

Data Types

Built-In

- String
- Number
- Boolean
- Date
- Object
- Array

Not built-in

- Whatever you want!
- Just define it with a class
- Create instances of it with *new*

Screen Recorder is sharing your screen and audio.

Stop sharing

Hide



Parts of defining a class

- The name of the class
- Constructor
 - Used to initialize an object
 - Used to communicate required data to initialize an object!
- Instance variables: the state of the thing
- Instance methods: how the world interacts with the thing
 - Accessors: get some data from the object
 - Mutators: change the state of the object



Parts of defining a class in JavaScript

- Class

Use the `class` keyword and then a `TitleCase` name

- Instance methods

A `camelCase` name, a parameter list, and a block of code

- Constructor

A special method named `constructor`

- Instance variables

Use `this.camelCaseName` to get and set the value



- Instance variables

Use `this.camelCaseName` to get and set the value

“Module”

is just a JavaScript file

Use `exports` when you want to use the dot notation.

```
exports.Person = class Person {  
  // Person class definition  
};
```

```
exports.MAX_TEMP = 122;
```

exports
module-level

Use `module.exports` when you want to use the object literal notation.

```
class Person {  
  // Person class definition  
}
```

```
const MAX_TEMP = 122;
```

```
module.exports = {  
  Person: Person,  
  MAX_TEMP: MAX_TEMP,  
};
```

|| Screen Recorder is sharing your screen and audio.

Stop sharing

Hide





Importing Modules

If they're *your* modules in *your* code, use a relative path in the require function with *no* ".js" extension.

```
// your module in the same
// directory
const hr = require(`./hr`);

// your module in another directory
const tax = require(`../util/tax`);
```

If they're modules that are built into Node or are installed with npm, don't use a prefix nor a ".js" extension.

```
// built-in
const path = require(`path`);

// installed via npm
const express = require(`express`);
```

Export and Import

EXPORTED from “./hr.js”:

```
exports.Person = class Person {  
  // Person class definition  
};
```

```
exports.MAX_TEMP = 122;
```

IMPORTED into current module:

```
const hr = require('./hr');  
  
const bob = new hr.Person('Bob');  
  
if (temp > hr.MAX_TEMP) {  
  console.error('We will melt!');  
}
```


Export and Import with Destructuring



EXPORTED from `"./hr.js"`:

```
exports.Person = class Person {  
  // Person class definition  
};
```

```
exports.MAX_TEMP = 122;
```

IMPORTED into current module:

```
const {Person, MAX_TEMP} = require('./hr');
```

```
const bob = new Person('Bob');
```

```
if (temp > MAX_TEMP) {  
  console.error('We will melt!');  
}
```

What we'll be “modeling” in code

A quiz which has

- a name
- a collection of questions
- a way to administer it
- a score


A couple kinds of questions, each of which is

- some text
- a set of answers
- an indication which is the correct answer
- a way to check an answer

Adding money to a Wallet

```
class Wallet {  
    removeMoney(amount) {  
        this.balance -= amount;  
    }  
    // More stuff inside  
}
```

 <i>Dog</i>
Name
Age
Tricks
speak bark learnNewTrick performTricks

 <i>Cat</i>
Name
Age
speak meow

Parts of defining a class

- The name of the class
- Constructor
 - Used to initialize an object
 - Used to communicate required data to initialize an object!
- Instance variables: the state of the thing
- Instance methods: how the world interacts with the thing
 - Accessors: get some data from the object
 - Mutators: change the state of the object



Exporting Your Classes, Functions, and Data

Use `exports` when you want to use the dot notation.

```
exports.Person = class Person {  
    // Person class definition  
};
```

```
exports.MAX_TEMP = 122;
```

exports
module-level
variable

Use `module.exports` when you want to use the object literal notation.

```
class Person {  
    // Person class definition  
}
```

```
const MAX_TEMP = 122;
```

```
module.exports = {  
    Person: Person,  
    MAX_TEMP: MAX_TEMP,  
};
```



How much is in the Wallet?

```
class Wallet {  
    getBalance() {  
        return this.balance;  
    }  
    // More stuff inside  
}
```

Get a card out of the Wallet?

```
class Wallet {  
  getCardOut(name) {  
    const index = this.cards.findIndex(x => x.name === name);  
  
    if (index === -1) return null;  
  
    const card = this.cards[index];  
    this.cards.splice(index, 1); // take card out of array  
    return card;  
  }  
  // More stuff inside  
}
```



Parts of defining a class in JavaScript

- Class

Use the `class` keyword and then a TitleCase name

- Instance methods

A camelCase name, a parameter list, and a block of code

- Constructor

A special method named `constructor`

- Instance variables

Use `this.camelCaseName` to get and set the value



```
const readline = require('readline');

const rl = readline.createInterface(process.stdin, process.stdout);

// Reminder: rl.question is an asynchronous function
rl.question('Whatever prompt you want to ask the user', answer => {
  // Do stuff with that answer

  // rl.close() // If you want to close the readline interface
});
```

Use `exports` when you want to use the dot notation.

```
exports.Person = class Person {  
    // Person class definition  
};
```

```
exports.MAX_TEMP = 122;
```

The constructor for Wallet

```
class Wallet {  
    constructor(startAmount, cards) {  
        this.balance = startAmount;  
        this.cards = cards;  
    }  
    // More stuff inside  
}
```

