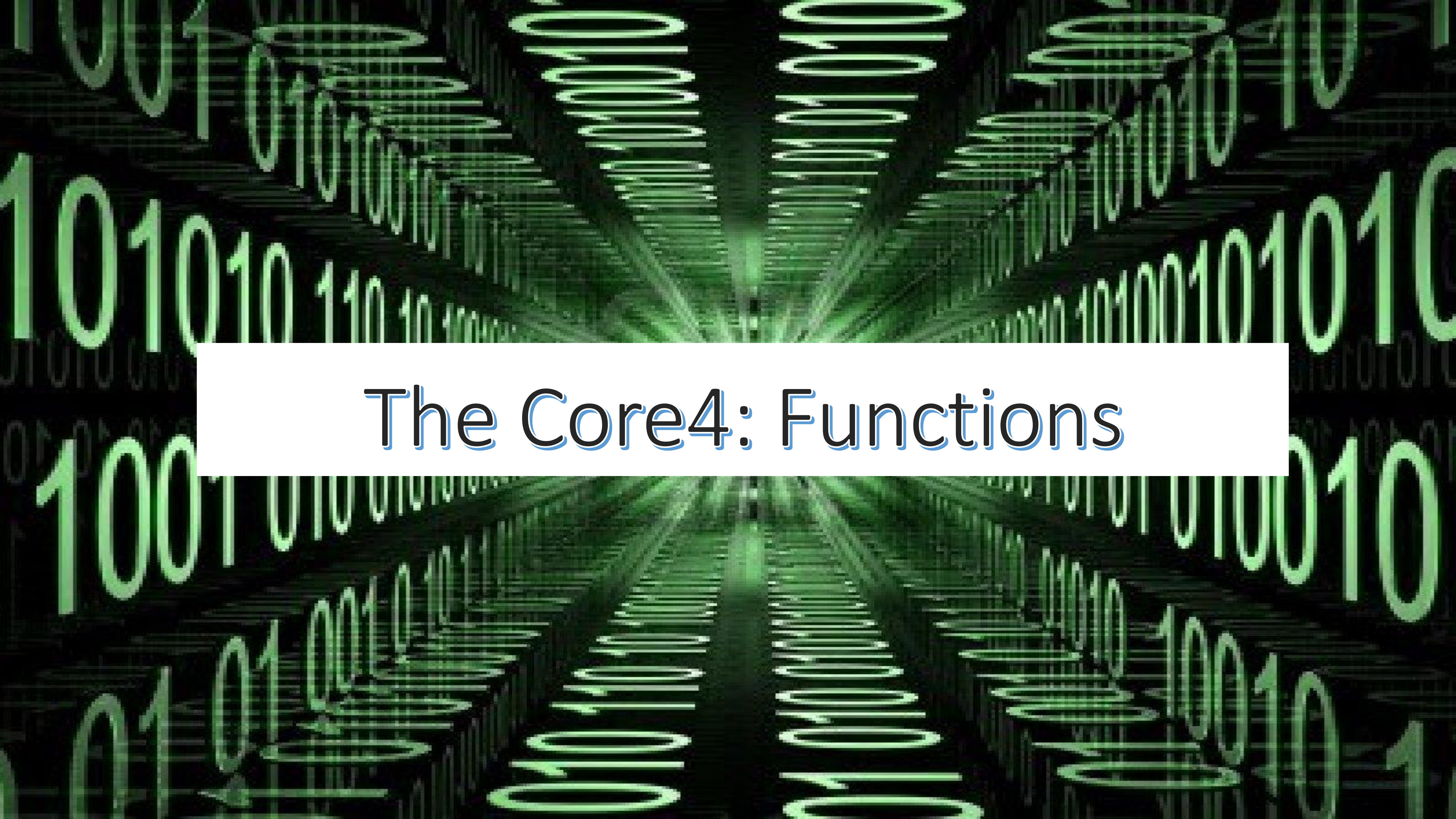
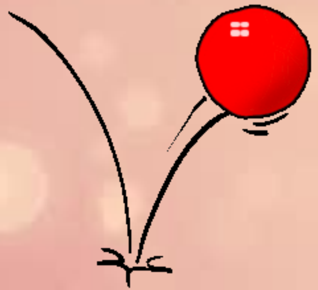


The Core4: Functions

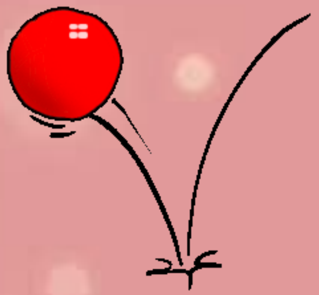
What are functions and why are they important?

