JavaScript Essentials

Strings

Table of Contents

Lesson Objectives

- Understand the power of text in JavaScript and Front-end Development
- Able to create/concatenate strings
- Understand the difference of strings and numbers
- Able to use template strings concatenate strings

Section 1

Overview

Overview – Power of text

- Words are very important to humans

- Encounter text every where in Front-end Developer

- Letters: a-zA-Z
- Number: 0-0
- Text everywhere

Overview – Summary

- Pretty much all of the programs we've shown you so far in the course have involved some string manipulation.
- Understand strings give us a huge advantage in Programming

Section 2

Strings basics

Strings basics – Create a string

```
1 let string = 'The revolution will not be televised.';
2 string;
```

```
1 let badString = This is a test;
2 let badString = 'This is a test;
3 let badString = This is a test';
```

```
1 let badString = string;
2 badString;
```

Strings basics – Single or Double

```
1 | let sgl = 'Single quotes.';
2 | let dbl = "Double quotes";
3 | sgl;
4 | dbl;
```

```
1 | let badQuotes = 'What on earth?";
```

☹

are okay!

```
1 | let sglDbl = 'Would you eat a "fish supper"?';
2 | let dblSgl = "I'm feeling blue.";
3 | sglDbl;
4 | dblSgl;
```

Strings basics – Escaping characters

- To fix our previous problem code line, we need to escape the problem quote mark. Escaping characters means that we do something to them to make sure they are recognized as text, not part of the code. In JavaScript, we do this by putting a backslash just before the character. Try this:
- let bigmouth = 'I\'ve got no right to take my place...';
bigmouth;

Strings basics – Concatenating strings

- Concatenate is a fancy programming word that means "join together". Joining together strings in JavaScript uses the plus (+) operator, the same one we use to add numbers together, but in this context it does something different. Let's try an example in our console.
  - ✓ `let one = 'Hello, ‘; let two = 'how are you?’; let joined = one + two; joined;`
- In the last instance, we joined only two strings, but you can join as many as you like, as long as you include a + between each pair. Try this:
  - ✓ `let multiple = one + one + one + one + two; multiple;`
- You can also use a mix of variables and actual strings. Try this:
  - ✓ `let response = one + 'I am fine — ' + two; response;`

Strings basics – Numbers and strings

- So what happens when we try to add (or concatenate) a string and a number? Let's try it in our console:
  - ✓ `'Front ' + 242;`
  - ✓ You might expect this to return an error, but it works just fine. Trying to represent a string as a number doesn't really make sense, but representing a number as a string does, so the browser rather cleverly converts the number to a string and concatenates the two strings.

1. You can even do this with two numbers — you can force a number to become a string by wrapping it in quote marks. Try the following (we are using the typeof operator to check whether the variable is a number or a string):
   1. `let myDate = '19' + '67'; typeof myDate;`
2. If you have a numeric variable that you want to convert to a string but not change otherwise, or a string variable that you want to convert to a number but not change otherwise, you can use the following two constructs:
   1. The Number object converts anything passed to it into a number, if it can. Try the following:`let myString = '123'; let myNum = Number(myString); typeof myNum;`
   2. Conversely, every number has a method called toString() that converts it to the equivalent string. Try this:`let myNum = 123; let myString = myNum.toString(); typeof myString;`
3. These constructs can be really useful in some situations. For example, if a user enters a number into a form's text field, it's a string. However, if you want to add this number to something, you'll need it to be a number, so you could pass it through Number() to handle this. We did exactly this in our Number Guessing Game, in line 54.

Practice Strings manipulation

Strings basics - Summary

- Create a string using single quote (‘), double quote (“)
- When your string has single quote or double quote, escape it with \’, \”
- Concatenating strings using + operators
- You can concatenate strings and number with + operators

Section 3

Template strings

Template strings

- Another type of string syntax that you may come across is **template literals** (sometimes referred to as template strings). This is a newer syntax that provides more flexible, easier to read strings.
- To turn a standard string literal into a template literal, you have to replace the quote marks (' ', or " ") with backtick characters (` `). So, taking a simple example:
- let song = 'Fight the Youth';
  Would be turned into a template literal like so:
- song = `Fight the Youth`;

Template strings - Concatenation

- If we want to concatenate strings, or include expression results inside them, traditional strings can be fiddly to write:
- `let score = 9; let highestScore = 10; let output = 'I like the song "' + song + '". I gave it a score of ' + (score/highestScore * 100) + '%.';`
  Template literals simplify this enormously:
- `output = `I like the song "${ song }". I gave it a score of ${ score/highestScore * 100 }%.`;`
  There is no more need to open and close multiple string pieces — the whole lot can just be wrapped in a single pair of backticks. When you want to include a variable or expression inside the string, you include it inside a ${ } construct, which is called a *placeholder*.
- You can include complex expressions inside template literals, for example:
- `let examScore = 45; let examHighestScore = 70; examReport = `You scored ${ examScore }/${ examHighestScore } (${ Math.round((examScore/examHighestScore*100)) }%). ${ examScore >= 49 ? 'Well done, you passed!' : 'Bad luck, you didn\'t pass this time.' }`;`
  The first two placeholders here are pretty simple, only including a simple value in the string.
- The third one calculates a percentage result and rounds it to the nearest integer.
- The fourth one includes uses a ternary operator to check whether the score is above a certain mark and print a pass or fail message depending on the result.

Template strings - Concatenation

- Another point to note is that if you want to split a traditional string over multiple lines, you need to include a newline character, \n:
- output = 'I like the song "' + song + '".\nI gave it a score of ' + (score/highestScore * 100) + '%.';
  Template literals respect the line breaks in the source code, so newline characters are no longer needed. This would achieve the same result:
- output = `I like the song "${ song }". I gave it a score of ${ score/highestScore * 100 }%.`;

Practice Template Strings

Assignment operators - Summary

- Assignment operators provide useful shortcuts to keep our code cleaner and more efficient
- Can use other variables on the right hand side of each expression as well

Thank you

Q&A