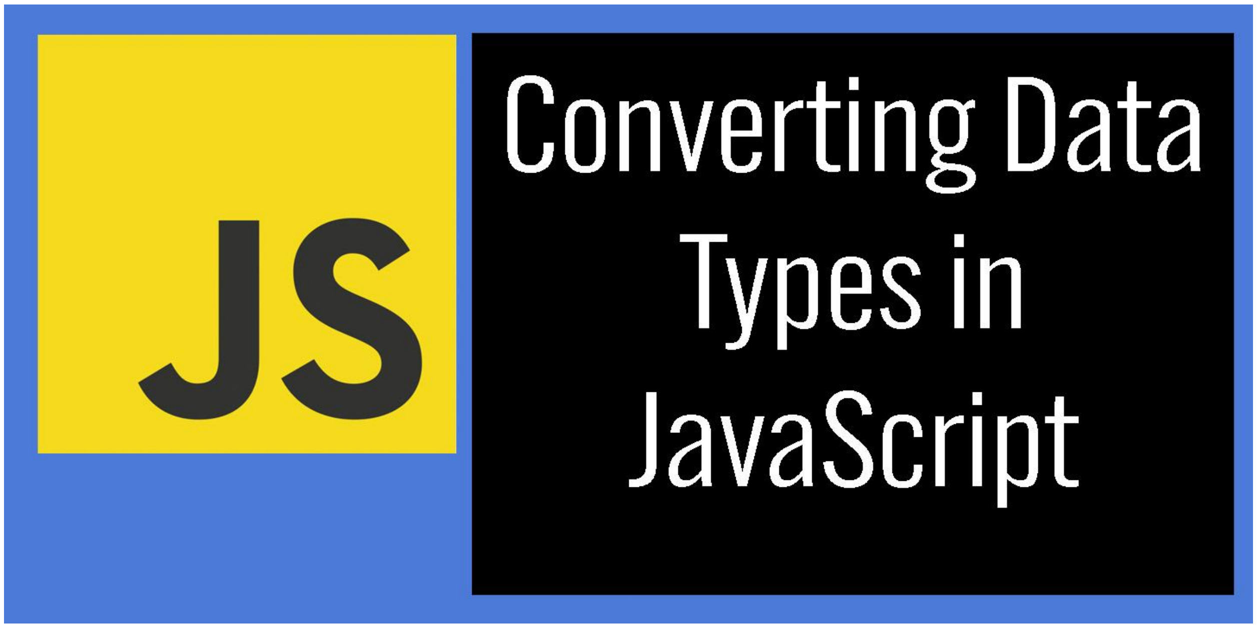
Converting Data Types in JavaScript



1. Implicit Type Conversion: **2**
2. Explicit Type Conversion: **3**
3. Truthy and Falsy Values: **4**
4. parseInt and parseFloat: **4**
5. NaN (Not-a-Number): **5**
6. Type Coercion in Operations: **5**
10 coding exercises converting data types in JavaScript **6**
JavaScript Code Example 1: Convert to Number 6
JavaScript Code Example 2: Convert to String 6
JavaScript Code Example 3: Convert to Boolean 7
JavaScript Code Example 4: Parse Integer from String 7
JavaScript Code Example 5: Parse Float from String 8
JavaScript Code Example 6: Convert to Boolean with Truthy/Falsy Check 8
JavaScript Code Example 7: Convert to Number with Type Coercion 9
JavaScript Code Example 8: Convert to Number with Type Coercion 9

JavaScript Code Example 9: Handling NaN 10
JavaScript Code Example 10: Type Conversion in a Mixed Expression 11
Coding Exercise with source code 11
Exercise 1: Convert to Number and Add 11
Exercise 2: Convert to String and Concatenate 12
Exercise 3: Convert to Boolean and Check Falsy 13
Exercise 4: Parse Integer from String and Multiply 14
Exercise 5: Convert to String and Add Leading Zeros 15

In JavaScript, data type conversion, also known as type coercion, is the process of converting a value from one data type to another. This is a fundamental concept because JavaScript is dynamically typed, meaning variables can change their data type during execution. Understanding how to convert between data types is essential.

Let's delve into a detailed explanation of data type conversion in JavaScript, including coding examples.

