SOEN 287

Chapter 4: JavaScript (4)

Dr. Yuhong Yan
CSE, Concordia University
Winter, 2023

`Const` – variable declaration

- The `const` keyword was introduced in ES6 (2015)
- Variables defined with `const` cannot be redeclared.
- Variable defined with `const` cannot be reassigned.
- Variable defined with `const` have block scope.

`const` variables must be assigned a value at declaration

- Correct

```
const PI = 3.14159265359;
```

- Incorrect

```
const PI;
PI = 3.14159265359;
```

- Always declare a variable with const when you know the value should not be changed

Use `const` when you declare

- A new array
- A new Object
- A new Function
- A new RegExp

Object Creation and Modification

- Creation

```
var myObject = new Object();
```

- The new object has no properties - a blank object
- Properties can be added to an object, any time

```
var myCar = new Object();
myCar.make = "Ford";
myCar.model = "Focus";
```

- Properties can be accessed by dot notation or in array notation, as in

```
var property1 = myCar["model"];

delete myCar.model;
```

Object Creation and Modification (2)

- An Abbreviated way

```
var myCar = {make:"ford", model: "Contour SVT"};
```

- Also called the JSON way
- Or use `const`

```
const myCar = {make:"ford", model: "Contour SVT"};
//add a property
myCar.color = “red”;
//change a property
myCar.color = “blue”;
```

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_const_object

Object Creation and Modification (3)

- Or use `const`

```
// You can create a const object:
const car = {type:"Fiat", model:"500", color:"white"};
// You can change a property:
car.color = "red";
// You can add a property:
car.owner = "Johnson";
```

- But you cannot reassign an object

```
const car = {type:"Fiat", model:"500", color:"white"};

car = {type:"Volvo", model:"EX60", color:"red"};    //
ERROR
```

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_const_object

Visit the properties in an object

- Is the property in an object?

```
var myCar = {make:"ford", model: "Contour SVT"};
'make' in myCar;
```

- Traverse all the properties

```
for (var prop in myCar)
  document.write(myCar[prop] + "<br />");
```

→**SHOW** `testObject1.html` **and display**

instanceof

- Is object instanceof constructor?

```
var myCar = {make:"ford", model: "Contour SVT"};
myCar instanceof myCar;      ✗
myCar instanceof Object;     true
myCar instanceof Date;       false
```

Array is an object

```
var myList = new Array(24, "bread", true);
var myList2 = [24, "bread", true];
var myList3 = new Array(24); //!

myList[122] = "bits";  // length is 123

myList.length = 150;
```

Array

```
var myList = new Array(24, "bread", true);
var myList2 = [24, "bread", true];
var myList3 = new Array(24); //!

 myList[122] = "bits";  // length is 123

myList.length = 150;
```

Use const to declare an array

```
// You can create a constant array:
const cars = ["Saab", "Volvo", "BMW"];

// You can change an element:
cars[0] = "Toyota";

// You can add an element:
cars.push("Audi");
```

- You cannot reassign the array

```
const cars = ["Saab", "Volvo", "BMW"];

cars = ["Toyota", "Volvo", "Audi"];    // ERROR
```

Array

- `join` – e.g., `var listStr = list.join(", ");`
- `reverse` – change the original array
- `sort` – e.g., `names.sort();` change the original array
  - Coerces elements to strings and puts them in alphabetical order
- `concat` – e.g., `newList = list.concat(47, 26);`
- `slice`

```
listPart = list.slice(2, 5);
listPart2 = list.slice(2);
listPart2 = list.slice(-2);
```

- `toString`
  - Coerce elements to strings, if necessary, and cantenate them together, separated by commas (exactly like `join(", ")`)
- `push`, `pop`, `unshift`, and `shift`

function

```
function function_name([formal_parameters]) {
  -- body -
 }
```

- Return value is the parameter of `return`
  - If there is no `return`, or if the end of the function is reached, `undefined` is returned
  - If `return` has no parameter, `undefined` is returned
- `functions` are objects

```
ref_fun = fun;
   ...
ref_fun();  /* A call to fun */
```

- Functions are defined in the head of the HTML file

function

- No type checking, no number of parameters checking
- What happens to the parameters?

