Web Engineering Front-end Pt. 3

3. JavaScript: Introduction III

Revision

String

- Strings are used to store text data and are represented by being surrounded by quotes (e.g. “This is a string”).

Number

- Numbers are used to store numeric values (e.g. 42, -250, 33.333)

Contents

1. String Methods
2. String Formatting
3. Number Properties and Methods

3.1 Strings Methods

What is a method?

- Some common actions for data types exist as methods to make them more convenient to use.
- We will learn more about how methods work in a future lesson, for now just concentrate on using them.

length

Use .length to find the number of characters in a string.

e.g. “Hello”.length; // 5

length

```
var x = 0;


while (“some text”[x] !== undefined){
        x++;
}


x; // 9
```

length

“some text”.length; // 9

Which is faster?

search()

Use search() to find the position of specific character(s) in a string.

Note: JavaScript will start counting from 0.

e.g.

“Method”.search(“t”); // 2

“Method”.search(“M”); // 0

search()

search() will return -1 if the characters can't be found.

e.g.

"Method".search("a"); // -1

charAt()

charAt() does the opposite of search(), where you specific the position and it returns the character.

e.g.

“Method”.charAt(1); // “e”

slice()

Use slice() to extract a portion of a string and return it using the specified start and end positions.

Note: The end position character is not included in the extracted string.

slice()

```
var string = “One, Two, Three”;

var slice = string.slice(5, 8);

slice; // “Two”
```

slice()

You can also negative numbers for the position, in which case it will be counted from the end of the string.

slice()

```
var string = “One, Two, Three”;

var slice = string.slice(-10, -7);

slice; // “Two”
```

slice()

If you only enter 1 position, then the rest of the string will be sliced out.

e.g.

```
var string = “One, Two, Three”;

var slice = string.slice(5);

slice; // “Two, Three”
```

slice()

e.g.

```
var string = “One, Two, Three”;

var slice = string.slice(-10);

slice; // “Two, Three”
```

concat()

Use concat() to join multiple strings together.

e.g.

```
var x = “Good”;

var y = “bye”;

var z = x.concat(y);

z; // “Goodbye”
```

concat()

You can also use the + operator for the same result.

e.g.

```
var z = “Good” + “bye”;

z; // “Goodbye”
```

concat() example

```
var price = 10;

var string = “This hotdog costs $”+price+”.”;

string; // “This hotdog costs $10.
```

Practice

Spend some time to practice using string methods.

3.2 String Formatting

Escape characters

Because JavaScript strings must be placed between quotes, the following code would be misinterpreted.

var string = "John said, "Good morning!"";

Only the highlighted portion will be considered part of the string to JavaScript.

Escape characters

- Because JavaScript strings must be placed between quotes, the following code would be misinterpreted.

var string = "John said, "Good morning!"";

- Only the highlighted portion will be considered part of the string to JavaScript.
- To avoid this, we can use special escape characters.

Escape characters examples

Code	Output
\\	\ (backslash)
\'	‘ (single quote)
\"	“ (double quote)
\&	& (ampersand)
\t	tab
\n	newline

Escape character example

```
var string = "\'line1\'\nline2";
string;
/*
'line1'
line2
*/
```

Practice: String formatting

Spend some time to practice string formatting. As you get more comfortable, try testing with longer, more complex strings such as multi-line strings.

3.3 Number Properties and Methods

Number types?

Unlike other programming languages such as Python, JavaScript does not separate numbers into different types, such as integers(whole numbers), floating point (decimals). Instead they are always stored as double-precision floating point numbers.

Number precision

JavaScript integers are accurate up to 15 digits.

e.g.

```
var fourteen = 999999999999999;

fourteen; // 999999999999999

var fifteen = 9999999999999999;

fifteen;// 10000000000000000
```

Number precision

Floating point calculations can sometimes be inaccurate in JavaScript.

e.g.

```
var x = 0.2 + 0.3;
x; // 0.5
var y= 0.3 + 0.6;
y; // 0. 8999999999999999
```

Discussion: Fixing floating points

Can you think of a way to prevent inaccuracies when working with floating points?

toPrecision()

Use toPrecision() to round off numbers to a specified length.

e.g.

```
var pi = 3.14159;

pi.toPrecision(3); // 3.14
```

toFixed()

toFixed() is similar to toPrecision(), but you specify the number of decimals to round to instead.

e.g.

```
var pi = 3.14159;

pi.toFixed(3); // 3.141
```

Scientific notation

Scientific notation is also supported in JavaScript.

e.g.

```
var x = 54e+10;

x; // 540000000000
```

toExponential()

Use toExponential to convert a number into scientific notation. You can also specify the number of decimal places to round it to.

e.g.

```
var x = 204227677;
x.toExponential(); // 2.04227677e+8
x.toExponential(3); // 2.042e+8
```

toString()

Use toString(x) to output numbers into base x.

e.g.

var x = 64

x.toString(10); // “64” (Base 10 Default)

x.toString(2): // “1000000” (Base 2 Binary)

x.toString(16): // “40” (Base 16 Hexadecimal)

Integer overflow/underflow

Integer overflow and underflow is when an arithmetic operation tries to return a number that the language cannot represent.

Integer overflow/underflow

Use Number.MAX_VALUE to find the largest value supported by JavaScript, and Number.MIN_VALUE to find the smallest.

Try it yourself!

Infinity

In the event of integer overflow, JavaScript will output Infinity instead.

e.g.

```
var x = Number.MAX_VALUE * 2;
x; // Infinity
var y = 1 / 0;
y; // Infinity
```

Infinity

In the event of integer underflow, JavaScript will output negative Infinity.

e.g.

```
var x = -Number.MAX_VALUE * 2

x; // -Infinity
```

NaN

JavaScript will use NaN (Not a Number) to represent illegal numbers. For example, when you try to perform arithmetic on a non-numeric string.

```
var x = 10 / “Hello”;
x; // NaN
```

NaN

However, the following will NOT return NaN.

```
var x = 10 / “2”;

x; // 5
```

isNaN()

You can use the isNan() function to check if a number is legal or not.

e.g.

isNan(5 + “number”); // True

Practice: Number properties and methods

Spend some time to practice getting number properties and using number methods.

Methods

There are many more methods for all data types, these were just some examples to get you started. Try to look online for some more methods that may be useful for you.

The End

References

Reference 1: W3 Schools JavaScript Tutorial https://www.w3schools.com/js/default.asp

Reference 2: MDN JavaScript reference https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference