Module 3: Introduction to JavaScript

Casting Types

Casting refers to the process of converting a value from one data type to another. This is sometimes necessary when you want to perform operations or comparisons involving values of different types.

There are several ways to perform type casting to change one data type to another in JavaScript:

- Implicit type casting
- Explicit type casting
    - Using functions
    - Using a type-specific operator
    - Using the Boolean() constructor

Implicit Type Casting

JavaScript performs implicit type conversion in certain situations, that is, it automatically performs type casting. For example, when you use the + operator with different data types, JavaScript will attempt to convert them to a common type and perform the operation.

```
var num = 5;         // Number
var str = "10";     // String
var result = num + str; // JavaScript converts num to a string and
performs concatenation
console.log(result);
// Output: "510"
```

Explicit Type Casting

With this kind of type casting, you need to explicitly perform the type casting. You can perform explicit type casting in many ways. Some of these ways are explained below.

Using functions:

```
parseInt(): Converts a string to an integer.
var str = "10";
var num = parseInt(str);
console.log(num); // Output: 10
```

parseFloat(): converts a string to a floating-point number.

```
var str = "3.14";
var num = parseFloat(str);
console.log(num); // Output: 3.14
```

Number(): converts a value to a number.

```
var str = "42";
var num = Number(str);
console.log(num); // Output: 42
```

String(): converts a value to a string.

```
var num = 42;
var str = String(num);
console.log(str); // Output: "42"
```

Using a type-specific operator:

parseInt() and parseFloat() can also be used as type casting functions, but they are typically used for converting strings to numbers as discussed above.

Using a Boolean() constructor:

This converts a value to a Boolean. This is commonly used to check the "truthiness" or "falsiness" of a value. Any value that is defined, not null, and not zero can be converted to false and vice versa.

```
var value = "Hello";
var boolValue = Boolean(value);
console.log(boolValue); // Output: true
```

Type of Operator

You can use the type of operator to check the data type of a value. This doesn't perform any casting but can help you determine the variable type.

```
var num = 42;
var str = "42";
console.log(typeof num); // Output: "number"
console.log(typeof str); // Output: "string"
```