→ **SHOW `params.js` and output**

Params.js

```
 function params(a, b) {
        document.write("Function params was passed ",
            arguments.length, " parameter(s) <br />");
        document.write("Parameter values are: <br />");

        for (var arg = 0; arg < arguments.length; arg++)
          document.write(arguments[arg], "<br />");
        document.write("a="+a+" "+"b="+b+"<br />");
        document.write("<br />");
        }
// A test driver for function params
      params("Mozart");
      params("Mozart", "Beethoven");
      params("Mozart", "Beethoven", "Tchaikowsky");
```

function as a parameter

- Revisit array's sort()
- Work for the numbers? ✗
- Pass a function as a parameter to sort()

```
function sortNumber(a, b)
{
        return a - b;
}

var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));
```

→ **SHOW** `sort.html`

Sort.html

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort()+"<br/>");

var numbers = [1, 5, 100, 40];
document.write(numbers.sort()+"<br/>");

function sortNumber(a, b){
   return a-b;}

var  n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));
```

```
Apple,Banana,Mango,Orange
1,100,40,5
1,5,10,25,40,100
```

array.sort(sortfunction)

```
function sortfunction(a, b)
{
     //Compare "a" and "b" in some fashion, and return -1, 0,
or 1

     return (a - b);  //causes an array to be sorted
numerically and ascending

}
```

- **Less than 0**: Sort "a" to be a lower index than "b"
- **Zero**: "a" and "b" should be considered equal, and no sorting performed.
- **Greater than 0**: Sort "b" to be a lower index than "a".

Anonymous Function

```
var f = function(x,y) {return x+y;}
f(1,2);
```

Object Creation and Modification

- Creation

```
var myObject = new Object();
```

- The new object has no properties - a blank object
- Properties can be added to an object, any time

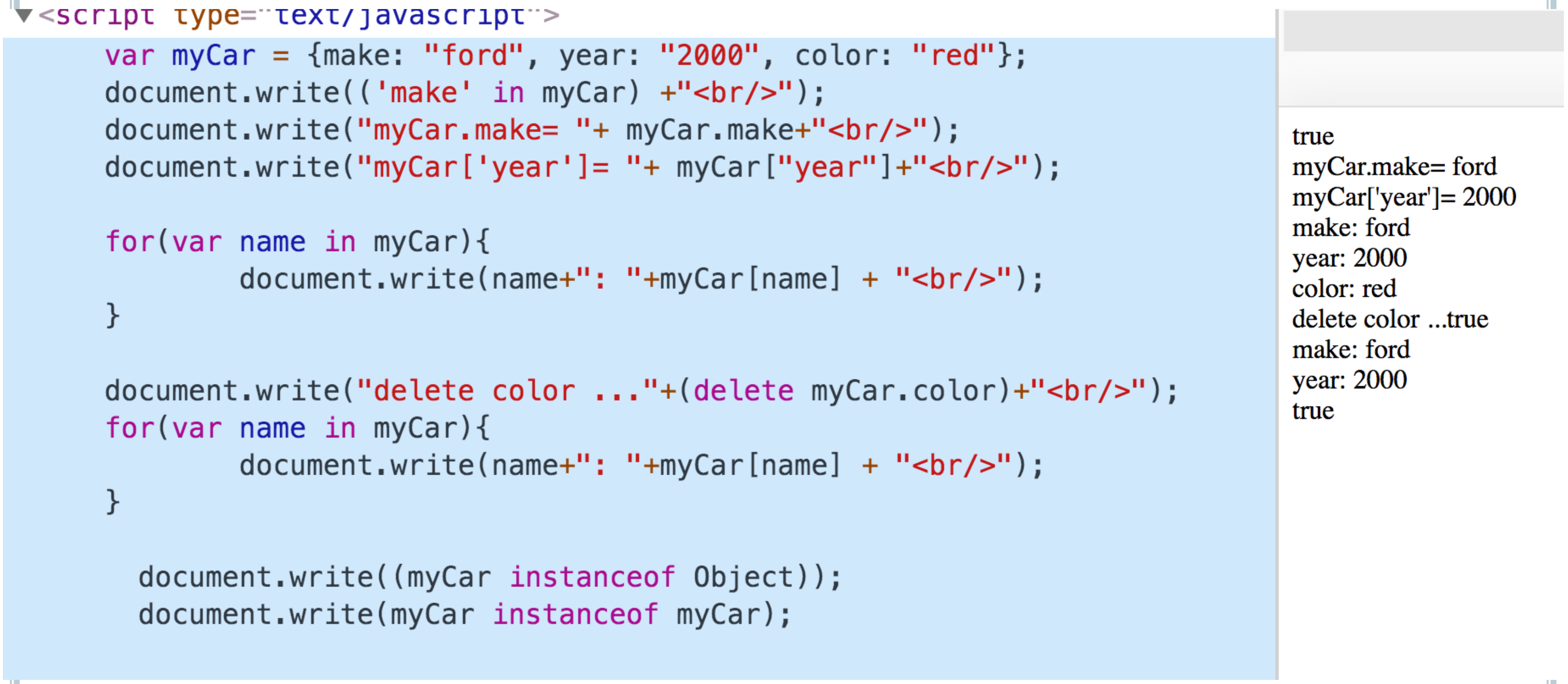
```
var myCar = new Object();
myCar.make = "Ford";
myCar.model = "Focus";
```

- Properties can be accessed by dot notation or in array notation, as in

