Functions

Objectives

- Understand functions and why they are used
- Understand how functions can be used to promote code reuse within your application
- Introduce Anonymous functions and understand how they are different than named functions

Agenda

- Discuss Functions
- Code Along: Geometry Formulas using Functions
- Code Along: Cash Register
- Anonymous Functions
- Code Along: Cash Register using an Anonymous Function
- Lab: HiLo

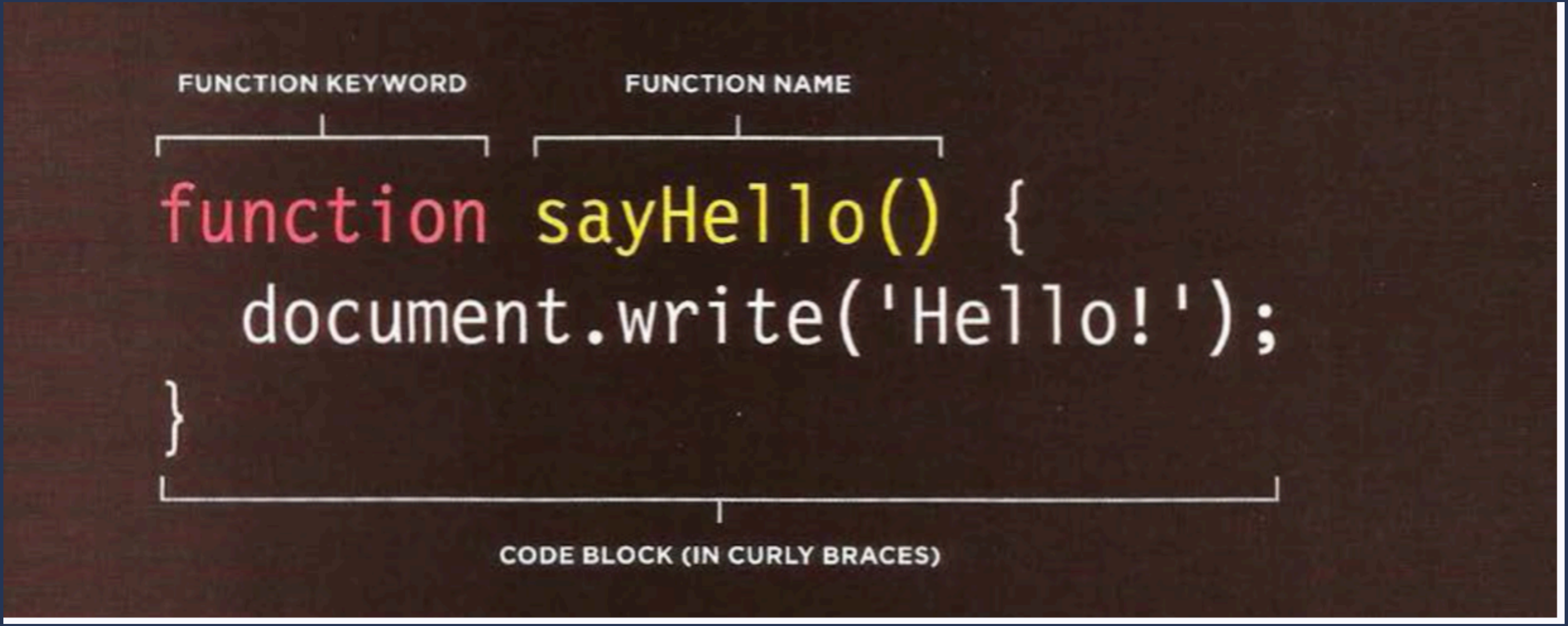
What are Functions?

- Allows you to group a series of statements together to perform a specific task
- Functions are used to promote “code reuse”
- You can control when functions are executed, for example - you can write functions that only get executed (or called) when a user clicks a specific button
- Many times you will write a function and expect a value to be returned - this is called a return value

Simple Function

```
// a simple function that greets you with 'Good Morning'

// 1) Declare a function named greeting

function greeting(){
  alert('Good Morning');
};


// 2) Call (or run) the function

greeting();
```

