launch_code

More on Objects
Chapter 13: Modules

LC101 ▸ UNIT 1 ▸ CLASS 8 ▸ OCTOBER 17, 2022

Class Agenda

Announcements

Lecture: More on Objects

Lecture: Modules

Demo: 21 Game

Studio: Chapter 13

Announcements

Catch-up Class!

Thursday, 10/20 - Go straight to your studio Zoom to work on Graded Assignment #2 and get help from your TAs.

Graded Assignment #2 Deadline

Monday, 10/24 - But be kind to your TAs and turn it in a couple days early so they can give you feedback before the real deadline!

More on Objects

ADDITIONAL PRACTICAL KNOWLEDGE

launch_code

More on Objects

launch_code

Review of Object Basics

Special JS Object Methods

Nested Objects

More on Objects

Review of Object Basics

Structure and Syntax

- An **object** is represented in JavaScript with `{}`
- Data is stored within an object using **key/value pairs**
- A key/value pair is a **method** if the **value** is a function; otherwise it's a regular **property**
- Use **bracket** *or* **dot notation** to access or modify a property
- Use **dot notation** with `()` to call a method

```
let novel = {
  title: "Murder on the Orient Express",
  author: "Agatha Christie",
  numPages: 256
  read: function () {
    console.log(`I read ${this.title}, and
    WHAT a twist ending!`);
  }
};
```

```
novel["author"] ⇒ "Agatha Christie"
novel.numPages ⇒ 256
novel.read() ⇒ "I read Murder on the
               Orient Express, and WHAT
               a twist ending!"
```

More on Objects

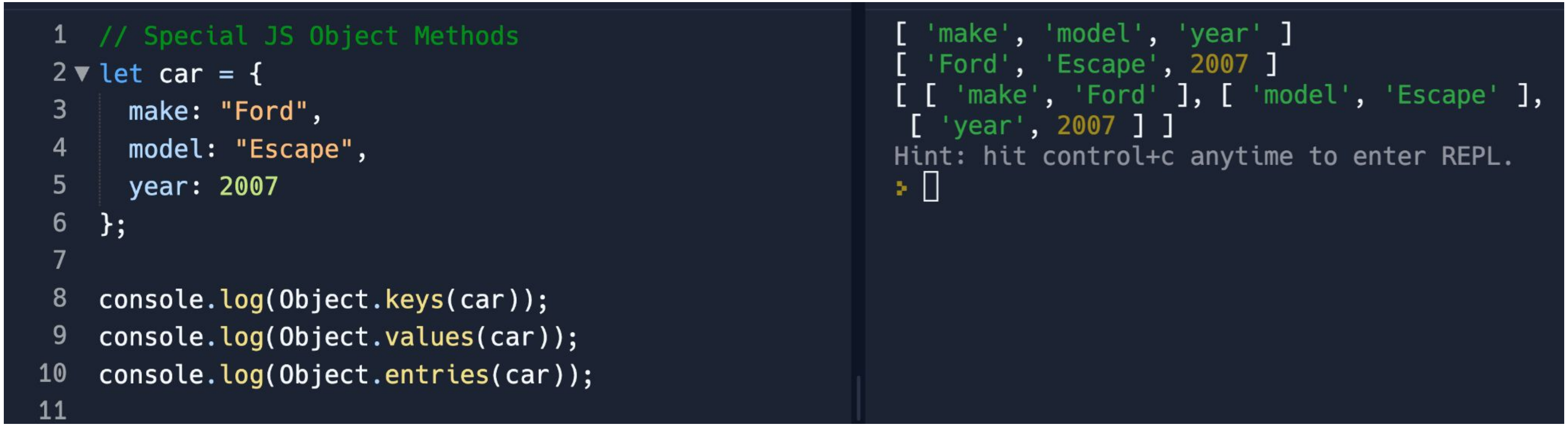
Special JS Object Methods

Accessing Keys and Values

- `Object.keys(obj)` will return an array of all the keys of `obj`
- `Object.values(obj)` will return an array of all the values of `obj`
- `Object.entries(obj)` will return an array of tuples (arrays with two elements) in the format `[key, value]`

```
let car = {
  make: "Ford",
  model: "Escape",
  year: 2007
};

Object.keys(car) ⇒

    ["make", "model", "year"]

Object.values(car) ⇒

    ["Ford", "Escape", 2007]
```

```
Object.entries(car) ⇒
    [["make", "Ford"], ["model", "Escape"], ["year", 2007]]
```



```javascript
// Special JS Object Methods
let car = {
  make: "Ford",
  model: "Escape",
  year: 2007
};

console.log(Object.keys(car));
console.log(Object.values(car));
console.log(Object.entries(car));
```

```
[ 'make', 'model', 'year' ]
[ 'Ford', 'Escape', 2007 ]
[ [ 'make', 'Ford' ], [ 'model', 'Escape' ],
  [ 'year', 2007 ] ]
Hint: hit control+c anytime to enter REPL.
> 
```