- Think of a computer program as a factory and a “function” as a factory worker.
  - Workers have one small job that they are in charge of performing.
  - There can be many workers or only a few depending on how large the factory is.
  - Workers are in charge of the success of the factory and the overall operations.

Functions in a computer program work the same way! Functions have the purpose of distributing work to encourage organization and help make code bug-free.



How do functions act in a program?



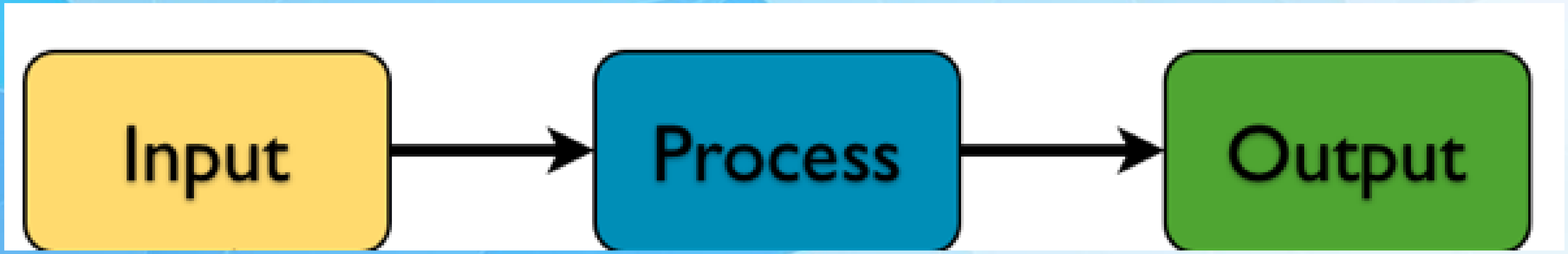
Example:

- A program running a game of Pinball might have the following functions:
  - **Function #1**: Does the math calculations to bounce the ball.
  - **Function #2:** Colors the appropriate tiles that the ball touches.
  - **Function #3:** Adds the points the player has won.
- Functions are useful for narrowing down mistakes in your program.
- If you made a Pinball game with the functions above and the game ***does not*** *show the ball bouncing correctly*, which function above might the problem be in?

How Functions Work

- Factory workers are usually told what to do, how to do it, and how to report their progress. Functions act the same way, requiring multiple (or no forms of input), an order of steps on how to perform a task, and the result of the task.

Layout of a Function that detects numbers greater than 10



Insert a number into the above function

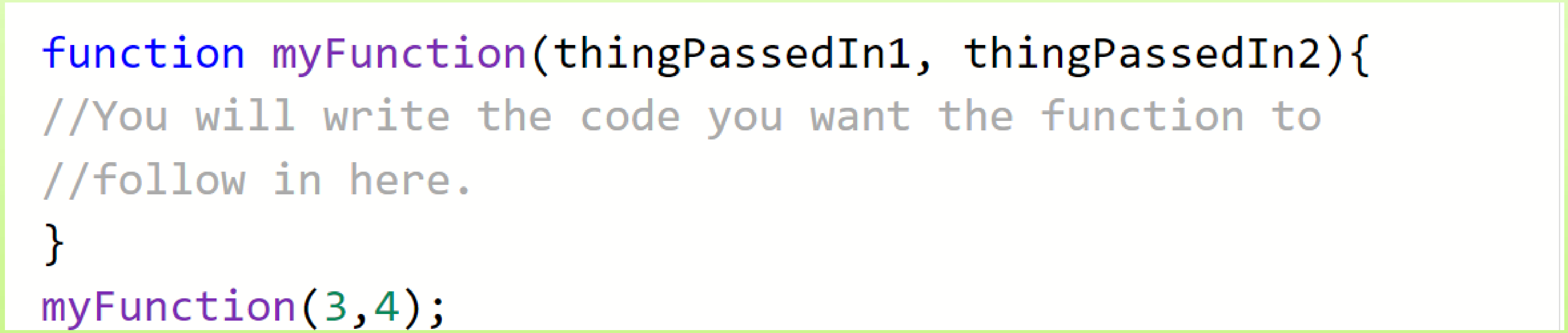
Compare the number to see if it is greater than 10

“The Number entered was greater than 10”

Features of a Function

- Functions include variables and conditionals (**Remember: conditionals** are **“if”, “else if”, and “else” statements**). A function uses the information and instructions provided to it to complete the task.
- Functions are also useful because they are re-usable! Instead of writing the same code over again, you can simply call the function.
- If you had a website that greeted every guest by time of day and name, would you rather write a thousand programs for each possible name in the world? Or a single function that could handle ANY name that is passed in?