1. Implicit Type Conversion:

JavaScript performs implicit type conversion automatically when an operation involves different data types. For example, when you concatenate a string and a number, JavaScript converts the number to a string.

```
let num = 42;
let str = "The answer is " + num; // Implicitly
converts num to a string
```

```
console.log(str); // "The answer is 42"
```

2. Explicit Type Conversion:

You can explicitly convert data types using functions like Number(), String(), and Boolean().

To Number:

```
let strNum = "123";
let num = Number(strNum); // Explicitly convert to a
number
console.log(num); // 123
```

To String:

```
let num = 456;
let str = String(num); // Explicitly convert to a
string
console.log(str); // "456"
```

To Boolean:

```
let num = 0;
let bool = Boolean(num); // Explicitly convert to a
boolean
console.log(bool); // false
```

3. Truthy and Falsy Values:

JavaScript has a concept of truthy and falsy values. Empty strings, 0, null, undefined, NaN, and false are considered falsy, while all other values are truthy.

```
let value = ""; // Falsy
let isTruthy = Boolean(value); // Convert to boolean
console.log(isTruthy); // false
```

4. parseInt and parseFloat:

Use parseInt() and parseFloat() to convert strings to integers or floating-point numbers:

```
let strInt = "42";
let int = parseInt(strInt); // Convert to integer
console.log(int); // 42

let strFloat = "3.14";
let float = parseFloat(strFloat); // Convert to
floating-point number
console.log(float); // 3.14
```

5. NaN (Not-a-Number):

If a conversion is not possible, JavaScript returns NaN. You can use isNaN() to check if a value is NaN.

```
let invalidNum = "abc";
let converted = Number(invalidNum);
console.log(converted); // NaN
console.log(isNaN(converted)); // true
```

6. Type Coercion in Operations:

JavaScript performs type coercion in operations, which can lead to unexpected results. Be aware of how types are coerced in expressions:

```
let addition = "5" + 2; // Implicit conversion to
string and concatenation
console.log(addition); // "52"

let subtraction = "5" - 2; // Implicit conversion to
number and subtraction
console.log(subtraction); // 3
```

Understanding data type conversion in JavaScript is crucial for writing robust and predictable code. It's essential to know when and how to perform explicit type conversions to ensure that your program behaves as expected, especially when dealing with user input and data manipulation.

10 coding exercises converting data types in JavaScript

JavaScript Code Example 1: Convert to Number

Convert a string representing a number to an actual number.

```
// Step 1: Declare a string with a number
let strNum = "123";

// Step 2: Convert to a number
let num = Number(strNum);

// Step 3: Log the result
console.log(num); // 123
```

JavaScript Code Example 2: Convert to String

Convert a number to a string.

```
// Step 1: Declare a number
let num = 456;

// Step 2: Convert to a string
let str = String(num);
```

```
// Step 3: Log the result
console.log(str); // "456"
```

JavaScript Code Example 3: Convert to Boolean

Convert a value to a boolean.

```
// Step 1: Declare a value
let value = 0;

// Step 2: Convert to a boolean
let bool = Boolean(value);

// Step 3: Log the result
console.log(bool); // false
```

JavaScript Code Example 4: Parse Integer from String

Parse an integer from a string.

```
// Step 1: Declare a string with an integer
let strInt = "42";

// Step 2: Parse to an integer
```

```
let int = parseInt(strInt);

// Step 3: Log the result
console.log(int); // 42
```

JavaScript Code Example 5: Parse Float from String

Parse a floating-point number from a string.

```
// Step 1: Declare a string with a floating-point
number
let strFloat = "3.14";

// Step 2: Parse to a floating-point number
let float = parseFloat(strFloat);

// Step 3: Log the result
console.log(float); // 3.14
```

JavaScript Code Example 6: Convert to Boolean with Truthy/Falsy Check

Convert a value to a boolean and check its truthiness.

```
// Step 1: Declare a value
let value = "";
```

```
// Step 2: Convert to a boolean and check truthiness
let isTruthy = Boolean(value);

// Step 3: Log the result
console.log(isTruthy); // false
```