Declaring a Function

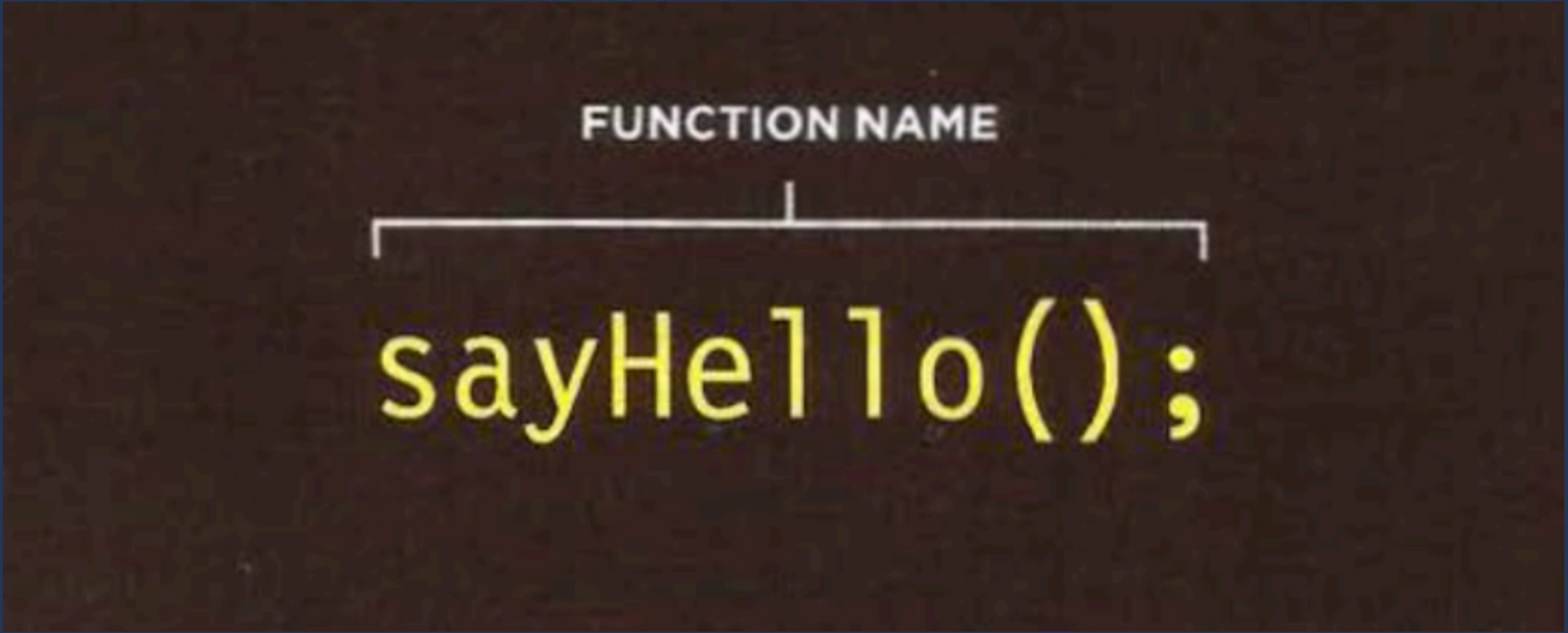


```
function sayHello() {
  document.write('Hello!');
}
```

Declaring a Function

- Use the “function” keyword to declare a function
- Functions can be given a name (see example on next page)
- The name must be followed by parentheses
- The opening and closing curly braces indicate a “code block”
- The statements for your function goes within the code block
- Simply declaring the function will not run this code, this function must be “called” in order for the code inside the function to be run

