JS Webinar: Week 1

JavaScript Now
7 December 2015

Troy Miles

- Troy Miles aka the RocknCoder
- Over 35 years of programming experience
- Wrote a few games for Interplay in the 80's and 90’s
- Speaker and writer on all things web & mobile
- rockncoder@gmail.com
- @therockncoder

Today's Agenda

- Data Types
- Constructors
- Objects
- Arrays
- Functions
- ES5
- Promises

Names

- Start with a letter, underscore, or dollar sign
- Followed by letters, underscores, dollar signs or numbers
- Used for statements, variables, parameters, property names, operators, and labels
- A name can't be a reserved words

ES6 reserved words

- break case class catch const continue debugger
- default delete do else export extends finally for function
- if import in instanceof let new return super switch
- this throw try typeof var void while with yield

Upcoming reserved words

- await
- enum

Strict mode reserved words

- implements interface
- package private protected public static

ES1 - ES3 reserved words

- abstract boolean byte char double
- final float goto int long native
- short synchronized transient volatile

Data types

- Boolean
- Number
- String
- Object
- Array

Boolean

- true or false
- JavaScript also has “falsey” and “truthy”

Falsey values

- false
- null
- undefined
- The empty string ‘’
- The number 0
- The number NaN

Truthy values

- All other values are truthy, including:
  - true
  - the string ‘false’
  - all objects, including empty ones

Numbers

- Single 64-bit floating point, number type
- No separate integer type
- Exponents - 1e3 === 1000
- NaN - not a number
- Based on IEEE 754

NaN, Not a number

- When there is an error while operating with numbers, NaN is generated
- NaN != NaN
- To detect: isNaN(value)

Strings

- Zero or more characters wrapped in either single or double quotes
- The backslash is the escape character
- Characters are 16 bit wide
- Strings are immutable

Escaped characters

Character	Meaning
\b	backspace
\f	form feed
\n	new line
\r	carriage return
\t	tab
\v	vertical tab
\'	single quote (apostrophe)
\"	double quote
\\	backslash

Special symbols

Character	Meaning
\XXX	Latin-1 encoding using 3 octal digits
\xXX	Latin-1 encoding using 2 hexadecimal digits
\uXXXX	Unicode using 4 hexadecimal digits

Special symbols

- Unicode values easier to find than Latin-1
- One resource is: http://unicode-table.com/en/

Type constructors

- DON’T USE THESE!
- Array()
- Object()
- Number(object)
- String(object)
- Boolean()

Objects

- key, value pairs
- the key can be any valid JavaScript string
- if the key is also a valid JavaScript name, can use dot notation
- all keys accessible via bracket notation

Arrays

- Are implement like special purpose objects
- Not true arrays as in other languages
- The length is not necessarily equal to the number of elements

Functions

- Can be treated like values
- They can be passed to functions
- Returned from functions
- Stored in variables

Everything is an object

- In JavaScript arrays, objects, and functions all behave like objects
- You can add properties to arrays and to functions
- Those properties include functions

ECMAScript 5 (ES5)

- Strict mode
- new String methods
- new Date methods
- new Array methods
- JSON object

ECMAScript Versions

Version	Date
ES1	June 1997
ES2	June 1998
ES3	December 1999
ES4	DOA 2006
ES5	December 2009
ES6/ES2015	June 2015
ES2016	2016

Strict Mode

- 'use strict’; or "use strict;"
- First line of file or function
- Can break working code!!
- More stringent checking
- Enables ES5 features

String method

- trim() - removes whitespace from beginning and end
- Allows adds read-only string bracket notation
  - string[0]

Date methods

- .now()
  - Shows the current time without creating an object
  - In milliseconds since 1 Jan 1970
- .toISOString()
  - the time, as a string, in “full” format

JSON

- JSON strings must be double quote
- key/value pairs
- key is any valid JS string
- value is an valid JS data type
- .stringify() - converts object to JSON string
- .parse() - converts JSON string to object

JSON sample

```
1    {
2      "title": "U.S. Citizenship",
3      "tagLine": "Think you can pass this quiz of
basic knowledge of American history?",
4      "added": "2015-07-04T18:25:43.511Z",
5      "rating": 3,
6      "tags": [
7        "history",
8        "geography",
9        "United States"
10     ]
11   }
```

Array functions

- .isArray()
- .every()
- .forEach()
- .indexOf()
- .lastIndexOf()
- .some()
- .map()
- .reduce()
- .filter()

Promises

- A design pattern to help with one kind of async programming
- Return a *Promise* object
- The caller registers callback with the Promise
- ES6 implements Promise/A+

Promise states

- Pending: the results hasn’t been computed
- Fulfilled: the results were successfully computed
- Rejected: a failure occurred
- A promise can only be in one state

ES6 vs jQuery

- jQuery's promises derive from the Deferred object
- ES6's promises are a stand-alone object
- ES6's promises are lighter-weight
- On both you return a promise to call and either resolve() or reject() the promise

In our next episode...

- Design Patterns
- Setting up a developer environment
- JS build pipeline
- Reactive Extension (RxJS)