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class8Examples-MoreOnObjects

EXAMPLE ▸ Special JS Object Methods

More on Objects

Nested Objects

Arrays within Objects

Nested arrays require more complex bracket notation to access nested **elements**

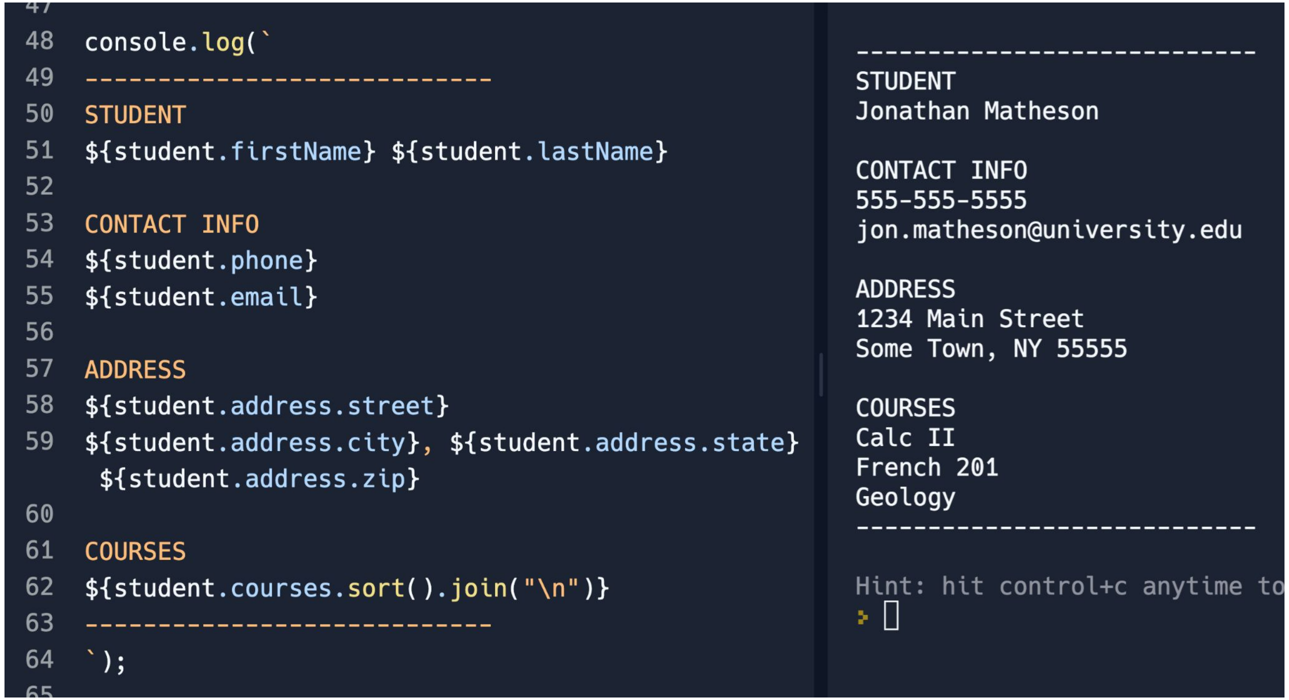
Objects within Objects

Nested objects require more complex bracket or dot notation to access nested **properties**

```
let student = {
  firstName: "Jonathan",
  lastName: "Matheson",
  address: {
    street: "1234 Main Street",
    city: "Some Town",
    state: "NY",
    zip: "55555"
  },
  email: "jon.matheson@university.edu",
  phone: "555-555-5555",
  courses: ["Geology", "Calc II", "French 201"]
};
```

nested object

nested array



```
console.log(`
----------------------------
STUDENT
${student.firstName} ${student.lastName}

CONTACT INFO
${student.phone}
${student.email}

ADDRESS
${student.address.street}
${student.address.city}, ${student.address.state}
 ${student.address.zip}

COURSES
${student.courses.sort().join("\n")}
----------------------------
`);
```

```
----------------------------
STUDENT
Jonathan Matheson

CONTACT INFO
555-555-5555
jon.matheson@university.edu

ADDRESS
1234 Main Street
Some Town, NY 55555

COURSES
Calc II
French 201
Geology
----------------------------

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

Note:
Could we have referenced each course separately?

Example:

```
${student.courses[1]}
${student.courses[2]}
${student.courses[0]}
```

Sure. But imagine using this code for hundreds of students and needing it to work for each one regardless of how many courses and what order they were in!

Fork & explore on Repl.it:
https://replit.com/@CarolineRose/Class8Examples-MoreOnObjects

EXAMPLE ▸ Nested Objects

Modules

MULTIPLE JAVASCRIPT FILES

launch_code

Modules

launch_code

The Concept of Modules

Terminology

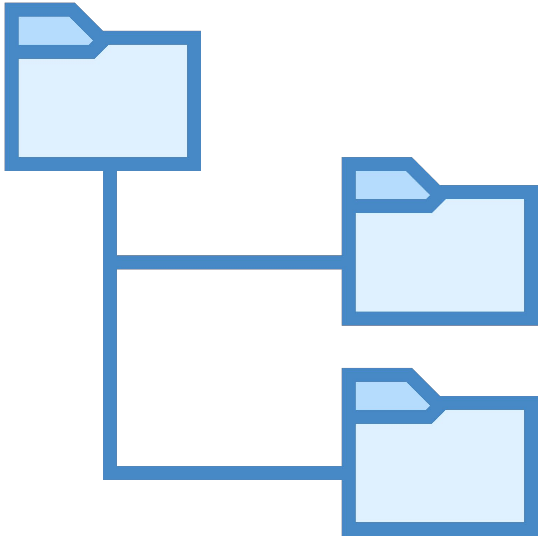
Module Basics

Modules

The Concept of Modules

Think Big

- A website, game, or application can have **millions** of lines of code
- Can you imagine if all that code was in a single file?
- **Modules** allow us to split our code into multiple files, making it:
  - More organized
  - Easier to navigate
  - Reusable
- Modules can also be **shareable** so others can use it



Modules

Terminology

Import/Export

- You must explicitly **export** data and functions in order for them to be usable in another file
- You must **import** items from other files at the top of whichever file they're needed in
- That's all there is to creating and using modules!

Module vs Package vs Library vs Dependency

- A **module** is a single file with exported code
- A **package** is a group of modules (and/or smaller packages)
- A **library** is a blanket term for reusable code, but generally refers to packages that have been published for others to use
- A **dependency** is what you call a library or package once it has been installed in your application

Modules

Module Basics

Export from One File

- At the bottom of the file, type `module.exports =`
- If multiple items need to be exported, set an object:
    - Each **key** should be the name you want to call it in the other file where it's imported
    - Each **value** should be the name of the item inside the current file above.
    - If you're exporting functions, remember to only **reference** their names, not call them.
- If only a single item (data or function) needs to be exported, just set it directly.

```
let albus = {
  fName: "Albus",
  lName: "Dumbledore"
};

function concatName() {
  return `${albus.fName} ${albus.lname};
}

module.exports = {
  headmaster: albus,
  getFullName: concatName
}
```

Modules

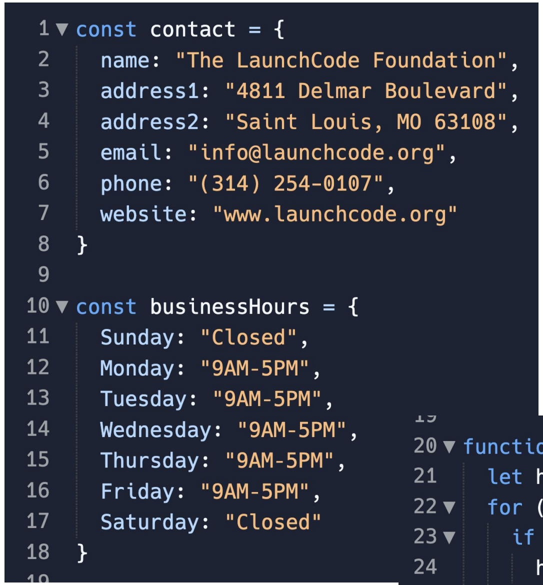
Module Basics

Import into a File

- All import statements go at the **top** of the file
- **Option 1:** Import a **single variable** to represent `module.exports` and use **dot notation** to access each item if it's an object with multiple exported items
- **Option 2:** You can use a syntax called **destructuring** to "reach into" the `module.exports` object and reference each item directly

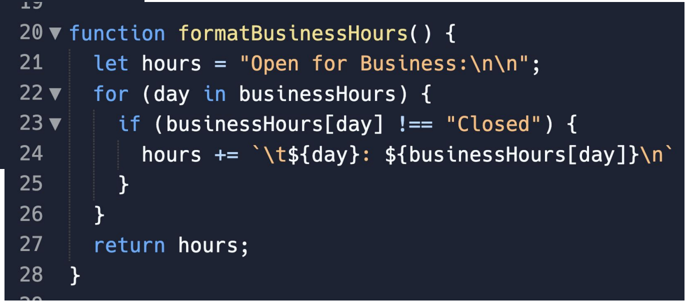
```
// Option 1
const stuff = require('./stuff.js');

console.log(stuff.headmaster);
console.log(stuff.getFullName());

