launch_code

Chapter 10: Functions

LC101 ▸ UNIT 1 ▸ CLASS 5 ▸ OCTOBER 6, 2022

Class Agenda

Announcements

Lecture: Functions

Studio: Chapter 10

Studio Review

Announcements

Graded Assignment Deadline

Completed Assignment #1 due **Monday, 10/10** (enrollment deadline)

Lecture Flow

Focus of lecture is to **build on what you've already learned** during your prep (reading, quiz, exercises) - I will give you more involved examples, connect the dots further, etc.

Post questions in **#lecture-questions**, which will be monitored by TAs.

I will stop at certain points during lecture to answer a few questions live.

Functions

PUTTING CODE TO WORK FOR YOU

launch_code

Functions

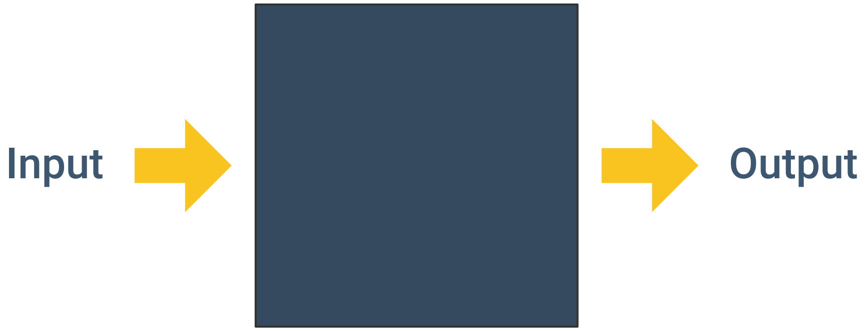
launch_code

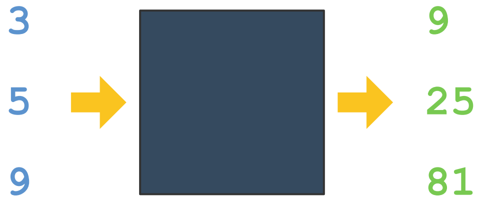
Functions

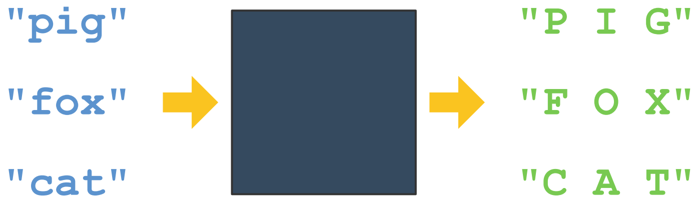
The Black Box

Producing Reliable Results

If the code inside the box is working properly, you should always get the expected output for any input you give it







Functions

Familiar Functions

JS Data Type Conversion

`Number("62.3")` ⇒ `62.3`

`String(594)` ⇒ `"594"`

`Boolean(0)` ⇒ `false`

LaunchCode Modules

```
askQuestion()
gradeQuiz(candidateAnswers)
runProgram()
```

Console API - Log Method

`console.log("hello")` → prints to console

readline-sync Library - Question Method

`input.question("How many cookies? ")` →
prints to console, user enters "3" ⇒ `"3"`

