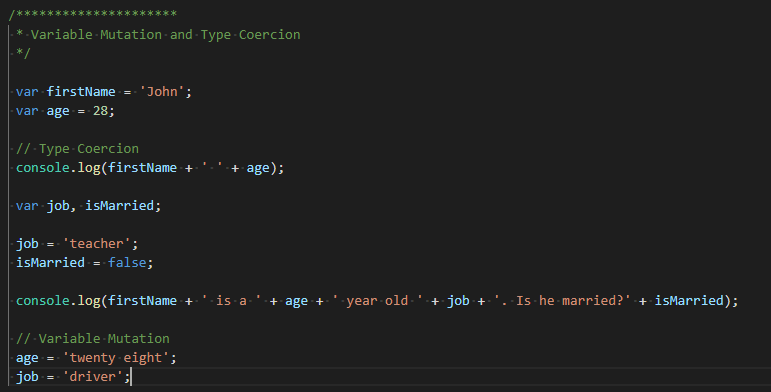
As mentioned before this is known as coercion, which will automatically convert the data type to ensure that the value is able to be used as a string, even though it is an integer(number). JavaScript does this all on its own, whereas with other languages like C# you would have to handle the conversion yourself, otherwise it would throw errors during run-time. We are now going to start looking into creating multiple variables and then initializing them at a later point, but all within one line when the declaration of the variables is first created like below.



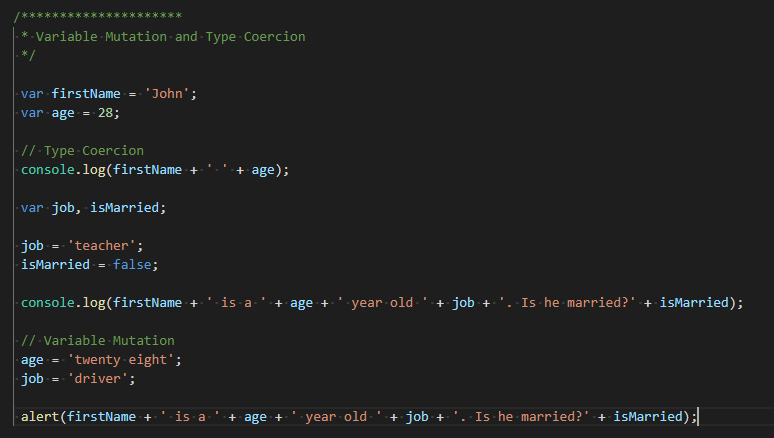
It is very important to note that if you do declare your variables like this, that you must initialize their values, or they will error as undefined when you attempt to compile. So now let's try to write to the console John is a 28-year-old teacher. Is he married? False. This is going to require a lot of concatenation or coercion; whichever term is easier for you to remember. So, let's look at how to accomplish this.

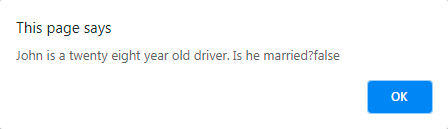


We have now finished going over coercion, so this is going to leave us with variable mutation. I am sure that this word may seem a bit intimidating, but all it means is to change the value that is stored within your variable. Let's store the string 'twenty-eight' into the variable. NOTE: The variable has already been created, so you don't have to use the 'var' keyword before you do it. Remember the magic in JS is the fact that it figures out the data type on its own based off the value that is stored within the variable. In this instance, it would now recognize the variable as a string the given context. In addition to that change we are also going to change the job to be 'driver' instead.

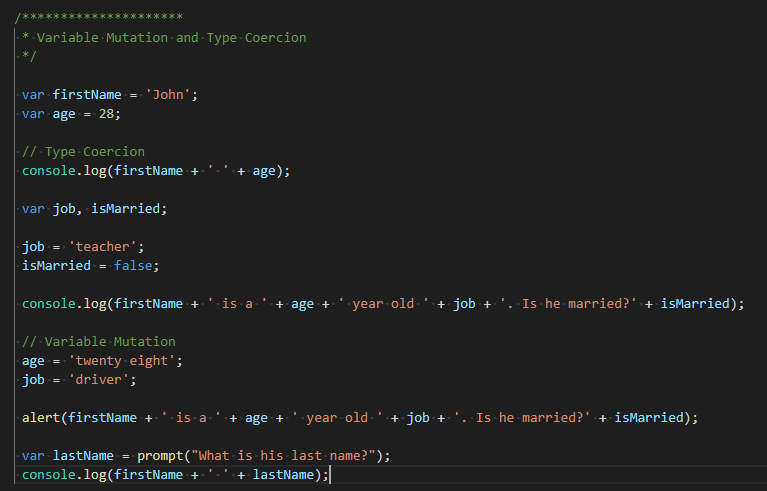


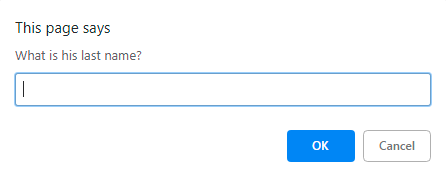
Unlike with the previous examples where we were writing to the console, we are going to instead focus on the alert function that is in JS. This is going to give you a user-friendly popup alert that is going to have an OK button that will just simply close the alert. Let's go ahead and look at this.



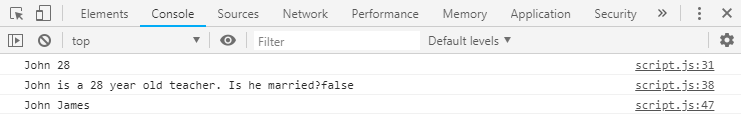


That is great right!? Well this is going to bring up a hypothetical situation. What happens if we are wanting to receiving input from a user? We can make use of the prompt function within JS. Use the prompt function just like with the alert, but instead ask the question 'What is his last name?'. The task you are given for this is to take this function/question combo and store it into a variable called lastName. Lastly, make sure to log the response with first name and last name to the console so we can get visual verification that this is working properly.

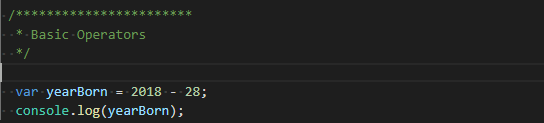


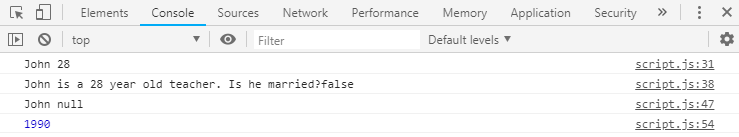


That is cool huh? That isn't the only thing. Look at the console and you will notice that it now is outputting the first name and last name together.

