JavaScript Variables and Data Types Review

Working with HTML, CSS, and JavaScript

While HTML and CSS provide website structure, JavaScript brings interactivity to websites by enabling complex functionality, such as handling user input, animating elements, and even building full web applications.

Data Types in JavaScript

Data types help the program understand the kind of data it's working with, whether it's a number, text, or something else.

- **Number**: A number represents both integers and floating-point values. Examples of integers include 7, 19, and 90.
- **Floating point**: A floating point number is a number with a decimal point. Examples include 3.14, 0.5, and 0.0001.
- **String**: A string is a sequence of characters, or text, enclosed in quotes. "I like coding" and 'JavaScript is fun' are examples of strings.
- **Boolean**: A boolean represents one of two possible values: true or false. You can use a boolean to represent a condition, such as isLoggedIn = true.
- **Undefined and Null**: An undefined value is a variable that has been declared but not assigned a value. A null value is an empty value, or a variable that has intentionally been assigned a value of null.
- **Object**: An object is a collection of key-value pairs. The key is the property name, and the value is the property value.

Here, the pet object has three properties or keys: name, age, and type. The values are Fluffy, 3, and dog, respectively.

Example Code

```
let pet = {
  name: "Fluffy",
  age: 3,
  type: "dog"
```

};

- **Symbol**: The Symbol data type is a unique and immutable value that may be used as an identifier for object properties.

In this example below, two symbols are created with the same description, but they are not equal.

Example Code

```
const crypticKey1= Symbol("saltNpepper");

const crypticKey2= Symbol("saltNpepper");

console.log(crypticKey1 === crypticKey2); // false
```

- **BigInt**: When the number is too large for the Number data type, you can use the BigInt data type to represent integers of arbitrary length.

By adding an n to the end of the number, you can create a BigInt.

Example Code

```
const veryBigNumber = 12345678901234567890123456789012345678901234567890n;
```

Variables in JavaScript

- Variables can be declared using the let keyword.

Example Code

```
let cityName;
```

- To assign a value to a variable, you can use the assignment operator =.

Example Code

```
cityName = "New York";
```

- Variables declared using let can be reassigned a new value.

Example Code

```
cityName = "Los Angeles";

console.log(cityName); // Los Angeles
```

- Apart from let, you can also use const to declare a variable. However, a const variable cannot be reassigned a new value.

Example Code

```
const cityName = "New York";

cityName = "Los Angeles"; // TypeError: Assignment to constant variable.
```

- Variables declared using const find uses in declaring constants, that are not allowed to change throughout the code, such as PI or MAX_SIZE.

Variable Naming Conventions

- Variable names should be descriptive and meaningful.
- Variable names should be camelCase like cityName, isLoggedIn, and veryBigNumber.
- Variable names should not start with a number. They must begin with a letter, _, or $.
- Variable names should not contain spaces or special characters, except for _ and $.
- Variable names should not be reserved keywords.
- Variable names are case-sensitive. age and Age are different variables.

Strings and String immutability in JavaScript

- Strings are sequences of characters enclosed in quotes. They can be created using single quotes and double quotes.

Example Code

```
let correctWay = 'This is a string';

let alsoCorrect = "This is also a string";
```

- Strings are immutable in JavaScript. This means that once a string is created, you cannot change the characters in the string. However, you can still reassign strings to a new value.

Example Code

```
let firstName = "John";

firstName = "Jane"; // Reassigning the string to a new value
```

String Concatenation in JavaScript

- Concatenation is the process of joining multiple strings or combining strings with variables that hold text. The + operator is one of the simplest and most frequently used methods to concatenate strings.

Example Code

```
let studentName = "Asad";

let studentAge = 25;

let studentInfo = studentName + " is " + studentAge + " years old.";

console.log(studentInfo); // Asad is 25 years old.
```

- If you need to add or append to an existing string, then you can use the += operator. This is helpful when you want to build upon a string by adding more text to it over time.

Example Code

```
let message = "Welcome to programming, ";

message += "Asad!";

console.log(message); // Welcome to programming, Asad!
```

- Another way you can concatenate strings is to use the concat() method. This method joins two or more strings together.

Example Code

```
let firstName = "John";

let lastName = "Doe";

let fullName = firstName.concat(" ", lastName);

console.log(fullName); // John Doe
```

Logging Messages with console.log()

- The console.log() method is used to log messages to the console. It's a helpful tool for debugging and testing your code.

Example Code

```
console.log("Hello, World!");

// Output: Hello, World!
```

Semicolons in JavaScript

- Semicolons are primarily used to mark the end of a statement. This helps the JavaScript engine understand the separation of individual instructions, which is crucial for correct execution.

Example Code

```
let message = "Hello, World!"; // first statement ends here

let number = 42; // second statement starts here
```

- Semicolons help prevent ambiguities in code execution and ensure that statements are correctly terminated.

Comments in JavaScript

- Any line of code that is commented out is ignored by the JavaScript engine. Comments are used to explain code, make notes, or temporarily disable code.
- Single-line comments are created using //.

Example Code

```
// This is a single-line comment and will be ignored by the JavaScript engine
```

- Multi-line comments are created using /* to start the comment and */ to end the comment.

Example Code

```
/*

This is a multi-line comment.

It can span multiple lines.

*/
```

JavaScript as a Dynamically Typed Language

- JavaScript is a dynamically typed language, which means that you don't have to specify the data type of a variable when you declare it. The JavaScript engine automatically determines the data type based on the value assigned to the variable.

Example Code

```
let error = 404; // JavaScript treats error as a number
```

error = "Not Found"; // JavaScript now treats error as a string

- Other languages, like Java, that are not dynamically typed would result in an error:

Example Code

int error = 404; // value must always be an integer

error = "Not Found"; // This would cause an error in Java

Using the typeof Operator

- The typeof operator is used to check the data type of a variable. It returns a string indicating the type of the variable.

Example Code

let age = 25;

console.log(typeof age); // "number"

let isLoggedIn = true;

console.log(typeof isLoggedIn); // "boolean"

- However, there's a well-known quirk in JavaScript when it comes to null.
  The typeof operator returns "object" for null values.

Example Code

let user = null;

console.log(typeof user); // "object"