University of Global Village (UGV), Barishal

Content of the Sessional Course
University Student (UGV) Format

Program: Bachelor of Science in Computer Science Engineering (CSE)

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

Course Code	--
Name of Course Title	Web Page design & development (Level-2)
Course Type	Skill Course
Level	2nd Semester
Academic Session	Winter 2025
Name(s) of AcademicCourse teacher(s)	Sohag Mollik, Lecturer, CSE. Mobile: 01304142395 E-mail: sohag.cse.just@gmail.com
Consultation Hours:	

Web Page design & development Lab Student (UGV) Format	
Course Code: --	Credits: --
Exam Hours: --	CIE Marks: 30
Course for 2nd Semester, Bachelor of Science in Computer Science Engineering (CSE)	SEE Marks: 20

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

1. **Course Learning Outcome (CLO) at the end of the course, the students will be able to-**

CLO 1: Understand JavaScript Fundamentals.
CLO 2: Work with JavaScript Objects and Arrays.
CLO 3: Handle Strings, Numbers, and Dates.
CLO 4: Utilize Advanced JavaScript Concepts.
CLO 5: Data Handling and APIs.
CLO 6: Applications and Real-world Problem Solving.

2.Topics to be covered

Week	Topics	Teaching-Learning Strategy(s)	Class Hour	Practice Hour	Assessment Strategy(s)	Mapping with CLO
01	JavaScript syntax, variables, operators, and data types.	Lecture, Live demonstration & Hands-on exercises.	5h	4h	Participation, Lab Performance	CLO 1
02	Control structures: if-else, switch- case, loops (for, while, do-while).	Lecture, Interactive coding examples & Exercises.	5h	4h	Short quiz, Lab tasks	CLO 1
03	Functions: function declaration, expression, and arrow functions.	Lecture, Code-along & Problem-solving tasks.	5h	4h	Code reviews, Participation	CLO 1
04	Objects and arrays: properties, methods, and manipulation techniques.	Lecture, Hands-on exercises & Real-world examples.	5h	4h	Lab assignments, Class discussion.	CLO 2
05	DOM manipulation: element selection, events, and updates.	Lecture, Live demos, & Group exercises.	5h	5h	Participation, Mini-projects.	CLO 2
06	Strings and numbers: methods, search, templates, and properties.	Code-along, Practice tasks & Problem-solving.	5h	4h	Lab exercises, Quiz.	CLO 2
07	JavaScript dates, math operations, and random number generation.	Interactive demonstration, Practical examples.	5h	5h	Lab assignments, Participation.	CLO 2

08	Advanced JavaScript: destructuring, hoisting, `this` keyword, and scope.	Lecture, Hands-on challenges, Debugging tasks.	5h	4h	Debugging tasks, Short quiz.	CLO 3
09	Classes, constructors, and modules.	Code-along, Problem-solving & Case studies.	5h	4h	Lab exercises, Participation.	CLO 3
10	JSON: working with structured data, parsing, and manipulation.	Lecture, Data-driven tasks, Hands-on practice.	5h	4h	Lab assignments, Quiz.	CLO 4
11	Type conversion and regular expressions.	Lecture, Pattern-matching exercises.	5h	5h	Lab assignments, Quiz.	CLO 4
12	Sets and maps: methods, iteration, and usage.	Lecture, Group exercises, and Coding practice.	5h	3h	Participation, Mini-projects.	CLO 4
13	Debugging techniques and browser developer tools.	Practical debugging sessions, Code walkthrough.	5h	4h	Code debugging tasks.	CLO 5
14	JavaScript style guide, best practices, and performance optimization.	Lecture, Refactoring tasks, Group discussion	5h	5h	Lab reviews, Quiz.	CLO 5
15	Error handling and common JavaScript mistakes.	Lecture, Practical error-handling sessions.	5h	5h	Code review, Participation.	CLO 5
16	Real-world web application: integrating JavaScript with HTML and CSS.	Group project, Live guidance.	5h	5h	Project evaluation, Participation.	CLO 6
17	Final project development and deployment.	Project work, Mentoring	5h	2h	Project demonstration	CLO 6

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

3. Teaching-Learning Strategy:

•**Lecture:** Explain concepts with real-world examples and visual aids.
•**Live Demonstration:** Show step-by-step coding and debugging in real-time.
•**Interactive Coding Examples:** Engage students with challenges during class.
•**Hands-on Exercises:** Provide structured practice aligned with topics.
•**Code-along Sessions:** Guide students through practical coding implementations.
•**Problem-solving Tasks:** Assign real-world challenges to apply concepts.
•**Group Discussions:** Facilitate discussions on best practices and peer reviews.
•**Mini-projects:** Assign small, focused projects incorporating multiple concepts.
•**Debugging Sessions:** Teach error identification and resolution with tools.
•**Final Project Work:** Mentor students in developing a complete web application.

4.Assessment Strategy:

- ❖ **Lab Performance:** 30% (Lab participation, hands-on exercises, and weekly assessments)
- ❖ **Quizzes and Short Tests:** 20% (Regular quizzes on theoretical concepts)
- ❖ **Assignments and Reports:** 20% (Assignments related to data management, cloud integration, and security)
- ❖ **Project Evaluation:** 30% (Progress, final project implementation, and presentation)

5.Instructional Materials and References: Textbooks:

1."Eloquent JavaScript" by Marijn Haverbeke
2."JavaScript: The Definitive Guide" by David Flanagan

Additional References:

Follow w3school JavaScript & others website.

WEEK 1

Page
7-15

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

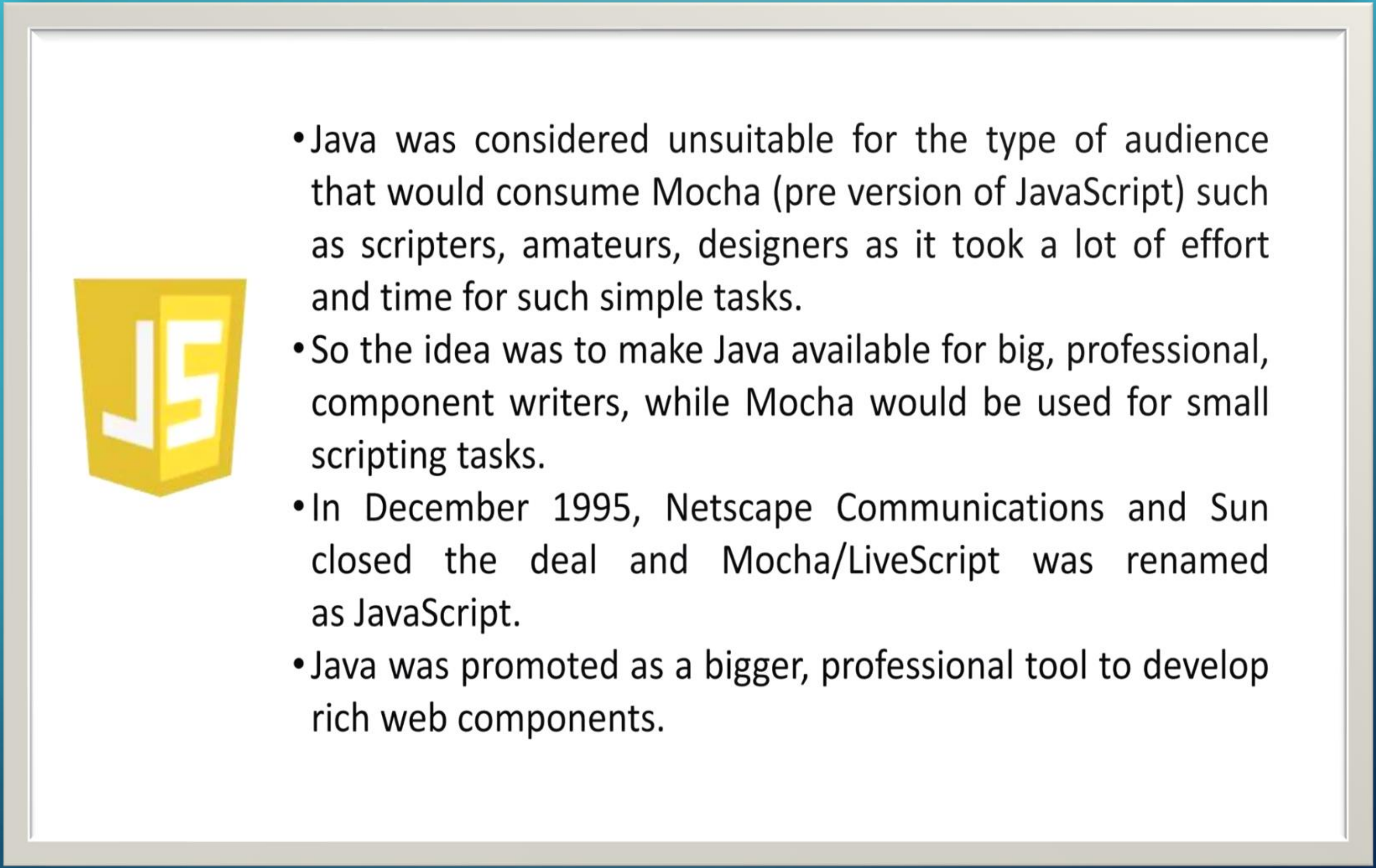
JavaScript Basic

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

```
<script>
if(age > 19){
    alert("Adult");
} else{
    alert("Teenager");
}
</script>
```

Origin of JavaScript

- **Netscape** Communications had the vision that the web needed a way to become more **dynamic**.
- They wanted Animations, Interaction and other forms of small Automation as part of the **web** of the future.
- The goal was to seek ways to *expand the web*.
- And that is exactly what gave birth to JavaScript.
- **Brendan Eich**, the father of **JavaScript**, was contracted by Netscape Communications to develop a scheme for the browser.



- Java was considered unsuitable for the type of audience that would consume Mocha (pre version of JavaScript) such as scripters, amateurs, designers as it took a lot of effort and time for such simple tasks.
- So the idea was to make Java available for big, professional, component writers, while Mocha would be used for small scripting tasks.
- In December 1995, Netscape Communications and Sun closed the deal and Mocha/LiveScript was renamed as JavaScript.
- Java was promoted as a bigger, professional tool to develop rich web components.

Introduction to JavaScript

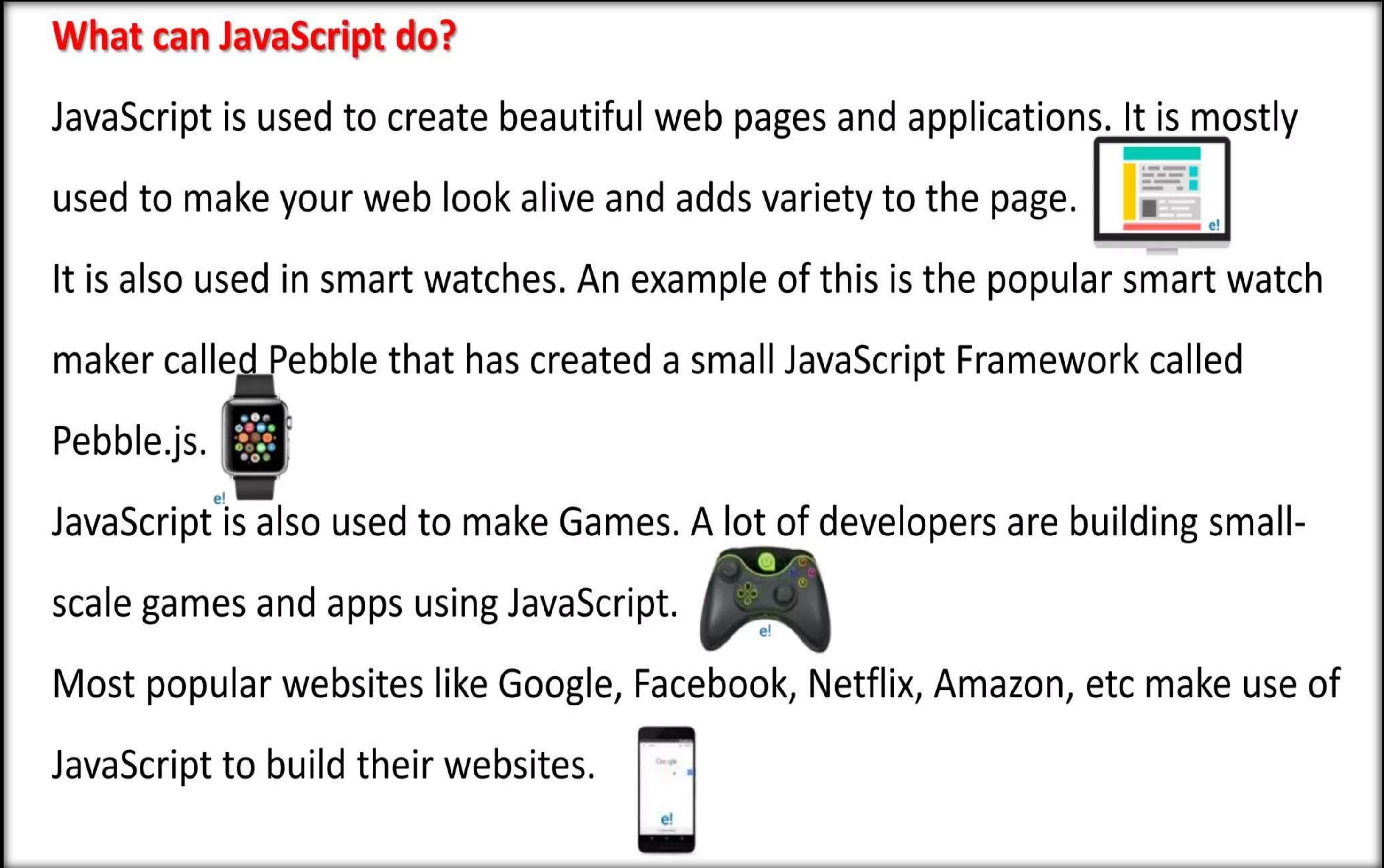
- ✓ JavaScript is a high level, interpreted, programming language used to make web pages more interactive.
- ✓ It is a very powerful client-side scripting language which makes your webpage more lively and interactive.
- ✓ It is a programming language that helps you to implement a complex and beautiful design on web pages.
- ✓ If you want your web page to look alive JavaScript is a must.

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

Features of JavaScript:

- It is a Scripting Language and has nothing to do with Java. Initially, It was named Mocha, then changed to LiveScript and finally it was named as JavaScript.
- JavaScript is an object-based programming language that supports polymorphism, encapsulation, and inheritance as well.
- You can run JavaScript not only in the browser but also on the server and any device which has a JavaScript Engine.

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



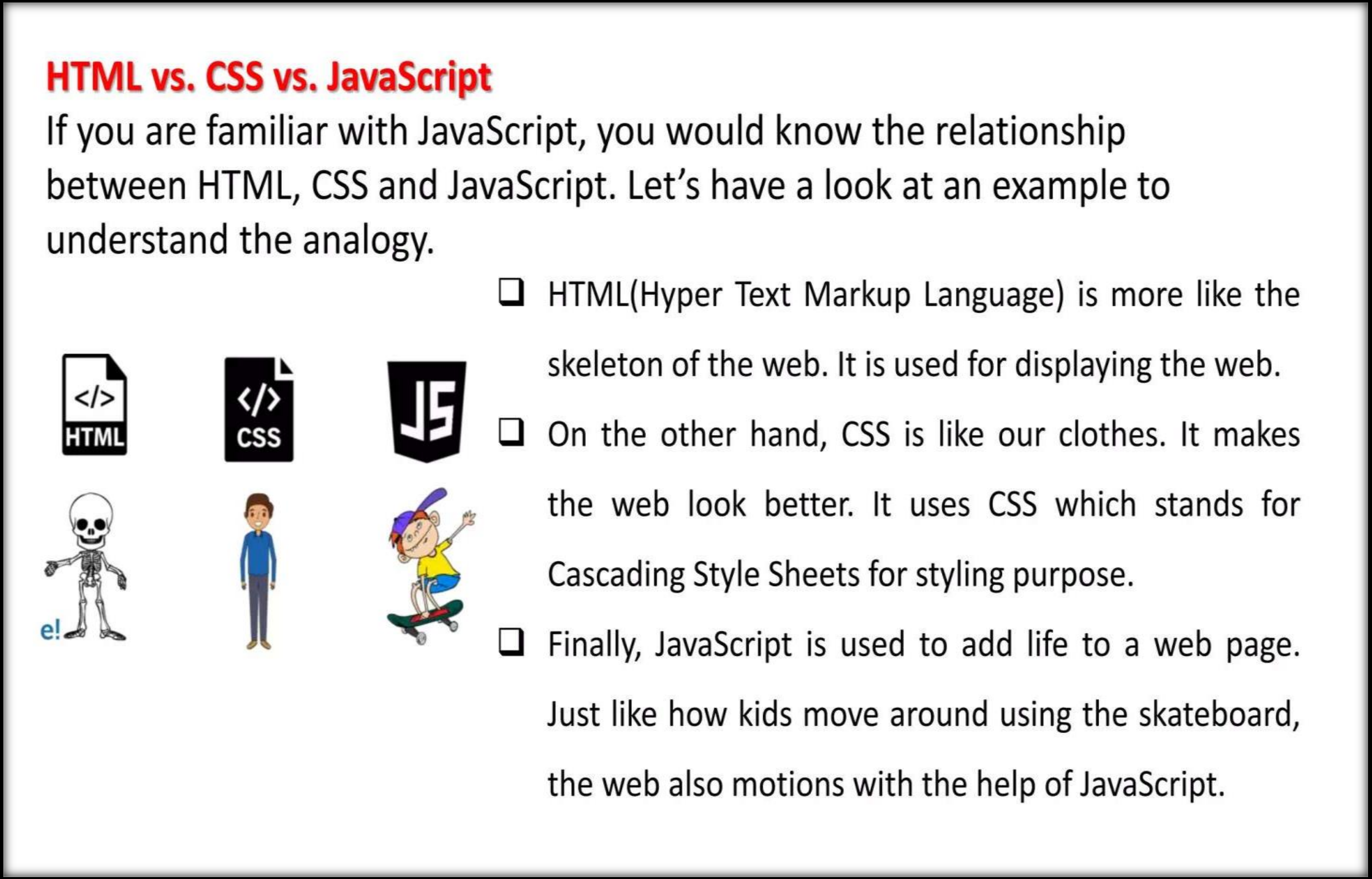
What can JavaScript do?