JavaScript Code Example 7: Convert to Number with Type Coercion

Observe how JavaScript performs type coercion in an addition operation.

```
// Step 1: Declare a string and a number
let str = "5";
let num = 2;

// Step 2: Perform addition (implicit conversion)
let addition = str + num;

// Step 3: Log the result
console.log(addition); // "52"
```

JavaScript Code Example 8: Convert to Number with Type Coercion

Observe how JavaScript performs type coercion in a subtraction operation.

```
// Step 1: Declare a string and a number
let str = "5";
let num = 2;

// Step 2: Perform subtraction (implicit conversion)
let subtraction = str - num;

// Step 3: Log the result
console.log(subtraction); // 3
```

JavaScript Code Example 9: Handling NaN

Convert an invalid string to a number and check for NaN.

```
// Step 1: Declare an invalid string
let invalidNum = "abc";

// Step 2: Convert to a number (results in NaN)
let converted = Number(invalidNum);

// Step 3: Check if it's NaN
console.log(converted); // NaN
console.log(isNaN(converted)); // true
```

JavaScript Code Example 10: Type Conversion in a Mixed Expression

Create an expression involving different data types and observe how JavaScript performs type coercion.

```
// Step 1: Declare a string and a number
let str = "5";
let num = 2;

// Step 2: Create an expression involving different
data types
let result = str + num - "1";

// Step 3: Log the result
console.log(result); // "51" (coercion: "52" - "1" =>
51)
```

Coding Exercise with source code

Exercise 1: Convert to Number and Add

Description: Create a function addNumbers that takes two arguments, a (a string representing a number) and b (a number). Convert a to a number, add it to b, and return the result.

Code:

```
function addNumbers(a, b) {
  // Convert 'a' to a number
  let numA = Number(a);

  // Add 'numA' to 'b'
  let result = numA + b;

  return result;
}

// Example usage:
let sum = addNumbers("42", 8);
console.log(sum); // Should log 50
```

Exercise 2: Convert to String and Concatenate

Description: Create a function concatenateStrings that takes two arguments, a (a number) and b (a string). Convert a to a string, concatenate it with b, and return the result.

Code:

```
function concatenateStrings(a, b) {
  // Convert 'a' to a string
  let strA = String(a);
  // Concatenate 'strA' with 'b'
  let result = strA + b;

  return result;
}

// Example usage:
let combined = concatenateStrings(42, " is the
answer");
console.log(combined); // Should log "42 is the answer"
```

Exercise 3: Convert to Boolean and Check Falsy

Description: Create a function checkFalsy that takes a value as an argument. Convert the value to a boolean and return true if the converted value is falsy; otherwise, return false.

Code:

```
function checkFalsy(value) {
  // Convert 'value' to a boolean
  let boolValue = Boolean(value);

  // Check if 'boolValue' is falsy
  return !boolValue;
}

// Example usage:
let isFalsy = checkFalsy(""); // Falsy value
console.log(isFalsy); // Should log true
```

Exercise 4: Parse Integer from String and Multiply

Description: Create a function multiplyByTwo that takes a string str containing an integer. Parse the integer from str, multiply it by 2, and return the result as a number.

Code:

```
function multiplyByTwo(str) {
  // Parse the integer from 'str' and multiply by 2
  let num = parseInt(str) * 2;

  return num;
}

// Example usage:
let result = multiplyByTwo("7"); // Should parse "7" to
7 and return 14
console.log(result);
```

Exercise 5: Convert to String and Add Leading Zeros

Description: Create a function addLeadingZeros that takes two arguments, number (a number) and totalDigits (a number representing the total number of digits required). Convert number to a string and add leading zeros to make it the specified number of digits. Return the resulting string.

Code:

```
function addLeadingZeros(number, totalDigits) {
  // Convert 'number' to a string
  let strNumber = String(number);

  // Calculate the number of leading zeros required
  let zerosToAdd = totalDigits - strNumber.length;

  // Add leading zeros
  let result = "0".repeat(zerosToAdd) + strNumber;

  return result;
}

// Example usage:
```

```
let formattedNumber = addLeadingZeros(42, 5); // Should
return "00042"
console.log(formattedNumber);
```

These exercises provide practical scenarios for converting data types in JavaScript and demonstrate how to use conversion functions to manipulate and work with different data types effectively.