Hello

Twitter

@Codersinhoods

Slack

Youtube

/Codersinhoods

HELLO!

I am Sam Barker

- *Lead Instructor at Flatiron School*

samuel.e.barker@gmail.com

Coders In Hoods World

2900+

In Twitter

530+ users

In slack

20

Time zones

Change Your life

Our main goal is to help people change their lives, and make software education affordable for everyone.

JavaScript

Lesson 5

ES6

=> destructuring with ES6
=> spread operator in ES6
=> Dynamic object keys
=> Default argument values
=> Classes in ES6 (OOP)
=> Class Inheritance

Destructuring

Destructuring *is a way of extracting multiple values from data stored in (possibly nested)* ***objects*** *and* ***Arrays****. It can be used in locations that receive data (such as the left-hand side of an assignment).*

Array Destructuring

ES5 - the old way!

```
1 const animals = ['cat', 'rabbit', 'dog', 'giraffe'];
2 const firstAnimal = animals[0]
3 const secondAnimal = animals[1]
4
5 console.log(firstAnimal, secondAnimal) // 'cat rabbit'
```

ES6

```
1 const animals = ['cat', 'rabbit', 'dog', 'giraffe'];
2 const [firstAnimal, secondAnimal] = animals;
3
4 console.log(firstAnimal, secondAnimal) // 'cat rabbit'
```

If you need `nth` element from array

```
1 const animals = ['cat', 'rabbit', 'dog', 'giraffe'];
2 const [,, thirdAnimal] = animals;
3
4 console.log(thirdAnimal) // 'dog'
```

Nested Array Destructuring

```
1 const myArray = ['cat', [23, 54, 234], 'dog', 'giraffe'];
2 const [, [, secondNumber], ,forthAnimal] = myArray;
3
4 console.log(secondNumber, forthAnimal) // `54 "giraffe"`
```

Object Destructuring

ES5

```
const student = {
    firstName: 'Alex',
    lastName: 'Smith',
    country: 'England'
};

const studentName = student.firstName;
const studentLastName = student.lastName;
const studentCountry = student.country;
```

ES6

```
1 const student = {
2     firstName: 'Alex',
3     lastName: 'Smith',
4     country: 'England'
5 };
6 const { firstName, lastName, country } = student;
7 console.log(firstName, lastName, country) // Alex Smith England
```

Default Values on destructuring

*Assign a variable corresponding to a **key** that does not exist on the destructured **object** will cause the **value undefined** to be assigned instead. You can pass **default values** that will be assigned to such variables instead of **undefined***

Example

```
1 const student = {
2     firstName: 'Alex',
3     lastName: 'Smith',
4     country: 'England'
5 };
6 const { firstName, lastName, age = 20 } = student;
7 console.log(firstName, lastName, age) // Alex Smith 20
```

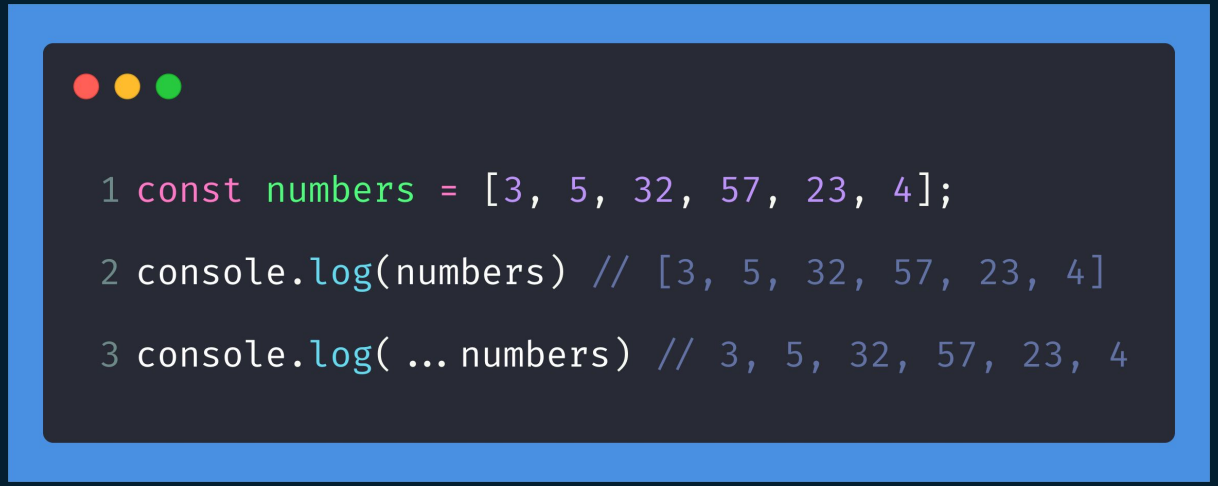
Reassign to a new name

```
1 const student = {
2     firstName: 'Alex',
3     lastName: 'Smith',
4     country: 'England'
5 };
6 const { firstName: name, lastName: surname } = student;
7 console.log(name, surname) // Alex Smith
```

Spread operator

*"**Spread** syntax allows an **iterable** such as an **array** expression or string to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an **object** expression to be expanded in places where zero or more **key-value pairs** (for object literals) are expected."*

– MDN


1 const numbers = [3, 5, 32, 57, 23, 4];
2 console.log(numbers) // [3, 5, 32, 57, 23, 4]
3 console.log(... numbers) // 3, 5, 32, 57, 23, 4

