Chapter 07
JavaScript Advanced

Open Source SW Development
CSE22300

Function

Function

- **Reusable code which can be called anywhere**
  - **Eliminates the need of writing the same code again and again**
  - **Divides a big program into a number of small and manageable functions**

- **Definition**
  - **function keyword**
  - **list of parameter**
  - **statement**

```
function functionname(parameter-list)
{
  statements
}
```

```
function sayHello()
{
   alert("Hello there");
}
```

Calling a Function

- **To invoke a function**
  - **Write the name of that function**

```
function sayHello()
{
   console.log ("Hello there!");
}

sayHello();
```

Parameters

- **Multiple parameters separated by comma**

```
function sayHello(name, age)
{
   console.log (name + " is " + age + " years old.");
}
sayHello('Zara', 7);
```

Return Statement

- **Return a value from a function**

```
function concatenate(first, last)
{
   var full;
   full = first + last;
    return full;
}
function secondFunction()
{
   var result;
   result = concatenate('Zara', 'Ali');
   console.log (result );
}
secondFunction();
```

Function() Constructor

- **Define your function dynamically using Function() constructor along with the new operator**
  - **anonymous functions**

```
var variablename = new Function(Arg1, Arg2..., "Function Body");
```

```
var func = new Function("x", "y", "return x*y;");
function secondFunction() {
    var result;
    result = func(10,20);
    console.log(result);
}
secondFunction();
```

Function() Literals

- **Another new way of defining functions.**
  - **Unnamed function**

```
var variablename = function(Argument List){
   Function Body
};
```

```
var func = function(x,y){ return x*y };
function secondFunction() {
   var result;
   result = func(10,20);
   console.log ( result );
}
secondFunction();
```

Objects

Objects

- **JavaScript is an OOP language**
- **Objects are composed of properties and methods**

- **Object Properties**
  - **Can be any of the three primitive data types, or any of the abstract data types, such as another object**
  - **Are used internally in the object's methods, but can also be globally visible variables**

- **Object Methods**
  - **Functions that let the object do something**

User-Defined Objects

- **new operator**
  - **create an instance of an object**

```
var employee = new Object();
var books = new Array("C++", "Perl", "Java");
var day = new Date("August 15, 1947");
```

- **Constructor**
  - **Creates and initializes an object**
  - **JavaScript provides a special constructor function**
    - **Object()**

```
var book = new Object();
book.subject = "Perl";
book.author = "Mohtashim";
console.log("Book name is : " + book.subject + "<br>");
console.log("Book author is : " + book.author + "<br>");
```

User-Defined Objects

- **User-Defined Constructor**
  - **create an instance of an object**

```
function book(title, author) {
    this.title = title;
    this.author = author;
}
var myBook = new book("Perl", "Mohtashim");
console.log("Book title is : " + myBook.title);
console.log("Book author is : " + myBook.author);
```

User-Defined Objects

- **Defining Methods**
  - **create an instance of an object**

```
function addPrice(amount) {
   this.price = amount;
}
function book(title, author) {
   this.title = title;
   this.author = author;
   this.addPrice = addPrice;
}
var myBook = new book("Perl", "Mohtashim");
myBook.addPrice(100);
console.log("Book title is : " + myBook.title);
console.log("Book author is : " + myBook.author);
console.log("Book price is : " + myBook.price);
```

Number

Number

- **Number Object**
  - **Represents numerical date, either integers or floating-point numbers**
- **Syntax**

```
var val = new Number(number);
```

Number

- **Property**

Property	Description
MAX_VALUE	The largest possible value a number
MIN_VALUE	The smallest possible value a number
NaN	Equal to a value that is not a number
NEGATIVE_INFINITY	A value that is less than MIN_VALUE
POSITIVE_INFINITY	A value that is greater than MAX_VALUE
prototype	The prototype property allows you to add properties and methods
constructor	It returns a reference to the Number function that created the instance's prototype

Number

- **Methods**

Method	Description
toExponential()	Represents a number in exponential notation.
toFixed()	Represents a number in fixed-point notation.
toLocaleString()	Converts a number to a string by using the current or specified locale.
toPrecision()	Represents a number either in exponential or fixed-point notation with a specified number of digits.
toString()	Returns the string representation of the number's value.
valueOf()	Returns the number's value.