String & Array Methods

`arr.pop()` → removes element from last index ⇒ `"bye"`

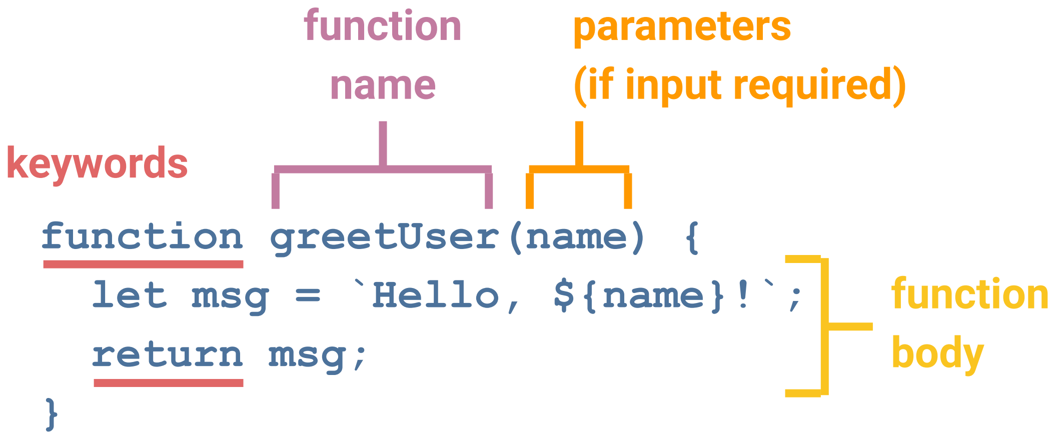
`str.split()` ⇒ `["a", "b", "c"]`

Functions

Creating Functions

Anatomy & Terminology

- Use the keyword **function**
- Give it a descriptive **name** using **camelCase** - verb/noun combo
- `()` must follow the name and hold any parameters if the function requires input
- The code to be executed goes inside the curly braces - **function body**
- The **return** keyword is optional if you aren't returning a value



```
function greetUser(name) {
  let msg = `Hello, ${name}!`;
  return msg;
}
```

Functions

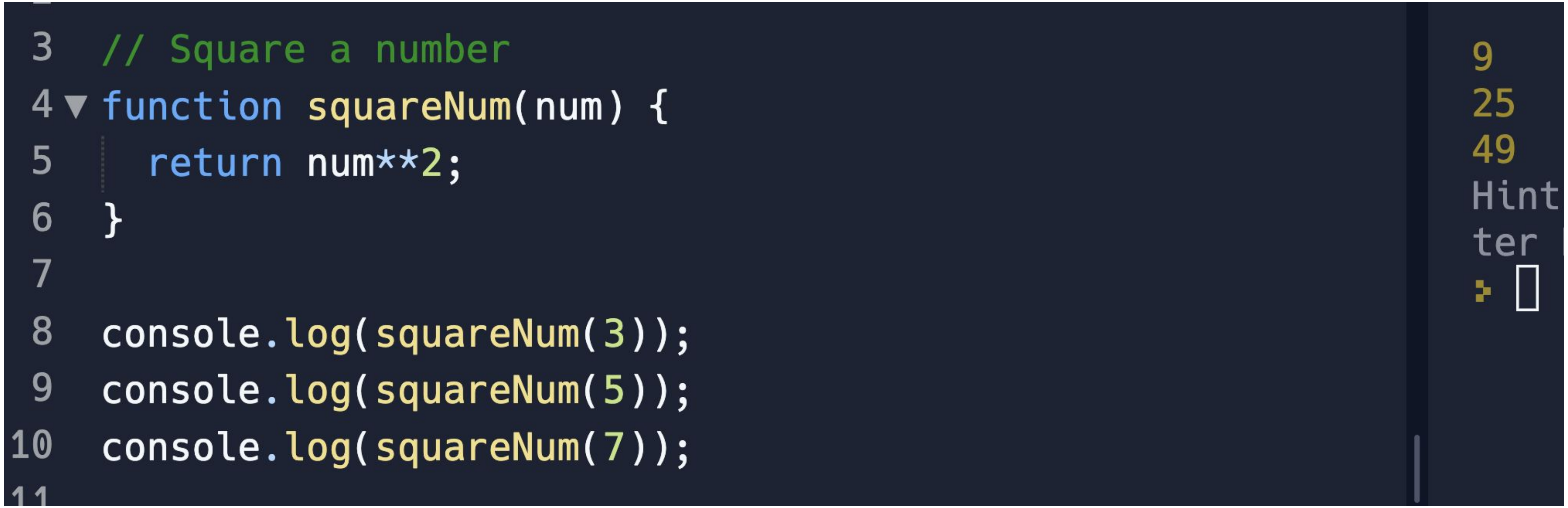
Creating Functions

Parameters are Variables!

- If your function needs to take input, assign **parameters** to represent the data
- Use descriptive names with **camelCase**
- Parameters can be used as variables anywhere **inside** the function body

```
function formatSSN(ssn1, ssn2, ssn3) {
  return `${ssn1}-${ssn2}-${ssn3}`;
}

formatSSN('123', '45', '6789') ⇒ "123-45-6789"
```



