JavaScript Data Types and Dynamic Typing

Introduction to Data Types in JavaScript

In the previous lecture, we discussed values and variables. In every programming language, values can have different types depending on the kind of data we want them to hold. We have already seen strings and numbers, but there are actually more data types. Let us now take a look at all the types available in JavaScript.

Objects and Primitive Data Types

In JavaScript, every value is either an object or a primitive value. A value is only a primitive when it is not an object. We will learn all about objects later, but for now, let us focus on primitive data types.

There are seven primitive data types in JavaScript:

- number
- string
- boolean
- undefined
- null
- symbol
- BigInt

Let us look at them one by one.

Number Data Type

Numbers in JavaScript are always floating point numbers, which means they always have decimals, even if we do not see or define them. For example, the value 23 is exactly like having 23.0. Both are simply of the number data type. In many other programming languages, you will find different data types for integers and decimals, but not in JavaScript. All numbers are simply of the type number.

String Data Type

Strings are simply a sequence of characters and are used for text. Always put them in quotes, whether double or single quotes. Otherwise, JavaScript will confuse them with variable names.

Boolean Data Type

The boolean data type is a logical type that can only take one of the logical values true or false. In other words, a boolean is always either true or false. We use boolean values to make decisions with code.

These are the three most important data types, but there are still four more which might be a bit more confusing.

Undefined and Null

Undefined is the value taken by a variable that is not yet defined. A variable that is not yet defined is simply a variable that we declared but did not assign a value. Basically, undefined means empty value. Null is actually pretty similar because it also means empty value, but it is used in different circumstances.

Symbol and BigInt

Symbol was introduced in ES2015 and defines a value that is unique and cannot be changed. We will talk about this briefly by the end of the course. Starting in ES2020, there is also BigInt, which is for integers that are too large to be represented by the number type.

Dynamic Typing in JavaScript

JavaScript has a feature called dynamic typing. This means that when you create a new variable, you do not have to manually define the data type of the value that it contains. In many other programming languages, you have to do that, but not in JavaScript. JavaScript automatically determines the data type of a value when it is stored into a variable. It is the value that has a type, not the variable. Variables simply store values that have a type.

Another important application of dynamic typing is that later in our code, we can assign a new value with a different data type to the same variable without a problem. For example, a variable can initially be a number and then later a string. This can be very useful but can also be the source of some difficult-to-find bugs.

Comments in JavaScript

In programming, we use comments to document code or to deactivate code without deleting it. For example, you can write a single line comment using // and then describe what the code is doing. JavaScript will completely ignore anything that is a comment.

javascript Code Sample

// This is a single line comment

You can also create multi-line comments using /* and */. Anything between these markers is ignored by JavaScript.

javascript Code Sample

/*

This is a multi-line comment.

It can span multiple lines.

*/

Comments can be used to divide code into different sections and to explain what different pieces of code are doing. They can also be used to comment out code that you do not want to be executed without deleting it.

Examples of Data Types

We have already seen numbers and strings. Here is an example of a value which is a number, and another which is a string.

javascript Code Sample

let age = 23;

let firstName = 'Jonas';

Boolean is a value that can only be true or false. For example, if you write true, that is automatically a boolean value.

javascript Code Sample

let isJavaScriptFun = true;

You can log values to the console to see their type. Strings are printed in white, and boolean values are printed in pink in the console.

javascript Code Sample

console.log(true);

Variables can store booleans as well. Use descriptive variable names and camel case notation.

javascript Code Sample

console.log(isJavaScriptFun);

Remember that JavaScript programs are executed from top to bottom. Variable declarations need to happen before you use them.

The typeof Operator

The **typeof** operator is used to show the type of a value. For example:

javascript Code Sample

console.log(typeof true);

The result of using this operator on a value will be a new value, which is a string with the type of the value.

javascript Code Sample

console.log(typeof isJavaScriptFun);

console.log(typeof 23);

console.log(typeof 'Jonas');

JavaScriptIsFun is a boolean, 23 is a number, and 'Jonas' is a string.

Always use quotes to create a string. If you do not use quotes, JavaScript will see it as a variable and you may get a reference error.

javascript Code Sample

console.log('Jonas');

console.log("Jonas");

If you forget the quotes, JavaScript will look for a variable with that name and throw an error if it does not exist.

Dynamic Typing in Action

Dynamic typing means we can easily change the type of a value held by a variable. For example, we can reassign a variable from a boolean to a string.

javascript Code Sample

isJavaScriptFun = 'YES!';

When reassigning, do not use the **let** keyword again. The first time you declare a variable, use **let**, but when changing its value, simply assign a new value.

javascript Code Sample

```
console.log(typeof isJavaScriptFun);
```

Initially, the type of isJavaScriptFun is boolean, but after reassignment, it becomes a string. This demonstrates dynamic typing.

Undefined Data Type

Undefined is the value taken by a variable that is not yet defined. It means an empty value. It is legal in JavaScript to define a variable without a value.

javascript Code Sample

```
let year;

console.log(year);

console.log(typeof year);
```

Whenever we declare an empty variable, the value of the variable will be undefined, and the type of the variable is also undefined.

We can also reassign this variable to a number.

javascript Code Sample

```
year = 1991;

console.log(typeof year);
```

Now the type of year is number, again showing dynamic typing.

The typeof null Bug

There is a known bug in JavaScript where **typeof null** returns 'object'. This does not make sense, and it is regarded as a bug in JavaScript. However, this bug was never corrected for legacy reasons. It should return null, just as **typeof undefined** returns undefined.

javascript Code Sample

```
console.log(typeof null);
```

Keep this in mind when working with **typeof**, and we will revisit this later in the course to avoid accidental bugs due to this behavior.

That concludes the discussion on data types in JavaScript.

Key Takeaways

- JavaScript has seven primitive data types: number, string, boolean, undefined, null, symbol, and BigInt.
- JavaScript uses dynamic typing, meaning variables can hold values of any type and their type can change at runtime.
- The **typeof** operator is used to check the type of a value, but has a known bug where **typeof null** returns 'object'.
- Comments in JavaScript can be single-line (//) or multi-line (/* ... */), and are useful for documenting and deactivating code.

Data Type	Example
Number	10, 3.14
String	'JS', "Hello"
Boolean	true, false
Undefined	let x;
Null	let x = null;
Symbol	Symbol('id')
BigInt	123n