edureka!
a Veranda Enterprise

Full Stack Web Development Program

edureka!
a Veranda Enterprise

Day 18 - Data Types and Operators in JavaScript

Titles - I

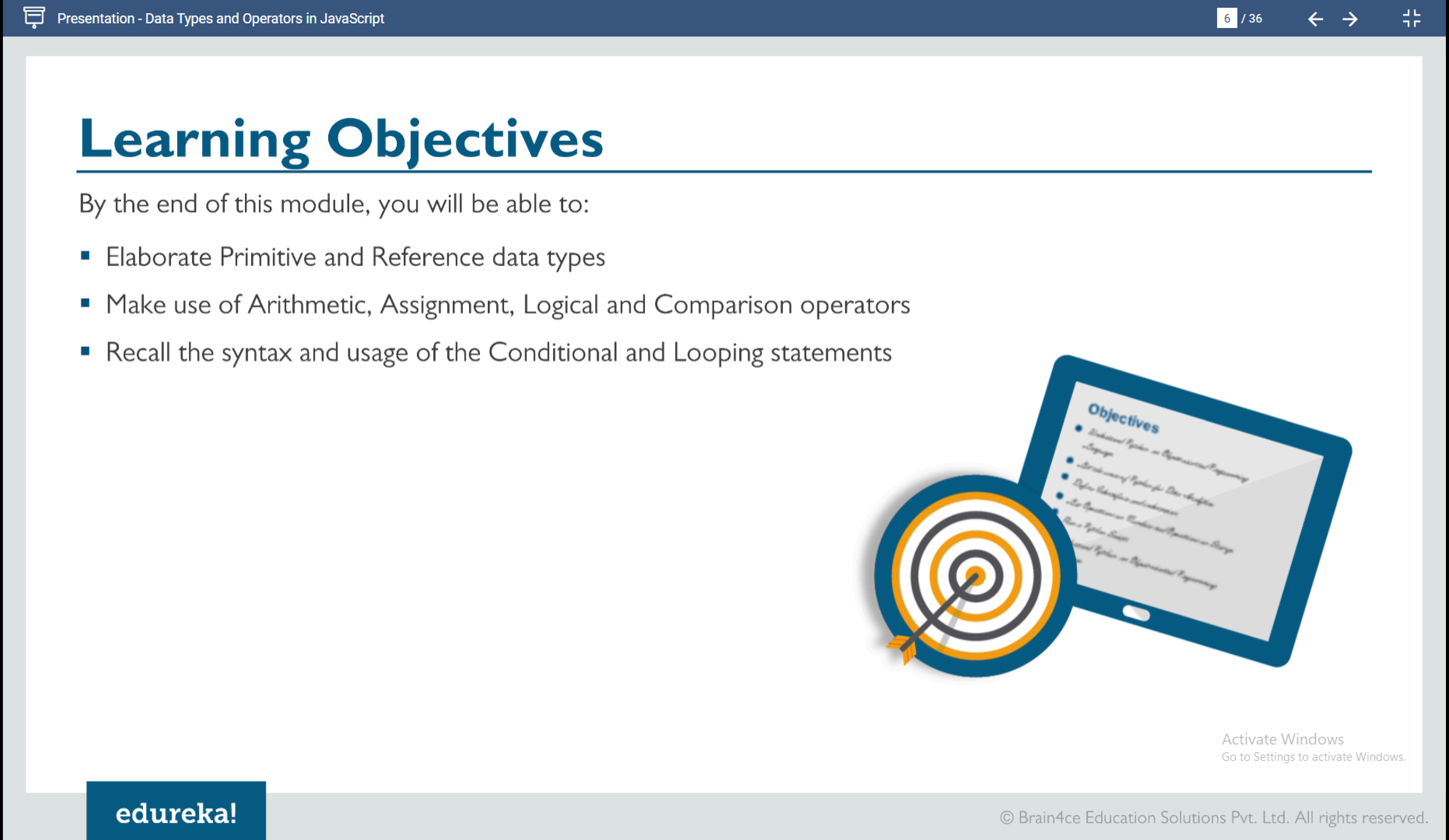
- String
- Number
- Boolean
- Undefined
- Null
- BigInt
- Symbol
- Arrays
- Objects
- Function
- Operators

Titles - II

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- Logical Operators
- Conditional Statements
- If Statement
- If-else Statement
- else if Statement
- Switch Statement
- Looping Statements
- for Loop

