BTI220 - Internet Architecture and Development

Week 3: Object-Orient JavaScript

Agenda

- JavaScript objects
- Standard Built-in objects
  - String
  - Array
  - RegExp
- User-defined objects
- Prototypal inheritance

JavaScript Objects

- In JavaScript an "object" is a self-contained entity comprising of "properties" (variable), and "methods" (functions).
  - an object can store data in its properties - state
  - and perform actions with its methods - behavior.

- In JavaScript, you can create a new object without creating a class.
  - The object does not belong to any class; it is the only one of its kind, a singleton

JavaScript Object Categories

- There are three kinds of JavaScript object:
  - Built-in objects
    - an intrinsic part of JavaScript, providing useful features, such as String, Date, Math and JSON
  - Host Objects
    - objects that are supplied to JavaScript by the browser environment. e.g. DOM/BOM objects: window, document and form, …
  - Custom objects
    - user-defined object to store data and provide functionality in a single object

JavaScript Built-in Objects

- JavaScript provides many predefined, built-in objects that enable you to work with Strings and Dates, perform mathematical operations, and etc.:
  - String
  - Array
  - Date
  - Math
  - Number
  - Boolean
  - RegExp
  - JSON
- We'll cover String, Array, RegExp in this week.

JavaScript String Object

- Strings enclosed within double or single quotes are used for holding data that can be represented in text format.
- Some of the most-used operations on strings are to
  - check their length, and to
  - concatenate strings using the + and += string operators.

String object - properties and methods

Member	Type	Example
length	property	stringName.length
charAt(n)	method	stringName.charAt(n)
charCodeAt(n)	method	stringName.charCodeAt(n)
concat(string2, string3)	method	stringName.concat(string2, string3)
indexOf('x')	method	stringName.indexOf('x')
lastIndexOf('x')	method	stringName.lastIndexOf('x')
split('x')	method	var arrayName = stringName.split('x')
substr(x,y)	method	stringName.substr(x,y) – x=from, y=length
substring(x,y)	method	stringName.substring(x,y) – x=from (inclusive) y=to (not inclusive)
toLowerCase()	method	Converts a string to lowercase.
toUpperCase()	method	Converts a string to uppercase.
trim()	method	Removes whitespaces from the left and right of a string.
prototype	property	Allows you to add properties and methods to an object

JS String object - property

- Length
  - The length property returns the number of characters in a string.
  - Syntax: stringName.length

```
Position/index      »          012345
var  myString       =         "INT222";
```

```
myString.length  // returns 6
```

JS String object - methods

- charAt(index)
  - The method returns the character at the specific index.
  - Characters in a string are indexed from left to right
    - Index start from 0 to one less than the length.
  - The index of the last character in a string called myString is myString.length - 1
  - If the index you supply is out of range, JavaScript returns an empty string
  - Syntax: stringName.charAt(index)

```
Position/index      »          012345
var  myString       =         "INT123";
```

```
myString.charAt(0) // returns  I
myString.charAt(1) // returns  N
myString.charAt(2) // returns  T
myString.charAt(3) // returns  1
myString.charAt(4) // returns  2
myString.charAt(5) // returns  3
myString.charAt(6) // returns
```

JS String object - methods

- charCodeAt(index)
  - The method returns the unicode of a character.
  - Index can be a value from 0 to one less than the length.
  - Syntax: stringName.charCodeAt(index)

```
Position/index      »          012345
var  myString       =         "AZaz19";

   myString.charCodeAt(0) //  A  returns  65
   myString.charCodeAt(1) //  Z  returns  90
   myString.charCodeAt(2) //  a  returns  97
   myString.charCodeAt(3) //  z  returns  122
   myString.charCodeAt(4) //  1  returns  49
   myString.charCodeAt(5) //  9  returns  57
   myString.charCodeAt(6) //     returns  NaN
```

Notes

- About Unicode
  - Unicode provides a unique number for every character, no matter what the platform is.
  - UTF-8: a character encoding has become the dominant for the World Wide Web.
  - See unicodes table.
- Using array index to access character in a string

```
var example2 = "INT222";
alert(example2[2]); // returns T
```

JS String object - methods

- concat(string2,string3,...)
  - The concat(....) method combines the text of two or more strings.
  - It is always recommended that you use the assignment operators **(+, +=)** instead of the concat method.
  - Syntax: stringName.concat(string2, string3)

```
var myString0 = "My courses are : ";
var myString1 = "INT222";
var myString2 = "OOP222";

myString0 = myString0.concat(myString1, " & " ,
myString2);
// returns  My courses are : INT222 & OOP244
```

JS String object - methods

- indexOf(subStr)
  - returns the position at which the character or string begins. indexOf returns only the first occurrence of your character or string.
  - If indexOf returns zero, the character or the string you are looking for begins at the 1st character.
  - If indexOf returns **-1**, the character or string you searched for is not contained within the string.

```
Position/index      »          012345
var  myString       =         "INT222";

    myString.indexOf('I')    // returns  0
    myString.indexOf('N')    // returns  1
    myString.indexOf('2')    // returns  3
    myString.indexOf('T2') // returns  2
    myString.indexOf('3')    //  returns  -1
```

indexOf("subStr",[optional]) method

- indexOf(subStr, *from*)
  - *from* : starting the search from position from.

```
Position/index      »          012345
var  myString       =         "INT222";
```

```
myString.indexOf('I') // returns  0
myString.indexOf('2') // returns  3
myString.indexOf('2',5) // returns  5
myString.indexOf('3') // returns  -1
```

JS String object - methods

- lastIndexOf(subStr)
  - returns the position at which the last occurrence of your character or string – searching backwards.
  - If lastIndexOf returns -1, the character or string you searched for is not contained within the string.

```
Position/index      »          012345
var  myString       =         "INT222";

    myString.lastIndexOf('I')    // returns  0
    myString.lastIndexOf('N')    // returns  1
    myString.lastIndexOf('2')    // returns  5
    myString.lastIndexOf('22') // returns  4
    myString.lastIndexOf('3')    // returns  -1
```

lastIndexOf("x",[optional]) method

- lastIndexOf(subStr, *from*)
  - *from* : starting the search from position from – searching backwards

```
Position/index    »        012345
var  myString     =       "INT222";
```

```
myString.lastIndexOf('I')     // returns  0
myString.lastIndexOf('2')     // returns  5
myString.lastIndexOf('2',4) // returns  4
myString.lastIndexOf('3')     // returns  -1
```

JS String object - methods

- split(x)
  - The split(' ') uses the specified character(s) to break the argument string into an array.
  - Syntax: stringName.split(x)

```
Position/index      »          0123456
var  myString       =         "INT 222";
var myArray         =         myString.split(' ');
```

A split on a blank will return the following:

myArray[0] // element 0 returns INT
myArray[1] // element 1 returns 222

split(x)

```
Position/index      »       0123456
var  myString       =       "INT 222";
var myArray         =       myString.split('2');
```

A split on 2 will return the following:

```
myArray[0] // element 0 returns INT    (actually: "INT ")
myArray[1] // element 1 returns         (actually: "" – empty string)
myArray[2] // element 2 returns         (actually: "" – empty string)
myArray[3] // element 3 returns     (actually: "" – empty string)
```

```
myArray = myString.split('22'); // - a split on 22 will return the following:

myArray[0] // element 0 returns   INT
myArray[0] // element 1 returns   2
```

split(x)

```
Position/index    »      0123456
var  myString     =      "INT 222";
var myArray       =      myString.split('222');
```

A split on 222 will return the following:

myArray[0] // element 0 = INT // (“INT ”)
myArray[0] // element 1 = // “”

JS String object - methods

- substr(x, y)
  - The substr(x, y) returns a sub string where:
    x is the from
    y is the length
  - Syntax: stringName.substr(x,y)

```
Position/index      »          012345
var  myString       =        "INT222";
```

```
myString.substr(1,4) // returns  NT22
myString.substr(4,4) // returns  22
myString.substr((myString.length-4),1) // returns  T
myString.substr(4) // returns  22
```

JS String object - methods

- substring(x, y)
  - The substring(x,y) returns a sub string where:
    - x starting from (index) - inclusive - if x > than y, then switch
    - y to (index) - not inclusive - if y < than x, then switch

```
Position/index      »          012345
var  myString       =         "INT222";
```

```
myString.substring(1,4)  // returns  NT2
myString.substring(4,4)  // returns
myString.substring(4,2)  // returns  T2
myString.substring(-4,4) // returns  INT2
myString.substring(0,6)  // returns  INT222
myString.substring((myString.length-1),1) // returns  NT22
myString.substring(4)    // returns  22
```

JS String object - methods

- toLowerCase()
  - Converts a string to lower case
  - Syntax: stringName.toLowerCase()

```
Position/index    »        012345
var  myString     =       "INT222";
```

myString.length returns 6
myString.toLowerCase() // returns int222

JS String object - methods

- toUpperCase()
  - Converts a string to upper case
  - Syntax: stringName.toUpperCase()

```
Position/index    »        012345
var  myString     =       "INT222";
```

myString.length returns 6
myString.toUpperCase() returns INT222

JS String object - methods

- trim()
  - The trim() method removes whitespace (blank characters) from the left and right of the string.
  - trimLeft() & trimRight() methods work with some browsers but not others - Don't use them.
  - Syntax: stringName.trim()

```
var  myString    =    "     INT222     "

    // The length of myString is 14
    myString = myString.trim();
    alert( myString );          // INT222
    alert( myString.length ); // 6
```

JS String object - property

- prototype
  - Allows you to add properties and methods to an object
- e.g.