JavaScript is used to create beautiful web pages and applications. It is mostly used to make your web look alive and adds variety to the page.

It is also used in smart watches. An example of this is the popular smart watch maker called Pebble that has created a small JavaScript Framework called Pebble.js.

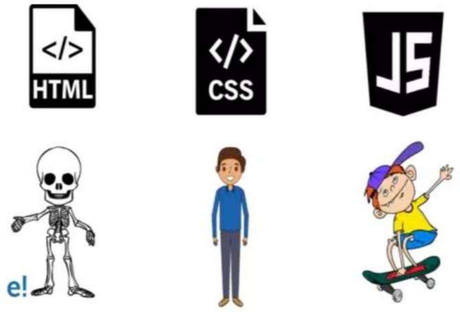
JavaScript is also used to make Games. A lot of developers are building small-scale games and apps using JavaScript.

Most popular websites like Google, Facebook, Netflix, Amazon, etc make use of JavaScript to build their websites.

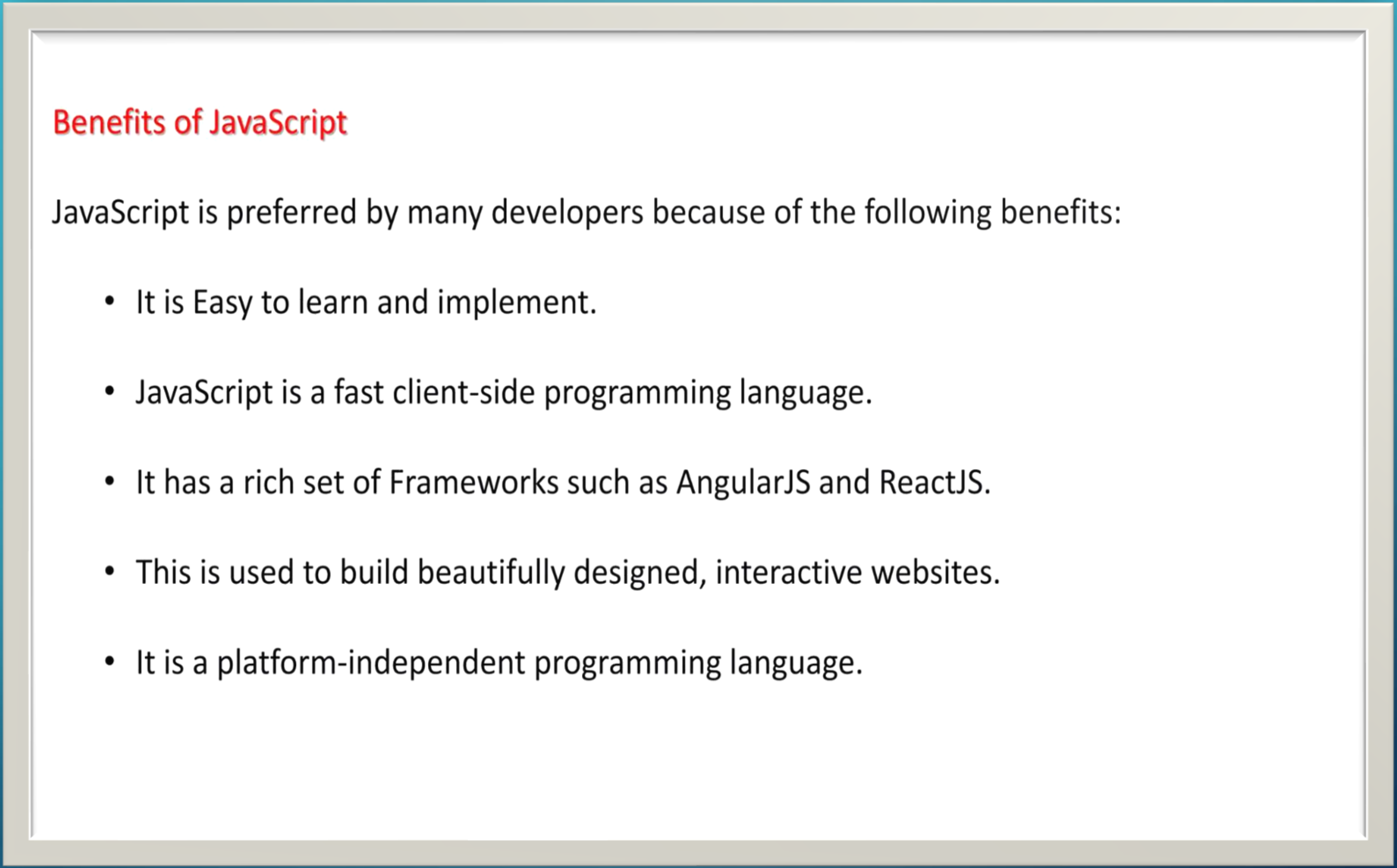


HTML vs. CSS vs. JavaScript

If you are familiar with JavaScript, you would know the relationship between HTML, CSS and JavaScript. Let's have a look at an example to understand the analogy.



- ❑ HTML(Hyper Text Markup Language) is more like the skeleton of the web. It is used for displaying the web.
- ❑ On the other hand, CSS is like our clothes. It makes the web look better. It uses CSS which stands for Cascading Style Sheets for styling purpose.
- ❑ Finally, JavaScript is used to add life to a web page. Just like how kids move around using the skateboard, the web also motions with the help of JavaScript.



Benefits of JavaScript

JavaScript is preferred by many developers because of the following benefits:

- It is Easy to learn and implement.
- JavaScript is a fast client-side programming language.
- It has a rich set of Frameworks such as AngularJS and ReactJS.
- This is used to build beautifully designed, interactive websites.
- It is a platform-independent programming language.

For more information please visit this link

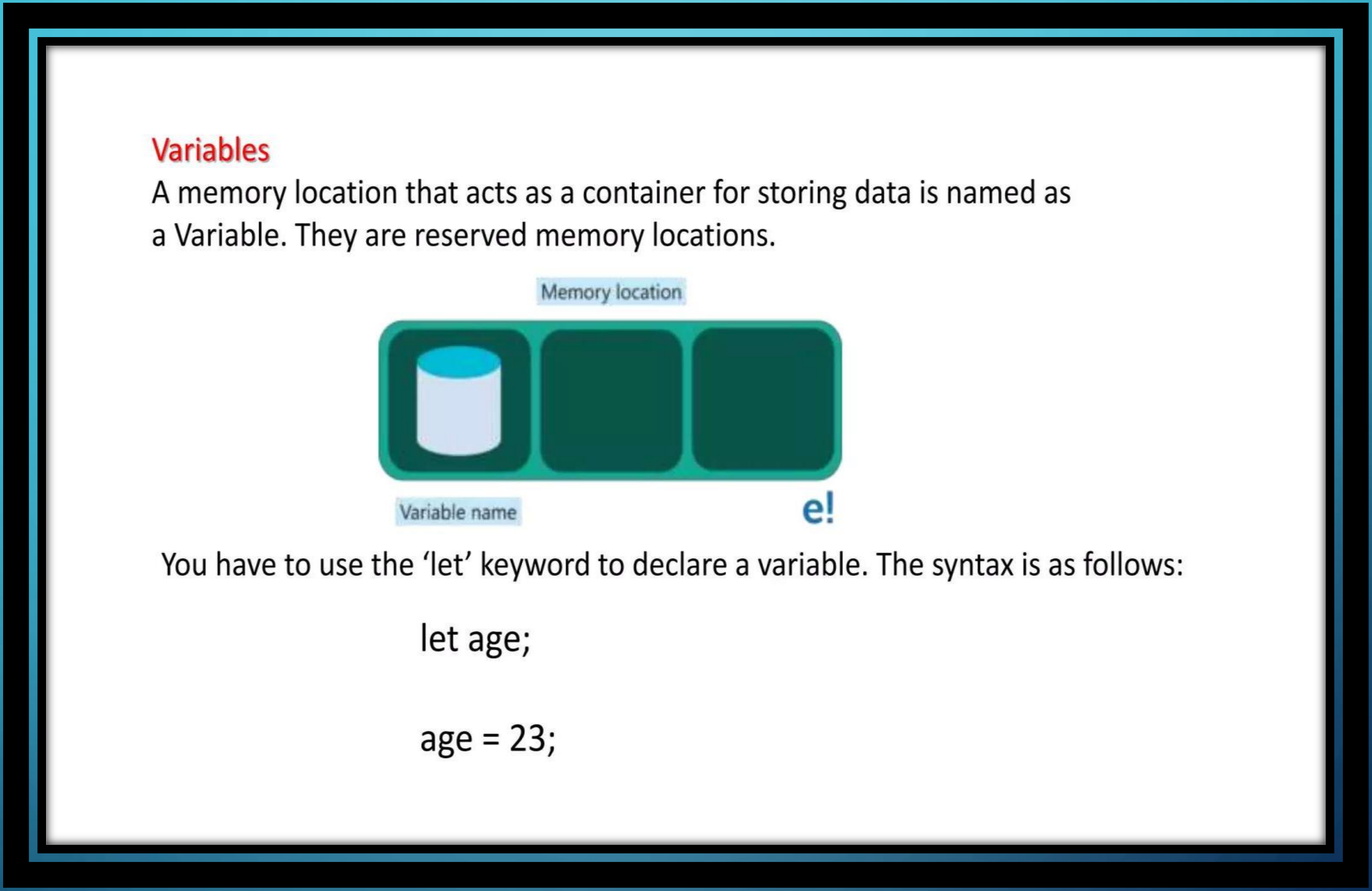
https://www.edureka.co/blog/javascript-tutorial/#variables

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

WEEK 2

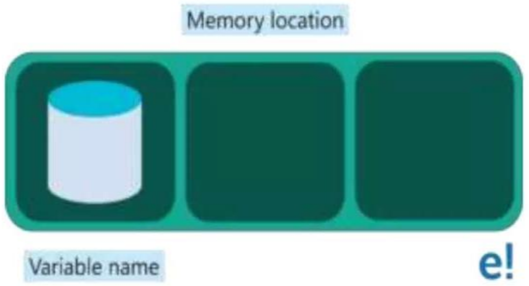
Page
17-21

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



Variables

A memory location that acts as a container for storing data is named as a Variable. They are reserved memory locations.



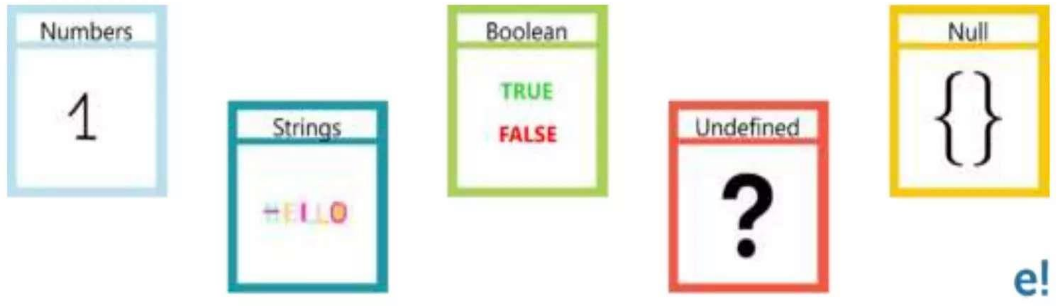
You have to use the 'let' keyword to declare a variable. The syntax is as follows:

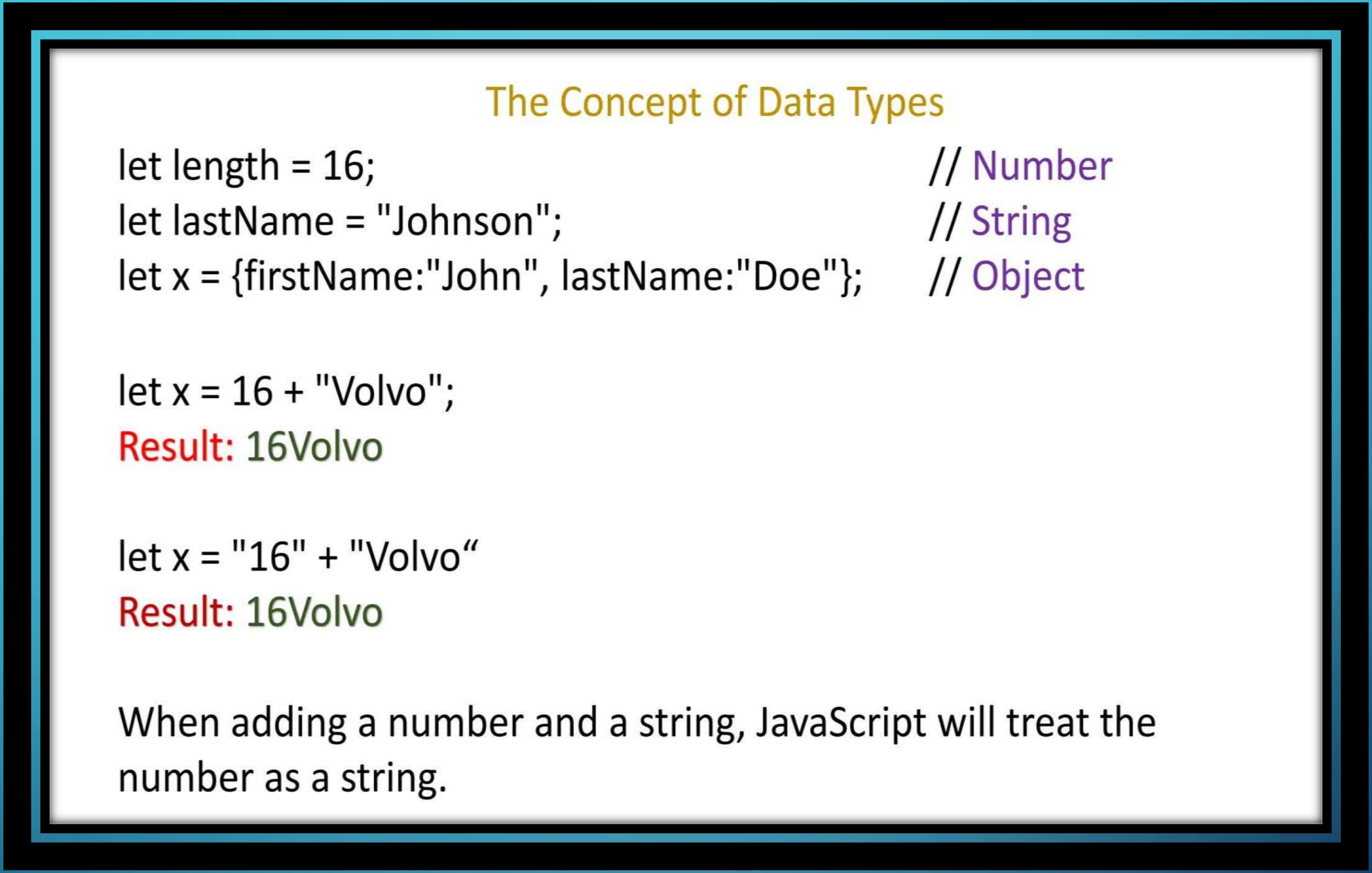
```
let age;

age = 23;
```

Data Types

You can assign different types of values to a variable such as a number or a string. There are different data types such as:

- Numbers
- Strings
- Boolean
- Undefined
- Null





The Concept of Data Types

```
let length = 16;                                    // Number
let lastName = "Johnson";                           // String
let x = {firstName:"John", lastName:"Doe"};         // Object

let x = 16 + "Volvo";
```

Result: 16Volvo

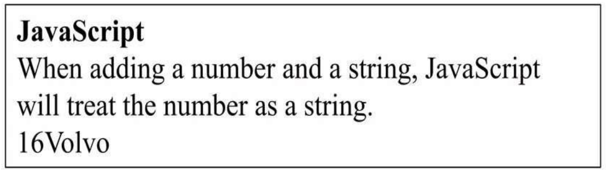
```
let x = "16" + "Volvo“
```

Result: 16Volvo

When adding a number and a string, JavaScript will treat the number as a string.

```html
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript</h2>

<p>When adding a number and a string, JavaScript will treat
the number as a string.</p>

<p id="demo"></p>

<script>
let x = 16 + "Volvo";
document.getElementById("demo").innerHTML = x;
</script>

</body>
</html>
```

Result:



JavaScript

```
When adding a number and a string, JavaScript
will treat the number as a string.
16Volvo
```

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

```
let x = 16 + 4 + "Volvo";
```
Result: 20Volvo

