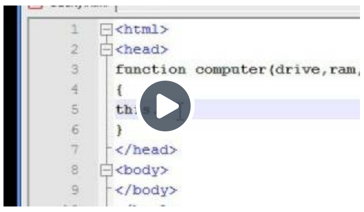
Day 4: Create a Rectangle Object

by AvmnuSng

Objects in JavaScript



```
<html>
<head>
function computer(drive,ram,
{
th
}
</head>
<body>
</body>
```

Object Basics

We define the following:

- *Object:* A collection of properties.
- *Property:* An association between a *name* (i.e., *key*) and a *value*. Note that when the value associated with a key is a function, we call the property a *method*. A property name can be any valid string, or anything that can be converted into a string (including the empty string).

An object has properties associated with it, and we explain an object's properties as variables that are part of the object. We can think of an object's properties as a set of regular variables specific to that object that define its characteristics.

Let's say we have an object named $objectName$ and a property named $propertyName$. We can access this property in the following ways:

1. *Dot Notation:* Call `objectName.propertyName`.
2. *Bracket Notation:* Call `objectName['propertyName']`. Note that $propertyName$ must be enclosed in string quotes and is *case-sensitive*. Any property name that's not a valid JavaScript identifier (e.g., starts with a number, contains a space or hyphen, etc.) can only be accessed using bracket notation. This type of notation is also good to use when property names are dynamically determined (i.e., not known until runtime).

We can *add a new property to an existing object* by assigning a value to it using either dot or bracket notation (see the example below).

- EXAMPLE

```
/*
 * Create an object with two properties, 'Name' and 'Age'
 */
var actor = {
    Name: 'Julia Roberts',
    Age: 36
};

// Print the object
console.log('The \'actor\' object:', actor);

// Access object's properties using bracket notation
console.log('The \'Name\' property:', actor['Name']);
console.log('The \'Age\' property:', actor['Age']);

// Access object's properties using dot notation
console.log('The \'Name\' property:', actor.Name);
console.log('The \'Age\' property:', actor.Age);

// Add a new property called 'EyeColor'
actor.EyeColor = 'Brown';

// Print the object
console.log('The updated \'actor\' object:', actor);

```

```
// Trying to access undefined property results in 'undefined'
console.log('Attempt to access an undefined property (\'HairColor\')
    actor.HairColor);
```

Output

Run

Creating Objects

We can create objects using an *object initializer*, or we can first create a *constructor function* and then instantiate an object using that function's name in conjunction with the *new* operator.

1. Using Object Initializers

We can initialize an object using `new Object()`, `Object.create()`, or by using the *literal* (or *initializer*) notation. An object initializer is a comma-separated list of zero or more property name-value pairs defining an object, enclosed in curly braces (i.e., `{}`).

- EXAMPLE

Using Initializer Notation

```
var a = 3;
var b = 'Rome';
var c = false;

var o = {a, b, c};

console.log('Object \'o\':', o);

var p = {
    a: 3,
    b: 'Rome',
    c: false
};

console.log('Object \'p\':', p);

var q = {};
console.log('Object \'q\' (Initial):', q);
q.a = a;
q.b = b;
q.c = c;
console.log('Object \'q\' (Updated):', q);
```

Output

Run

- EXAMPLE

Using `new Object()`

```
var o = new Object();

o.a = 4;
o.b = 'Rome';
o.c = true;

console.log('Object \'o\':', o);
```

Output

Run

- EXAMPLE

Using `Object.create()`

```
var x = {
    a: 5,
    foo: function() {
        return this.a * this.a;
    }
};

var o = Object.create(x);

console.log('\'x\':', x);
console.log('Object \'o\':', o);
console.log('Property \'o.a\':', o.a);
console.log('Method \'o.foo()\':', o.foo());

o.a = 7;

console.log('Property \'o.a\':', o.a);
console.log('Method \'o.foo()\':', o.foo());
```

Output

Run

2. Using a Constructor Function

We can use a *constructor function* to create an object in two steps:

1. Define the object type by writing a constructor function. The strong convention here is for the function's name to be in UpperCamelCase.
2. Use the `new` operator to create an instance of the object.

- EXAMPLE

```
function Actor(firstName, lastName, Age) {
    this.firstName = firstName;
    this.lastName = lastName;
    this.Age = Age;
}

var a1 = new Actor('Julia', 'Roberts', 48);
var a2 = new Actor('Kate', 'Winslet', 40);

console.log('Object \'a1\':', a1);
console.log('Object \'a2\':', a2);
```

Output

Run