// Option 2
const { headmaster, getFullName } =
  require('./stuff.js');

console.log(headmaster);
console.log(getFullName());
```

EXAMPLE ▸ Module Basics - Single Export - `contact.js`

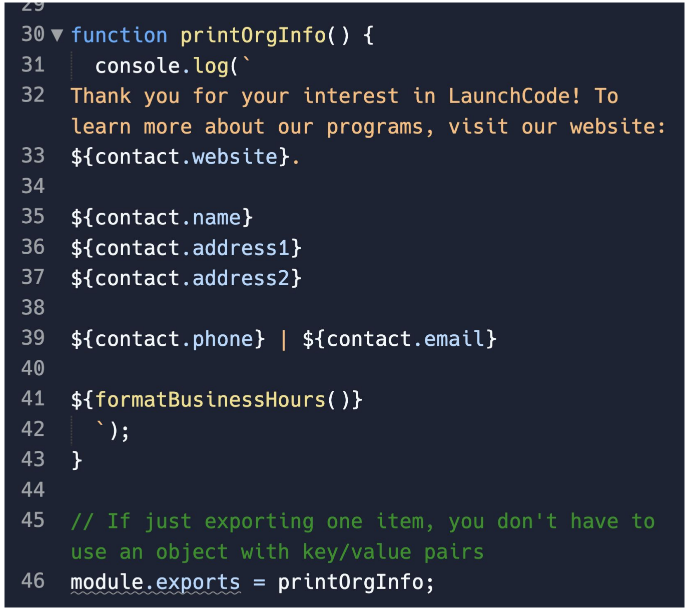


```javascript
const contact = {
  name: "The LaunchCode Foundation",
  address1: "4811 Delmar Boulevard",
  address2: "Saint Louis, MO 63108",
  email: "info@launchcode.org",
  phone: "(314) 254-0107",
  website: "www.launchcode.org"
}

const businessHours = {
  Sunday: "Closed",
  Monday: "9AM-5PM",
  Tuesday: "9AM-5PM",
  Wednesday: "9AM-5PM",
  Thursday: "9AM-5PM",
  Friday: "9AM-5PM",
  Saturday: "Closed"
}
```

This module contains multiple objects and functions, but only the final function needs to be exported for use in `index.js`.

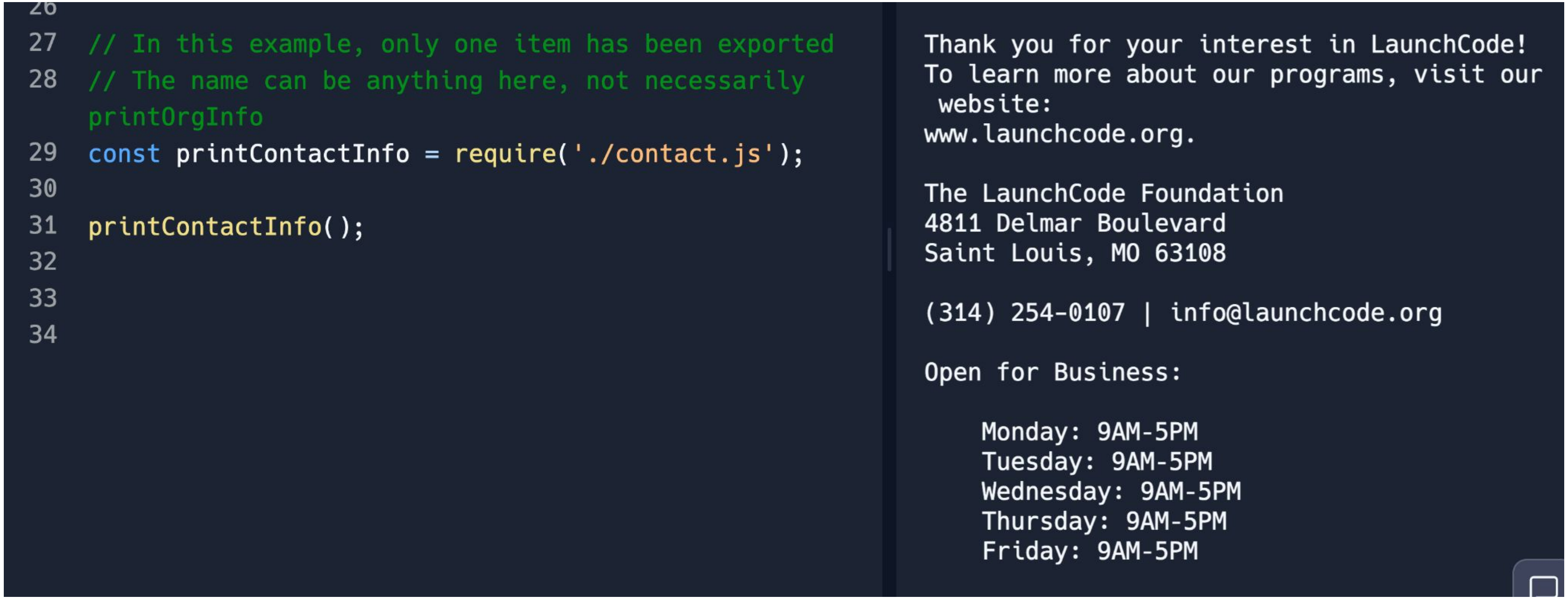


```javascript
function formatBusinessHours() {
  let hours = "Open for Business:\n\n";
  for (day in businessHours) {
    if (businessHours[day] !== "Closed") {
      hours += `\t${day}: ${businessHours[day]}\n`
    }
  }
  return hours;
}
```



```javascript
function printOrgInfo() {
  console.log(`
Thank you for your interest in LaunchCode! To learn more about our programs, visit our website:
${contact.website}.

