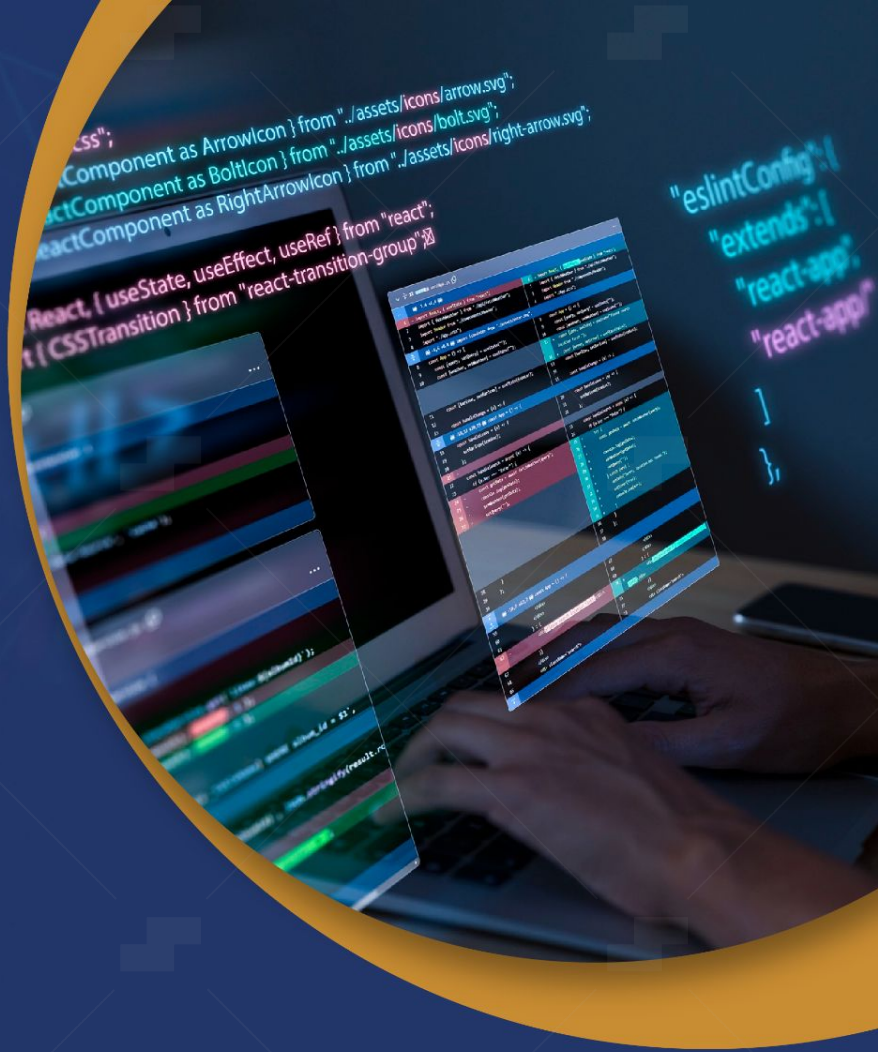
SUNSTONE

JavaScript

Lecture 3

Types, Values, and Variables

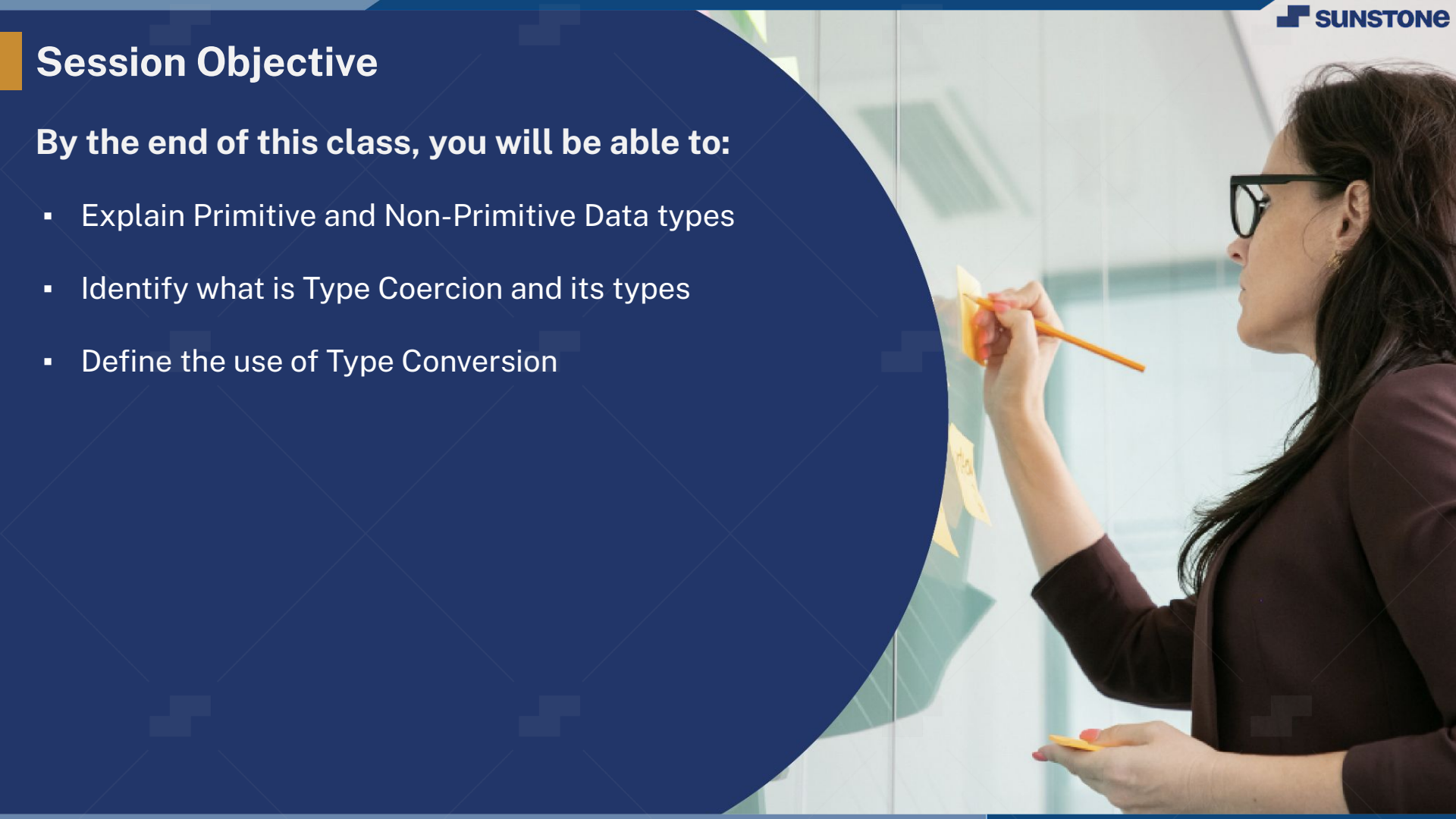


1. The use of Character Sets and Unicode in JavaScript
2. How are Tokens used in JavaScript?
3. List the usage of Variables in JavaScript

Session Objective

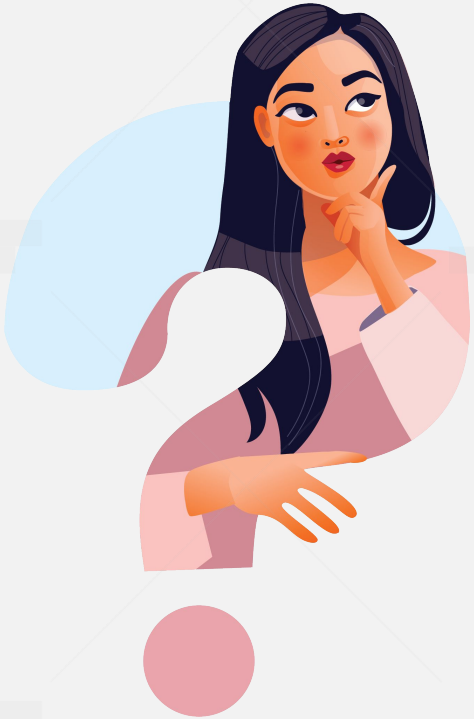
By the end of this class, you will be able to:

- Explain Primitive and Non-Primitive Data types
- Identify what is Type Coercion and its types
- Define the use of Type Conversion



Checkpoints

- Explain Primitive and Non-Primitive Data types
- Identify what is Type Coercion and its types
- Define the use of Type Conversion



What are Data Types?

Character Set

Data Types refer to the categorization and classification of data based on their characteristics, such as the values they can hold and the operations that can be performed on them. Data types determine the type of data that can be stored in a variable or used as a parameter for a function.

