Lesson:

Type Conversion & Types

Topics

- Introduction to Type Conversion
- Need of type conversion
- Types of type conversion (Implicit and Explicit type with example)

Introduction to Type Conversion

In JavaScript, Type conversion, also known as "Type Casting" or "Coercion", is a fundamental concept in JavaScript. It involves changing the data type of a value from one type to another. JavaScript is a loosely typed language, meaning that variables can hold values of any data type and type conversion can happen implicitly or explicitly.
Understanding the type conversion is crucial for writing robust and flexible Javascript code. It enables us to work with different types of data, perform operations, and handle various scenarios effectively.

String conversion - string conversion happens when we need the string form of a value, string conversion can be done using the String constructor as shown in the example below.

Example -

```
Unset
const present = true
console.log(typeof present) // boolean

const present1 = String(present)
console.log(typeof present1) // string

// Note- typeof is a operators use to find data type of a variable
```

Numeric Conversion - numeric conversion in mathematical functions and expressions happens automatically. It can be also done manually by using the number constructor as shown in the example.

Example

```
Unset
const num = "5"
const result = Number (num)
console.log(typeof result) // number


console.log("25"/"5") // output - 5
// both are string but treated as number because of the mathematical operation
```

Boolean Conversion - similar to the string and numeric conversion the Boolean conversion can be done by using the Boolean constructor

Example

```
Unset
const num1= 0
const num2= 1

console.log(num1) // 0
console.log(num2) // 1

console.log(Boolean(num1)) // false
console.log(Boolean(num2)) // true
```

Some more examples of Type conversion

```
JavaScript
// Convert Strings to Numbers
const numb = Number("3.14")
console.log(typeof numb) // number

// Convert Numbers to String
const str = String(3.14)
console.log(typeof str) // string
 // Convert Dates to Numbers
 const date = Number (new Date())
 console.log(typeof date) // number

 // Convert Numbers to Dates
 let timestamp = 1640323200000;
 let date1 = new Date(timestamp); // Converts to a Date object
 console.log(typeof date1); // date object
// Convert Boolean to Numbers
let bool = true
let num = +bool
console.log(num) // 1
console.log(typeof num) // number

// Convert Numbers to Boolean
let bool = Boolean(21) // coverts to a boolean with value true
console.log(typeof book) // false
```

Need of type conversion

The Type conversion is often necessary in JavaScript because it is a weakly typed language. This means that variables do not have a specific data type assigned to them at compile time. Instead, the data type of a variable is determined by the value assigned to it.

The weak typing can make JavaScript code more flexible, but it can lead to unexpected results if you are not being careful.
For example, if you try to add a number to a string without explicitly converting the string to a number, JavaScript will automatically convert the string to a number. This can lead to errors if the string contains non-numeric characters.

Some of the other reasons why there is a need for Type Conversion can be -

- Compatibility and interoperability - Type conversion allows you to work with values of different data types, promoting compatibility and interoperability in your code

Example -

```
Unset
const num = '10'; // string
const result = num + 5;
console.log(result); // output - 105
```

- User Input and Interaction - user input forms or other source data often comes in the string, Type conversion allows you to convert these string input to appropriate data types for processing and manipulation.

Example -

```
Unset
const input = '11' //Assuming the user enters '11'.
const num = Number(input);  // Convert the input to a number.
console.log(num);  // Output: 11 (number)
```

- Conditional Statement and Comparison - different data types may behave unexpectedly when compared. Type conversions ensure consistency and predictable behavior during comparisons

Example -

```
Unset
const num  = 10;
const num1 = '10';
console.log(num == num1);//Output:true(implicit conversion of strNum to a
number for comparison)
console.log(num === num1);//Output:false(strict equality without type
// conversion)
```

- Mathematical Operation - for mathematical operations, it's essential to have the correct data types to obtain accurate results.

Example -

```
Unset
const input= '5';
const number = Number(input);

const result = number * 2;  // Numeric multiplication requires the number to be
of a numeric data type.
console.log(result);  // Output: 10 (number)
```

Types of type Conversion

In JavaScript, The Type conversion can be of two types i.e. Explicit and Implicit.

Implicit type conversion -
The implicit type conversion is the default behavior of JavaScript, also known as coercion. JavaScript automatically converts a value to another type in order to perform an operation.
For example, if you add a number to a string, Javascript will convert the string to a number before performing the addition.

```
Unset
// Implicit type conversion
const number = 5;
const string = "10";

const result = number + string; // "510"
```

Explicit type conversion

In JavaScript, The Explicit type conversion is when you manually convert a value to another type using a built-javascript function.
For Example - you can use the **Number()** function to convert a string to a number or use the String function to convert a number to a string.

```
Unset
// Explicit type conversion
const numberFromString = Number(string); // 10
console.log(numberFromString)
```