ES5 / p.2

Whyte @ Webdev Club

JavaScript is ECMAScript *implementation*.

The main difference between JS and ES:
Abstract ES syntax is applied on API(s) given by browser.

With JS we can write code using multiple paradigms:
generic, functional, imperative, object-oriented, prototype-based, event-driven.

Important attributes of <script> tag are:
src, charset, defer, async

Statements, expressions.
Variables.
‘use strict’ directive.

/* and, of course, comments */

typeof operator

primitives: *number, string, boolean, null, undefined, symbol*

objects: *object, function, array*

each type has a *constructor* except undefined and null

we can use *casting* to convert type of value

auto-boxing is what language do on background when we use object-like syntax on primitives

undefined, null

Let’s continue…

Data Types - *number*

typeof 123 === *'number'*
Number constructor
double precision floating point
64 bit
no quotes

Data Types - *number*

0.1 + 0.2

===

...

Data Types - *number*

0.1 + 0.2

===

0.30000000000000004

Data Types - *number*

$$(0.1 * 100 + 0.2 * 100) / 100$$

$$===$$

$$0.3$$

Data Types - *number*

9999999999999999

16 x 9

Data Types - *number*

“IEEE-754”

C, Java, PHP, Ruby…

Data Types - *number*

Math.pow(2, -52)

```
function compareNumbers(n1, n2){
        return Math.abs(n1 – n2) < Math.pow(2, -52);
}
```

Data Types - *number*

123 === 123.0 === 123.

Data Types - *number*

123.toString(2)

Exception: identifier starts immediately after numeric literal

This is same as: 123toString(2)

(123).toString(2)
Or
123.0.toString(2)
Or
123..toString(2)
Or
123 .toString(2)

Data Types - *number*

123 === 0x7b === 0173

Data Types - *number*

1e3 === 1000

1e-3 === 0.001

0.5 === .5

Data Types - *number*

1 / 0 === Infinity

-Infinity

NaN

No exceptions with NaN

Btw it's still 'number'

Data Types - *number*

typeof ’123’ === ‘string’
typeof Number(‘123’) === ‘number’

Data Types - *number*

typeof +’123’ === ‘number’

+’’ === 0

- / * % and more…

Data Types - *number*

typeof parseInt(‘123asd’, 10) === ‘number’
typeof parseFloat(‘ 123.321qwe’, 10) === ‘number’

Data Types - *number*

Whitespace on the *left* or on the *right* side is okay:

```
+‘  123   ‘ === 123
```

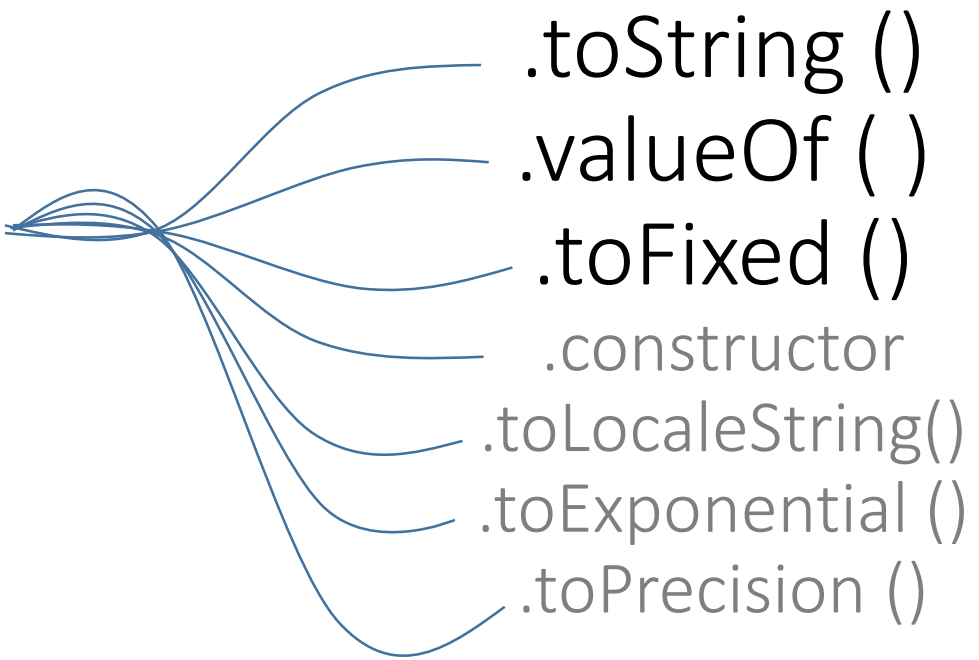
Data Types - *number*

isNaN(), isFinite()
parseInt(), parseFloat()