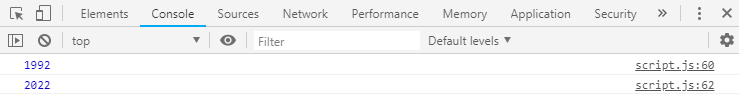


We will now be moving on to operators and the different ones that you have at your disposal. So, what is an operator? It is a symbol that usually represents and action or process. Symbols were adapted from mathematics and logic. They can manipulate a certain value or operand. We know that the year is 2018 and that John is 28, so why not try to calculate the that John was born? Let's begin with simply creating a variable 'yearMark' which is going to figure 2018 minus his age. Lastly, make sure that you write this to the console so we can see it in the browser.

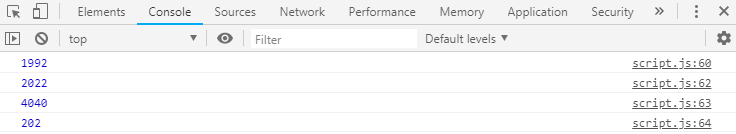
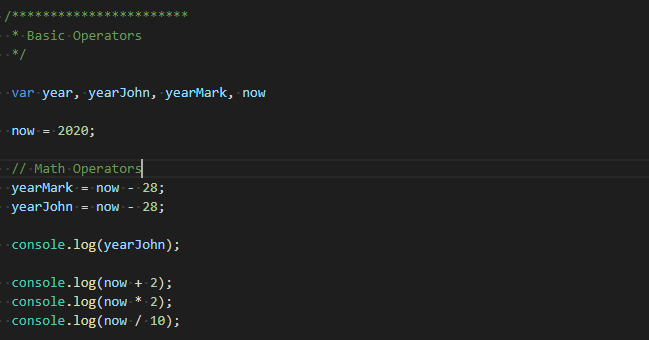
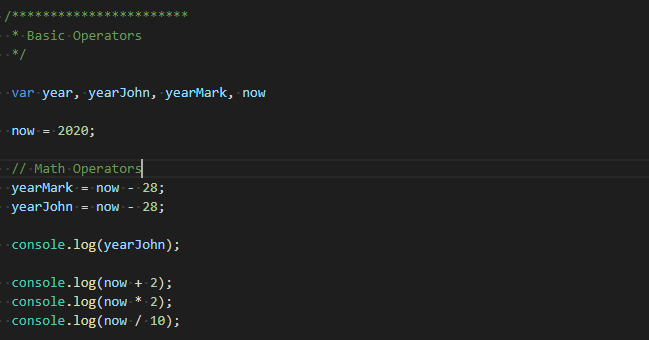




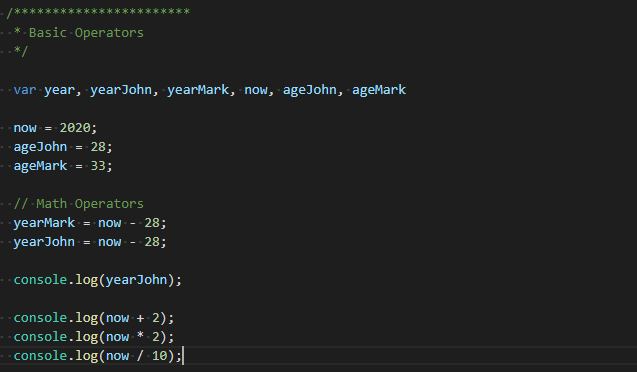
So, let's assume that you have another person that you need to find the year they were born as well. This person is going to be Mark and he is 33 years old, so go ahead and situate that. However, it is important to notice that we are repeating a given value, so why not store that within a variable(2018)? Let's declare all these variables together and then initialize their values below the declarations. In addition to this, notice that the year could now be changed to whatever year you want it to be. So, let's change the year to be 'now' instead. Then we are going to log to the console the 'now' value plus 2 years.



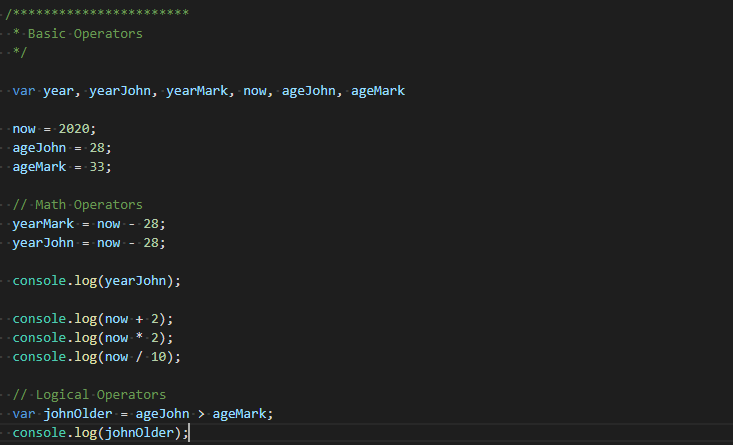
You can think of all of these as Math Operators. In addition to adding years, we can also multiply and divide the years like so...

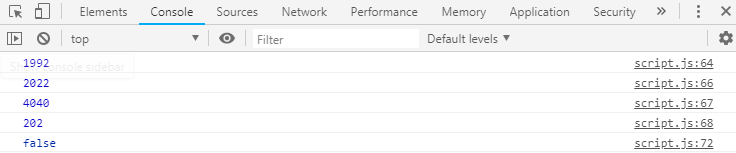


Which is going to take us into the Logical Operators. Let's say that you want to evaluate whether John is older than Mark. We now know how old both John and Mark are, so we are going to store these into 'ageJohn' and 'ageMark' instead of using it directly within the calculation. Make sure to remove the values for the calculation and replacing it with the variables we have assigned them to. Refer to below.

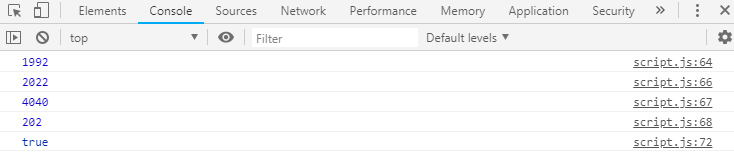
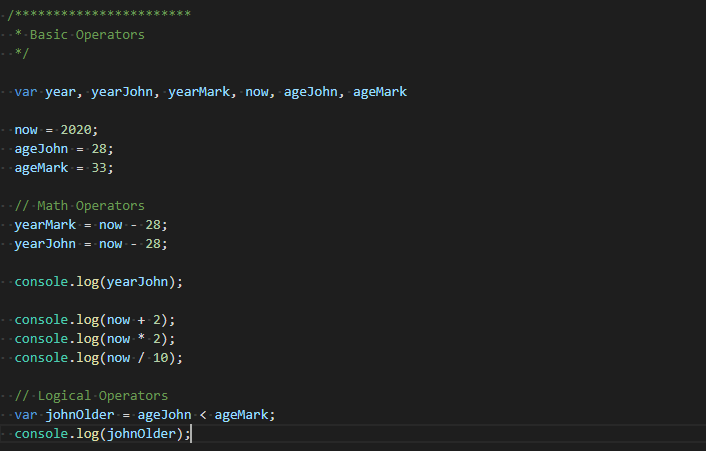
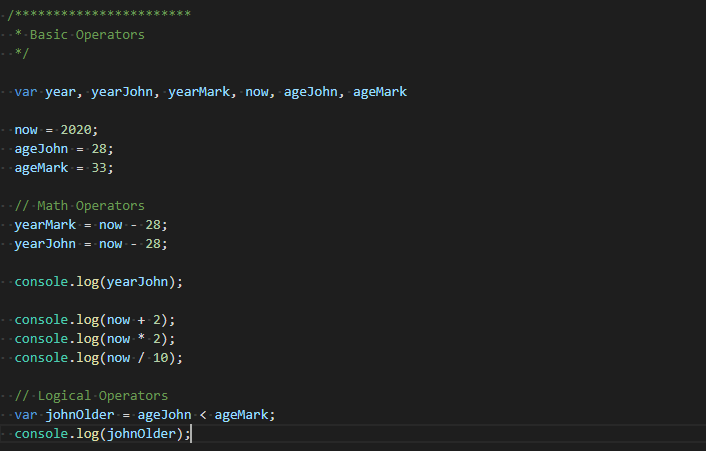


We are now going to try one of these logical operators. So, create a variable 'johnOlder' which is assigned ageJohn > ageMark. Next, make sure that you log the johnOlder to the console, save and then refresh.

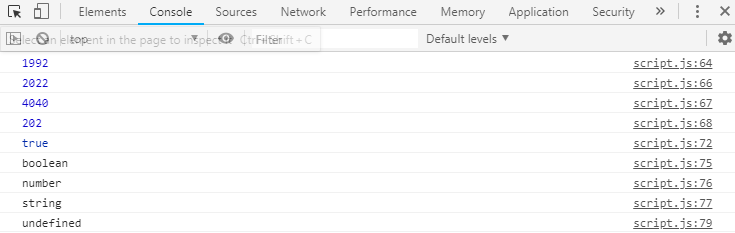
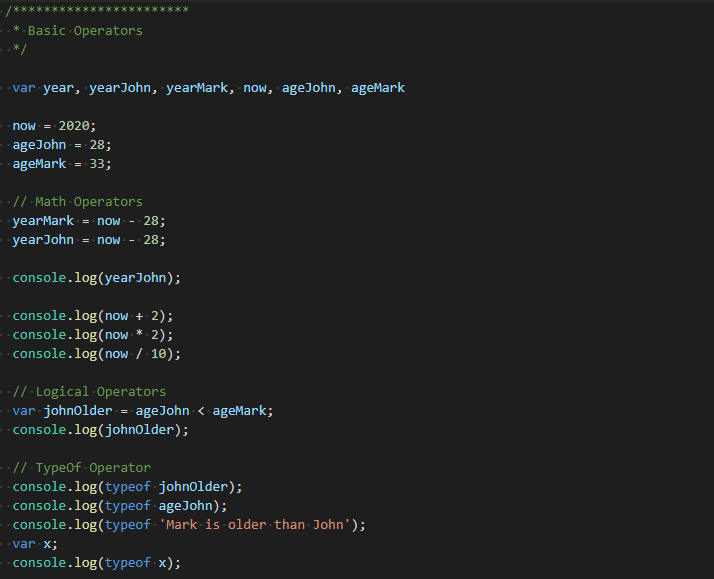
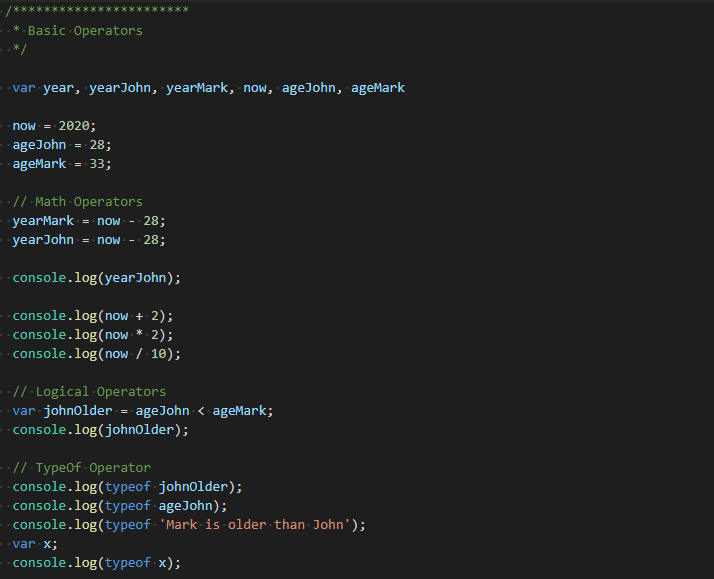




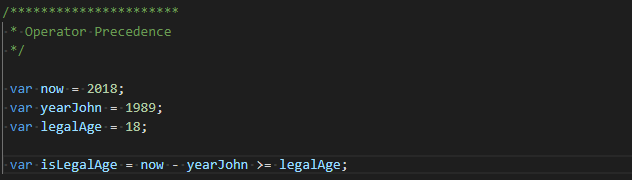
You may be wondering but why did I get false and not a different value? This is just a comparison where it checks to see whether the question, we ask is true or false. The way we wrote this, we asked if John was older than Mark, which we know to be false as 33 is greater than 28. Based off that, if we were to flip this operation around to see if John was younger than Mark, we would have a totally different result.



Which is going to bring us to another operator which is the TypeOf Operator. Both of our previous operators both required the need to have two things whether its two numbers to calculate or to compare against. With this operator though we only need one. This operator will take a variable and then return what its data type was. So, let's do this for johnOlder, ageJohn, 'Mark is older than John', then create a variable 'x' but assign no value. Now use the typeof operator on all of these like below.



That is going to wrap up operators, so now we are going to work on focusing on their precedence. Let's say we want to calculate if John if he is no longer a minor(18). Let's begin by create a few variables now(2018), yearJohn(1989) and legalAge(18). Once you have that create a new variable isLegalAge which you are going to check the current year minus John's birth year. Which we are wanting to see if it is greater or equal to the legal age. Refer to below...

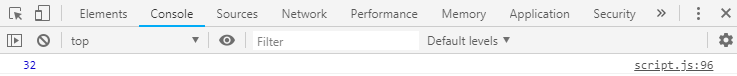


So why does this work, we are doing two different things within this? This is where precendence comes into play. When you run this, it works even though your brain would tell you that it wouldn't when you first see it. Let's take a look at the operator precedence at the following link.