A Function’s Anatomy in Javascript



```
function myFunction(thingPassedIn1, thingPassedIn2){
//You will write the code you want the function to
//follow in here.
}
myFunction(3,4);
```

Things to note:

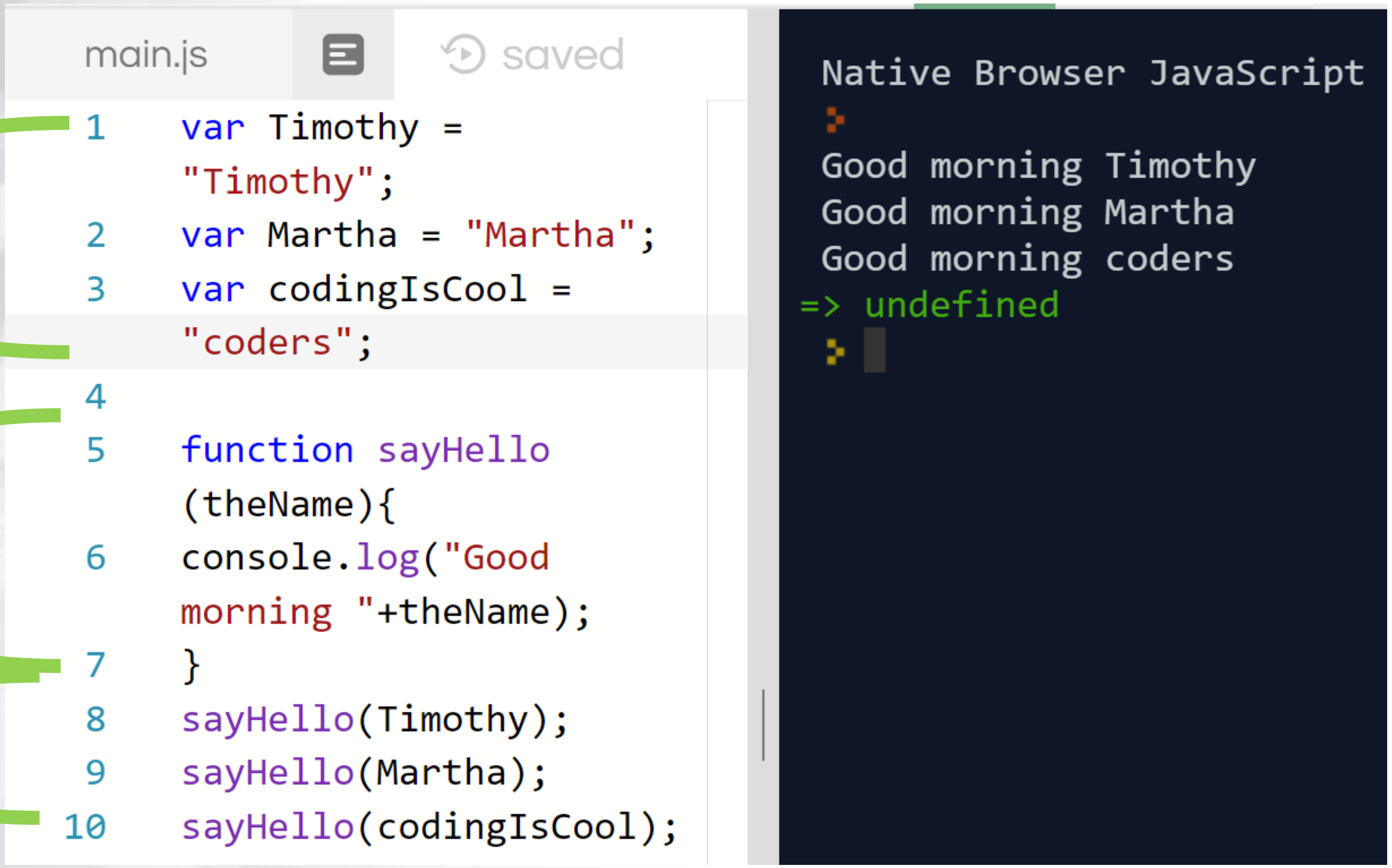
- A function in Javascript starts with the keyword “function”, a space, the name you want the function to be, and optionally variables (Variables passed into functions are called parameters).
- Write the instructions you want the function to follow inside the brackets.
- When you call a function, type its name EXACTLY. Follow it with a pair of parenthesis, variables passed in (if any) inside the parenthesis, and a semi colon.