Calling a Function

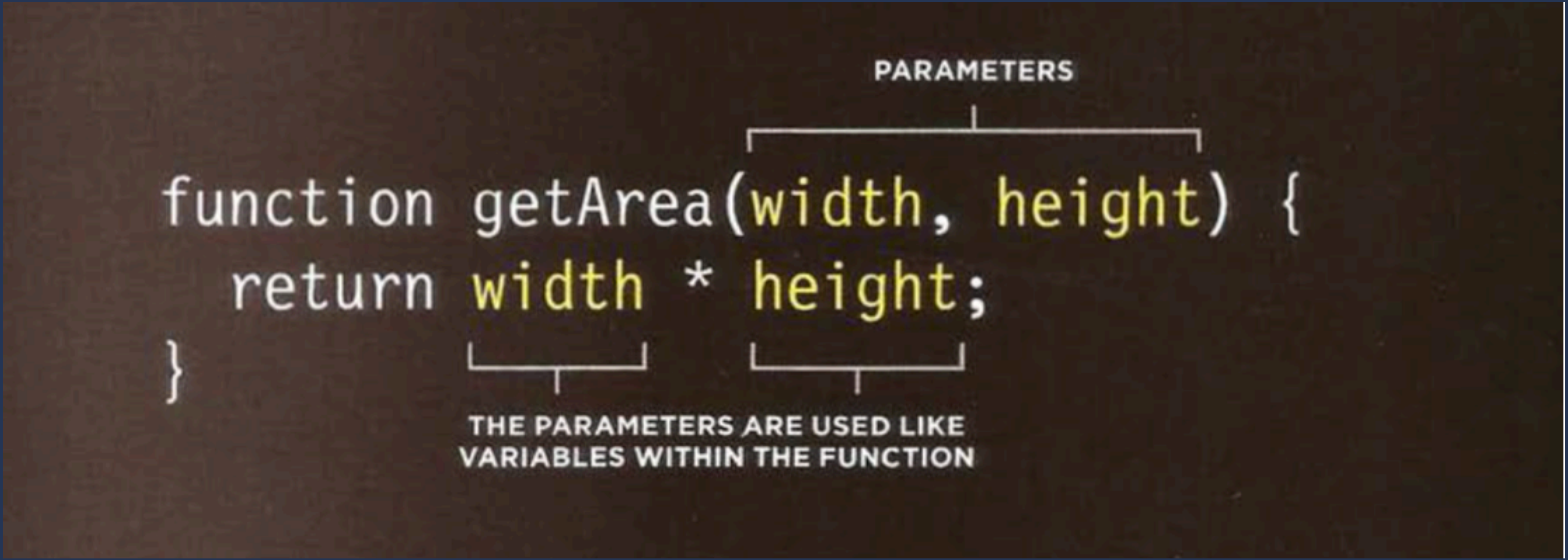


```
sayHello();
```

Calling a Function

- To run the code inside of a function you use the function name followed by parentheses (don't forget the parentheses!)
- Now you can call this function as many times as you want

Declaring Functions that need information



```
function getArea(width, height) {
  return width * height;
}
```

Declaring Functions that need information

- Some functions need additional information in order to perform a specific task
- This additional information is referred to as “parameters”
- To provide parameters to a function, you specify them inside the parentheses after the parameter name
- The parameters are used like variables within the function body
- We use the “return” keyword when we want to retrieve a value from our function, in the case of the example we want to retrieve the result of the multiplying the width times the height

Calling Functions that need information

```
// Calling the getArea() function with values

getArea(7, 5); // returns 35


// Calling the getArea() function with variables

var doorWidth = 2;
var doorHeight = 8;

getArea(doorWidth, doorHeight); // returns 16
```

Functions can call other functions

```
// Function that calculates area of a square

function areaOfSquare(side){
  return side * side;
};

areaOfSquare(3); // returns 9

// Calculates surface area of a cube and *reuses* areaOfSquare function

function surfaceAreaOfCube(side){
  return 6 * areaOfSquare(side);
};

surfaceAreaOfCube(7); // returns 42
```

Functions can call other functions

- Function reusability is key results in cleaner code
- Reusing functions leverages key programming principle - Don't Repeat Yourself

Code Along: Geometry Formulas with Functions

Preventing Default Behavior

- Some events, such as clicking on links and **submitting forms**, take the user to another page or expects data to sent to a server
- There are times when you don't want that default behavior to happen
- Javascript gives you the ability to prevent the default behavior using preventDefault()
- For more information on preventDefault click here

Preventing Default Behavior

```
// All javascript functions get an 'event' object as a parameter by default, most of the time
// you can ignore this, but you will need it if you want to prevent default behavior

// Define a function when
function handleFormSubmision(event){
  event.preventDefault();

  // the rest of your code goes here

}
```

Code Along: Cash Register

Anonymous Functions

- Functions that do not have a name
- Used extensively in jQuery

Anonymous Functions

```
// An anonymous function is a function without a name

function(width, height){
  return width * height;
};


// this function can later be stored as a variable and used later in your code

var area = function(width, height){
  return width * height;
};

// Call anonymous function stored as a variable

area(4, 5); // returns 20
```

When should you use Anonymous Functions?

- Use for code that only needs to run once within a task
- Use as event handlers and listeners to perform a task when an event occurs

Using an anonymous function as an event handler/listener

```
// Uses a *named function* to serve as a event handler (function that is called as a result of an event)

$('#blueButton').click(blueEffect);

function blueEffect(){
  $('body').css('background-color', 'blue');
};


// Results are same as above example, but instead we use an anonymous function as a click handler
// This is the common jQuery pattern

$('#blueButton').click(function(){
  $('body').css('background-color', 'blue');
});
```

Code Along: Cash Register with Anonymous Function

Lab: HiLo

- Description: Create a game that has users try to guess a secrect number from 1 to 100
- In groups of four review the requirements and write pseudo code for the application
- What data do you need to keep track of? Should you store these in variables?
- What specific tasks will your application need to carry out (these could be your functions)?
- What user events will you application need to listen for?