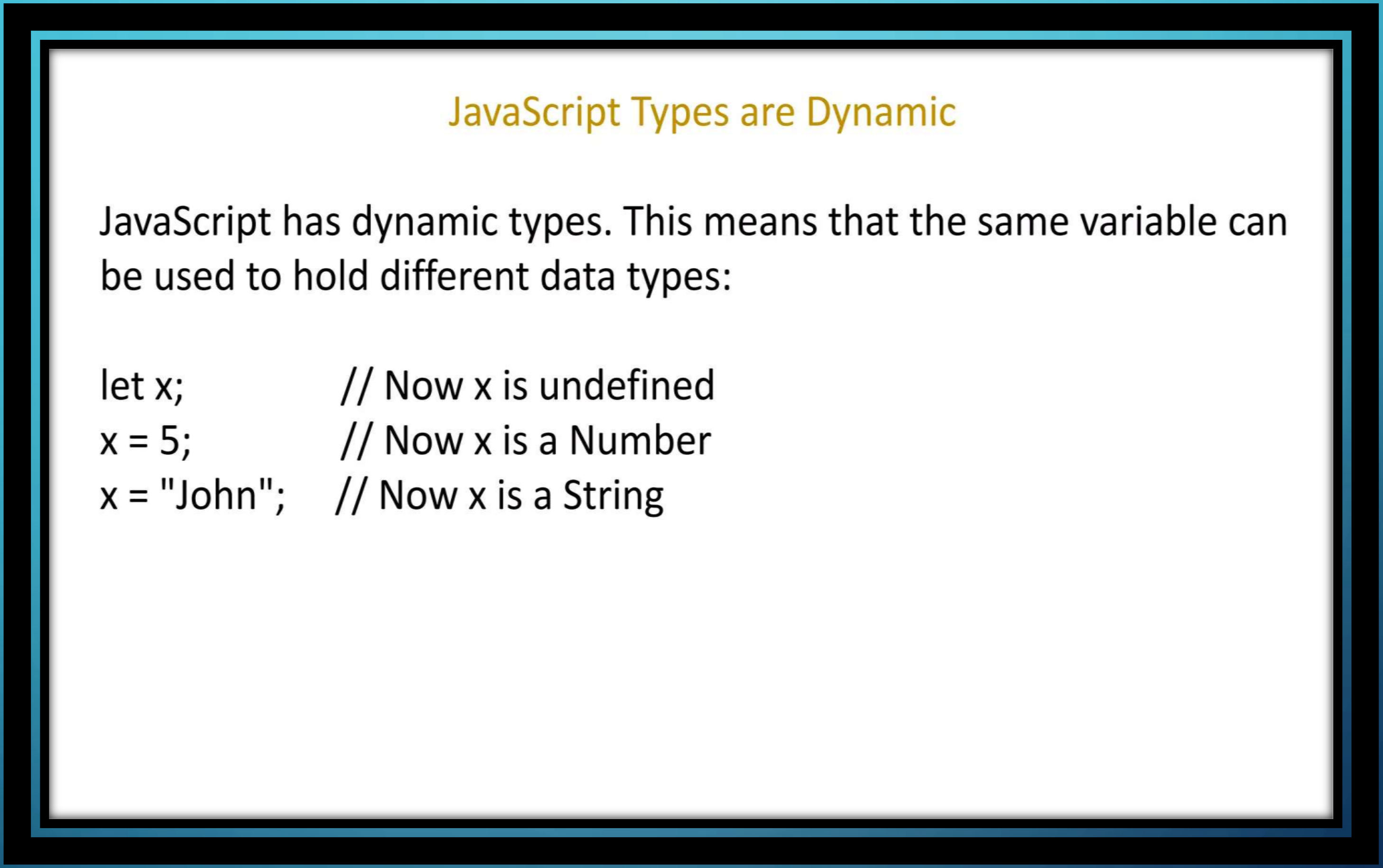
JavaScript treats 16 and 4 as numbers, until it reaches "Volvo".

```
let x = "Volvo" + 16 + 4;
```
Result: Volvo164

Since the first operand is a string, all operands are treated as strings.

WEEK 3

Page
23-27

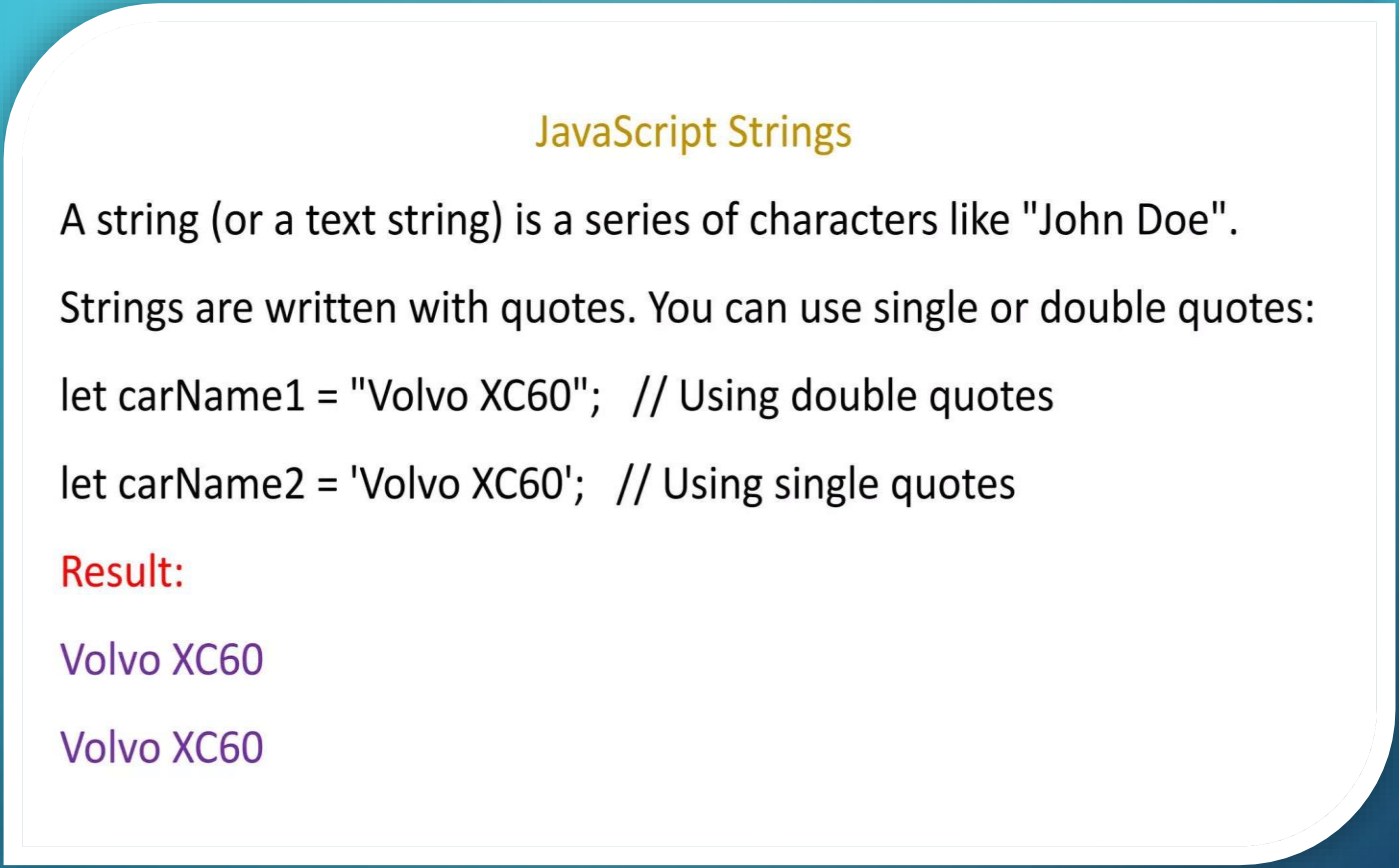


JavaScript Types are Dynamic

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

```
let x;          // Now x is undefined
x = 5;          // Now x is a Number
x = "John";     // Now x is a String
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Strings

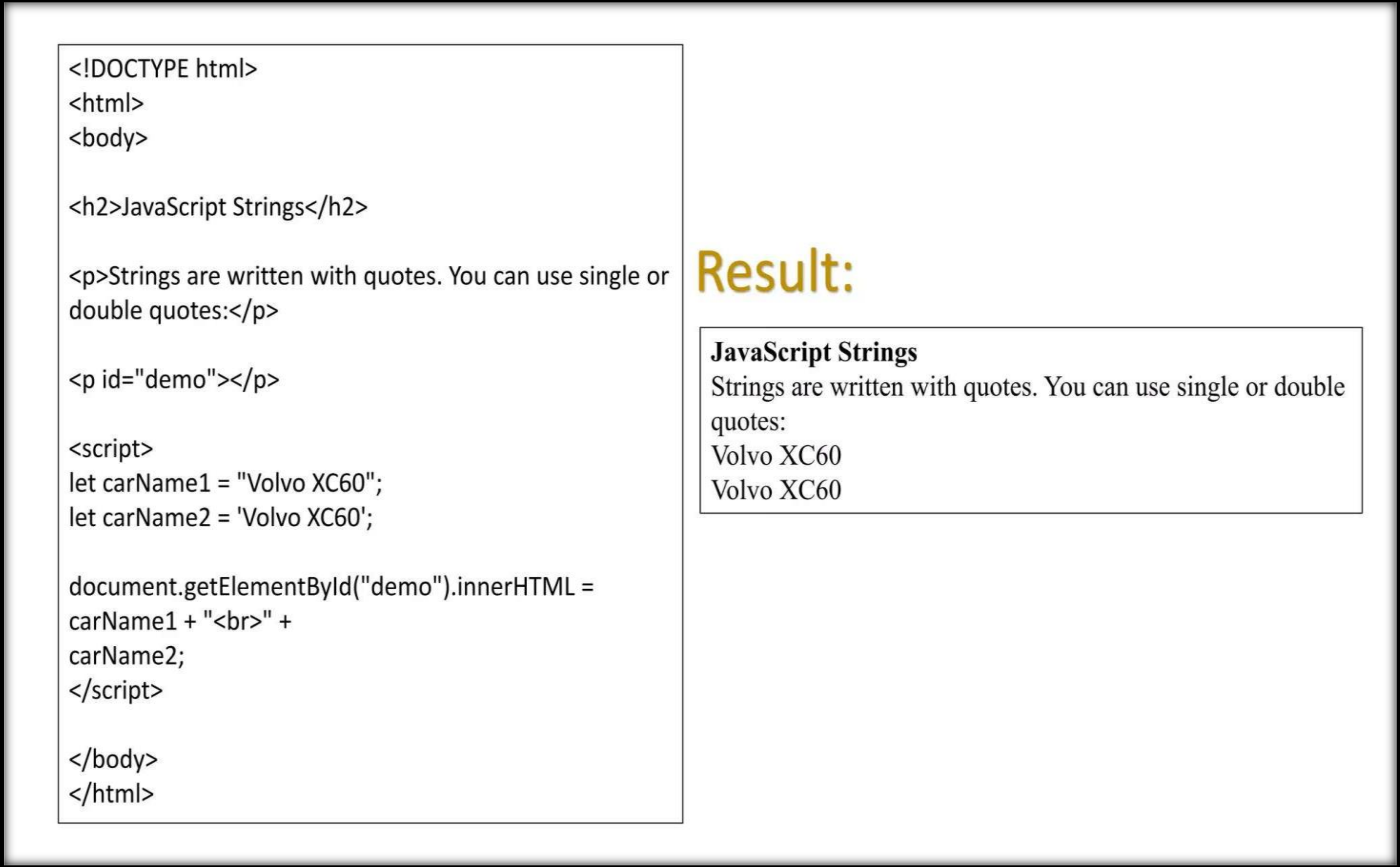
A string (or a text string) is a series of characters like "John Doe".

Strings are written with quotes. You can use single or double quotes:

```
let carName1 = "Volvo XC60";   // Using double quotes
let carName2 = 'Volvo XC60';   // Using single quotes
```

Result:

```
Volvo XC60

Volvo XC60
```



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Strings</h2>

<p>Strings are written with quotes. You can use single or
double quotes:</p>

<p id="demo"></p>

<script>
let carName1 = "Volvo XC60";
let carName2 = 'Volvo XC60';

document.getElementById("demo").innerHTML =
carName1 + "<br>" +
carName2;
</script>

</body>
</html>
```

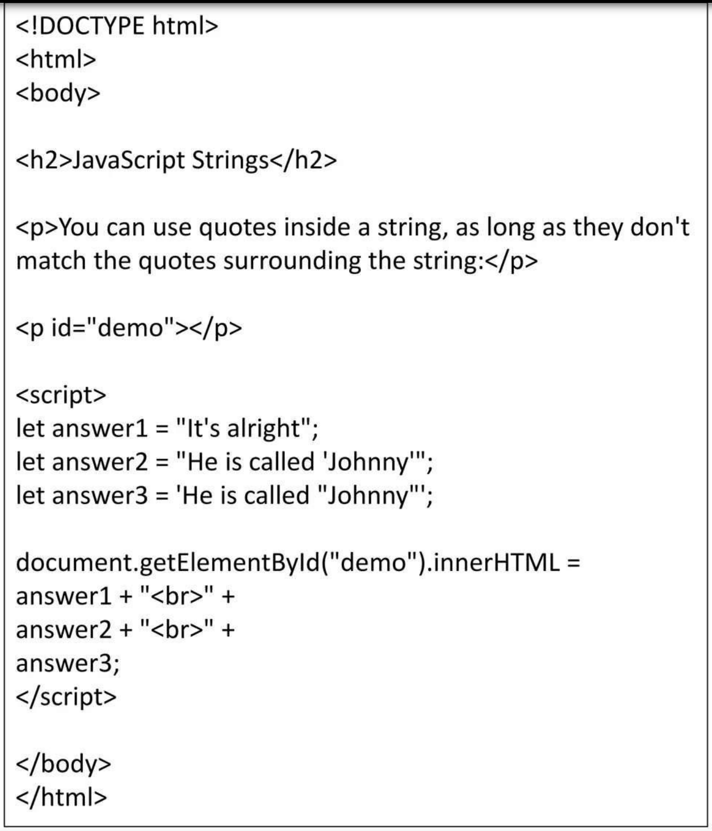
Result:

JavaScript Strings
Strings are written with quotes. You can use single or double quotes:
Volvo XC60
Volvo XC60

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

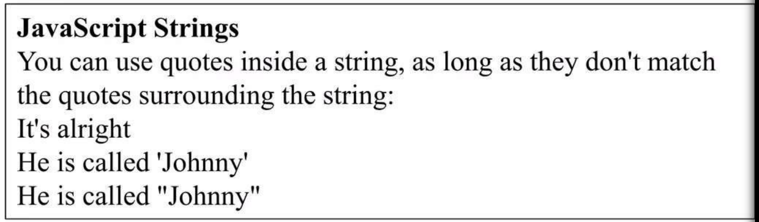
You can use quotes inside a string, as long as they don't match the quotes surrounding the string:

```
let answer1 = "It's alright";             // Single quote inside double quotes

let answer2 = "He is called 'Johnny'";    // Single quotes inside double quotes

let answer3 = 'He is called "Johnny"';    // Double quotes inside single quotes
```



```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Strings</h2>

<p>You can use quotes inside a string, as long as they don't
match the quotes surrounding the string:</p>

<p id="demo"></p>

<script>
let answer1 = "It's alright";
let answer2 = "He is called 'Johnny'";
let answer3 = 'He is called "Johnny"';

document.getElementById("demo").innerHTML =
answer1 + "<br>" +
answer2 + "<br>" +
answer3;
</script>

</body>
</html>
```

Result:



JavaScript Strings
You can use quotes inside a string, as long as they don't match the quotes surrounding the string:
It's alright
He is called 'Johnny'
He is called "Johnny"

WEEK 4

Page
29-30

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

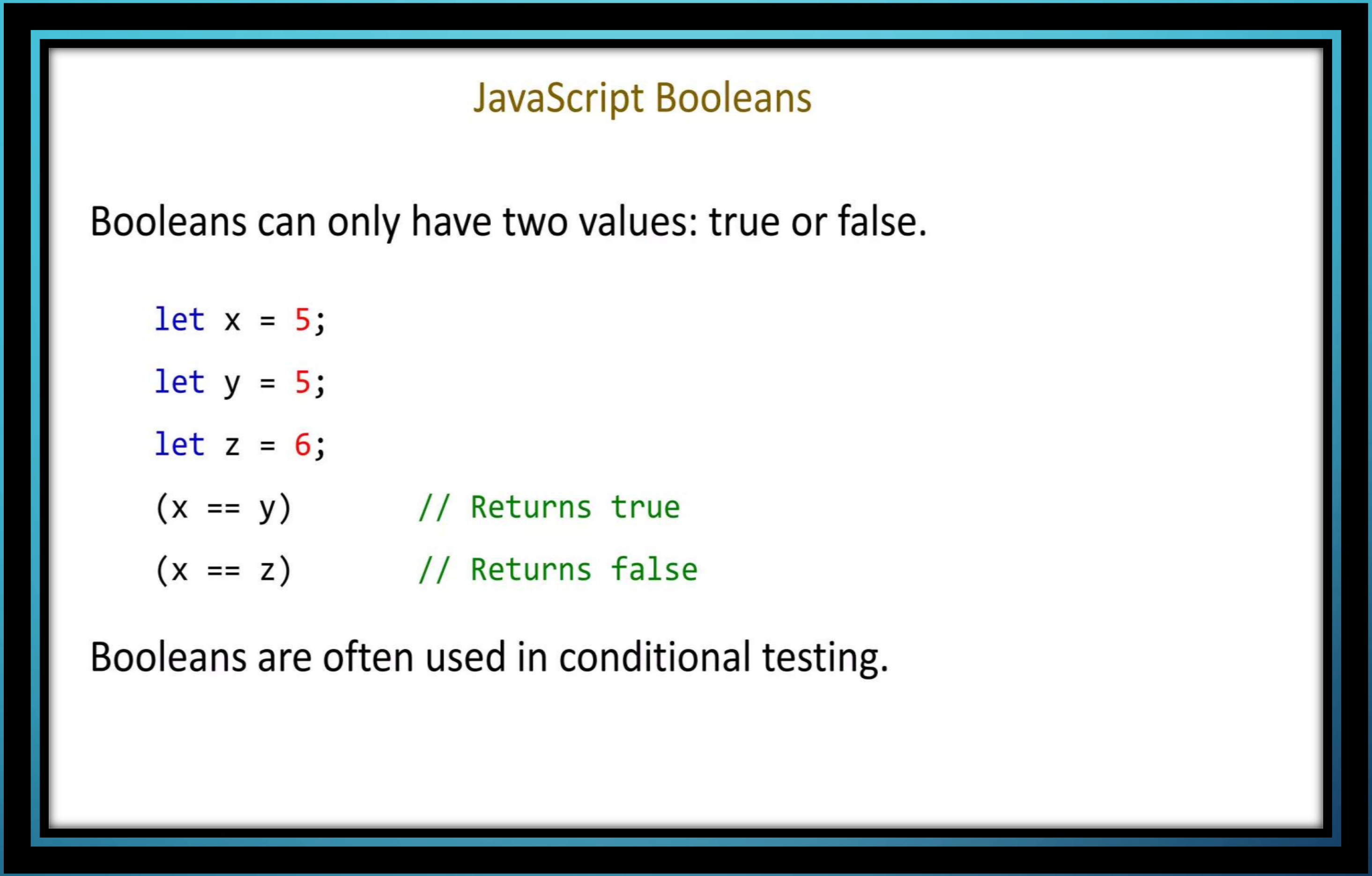
JavaScript Numbers

JavaScript has only one type of numbers.
Numbers can be written with, or without decimals:

```
let x1 = 34.00;     // Written with decimals

let x2 = 34;        // Written without decimals
```