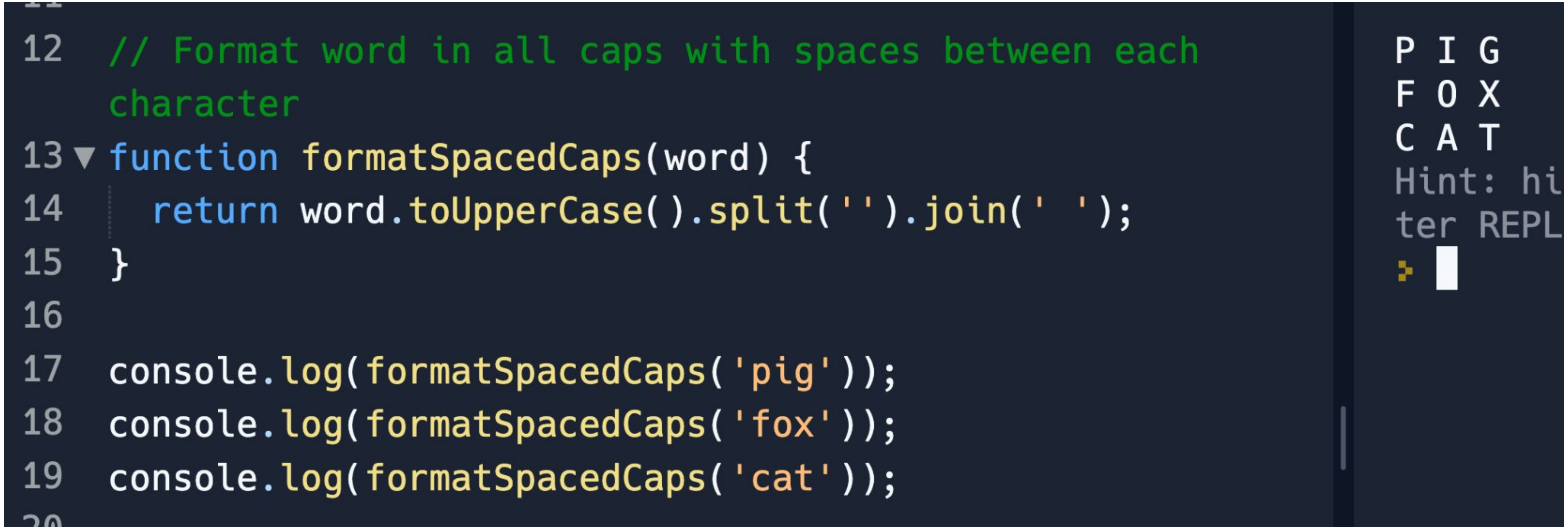
```
// Square a number
function squareNum(num) {
  return num**2;
}

console.log(squareNum(3));
console.log(squareNum(5));
console.log(squareNum(7));
```

```
9
25
49
Hint
ter
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▸ Simple Function with Input and Output



```
// Format word in all caps with spaces between each character
function formatSpacedCaps(word) {
  return word.toUpperCase().split('').join(' ');
}

console.log(formatSpacedCaps('pig'));
console.log(formatSpacedCaps('fox'));
console.log(formatSpacedCaps('cat'));
```

```
P I G
F O X
C A T
Hint: hi
ter REPL
> 
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▶ Another Simple Function with Input and Output

Functions

Creating Functions

Variable Scope

- It's common to use other variables in your function that were **declared outside** the function.
- Parameters have **local** scope only and cannot be accessed from outside the function
- Any variables **declared inside** the function also have **local** scope

