Strings

With ES6, the standard library has grown immensely. Along with these changes are new methods which can be used on strings, such as `.includes()` and `.repeat()`.

.includes( )

```
var string = 'food';
var substring = 'foo';

console.log(string.indexOf(substring) > -1);
```

Instead of checking for a return value `> -1` to denote string containment, we can simply use `.includes()` which will return a boolean:

```
const string = 'food';
const substring = 'foo';

console.log(string.includes(substring)); // true
```

.repeat( )

```
function repeat(string, count) {
    var strings = [];
    while(strings.length < count) {
        strings.push(string);
    }
    return strings.join('');
}
```

In ES6, we now have access to a terser implementation:

```
// String.repeat(numberOfRepetitions)
'meow'.repeat(3); // 'meowmeowmeow'
```

Template Literals

Using **Template Literals**, we can now construct strings that have special characters in them without needing to escape them explicitly.

```
var text = "This string contains \"double quotes\" which are escaped.";
```

```
let text = `This string contains "double quotes" which are escaped.`;
```

Template Literals also support interpolation, which makes the task of concatenating strings and values:

```
var name = 'Tiger';
var age = 13;

console.log('My cat is named ' + name + ' and is ' + age + ' years old.');
```

Much simpler:

```
const name = 'Tiger';
const age = 13;

console.log(`My cat is named ${name} and is ${age} years old.`);
```

In ES5, we handled new lines as follows:

```
var text = (
    'cat\n' +
    'dog\n' +
    'nickelodeon'
);
```

Or:

```
var text = [
    'cat',
    'dog',
    'nickelodeon'
].join('\n');
```

Template Literals will preserve new lines for us without having to explicitly place them in:

```
let text = ( `cat
dog
nickelodeon`
);
```

Template Literals can accept expressions, as well:

```
let today = new Date();
let text = `The time and date is ${today.toLocaleString()}`;
```