Extra large or extra small numbers can be written with scientific (exponential) notation:

```
let y = 123e5;      // 12300000
let z = 123e-5;     // 0.00123
```



JavaScript Booleans

Booleans can only have two values: true or false.

```
let x = 5;
let y = 5;
let z = 6;
(x == y)       // Returns true
(x == z)       // Returns false
```

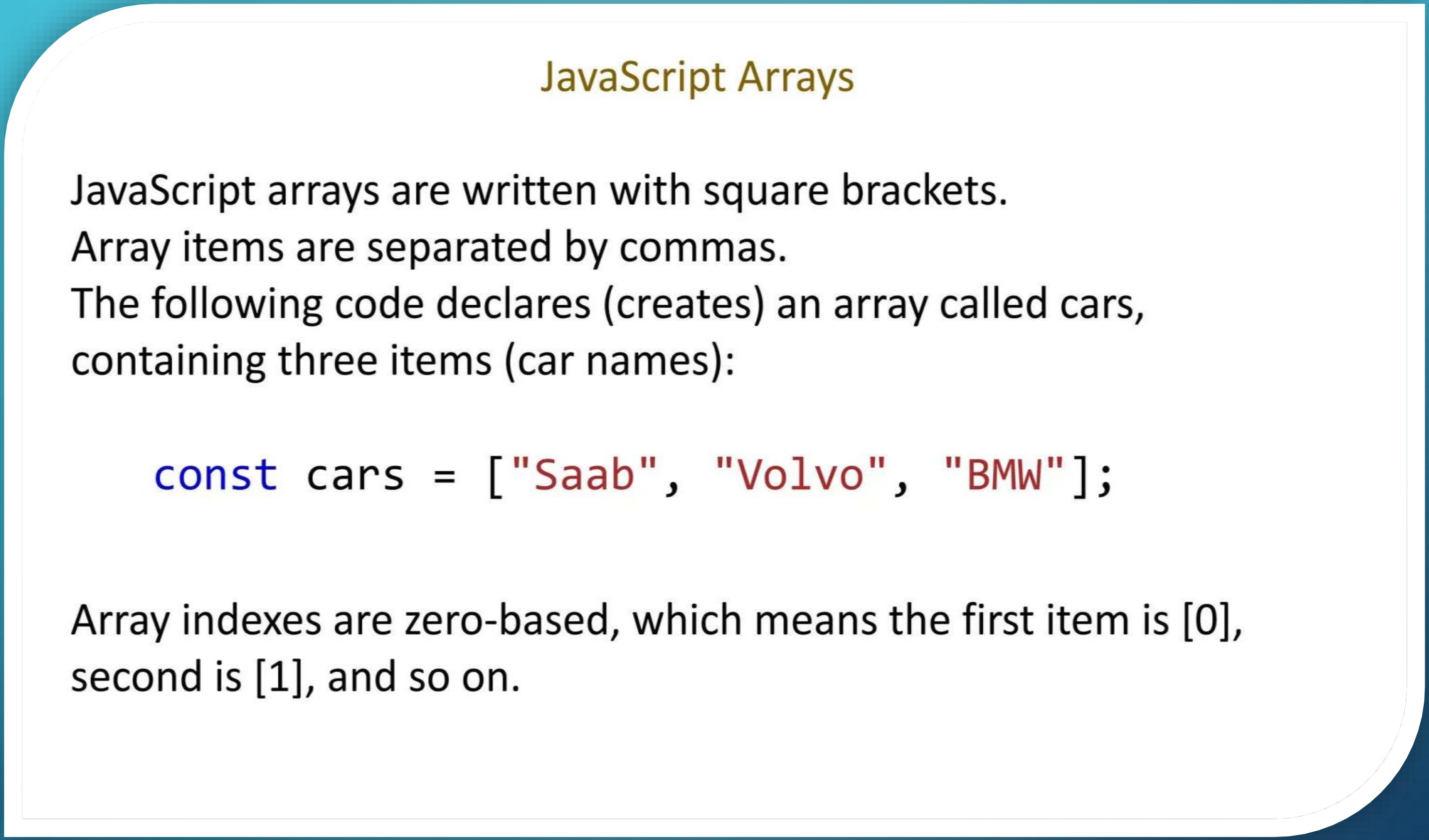
Booleans are often used in conditional testing.

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

WEEK 5 & 6

Page
32-37

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Arrays

JavaScript arrays are written with square brackets.
Array items are separated by commas.
The following code declares (creates) an array called cars, containing three items (car names):

```
const cars = ["Saab", "Volvo", "BMW"];
```

Array indexes are zero-based, which means the first item is [0], second is [1], and so on.

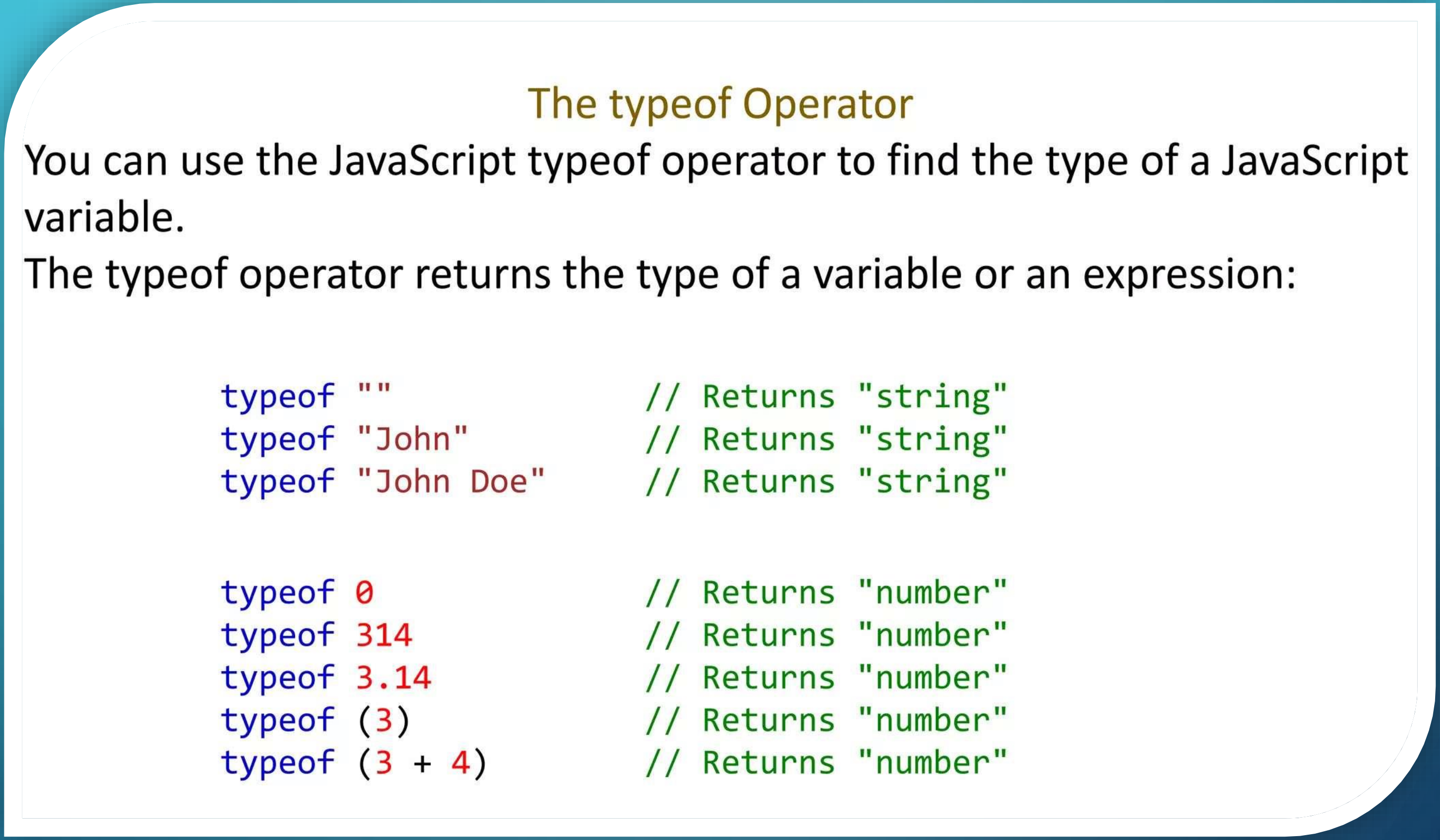
JavaScript Objects

JavaScript objects are written with curly braces {}.
Object properties are written as name: value pairs, separated by commas.

```
const person = {firstName:"John", lastName:"Doe",
age:50, eyeColor:"blue"};
```

The object (person) in the example above has 4 properties: firstName, lastName, age, and eyeColor.

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



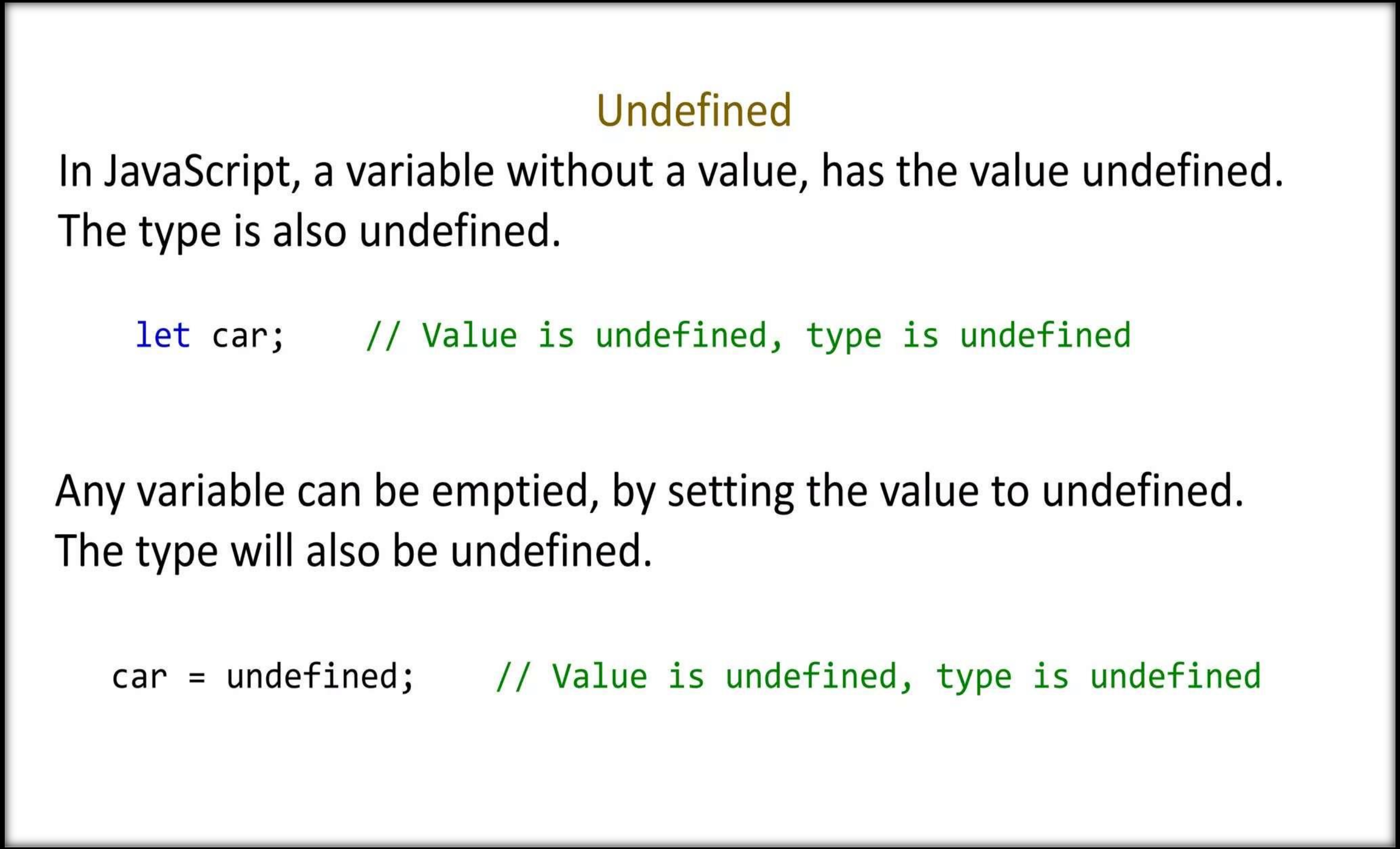
The typeof Operator

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

```
typeof ""             // Returns "string"
typeof "John"         // Returns "string"
typeof "John Doe"     // Returns "string"

typeof 0              // Returns "number"
typeof 314            // Returns "number"
typeof 3.14           // Returns "number"
typeof (3)            // Returns "number"
typeof (3 + 4)        // Returns "number"
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



Undefined

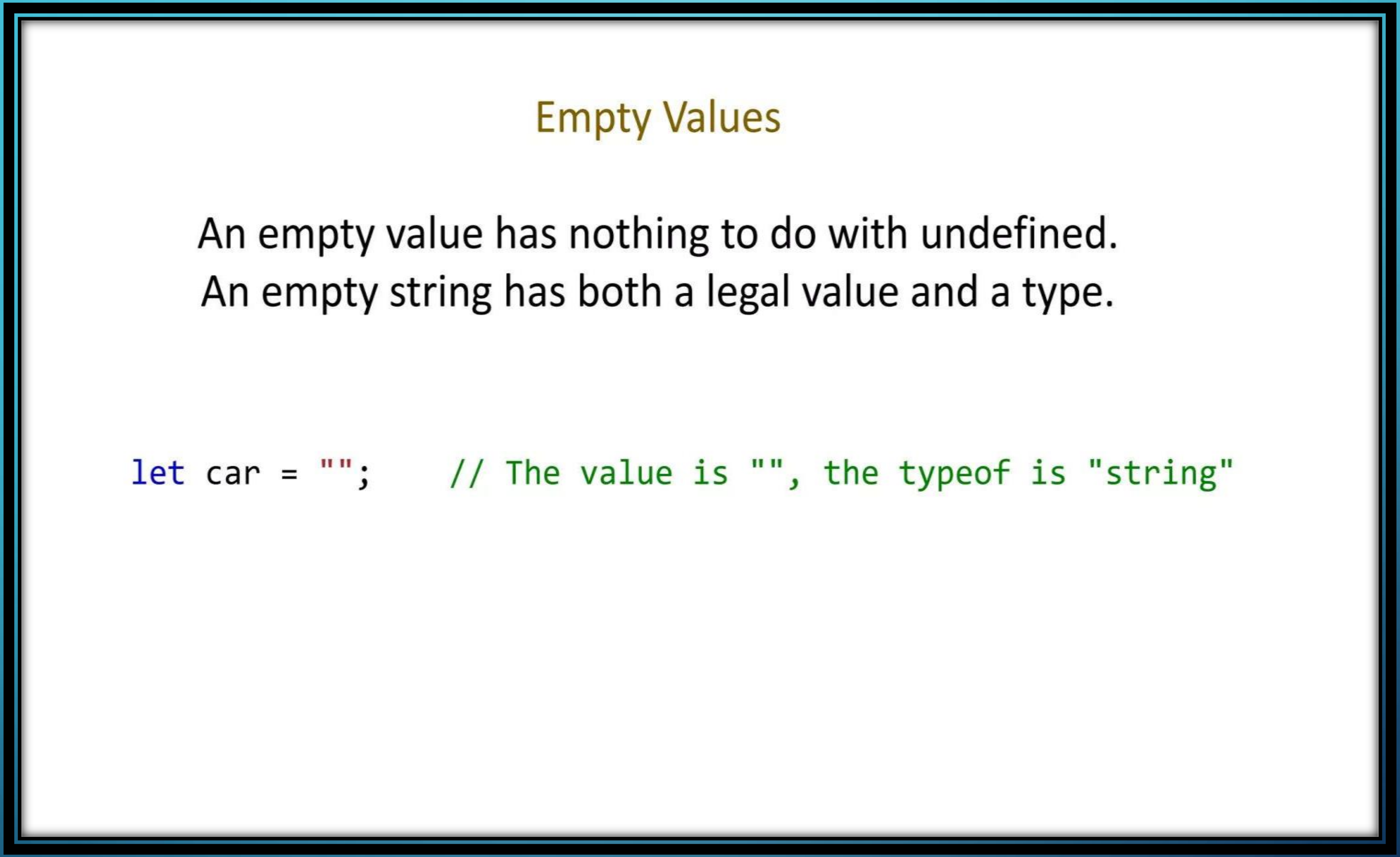
In JavaScript, a variable without a value, has the value undefined. The type is also undefined.