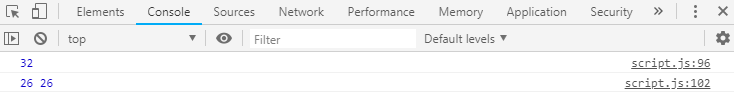
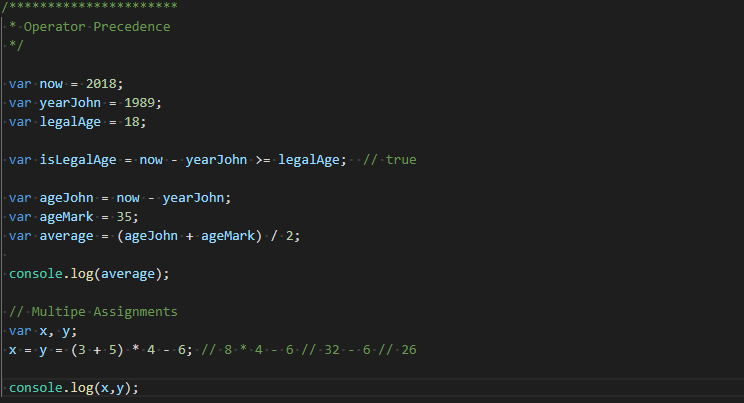
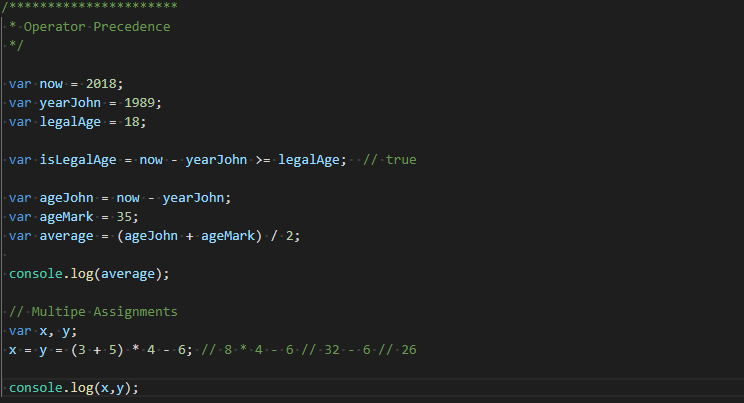
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Operator_Precedence>

On this page there is a section under table, which is going to show you all the precedence that each different keyword has within JS. This goes from the most important to the least important. You will notice that the – operator has a higher precedence than the greater or equal to operator. It is very important to become familiar with this to an extent as you continue to develop.

Moving on we are going to work on our next example where we want to find the average age between two people. First, we are going to take the 'now' and subtract 'yearJohn' and store into a new variable 'ageJohn', an ageMark which is simply just 35 and then an 'average' that takes ageJohn plus ageMark divided by 2. Based off the precedence table, this isn't correct as division is handled before addition. You should however enclose both ages within () as they have the highest precedence and then divide. Lastly, write this average to the console to see your results.

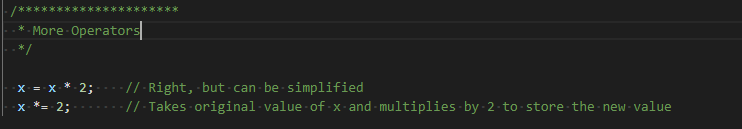


There will be situations where you want to assign to different values at the same time. So, let's get to work on this. First create to variable x and y. You are going to want to assign (3 + 5) \* 4 – 6 to x which I have handled the calculation in the comments next to it below, you can log this if you want to check it. So, which is going to bring us to how you can assign x and y at the same time. You can simply do x = y = the previous evaluation. To check these results, you can comma separate x and y on the log like below.

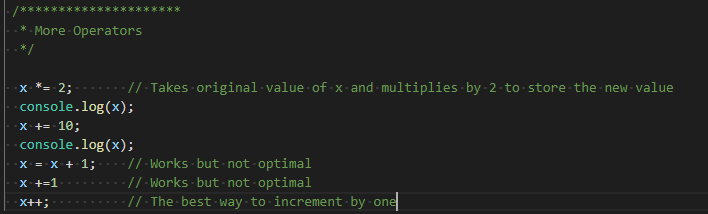


I am sure you are thinking shouldn't this be undefined since it is equaling a value that was never defined prior to this. Let's look at this a bit. Refer to the table for precedence there is a column called associativity which determines which direction the code is evaluated. Take for instance multiplication it goes left to right, which means it handles left values before the right so would take 2 + 4 + 5 and then have 6 + 5 which is 11. However, when you look at the assignment it handles right to left, which means that it is going to evaluate the right side of the equals first and then assign it to the left whether it is one variable or two like we have seen here. Based off both of those image that it takes that calculation and stores it into y and then y is stored into x after this has been completed.

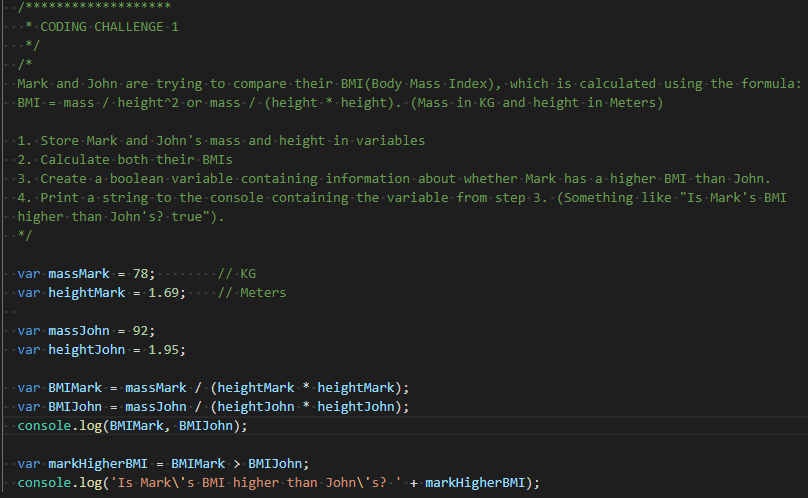
That brings us to the end of multiple assignments, so now let's shift our focus to some additional operators. Let's say we want to double x's value. You would probably be thinking x = x \* 2 which would get you there, but there is an easier way. You can use the \*= operator that is going to take the original value and multiply it by 2 and then store the new value. You can check the value if you want through the console.



The great thing about these is that you can use these for multiplication, addition, subtraction, division, etc. So, this is a very strong tool that you have at your disposal. Assume that you want to increment a number/variable by one then you can use the ++ operator. You can also use – which will decrement the value by one, which are both big operators you will find yourself using all the way throughout your career. Take for instance.

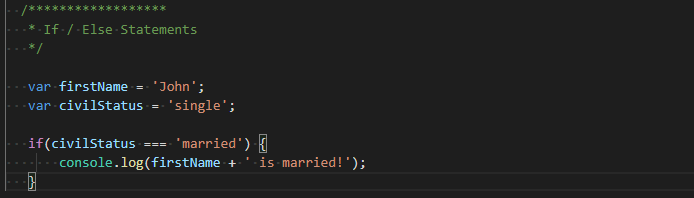


Which is going to bring you to the first exercise/coding challenge. Mark and John are trying to compare their BMI(Body Mass Index), which is calculated using the formula BMI = mass / height^2 or mass / (height \* height) Mass being in KG and height in meters. You need to store Mark and John's mass and height variables, calculate both of their BMIs, create a Boolean variable containing information about whether Mark has a higher BMI than John, and print a string to the console containing the variable from step 3(Something like “Is Mark's BMI higher than John's? True”). Below is a solution you can compare to. NOTE the escape key needed in strings with '. You must use \followed by asterisk.

