Data Types and Arithmetic Operators

This lesson will introduce you to data types to represent data and also to basic arithmetic operations you can perform with them.

Introduction

Most programming languages help you create values that symbolize a number, a character in a text, or a longer text. You can also symbolize the concept of true and false values using booleans. You can also create values that symbolize the absence of a value. These values are all called *primitive datatypes*.

The name primitive does not come from a negative place. These datatypes are neither stupid, nor inferior to any other datatypes we use in JavaScript. The name primitive comes from their simplicity. The other datatypes you will learn later are *composite*, and they consist of many primitive datatypes.

Primitive Types

In JavaScript, there are six primitive types:

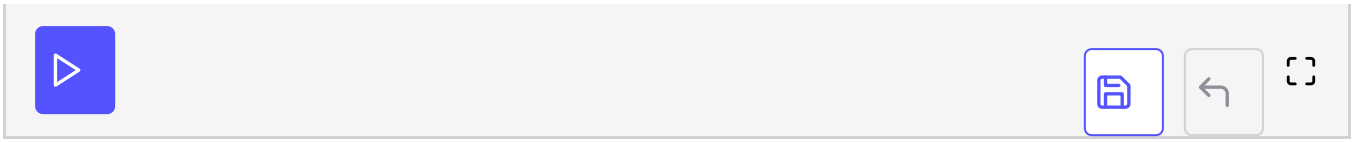
1. boolean (`true` or `false`)
2. number (including integers like `1`, `-2`, and floating point numbers like `1.1`, `2e-3`)
3. string (`‘’` or `“”`, `‘ES6 in Practice’` or `“ES6 in Practice”`)
4. null type (denoted by `null`)
5. undefined (denoted by `undefined`)
6. Symbol (don’t worry about them yet)

Integers and Arithmetic operators

Let’s try some *Javascript expression*.

```
console.log(5+2);
console.log(7%5);
```

```
console.log(5**2);
```



Handling integers is straightforward. You have the four arithmetic operations (`+`, `-`, `*`, `/`) available for addition, subtraction, multiplication, and division respectively.

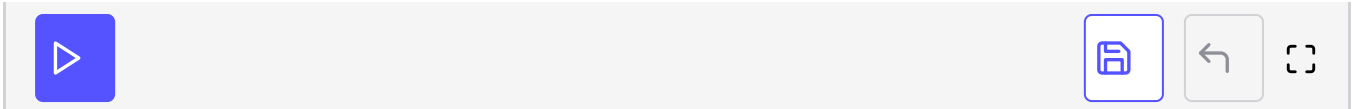
The `%` operator is called modulus. `a % b` returns the remainder of the division `a / b`. In our example, `7 / 5` is `1`, and the remainder is `2`. The value `2` is returned.

The `**` operator is called the exponential operator. `5 ** 2` is five raised to the second power.

Floating Numbers

Let's run the code below.

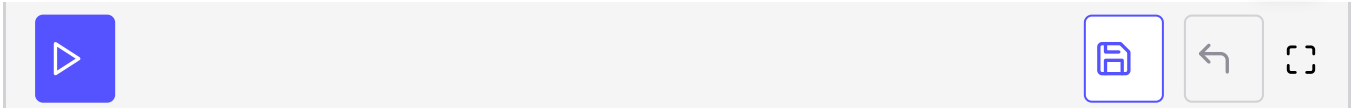
```
console.log(0.1+0.2);
console.log(3.1e-3);
```



Due to the way how numbers are represented, `0.1 + 0.2` is not exactly `0.3`. This is normal and occurs in most programming languages.

`3.1e-3` is the normal form of `0.0031`. Read it like the exact value of 3.1 times ten to the power of minus three. Although the form is similar to `3.1 * (10 ** -3)`, there are subtle differences. `3.1e-3` describes the exact value of `0.0031`. `3.1 * (10 ** -3)` describes a composite expression that needs to be calculated:

```
console.log(3.1 * (10 ** -3))
```

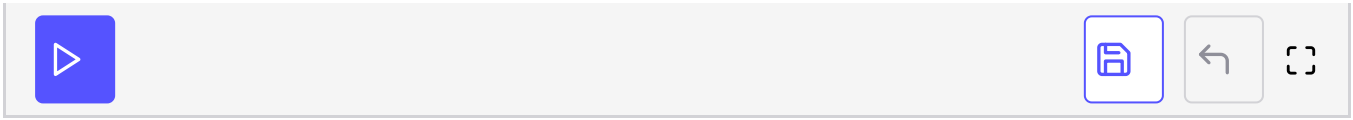


Even floating point arithmetics does not even make it exact.

NaN

The division `0 / 0` or using mismatching types creates a special number called *not a number* or `NaN`.

```
console.log(0 / 0);
console.log('Javascript in Practice'*2);
```

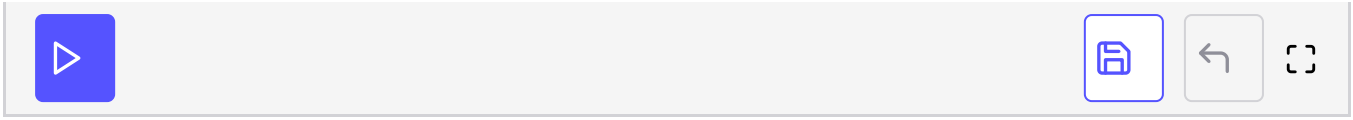


The latter in the code example above is interesting to Python users because, in Python, the result would have been `'Javascript in PracticeJavascript in Practice'`. JavaScript does not work like that.

Infinity Type

There is another interesting type: infinity.

```
console.log(1 / 0);
console.log(Infinity * Infinity);
console.log(-1 / 0);
console.log(1e+308);
console.log(1e+309);
```



JavaScript registers very large numbers as infinity. For instance, ten to the power of 309 is represented as infinity. Division by zero also yields infinity.

Now, this was just an overview of some data types. We will explore more of them in the future.