```
var property1 = myCar["model"];

delete myCar.model;
```

testObject1.html



```
<script type="text/javascript">
    var myCar = {make: "ford", year: "2000", color: "red"};
    document.write(('make' in myCar) +"<br/>");
    document.write("myCar.make= "+ myCar.make+"<br/>");
    document.write("myCar['year']= "+ myCar["year"]+"<br/>");

    for(var name in myCar){
            document.write(name+": "+myCar[name] + "<br/>");
    }

    document.write("delete color ..."+(delete myCar.color)+"<br/>");
    for(var name in myCar){
            document.write(name+": "+myCar[name] + "<br/>");
    }

      document.write((myCar instanceof Object));
      document.write(myCar instanceof myCar);
```

```
true
myCar.make= ford
myCar['year']= 2000
make: ford
year: 2000
color: red
delete color ...true
make: ford
year: 2000
true
```

Constructors

- Initialize object

```
function plane(newMake, newModel, newYear){
   this.make = newMake;
   this.model = newModel;
   this.year = newYear;
 }

 myPlane = new plane("Cessna",
                     "Centurnian",
                     "1970");
```

- How to display it?

→**SHOW** `testObject2.html` **and display**

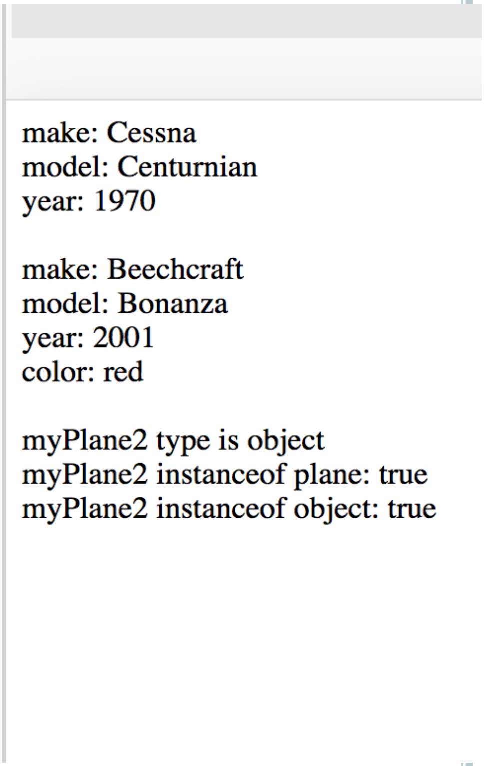
testObject2.html