```
let car;    // Value is undefined, type is undefined
```

Any variable can be emptied, by setting the value to undefined. The type will also be undefined.

```
car = undefined;    // Value is undefined, type is undefined
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



Empty Values

An empty value has nothing to do with undefined.
An empty string has both a legal value and a type.

```
let car = "";    // The value is "", the typeof is "string"
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

Exercise:

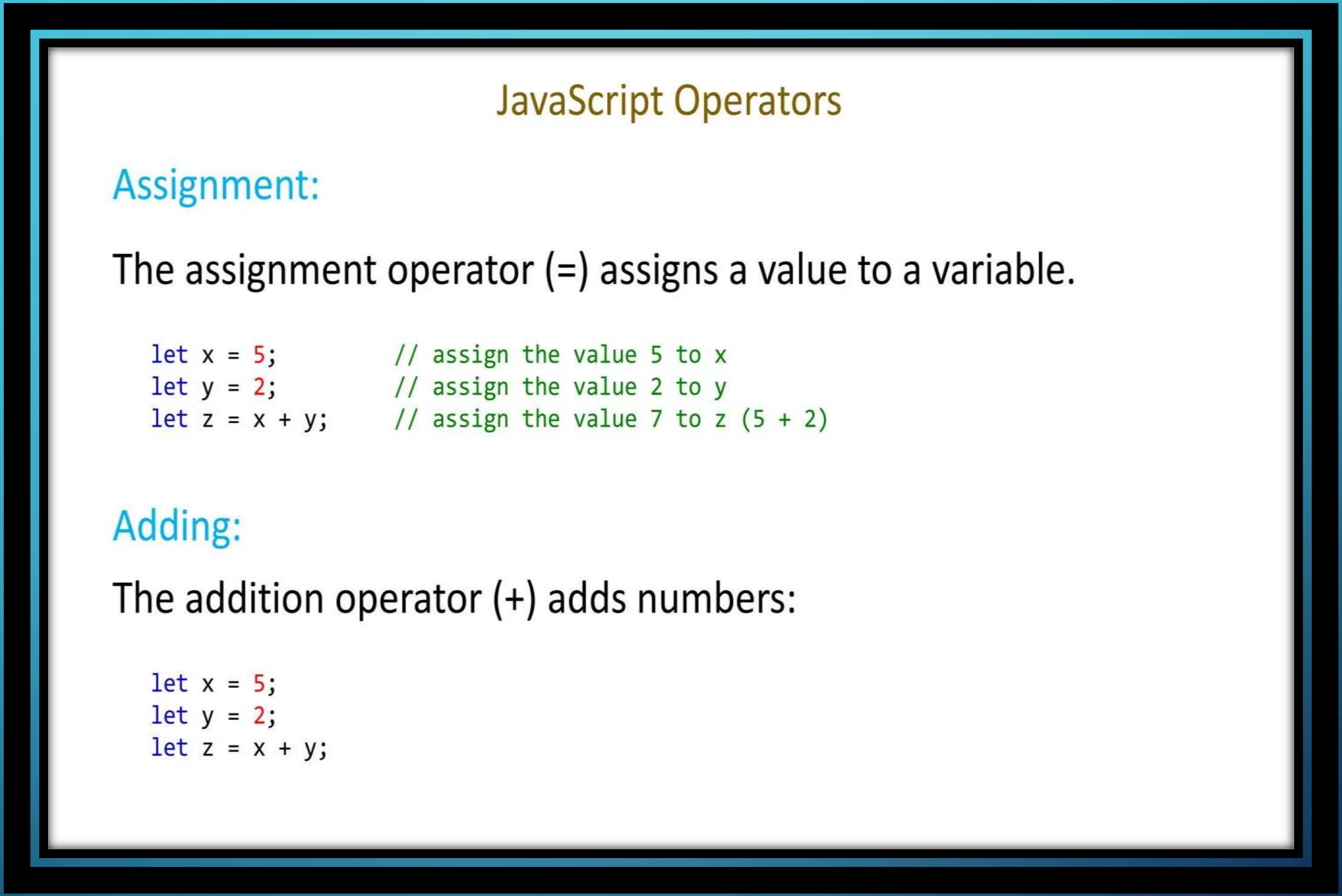
Use comments to describe the correct data type of the following variables:

```
let length = 16;                    //
let lastName = "Johnson";           //
const x = { firstName: "John", lastName: "Doe" }; //
```

WEEK 7

Page
39-42

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Operators

Assignment:

The assignment operator (=) assigns a value to a variable.

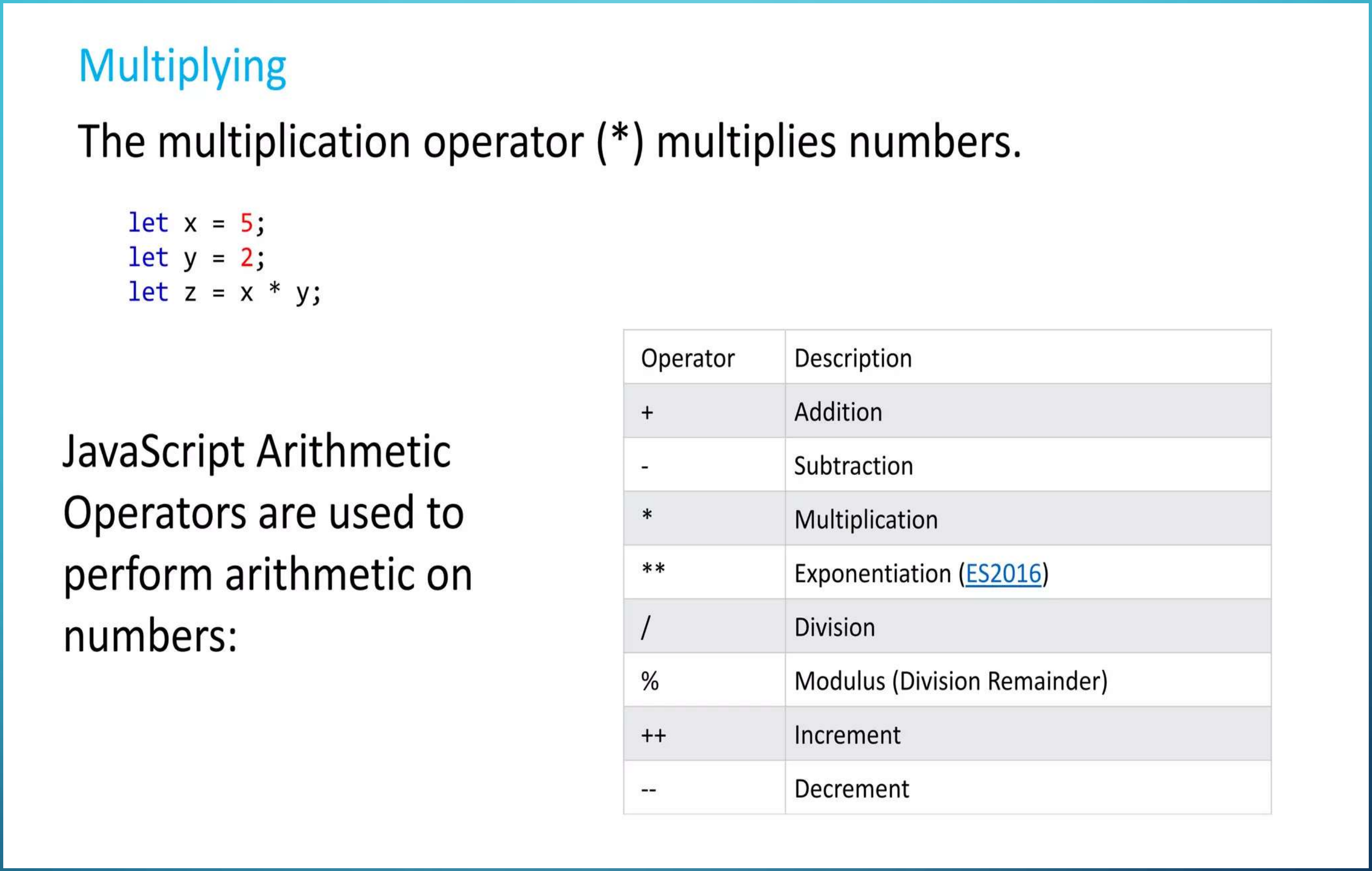
```
let x = 5;         // assign the value 5 to x
let y = 2;         // assign the value 2 to y
let z = x + y;     // assign the value 7 to z (5 + 2)
```

Adding:

The addition operator (+) adds numbers:

```
let x = 5;
let y = 2;
let z = x + y;
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



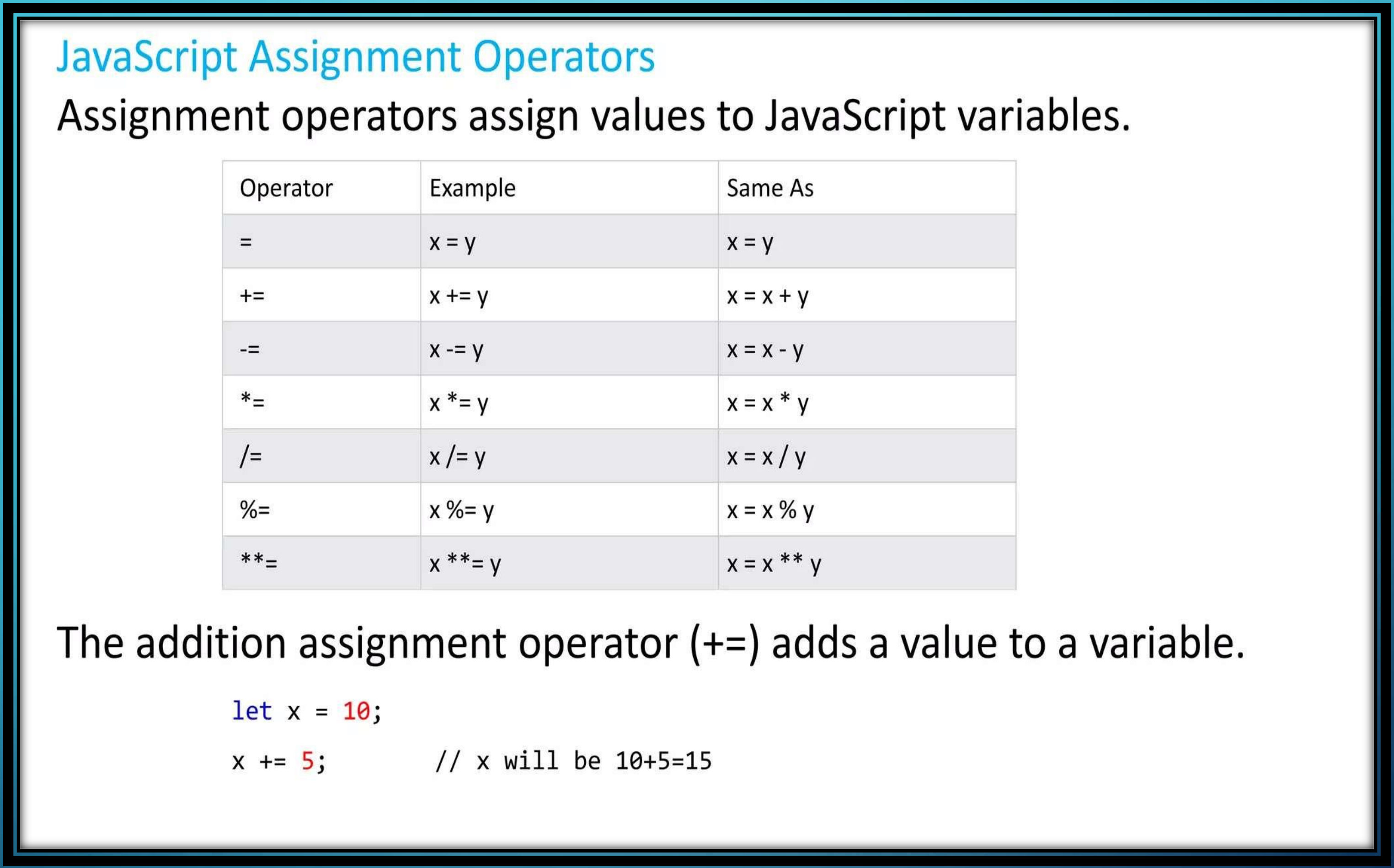
Multiplying

The multiplication operator (*) multiplies numbers.

```
let x = 5;
let y = 2;
let z = x * y;
```

JavaScript Arithmetic Operators are used to perform arithmetic on numbers:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement



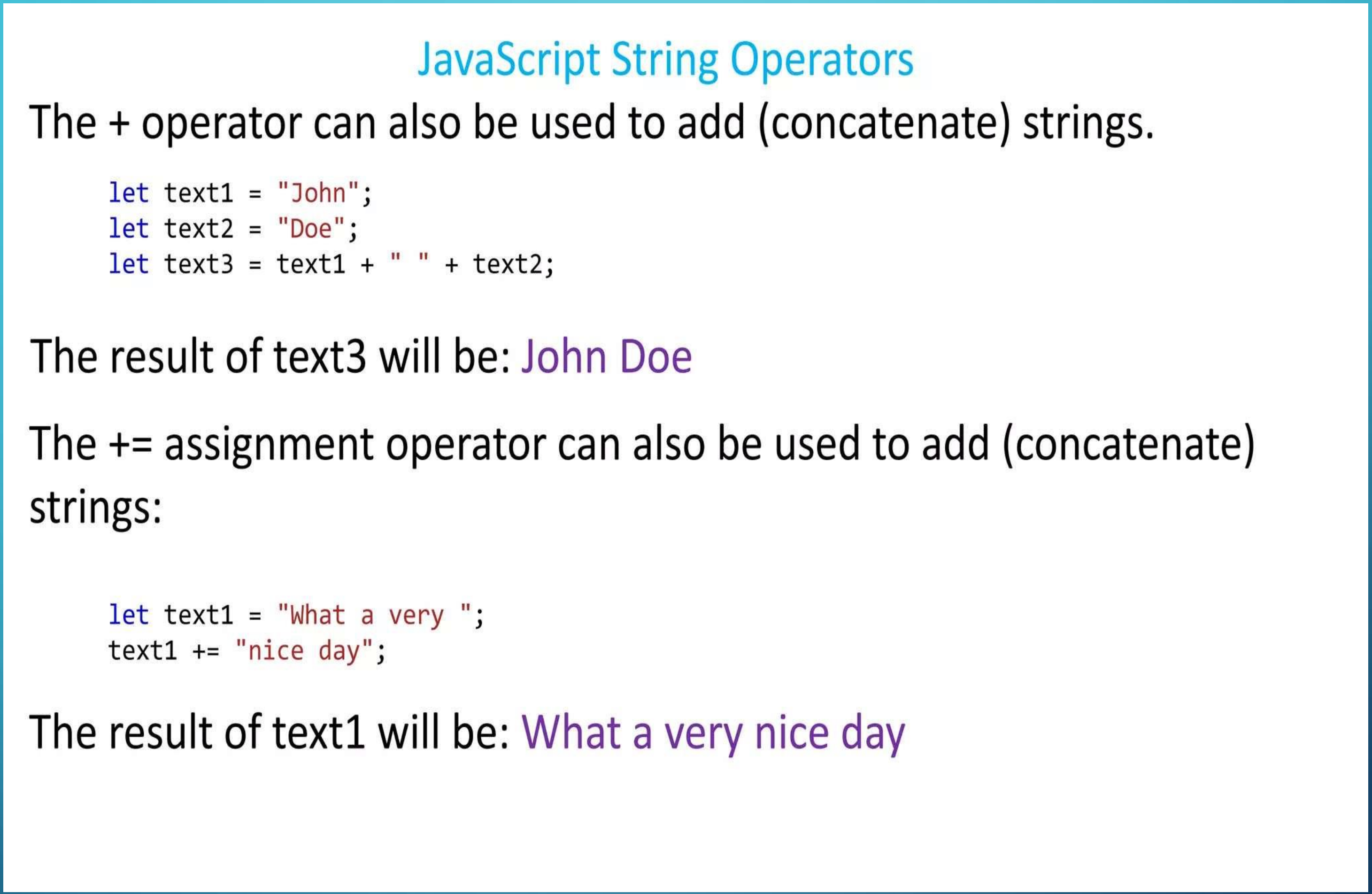
JavaScript Assignment Operators

Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

The addition assignment operator (+=) adds a value to a variable.

```
let x = 10;
x += 5;        // x will be 10+5=15
```



JavaScript String Operators

The + operator can also be used to add (concatenate) strings.