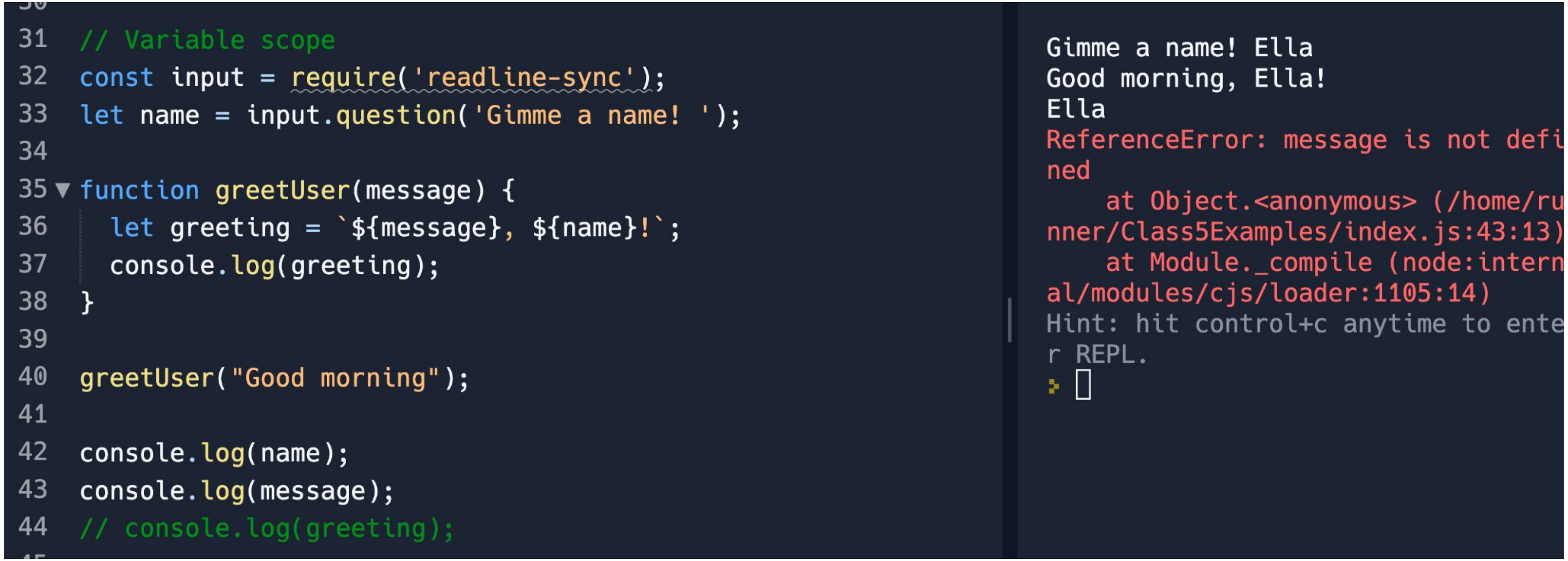
```
const input = require('readline-sync');
let name = input.question('Gimme a name! ');

function greetUser(message) {
  let greeting = `${message}, ${name}!`;
  console.log(greeting);
}
greetUser("Good morning");

console.log(name); // OK - name is in scope
// console.log(message);
// console.log(greeting);
```

```
Gimme a name! Ella
```

`greetUser("Good morning")` ⇒ `"Good morning, Ella!"`



```javascript
// Variable scope
const input = require('readline-sync');
let name = input.question('Gimme a name! ');

function greetUser(message) {
  let greeting = `${message}, ${name}!`;
  console.log(greeting);
}

greetUser("Good morning");

console.log(name);
console.log(message);
// console.log(greeting);
```

```
Gimme a name! Ella
Good morning, Ella!
Ella
ReferenceError: message is not defi
ned
    at Object.<anonymous> (/home/ru
nner/Class5Examples/index.js:43:13)
    at Module._compile (node:intern
al/modules/cjs/loader:1105:14)
Hint: hit control+c anytime to ente
r REPL.
> 
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▸ Variable Scope

Functions

Creating Functions

Variable Scope - Shadowing

- **Shadowing** is the concept of a local variable having the same name as another variable that was declared outside the function.
- JavaScript allows it, but...
- This is a bad practice - don't do it!
- Better to have clarity and prevent confusion

```
let color = "black";

function describeItem(item, color) {
  console.log(`It's a ${color} ${item}!);
}

describeItem("box", "blue");
console.log(color);
```

```
describeItem("box", "blue") ⇒ "It's a blue box!"    color ⇒ "black"
```

Functions

Creating Functions

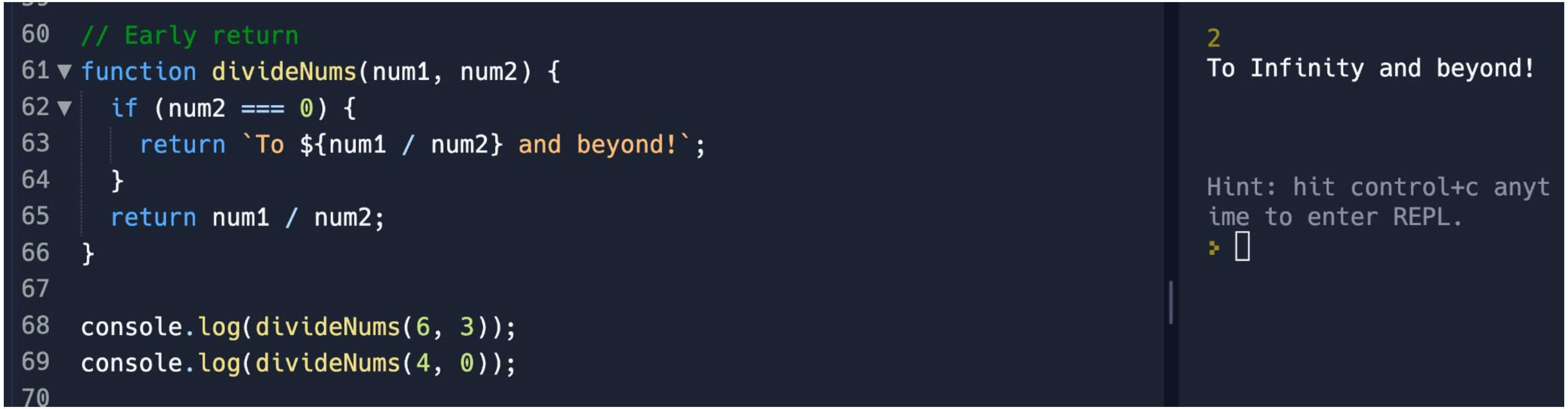
Stopping a Function Early

- The `return` keyword will always stop the function from continuing to execute its code
- Sometimes an **early return** is helpful
- If the condition isn't met it will continue to the end return as normal