Titles - III

- while Loop
- do..while Loop
- for..in loop
- for..of Loop



Learning Objectives

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

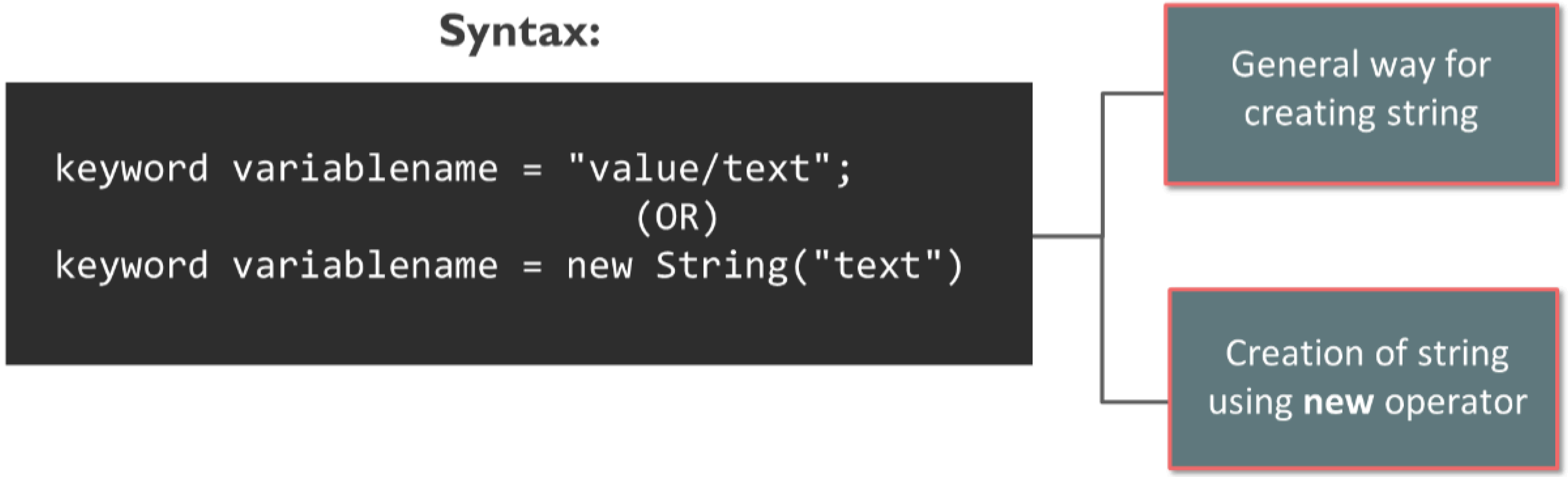
- Elaborate Primitive and Reference data types
- Make use of Arithmetic, Assignment, Logical and Comparison operators
- Recall the syntax and usage of the Conditional and Looping statements

Data Types and Operators in JavaScript

String

- JavaScript’s String is a type that is used to represent a sequence of characters.
- It is useful for holding data represented in text form.
- Sting data must be enclosed either in single(‘) or double quotation(“) marks.

Syntax:



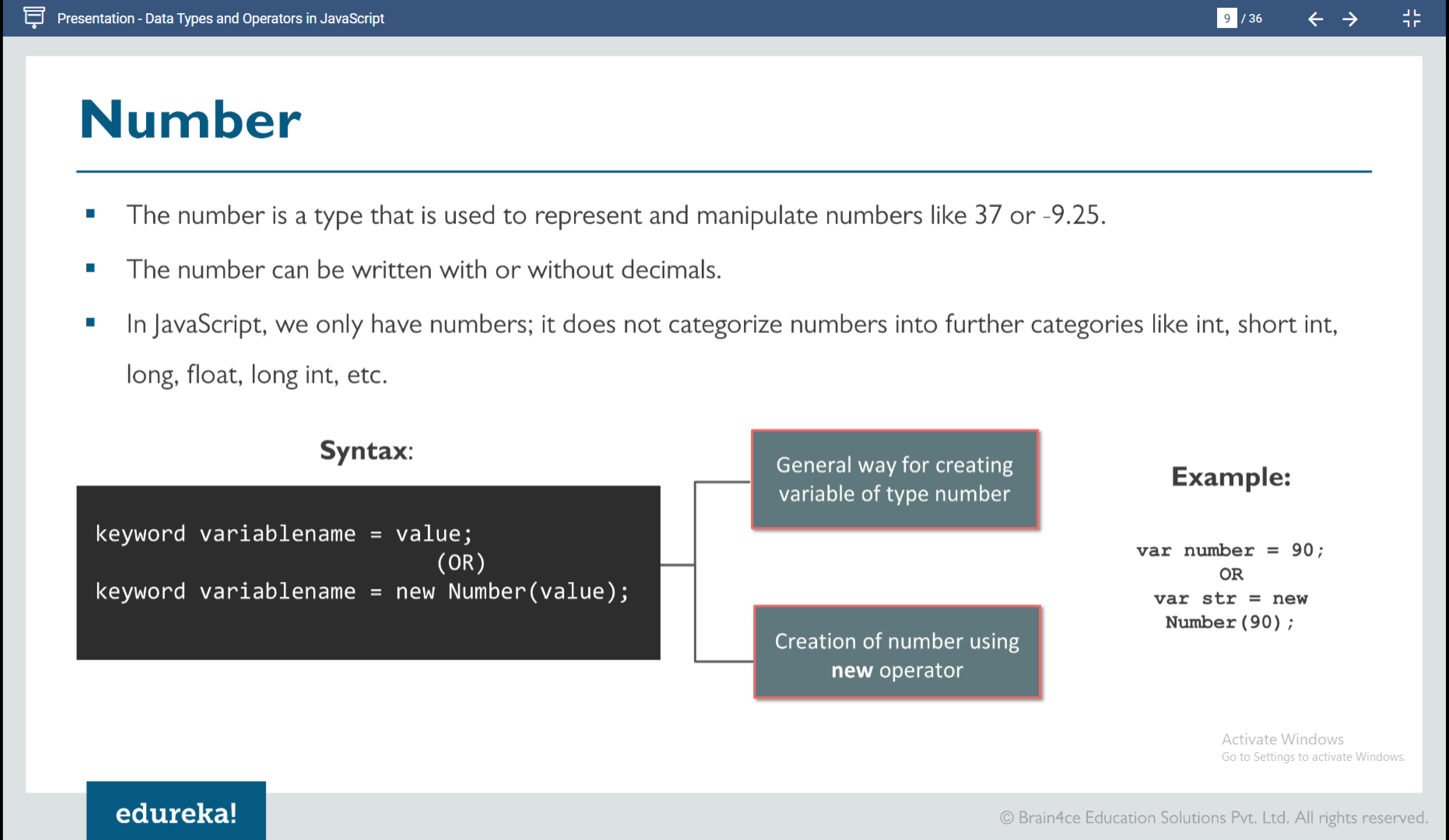
```
keyword variablename = "value/text";
                          (OR)
keyword variablename = new String("text")
```

