**Data Types**

JavaScript is a loosely typed dynamic language. In JavaScript there are seven primitive types. Everything else, including functions and arrays, is an object.

JavaScript primitives are as follows:

* Null: **null**
* Undefined: **undefined**
* Number: **1**, **1.5**, **-1e4**, **NaN**
* BigInt: **1n**, **9007199254740993n**
* String: **'str'**, **"str"**, **`str ${var}`**
* Boolean: **true**, **false**
* Symbol: **Symbol('description')**, **Symbol.for('namespace')**

The **null** primitive is typically used to describe the absence of an object, whereas **undefined** is the absence of a defined value. Any variable initialized without a value will be **undefined**. Any expression that attempts access of a non-existent property on object will result in **undefined**. A function without a **return** statement will return **undefined**.

The Number type is double-precision floating-point format. It allows both integers and decimals but has an integer range of -253-1 to 253-1. The BigInt type has no upper/lower limit on integers.

Strings can be created with single or double quotes, or backticks. Strings created with backticks are template strings, these can be multiline and support interpolation whereas normal strings can only be concatenated together using the plus (+) operator.

Symbols can be used as unique identifier keys in objects. The **Symbol.for** method creates/gets a global symbol.

Other than that, absolutely everything else in JavaScript is an object. An object is a set of key value pairs, where values can be any primitive type or an object (including functions, since functions are objects). Object keys are called properties. An object with a key holding a value that is another object allows for nested data structures:

**const obj = { myKey: { thisIs: 'a nested object' } }  
console.log(obj.myKey)**

All JavaScript objects have prototypes. A prototype is an implicit reference to another object that is queried in property lookups. If an object doesn't have a particular property, the object's prototype is checked for that property. If the object's prototype does not have that property, the object's prototype's prototype is checked and so on. This is how inheritance in JavaScript works, JavaScript is a prototypal language. This will be explored in more detail later in this section.