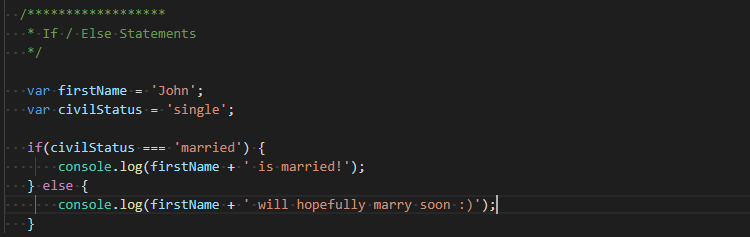


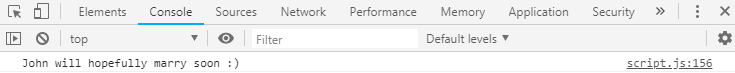
Which will bring us to the next topic which is going to involve the if/else statements. You may be wondering what they are, but you have already worked with these slightly without even realizing it. Almost every programming language out there has a control structure where certain code gets ran under specific conditions. This is where the if/else comes into play. It states that if this given condition has been met through a Boolean expression, then execute this code. Else, you would run the opposing code in the 'else' block of code.

Think about the exercise you just did where you stored Mark having the higher BMI, this could be used as a condition. I am sure you think this sounds confusing, so we are going to generate an example together. You need to create two variables. The first is going to be 'firstName' which is set to 'John' and then you need to create civilStatus which determines if a person is single, engaged, divorced, married, etc. For this instance, store 'single' into this variable. It is important to note in JS for equality checks you need to use '===' to evaluate if something is equal to a given value in an if statement. So, let's see if civilStatus is set to married.

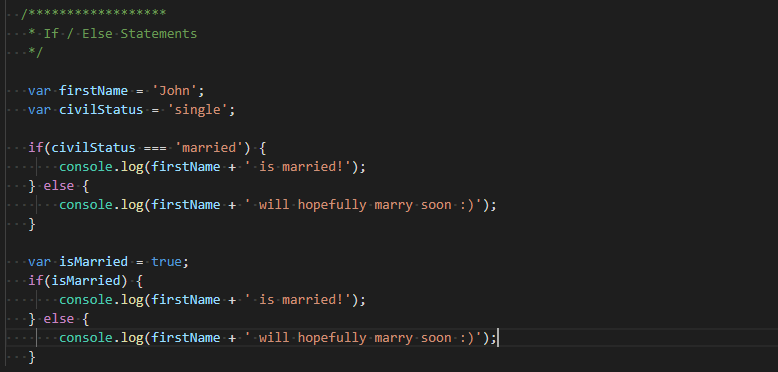


When you execute this though, there isn't any output!? That would be correct. Looking at the code above you will notice that our civil status was set to single, so the evaluation returned false and executed no code. Switch that value to be married and then run again, notice that you now see this being logged to the console. Now switch back to single, so that we can explore the else. This pretty much says if the first condition isn't true, execute this logging to the console like below.

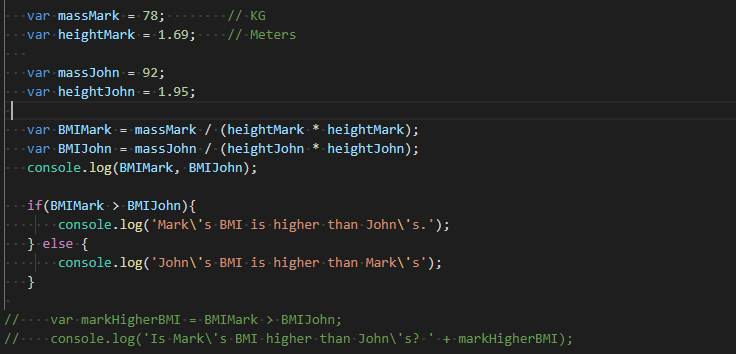




As stated, before you can create a variable that stores a Boolean expression within it that can be used in an if/else block of code. So, create a new variable isMarried and set it to true by default. Then simply take the same code that you had before and replace the condition with this variable like you see below.

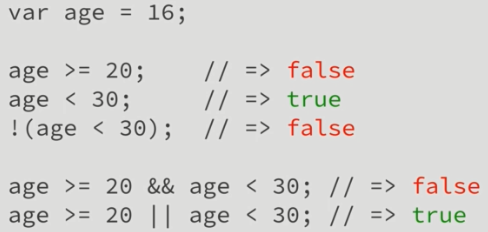
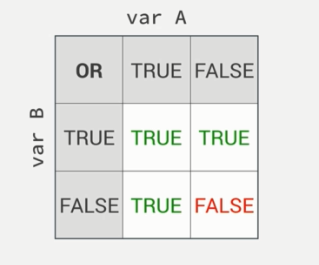
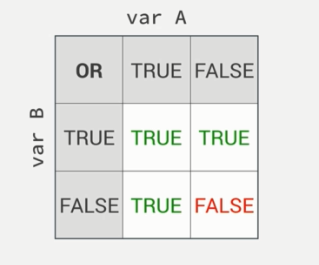
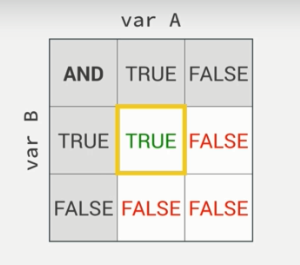


We could do limitless conditional checks throughout this and give plenty of examples, but we will instead go back to the exercise for the BMI. So, grab the code that you have from that as we are going to make modifications. First remove the last two lines of code where we set Mark having a higher BMI and the writing to the console. Now add an if/else block where it checks if Mark's BMI is greater than John's. If it is then it is written to the console 'Mark's BMI is higher than John's', otherwise John's BMI is higher than Mark's. Do not forget the escape characters. If you want, you can also change the values so that John's would be higher than Mark's.