General way for creating string
Creation of string using **new** operator

Example:

```
var str = " Edureka ";
OR
var str = new
String("Edureka " );
```

edureka!



Number

- The number is a type that is used to represent and manipulate numbers like 37 or -9.25.
- The number can be written with or without decimals.
- In JavaScript, we only have numbers; it does not categorize numbers into further categories like int, short int, long, float, long int, etc.

Syntax:

```
keyword variablename = value;
                         (OR)
keyword variablename = new Number(value);
```

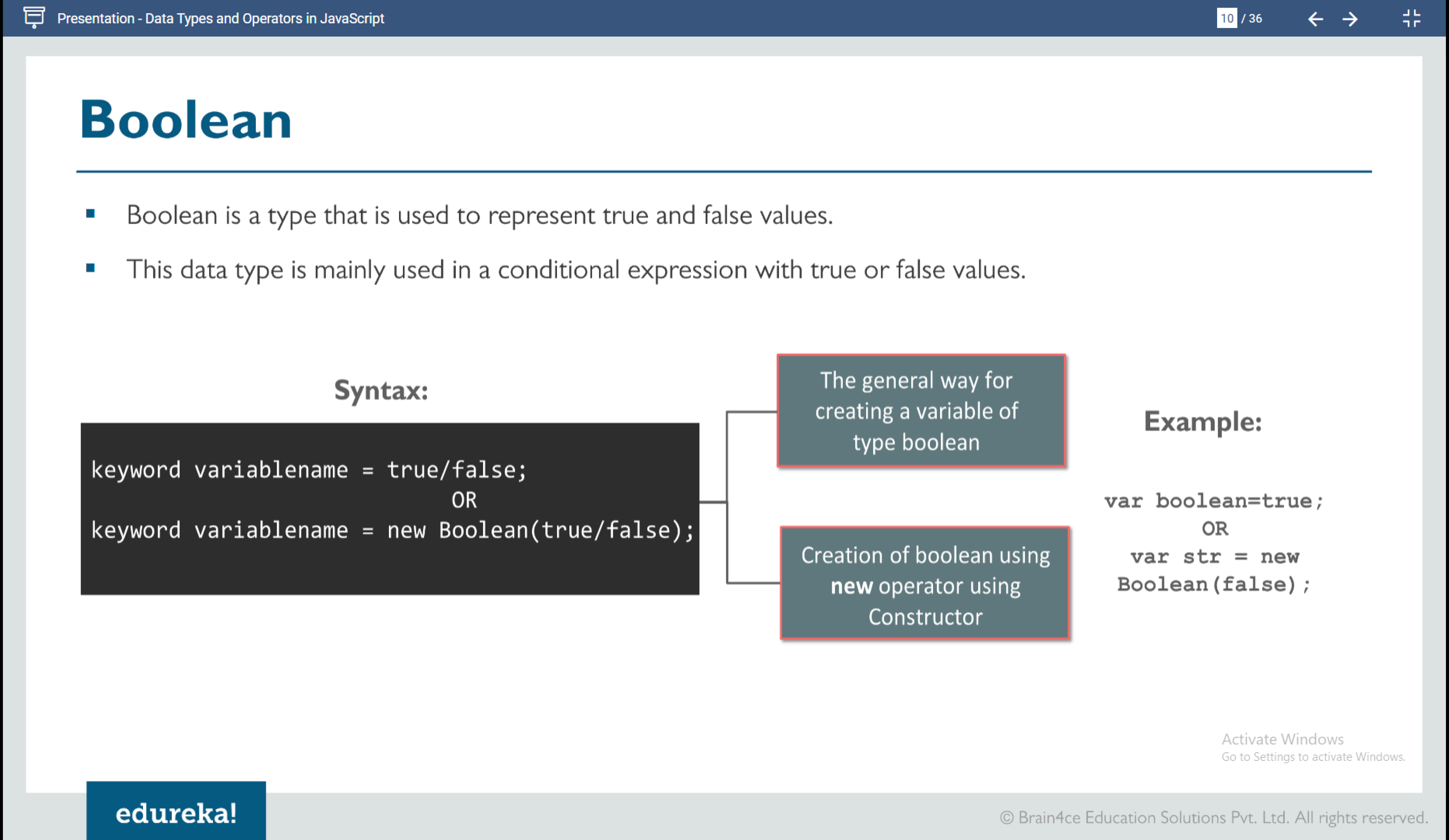
General way for creating variable of type number