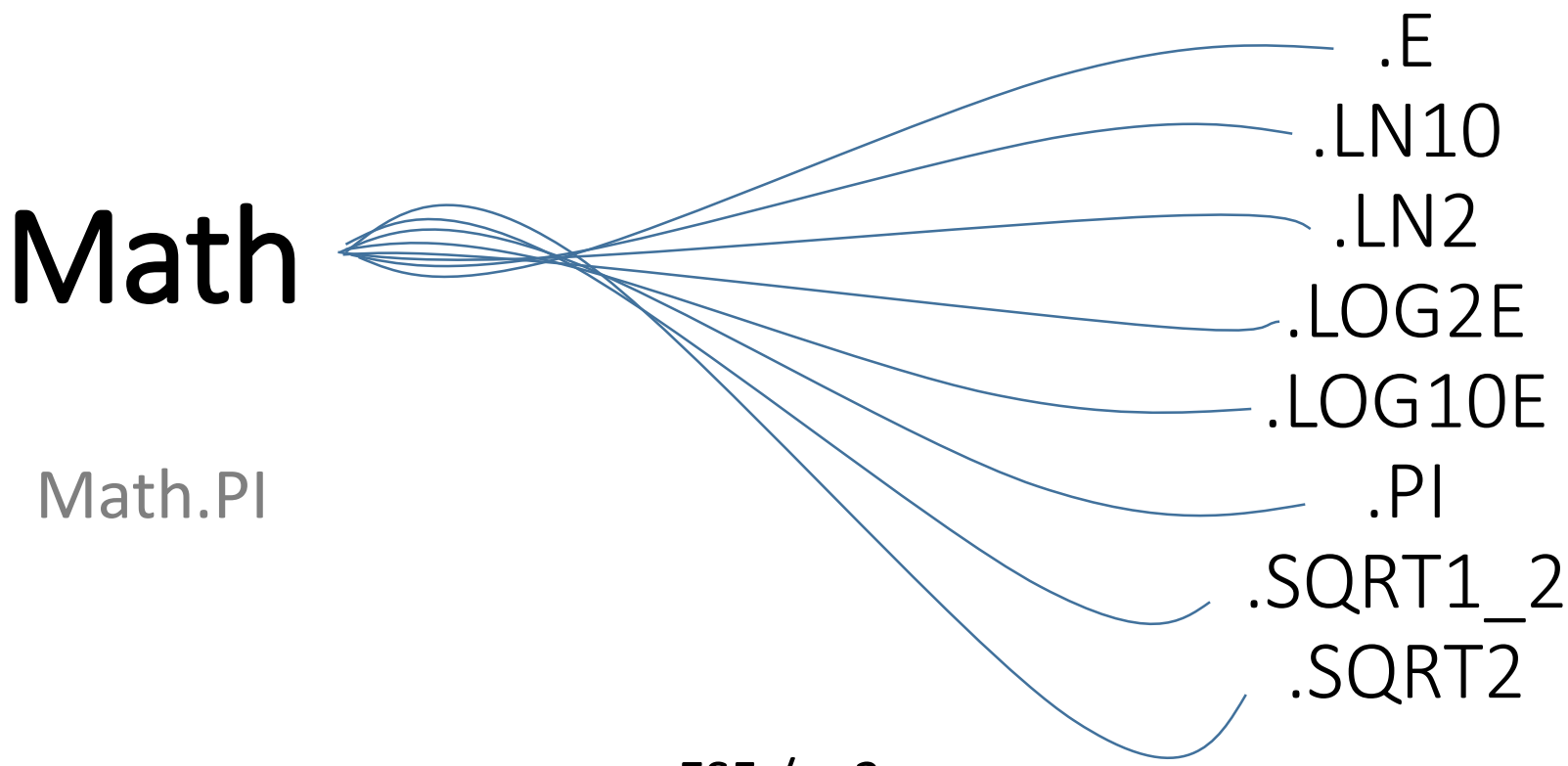
Data Types - *number*

Number.*prototype*

(123).*toString(2)*



Data Types - *number*

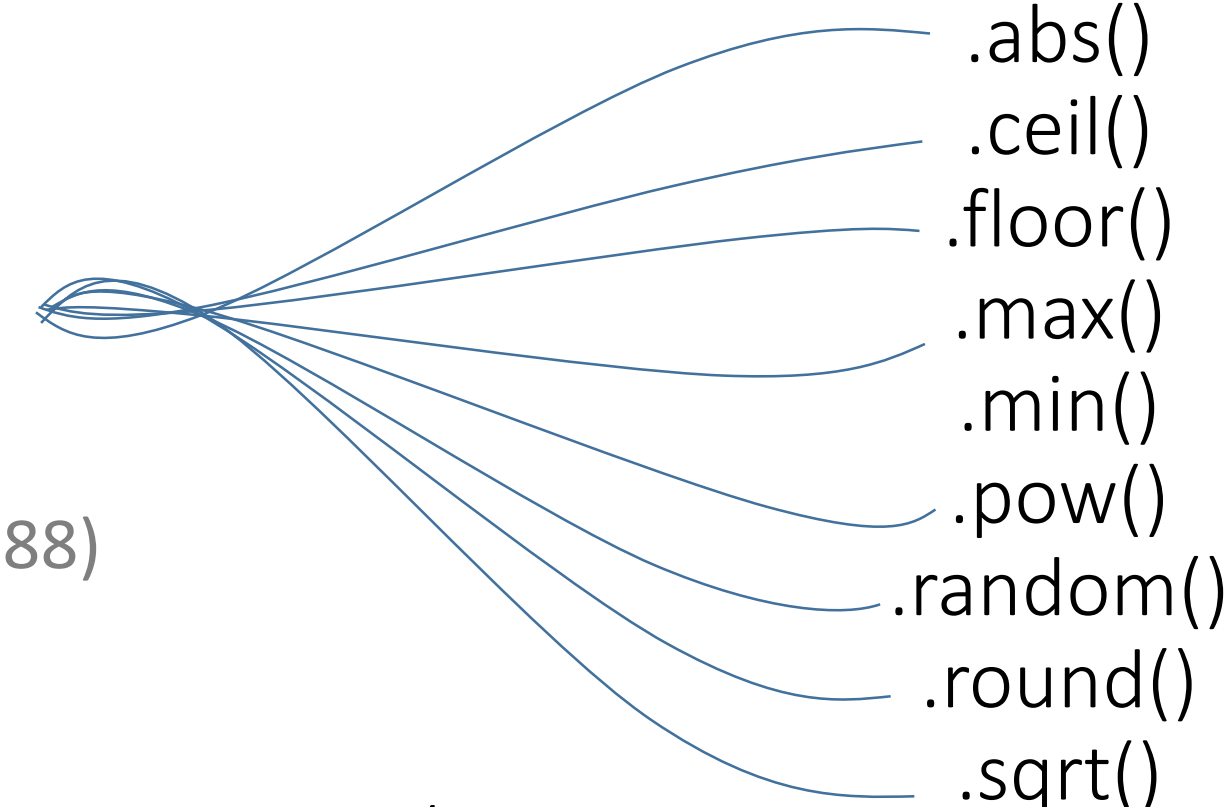


Math.PI

Data Types - *number*

Math

Math.floor(123.88)



Data Types - *number*

Bitwise magic

Data Types - *string*

‘asd’ “asd” `asd`

not an array of chars

String

‘string’

