Springboard

JS Fundamentals

js-building-blocks.pdf 11557.5KB

Booleans

- *true* or *false*

```
let isLoggedIn = true; let isGameOver = false; // this one
should never change! const isWaterWet = true;
```

Strings

A string is a sequence of Unicode characters. Unlike variable naming, strings support all Unicode including emojis!

As you can see, string literals are defined with either double or single quotes (no difference, up to you!)

```
let internationalStr = "中文 español देवनागरी العربية"; let
smileyFace = '😀';
```

Booleans

Strings

String Properties

String Methods - Slice

String Methods - Includes And Indexof

String Methods - Split

String Methods - Replace

String Concatenati...

Template Literals

Long Strings

Checking Variable Types

String Properties

Strings are immutable: you cannot change them after they are created.

Strings are sequences of characters that have indices, starting from zero.

```
let myName = 'Colt Steele'; myName[0]; // 'C' myName[3]; // 't'
let myName = 'Colt Steele'; myName[3] = 'd'; // THIS DOESN'T
WORK! console.log(myName); // 'Colt Steele';
```

```
.length tells you the number of characters make up the string
let alphabet = 'abcdefghijklmnopqrstuvwxyz'; alphabet.length;
// 26
```

String Methods - Slice

There are a few very important string methods. Note that unlike some of the number methods earlier, these do not have to be on the String() constructor. You can call them directly from a string literal.

.slice(start, end) - return a new string based on slicing the string from the start index up until the end index

```
let idk = 'I have no idea what I\'m doing'; idk.slice(0, 13);
// 'I have no idea';
```

end is optional, if not specified it will be set to string.length.

```
let fun = 'Are you having fun yet?'; fun.slice(15); // 'fun
yet?'
```

You can specify negative numbers to count backwards from the end.

```
let cool = 'I am cool'; cool.slice(-6, -3); // m c
```

String Methods - Includes And Indexof

.includes(substr) - return true if the substring is found within the string

```
let funFact = 'An axolotl is a tiny salamander in Mexico';
funFact.includes('axolotl'); // true
funFact.includes('lizard'); // false
```

.indexOf(substr) - return the string index where the substring exists, or negative 1 if it can't find it within the string

```
let waldo = 'Where\'s Waldo?'; waldo.indexOf('Waldo'); // 8
waldo.indexOf('Blue'); // -1
```

String Methods - Split

.split(separator) - convert a string to an array of substrings (more on arrays later), using the separator to determine what the substrings are

```
let phrase = 'Lannisters always pay their debts.';
phrase.split(' '); // use a space as a separator // [
'Lannisters', 'always', 'pay', 'their', 'debts.' ] Split on an
empty string to get an array of single characters. let cat =
'Blue'; cat.split(''); // [ 'B', 'l', 'u', 'e' ]
```

String Methods - Replace

.replace(regex, newSubstr) - replace characters in a string based on a regular expression with a new substring. A regular expression or RegEx is a language for matching patterns in strings - we won't worry about these for now!

This is very complex because RegExes are hard. Returns a new string.

```
let yay = 'Happy Happy, Joy Joy'; yay.replace(/H/g, 'Fl'); //
'Flappy Flappy, Joy Joy'
```

If you don't do a regular expression with /g, only the first instance of the sub-string is replaced.

```
let funnyString = 'lol'; funnyString.replace('l', 'p'); //
'pol'
```

String Concatenation

The most common operation we want to do with strings is to concatenate them (put multiple strings together).

The traditional way to do this is with the + operator.

```
let myName = 'Colt' + ' ' + 'Steele'; console.log(myName); //
'Colt Steele' This does the same thing as .concat method
hello.concat(' world'); // hello world
```

Template Literals

A fancy way to insert variables into your strings is by using template literals. Template strings use backticks instead of single or double quotes - - ***like this!***

Variables are interpolated with ${}

```
let name = 'Nicholas Cage'; let age = 54; let job = 'actor';
let bio = `${name} is a ${age} year-old ${job}.`;
console.log(bio); 'Nicholas Cage is a 54 year-old actor.';
```

This is more readable than the alternative:

```
let bio = name + ' is a ' + age + ' year-old ' + job + '.';
console.log(bio); 'Nicholas Cage is a 54 year-old actor.';
```

Long Strings

You can have strings that span multiple lines in several ways:

1. Use the + operator at the end or beginning of each line to concatenate multiple strings from different lines together.
2. Use back-slashes in the same string to ignore the line-breaks.

```
let hipsterIpsum = 'Lorem ipsum dolor amet biodiesel, ' + '
viral skateboard next level. Gentrify brooklyn roof party, ' +
' aesthetic distillery pinterest umami semiotics. '; let
hipsterIpsum2 = 'Gastropub vexillologist williamsburg \ pin
tattooed hashtag lo-fi master cleanse. Venmo ennui \ before
they sold out blue bottle pitchfork.';
```

An even easier way is to use backticks!

```
let hipsterIpsum = `` Lorem ipsum dolor amet biodiesel, viral
skateboard next level. Gentrify brooklyn roof party, aesthetic
distillery pinterest umami semiotics. ``
```

Checking Variable Types

You can usually* use typeof to determine what a variable is holding

```
let myName = 'Colt Steele'; typeof myName // "string" let
favoriteNumber = 420; typeof favoriteNumber // "number" let
awesome = false; typeof awesome; // "boolean" let
willILiveForever; typeof willILiveForever // "undefined"
```