Creation of number using **new** operator

Example:

```
var number = 90;
      OR
var str = new
 Number(90);
```

edureka!



Boolean

- Boolean is a type that is used to represent true and false values.
- This data type is mainly used in a conditional expression with true or false values.

Syntax:

```
keyword variablename = true/false;
                         OR
keyword variablename = new Boolean(true/false);
```

The general way for creating a variable of type boolean

Creation of boolean using **new** operator using Constructor

Example:

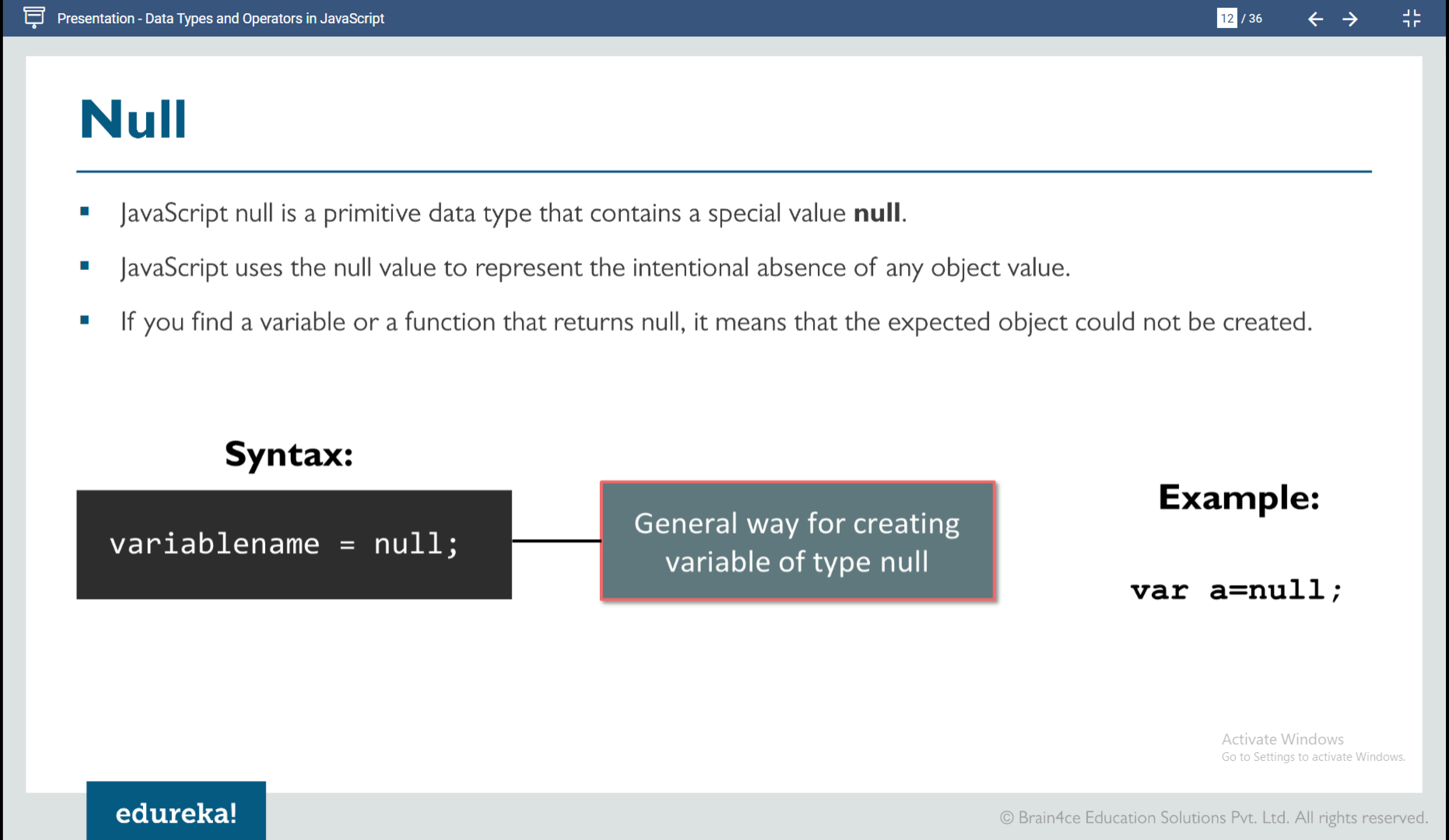
```
var boolean=true;
       OR
  var str = new
Boolean(false);
```