${contact.name}
${contact.address1}
${contact.address2}

${contact.phone} | ${contact.email}

${formatBusinessHours()}
  `);
}

// If just exporting one item, you don't have to use an object with key/value pairs
module.exports = printOrgInfo;
```

Fork & explore on Repl.it: https://replit.com/@CarolineRose/Class8Examples-ModuleBasics

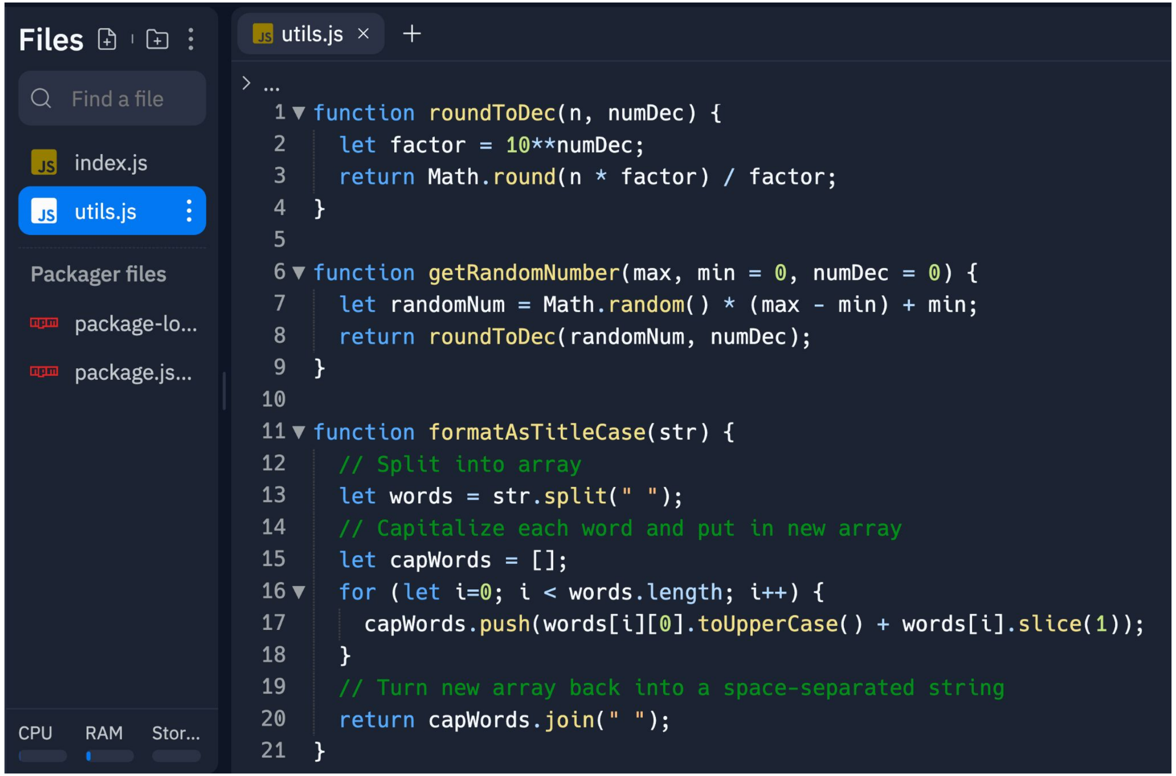


```
// In this example, only one item has been exported
// The name can be anything here, not necessarily
printOrgInfo
const printContactInfo = require('./contact.js');

printContactInfo();
```

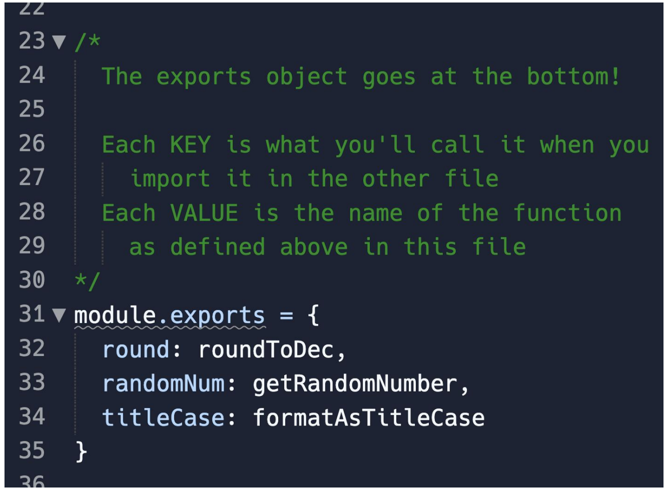
```
Thank you for your interest in LaunchCode!
To learn more about our programs, visit our
 website:
www.launchcode.org.

The LaunchCode Foundation
4811 Delmar Boulevard
Saint Louis, MO 63108

(314) 254-0107 | info@launchcode.org

Open for Business:

    Monday: 9AM-5PM
    Tuesday: 9AM-5PM
    Wednesday: 9AM-5PM
    Thursday: 9AM-5PM
    Friday: 9AM-5PM
```

Fork & explore on Repl.it: https://replit.com/@CarolineRose/Class8Examples-ModuleBasics

EXAMPLE ▶ Module Basics - Single Import - `index.js`



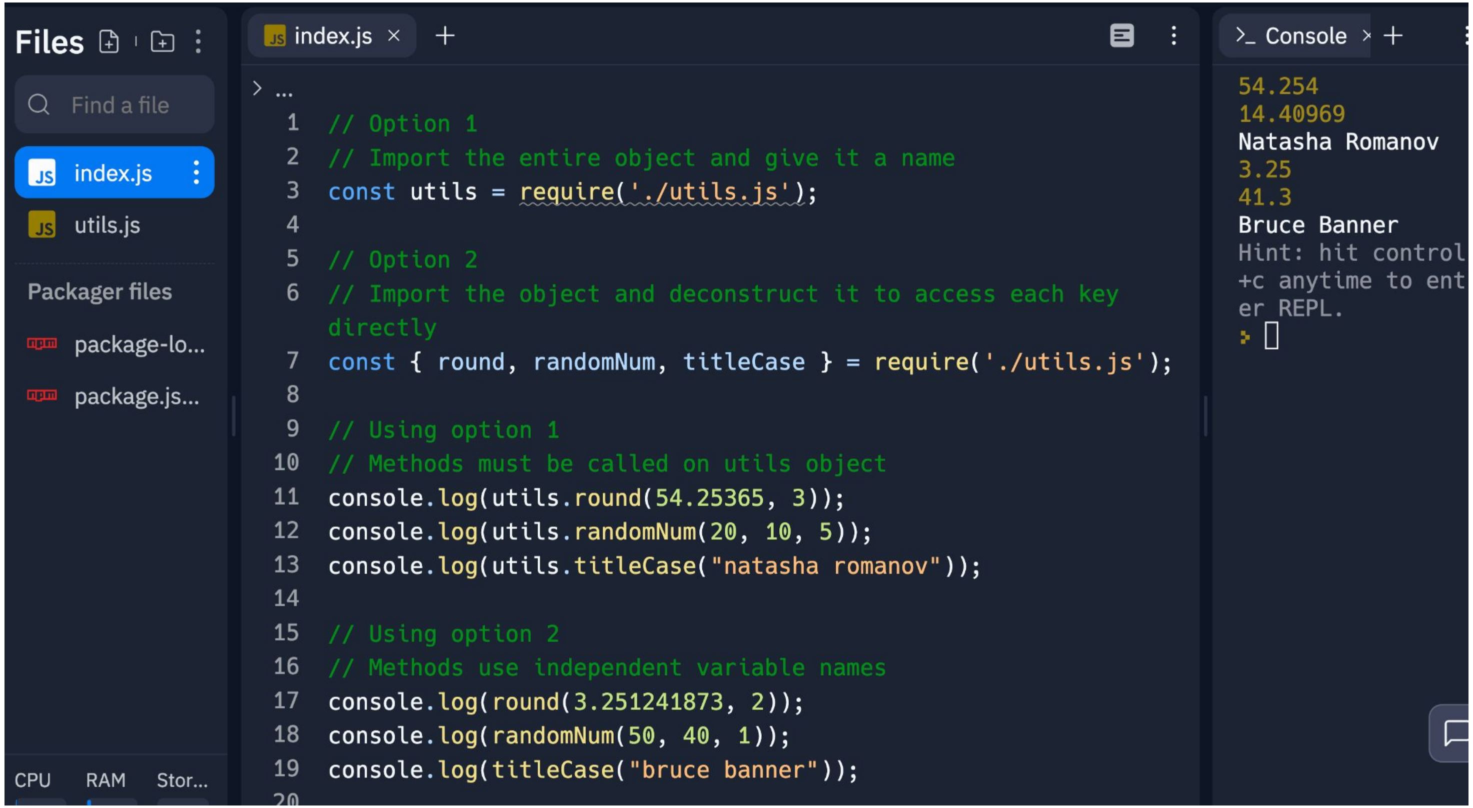
```javascript
function roundToDec(n, numDec) {
  let factor = 10**numDec;
  return Math.round(n * factor) / factor;
}

function getRandomNumber(max, min = 0, numDec = 0) {
  let randomNum = Math.random() * (max - min) + min;
  return roundToDec(randomNum, numDec);
}

function formatAsTitleCase(str) {
  // Split into array
  let words = str.split(" ");
  // Capitalize each word and put in new array
  let capWords = [];
  for (let i=0; i < words.length; i++) {
    capWords.push(words[i][0].toUpperCase() + words[i].slice(1));
  }
  // Turn new array back into a space-separated string
  return capWords.join(" ");
}
```



```javascript
/*
  The exports object goes at the bottom!

