CS 106: Web Technology - I Assignment #6

Anup Adhikari
anup.adhikari@gandakiuniversity.edu.np

Gandaki University January 31, 2023

1 JS Objects and Classes

Objectives

The objective of the lab is:

- To understand the handling of objects in JS.
- To understand the class concepts and objects creation.

Info: What is JavaScript?

JavaScript was initially created to "make web pages alive".

The programs in this language are called scripts. They can be written right in a web page's HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don't need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called Java.

Today, JavaScript can execute not only in the browser, but also on the server, or actually on any device that has a special program called the JavaScript engine.

Table 1: The CANs and CANTs of JavaScript

Add new HTML to the page, change the existing content, modify styles.	JavaScript on a webpage may not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS functions.
React to user actions, run on mouse clicks, pointer movements, key presses.	Different tabs/windows generally do not know about each other. Sometimes they do, for example when one window uses JavaScript to open the other one. But even in this case, JavaScript from one page may not access the other if they come from different sites (from a different domain, protocol or port).
Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).	JavaScript can easily communicate over the net to the server where the current page came from. But its ability to receive data from other sites/domains is crippled. Though possible, it requires explicit agreement (expressed in HTTP headers) from the remote side. Once again, that's a safety limitation.
Get and set cookies, ask questions to the visitor, show messages.	
Remember the data on the client-side ("local storage").	

Info: How do engines work?

Engines are complicated. But the basics are easy.

The engine (embedded if it's a browser) reads ("parses") the script. Then it converts ("compiles") the script to the machine language. And then the machine code runs, pretty fast.

The engine applies optimizations at each step of the process. It even watches the compiled script as it runs, analyzes the data that flows through it, and further optimizes the machine code based on that knowledge.

Objects

There are major 8 datatypes in JS:

1. number for numbers of any kind: integer or floating-point, integers are limited by ±(253-1).
2. bigint is for integer numbers of arbitrary length.
3. string for strings. A string may have zero or more characters, there's no separate single-character type.
4. boolean for true/false.
5. null for unknown values - a standalone type that has a single value null.
6. undefined for unassigned values - a standalone type that has a single value undefined.
7. object for more complex data structures.
8. symbol for unique identifiers.

In contrast to other datatypes, objects are used to store keyed collections of various data and more complex entities. In JavaScript, objects penetrate almost every aspect of the language.

An empty object ("empty cabinet") can be created using one of two syntaxes:

object.js

```
let user = new Object();//"object constructor"
let user = {};  // "object literal" syntax
```

Setting the properties is quite easy also.

object.js

```
let user = {     // an object
  name: "Birendra",  // by key "name" store value "John"
  age: 30        // by key "age" store value 30
};
```

Accessing the member properties:

object.js

```
// get property values of the object:
alert( user.name ); // Birendra
alert( user.age ); // 30
```

Setting the member properties:

object.js

```
user['name'] = "Shiva";
user.age = 34;
user.isAdmin = true;
```

Info: The new property **isAdmin** has been added to the object "user" with value "true"

Removing the member properties:

object.js

```
delete user.isAdmin;
```

Computed properties:

object.js

```
let fruit = prompt("Which fruit to buy?");
// let user enter the value apple
let bag = {
  [fruit]: 5, // the name of the property is taken
              from the variable fruit
};

alert( bag.apple ); // 5 if fruit="apple"
```

Object member names: As we already know, a variable cannot have a name equal to one of the language-reserved words like “for”, “let”, “return” etc.

But for an object property, there’s no such restriction:

object.js

```
// these properties are all right
let obj = {
  for: 1,
  let: 2,
  return: 3
};

alert( obj.for + obj.let + obj.return );  // 6
```

Question 1

Write a function in JS which allows the following tasks

1. Create an empty object user.
2. Add the property name with the value Shyam.
3. Add the property surname with the value Rana.
4. Change the value of the name to Krishna.
5. Remove the property name from the object.

Question 2

Write a method that finds shallow intersections of objects.
Expected Result: ({ a: 1, b: 2 }, { c: 1, b: 2 }) => { b: 2 }
@param Object<string | number> firstObj - Object with values of primitive data types
@param Object<string | number> secondObj - Object with values of primitive data types
@returns Object

Classes

In object-oriented programming, a class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

syntax

```
class MyClass {
  // class methods
  constructor() { ... }
  method1() { ... }
  method2() { ... }
  method3() { ... }
  ...
}
```

Question 3

You are asked to convert the following into a Clock Class.

object.js

```
<html lang="en">
<head>
    <title>Digital Clock</title>
</head>
<body>
    <div id="clock">8:10:45</div>
    <script>
        setInterval(showTime, 1000);
        function showTime() {
            let time = new Date();
            let hour = time.getHours();
            let min = time.getMinutes();
            let sec = time.getSeconds();
            am_pm = "AM";

            if (hour > 12) {
                hour -= 12;
                am_pm = "PM";
            }
            if (hour == 0) {
                hr = 12;
                am_pm = "AM";
            }

            hour = hour < 10 ? "0" + hour : hour;
            min = min < 10 ? "0" + min : min;
            sec = sec < 10 ? "0" + sec : sec;

            let currentTime = hour + ":"
                + min + ":" + sec + am_pm;

            document.getElementById("clock")
                .innerHTML = currentTime;
        }

        showTime();
    </script>
</body>
</html>
```