Learn web development

Handling text — strings in JavaScript

Previous Overview: First steps Next

Next we'll turn our attention to strings — this is what pieces of text are called in programming. In this article we'll look at all the common things that you really ought to know about strings when learning JavaScript, such as creating strings, escaping quotes in string, and joining them together.

Prerequisites:	Basic computer literacy, a basic understanding of HTML and CSS, an understanding of what JavaScript is.
Objective:	To gain familiarity with the basics of strings in JavaScript.

The power of words

Words are very important to humans — they are a large part of how we communicate. Since the Web is a largely text-based medium designed to allow humans to communicate and share information, it is useful for us to have control over the words that appear on it. HTML provides structure and meaning to our text, CSS allows us to precisely style it, and JavaScript contains a number of features for manipulating strings, creating custom welcome messages, showing the right text labels when needed, sorting terms into the desired order, and so much more.

Pretty much all of the programs we've shown you so far in the course have involved some string manipulation.

Strings — the basics

Strings are dealt with in a similar way to numbers at first glance, but you'll start to see some notable differences when you dig deeper. Let's start by entering some basic lines in a console to familiarize ourselves. We've provided one below (you can also open this console in a separate tab or window, or use the browser developer console if you'd prefer).

```
>
```

Creating a string

1. To start with, enter the following lines:

```
var string = 'The revolution will not be televised.';
string;
```

   Just like we did with numbers, we are declaring a variable, initializing it with a string value, and then returning the value. The only difference here is that when writing a string, you need to surround the value with quotes.

2. If you don't do this, or miss one of the quotes, you'll get an error. Try entering the following lines:

```
var badString = This is a test;
var badString = 'This is a test;
var badString = This is a test';
```

   These lines don't work because any text string without quotes around it is assumed to be a variable name, property name, reserved word, or similar. If the browser can't find it, then an error is raised (e.g. "missing ; before statement"). If the browser can see where a string starts, but can't find the end of the string, as indicated by the 2nd quote, it complains with an error (with "unterminated string literal"). If your program is raising such errors, then go back and check all your strings to make sure you have no missing quote marks.

3. The following will work if you previously defined the variable `string` — try it now:

```
var badString = string;
badString;
```

`badString` is now set to have the same value as `string`.

Single quotes versus double quotes

1. In JavaScript, you can choose single quotes or double quotes to wrap your strings in. Both of the following will work OK:

```
var sgl = 'Single quotes.'
var dbl = "Double quotes";
sgl;
dbl;
```

2. There is very little difference between the two, and which you use is down to personal preference. You should choose one and stick to it, however, differently quoted code can be confusing, especially if you use the different quotes on the same string! The following will return an error:

```
var badQuotes = 'What on earth?";
```

3. The browser will think the string has not been closed, because the other type of quote you are not using to contain your strings can appear in the string. For example, both of these are OK:

```
var sglDbl = 'Would you eat a "fish supper"?'
var dblSgl = "I'm feeling blue.";
sglDbl;
dblSgl;
```

4. However, you can't include the same quote mark inside the string as is being used to contain them. The following will error, as it confuses the browser as to where the string ends:

```
var bigmouth = 'I've got no right to take my place...';
```

This leads us very nicely onto our next subject.

Escaping characters in a string

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:

```
var bigmouth = 'I\'ve got no right to take my place...';
bigmouth;
```

This works fine. You can escape other characters in the same way, e.g. `\"`, and there are some special codes besides. See Escape notation for more details.

Concatenating strings

1. 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.

   ```
   var one = 'Hello, ';
   var two = 'how are you?';
   var joined = one + two;
   joined;
   ```

   The result of this is a variable called `joined`, which contains the value "Hello, how are you?".

2. In the last instance, we just joined two strings together, but you can do as many as you like, as long as you include a + between each one. Try this:

   ```
   var multiple = one + one + one + one + two;
   multiple;
   ```

3. You can also use a mix of variables and actual strings. Try this:

   ```
   var response = one + 'I am fine — ' + two;
   response;
   ```

Note: When you enter an actual string in your code, enclosed in single or double quotes, it is called a **string literal**.

Concatenation in context

Let's have a look at concatenation being used in action — here's an example from earlier in the course:

```
<button>Press me</button>
```

```
var button = document.querySelector('button');

```

```
button.onclick = function() {
  var name = prompt('What is your name?');
  alert('Hello ' + name + ', nice to see you!');
}
```

Press me

Here we're using a `Window.prompt()` function in line 4, which asks the user to answer a question via a popup dialog box then stores the text they enter inside a given variable — in this case `name`. We then use an `Window.alert()` function in line 5 to display another popup containing a string we've assembled from two string literals and the `name` variable, via concatenation.

Numbers versus strings

1. So what happens when we try to add (or concatenate) a string and a number? Let's try it in our console:

   ```
   1 'Front ' + 242;
   ```

   You might expect this to throw 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 together.

2. 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 var myDate = '19' + '67';
   2 typeof myDate;
   ```

3. 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:

   - The `Number` object will convert anything passed to it into a number, if it can. Try the following:

     ```
     1 var myString = '123';
     2 var myNum = Number(myString);
     3 typeof myNum;
     ```

   - On the other hand, every number has a method called `toString()` that will convert it to

the equivalent string. Try this:

```
var myNum = 123;
var myString = myNum.toString();
typeof myString;
```

These constructs can be really useful in some situations, for example if a user enters a number into a form text field it will be a string, but if you want to add it to something say, 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 63.

Conclusion

So that's the very basics of strings covered in JavaScript. In the next article we'll build on this, looking at some of the built-in methods available to strings in JavaScript and how we can use them to manipulate our strings into just the form we want.

Previous Overview: First steps Next