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

var myString = "INT222";
alert(myString.reverse());  // 222TNI
```

JavaScript RegExp Object

- Regular expressions are patterns used to match character combinations and perform search-and-replace functions in strings.
- In JavaScript, regular expressions are also objects.
- RegExp
  - is short for regular expression.
  - is the JavaScript built-in object.

- Regular Expressions Tutorial

Creating RegExp Object

- Syntax

```
var patt=/pattern/[modifiers];
```

or :

```
var patt=new RegExp(pattern[, modifiers]);
```

  - Pattern: the text of the regular expression.
  - Modifiers: if specified, modifiers can have any combination of the following values:
    - **g** - global match
    - **i** - ignore case-sensitivity
    - **m** - multiline;

String Method – match(RegExp)

- A String method that executes a search for a match in a string.
- It returns an array of the found text value. or null on a mismatch.
- Example 1

```
var str = "Welcome to Toronto";
var patt1 = /to/i;     // i: ignore case-sensitivity
// same as:  var patt1 = new RegExp("to", "i");
var result = str.match(patt1);
alert(result);                  // to
alert(result[0]);               // to
```

String Method – match(RegExp)

- Example 2

```
var str = "Welcome to Toronto";
var patt1 = /to/g;        // g: do a global search
var result = str.match(patt1);
alert(result);            // to,to
```

- Example 3

```
var str = "Welcome to Toronto";
var patt1 = /to/ig;       // ig: global and case-insensitive
var myArray = str.match(patt1);
alert(myArray);    // to, To, to
alert(myArray[1]);    // To
```

String Method – match(RegExp)

```
Position/index    »        012345
var  myString     =      "INT222";
```

myString.match(/I/g) - returns I
myString.match(/2/g) - returns 2,2,2
myString.match(/2/) - returns 2
myString.match(/[A-Z]/) - returns I
myString.match(/[A-Z]/g) - returns I,N,T
myString.match(/\D/) - returns I
myString.match(/\D/g) - returns I,N,T
myString.match(/\d/) - returns 2
myString.match(/\d/g) - returns 2,2,2

String Method – replace(RegExp, replacement)

- A String method that executes a search for a match in a string, and replaces the matched substring with a replacement substring.

```
Position/index    »        012345
var  myString     =      "INT222";
```

myString.replace(/222/,"322") returns INT322

String Method – search(RegExp)

- A String method that tests for a match in a string. It returns the index of the match, or -1 if the search fails.

```
Position/index    »        012345
var  myString     =      "INT222";

myString.search(/222/)   // returns 3
myString.search(/I/)     // returns 0
myString.search(/322/)   // returns -1
```

Similar to indexOf and lastIndexOf methods

String Method – split(RegExp)

- A String method that uses a regular expression or a fixed string to break a string into an array of substrings.

RegExp Method – test(str)

- A RegExp method that tests for a match in a string.
- It returns true or false.
- Example

```
var str = "Welcome to Toronto";
var patt1 = /Me/; // or: new RegExp("Me");
var patt2 = /Me/I // or: new RegExp("Me","i");
var result1 = patt1.test(str);
var result2 = patt2.test(str);
alert(result1 + "\n" + result2);   // false
                                   // ture
```

RegExp Examples

- To validate: minimum of 4 alphabetical numbers
  - var pattern1 = /^[a-zA-Z]{4,}$/;
- To validate: telephone format ###-###-####
  - var pattern2 = /^([0-9]{3}[-]){2}[0-9]{4}$/;
- Example