There are two types of data types in JavaScript

1. *Primitive Data type*
2. *Non- Primitive Data type*

Primitive Data Type

- Primitive data types are fundamental data types that are built-in to a programming language and represent basic values. They are usually simple, atomic values and have a fixed size.
- Primitive data types are immutable, meaning their values cannot be changed once they are assigned. Operations on primitive types typically involve creating new values.

Types of Primitive Data Type

Number

Represents numeric values, including integers and floating-point numbers.

For example:

```
let age = 25;
```

String

Represents a sequence of characters.

For example:

```
let name = "John";
```

Types of Primitive Data Type

Boolean

Represents a logical value of either true or false.

For example:

```
let isActive = true;
```

Null

Represents the intentional absence of any object value.

For example:

```
let person = null;
```

Types of Primitive Data Type

Undefined

Represents a variable that has been declared but not assigned a value.

For example:

```
let city;
```

Symbol

Represents a unique identifier.

For example:

```
let id = Symbol();
```

Non-Primitive (Reference) Data Type

- Non-primitive or reference data types are more complex data types that are composed of multiple values or have behavior associated with them.
- They are usually represented as objects and can be of varying sizes.
- Non-primitive data types are mutable, meaning their values can be changed. Operations on non-primitive types involve manipulating the object directly or accessing its properties and methods through references. It is also known as reference or object data type.

Types of Non-Primitive Data Type

Object

Represents a collection of key-value pairs or properties. Objects can be created using the object literal syntax ({ }) or the new keyword.

Array

Represents an ordered collection of values. Arrays are a type of object but with additional behavior and methods specifically for working with ordered collections.

Types of Non-Primitive Data Type

Function

Represents a reusable block of code that can be invoked by calling it.

Date

Represents a specific date and time.

Types of Non-Primitive Data Type

RegExp

Represents a regular expression, used for pattern matching in strings.

Error

Error data type is a built-in object type used to represent and handle errors and exceptions in the code. It provides a standardized way to communicate and handle exceptional situations during program execution.

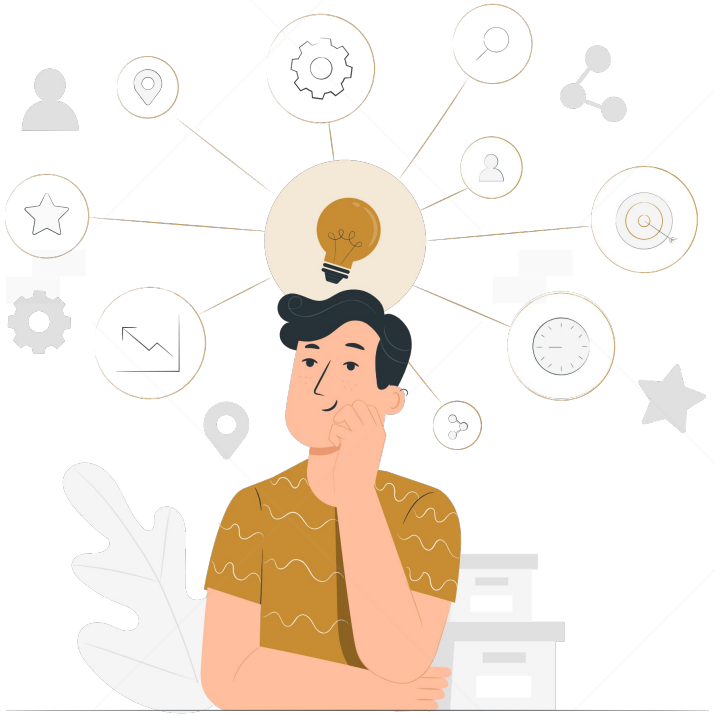
SUNSTONE

Checkpoints

- **Explain Primitive and Non-Primitive Data types**
- Identify what is Type Coercion and its types
- Define the use of Type Conversion

What is Type Coercion?

- JavaScript, type coercion refers to the automatic conversion of one data type to another by the JavaScript engine during certain operations. Type coercion can occur when you use operators or functions that expect operands or arguments of a specific type.
- JavaScript has two types of type coercion:
    1. Explicit
    2. Implicit



Type Coercion: Explicit Type Coercion

- Explicit type coercion occurs when you intentionally convert a value from one type to another using built-in functions or operators. The most commonly used methods for explicit type coercion are:

- **String(value):** Converts a value to a string.
- **Number(value):** Converts a value to a number.
- **Boolean(value):** Converts a value to a boolean.

