avionschool

Lesson 11.0: Objects and Properties

BATCH 4

DECEMBER 18, 2020

Objects: Anatomy

DEFINITION

- Objects hold a lot of information about one thing, while arrays hold multiple items related to each other.
- Objects are instantiated by using braces ({})
- Inside the braces, there is a list of properties separated by commas. Each property has a name followed by a colon and a value.
- Properties whose names aren’t valid binding names or valid numbers have to be quoted.
- Values of the type object are arbitrary collections of properties.

CREATE-ing Objects

{ } expression

{ } expression

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}
```

{ } expression

- This means that braces have two meanings in JavaScript. At the start of a statement, they start a block of statements. In any other position, they describe an object.

Object() constructor

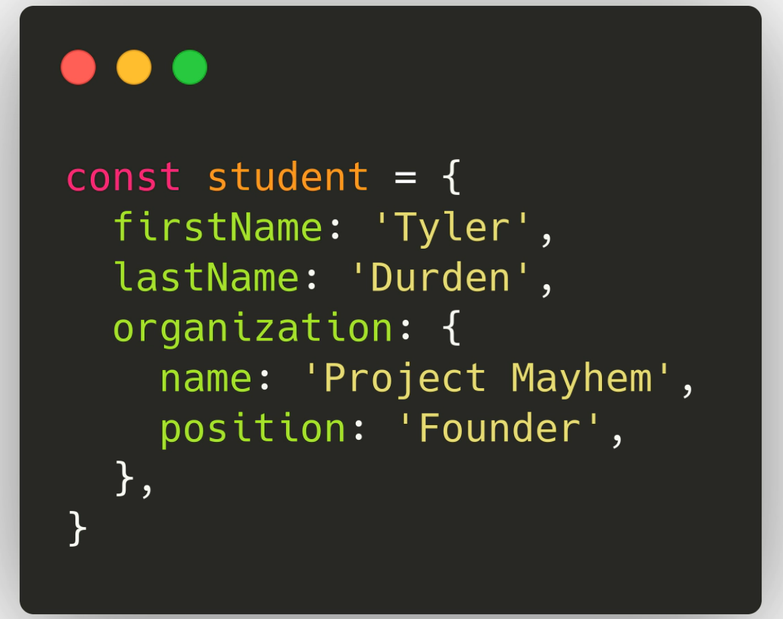
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

```
let student = Object()

let student = Object(1)

let student = Object('Tyler Durden')

let student = Object([
 'Durden, Tyler',
])

let student = Object({
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
})
```

READ-ing Objects

Key: value pairs

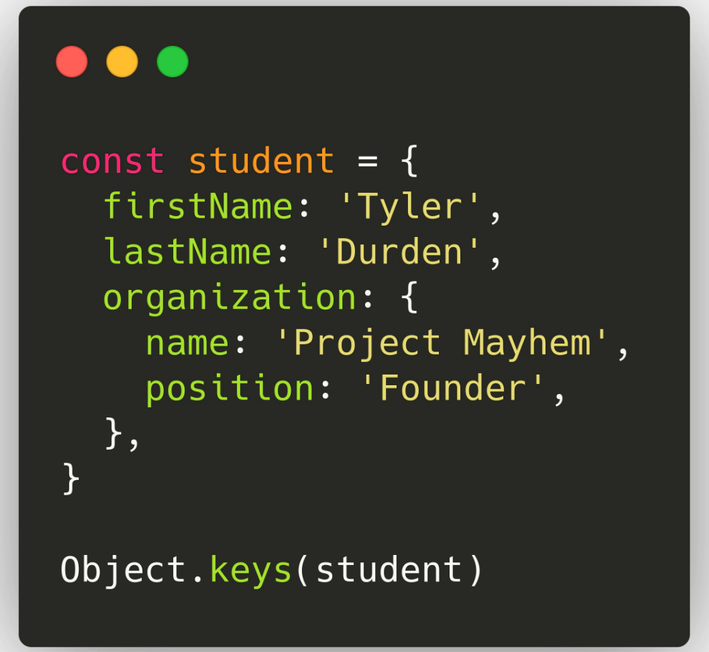
- Unlike arrays that have index valued items, objects use a concept called key: value pairs.
- The key is the value we want to save to that key.
- To find out what properties an object has, you can use the Object.keys function.



```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}
```

Object.keys()

- To find out what properties an object has, you can use the Object.keys function. You give it an object, and it returns an array of strings—the object's property names.



```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

Object.keys(student)
```

Accessing values

- Once we have key: value pairs we can access those values by calling the object name and the key.
- There are two ways to do this, dot notation and bracket notation.
- Reading a property that doesn't exist will give you the value undefined.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

student.firstName === student['firstName'] // true
student.lastName === student['lastName'] // true
student.organization === student['organization'] // true
student.organization.name === student['organization']['name'] // true
```

UPDATE-ing Objects

Assigning values

- Assigning values works just like accessing them.
- We can assign them, when we create the object, with dot notation, or with bracket notation:
- It is possible to assign a value to a property expression with the = operator.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

student.firstName = "Marla"
student['lastName'] = "Singer"
student.organization['position'] = "Member"
```

Methods

- In objects, values can be set to functions. Functions saved on an object are called methods.
- We can set a key to a name, and the value to a function.
- Just like other times we call methods, we will call this method using dot notation and trailing parenthesis.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

function recruitMembers (location) {
  console.log(`Recruiting members ${location ? `in ${location}` : '...'}`)
}

student.assignment = recruitMembers
```

this keyword

DEFINITION

- Objects have a self referential keyword that may be applied in each object called `this`. When called inside of an object it is referring to that very object.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

function recruitMembers (location) {
  console.log(`${this.firstName} is recruiting members ${location ? `in ${location}` : '...'}`)
}

student.assignment = recruitMembers
```

Object.assign()

- Objects have a self referential keyword that may be applied in each object called `this`. When called inside of an object it is referring to that very object.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

function recruitMembers (location, member) {
  console.log(this)
  let newMember = null
  if (member) {
    newMember = Object.assign({}, {
      firstName: member.split(' ')[0],
      lastName: member.split(' ')[1],
      organization: this.organization,
    })

    newMember.organization.position = 'Member'
    console.log(`${this.firstName} recruited ${member} ${location ? `in ${location}` : '...'}`)
  } else {
    console.log(`${this.firstName} is recruiting members ${location ? `in ${location}` :
'...'}`)
  }

  return newMember
}



student.assignment = recruitMembers

let newStudent = student.assignment('New York', 'Marla Singer')
```

DELETE-ing Objects

delete Operator

- The difference between setting a property to undefined and actually deleting it is that, in the first case, the object still has the property (it just doesn't have a very interesting value), whereas in the second case the property is no longer present and in will return false.

```
const student = {
  firstName: 'Tyler',
  lastName: 'Durden',
  organization: {
    name: 'Project Mayhem',
    position: 'Founder',
  },
}

student.organization = undefined

delete student.organization
```

JSON

JavaScript Object Notation

- https://www.json.org/json-en.html
- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array. An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.
- https://jsonformatter.org/