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

The result of text3 will be: John Doe

The += assignment operator can also be used to add (concatenate) strings:

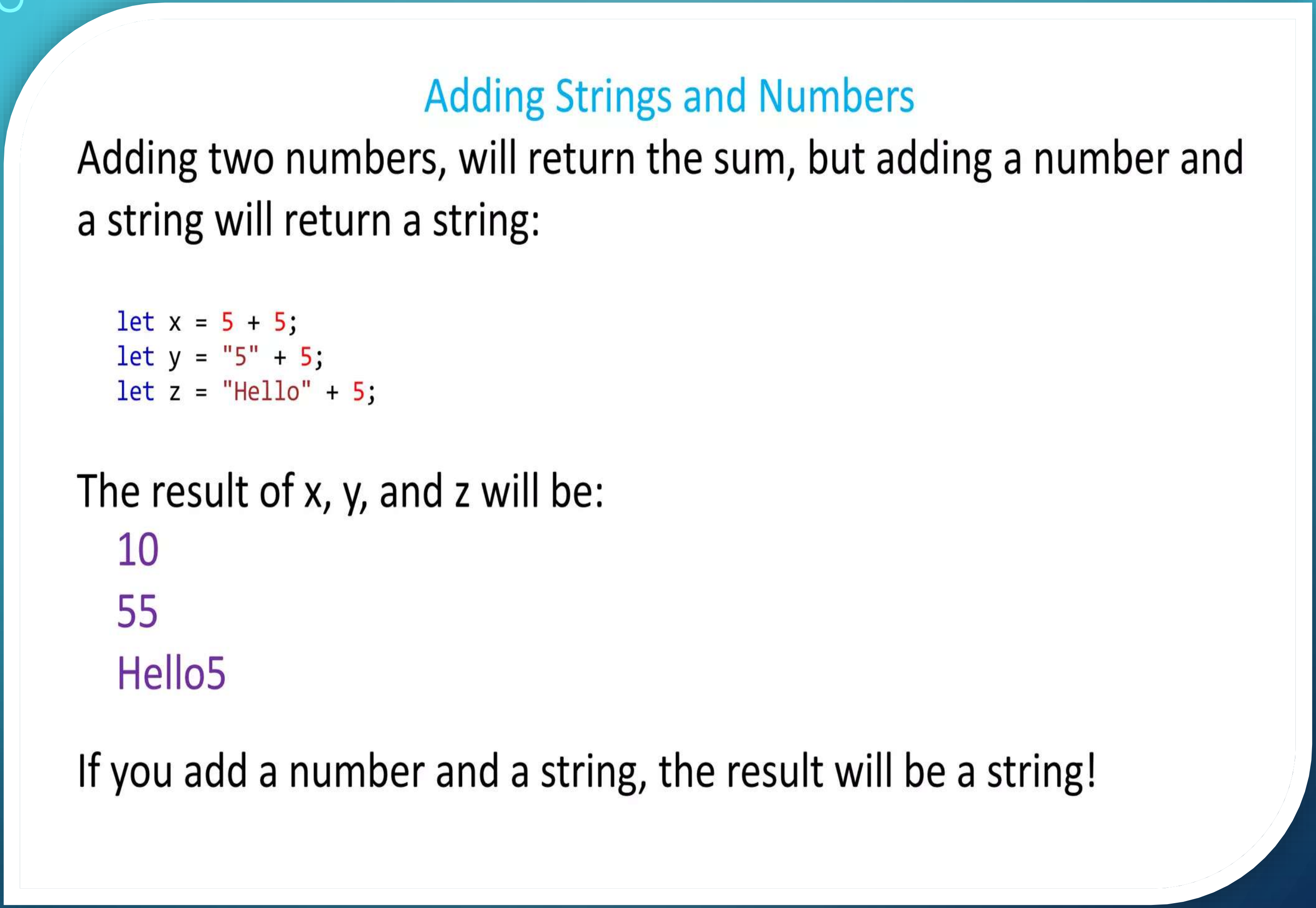
```
let text1 = "What a very ";
text1 += "nice day";
```

The result of text1 will be: What a very nice day

WEEK 8

Page 43 - 47

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



Adding Strings and Numbers

Adding two numbers, will return the sum, but adding a number and a string will return a string:

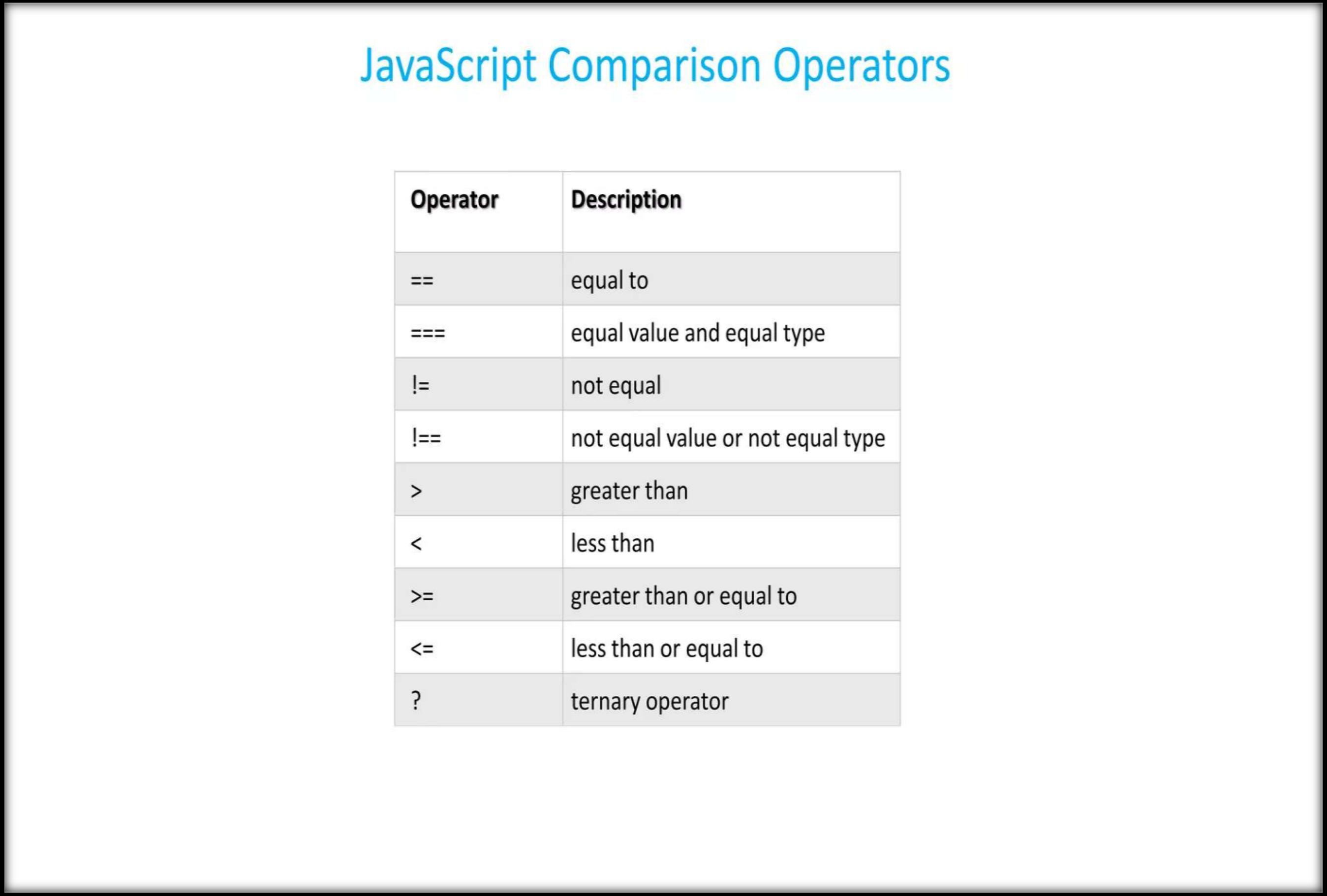
```
let x = 5 + 5;
let y = "5" + 5;
let z = "Hello" + 5;
```

The result of x, y, and z will be:

```
10
55
Hello5
```

If you add a number and a string, the result will be a string!

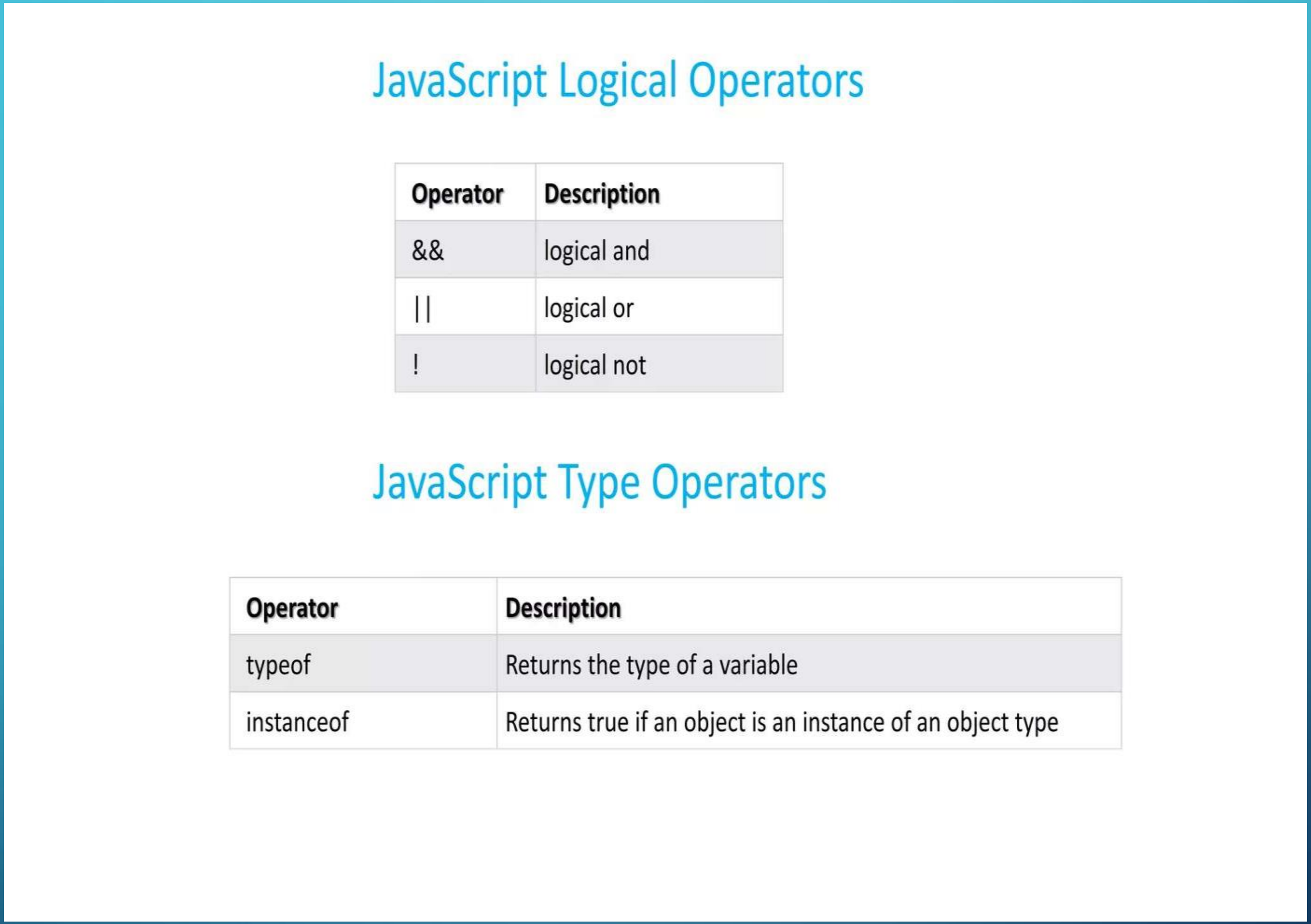
Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

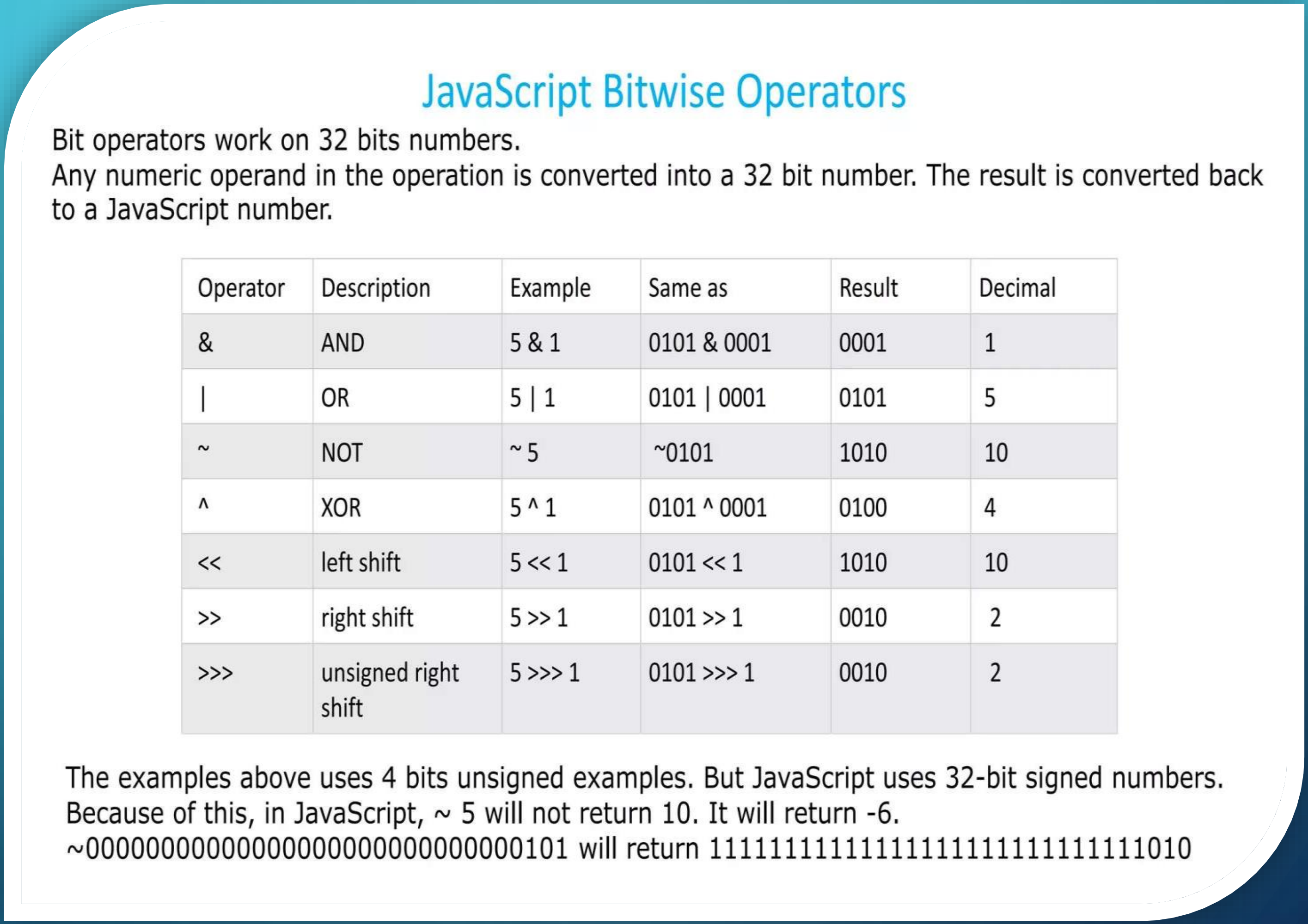


JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type



JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.
Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

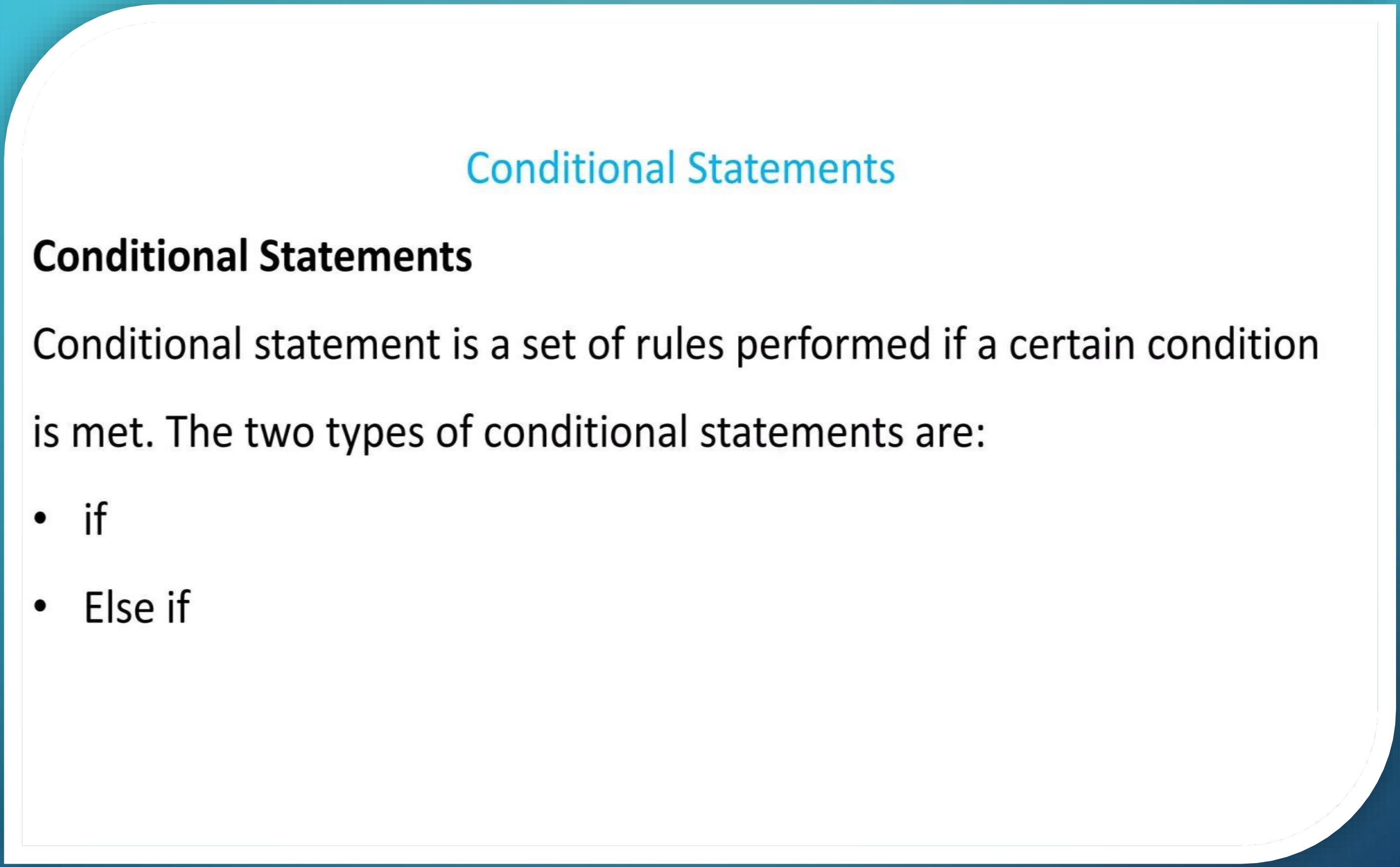
The examples above uses 4 bits unsigned examples. But JavaScript uses 32-bit signed numbers.
Because of this, in JavaScript, ~ 5 will not return 10. It will return -6.
~00000000000000000000000000000101 will return 11111111111111111111111111111010

