INT222 - Internet Fundamentals

Week 9: JavaScript String, RegExp and Array Objects

(Version 2)

Agenda

- String Object
- RegExp Object
- Array Object
- Lab 4

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

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.

Character access

There are two ways to access an individual character in a string:

- Using charAt() method

```
var example1 = "INT222";
alert(example1.charAt(2)); // return T
```

- Using array index

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

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.

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

View source

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** View source

About Unicode

- Unicode provides a unique number for every character, no matter what the platform is.
- See unicodes table.

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
```

View source

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
```

View source

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

View source

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

View source

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

View source

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 - element 0 returns INT
myArray - element 1 returns 222

View source

split(x)

Position/index » 0123456

```
var  myString  =  "INT 222";
var myArray    =  myString.split('2');
```

A split on 2 will return the following:

myArray - element 0 returns INT (actually: "INT ")
myArray - element 1 returns (actually: "" – empty string)
myArray - element 2 returns (actually: "" – empty string)
myArray - element 3 returns (actually: "" – empty string)

View source

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

myArray - element 0 returns INT
myArray - element 1 returns 2

View source

split(x)

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

A split on 222 will return the following:

myArray - element 0 = INT
myArray - element 1 =

View source

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

View source

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

View source

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

View source

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

View source

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 value    =    "      INT222      "
```

The length of myString is 14
myString = myString.trim(); returns INT222

View source

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

or :

```
var patt=/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

View source

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

View source

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 the new keyword
  - e.g. 1

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

  - e.g. 2

```
var arrayName2 = new Array();
// var arrayName2 = new Array(4);
    arrayName2[0] = "brown";
    arrayName2[1] = "blue";
    arrayName2[2] = 15;
    arrayName2[3] = "red";
```

- Using an array literal
  - e.g.

```
var arrayName3 = [11, 15, 13, "blue", 24, 35, 05];
```

Array object - properties and methods

Member	Type	Example
length	property	arrayName.length
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 View source

```
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
```

View source

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

View source

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
```

View source

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
```

View source

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
```

  - View source

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

View source

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
```

View source

JS Array object - methods

- splice()
  - The splice() method adds/removes array entry/entries and returns a new array.
  - Syntax:
    arrayName.splice(index,homany,[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.

splice()

```
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
```

View source

JavaScript - array for and for in loop

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

function showArray1() {
   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() {
   var message = "function showArray2()\n\n";
   for (var x in myColors) {
      message+= myColors[x] + "\n";
   }
   alert(message);
} // End of function

showArray1();
showArray2();
```

Resourceful Links

- Standard built-in objects - JavaScript | MDN
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects
- Regular Expressions Tutorial

Thank You!