```
 function plane(newMake, newModel, newYear){
     this.make = newMake;
     this.model = newModel;
     this.year = newYear;
 }

var myPlane = new plane("Cessna", "Centurnian", "1970");
var myPlane2 = new plane("Beechcraft", "Bonanza", "2001");

myPlane2.color = "red";

for(var name in myPlane){
    document.write(name +": "+myPlane[name] + "<br/>");
}
document.write("<br/>");
for(var name in myPlane2){
    document.write(name +": "+myPlane2[name] + "<br/>");
}
```

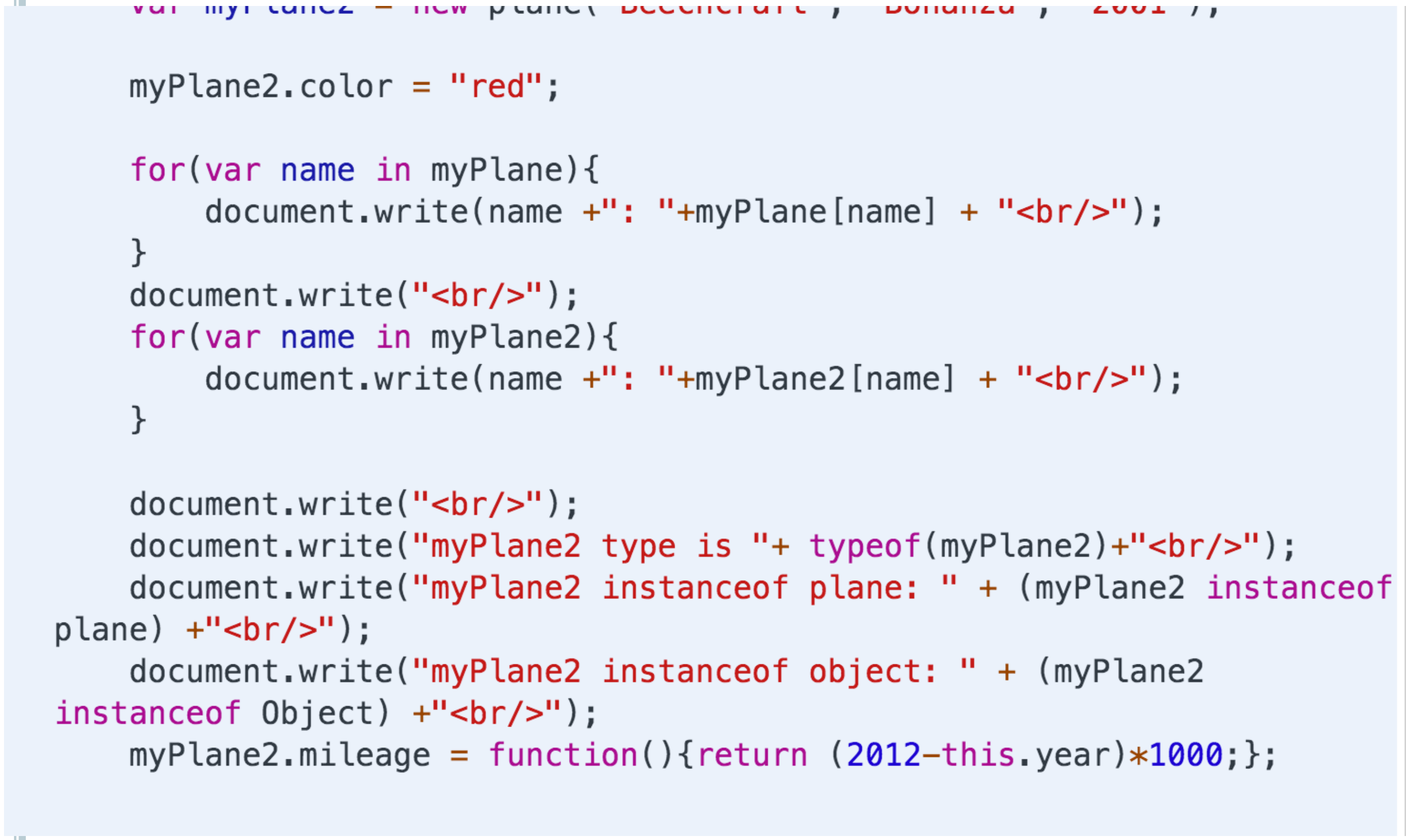