Undefined

- It is a primitive data type in JavaScript.
- The undefined property indicates that a variable has not been assigned a value or not declared at all.
- It indicates there is no value in the variable, or you can assume a variable has nothing inside it.

Variable with value – (Defined)

Variable without value – (Undefined)



Null

- JavaScript null is a primitive data type that contains a special value **null**.
- JavaScript uses the null value to represent the intentional absence of any object value.
- If you find a variable or a function that returns null, it means that the expected object could not be created.

Syntax:

```
variablename = null;
```

General way for creating variable of type null

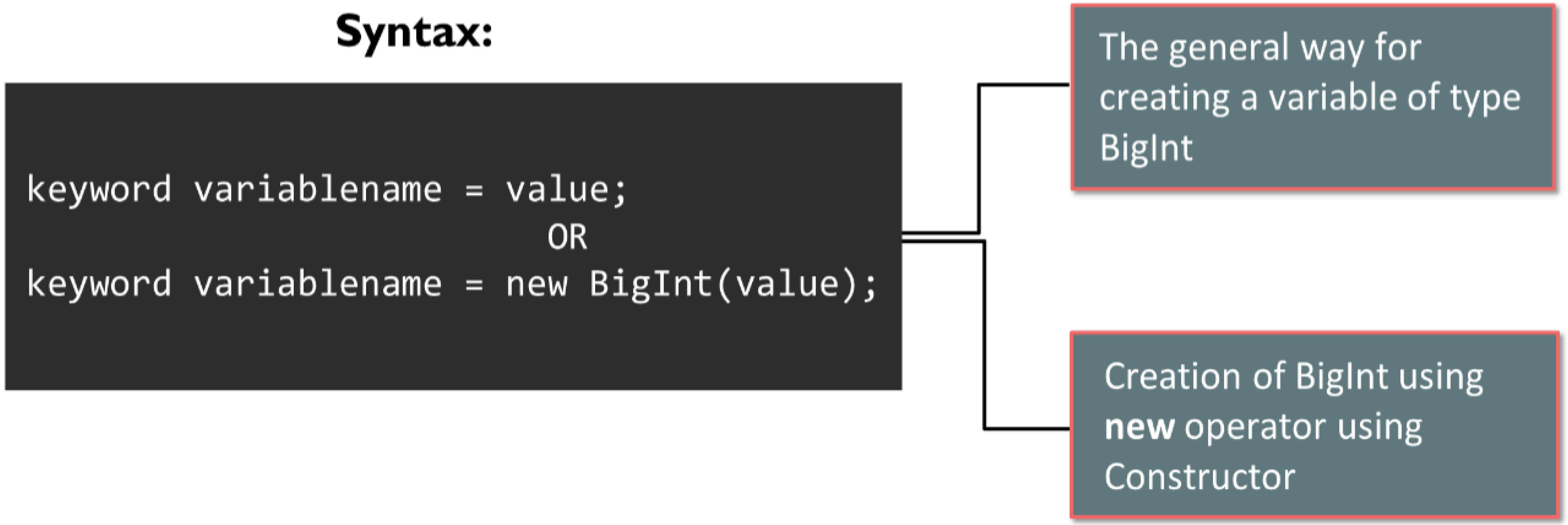
Example:

```
var a=null;
```

BigInt

- BigInt is a numeric primitive data type in JavaScript.
- The name of the data type itself tells that it can store large numbers. It stores a large number which is not possible for the Number data type to store.
- We must add **n** at the end of the number while assigning value to a BigInt variable.