Type Coercion: Implicit Type Coercion

- Implicit type coercion occurs when the JavaScript engine automatically converts values of one type to another without explicit instruction from the developer. It typically happens when you use operators between values of different types.
- Some common examples of implicit type coercion include:
    - **Addition operator (+)**
    - **Comparison operators (== and !=)**
    - **Logical operators (&& and ||)**

Type Coercion: Implicit Type Coercion

- Some common examples of implicit type coercion include:
    - **Addition operator (+):**

```
var num = 10;
var str = "5";
var result = num + str;
//Implicitly converts num to a string and performs string concatenation
console.log(result);
// Output: "105"
```

Type Coercion: Implicit Type Coercion

- Some common examples of implicit type coercion include:
    - **Comparison operators (== and !=):**

```
var num = 10;
var str = "10";
console.log(num == str);
// Implicitly converts str to a number and performs the comparison
// Output: true
```

Type Coercion: Implicit Type Coercion

- Some common examples of implicit type coercion include:
  - **Logical operators (&& and ||):**

```
var num = 10;
var str = "Hello";
console.log(num && str);
// Implicitly converts num to a boolean and performs logical AND operation
// Output: "Hello"
```

Time for Knowledge Check !!

Quiz

Which method can be used for explicit type coercion to convert a value to a string?

Choose the Correct Answer

a. string()

b. convertToString()

c. String(value)

d. value.toString()

Quiz

Which method can be used for explicit type coercion to convert a value to a string?

Choose the Correct Answer

a. string()

b. convertToString()

c. **String(value)**

d. value.toString()

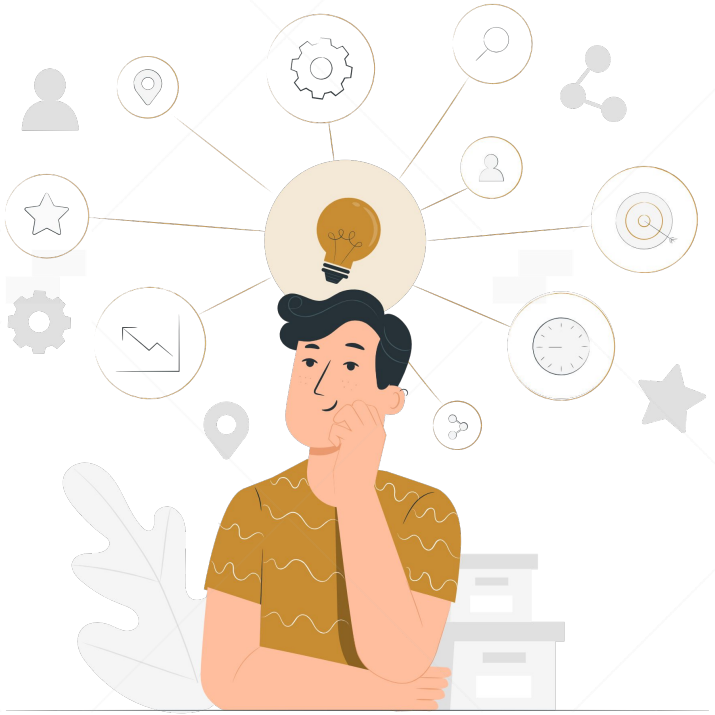
Checkpoints

- Explain Primitive and Non-Primitive Data types
- Identify what is Type Coercion and its types
- Define the use of Type Conversion

What is Type Conversion?

Type Conversion

- In JavaScript, type conversion refers to the process of converting a value from one data type to another. It allows you to change the representation of a value, enabling you to perform operations or manipulations that require a specific data type.



Type Conversion

- JavaScript provides built-in functions and operators that facilitate type conversion. These methods allow you to explicitly convert values from one type to another, ensuring the desired behavior and compatibility in your code.
- Type conversion can involve converting values between primitive data types, such as converting a string to a number or a boolean to a string. It can also involve converting between complex data types, such as converting an array to a string or an object to a number.

