Object Methods

Overview

```
1   /*
2     - What is a method?
3     - Creating, accessing, running a method
4     - Introduction to ‘this’
5   */
6
7
8
9
10
11
12
13
14
```

What is a method?

```
1   /* Methods are actions you can perform on a value */
2
3   /* We've been using methods throughout this course, like .indexOf */
4
5   /* these methods are built into JS; we don't have to define them
6      ourselves */
7
8   let countries = ['Argentina', 'Bolivia', 'Brazil', 'Chile'];
9   console.log(countries.indexOf('Brazil'));
10
11
12
13
14
```

Creating a method

```
1   /* We can create our own methods too! */
2
3   /* Consider this object: */
4
5   let graceHopper = {
6     first: 'Grace',
7     last: 'Hopper',
8     rank: 'Rear Admiral'
9   };
10
11  /* objects can hold any type of value, including functions! */
12
13  /* if we add a function to this object, that function is now a method of
14     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();
```

Creating and calling a method

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

```
1   /* methods on an object generally should perform an action that's relevant
2      to the idea or concept represented by the object itself */
3   let graceHopper = {
4     first: 'Grace',
5     last: 'Hopper',
6     rank: 'Rear Admiral',
7     sayQuote: function() {
8       console.log("It's easier to ask forgiveness than it is to get
9       permission.");
10    }
11  };
12
13  graceHopper.sayQuote();
14
```

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(2020));
```

Hi Karen I am Rear Admiral Hopper

Creating and calling a method

```
1   /* let's give Grace a greet method */
2
3   let graceHopper = {
4     first: 'Grace',
5     last: 'Hopper',
6     rank: 'Rear Admiral',
7     greet: function(name) {
8       console.log('Hi', name, 'I am Rear Admiral Hopper')
9     }
10  };
11
12  /* greet will work, but it's a bit redundant to type out Grace's rank
13     and last name twice in the same object */
14  graceHopper.greet('Karen');
```

Hi Gabe I am Rear Admiral Hopper

Creating and calling a method

```
1   /* methods frequently reference other properties in the same object */
2
3   /* this could work: */
4
5   let graceHopper = {
6     first: 'Grace',
7     last: 'Hopper',
8     rank: 'Rear Admiral',
9     greet: function(name) {
10      console.log('Hi', name, 'I am', graceHopper.rank, graceHopper.last)
11    }
12  };
13
14  graceHopper.greet('Gabe');
```

Hi Kate I am Rear Admiral Hopper

Introduction to ‘this’

```
1   /* JS has a keyword, this, you can use to reference the object in a
2      method */
3
4   let graceHopper = {
5     first: 'Grace',
6     last: 'Hopper',
7     rank: 'Rear Admiral',
8     greet: function(name) {
9       console.log('Hi', name, 'I am', this.rank, this.last)
10    }
11  };
12
13  graceHopper.greet('Kate');
14
```

Grace
true

Introduction to this

```
1   /* this is literally just another reference to the graceHopper object */
2
3   let graceHopper = {
4     first: 'Grace',
5     last: 'Hopper',
6     rank: 'Rear Admiral',
7     getThis: function() {
8       return this;
9     }
10  };
11
12  let returnedObject = graceHopper.getThis();
13  console.log(returnedObject.first);
14  console.log(returnedObject === graceHopper);
```

Example: calc

```
1   let calc = {
2     num1: 20,
3     num2: 30,
4     sum: function() {
5       return this.num1 + this.num2;
6     },
7     difference: function() {
8       return this.num1 - this.num2;
9     }
10  };
11
12  console.log(calc.sum());
13  calc.num2 = 15;
14  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
*/
```

Review

Creating an object

```
let pusheen = {
    name: 'Pusheen',
    age: 7,
    color: 'gray and tabby'
};
```

Accessing a value

```
Pusheen
Pusheen
7
gray and tabby
undefined
```

```
1   /* use bracket notation and dot notation to access a value */
2
3   /* pass a string into the brackets that corresponds with a key in the object */
4   let pusheen = {
5     name: 'Pusheen',
6     age: 7,
7     color: 'gray and tabby'
8   };
9
10  console.log(pusheen['name']);
11  console.log(pusheen.name);
12  console.log(pusheen['age']);
13  console.log(pusheen['color']);
14  console.log(pusheen['notAKeyInTheObject']);
```

Changing a value

```
1   /* use bracket notation or dot notation to change a value */
2
3   let pusheen = {
4     name: 'Pusheen',
5     age: 7,
6     color: 'gray and tabby'
7   };
8
9   pusheen['age'] = 8;
10  pusheen.age++ // same as saying pusheen.age += 1
11
12  console.log(pusheen.age);
13
14
```

{ name: Pusheen }

Deleting a key/value pair

```
1   /* use the delete keyword to delete a key/value pair */
2
3   let pusheen = {
4     name: 'Pusheen',
5     age: 7,
6     color: 'gray and tabby'
7   };
8
9   delete pusheen['age'];
10  delete pusheen.color;
11
12  console.log(pusheen);
13
14
```

in operator

```
1   /* use the in operator to check if a key is in the object */
2
3   let pusheen = {
4     name: 'Pusheen',
5     age: 7,
6     color: 'gray and tabby'
7   };
8
9   console.log('name' in pusheen);
10  console.log('sadness' in pusheen);
11
12
13
14
```

```
true
false
```

for...in loop

```
Pusheen's name is Pusheen
Pusheen's age is 7
Pusheen's color is gray and tabby
```

```
1   /* use the for…in loop to loop through all of the keys in an object */
2
3   let pusheen = {
4     name: 'Pusheen',
5     age: 7,
6     color: 'gray and tabby'
7   };
8
9   for (let key in pusheen) {
10    console.log("Pusheen's", key, 'is', pusheen[key]);
11  }
12
13
14
```

[name, age, color]

Object.keys()

```
1   /* use Object.keys() to get an array of the keys in the object */
2
3   let pusheen = {
4     name: 'Pusheen',
5     age: 7,
6     color: 'gray and tabby'
7   };
8
9   console.log(Object.keys(pusheen));
10
11
12
13
14
```

nested arrays

```
/* objects can store any type of value, including arrays and other
   objects */

let pusheen = {
  name: 'Pusheen',
  age: 7,
  colors: ['gray', 'tabby']
};

console.log(pusheen.colors[0]);
console.log(pusheen.colors[1]);
```

```
gray
tabby
```

Stormy
Pip

nested objects

```
1   /* objects can store any type of value, including arrays and other
2      objects */
3
4   let pusheen = {
5     name: 'Pusheen',
6     age: 7,
7     siblings: {
8       sister: 'Stormy',
9       brother: 'Pip'
10    }
11  };
12
13  console.log(pusheen.siblings.sister);
14  console.log(pusheen.siblings.brother);
```

Hi Kate I am Rear Admiral Hopper

Introduction to ‘this’

```
1   /* JS has a keyword, this, you can use to reference the object in a
2      method */
3
4   let graceHopper = {
5     first: 'Grace',
6     last: 'Hopper',
7     rank: 'Rear Admiral',
8     greet: function(name) {
9       console.log('Hi', name, 'I am', this.rank, this.last)
10    }
11  };
12
13  graceHopper.greet('Kate');
14
```