Object Methods

Overview

```
/*
  - What is a method?
  - Creating, accessing, running a method
  - Introduction to this
*/
```

What is a method?

```
/* Methods are actions you can perform on a value */

/* We've been using methods throughout this course, like .indexOf */

/* these methods are built into JS; we don't have to define them
   ourselves */

let countries = ['Argentina', 'Bolivia', 'Brazil', 'Chile'];
console.log(countries.indexOf('Brazil'));
```

Creating a method

```
/* We can create our own methods too! */

/* Consider this object: */

let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral'
};

/* objects can hold any type of value, including functions! */

/* if we add a function to this object, that function is now a method of
   the object */
```

I'm from a method!

Creating and calling a method

```
let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  myMethod: function() {
    console.log("I'm from a method!");
  }
};

/* we can call our own methods the same way we call built-in methods */
graceHopper.myMethod();
```

It's easier to ask forgiveness than it is to get permission.

Creating and calling a method

```
/* methods on an object generally should perform an action that's relevant
   to the idea or concept represented by the object itself */
let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  sayQuote: function() {
    console.log("It's easier to ask forgiveness than it is to get
    permission.");
  }
};

graceHopper.sayQuote();
```

Creating and calling a method

```
let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  sayQuote: function() {
    console.log("It's easier to ask forgiveness than it is to get
    permission.");
  },
  getAge: function(year) {
    return year - 1906;
  }
};

console.log(graceHopper.getAge(2018));
```

Hi Karen I am Rear Admiral Hopper

Creating and calling a method

```
/* let's give Grace a greet method */

let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  greet: function(name) {
    console.log('Hi', name, 'I am Rear Admiral Hopper')
  }
};

/* greet will work, but it's a bit redundant to type out Grace's rank
   and last name twice in the same object */
graceHopper.greet('Karen');
```

```
Hi Gabe I am Rear Admiral Hopper
```

Creating and calling a method

```
/* methods frequently reference other properties in the same object */

/* this could work: */

let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  greet: function(name) {
    console.log('Hi', name, 'I am', graceHopper.rank, graceHopper.last)
  }
};

graceHopper.greet('Gabe');
```

```
Hi Kate I am Rear Admiral Hopper
```

Introduction to this

```
/* JS has a keyword, this, you can use to reference the object in a
   method */

let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  greet: function(name) {
    console.log('Hi', name, 'I am', this.rank, this.last)
  }
};

graceHopper.greet('Kate');
```

Grace
true

Introduction to this

```
/* this is literally just another reference to the graceHopper object */

let graceHopper = {
  first: 'Grace',
  last: 'Hopper',
  rank: 'Rear Admiral',
  getThis: function() {
    return this;
  }
};

let returnedObject = graceHopper.getThis();
console.log(returnedObject.first);
console.log(returnedObject === graceHopper);
```

50
5

Example: calc

```
let calc = {
  num1: 20,
  num2: 30,
  sum: function() {
    return this.num1 + this.num2;
  },
  difference: function() {
    return this.num1 - this.num2;
  }
};

console.log(calc.sum());
calc.num2 = 15;
console.log(calc.difference());
```

Disclaimer!

```
/* there is so much more to 'this'! */

/* JS is an object-oriented language, which means objects and their
   methods play a key role in most production JS code bases */

/* we're only skimming the surface of this and object methods in this
   course; lots more to learn as you progress!  */
```

Recap

```
/*
  - What is a method?
  - Creating, accessing, running a method
  - Introduction to this
*/
```