Boolean

Boolean

- **Represents two values, either "true" or "false".**
- **Value parameter is omitted or is 0, -0, null, false, NaN, undefined, or the empty string (""),**
  - **False**

```
var val = new Boolean(value)
```

Boolean

- **Property**

Property	Description
prototype	The prototype property allows you to add properties and methods
constructor	It returns a reference to the Number function that created the instance's prototype

String

String

- **String Object**
  - **JavaScript's string primitive data type with a number of helper methods**
- **Syntax**

```
var val = new String(string);
```

String

- **Property**

Property	Description
length	Returns the length of the string
prototype	The prototype property allows you to add properties and methods
constructor	It returns a reference to the Number function that created the instance's prototype

- **Length**

```
var str = new String( "This is string" );
console.log("str.length is:" + str.length);
```

String

- **Methods**

Method	Description
charAt()	Returns the character at the specified index.
charCodeAt()	Returns a number indicating the Unicode value of the character at the given index
concat()	Combines the text of two strings and returns a new string
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found
lastIndexOf()	Returns the index within the calling String object of the last occurrence of the specified value, or -1 if not found
localeCompare()	Returns a number indicating whether a reference string comes before or after or is the same as the given string in sorted order

String

- **Methods**

Method	Description
match()	Used to match a regular expression against a string
replace()	Used to find a match between a regular expression and a string, and to replace the matched substring with a new substring
search()	Executes the search for a match between a regular expression and a specified string
slice()	Extracts a section of a string and returns a new string
split()	Splits a String object into an array of strings by separating the string into substrings.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.

String

- **Methods**

Method	Description
substring()	Returns the characters in a string between two indexes into the string
toLocaleLowerCase()	The characters within a string are converted to lower case while respecting the current locale.
toLocaleUpperCase()	The characters within a string are converted to upper case while respecting the current locale
toLowerCase()	Returns the calling string value converted to lower case
toString()	Returns a string representing the specified object.
toUpperCase()	Returns the calling string value converted to uppercase
valueOf()	Returns the primitive value of the specified object

String

- **charAt()**
  - **Returns the character from the specified index**

```
string.charAt(index)
```

```
var str = new String( "This is string" );
console.log("str.charAt(0) is:" + str.charAt(0));
```

- **Concat()**
  - **Returns a single concatenated string**

```
string.concat(string2, string3[, ..., stringN]);
```

```
var str1 = new String( "This is string one" );
var str2 = new String( "This is string two" );
var str3 = str1.concat( str2 );
console.log("Concatenated String :" + str3);
```

String

- **indexOf()**
  - **Returns the index of the found occurrence, otherwise -1 if not found**

```
string.indexOf(searchValue[, fromIndex])
```

```
var str1 = new String( "This is string one" );
var index = str1.indexOf( "string" );
console.log("indexOf found String :" + index );
```

- **lastIndexOf()**
  - **Returns the index of the last found occurrence, otherwise -1 if not found**

```
string.lastIndexOf(searchValue[, fromIndex])
```

```
var str1 = new String( "This is string one and again string" );
var index = str1.lastIndexOf( "string" );
console.log("lastIndexOf found String :" + index );
```

String

- **match()**
  - **Retrieve the matches when matching a string against a regular expression**

```
string.match ( param )
```

```
var str = "For more information, see Chapter 3.4.5.1";
var re = /Chapter/;
var found = str.match( re );
console.log(found );
```

- **replace()**
  - **Replaces the matched substring with a new substring.**

```
string.replace(regexp/substr, newSubStr/function[, flags]);
```

```
var str = "For more information, see Chapter 3.4.5.1";
var re = /Chapter/;
var found = str.replace( re , “aaa”);
console.log(found );
```

String

- **slice()**
  - **Extracts a section of a string and returns a new string**

```
string.slice( beginslice [, endSlice] );
```

```
var str = "Apples are round, and apples are juicy.";
var sliced = str.slice(3, -2);
console.log( sliced );
```

- **split()**
  - **Splits a String object into an array of strings by separating the string into substrings.**

```
string.split([separator][, limit]);
```

```
var str = "Apples are round, and apples are juicy.";
var splitted = str.split(" ", 3);
console.log ( splitted );
```

