Codesmith

CSPrep

Day 3

Overview

POD

The “new” keyword & classes

Functions

- Arguments vs. parameters
- Arguments object
- Spread syntax
- Rest parameters

Review/Q&A: Units 1 and 2 challenges

Pair Programming

- What if we could automate the creation of the newUser object?
- We can… using the “new” keyword!

It will

- Automatically create an object for us
- Automatically create the __proto__ link.. But to what?
- Automatically return that object out

```
32
33  function userCreator(name, score) {
34    const newUser = Object.create(userFunctionStore);
35    newUser.name = name;
36    newUser.score = score;
37    return newUser;
38  }
39
40  const userFunctionStore = {
41    increment: function () { this.score++ },
42    login: function () { console.log('Logged in!') }
43  }
44
45  const user1 = userCreator('Dulio', 3);
46  const user2 = userCreator('Kedon', 5);
47  user1.increment();
```

It will

- Automatically create an object for us
- Automatically create the __proto__ link..

  But to what?
- Automatically return that object out

```
39  function userCreator(name, score) {
40    const newUser = Object.create(userFunctionStore);
41    newUser.name = name; this.name = name;
42    newUser.score = score; this.score = score;
43    return newUser;
44  };
45
46  functionStore
47   userCreator.prototype
48   userCreator.prototype.increment = function() {
49       this.score++;
50   }
51
52
53  const user1 = new userCreator('Bree', 3);
54
55
```

Functions in javascript are both functions AND objects

```
49
50  function multiplyByTwo(num) {
51      return num * 2;
52  }
53
54  multiplyByTwo.stored = 5;
55  multiplyByTwo(3);
56
57  console.log(multiplyByTwo.stored);
58  console.log(multiplyByTwo.prototype);
59
```

```
61
62  function userCreator(name, score) {
63    this.name = name;
64    this.score = score;
65  }
66
67  userCreator.prototype.increment = function() { this.score++ };
68  userCreator.prototype.login = function() { console.log('logged in')};
69
70  const user1 = new userCreator('Rajeeb', 5);
71  user1.increment();
```

```
61
62  function userCreator(name, score) {
63    this.name = name;
64    this.score = score;
65  }
66
67  userCreator.prototype.increment = function() { this.score++ };
68  userCreator.prototype.login = function() { console.log('logged in')};
69
70  const user1 = new userCreator('Rajeeb', 5);
71  user1.increment();
```

```
61
62  function UserCreator(name, score) {
63    this.name = name;
64    this.score = score;
65  }
66
67  UserCreator.prototype.increment = function() { this.score++ };
68  UserCreator.prototype.login = function() { console.log('logged in')};
69
70  const user1 = new UserCreator('Rajeeb', 5);
71  user1.increment();
72
```

The ‘class’ keyword

```
74  class UserCreator {
75    constructor(name, score) {
76      this.name = name;
77      this.score = score;
78    }
79    increment() {
80      this.score++;
81    }
82    login() {
83      console.log('logged in');
84    }
85  }
86
87  const user1 = new UserCreator('Edward', 5);
88  user1.increment();
89
```

Arguments vs. Parameters

Parameters: variables listed in a function definition

```
function sumTwoNums(a, b) {
    return a + b
}
```

Arguments: values passed into a function when its invoked

```
sumTwoNums(1, 3);
```

What happens if you pass a different number of arguments into a function from the number of defined parameters?

```
sumTwoNums(1, 3, 5)
????
```

Arguments Object

How do you handle an unknown number of arguments in a function?

```
function sumNums() {
    // Where are my parameters!?
}

sumNums(1, 2);
sumNums(1, 2, 3);
```

Arguments Object (continued)

MDN: “The arguments object is an Array-like object corresponding to the arguments passed to a function”

- NOT an array
- Can refer to arguments [0] , arguments [1], etc.
- Has length property… and that’s it
- Available in non-arrow functions

ES6+ Spread Syntax

Three dots: ...

Allows you to use a single variable name to represent more items (e.g. array elements or object)

```
const arr1 = ['first', 'second'];
const arr2 = ['third', 'fourth'];
const allFour = [ ...arr1, ...arr2 ];

console.log(allFour) //['first', 'second', 'third', 'fourth']
```

ES6+ Spread Syntax continued

```
const coupons = { "4THOFJULY": 2, "NEWCUSTOMER": 5, "BESTFRIEND": 7 }

function getMaxReduce() {

  return Object.values(coupons).reduce((a, b) => Math.max(a, b));

}

vs.

function getMax() {

  return Math.max(...Object.values(coupons));

}
```

ES6+ Rest Parameters

MDN: “The rest parameters syntax allow us to represent an indefinite number of arguments as an array”

```
function restIsGreat(...args) {
  console.log(args);
}

restIsGreat(1, 'hello', true); // logs: [ 1, 'hello', true ]
```

Review / Q&A / Pairing