Data Types - *string*

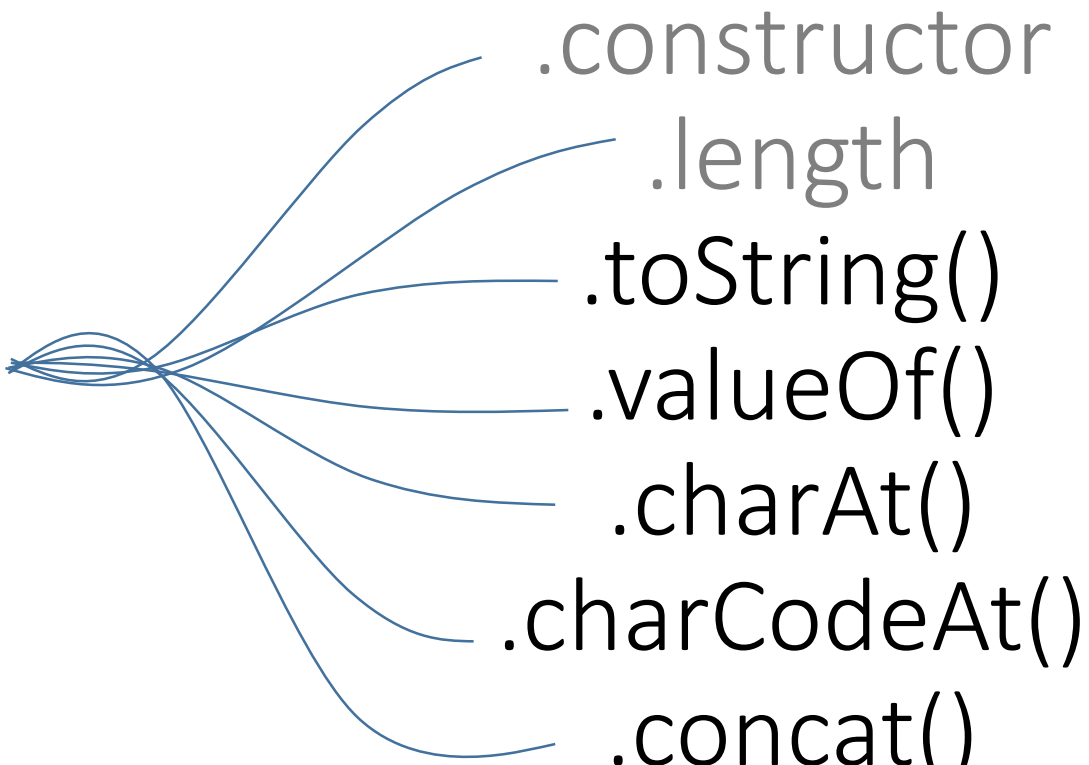
‘\n’, ‘\t’, ‘\r’, ‘\f’, ‘\b’, ‘\\’, ‘\’’

‘\x61\x68\x6f\x6a’ === ‘ahoj’

‘\u0061\u0068\u006f\u006a’ === ‘ahoj’