```
make: Cessna
model: Centurnian
year: 1970

make: Beechcraft
model: Bonanza
year: 2001
color: red

myPlane2 type is object
myPlane2 instanceof plane: true
myPlane2 instanceof object: true
```

testObject2.html (2)

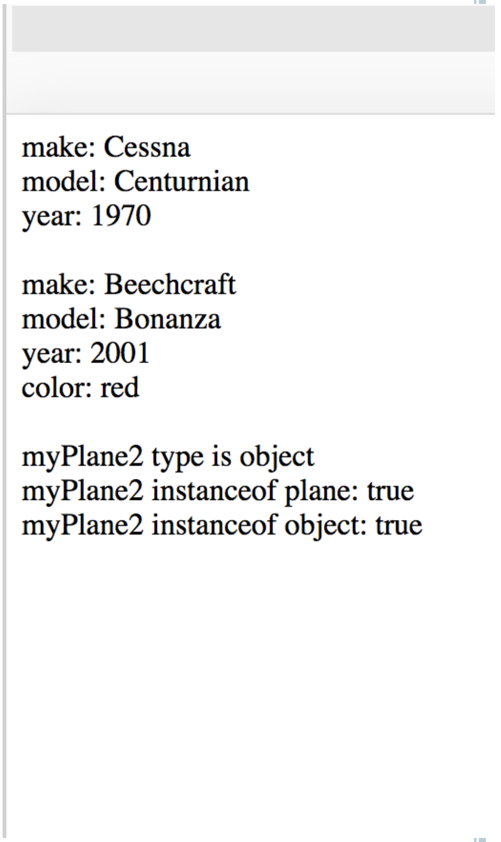


```
    var myPlane2 = new plane("Beechcraft", "Bonanza", "2001");

    myPlane2.color = "red";

    for(var name in myPlane){
        document.write(name +": "+myPlane[name] + "<br/>");
    }
    document.write("<br/>");
    for(var name in myPlane2){
        document.write(name +": "+myPlane2[name] + "<br/>");
    }

    document.write("<br/>");
    document.write("myPlane2 type is "+ typeof(myPlane2)+"<br/>");
    document.write("myPlane2 instanceof plane: " + (myPlane2 instanceof
plane) +"<br/>");
    document.write("myPlane2 instanceof object: " + (myPlane2
instanceof Object) +"<br/>");
    myPlane2.mileage = function(){return (2012-this.year)*1000;};
```



```
make: Cessna
model: Centurnian
year: 1970

make: Beechcraft
model: Bonanza
year: 2001
color: red

myPlane2 type is object
myPlane2 instanceof plane: true
myPlane2 instanceof object: true
```

Can also use anonymous function in constructor

```
var f = function(x,y) {return x+y;}
f(1,2);
```

A function to display the properties

```
function displayPlane() {
   document.write("Make: ", this.make,
                  "<br />");
   document.write("Model: ", this.model,
                  "<br />");
   document.write("Year: ", this.year,
                  "<br />");
 }
```

- How to call it?

```
this.display = displayPlane;
…
var myPlane = new plane("Cessna","Centurnian",
"1970");
myPlane.display();
```

→ **SHOW** `plane.js`

Object.prototype

- Use it to change the template of the object
- It affects all the objects of the same type
- You can put the functions into the prototype
- The linkage model
- Use prototype to build longer chain of inheritance

→**SHOW** `objectInheritance2.html` **and display**

→**SHOW** `objectInheritance_Person1.html` **and display**

Built-in JavaScript Constructors

- new String()
- new Number()
- new Boolean()
- new Object()
- new Array()
- new RegExp()
- new Function()
- new Date()

JavaScript Class

```
class Car {
  constructor(name, year) {
    this.name = name;
    this.year = year;
  }
}

const myCar = new Car("Ford", 2014);
```

→**SHOW** `JSClass.html` **and display**

→**SHOW** `JSClass.htm2` **and display**

→**SHOW** `JSClass.htm3` **and display**

The End