Syntax:



```
keyword variablename = value;
                         OR
keyword variablename = new BigInt(value);
```

The general way for creating a variable of type BigInt

Creation of BigInt using **new** operator using Constructor

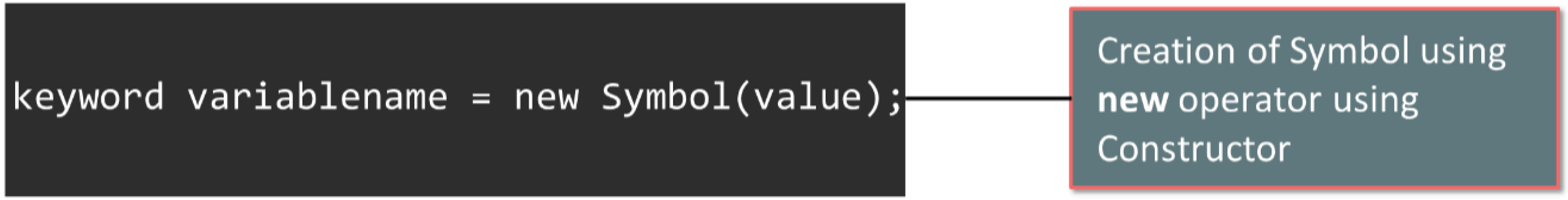
Example:

```
var a=39898998n;
OR
var
a=BigInt(59898)
```

Symbol

- A Symbol is an immutable primitive data type.
- It can be used to store the symbol values.
- It can be used as the key to an object

Syntax:



```
keyword variablename = new Symbol(value);
```

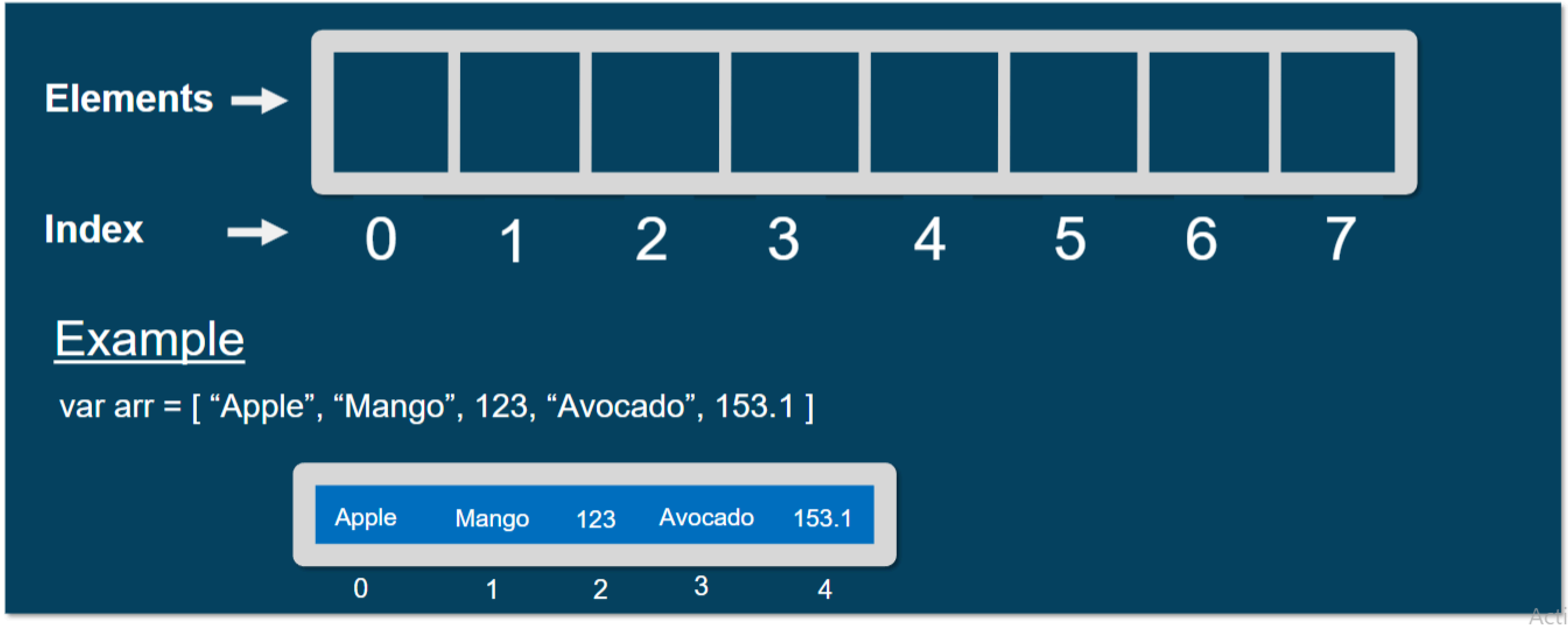
Creation of Symbol using **new** operator using Constructor