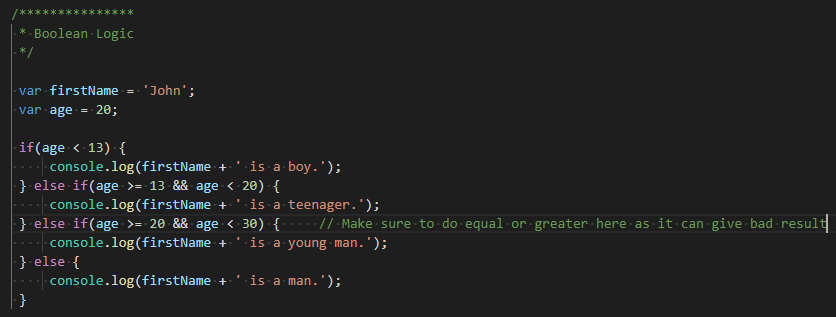
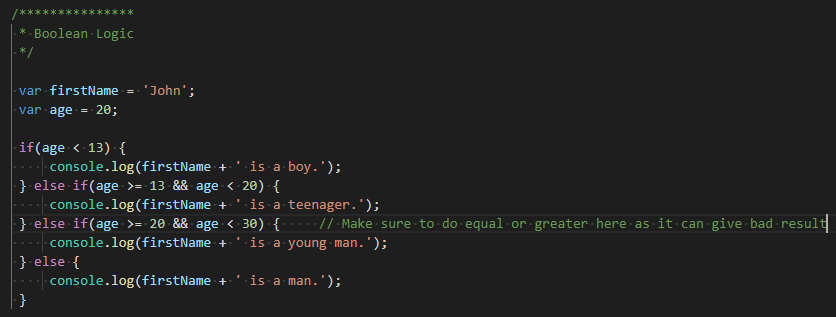


So now let's take conditions to the next level! We are going to accomplish this by learning Boolean logic. In this example we want to determine if John is a boy, teenager, young man or a man based off his age. First define firstName to be John and age to be 16 to start off. Next set up your condition to see if his age is under 13, if so, print John is a boy. We could set the else to be John is a man, but then that doesn't fulfill our other possibilities.

Which brings us to introducing the 'else if', which can be used after the if and in between the else as many times as necessary to check all conditions. First though lets go over AND(&&), OR(||) and NOT(!). When using AND both conditions must be true for the condition to evaluate to true. Next, the OR checks to see if one condition or the other is true(can be multiple conditions and not just two; same with AND). Then you have the NOT which inverts the condition that is placed evaluating the opposite to check for True or False. Check below for a visual representation of these.



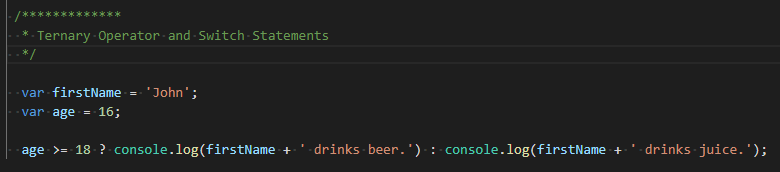
They require you to actually have a condition so let's go with checking to see if he is between 13 and 20 years old. To properly check for this there would be two conditions. The first is going to check if the age is greater than or equal to 13 and the other is going to check that the age is less than 20, if condition is met then John is a teenager. For the other we need to check that the age is greater or equal to 20 but less than 30 and update the age to 20.



# Ternary Operator/Switch Statements

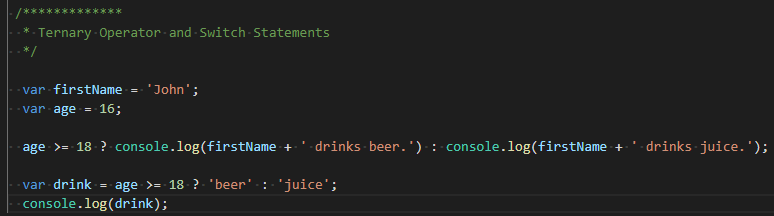
We will now be shifting gears a little bit from the different ways to check conditions to introduce the ternary(conditional operator) and switch statements. All a ternary operator is a shorthanded way of handling an If/Else code block with one statement.

Let’s get to work on this. First create a variable for first name and set to john, as well as a variable for age which will be set to 16. We are going to be checking to see if John can drink alcohol or only a juice. First you will check to see if age is greater or equal to 18, this will represent the condition we are checking. Following the condition, you will use the ‘?’ symbol which is followed by the return if the condition is true. Your first condition will log to the console the person’s name plus ‘ drinks beer.’. You will now need to use the ‘:’ which separates the first return result(true) followed by writing to the console that the person drinks juice.



If you are wondering why it is called ternary this is due to the fact that there are three operators that are being used within this statement, which are the ‘>=, ?, :’. The first possible return would be your if and the second would be your else counterpart. You can play around with the ages a little bit if you want to see how the condition evaluates.

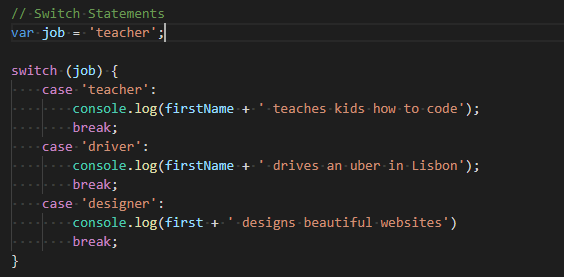
We will now explore another possible way that we can make the console log more dynamic. We are going to do this by create a variable called drink. We are then going to assign a ternary operator to this. We will do the same conditional check, except this time for true you will have ‘beer’ and for false you will set the value to ‘juice’. To check this, write to the console and pass the drink variable that you just created.



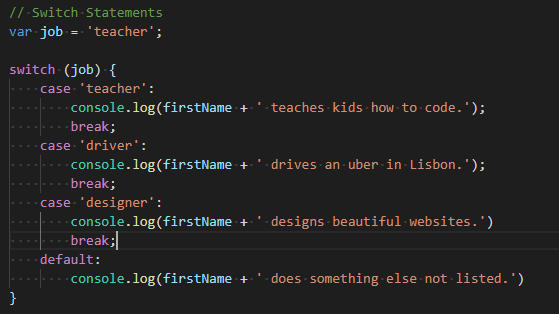
We will now look at everything that you would have to do to make this work without the use of the ternary operator. By doing this, you will be able to see why/how these are useful and can save you a lot of time, as well as clean up your code. Create an if statement that checks if the age is greater than or equal to 18, if so set var drink to ‘beer’. Otherwise, create the else block that gets set to ‘juice. Which do you think looks/feels better?



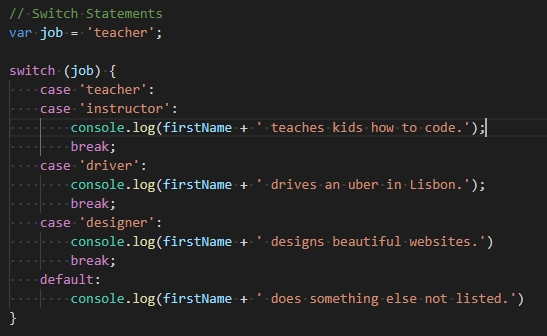
You now know about the power of a ternary operator, so we are going to shift gears towards the switch statement. You will need to create a variable job called job that is going to be set to ‘teacher’. You will then need to call the switch(), where you will pass in the variable here. Within the body{} of this you will have a case ‘teacher’:. Inside of the body of the case you will write to the console the person’s name followed by ‘ teaches kids how to code’. For our second case you will have driver, which will write out the person’s name followed by ‘ drives an uber in Lisbon’. Lastly, create a third for designer which will write the person’s name followed by ‘ designs beautiful websites’. Each case will need to be followed by a break statement so that way the switch has a way to escape from execution as we have found the evaluation that is expected.