String

- **substr()**
  - **returns the characters in a string beginning at the specified location through the specified number of characters**

```
string.substr(start[, length]);
```

```
var str = "Apples are round, and apples are juicy.";
console.log ("(1,2): " + str.substr(1,2));
console.log ("(-2,2): " + str.substr(-2,2));
console.log ("(1): " + str.substr(1));
console.log ("(-20, 2): " + str.substr(-20,2));
consloe.log("(20, 2): " + str.substr(20,2));
```

String

- **toLocaleLowerCase()**
  - **Returns a string in lowercase with the current locale.**

```
string.toLocaleLowerCase( )
```

```
var str = "Apples are round, and Apples are Juicy.";
console.log(str.toLocaleLowerCase( ));
```

Array

Array

- **Array Object**
  - **Lets you store multiple values in a single variable**

```
var fruits = new Array( "apple", "orange", "mango" );
var fruits = [ "apple", "orange", "mango" ];
```

Array

- **Property**

Property	Description
length	Reflects the number of elements in an array
input	This property is only present in arrays created by regular expression matches
index	The property represents the zero-based index of the match in the string
prototype	The prototype property allows you to add properties and methods
constructor	Returns a reference to the array function that created the object.

Array

- **Method**

Method	Description
concat()	Returns a new array comprised of this array joined with other array(s) and/or value(s).
every()	Returns true if every element in this array satisfies the provided testing function.
filter()	Creates a new array with all of the elements of this array for which the provided filtering function returns true.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the first (least) index of an element within the array equal to the specified value, or - 1 if none is found.
join()	Joins all elements of an array into a string.

Array

- **Method**

Method	Description
lastIndexOf()	Returns the last (greatest) index of an element within the array equal to the specified value, or - 1 if none is found.
map()	Creates a new array with the results of calling a provided function on every element in this array.
pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reduce()	Apply a function simultaneously against two values of the array (from left-to-right) as to reduce it to a single value.
reduceRight()	Apply a function simultaneously against two values of the array (from right-to-left) as to reduce it to a single value.

Array

- **Method**

Method	Description
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
shift()	Removes the first element from an array and returns that element.
slice()	Extracts a section of an array and returns a new array.
some()	Returns true if at least one element in this array satisfies the provided testing function.
toSource()	Represents the source code of an object
sort()	Sorts the elements of an array.

Array

- **Method**

Method	Description
splice()	Adds and/or removes elements from an array.
toString()	Returns a string representing the array and its elements.
unshift()	Adds one or more elements to the front of an array and returns the new length of the array.

Array

- **concat()**
  - **Returns a new array comprised of this array joined with two or more arrays.**

```
array.concat(value1, value2, ..., valueN);
```

```
var alpha = ["a", "b", "c"];
var numeric = [1, 2, 3];
var alphaNumeric = alpha.concat(numeric);
console.log("alphaNumeric : " + alphaNumeric );
```

- **IndexOf()**
  - **Returns the first index at which a given element can be found in the array**

```
array.indexOf(searchElement[, fromIndex]);
```

Array

- **pop()**
  - **Removes the last element from an array and returns that element.**

```
Array.pop();
```

```
var numbers = [1, 4, 9];
var element = numbers.pop();
console.log("element is : " + element );
```

- **push()**
  - **Appends the given element(s) in the last of the array and returns the length of the new array**

```
Array.push();
```

```
var numbers = new Array(1, 4, 9);
var length = numbers.push(10);
console.log("new numbers is : " + numbers );
```

Array

- **reverse()**
  - **Reverses the element of an array**

```
array.reverse();
```

```
var arr = [0, 1, 2, 3].reverse();
console.log ("Reversed array is : " + arr );
```

- **shift()**
  - **Removes the first element from an array and returns that element**

```
array.shift();
```

```
var element = [105, 1, 2, 3].shift();
console.log ("Removed element is : " + element );
```

Array

- **unshift()**
  - **Adds one or more elements to the beginning of an array and returns the new length of the array**

```
array.unshift( element1, ..., elementN );
```

```
var arr = new Array("orange", "mango", "banana", "sugar");
var length = arr.unshift("water");
console.log("Returned array is : " + arr );
console.log("Length of the array is : " + length );
```