Objects, JSON and Prototypes

Mark Zamoyta

markzamoyta@gmail.com

Overview

Simple Objects and JSON

Understanding Prototypes

Working With Prototypes

Object.create()

Object. defineProperty()

Miscellaneous Object Functions

Simple Objects and JSON

```
var project = new Object();
project.name = 'Project Phoenix';
console.log(project.name);
```

Question

What shows in the console?

Answer

```
Project Phoenix
```

```
var project = new Object();
project.name = 'Project Phoenix';
project.securityLevel = 15;
project.updateDueDate = function () {
    return true;
};
console.log(project.updateDueDate());
```

Question

What shows in the console?

Answer

```
true
```

```
var project = new Object();
project.name = 'Project Phoenix';
project.securityLevel = 15;
project.updateDueDate = function () {
    return true;
};
console.log(project['securityLevel']);
```

Question

What shows in the console?

Answer

15

```
var project = new Object();
project.name = 'Project Phoenix';
project.securityLevel = 15;
project.updateDueDate = function () {
    return true;
};
var field = 'securityLevel';
console.log(project[field]);
```

Question

What shows in the console?

Answer

15

```
var project = {};
project.name = 'Project Phoenix';
console.log(project.name);
```

Question

What shows in the console?

Answer

```
Project Phoenix
```

```
var project = {
    name: 'Project Phoenix',
    securityLevel: 15,
    updateDueDate: function () {
        return true;
    }
};
console.log(project.name);
```

Question

What shows in the console?

Answer

```
Project Phoenix
```

```
var project = {
    name: 'Project Phoenix',
    securityLevel: 15,
    updateDueDate: function () {
        return true;
    },
    team: ['iguanaboy', 'cactusgirl',
           'Buffy']
};
console.log(project.team[2]);
```

Question

What shows in the console?

Answer

```
Buffy
```

```
var project = {
    tasks: [
        {
            taskName: 'first'
        },
        {
            taskName: 'second'
        }
    ]
};
console.log(project.tasks[1].taskName);
```

Question

What shows in the console?

Answer

```
second
```

Understanding Prototypes

Every JavaScript Object has a prototype property.

However, we don't always have access to it.

The prototype property is simply an object.

```
var project = anyObject;
project.someFunction();
```

How does JavaScript locate / resolve someFunction?

- `project.someFunction()`
- `project.prototype.someFunction()`
- `project.prototype.prototype.someFunction()`
- `project.prototype.prototype.prototype.someFunction()`
- `etc.`

In most browsers, we can access the prototype property by using the symbol

__proto__ .

(not safe!)

The prototype property is most easily accessed on constructor functions.

(Covered in the functions module of this course)

Working with Prototypes

```
var project = {
    name: 'Project Phoenix'
};
console.log(project.foo());
```

Question

What shows in the console?

Answer

```
TypeError: project.foo
is not a function
```

```
var project = {
    name: 'Project Phoenix'
};
console.log(project.toString());
```

Question

What shows in the console?

Answer

```
[object Object]
```

```
console.log(typeof Object.prototype);
```

Question

What shows in the console?

Answer

```
object
```

```
console.log(typeof Object.prototype.toString);
```

Question

What shows in the console?

Answer

```
function
```

```
var project = {
    name: 'Project Phoenix'
};
console.log(project.prototype);
```

Question

What shows in the console?

Answer

```
undefined
```

```
var project = {
    name: 'Project Phoenix'
};
console.log(typeof project.__proto__);
```

Question

What shows in the console?

Answer

```
object
```

```
var project = {
    name: 'Project Phoenix'
};
console.log(project.__proto__ === Object);
```

Question

What shows in the console?

Answer

```
false
```

```
var project = {
    name: 'Project Phoenix'
};
console.log(project.__proto__ === Object.prototype);
```

Question

What shows in the console?

Answer

```
true
```

Object.create() and Prototypes

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(secretProject.securityLevel);
```

Question

What shows in the console?

Answer

2

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(typeof secretProject.toString);
```

Question

What shows in the console?

Answer

```
function
```

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(secretProject.__proto__ === Object.prototype);
```

Q What shows in the console?

A `false`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(secretProject.__proto__.__proto__ === Object.prototype);
```

Q What shows in the console?