```
var patt1 = /^[a-zA-Z]{4,}$/;
while(true){
  var str = prompt("Please enter your last name","");
    if(str) {
        if(patt1.test(str))
            alert("Your Last Name: " + str);
        else
            alert("Characters only and at least 4! Try
again.");
    }
    else break;
}
```

Special characters in regular expressions

character	meaning
^	String begin
$	String end
.	Match – one character
?	Match – zero or one (preceding character)
*	Match – zero or more
+	Match – one or more
{m, n}	Match – m or n number of preceding character
[ ]	Character sets delimiters [ ]
\d	= [0-9], Match - digits
\D	= [^0-9], Match – non-digits
\w	= [A-Za-z0-9_], Match – alphanumeric
\s	= [\t\r\n], Match – whitespace

Array Object

- A group of variables that use an index (a number) to distinguish them from each other.
  - In JavaScript, variables in an array may have different data type.
- Items in an array are given the same name and are referenced by the name and the index (the occurrence).
- Index starts from 0.

Creating Arrays

- Using an array literal (recommended)
  - e.g. 1

    var arrayName1 = [11, 15, 13, "blue", 24, 35, 05];
  - e.g. 2

    var arrayName2 = [];
    arrayName2.push("brown");
    arrayName2.push(100);
- Using the new keyword
  - e.g. 1

    var arrayName1 = new Array (11, 15, 13, "blue", 24, 35, 05);
  - e.g. 2

    var arrayName2 = new Array(); // or: var arrayName2 = new Array(4);
    arrayName2[0] = "brown";
    arrayName2[2] = 100;

Array object - properties and methods

Member	Type	Example
length	property	arrayName.length
prototype	property	Allows you to add properties and methods to an Array object
concat()	method	arrayName. concat()
join()	method	arrayName.join() or arrayName.join("+")
pop()	method	arrayName.pop()
push()	method	arrayName.push()
reverse()	method	arrayName.reverse()
slice()	method	arrayName.slice(x,y)
sort()	method	arrayName.sort()
splice()	method	splice(x,y,[....,....])
for loop		for and for each element in array

JS Array object - property

- length
  - The length property returns the number of elements / occurrences in the array

```
var arrayName1 = new Array (11, 15, 13, "blue", 24, 35, 05);
```

The arrayName1.length returns 7

```
var arrayName2 = new Array();
    arrayName2[0] = "brown";
    arrayName2[1] = "blue";
    arrayName2[2] = 15;
    arrayName2[3] = "red";
```

The arrayName1.length returns 4

```
var arrayName3 = [“Red", “Green", “Blue", 3];
```

JS Array object - methods

- concat()
  - The concat() method Concatenates two or more arrays and returns a new array
  - Syntax: arrayName.concat(anotherArray, …)

```
var arrayName1 = ["Red", "Black", "Yellow", "Blue“];
var arrayName2 = new Array (1, 2, 3, "Blue");

var newArray = arrayName1.concat(arrayName2) ;

//   newArray will contains the elements of
//   Red,Black,Yellow,Blue, 1,2,3,Blue
```

JS Array object - methods

- join()
  - The join() method is used to join all the elements of an array into a single string separated by a specified string separator
    - if non separator is specified, the default is a comma.
    - compare with split() method of a String. …?
  - Syntax: arrayName.join(str)

```
var arrayName1 =   new Array ("Red", "Black", "Yellow", "Blue");
```

The arrayName1.length returns 4
arrayName1.join() will yield Red,Black,Yellow,Blue
arrayName1.join('+') will yield Red+Black+Yellow+Blue
arrayName1.join(' ') will yield Red Black Yellow Blue
arrayName1.join('-') will yield Red-Black-Yellow-Blue

JS Array object - methods

- pop()
  - The pop() method removes an entry from the end of the array and return the removed element.
  - Syntax: arrayName.pop()

```
var colors = ["Red", "Black", "Yellow", "Blue“];
var removed= colors.pop(); // will remove Blue from the array.

alert(colors); // Red,Black,Yellow
alert(removed); // Blue
```

JS Array object - methods

- push()
  - The push() method adds an entry to the end of the array
  - Syntax: arrayName.push(newEntry)

```
var colours = new Array ("Red", "Black", "Yellow", "Blue");
colours.push("Pink");  // will add Pink to the end of the array.
alert(colours); // Red,Black,Yellow,Blue,Pink
```

JS Array object - methods

- reverse()
  - The elements in the array are reversed. First becomes last, second becomes second last, etc..
  - Syntax: arrayName.reverse()

```
var arrayName1 = ["Red", "Black", "Yellow", "Blue"];
alert(arrayName1 ); // Red,Black,Yellow,Blue

arrayName1.reverse();
alert(arrayName1 ); // Blue,Yellow,Black,Red
```

JS Array object - methods

- sort()
  - The elements in the array are sorted based on their ASCII code.
  - Syntax: arrayName.sort()

```
var arrayName1 = ["Red", "Black", 15, "Yellow", 101, "Blue“];
```

The arrayName1.length returns 6
The array before : Red,Black,15,Yellow,101,Blue
arrayName1.sort();
The array after: 101,15,Black,Blue,Red,Yellow

JS Array object - methods

- slice()
  - The slice() method extracts part of an array and returns a new array.
  - Syntax: arrayName.slice(index1, index2)