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

WEEK 9

Page 48 - 54

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

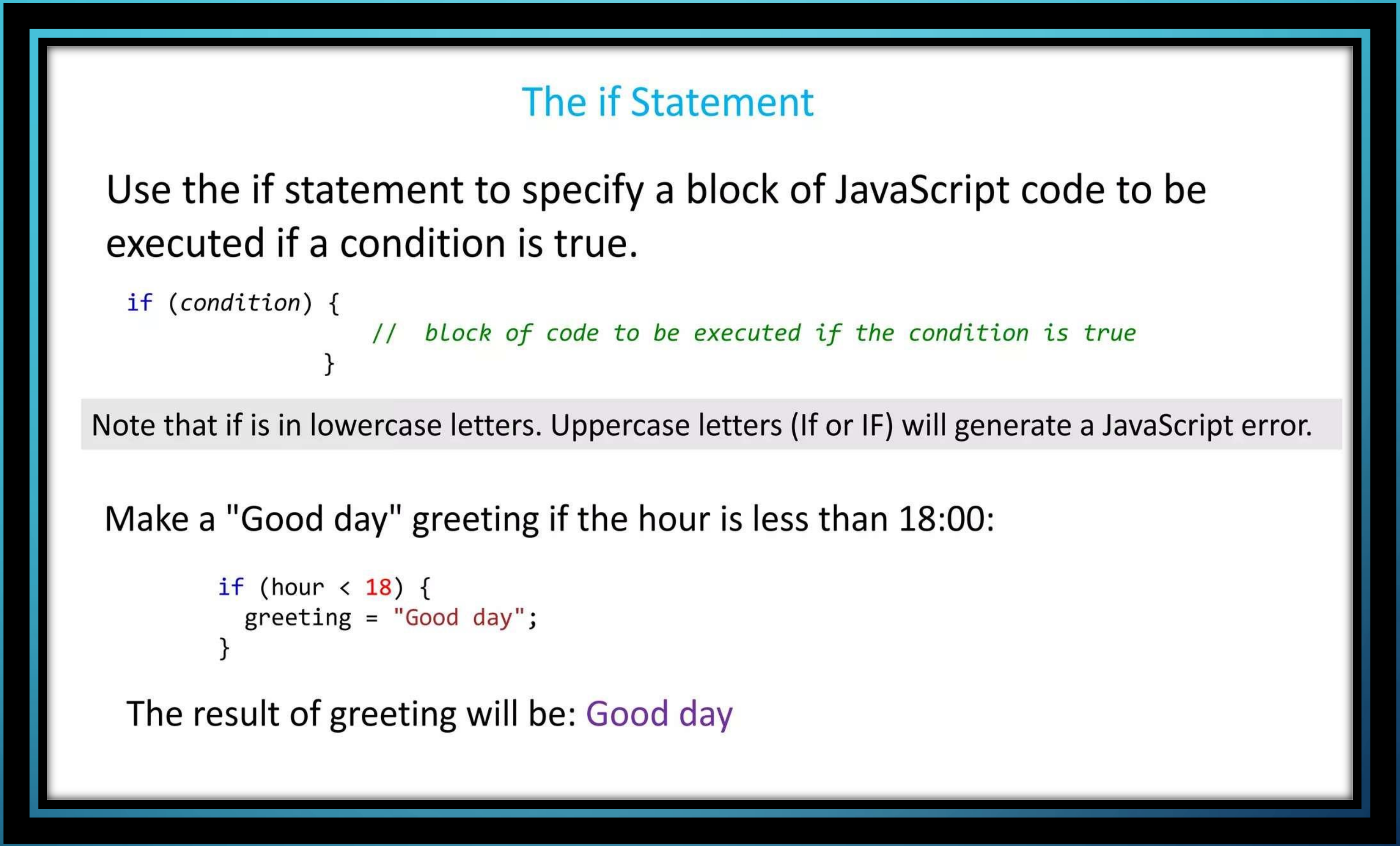


Conditional Statements

Conditional Statements

Conditional statement is a set of rules performed if a certain condition is met. The two types of conditional statements are:

- if
- Else if



The if Statement

Use the if statement to specify a block of JavaScript code to be executed if a condition is true.

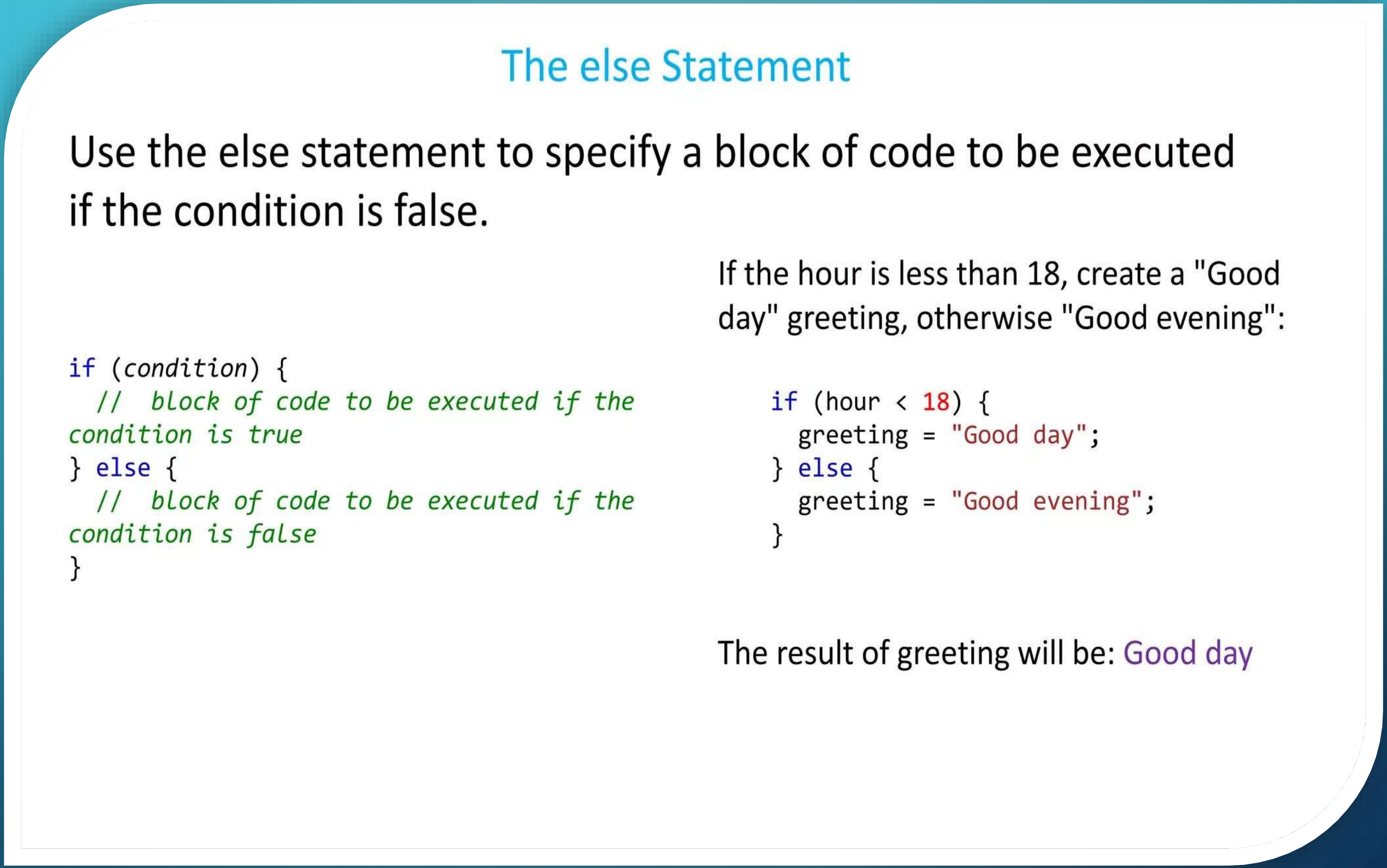
```
if (condition) {
                    //  block of code to be executed if the condition is true
               }
```

Note that if is in lowercase letters. Uppercase letters (If or IF) will generate a JavaScript error.

Make a "Good day" greeting if the hour is less than 18:00:

```
if (hour < 18) {
  greeting = "Good day";
}
```

The result of greeting will be: Good day



The else Statement

Use the else statement to specify a block of code to be executed if the condition is false.

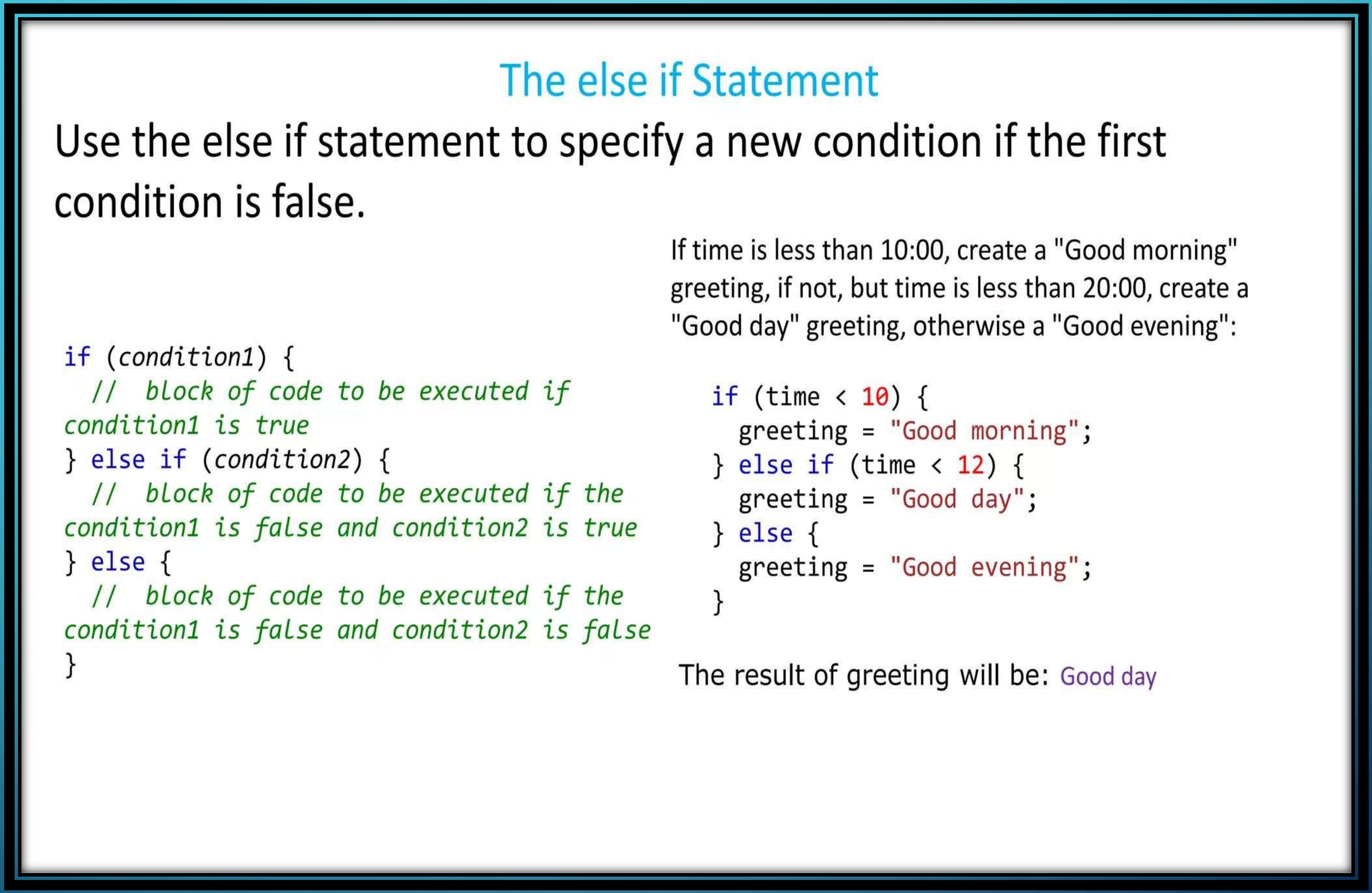
```
if (condition) {
  //  block of code to be executed if the
condition is true
} else {
  //  block of code to be executed if the
condition is false
}
```