A `true`

Object.defineProperty()

```
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!'
});
console.log(task.text);
```

Question

What shows in the console?

Answer

```
Get this job done!
```

```javascript
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!',
    writable: true
});
task.text = task.text + ' ... NOW!';
console.log(task.text);
```

Question

What shows in the console?

Answer

```
Get this job done! ...
NOW!
```

```
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!'
    // writable: true
});
task.text = task.text + ' ... NOW!';
console.log(task.text);
```

Question

What shows in the console?

Answer

```
TypeError: Cannot
assign to read only
property...
```

```
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!',
    enumerable: true
});
for (var f in task)
    console.log(f);
```

Question

What shows in the console?

Answer

```
text
```

```
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!'
    //enumerable: true
});
for (var f in task)
    console.log(f);
```

Question

What shows in the console?

Answer

```
(nothing shows)
```

```javascript
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!',
    configurable: false
});
Object.defineProperty(task, 'text', {
    value: 'Done!'
});
console.log(task.text);
```

Question

What shows in the console?

Answer

```
TypeError: Cannot
redefine property: text
```

```
'use strict';
var task = { };
Object.defineProperty(task, 'text', {
    value: 'Get this job done!',
    configurable: true
});
Object.defineProperty(task, 'text', {
    value: 'Done!'
});
console.log(task.text);
```

Question

What shows in the console?

Answer

Done!

```javascript
'use strict';
var task = {
    _dueDate: '1/15/16'
};
Object.defineProperty(task, 'dueDate', {
    get: function () {
        return this._dueDate;
    }
});
console.log(task.dueDate);
```

Question

What shows in the console?

Answer

1/15/16

```
'use strict';
var task = {
    _dueDate: '1/15/16'
};
Object.defineProperty(task, 'dueDate', {
    get: function () {
        return this._dueDate;
    },
    set: function (newValue) {
        this._dueDate = newValue;
    }
});
task.dueDate = '2/2/16';
console.log(task.dueDate);
```

Question

What shows in the console?

Answer

2/2/16

```
'use strict';
var task = { };
Object.defineProperties(task, {
    'text': {
        value: 'New Task'
    },
    'dueDate': {
        value: '1/15/16'
    }
});
console.log(task.text + ' Due: ' +
   task.dueDate);
```

Question

What shows in the console?

Answer

```
New Task Due: 1/15/16
```

```
'use strict';
var task = { };
Object.defineProperties(task, {
    'text': {
        value: 'New Task'
    }
});
var descriptor = Object.getOwnPropertyDescriptor(task, 'text');
console.log(descriptor);
```

Q What shows in the console?

A

```
Object {value: "New Task", writa
ble: false, enumerable: false, c
onfigurable: false}
```

Miscellaneous Object Functions

```
'use strict';
var project = {
    name: 'Top Secret Project',
    dueDate: '1/1/2016'
};
console.log(project.hasOwnProperty('name'));
```

Question

What shows in the console?

Answer

```
true
```

```javascript
'use strict';
var project = {
    name: 'Top Secret Project',
    dueDate: '1/1/2016'
};
console.log(project.hasOwnProperty('toString'));
```

Question

What shows in the console?

Answer

```
false
```

```
'use strict';
var project = {
    name: 'Top Secret Project',
    dueDate: '1/1/2016'
};
console.log(project.__proto__.hasOwnProperty('toString'));
```

Q What shows in the console?

A `true`

```
'use strict';
var project = {
    name: 'Top Secret Project',
    dueDate: '1/1/2016'
};
console.log(Object.prototype.isPrototypeOf(project));
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(project.isPrototypeOf(secretProject));
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(Object.prototype.isPrototypeOf(secretProject));
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log(project.__proto__.isPrototypeOf(secretProject));
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log("securityLevel" in secretProject);
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log("toString" in secretProject);
```

Q What shows in the console?

A `true`

```
var project = {
    securityLevel: 2
};
var secretProject = Object.create(project);
console.log("securityLevel" in Object.prototype);
```

Q What shows in the console?

A `false`

Summary

JavaScript

- Simple Objects and JSON
- Understanding Prototypes
- Working With Prototypes
- Object.create()
- Object.defineProperty()
- Miscellaneous Object Functions