The Function Process

Variables are created

A function called sayHello is created. It asks for a variable and prints to the console using the variable given along with a greeting

The function sayHello is called. It tells program to send the variable “Timothy” into the function sayHello so that it can follow the code inside of sayHello.



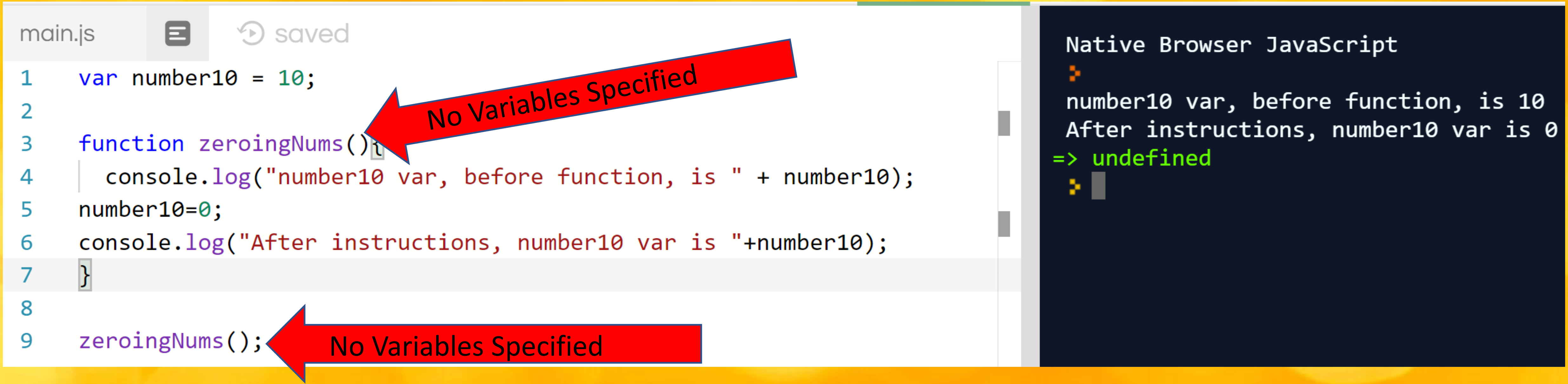
main.js saved

```
var Timothy = "Timothy";
var Martha = "Martha";
var codingIsCool = "coders";

function sayHello (theName){
console.log("Good morning "+theName);
}
sayHello(Timothy);
sayHello(Martha);
sayHello(codingIsCool);
```

```
Native Browser JavaScript

Good morning Timothy
Good morning Martha
Good morning coders
=> undefined
```

Functions without Parameters

- Functions do not need to have information passed in to complete a task.
- The following function is simply in charge of zeroing out a number. Notice it doesn't have variables passed into the function or when it is called on the last line of code.



