Lesson Weekend

Introduction to Programming (/introduction-to-programming) / JavaScript and Web Browsers (/introduction-to-programming/javascript-and-web-browsers) / Data Types: Detection, Conversion, and Review

Text Cheat sheet

We discussed basic data types right when we got started with JavaScript. However, it's so important to understand and know how to use these data types that we will review them in this lesson. Keep in mind that JavaScript data types are divided into two categories: **primitives** and **objects**. In this lesson, we'll review JavaScript primitives.

We'll also learn about data type **conversion** and **detection**!

Review of JavaScript Data Types: Primitives

You've already seen **numbers**, **strings**, **booleans**, and **undefined**. These are 4 of the 7 basic data types, or **primitives**, in JavaScript. The fifth is **null**, which represents nothingness. Don't worry about

null for now — we'll learn more about it down the road. And as we mentioned before, we won't concern ourselves with **symbols** or **bigint** in this course.

Number

Unsurprisingly, the number type represents numbers.

- `42`
- `3.14`
- `-10`
- `0`
- `NaN` (stands for "not a number", but is considered a number)
- `Infinity`
- `-Infinity`
- `const favoriteNumber = 42;`

You can use certain methods on numbers.

```
> 3.14159.toFixed(2);
"3.14"
```

String

As you've seen, the string type represents text.

- `"hello, world!"`
- `const greeting = "hello, world!";`

You can use certain methods on strings.

```
> "hello".toUpperCase();
"HELLO"
> "hello".charAt(2);
"l"
> "hello".toUpperCase().charAt(2);
"L"
```

Boolean

Booleans can only hold 2 possible values — `true` or `false`.

```
> 5 > 3;
true
> 5 > 10;
false
> "hello".charAt(2) === "e";
false
> const enrolledAtEpicodus = true;
> enrolledAtEpicodus;
true
```

You'll discover soon just how important booleans are to programming.

Undefined

The undefined data type has only one possible value: `undefined`.

When declaring a variable without giving it a value (`let myNumber;`), JavaScript creates the variable without a value, and so its value is **`undefined`**. Even when declaring a variable and assigning it a value at the same time (`let favoriteNumber = 42;`), JavaScript actually creates the variable initially without a value, temporarily giving it a value of `undefined`, before then assigning it the value to the right of the equals sign. This is kind of a subtle point, so don't worry too much about it at the moment.

Additionally, there are some functions and methods that do not return any value, in which case the return value is actually `undefined`. You'll learn more about this when we learn how to write our own methods and functions.

Data Type Detection

It's important to understand the difference between the number `5` and the string `"5"`. To the computer, they are two entirely different things, as illustrated by this example:

```
> const myNumber = 5;
> const myOtherNumber = 10;
> const myText = "5";
> const myOtherText = "10";
> myNumber + myOtherNumber;
15
> myText + myOtherText;
"510"
```

When we added `10` to `5` we got 15, but when we added `"10"` to `"5"` it concatenated the two strings together.

Likewise, the boolean `true` is not the same as the string `"true"`.

In the example above, somewhat confusingly, the `+` operator works on both numbers and strings — just with different results. However, methods will only work on a specific data type. For example, `48432.78.toFixed(1);` works just fine, but trying to do `"48432.78".toFixed(1);` results in an error because there is no `String.prototype.toFixed()` method that works on a string. In other words, `Number.prototype.toFixed()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed) is a method that exists in JavaScript, but `String.prototype.toFixed()` is not.

Likewise, `"hello".charAt(2);` works, but `314159.charAt(2)` does not. This is because `String.prototype.charAt()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/charAt) is a JavaScript method, and `Number.prototype.charAt()` is not.

We can check the data type of a variable or value with the `typeof` JavaScript operator as follows:

```
> typeof 5;
"number"
> typeof "5";
"string"
> typeof true;
"boolean"
> typeof "true";
"string"
> const greeting = "hello world";
> typeof greeting;
"string"
```

Documentation on the `typeof` Operator

Visit this link for the reference page on the `typeof` operator:

- **typeof (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof)**

Data Type Conversion

Often we'll need to convert data types in order to manipulate them or use them in a way that works for our program. Let's look at a few different examples now.

Changing a String into a Number

As we'll soon learn, input from a web browser will come in as a string and we will need to convert it to a number before working with it.

We can convert a string to a number by passing a string to JavaScript's built-in **`parseInt()`** function (we'll cover more on functions soon):

```
> const inputtedAge = "45";
> inputtedAge;
"45"
> typeof inputtedAge;
"string"
> const myAge = parseInt(inputtedAge);
> myAge;
45
> typeof myAge;
"number"
```

Note that if you try to use `parseInt()` to convert a string not actually containing a number, the result is the number `NaN`.

```
> const name = "Andrea";
> const nonsense = parseInt(name);
> nonsense;
NaN
```

Changing a Number into a String

If you need to go the other way around, you can convert a number to a string by calling the **`Number.prototype.toString()`** method on the number:

```
> const myNumber = 42;
> myNumber;
42
> typeof myNumber;
"number"
> const convertedNumber = myNumber.toString();
> convertedNumber;
"42"
> typeof convertedNumber;
"string"
```

We can also call the `Number.prototype.toString()` on a number directly instead of on a variable that has a number value:

```
// two dots here is correct
> 42..toString();
"42"
> (42).toString();
"42"
```

The two dots in this example are correct! Try it in the browser's DevTools console if you're not sure. The first dot is interpreted by JavaScript as a decimal point. For example, in the number `42.0` there is a decimal point between the 2 and the 0. `42.0.toString()` is the shorter version of `42..toString()`. It's not really important to understand why JavaScript is like this at this stage in learning. You can always look up a method when you are in doubt about how to use it. If you prefer, you can choose to use parentheses like `(42).toString()` for the same results.

Changing a Boolean into a String

You can also convert a boolean to a string by calling the **`Boolean.prototype.toString()`** method on the boolean:

```
> const enrolledInEpicodus = true;
> enrolledInEpicodus;
true
> typeof enrolledInEpicodus;
"boolean"
> const convertedBool = enrolledInEpicodus.toString();
> convertedBool;
"true"
> typeof convertedBool;
"string"
```

We can also call the `Boolean.prototype.toString()` on a boolean directly instead of on a variable that has a boolean value:

```
> false.toString();
"false"
```

MDN Documentation Links

Read more about `parseInt()`, `Number.prototype.toString()`, and `Boolean.prototype.toString()` by following these links:

- **`parseInt()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseInt)**
- **`Number.prototype.toString()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toString)**
- **`Boolean.prototype.toString()` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Boolean/toString)**

Previous (/introduction-to-programming/javascript-and-web-browsers/practice-assignment-comparison-and-equality-operators)
Next (/introduction-to-programming/javascript-and-web-browsers/another-look-at-javascript-objects)

Lesson 20 of 75
Last updated March 24, 2023

disable dark mode

(http://www.epicodus.com)