```
function divideNums(num1, num2) {
  if (num2 === 0) {
    return `To ${num1 / num2} and beyond!`;
  }
  return num1 / num2;
}

console.log(divideNums(6, 3));
console.log(divideNums(4, 0));
```

```
divideNums(6, 3) ⇒ 2        divideNums(4, 0) ⇒ "To Infinity and beyond!"
```



```
// Early return
function divideNums(num1, num2) {
  if (num2 === 0) {
    return `To ${num1 / num2} and beyond!`;
  }
  return num1 / num2;
}

console.log(divideNums(6, 3));
console.log(divideNums(4, 0));
```

```
2
To Infinity and beyond!


Hint: hit control+c anyt
ime to enter REPL.
›
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▸ Early Return

Functions

Creating Functions

To Return or Not To Return?

- Some functions don't need to return a value
- The `return` keyword is optional in this case

Common Scenarios

- Changing the value of an external variable, like the contents of an array or toggling a boolean
- Logging a message to the console
- Triggering another function, conditionally

```
let allPrepWork = [];
let startedPrepWork = false;

function submitPrepWork(work) {
  allPrepWork.push(work);
  startedPrepWork = true;
}
submitPrepWork("reading");
console.log(startedPrepWork);
submitPrepWork("exercises");
console.log(allPrepWork);
```

```
startedPrepWork ⇒ true        allPrepWork ⇒ ["reading", "exercises"]
```

Functions

Using Functions

Terminology

- You **call** or **invoke** a function once it has been defined
- For each parameter, pass in an **argument** (actual data the function needs) in the same order

```
function formatDate(weekday, mm, dd, yyyy) { parameters
  return `${weekday}, ${mm}/${dd}/${yyyy}`;
}
let formattedDate = formatDate( "Friday", "10", "07", "2022" ); arguments
console.log(formattedDate);

formattedDate ⇒ "Friday, 10/07/2022"
```

Functions

Using Functions

Making Use of Return Values

If a function returns a value, you need to do something with it!