Example:

```
var a= Symbol(87)
```

Arrays

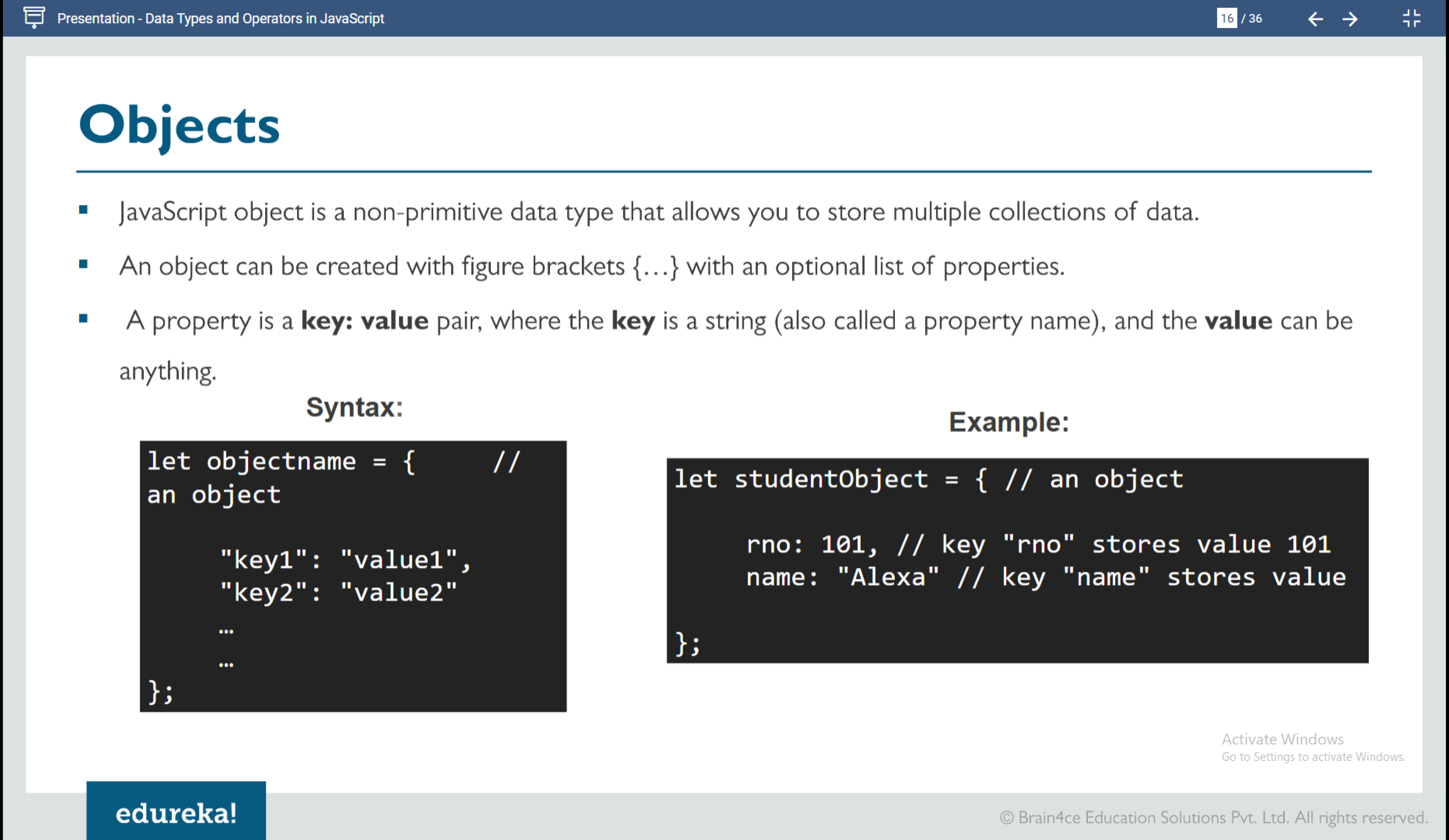
- Array is defined as a collection of heterogeneous elements.
- It helps to store multiple values of the different or same data type in a single variable.



Example

```
var arr = [ “Apple”, “Mango”, 123, “Avocado”, 153.1 ]
```

Apple	Mango	123	Avocado	153.1
0	1	2	3	4



Objects

- JavaScript object is a non-primitive data type that allows you to store multiple collections of data.
- An object can be created with figure brackets {…} with an optional list of properties.
- A property is a **key: value** pair, where the **key** is a string (also called a property name), and the **value** can be anything.

Syntax:

```
let objectname = {      //
an object

     "key1": "value1",
     "key2": "value2"
     …
     …
};
```

Example:

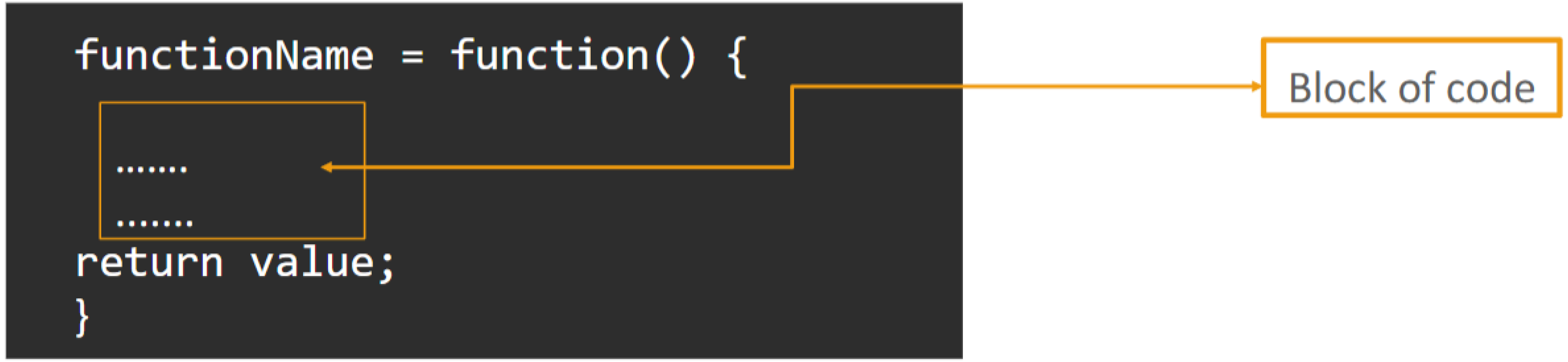
```
let studentObject = { // an object

     rno: 101, // key "rno" stores value 101
     name: "Alexa" // key "name" stores value

};
```

edureka!

Function

- JavaScript functions provide encapsulated environments, isolating their variables and logic from the global scope.
- Functions in JavaScript are versatile objects, usable as variables, arguments, and with properties, supporting advanced programming patterns.
- JavaScript functions can be used as callbacks, enabling asynchronous programming and handling events or data after an operation completes.

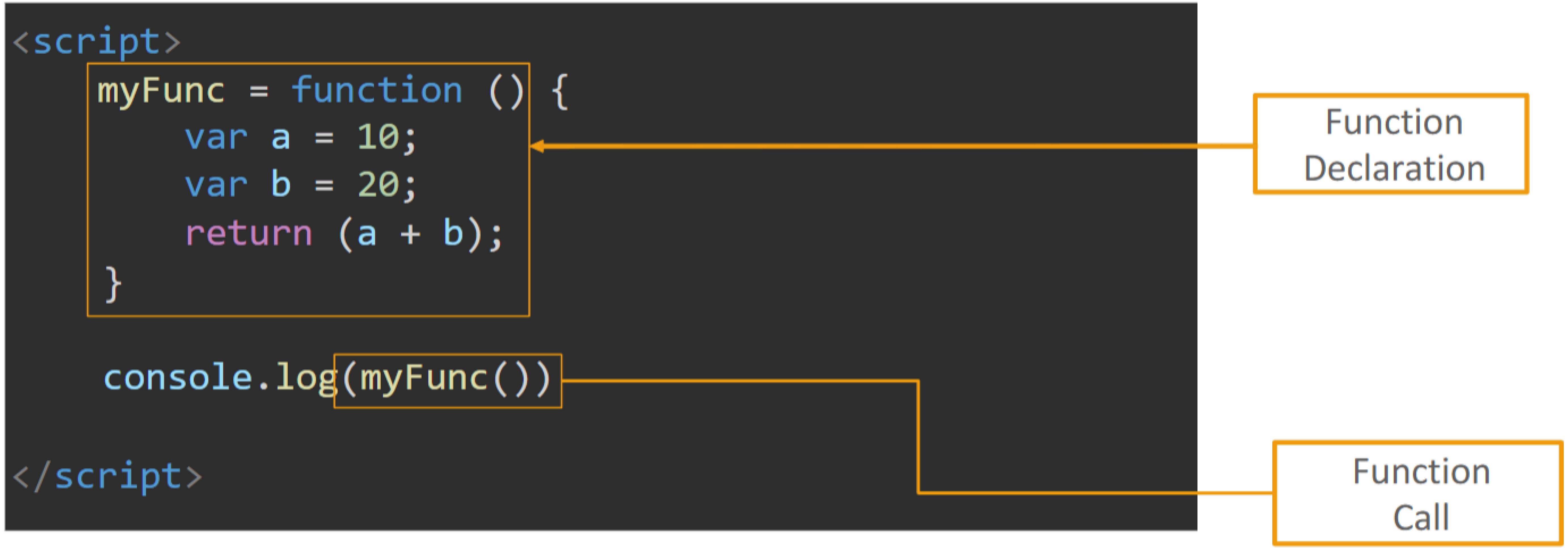
Syntax:



```
functionName = function() {
  .......
  .......
return value;
}
```

Block of code

Function - Example

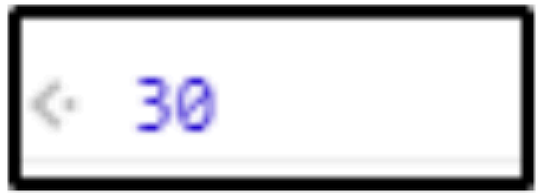


```
<script>
    myFunc = function () {
        var a = 10;
        var b = 20;
        return (a + b);
    }

    console.log(myFunc())

</script>
```

Function Declaration

Function Call

Output:



```
<· 30
```

edureka!