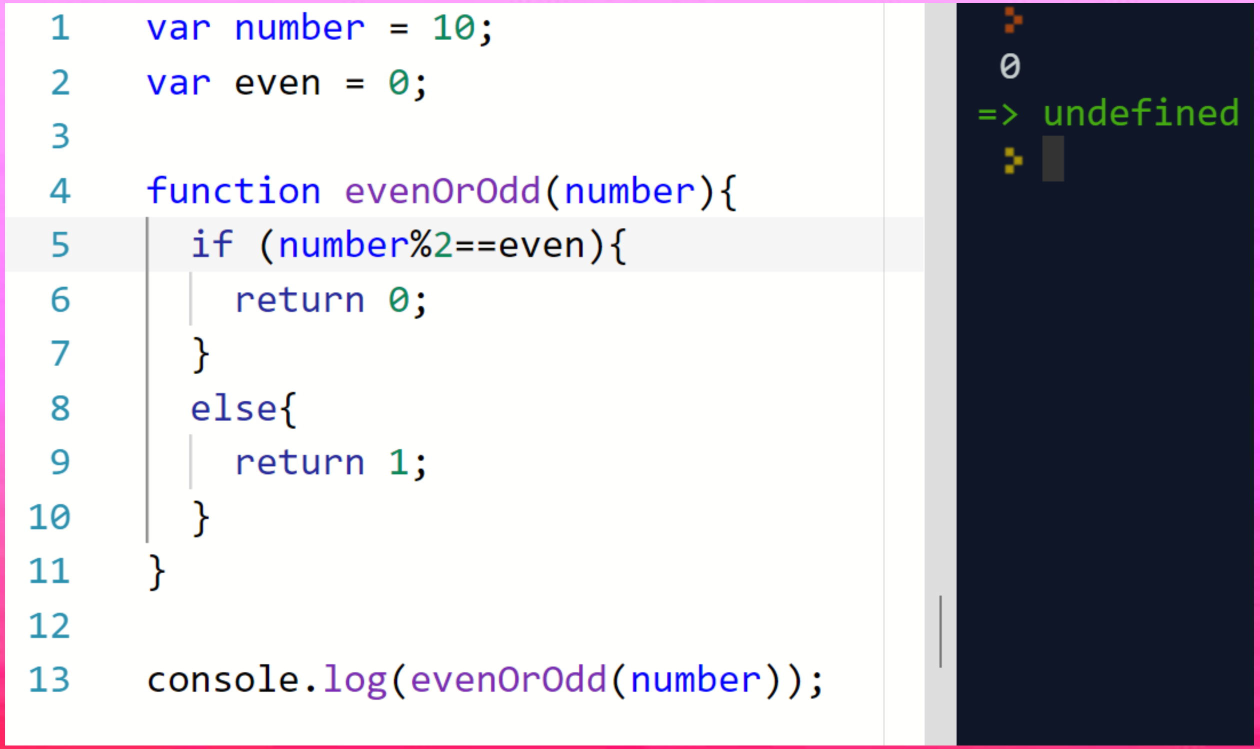
```
var number10 = 10;

function zeroingNums(){
  console.log("number10 var, before function, is " + number10);
number10=0;
console.log("After instructions, number10 var is "+number10);
}

zeroingNums();
```

```
Native Browser JavaScript
›
number10 var, before function, is 10
After instructions, number10 var is 0
=> undefined
›
```

The “Return” Keyword

- “Return” is a keyword used by a function. It says “If I am called in the program, *I will return a variable depending on the conditionals’*.”
- Example use: You want to find out if a number is even or not. After figuring out if a number is even or odd, the function can return 1 for odd and 0 for even to signal the outcome.

Note: **%** is a symbol meaning “give me the remainder of the first number divided by second number.” Even numbers can be divided into 2 with no remainder.