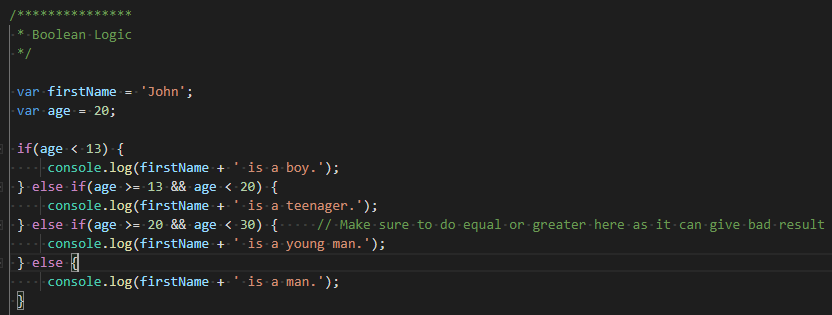
What happens though if the value that is being evaluated doesn’t match any of the three above? This is where the default comes into play. You will use this to handle this specific issue with not evaluating true for any of the conditions. To write them they are handle the same way as the cases, except you will use the ‘default’ keyword instead. For your default have the name written to the console followed by ‘ does something else not listed’. You can think of the switch case as a big nested If/Else/ElseIf structure, but without all of the additional syntax.

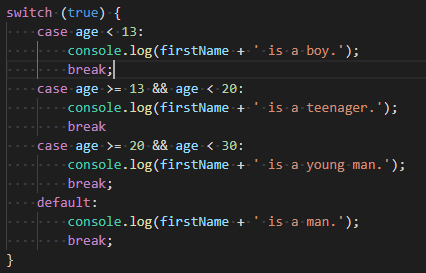


You should experiment a little bit with the values so that you can see all of these in action. Will help you with the understanding of how they operate and output data. Next, I am going to introduce you to the adding an extra case that has the same output. Let’s say that we have an instructor job as well that teaches kids how to code. You would simply add a direct case below the one in question like you see below.



For the next part of this we are going to refactor your Boolean Logic code into a switch statement, so retrieve the code from above. Let’s start off with switch(true), which forces this evaluation to take place. First add your case for age < 13 and just copy the console execution, followed by a break. Repeat these steps for the additional conditions that can take place until you have constructed the whole set.



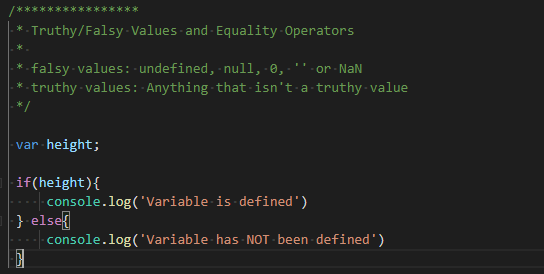


You can play around the with values a bit to try to get the different edge cases to execute their logging. Another thing you can do is take these conditions we are checking and run the evaluations on the Boolean variable that they are assigned to. In the end, this is more of a preference choice that you will need to make as you dig deeper into your career. Finding the most practical usages and what feels the most natural. Another thing to note is that you can also pass a Boolean variable directly into the switch like just like we did with the If/Else section we covered. When you are comfortable with working with ternary and switches, feel free to move onto the next section.

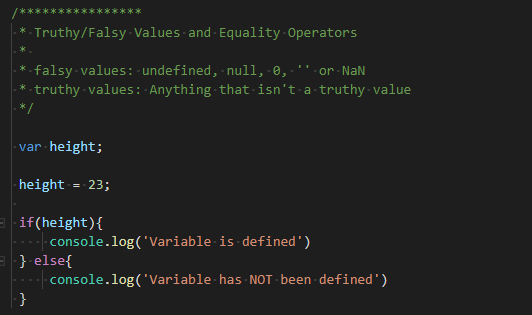
# Truthy/Falsy Values and Equality Operators

In this section we are going to be starting to learn about truthy and falsy values. For falsy these are values are determined based off a false evaluation. These can be the following values: undefined, null, 0, ‘’ or NaN. What does this mean? This means that when these values are returned in an if/else evaluation they are returned as false. They are not necessarily false but will be converted to a false value when ran in a conditional check.

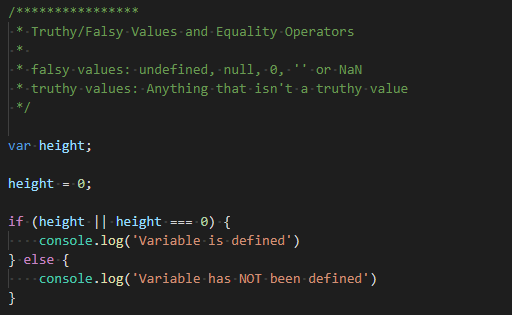
In addition to the falsy, we also have its counterpart truthy values. These are going to be any other value that is used that doesn’t convert to falsy values. With instances like this, they will return true instead of false like with the other values we discussed. We will now create an example of this. You are going to first need a variable ‘height’. You will now use that variable in an if check. If the condition returns true, then write to the console that the ‘Variable is defined’. If this evaluates false, you are going to want to write to the console ‘Variable has not been defined’.



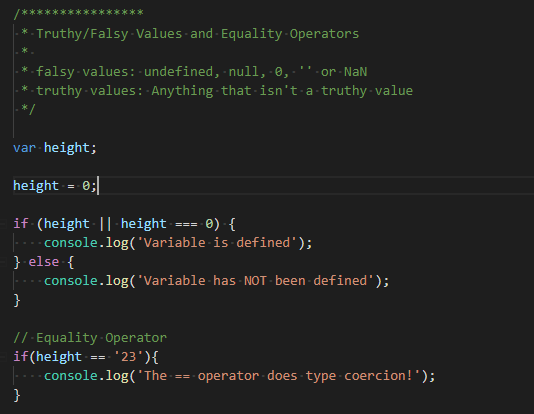
When you go to run the page, you are going to notice that the value hasn’t been defined yet. Why is this happening, I declared the value though? The reason this happens is because there hasn’t been a value that has been assigned to the variable. Which means your next step is to add a value to height, set this to ‘23’. When you run the page this time you will notice that the variable is no longer undefined.



