WEB222 - Web Programming Principles

Week 4: Object-Orient JavaScript

Agenda

- Standard Built-in objects
  - Date, Math
- User-defined objects
- Prototypal inheritance

Date Object

- Enables basic storage and retrieval of dates and times.
- Create a Date object with current date and time:

```
var today = new Date();
console.log("The date is " + today);
```

- Will show the date string:

The date is Mon Sep 12 2017 15:22:15 GMT-0400 (Eastern Standard Time)

Date Object

- Create a Date object with a specific date and time:

```
var date1 = new Date(1996, 6, 6);
var date2 = new Date(2001, 4, 10, 11, 13, 15, 0);
var date3 = new Date("Sept 12, 2017");
console.log(date1 + "\n" + date2 + "\n" + date3);
```

- Output:

```
Sat Jul 06 1996 00:00:00 GMT-0400 (Eastern Standard Time)
Thu May 10 2001 11:13:15 GMT-0400 (Eastern Standard Time)
Tue Sep 12 2017 00:00:00 GMT-0400 (Eastern Standard Time)
```

Date Object: the get… Methods

- getMonth() method
  - Returns a Number - 0 to 11
  - Represents the months of January through December
  - e.g.

```
var myMonth = (new Date()).getMonth();
console.log(myMonth); // The myMonth is 8, if in September
```

- getDate() method
  - Returns a number - 1 to 31
  - e.g.

```
var myDay = (new Date()).getDate();
console.log(myDay );
```

Date Object: the get… Methods

- getDay()
  - Returns a Number: 0 for Sunday, 1 for Monday, …
  - e.g.

```
var myDayOfWeek = (new Date()).getDay();
console.log(myDayOfWeek); // 5 if on a Friday
```

- getFullYear()
  - Returns a 4 digit year
  - e.g.

```
var myYear = (new Date()). getFullYear();
console.log(myYear);  // e.g. 2017
```

Date Object: the get… Methods

- getHours() method
  - returns a number of 0 to 23
- getMinutes() method
  - returns a number of 0 to 59
- getSeconds() method
  - returns a number of 0 to 59
- e.g.

```
var myDate = new Date();
var myHour = myDate.getHours();
var myMinutes = myDate.getMinutes();
var mySeconds = myDate.getSeconds();
alert(myHour + ":" +  myMinutes  + ":" + mySeconds); // e.g. 10:9:5
```

Date Object: getting Date strings

- Methods
  - toString()
  - toLocaleString()
  - toUTCString() // UTC: Universal Time Coordinated
  - toDateString()
- For Example:

```
var dt = new Date();
console.log(dt);                    // Thu Jun 02 2016 11:11:50
GMT-0400 (Eastern Daylight Time)
console.log(dt.toString());         // Thu Jun 02 2016 11:11:50
GMT-0400 (Eastern Daylight Time)
console.log(dt.toLocaleString()); // 6/2/2016, 11:11:50 AM
console.log(dt.toUTCString());// Thu, 02 Jun 2016 15:11:50 GMT
console.log(dt.toDateString());    // Thu Jun 02 2016
```

Math Methods

Method	desciption	example	value
max(x, y)	maximum of n numbers	Math.max(0.52, 1)	1
min(x, y)	minimum of n numbers	Math.min(0.52, 1)	2
pow(x, y)	X to the power y	Math. pow(2, 8)	
sqrt(x)	Square root of x	Math. sqrt(9)	3
ceil(x)	integer closest to and not less than	Math.ceil(0.52)	1
floor(x)	integer closest to and not greater than	Math.floor(0.52)	0
round(x)	Integer closet to	Math.round(0.52)	1
random()	return a floating point number between 0 (inclusive) and 1 (exclusive)	Math.random()	0.03517110995016992

Math Object – Math functions (rounding)

- Math.ceil(ident_1)
  - integer closest to but not less than
  - e.g.

```
console.log( Math.ceil(0.52) ); // 1
console.log( Math.ceil(0.49) ); // 1
```

- Math.floor(ident_1)
  - integer closest to but not greater than
  - e.g.

```
console.log( Math.floor(0.52) ); // 0
```

- Math.round(ident_1)
  - integer closest to
  - e.g.

```
console.log( Math.round(0.52) ); // 1
console.log( Math.round(0.49) ); // 0
console.log( Math.round(0.5) );  // 1
```

