Code structure and data types

Script tag

You can insert JavaScript program into HTML document using the <script> tags.

You can write inline scripts directly between the tags or you can include external scripts. In order to include external script you would specify the `src` attribute for the script tag.

```
<script src="/path/to/script.js"></script>
```

If the `src` attribute is set, the script content is ignored.

Code Structure

Statement

Code in JavaScript can be ended with semicolons, or without semicolons if the statements are on line break

```
alert("Hello World");
alert("This is hello!");

or

alert("Hello World")
alert("This is hello!")
```

Having semicolons for the end of a statement is good practice because the JavaScript might do wrong assumption of when to actually insert or not insert the semicolon take a look at the example below:

```
alert("Hello");

[1, 2].forEach(alert);

// The two statement above will print out "Hello" then 1 and 2, which is what we expected
```

The two statement above will print out "Hello" then 1 and 2, which is what we expected.

However, if we change up the code to be the following:

```
alert("Hello")

[1, 2].forEach(alert);
```

The code will only print out "Hello", and then it will display an error. This is because JavaScript doesn't insert semicolon before square brackets, therefore the two lines are treated as if they are one line

```
alert("Hello")[1, 2].forEach(alert);
```

Comments

```
// This is a comment

/* This is a multi-line
comment :).
*/
```

Not nesting `/* */` comments just lie in C.

Data Types

A value in scripting language is always a certain type, and there are eight of the basic data types in JavaScript

A variable isn't associated with a type, but rather is associated with a value. The type of a variable can change per use. i.e. storing a string into a variable then later storing a number into it later.

Number

Represents both integer and floating point numbers.

Infinity, -Infinity, and NaN are speical numeric values.

When you are dividing undefined math operations you will get NaN, such as dividing a string with a number.

Math in JavaScript is "safe", meaning that your script will never result in a runtime error, at worse your result that you get will just be *NaN*.

BigInt

In JavaScript there is a maximum size integer limit, you cannot represent integer values larger than (2^53 - 1) to do that you have to use the BigInt data type.

To create a BigInt you append a n to the end of the number

```
const bigInt = 1293129312931293193912391n;
```

String

A string must be surrounded by quotes, and there are three way of quoting your string.

1. Double quotes: "Hello"
2. Single quotes: 'Hello'
3. Backticks: `Hello`

Double and single quotes are basically the same thing, they have no differences between them.

However, with backticks you are allowed to do variable interpolation, meaning you can incorporate variables into the string

```
let name = "John";


console.log(`Hello, ${name}`);
```

Besides incorporating variables, you can also call functions and the return value will be used in place for the string.

Boolean

A boolean only has two values, either `true` or `false`.

Any number besides 0 are also considered to be `true`.

Null value

The special `null` value don't belong to any type. It is basically a reference to a non-existing object, a "null pointer".

"undefined" value

Just like `null` it doesn't belong to any type, it is a type of its own. The meaning of `undefined` is that the value isn't assigned yet.

```
let x;
console.log("x"); // prints out undefined, because we haven't assigned it a value
```

Objects and Symbols

All the data types we have discussed so far are referred to as "primitives" meaning it is only storing one value of the corresponding type. Objects on the other hand you can used to store a collection of data to make up a more complex data type.

Symbol is used to refer to objects.

typeof operator

The typeof operator is used to return the type of the operand in **String**. Used to quickly check the actual type of the variable.

You can call it using `typeof x` or `typeof(x)` works as well.

Revision #9
Created 2022-12-16 22:36:38 UTC by Tamarine
Updated 2022-12-20 02:30:45 UTC by Tamarine