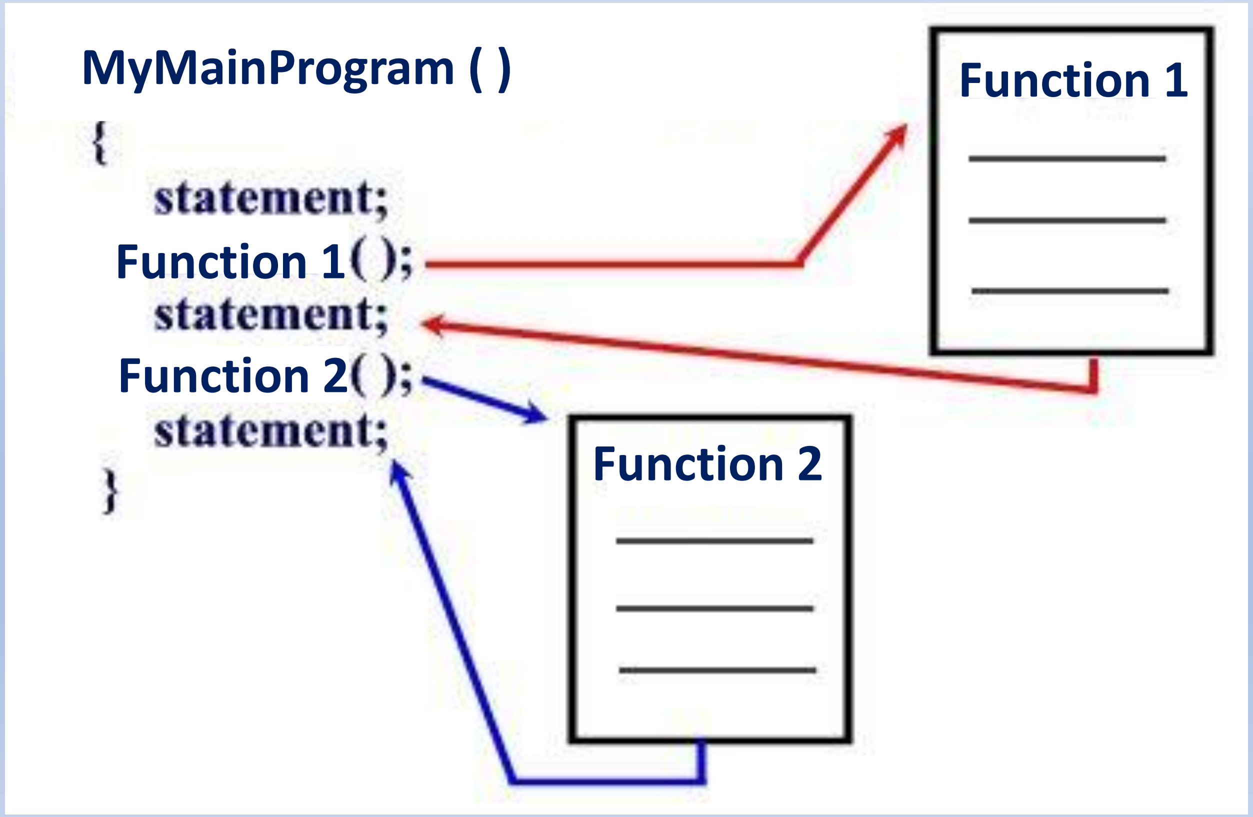


```javascript
var number = 10;
var even = 0;

function evenOrOdd(number){
  if (number%2==even){
    return 0;
  }
  else{
    return 1;
  }
}

console.log(evenOrOdd(number));
```

```
0
=> undefined
```

Multiple Function Calls

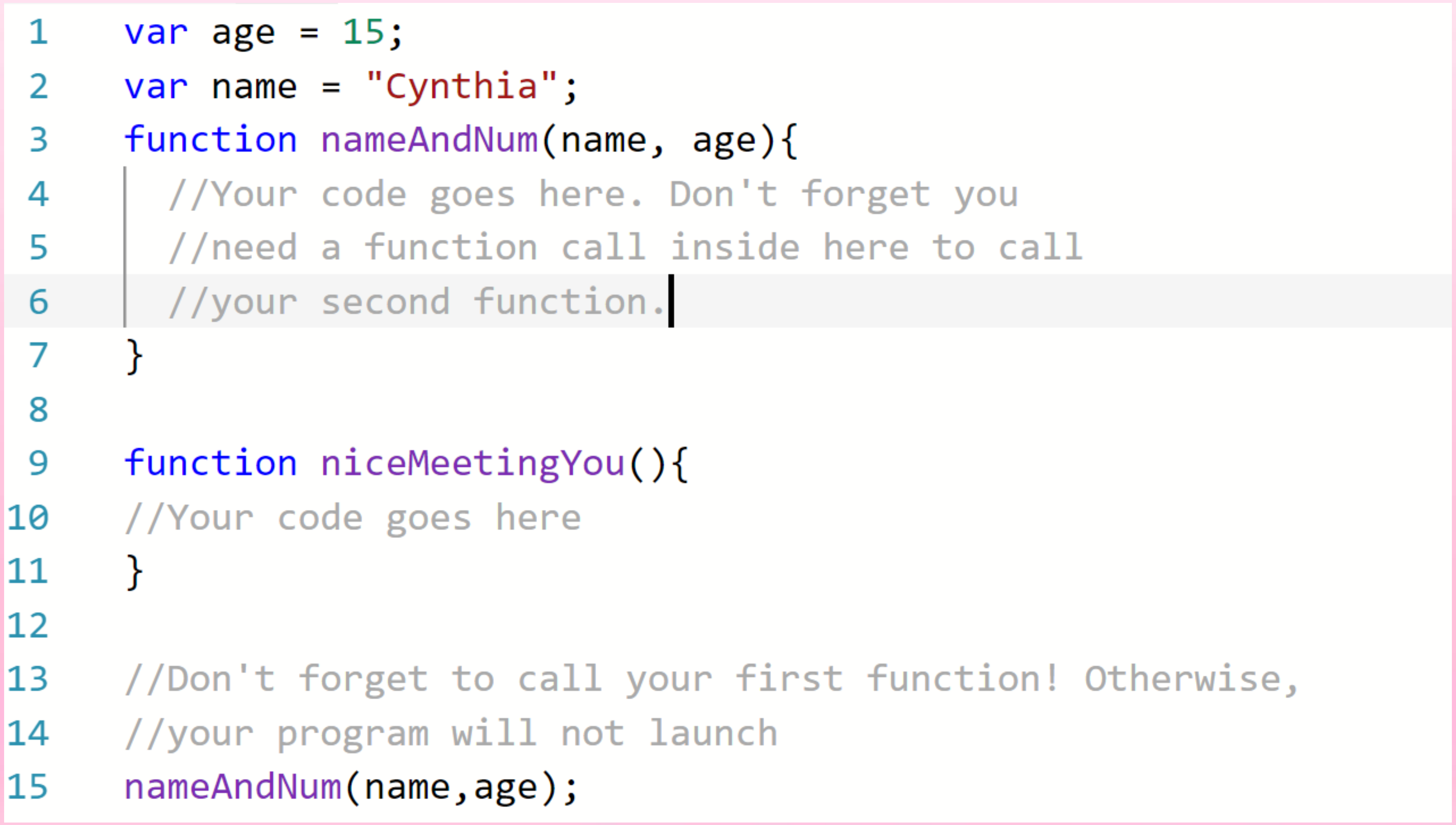
- One function can call another function!
- Every call in the main program is followed in order. So first statement gets executed. Then **ALL** of the code in function 1 is executed since it is called next. Then the statement after function 1 is executed. And so on.