Generating Random Numbers

- Math.random()
  - Generates a pseudorandom number between 0 (inclusive) and 1 (exclusive)
  - e.g:

```
// Returns a random number between 0 (inclusive) and 1 (exclusive)
Math.random();

// Returns a random number between min (inclusive) and max (exclusive)
Math.random() * (max - min) + min;

// Returns a random integer between min (included) and max (excluded)
Math.floor(Math.random() * (max - min)) + min;

// Returns a random integer between min (included) and max (included)
Math.floor(Math.random() * (max - min + 1)) + min;
```

Creating User-defined Objects

- JavaScript objects are associative arrays (or map, or dictionary - an data structure composed of a collection of key/value pairs), augmented with prototypes.
  - Object property names are string keys. They support two equivalent syntaxes:
    - dot notation (obj**.x** = 10)
    - bracket notation (obj**['x']** = 10)
  - Object properties and functions/methods can be added, changed, or deleted at run-time.

Create Objects - using literal notation

```
var person1 = { "name": "John", "age": 30 };
// can be simplified as:
//var person1 = { name: "John", age: 30 };

var person2 = {
    name: "Steven",
    age: 25,
    talk: function () {
        console.log('I am ' + this.name + ", and I'm " +
this.age + " years old.");
    }
};

console.log( person1.name );
person2.talk(); // My name is Steven, and I'm 25 years old.
```

The *this* Keyword

- In JavaScript, the thing called this, is the object that "owns" the JavaScript code.
  - The value of this, when used in a function, is the object that "owns" the function. e.g. "this" in week 3 example:

```
String.prototype.reverse = function () {…}
```

  - The value of this, when used in an object, is the object itself.

Create Objects - dynamically add/delete members

- Create an empty object; then dynamically add or delete properties and methods.

```
var person4 = {};
// equivalent to: var person4 = new Object();

person4.name = "James";
person4.age = 30;
person4.show = function () {
    console.log('My name is ' + this.name + ", and I'm "
        + this.age + " years old.");
};

person4.show();

delete person4.age;
person4.show();
```

Create Objects - using function constructor

- Using a **Function Constructor** to declare object type:

```
function Person(name, age) {
    this.name = name;
    this.age = age;
    this.show = function () {
        console.log('My name is ' + this.name + ", and I'm " + this.age +
" years old.");
    };
}

Person.prototype.say = function () {
        console.log('My name is ' + this.name + ", and I'm " + this.age +
" years old.");
}
```

Create Objects - using function constructor

- Then use the new operator creates an instance of an object type specified by the function:

```
var person3 = new Person("Steven", 30);
console.log( person3.age);
person3.show(); // My name is Steven, and I'm 30 years old.
person3.say(); // My name is Steven, and I'm 30 years old.
```

Using for-each loop for JS Objects

- The for-each (for-in) loop iterates over the enumerable properties of an object, in arbitrary order. For each distinct property, statements can be executed.

```
var student = {name:"John", program:"CPD", semester:2};
var str = "Student info:\n\t";

for (var x in student) { // x stands for …?
    str += x + ": " + student[x] + "\n\t";
}
console.log(str);
```

Advanced: JS Object with Closure

- Usually, JavaScript object properties are "public". This does not conform the basic principle of OOP – Encapsulation.
- JavaScript object with data hiding example:

```
function Person(name, age) {
  var name = name;
  var age = age;

  return { setName:   function(newName) {name = newName;},
           getName:   function() { return name; },
           setAge:    function(newAge) { age = newAge;},
           getAge:    function() { return age; }
         };
}

var person1 = new Person("John", 25);
console.log(person1.getName()); // John

person1.setAge(20);
console.log(person1.getAge());
```

Prototypal Inheritance

- JavaScript supports OOP in a special model: prototype-based programming.
- Prototypal Inheritance: Objects inherit from objects.
- In JavaScript, objects are not based on classes.
- JavaScript does not use the classical inheritance paradigm that is found in C++, Java, and C#.
- A new object can inherit the properties and methods of an existing object.
- Existing object: as prototype for creating the new object.
- "New object is a clone of the existing object."
- Note: do not to be confused with Prototype framework that is a JS library of prototype.js.