```
var colours = new Array ("Red", "Black", "Yellow", "Blue");

alert( colours.slice(1,2) ); // Black
alert( colours.slice(0,3) ); // Red,Black,Yellow

alert(colours);              // Red,Black,Yellow,Blue
```

JS Array object - methods

- splice()
  - The splice() method adds/removes array entry/entries and returns a new array.
  - Syntax:
    arrayName.splice(index,howMany,[entry/entries])
    - index = start at this location in the array.
    - howMany = the number of array elements to be removed. If howMany is 0, no elements are removed.
    - entry/entries = elements to add to the array. If not present, splice removes elements from the array.

Example

```
var colors = ["Red", "Black", "Yellow", "Blue"];
alert(colors); // Red,Black,Yellow,Blue

colors.splice(1,0,"Pink"); // Use the splice method to insert an array entry
alert(colors); // Red,Pink,Black,Yellow,Blue

colors.splice(3,0,"Green", "White"); // Use the splice method to insert an array entry
alert(colors); // Red,Pink,Black,Green,White,Yellow,Blue

colors.splice(3,1); // Use the splice method to remove an array entry - the Green

alert(colors); // Red,Pink,Black,White,Yellow,Blue

colors.splice(3,2); // Use the splice method to remove 2 array entries - the White & Yellow
alert(colors); // Red,Pink,Black,Blue
```

JavaScript Array: for and for in loop

```
var myColors = ["Pink", "Red", "Orange", "Blue"];

function showArray1() { // for loop
  var message = "function showArray1()\n\n";
  for (var x=0; x < myColors.length; x++) { // recommended
    message+= myColors[x] + "\n";
  }
  alert(message);
} // end of function

function showArray2() { // for in loop  is not recommended for Array
  var message = "function showArray2()\n\n";
  for (var x in myColors) {
    message+= myColors[x] + "\n";
  }
  alert(message);
} // end of function

showArray1();
showArray2();

// advanced: use forEach() method of Array object to do the same task
```

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) and
    - bracket notation (obj**['x']** = 10).
  - Object properties and functions/methods can be added, changed, or deleted at run-time.

Creating (Custom) Objects

- Using literal notation

```
var person1 = { name: "John", age: 30 };
//var person1 = { "name": "John", "age": 30 };

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

alert( person1.name );

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

Creating Objects

- Using a function constructor

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

var person1 = new Person("Steven", 30);
alert( persion1.age);
person1.talk(); // My name is Steven, and I'm 30 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.
  - The value of this, when used in an object, is the object itself.

Creating Objects

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

```
var person1 = {};
// equivalent to: var person = new Object();

person1.name = "Steven";
person1.age = 30;
person1.talk = function () {
    alert('My name is ' + this.name + ", and I'm "
        + this.age + " years old.");
};

alert(person1.name);
person.talk();
```

Advanced: JS Object and 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); //instanciate the Person class
var myName = person1.getName();
alert(myName); // John

person1.setAge(20);
alert(person1.getAge());
```

- This is actually a usage of JavaScript closure.

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, each object has a property named prototype.
- This prototype property is an object, from which the current object "inherits".
- Prototype chain: as an object, a prototype property also has its own prototype property.
- The Object.prototype is on the top of the prototype chain.
- All JavaScript objects inherits from Object.prototype.

Creating New Objects

- Using Object.create()

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

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

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

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

Creating New Objects

- Using the prototype property

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

function ColoredRectangle(color) {
  this.color = color;
}

ColoredRectangle.prototype = rectangle1; // set prototype for function constructor
ColoredRectangle.prototype.show = function () {
   return 'dimensions: ' + this.width + " x " + this.height + " \ncolor: " + this.color;
};                                                    // add the method at run-time

var triangle2 = new ColoredRectangle("blue");

alert(triangle2.show());
```

JS OOP Example

- 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 int222 = Object.create(subject);
int222.code = 'INT222';
int222.desc = 'Internet I - Internet Fundamentals ';
int222.prog = ['CPD', 'CPA'];
int222.info = { hours: 4, url:'http://scs.senecac.on.ca/course/int222' };
```

JS OOP Example

```
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

- All subjects

```
// Create a collection of all subject objects
var all = [int222, bti220, ipc144, 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);
```

Another Example

- 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;
    }
};
```

Another Example

- 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';
alert(stu1.name);
var x =stu1.prnt();
alert(x);
```

Another Example

```
// 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";

alert(tch1.name+ ", " + tch1.offc);
var x =tch1.prnt();
alert(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!