  Each KEY is what you'll call it when you
    import it in the other file
  Each VALUE is the name of the function
    as defined above in this file
*/
module.exports = {
  round: roundToDec,
  randomNum: getRandomNumber,
  titleCase: formatAsTitleCase
}
```

Fork & explore on Repl.it: https://replit.com/@CarolineRose/Class8Examples-ModuleBasics

EXAMPLE ▸ Module Basics - Multiple Exports - `utils.js`



```
// Option 1
// Import the entire object and give it a name
const utils = require('./utils.js');

// Option 2
// Import the object and deconstruct it to access each key directly
const { round, randomNum, titleCase } = require('./utils.js');

// Using option 1
// Methods must be called on utils object
console.log(utils.round(54.25365, 3));
console.log(utils.randomNum(20, 10, 5));
console.log(utils.titleCase("natasha romanov"));

// Using option 2
// Methods use independent variable names
console.log(round(3.251241873, 2));
console.log(randomNum(50, 40, 1));
console.log(titleCase("bruce banner"));
```

```
54.254
14.40969
Natasha Romanov
3.25
41.3
Bruce Banner
Hint: hit control
+c anytime to ent
er REPL.
```

Fork & explore on Repl.it: https://replit.com/@CarolineRose/Class8Examples-ModuleBasics

EXAMPLE ▶ Module Basics - Multiple Imports - `index.js`

Demo

21 Game

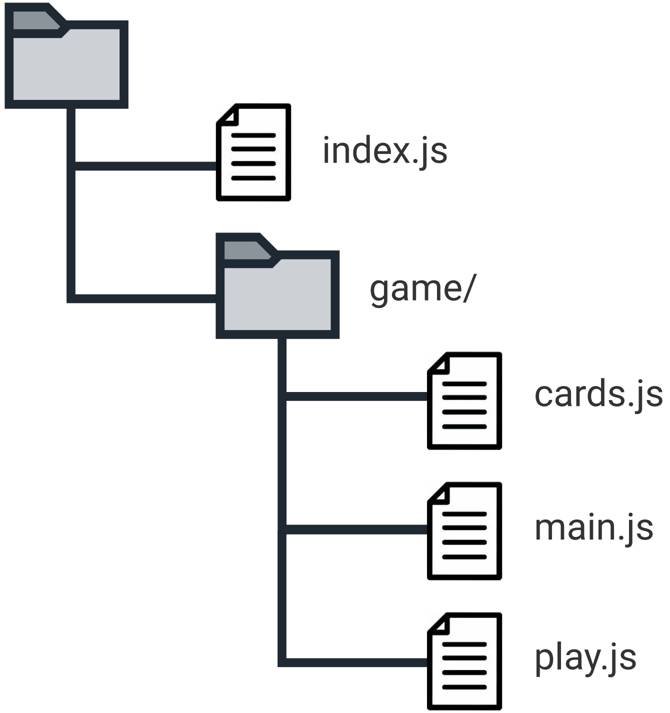
Modules

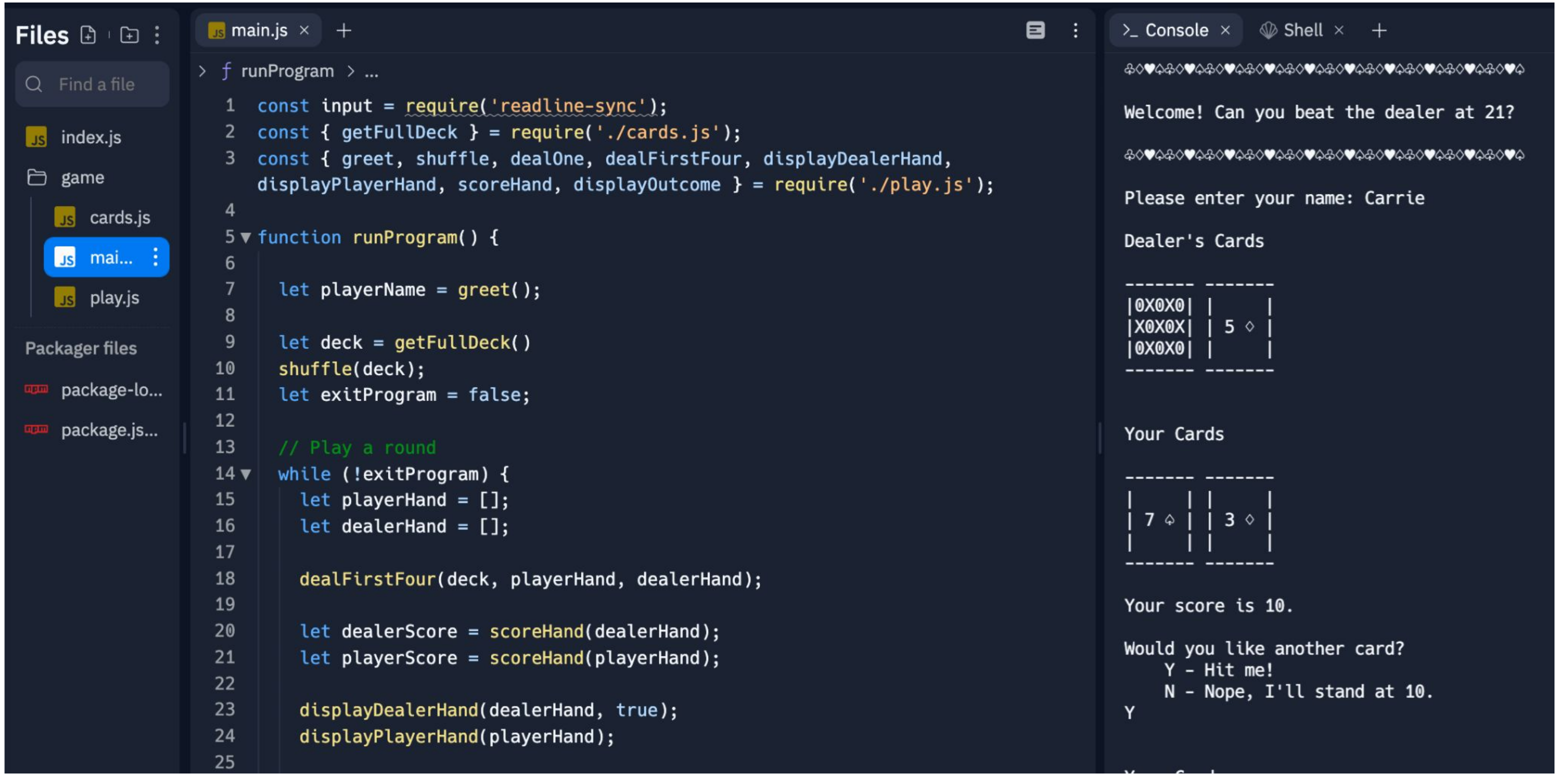
Example: 21 Game

Divide and Conquer

- **index.js** is the file JS compiles and runs - it just needs to import `runProgram()` from main.js
- **main.js** contains a single function to run the cycle of the game, but it depends on functions and data from cards.js and play.js
- **cards.js** just handles the creation of the card deck - an array of objects with data on each card
- **play.js** has many small functions to handle all the calculations and display most things to the player

IMPORTANT: It's *more than* okay if you don't understand everything you're about to see! This is a complex example.