Prototypal Chain

- In JavaScript, all objects have an internal "__proto__" (sometimes "[[prototype]]") property
- This property refers to an object, from which the current object "inherits" properties.
- If a new object was created using Literal Notation, we have access to a global function: Object.create(); which will allow us to create a second new object that uses the first as a prototype (Example 1)
- If a new object was created using a Function Constructor, we have access to a public "prototype" property which we can use to explicitly set a prototype (Example 2)
- For a more detailed explanation, see: https://developer.mozilla.org/en/docs/Web/JavaScript/Inheritance_and_the_prototype_chain

Example 1 - Object.create()

```
var rectangle1 = {
    width: 10,
    height: 15,
    show: function () {
        console.log('dimensions: ' + this.width + " x " + this.height);
    }
};

// creates a new rectangle using rectangle1 as prototype
var rectangle2 = Object.create(rectangle1); // clone
rectangle2.show();    // dimensions: 10 x 15

rectangle2.width = 20;
rectangle2.height = 25;

rectangle2.show();  // dimensions: 20 x 25
```

Example 2 - prototype property

- Allows you to add properties and methods to an existing object
- E.g.

```
String.prototype.reverse = function () {
    var rev = '';
    for (var i = this.length - 1; i >= 0; i--)
        rev += this[i]; // the string
    return rev;
};

var myString = "WEB222";
console.log( myString.reverse() );  // 222BEW
```

JS OOP Example – Subject

- Model of subjects for School of ICT

```
var subject = {
    code: "",
    desc: "",
    prog: [], //  the prog property is an array
    info: {}  //  the info property is an object
};
```

- Create subject instances using the Object.create method.

```
var web222   = Object.create(subject);
web222.code = 'WEB222';
web222.desc = 'Internet I - Internet Fundamentals ';
web222.prog = ['CPD', 'CPA'];
web222.info  = { hours: 4, url:'http://scs.senecac.on.ca/course/web222'
};
```

JS OOP Example – Subject

```
var bti220 = Object.create(subject);
bti220.code = 'BTI220';
bti220.desc = 'Internet Architecture and Development';
bti220.prog = ['BSD'];
bti220.info = { hours: 4, url:'http://scs.senecac.on.ca/course/bti220' }

var ipc144 = Object.create(subject);
ipc144.code = 'IPC144';
ipc144.desc = 'Introduction to Programming Using C';
ipc144.prog = ['CPD', 'CPA', 'CTY'];
ipc144.info = { hours: 5, url:'http://scs.senecac.on.ca/course/ipc144' }

var btc140 = Object.create(subject);
btc140.code = 'BTC140';
btc140.desc = 'Critical Thinking and Writing';
btc140.prog = ['BSD', 'IFS'];
btc140.info = { hours: 3, url:'http://scs.senecac.on.ca/course/btc140' }
```

JS OOP Example – Subject

- All subjects

```
// Create a collection of all subject objects
var all = [ web222, bti220, ipc144 ];
all.push(btc140);

// Declare and initialize an accumulator
var totalHours = 0;

// Go through the collection, accumulate hours, dump to the Web Console
for (var i = 0; i < all.length; i++) {
    totalHours += all[i].info.hours;
    console.log(all[i]);
};

// Report the total hours
console.log('Total hours is ' + totalHours);
```

JS OOP Example - Person

- Create a person object, with some properties that are common to all persons – name, birthday, etc.

```
var person = {
    name: "",
    bday: new Date(),
    mail: "",
    prnt: function () {
        return 'Info for ' + this.name + ', born on ' +
                this.bday.toLocaleDateString() + ', email ' + this.mail;
    }
};
```

- oop-seneca-subjects.js

JS OOP Example - Person

- Create new objects: students and teachers using person as the prototype.

```
// create student object
var student = Object.create(person,
                                { prog: { value: '' }, stid: { value: '' }});
var stu1 = Object.create(student);
stu1.name = 'Stanley';
stu1.bday = new Date(1983, 10, 15);
stu1.mail = 'stan@myseneca.ca';
stu1.prog = 'BSD';
stu1.stid = '012345678';
console.log(stu1.name);
var x =stu1.prnt();
console.log(x);
```

JS OOP Example - Person

```
// create teacher object using person as the prototype.

var teacher = Object.create(person, { offc: { value: "T2095" },
                web: { value: " www.senecacollege.ca"}});

var tch1 = Object.create(teacher);

tch1.name = "Peter";
tch1.bday = new Date(1900,1,1);
tch1.mail = "peter@senecacollege.ca";

//tch1.offc = "T2099";
//tch1.web = " www.senecacollege.ca";

console.log(tch1.name+ ", " + tch1.offc);
var x =tch1.prnt();
console.log(x);
```

Resourceful Links

- Introduction to Object-Oriented JavaScript - MDN
- Inheritance and the prototype chain - JavaScript | MDN
- Details of the object model - JavaScript | MDN
- Closures - JavaScript | MDN
- Standard built-in objects - JavaScript | MDN

Thank you!

Any Questions?