Shallow clone

arrays are reference types and hence they are assigned by reference instead of being copied.

```
1 const rainbow = ['red', 'orange', 'yellow', 'green', 'blue']
2 const newRainbow = rainbow
3 newRainbow == rainbow // TRUE
4 newRainbow === rainbow // TRUE
5
6 const oneMoreRainbow = [...rainbow];
7 oneMoreRainbow == rainbow // FALSE
8 oneMoreRainbow === rainbow // FALSE
```

Combine arrays

```
1 const numbers = [3, 5, 32, 57, 23, 4];
2 const moreNumbers = [4, 5, 7, 1];
3 const combinedNumbers = [ ... numbers,  ... moreNumbers];
4 console.log(combinedNumbers) // [3, 5, 32, 57, 23, 4, 4, 5, 7, 1]
```

Combine objects

```
1 const person = {
2     firstName: 'Alex',
3     lastName: 'Smith'
4 }
```

```
1 const user = {
2     ...person,
3     subscription: 'premium'
4 }
```

```
1 console.log(user);
2 // {firstName: "Alex", lastName: "Smith", subscription: "premium"}
```

Destructuring + Spread =

Rest

Rest Items

```
1 const rainbow = ['red', 'orange', 'yellow', 'green', 'blue'];
2 const [,orange, ...otherColors] = rainbow;
3
4 console.log(orange, otherColors);
5 // "orange ['yellow', 'green', 'blue']"
```

Rest properties

```
1 const student = {
2     firstName: 'Alex',
3     lastName: 'Smith',
4     country: 'England'
5 };
6 const { firstName, ...other } = student;
7 console.log(firstName, other)
8 // Alex {lastName: "Smith", country: "England"}
```

Dynamic object keys

You already know

```
const user = {
    firstName: 'Alex',
    lastName: 'Smith',
    age: 25,
    favouriteColor: 'green'
};

console.log(user.firstName) // Alex
console.log(user['firstName']) // Alex
```

Pass your key dynamically

```
1 const user = {
2     firstName: 'Alex',
3     lastName: 'Smith',
4     age: 25,
5     favouriteColor: 'green'
6 };
7
8 Object.keys(user).forEach(key ⇒ {
9     console.log(user[key])
10 })
```

Get selected properties

```
const user = {
    firstName: 'Alex',
    lastName: 'Smith',
    age: 25,
    favouriteColor: 'green'
};
const requiredProperties = ['firstName', 'age'];

requiredProperties.forEach(property => {
    console.log(user[property])
})
```

We can also use [ ] to set a key

```
const user = {
  firstName: 'Alex',
  lastName: 'Smith',
  age: 25,
  favouriteColour: 'green'
};

const keyToSet = 'height';
const valueToSet = 200;

const updatedUser = {
  ...user,
  [keyToSet]: valueToSet
};
```

Default argument values

Problem

```
1 const sum = (a, b) ⇒ {
2     return a + b;
3 }
4
5 sum(); // NaN
6 sum(2); // NaN
7 sum(2,3) // 5
```

Solution

```
1 const sum = (a = 0, b = 0) ⇒ {
2     return a + b;
3 }
4
5 sum(); // 0
6 sum(2); // 2
7 sum(2,3) // 5
```

Classes

Object Oriented Programming *(OOP) refers to using self-contained pieces of code to develop applications. We call these self-contained pieces of code objects, better known as Classes.*

```
// Class declaration
class Person {
    constructor(firstName, lastName, age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }
    getFullName() {
        return `${this.firstName} ${this.lastName}`;
    };
}
// Create new instance
const person1 = new Person('Alex', 'Smith', 25);
console.log(person1.getFullName());
```

Object Oriented Programming
Inheritance

Inheritance *refers to an object being able to inherit methods and properties from a parent object.*

Step 1

```
class Person {
    constructor(firstName, lastName, age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }
    getFullName() {
        return `${this.firstName} ${this.lastName}`;
    };
}
```

Step 2

```
class Student extends Person {
    constructor(firstName, lastName, age, uniName, averageMark = 0) {
      super(firstName, lastName, age);
      this.uniName = uniName;
      this.averageMark = averageMark;
    }
    getUniName() {
        return this.uniName;
    };
    getAvarageMark() {
        return this.averageMark;
    };
    setAvarageMark(mark = 0) {
        this.averageMark = mark;
    };
}
```

Result

You have access to all properties from both Classes

```
1 const student = new Student('Alex', 'Smith', 25, 'Oxford');
2 student.setAvarageMark(5);
3 student.getAvarageMark(); // 5
4 student.getFullName() // "Alex Smith"
```

Exercise

1. ***Create a class Dog with properties:***
   => numberOfLegs
   => height
   => weight
2. ***Create a class Pet which the Dog class extends.***
3. ***Add to class Pet properties like:***
   => name
   => mood
   => method to get all data about your pet
   => method to get just name
   => method to get name and mood.
4. ***Create an Instance of Pet for testing.***

What's next?

~~**Lesson 0 — JavaScript basics: Part 1**~~ 

~~**Lesson 1 — JavaScript basics: Part 2**~~

~~**Lesson 2 — DOM manipulation**~~

~~**Lesson 3 — Let's build a Project**~~

~~**Lesson 4 — Asynchronous Javascript**~~

~~**Lesson 5 — Classes in javascript**~~

Lesson 6 — Final Project.

THANKS!

Twitter

@Codersinhoods

Slack

Youtube

/Codersinhoods