Type Conversion

Here are a few common scenarios where type conversion is useful:

String to Number Conversion

Number to String Conversion

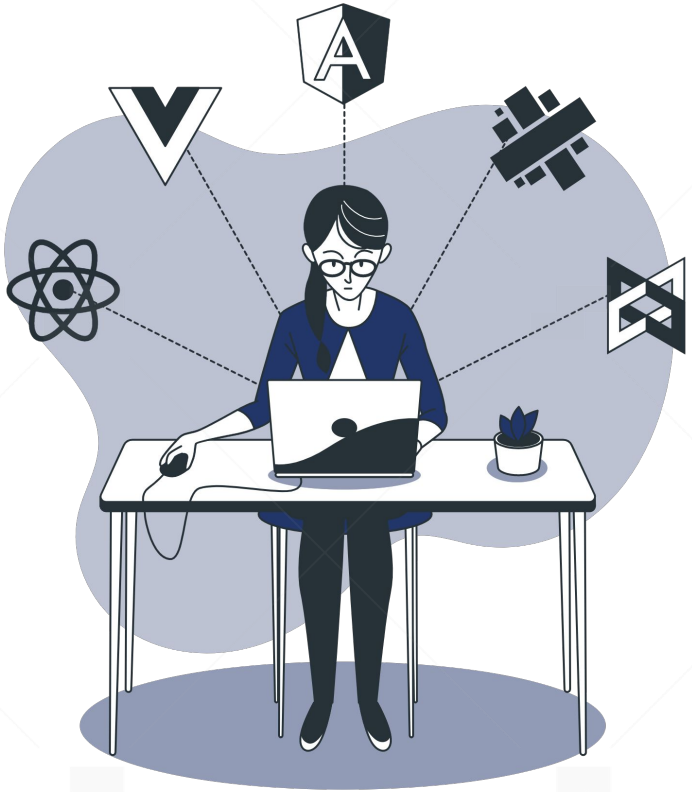
Boolean to String Conversion

String to Boolean Conversion

Type Conversion: Examples

String to Number Conversion:

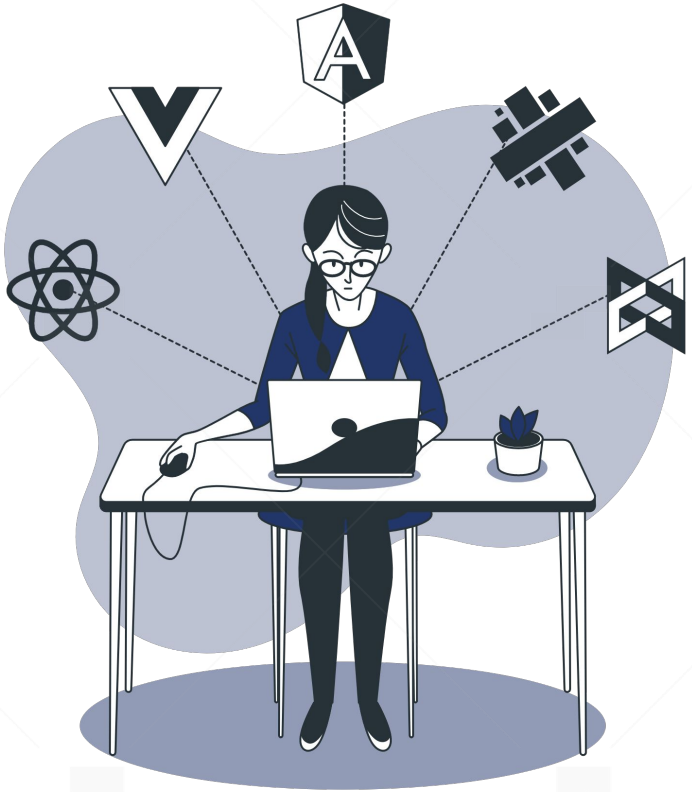
```
var str = "10";

var num1 = Number(str);

var num2 = parseInt(str);

console.log(typeof num1);

// Output: "number"

console.log(typeof num2);

// Output: "number"
```



Type Conversion: Examples

Number to String Conversion:

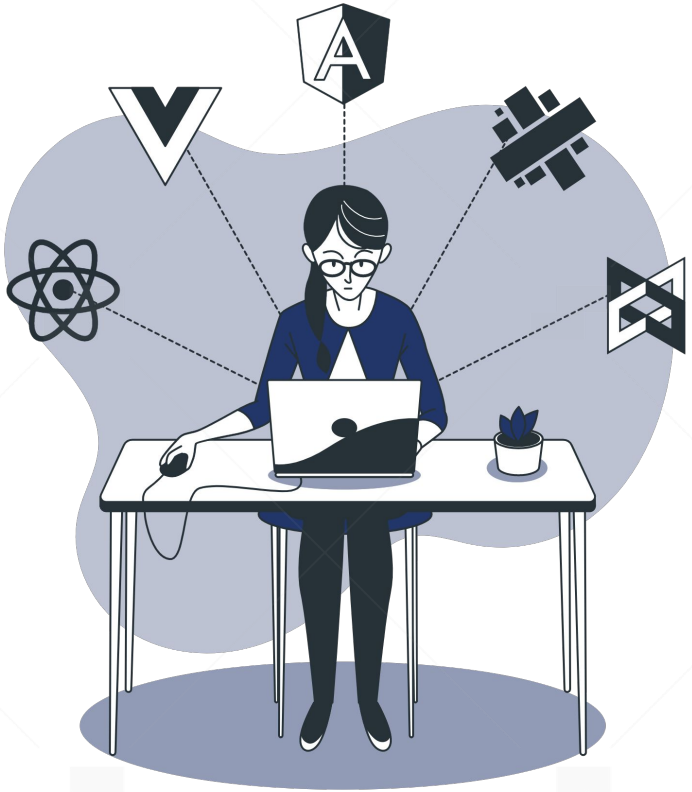
```
var num = 10;

var str1 = String(num);

var str2 = num + "";

console.log(typeof str1);

// Output: "string"

console.log(typeof str2);

// Output: "string"
```



Type Conversion: Examples

Boolean to String Conversion:

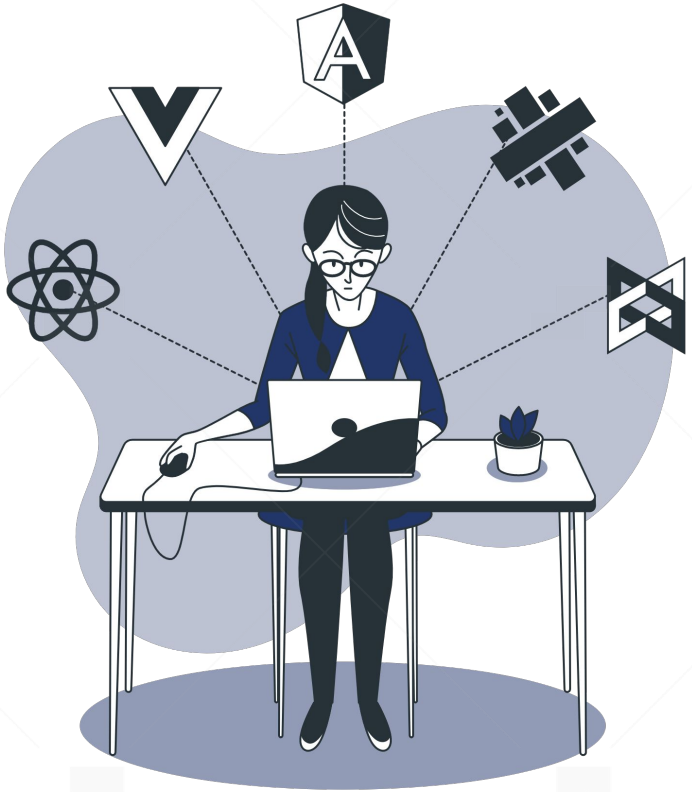
```
var bool = true;

var str1 = String(bool);

var str2 = bool + "";

console.log(typeof str1);

// Output: "string"

console.log(typeof str2);

// Output: "string"
```



Type Conversion: Examples

String to Boolean Conversion:

```
var str1 = "true";
var str2 = "";
var bool1 = Boolean(str1);
var bool2 = Boolean(str2);
console.log(bool1);
// Output: true
console.log(bool2);
// Output: false
```



SUNSTONE

DID YOU KNOW?

Type conversion plays an important role in JavaScript because of its dynamically-typed nature. It allows flexibility in manipulating and working with different types of data, adapting values to meet the requirements of specific operations or functions.

Time for Knowledge Check !!

Quiz

What will be the result of the following code snippet?

```
let str = ""false"";
let bool = Boolean(str);
console.log(bool);
```

Choose the Correct Answer

a. FALSE

b. TRUE

c. "false"

d. null

Quiz

What will be the result of the following code snippet?

```
let str = ""false"";
let bool = Boolean(str);
console.log(bool);
```

Choose the Correct Answer

a. **FALSE**

b. TRUE

c. “false”

d. null

ACTIVITY

Let's do a quick activity to understand type conversion in a better way

Instruct the students to form groups of 2-4 or as needed.

Instruct each group to convert a number to a string on the Console Window in Visual Studio or any online JS compiler.

Checkpoints

- Explain Primitive and Non-Primitive Data types
- Identify what is Type Coercion and its types
- Define the use of Type Conversion

Thank you!