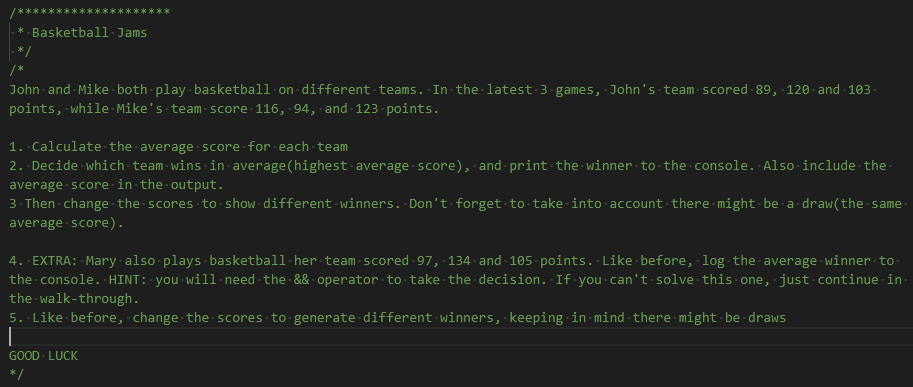
In the event you ever want to test if values are initializing you can use something like this to test against it. An important thing to note is that you can set this value to 0 and it will trigger the else block of code. What if I want the value to be able to be zero? You will have to add an additional OR(||) condition to be checked against. This condition will check to see if height is equal to 0. Using this statement says that the that if I have a value or if height is equal to zero. Only one of these conditions must evaluate true. This is a very common pattern that is used to check to see if values are defined.



So far, we have covered the ‘===’ operator, but we haven’t quite touched on the ‘==’ operator. The new operator will do type coercion. What is type coercion though? It is the conversion of one type of object to a different object type that has similar content. Let’s look at an example of this. Change your height value to be 0 again. You are then going to do a condition that checks if height is equal to ‘23’ using the ‘==’ operator. If this condition is met, then write to the console ‘The == operator does type coercion!’. What you are doing here is taking a string object and converting it to an integer value to be compared against the integer variable height. You must do this because if you used === you would return a false value, because it can’t compare a string and an integer.

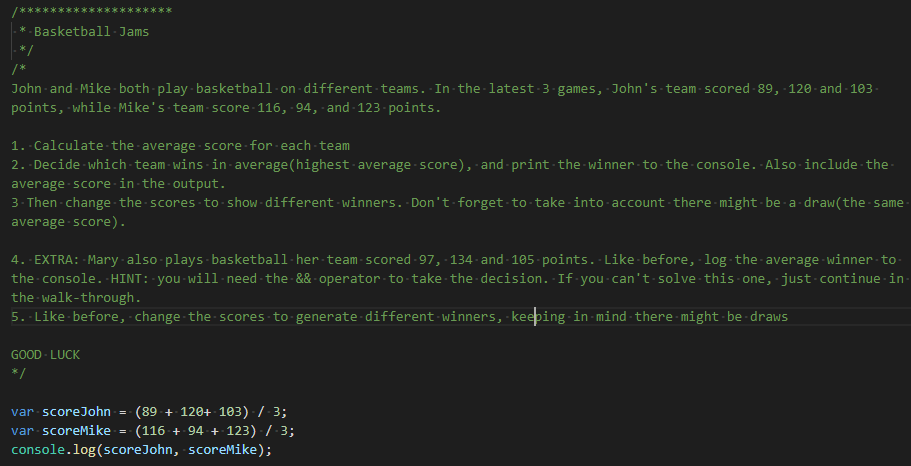


# Basketball Jams Coding Challenge

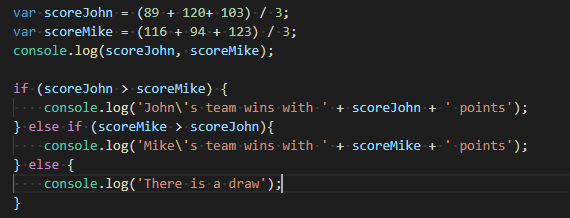


# Basketball Jams Solution

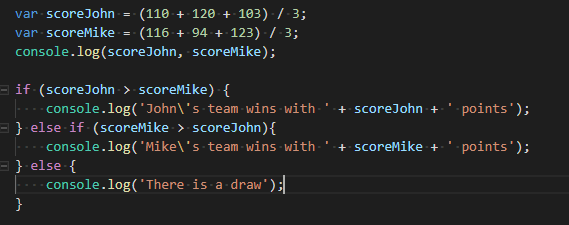
We are going to begin with populating the average score. You will need two different variables to stand for both players Mike and John. I used scoreJohn and scoreMike. You are going to need add the three game’s score within parenthesis and divide that by 3. You will repeat this for both variables. Now write to the console the two scores.



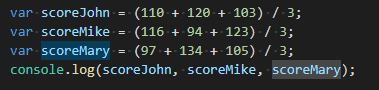
We now have both of our averages calculated and stored. Moving on we are going to first check if John’s score is higher in the if statement. If this evaluates to true, then console ‘John’s team wins with “score” points’. You are going to need a second condition which checks the opposite, so do the exact same thing for the reversed scenario. Great! However, you still are going to need your else which would mean that neither had a higher average and it was a draw.



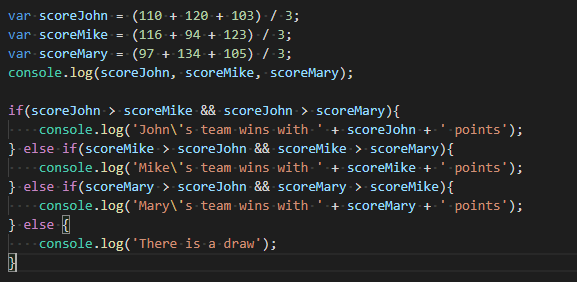
Of course, you are only going to have one winner with this equation, which is Mike’s team with 111-point average. I recommend changing the values around a bit so you can test the other conditions to ensure they work properly. Below I have a screenshot of how to get a draw without changing very many values.



Now things are going to get a little bit interesting by adding Mary into the game as well. Make sure to revert the score back to the original. You are going to do the same thing with the variable for storing her team’s average. You will also need to add the variable to the first instance you wrote to the console. You are going to notice when you run this that Mary has the highest average, but we are not done yet.



Begin by commenting out the if/elseif/else statement as we are going to start from scratch. Let’s first figure out how John wins. You are going to need his score to higher than Mike’s AND(&&) Mary’s. Next, we are going to be figuring how to calculate for Mike the exact same way, followed by doing the same for Mary. Do not forget to make sure that you check for the draw(add 3 to Mike’s score).

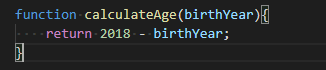


I would recommend playing around with the values some so you can see all your edge cases in action.

# Functions

This will now bring us to one of the most fundamental concepts of JavaScript. In JS you will use functions that are meant to be used numerous times or achieve some specific functionality that you need throughout your applications. You can think of them as containers that you are able to send parameters(arguments) to, that will then return a result.

To create one, you will have to use the ‘function’ keyword followed by the name you want to give this function. Generally, you should pick a meaningful name and should follow the camel case naming convention(myFunction – each first letter capitalized). As mention you can pass parameters or arguments to these functions, which in this case will be ‘birthYear’. It is important to note that you can either pass no arguments, one or many arguments to the function. Simply for the body you are going to return the current year(2018) in this instance and subtract the birthYear that was passed in through the parameter.



We have created the function, but we have yet to still actually use this. In the current state of the function it is simply just wanting a value, so we are going to pass ‘1985’ to represent my birth year. With function you can either simply just call them or you can store them into a variable to be used at your convenience.