Data Types - *string*

String.*prototype*

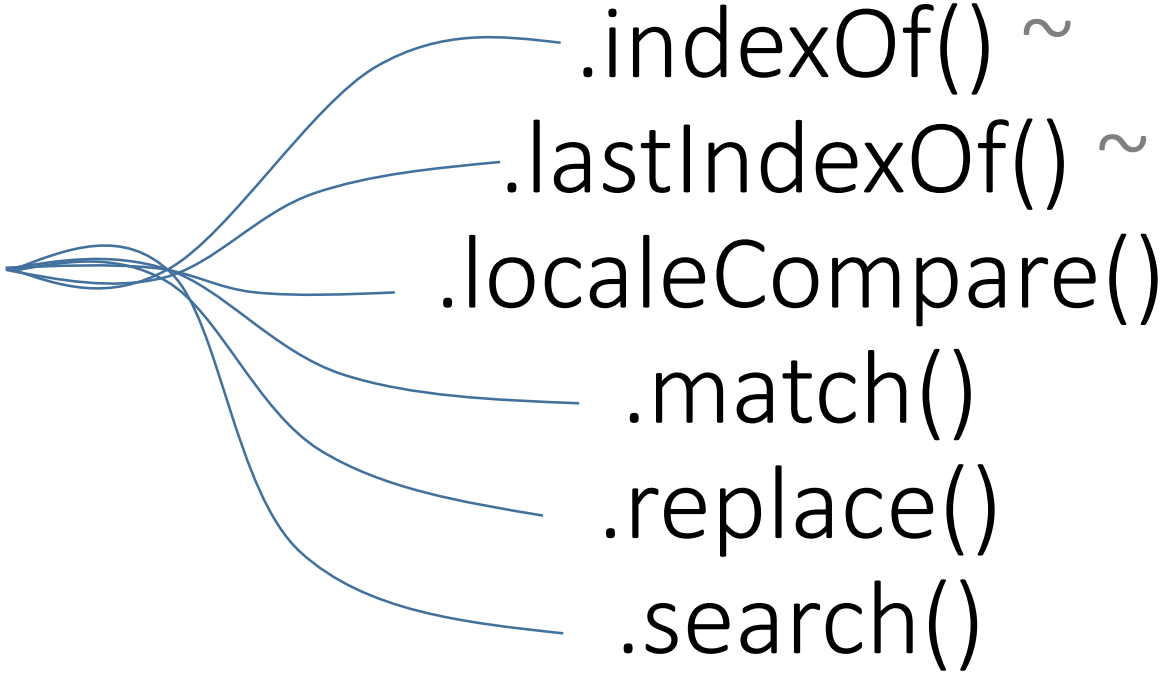


'asd'.*charCodeAt(1)*

Data Types - *string*

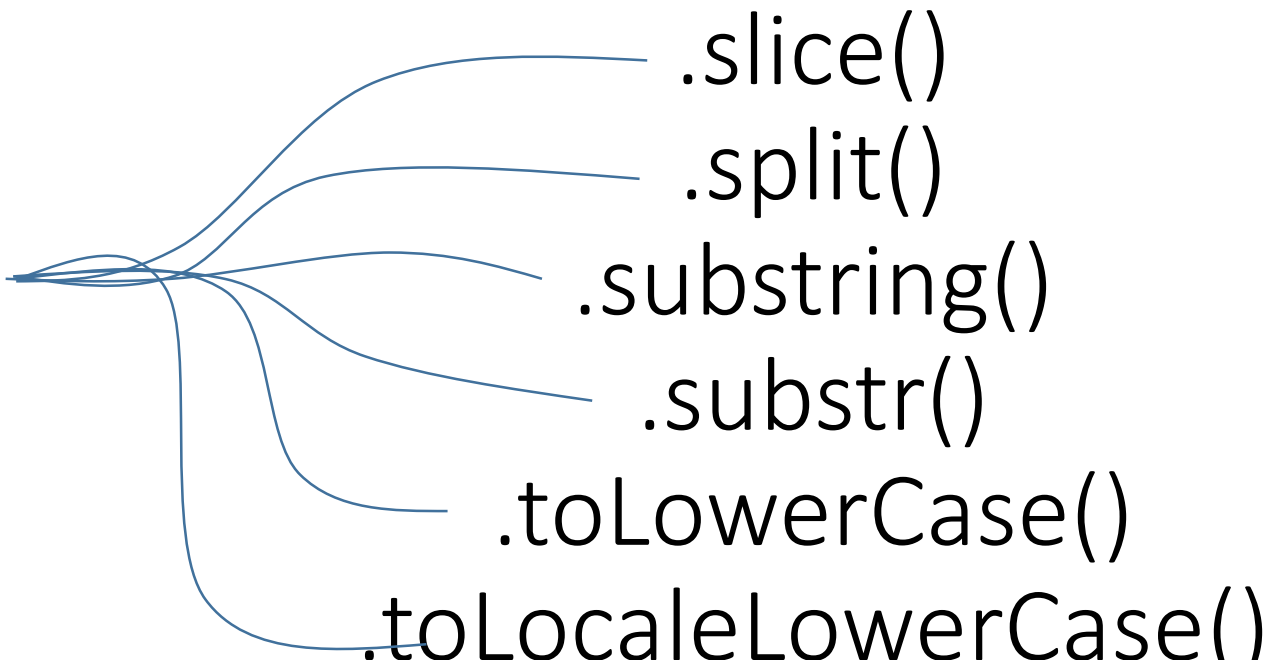
String.*prototype*

‘asd’.*indexOf(‘sd’)*



Data Types - *string*

String.*prototype*



'asd'.*substr(1, 1)*

Data Types - *string*

String.*prototype*

‘ asd ’.*trim()*

Data Types - *string*

String.*fromCharCode()*

String.fromCharCode(97, 104, 111, 106)

Data Types - *string*

“3” > “29999”

Data Types - *boolean*

true

false

!true === false

!false === true

Data Types - *boolean*

Boolean

!!

!0 === true

!1 === false

// end