Common Scenarios

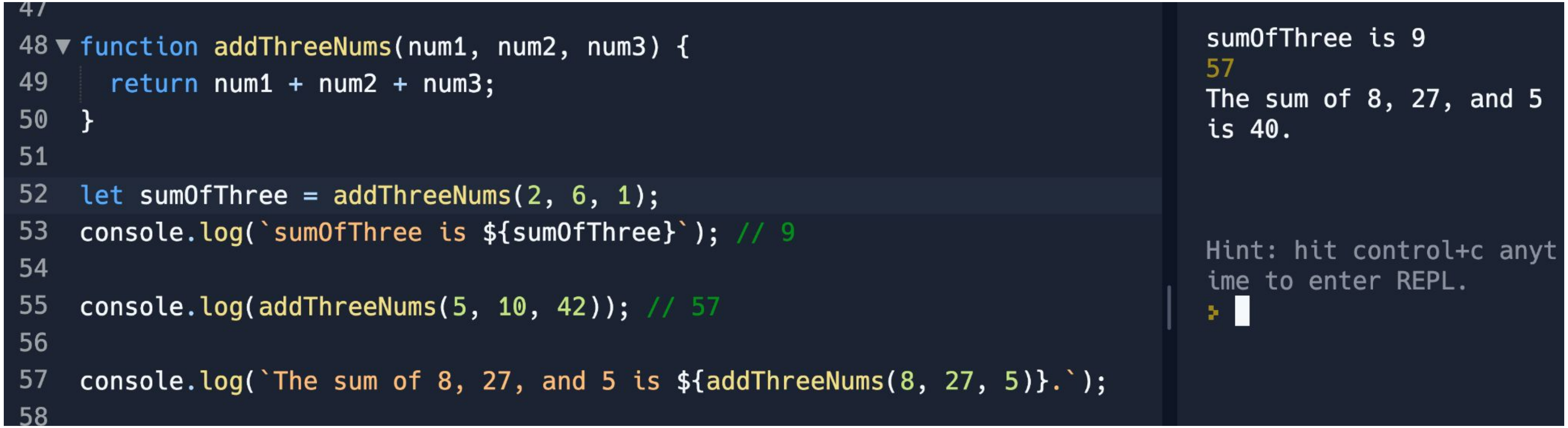
- Store output in a variable
- Log output directly to console
- Use output directly in another expression or template literal

```
function addThreeNums(num1, num2, num3) {
  return num1 + num2 + num3;
}

// Store in a variable to use later
let sumOfThree = addThreeNums(2, 6, 1);
console.log(`sumOfThree is ${sumOfThree}`);

// Print to console to see output;
console.log(addThreeNums(5, 10, 42));

// Call directly where value is needed
console.log(`The sum of 8, 27, and 5 is
  ${addThreeNums(8, 27, 5)}.`);
```



```
function addThreeNums(num1, num2, num3) {
  return num1 + num2 + num3;
}

let sumOfThree = addThreeNums(2, 6, 1);
console.log(`sumOfThree is ${sumOfThree}`); // 9

console.log(addThreeNums(5, 10, 42)); // 57

console.log(`The sum of 8, 27, and 5 is ${addThreeNums(8, 27, 5)}.`);
```

```
sumOfThree is 9
57
The sum of 8, 27, and 5
is 40.


Hint: hit control+c anyt
ime to enter REPL.
›
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▸ Using the Return Value of a Function

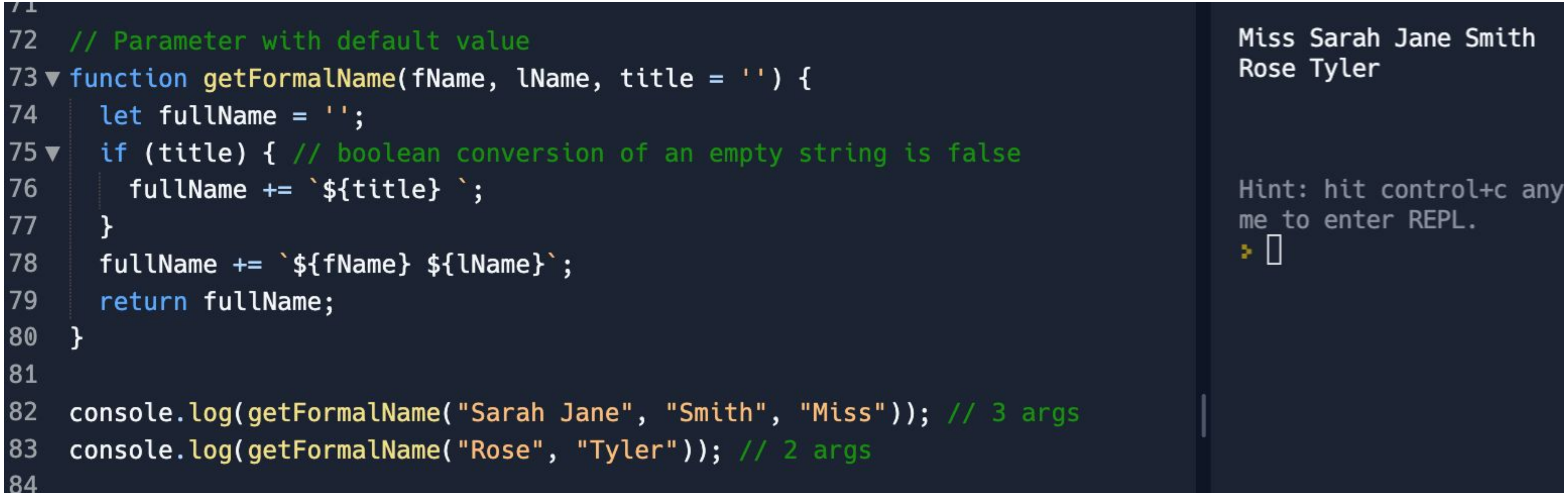
Functions

Parameters with Default Values

Flexible Functions

- It is possible to design a function to take **optional parameters**
- In order for this to work, you have to assign a **default value**
- This is done as part of the function definition.
- When calling the function, leave the optional parameter off and let it just take the default value