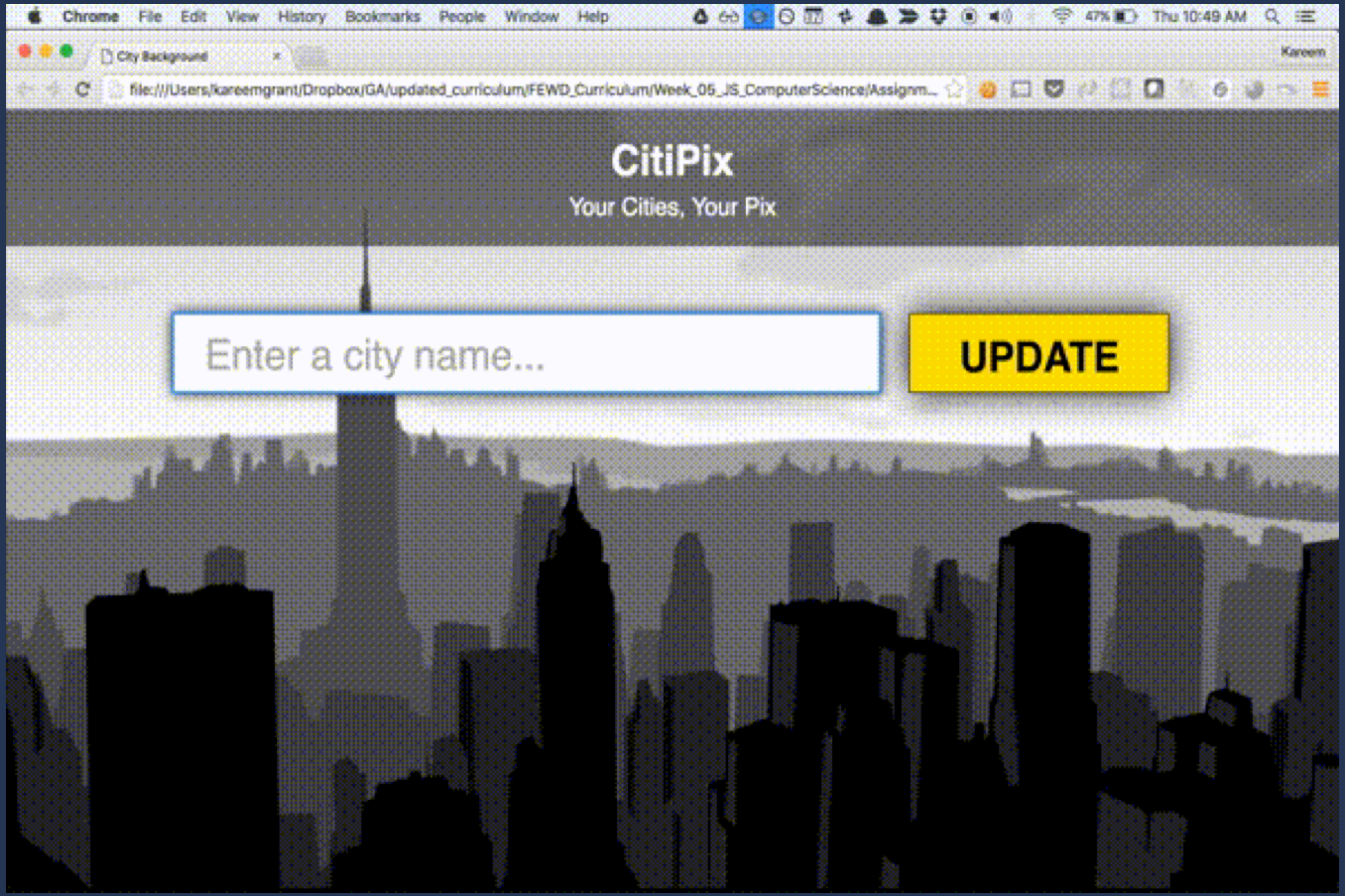
Lab: HiLo - Requirements

- Users will be given 5 guesses
- If a user enters a guess and it is **incorrect** the following should happen:
  - A notification will be displayed informing the user that their guess was either too loo or too high
  - The number of guesses remaining count will be decremented by 1
- If a user enters a guess and it is **correct** the following should happen:
  - A notification will be displayed informing the user that their guess was correct
  - The number of guesses remaining count will be reset back to 5
- If a user runs out of guesses (5) without correctly guessing the secret number the following should happen:
  - A notification will be displayed informing they user that they ran out of guesses and the game was over
- When users click on the reset button the following should happen:
  - The number of guesses remaining should be reset to 5
  - The message should be hidden

Lab: HiLo - Bonus Requirements

- At the start of each game the app should randomly generate the secret number (it should be a whole number)
- Apply the "success" class to the element containing the notification message when the user correctly guesses the secret number
- Apply the "error" class to the element containing the notification message when the user runs out of guesses

Homework #4 - CitiPix


Chrome File Edit View History Bookmarks People Window Help
City Background
file:///Users/kareemgrant/Dropbox/GA/updated_curriculum/FEWD_Curriculum/Week_05_JS_ComputerScience/Assignm...
CitiPix
Your Cities, Your Pix
Enter a city name...
UPDATE

Homework #4 - CitiPix (cont'd)

Directions:

1) Fork the following respository:

Homework #4

2) Review the assignment here

3) Use process covered in the Git/Github Tutorial to submit your assignment