

Strings

Working with strings in JavaScript

JavaScript Fundamentals
Telerik Software Academy
<http://academy.telerik.com>



- ◆ **Strings in JavaScript**
- ◆ **String Wrapper**
- ◆ **String Methods**
 - ◆ **trim, concat, charAt, substr, indexOf**
- ◆ **String Concatenation**
- ◆ **Escaping**
- ◆ **Useful Extensions**
 - ◆ **Trimming**
 - ◆ **Padding**

Strings in JavaScript

String in JavaScript

- ◆ A string is a sequence of characters
 - ◆ Text enclosed in single (' ') or double quotes (" ")

```
var str1 = "Some text saved in a string variable";  
var str2 = 'text enclosed in single quotes';
```

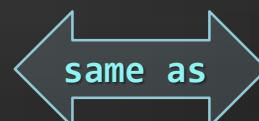
- ◆ String is a primitive type
 - ◆ It is copied / passed by value
- ◆ String is also immutable
 - ◆ Every time a string is changed, a new string is created

String Wrapper

String Wrapper

- ◆ As string is a primitive type, it has an object wrapper type
- ◆ Primitive types keep only their value
 - ◆ When a property is called, the JS engine converts the primitive into its corresponding object type and calls the property

```
var str = 'sample';
str.length;
```



```
var str = new String('sample');
var tempStr = str;
tempStr.length;
```

- ◆ Since primitive type wrappers are of type object, properties can be attached to them

String Wrapper

Live Demo

From Object to Primitive Type

- ◆ JavaScript have a simple parsing
 - ◆ From string to number
- ◆ Conversion from primitive to object type is introduced
 - ◆ `new String('...')` creates a string object
 - ◆ `String(strObject)` creates a primitive string

```
var base = 'string';
var strObj = new String(base);
var str = String(strObj);
```

String Methods

Working with Strings

- ◆ **string.length**
 - ◆ Returns the number of characters in the string
- ◆ **Indexer (string[index])**
 - ◆ Gets a single-character string at location **index**
 - ◆ If **index** is outside the range of string characters, the indexer returns **undefined**
 - ◆ **string[-1]** or **string[string.length]**
- ◆ **charAt(index)**
 - ◆ Gets a single-character string at location **index**
 - ◆ Much like the indexer

- ◆ **string.concat(string2)**
 - ◆ Returns a new string – the concatenation of the two strings
- ◆ **string.replace(str1, str2)**
 - ◆ Replaces first occurrence of str1 with str2
- ◆ **string.search(regex)**
 - ◆ Searches for a substring based on regular expression

String Methods (3)

- ◆ **string.indexOf(substring [,position])**
 - Returns the left-most occurrence of substring in string, that is after position
 - Position is optional and has default value of 0
 - If string doesn't contain substring, returns -1
- ◆ **string.lastIndexOf(substring [,position])**
 - Returns the right-most occurrence of substring in string, that is before position
 - Position is optional, default value is string.length
 - If string doesn't contain substring, returns -1

String Methods (4)

- ◆ **string.split(separator)**
 - ◆ Splits the string by separator and returns an array of strings, containing the separated parts
 - ◆ Separator can be a regular expression
- ◆ **string.trim()**
 - ◆ Removes whitespace from the beginning and end of the string
- ◆ **str.trimLeft(), str.trimRight()**
 - ◆ Remove whitespace from the left/right side of the string

String Methods (5)

- ◆ **string.substr(start, length)**
 - ◆ Returns a substring, starting from start and counting length characters
 - ◆ length is optional
- ◆ **string.substring(start, end)**
 - ◆ Returns a substring, starting from start and ending at end
- ◆ **string.valueOf()**
 - ◆ Returns the primitive value of the object string

String Methods

Live Demo

String Concatenation

How to concatenate strings

String Concatenation

- ◆ String is an immutable type
 - ◆ A value cannot be changed
 - ◆ Instead a new string is created
- ◆ There are a few ways to concatenate strings

```
var strConcat1 = str1 + str2;  
var strConcat2 = str.concat(str2);
```

- ◆ Concatenating strings is slow operation
 - ◆ Each concatenation allocates new memory

String Concatenation (2)

- ◆ String concatenation is one of the most used operations with strings
 - ◆ Yet it is hard to optimize it
 - ◆ Each browser makes optimizations of its own
 - ◆ Old browsers concatenate very slow with +
- ◆ If you support older browsers, use array.join("") for concatenation

```
[].push(str1,str2,str3,...).join('');
```

- ◆ Works like string builder
 - ◆ Slower in modern browsers

String Concatenation

Live Demo

Escaping

- ◆ What is escaping?
 - ◆ Replacing reserved characters with their escape sequence
 - ◆ Prevents JavaScript injection
- ◆ When using JavaScript client-side reserved characters are '<', '>', '&', "''" and "''''

```
var script = document.createElement('script');
script.innerHTML =
    'document.location = \'http://bad_place.com\'';
document.body.appendChild(script);
```

String Escaping (2)

- ◆ Escaping is done by just replacing the reserved characters with their escape sequence
 - ◆ Can be attached to the string prototype

```
String.prototype.htmlEscape = function (){  
    var escapedStr = String(this).replace(/&/g, '&amp;');  
    escapedStr = escapedStr.replace(/</g, '&lt;');  
    escapedStr = escapedStr.replace(/>/g, '&gt;');  
    escapedStr = escapedStr.replace(/"/g, '&quot;');  
    escapedStr = escapedStr.replace(/\'/g, ''');  
    return escapedStr;  
}
```

Escaping

Live Demo

Useful Extensions and Shims

String Extensions - Trim

- ◆ `string.trim()`, `string.trimLeft()`,
`string.trimRight()`
 - Supported in all modern browsers
 - For older browsers use shim
- ◆ `string.trimChars(chars)`,
`string.trimLeftChars(chars)`,
`string.trimRightChars(chars)`
 - Trim no-whitespace characters
 - No native implementation

Trimming

Live Demo

- ◆ **str.padLeft(count [,char]), str.padRight(count [,char])**
 - ◆ Pads a string to the left/right
 - ◆ Fills the padding with whitespace or character
 - ◆ No native implementation

```
String.prototype.padLeft = function (count, char) {  
    char = char || ' ';  
    if(char.length > 1) return String(this);  
    if(str.length >= count) return String(this);  
    var s = String(this);  
    for (var i = 0, len = this.len; i < count - len; i++) {  
        s = char + s;  
    }  
    return s;  
}
```

Padding

Live Demo

Questions?

Free Trainings @ Telerik Academy

- ◆ "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy



- ◆ html5course.telerik.com

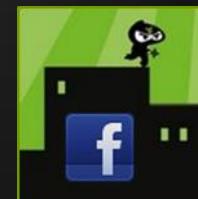
- ◆ Telerik Software Academy

- ◆ academy.telerik.com



- ◆ Telerik Academy @ Facebook

- ◆ facebook.com/TelerikAcademy



- ◆ Telerik Software Academy Forums

- ◆ forums.academy.telerik.com