The core4: Functions exercise!

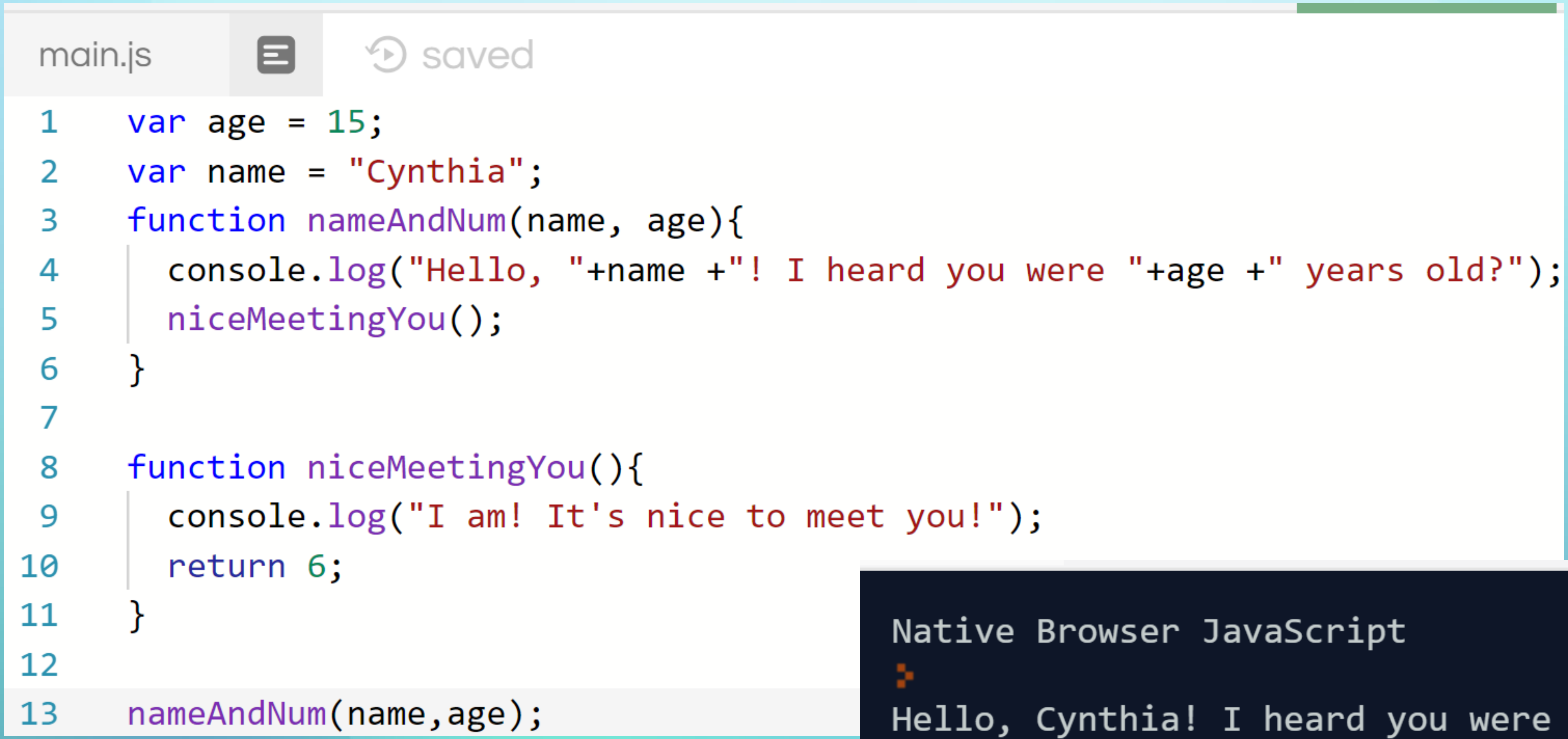
- Create two functions
  - Make a function (with any name you want) that takes in two variables: a name and an age. Have the function print out: “Hello, [name]! I heard you were [your age] years old?”
  - Make a second function that does not take in any variables. Have this function simply print out “I am! It’s nice to meet you!”
    - Have it return the number 6.
  - Have the first function you create call the second function.
  - Explore trying different names and numbers into your first function to see how the same code is flexible for any input!