```
function getFormalName(fName, lName, title = '') {
  let fullName = '';
  if (title !== '') {
    fullName += `${title} `;
  }
  fullName += `${fName} ${lName}`;
  return fullName;
}
```



```javascript
// Parameter with default value
function getFormalName(fName, lName, title = '') {
  let fullName = '';
  if (title) { // boolean conversion of an empty string is false
    fullName += `${title} `;
  }
  fullName += `${fName} ${lName}`;
  return fullName;
}

console.log(getFormalName("Sarah Jane", "Smith", "Miss")); // 3 args
console.log(getFormalName("Rose", "Tyler")); // 2 args
```

```
Miss Sarah Jane Smith
Rose Tyler


Hint: hit control+c any
me to enter REPL.
› 
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

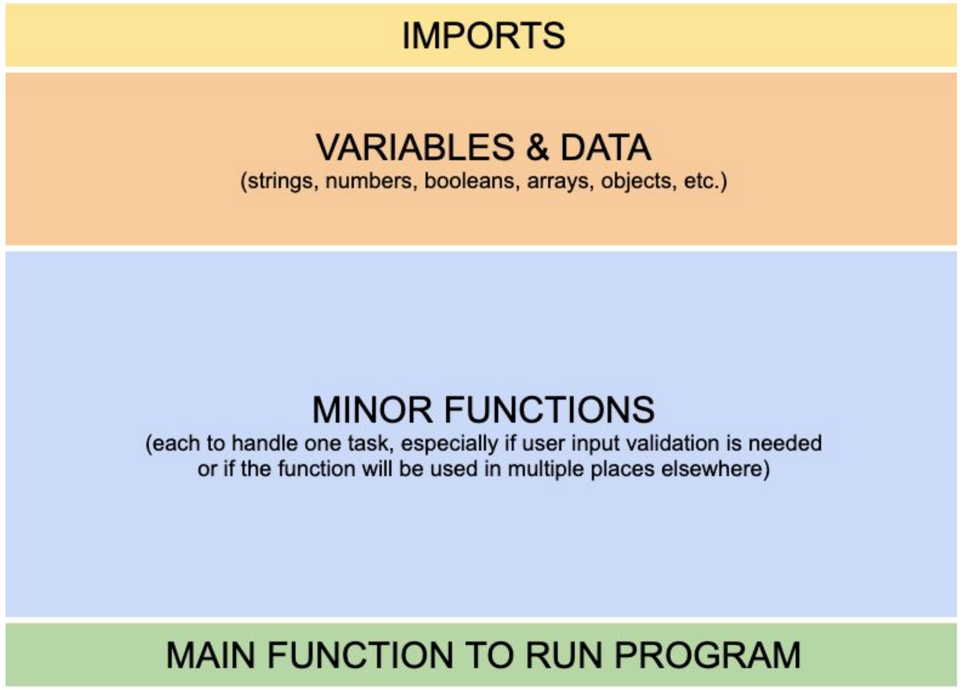
EXAMPLE ▸ Parameters with Default Values

Functions

Function Composition

Keep Functions Small, Focused

- Functions should typically do ONE thing. Especially **utility** or **helper** functions.
- You can have one main function to put it all together, and use lots of other functions to take care of specific small tasks.
- Functions don't have to be sequential - usually functions are put toward the end of a program.


IMPORTS
VARIABLES & DATA
(strings, numbers, booleans, arrays, objects, etc.)
MINOR FUNCTIONS
(each to handle one task, especially if user input validation is needed or if the function will be used in multiple places elsewhere)
MAIN FUNCTION TO RUN PROGRAM