```javascript
const input = require('readline-sync');
const { getFullDeck } = require('./cards.js');
const { greet, shuffle, dealOne, dealFirstFour, displayDealerHand,
displayPlayerHand, scoreHand, displayOutcome } = require('./play.js');

function runProgram() {

  let playerName = greet();

  let deck = getFullDeck()
  shuffle(deck);
  let exitProgram = false;

  // Play a round
  while (!exitProgram) {
    let playerHand = [];
    let dealerHand = [];

    dealFirstFour(deck, playerHand, dealerHand);

    let dealerScore = scoreHand(dealerHand);
    let playerScore = scoreHand(playerHand);

    displayDealerHand(dealerHand, true);
    displayPlayerHand(playerHand);

```

```
♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤

Welcome! Can you beat the dealer at 21?

♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤♧◇♥♤

Please enter your name: Carrie

Dealer's Cards

------- -------
|0X0X0| |     |
|X0X0X| | 5 ◇ |
|0X0X0| |     |
------- -------


Your Cards

------- -------
|     | |     |
| 7 ♤ | | 3 ◇ |
|     | |     |
------- -------

Your score is 10.

Would you like another card?
    Y - Hit me!
    N - Nope, I'll stand at 10.
Y
```

Fork & explore on Repl.it: https://replit.com/@CarolineRose/ModulesExample-21Game

EXAMPLE ▶ Modules Demo - 21 Game

Studio

Studio

Tonight's Studio - Chapter 13

Boosting Confidence

- This is a non-coding studio
- What do you do when you start doubting yourself?

Instructions

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

What's Next

Catch-Up Class - Thursday, 10/20

- Get help from TAs on GA#2

Class 9 - Unit Testing - Monday, 10/24

- Due before class
  - Prep work (reading, quiz, exercises) - chapter 14
  - Graded Assignment #2
- 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