Starting Code



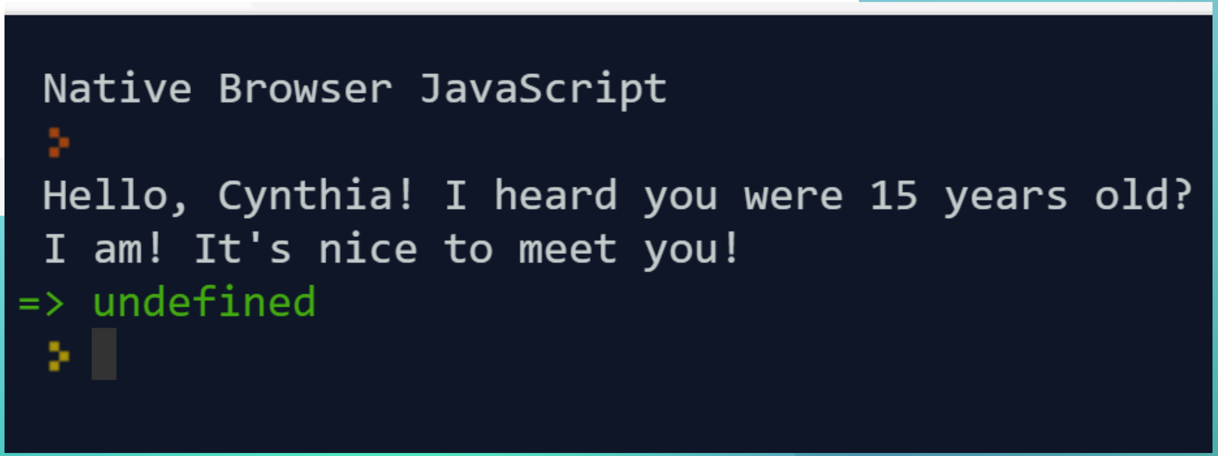
```
var age = 15;
var name = "Cynthia";
function nameAndNum(name, age){
  //Your code goes here. Don't forget you
  //need a function call inside here to call
  //your second function.
}

function niceMeetingYou(){
//Your code goes here
}

//Don't forget to call your first function! Otherwise,
//your program will not launch
nameAndNum(name,age);
```

One Possible Solution



main.js saved

```
var age = 15;
var name = "Cynthia";
function nameAndNum(name, age){
  console.log("Hello, "+name +"! I heard you were "+age +" years old?");
  niceMeetingYou();
}

function niceMeetingYou(){
  console.log("I am! It's nice to meet you!");
  return 6;
}

nameAndNum(name,age);
```



```
Native Browser JavaScript
›
Hello, Cynthia! I heard you were 15 years old?
I am! It's nice to meet you!
=> undefined
›
```

Quick Reflection

- Were there any coding problems that you ran into today? How did you solve them?
- What are some tips you can give your classmates when coding solutions?
- What is something you learned in this lesson that might be useful next time?

Console.log(“Have an amazing week!!”);