Functions

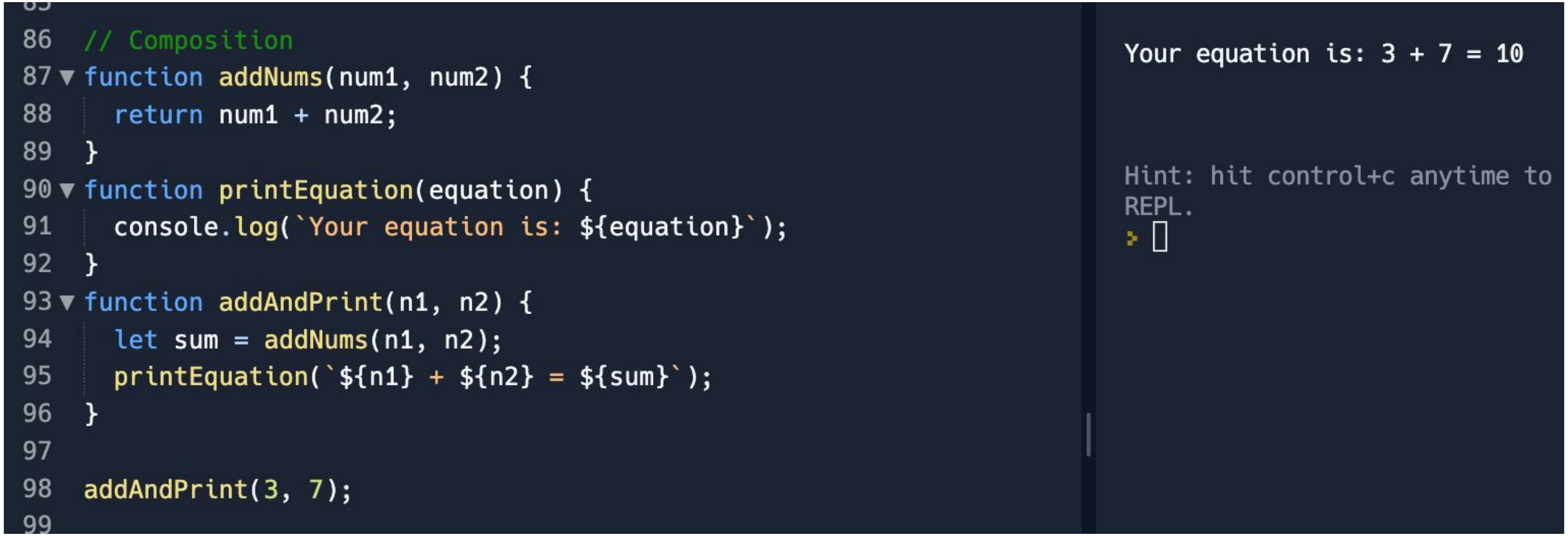
Function Composition

Composition

- Be smart about how you compose functions - break out smaller tasks
- Call functions from within other functions for different reasons:
  - Calculate and return values
  - Request user input
  - Validate user input
  - Print things to the console
  - etc...

```
function addNums(num1, num2) {
  return num1 + num2;
}
function printEquation(equation) {
  console.log(`Your equation is: ${equation}`);
}
function addAndPrint(n1, n2) {
  let sum = addNums(n1, n2);
  printEquation(`${n1} + ${n2} = ${sum}`);
}

addAndPrint(3, 7);
```



```
// Composition
function addNums(num1, num2) {
  return num1 + num2;
}
function printEquation(equation) {
  console.log(`Your equation is: ${equation}`);
}
function addAndPrint(n1, n2) {
  let sum = addNums(n1, n2);
  printEquation(`${n1} + ${n2} = ${sum}`);
}

addAndPrint(3, 7);
```

```
Your equation is: 3 + 7 = 10


Hint: hit control+c anytime to
REPL.
> 
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class5Examples#index.js

EXAMPLE ▶ Incrementing by a Number Greater than 1

Studio

launch_code

Studio

Tonight's Studio - Chapter 10

Function Composition

- Reverse characters
- Modify to reverse digits
- Reverse an array (while also using your other function to reverse the characters or digits of each element in the array)

Instructions

https://education.launchcode.org/intro-to-professional-web-dev/chapters/functions/studio.html

Solution

https://replit.com/@CarolineRose/FunctionsExercises03-05#index.js

What's Next

Class 6 - More on Functions - Monday, 10/10

- Due before class
  - Prep work (reading, quiz, exercises) - chapter 11
  - Graded Assignment 1 **COMPLETED**
- Lecture
- Studio
- Review

Class 7 - Objects - Thursday, 10/13

- Due before class
  - Prep work (reading, quiz, exercises) - chapter 12
- Lecture
- Studio
- Review

This lecture is part of a series.
Each class has two recorded sessions - lecture and post-studio review.

YouTube Playlist:
https://tinyurl.com/5n6usbef

launch_code