If the hour is less than 18, create a "Good day" greeting, otherwise "Good evening":

```
if (hour < 18) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The result of greeting will be: Good day

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



The else if Statement

Use the else if statement to specify a new condition if the first condition is false.

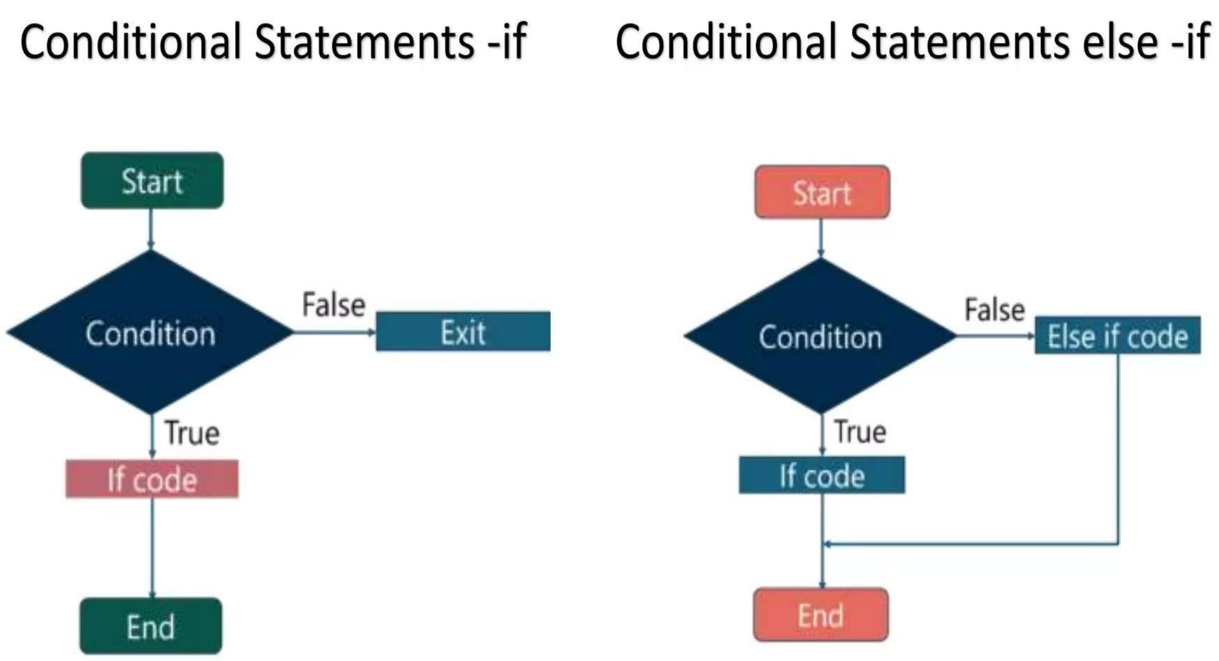
```
if (condition1) {
  //  block of code to be executed if
condition1 is true
} else if (condition2) {
  //  block of code to be executed if the
condition1 is false and condition2 is true
} else {
  //  block of code to be executed if the
condition1 is false and condition2 is false
}
```

If time is less than 10:00, create a "Good morning" greeting, if not, but time is less than 20:00, create a "Good day" greeting, otherwise a "Good evening":

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 12) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

The result of greeting will be: Good day

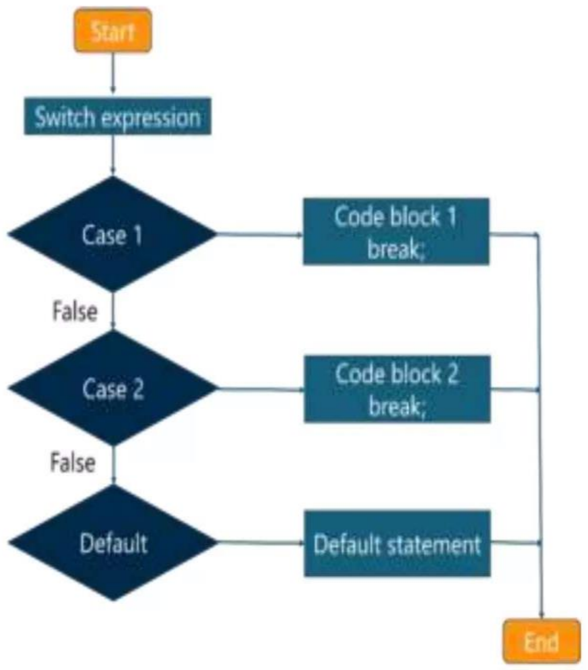
Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

Switch Case

The switch statement is used to perform different actions based on different conditions.

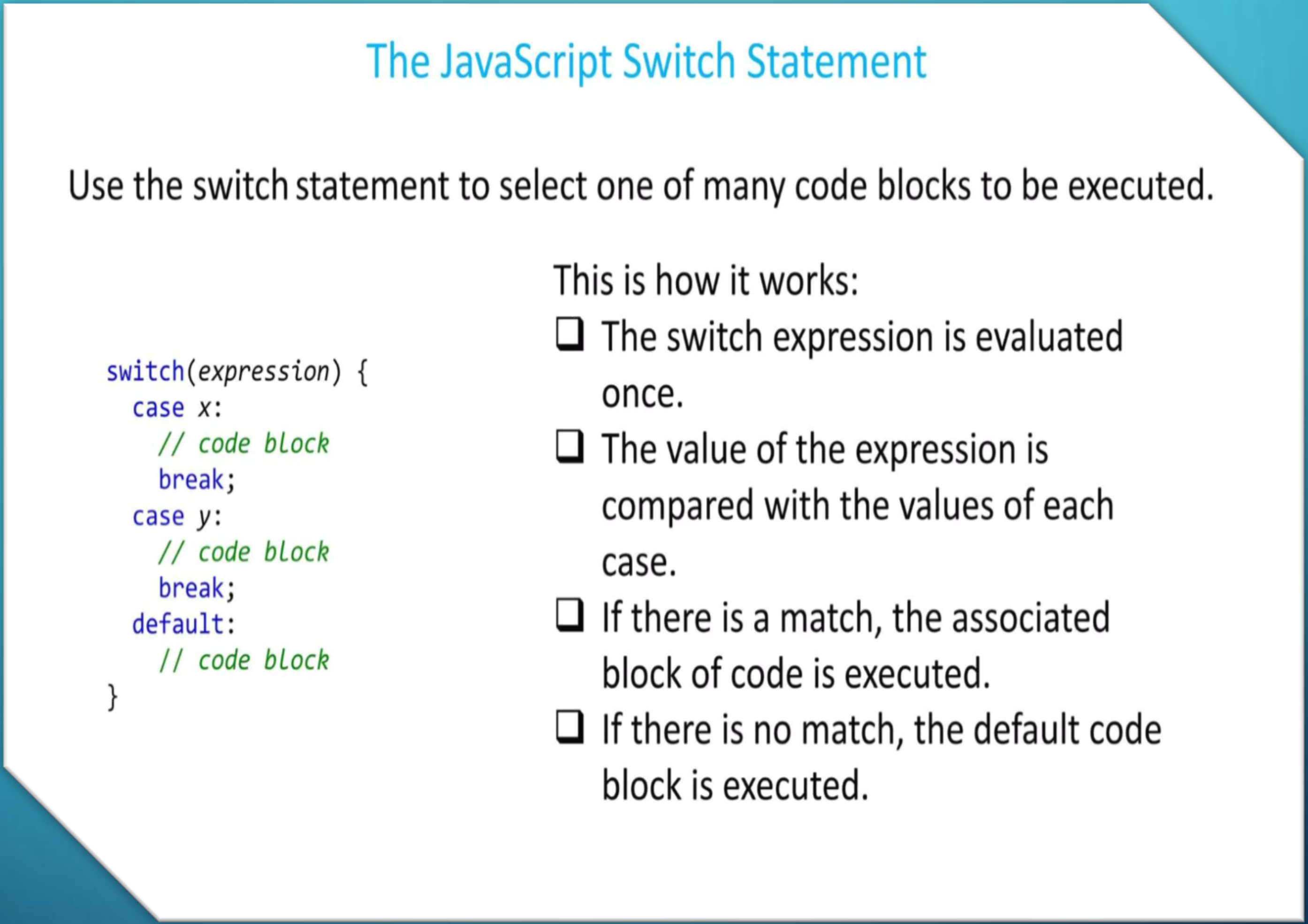


Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

WEEK 10 & 11

Page
56-65

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



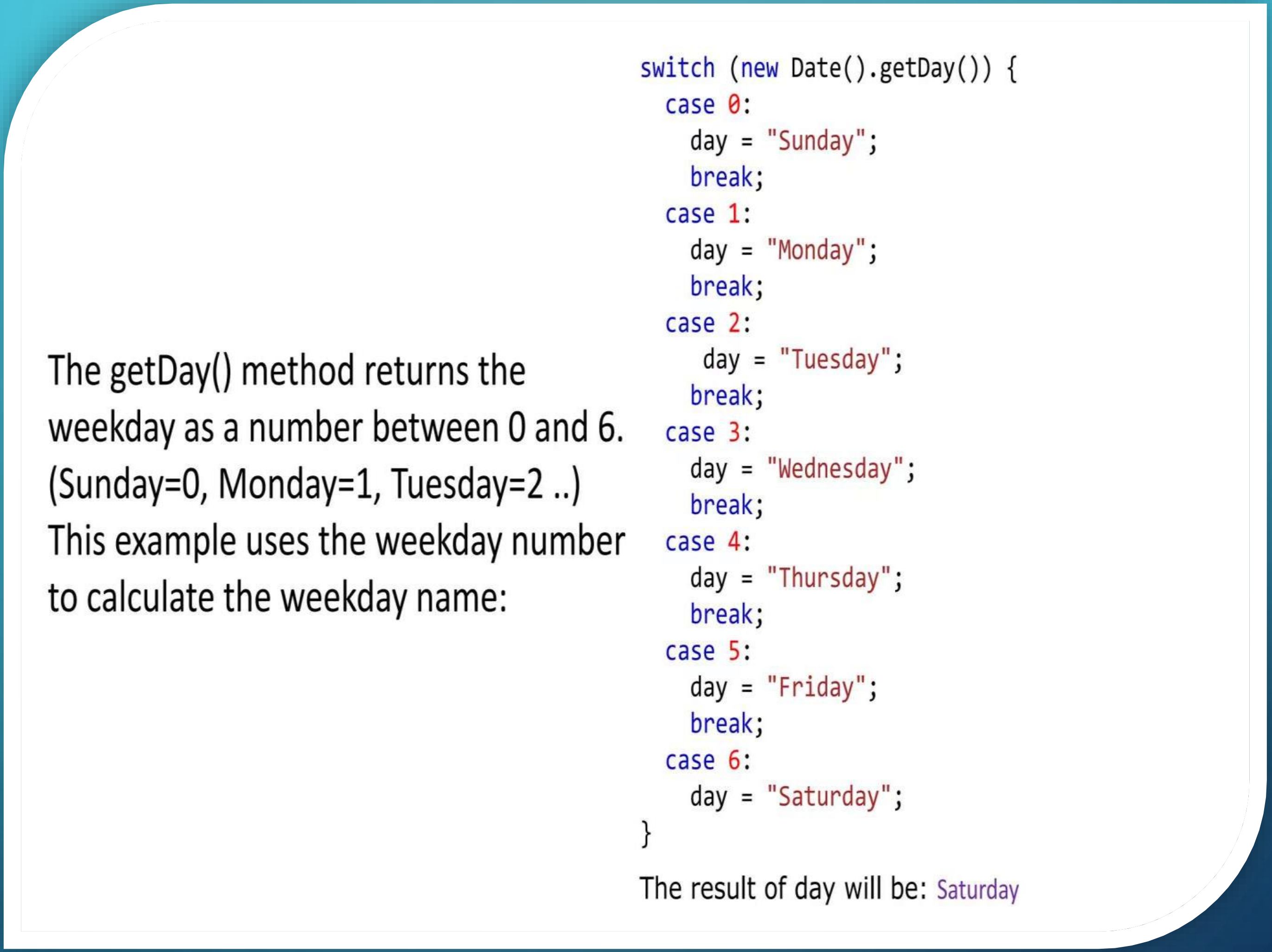
The JavaScript Switch Statement

Use the switch statement to select one of many code blocks to be executed.

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

This is how it works:

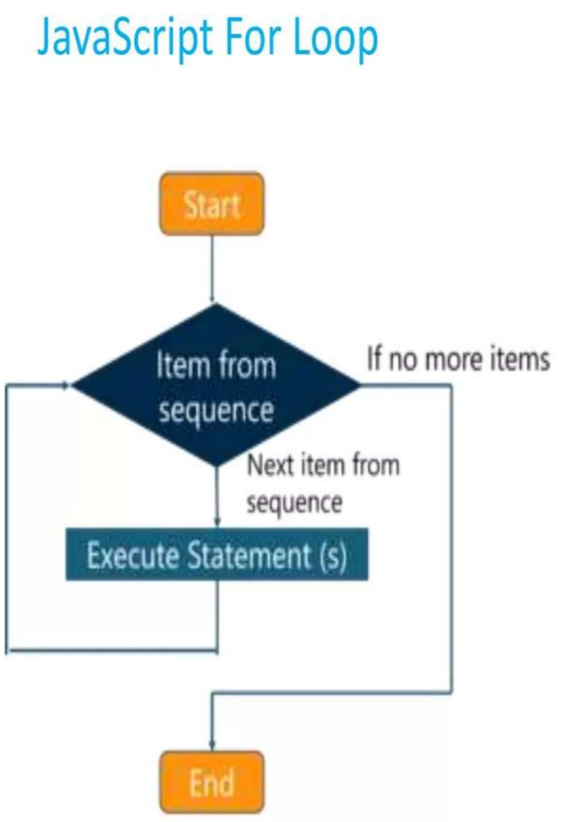
- ❑ The switch expression is evaluated once.
- ❑ The value of the expression is compared with the values of each case.
- ❑ If there is a match, the associated block of code is executed.
- ❑ If there is no match, the default code block is executed.

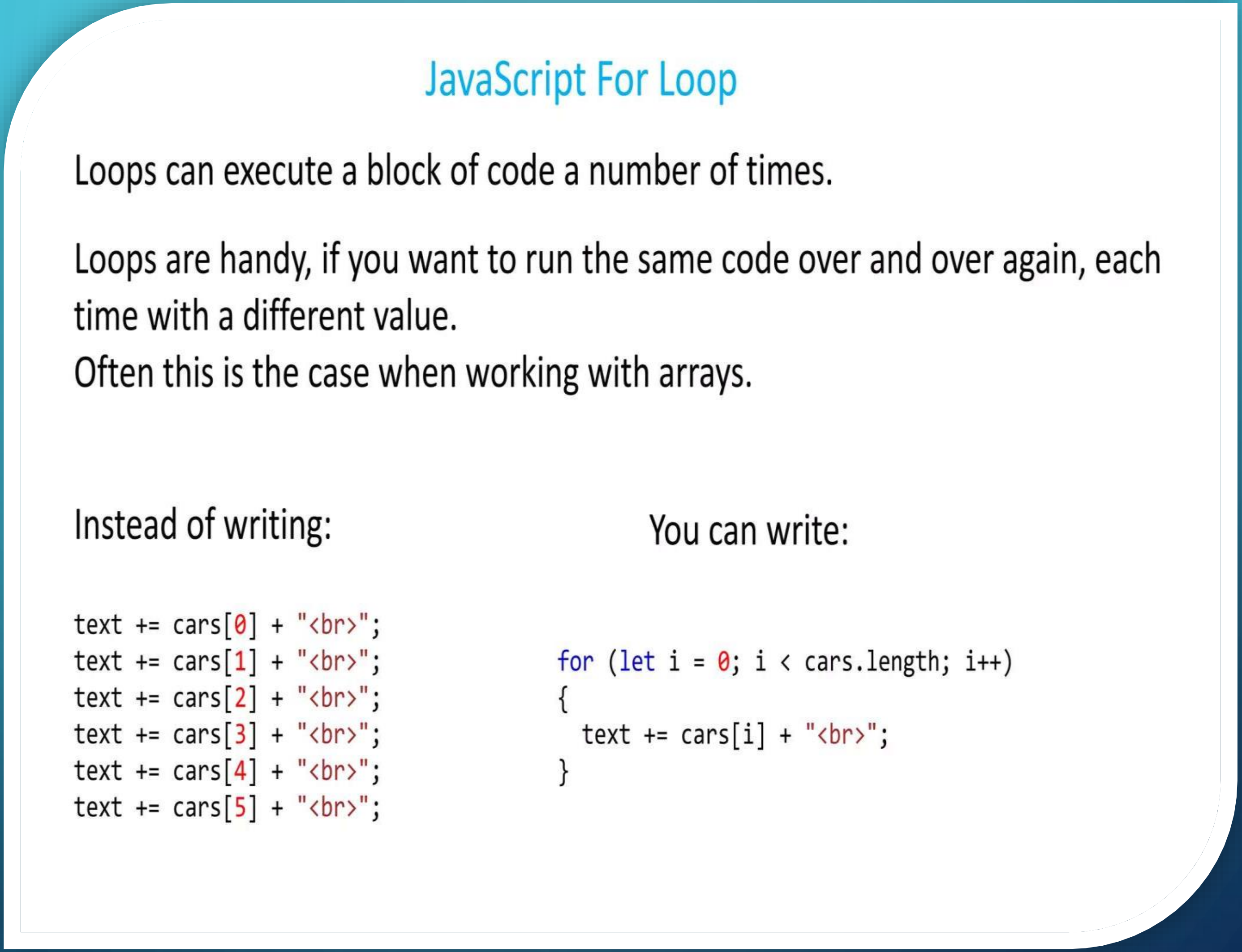


The getDay() method returns the weekday as a number between 0 and 6. (Sunday=0, Monday=1, Tuesday=2 ..) This example uses the weekday number to calculate the weekday name:

```
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
     day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
```

The result of day will be: Saturday


JavaScript For Loop
Start
Item from sequence
If no more items
Next item from sequence
Execute Statement (s)
End



JavaScript For Loop

Loops can execute a block of code a number of times.

Loops are handy, if you want to run the same code over and over again, each time with a different value.
Often this is the case when working with arrays.

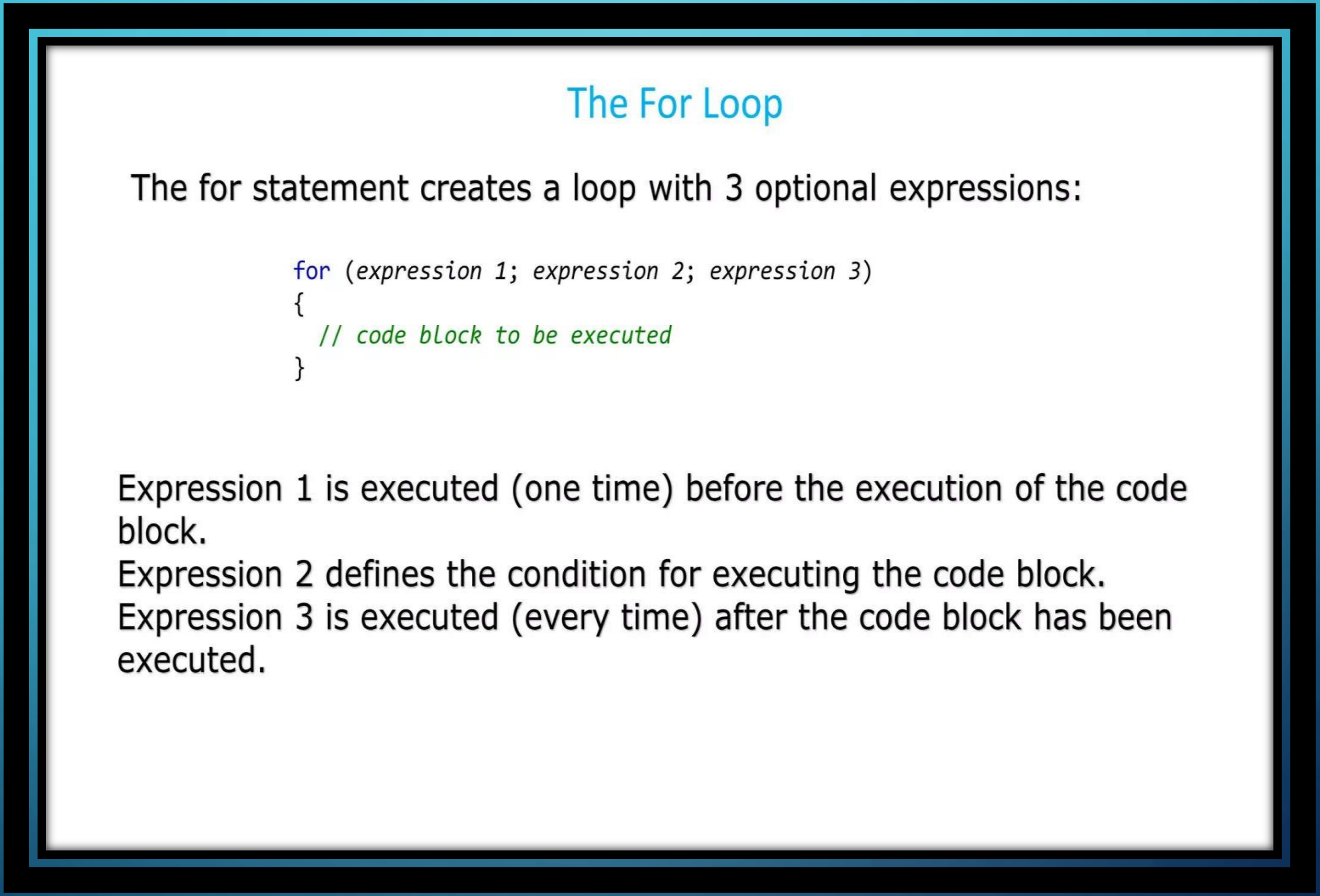
Instead of writing:

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

You can write:

```
for (let i = 0; i < cars.length; i++)
{
  text += cars[i] + "<br>";
}
```

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

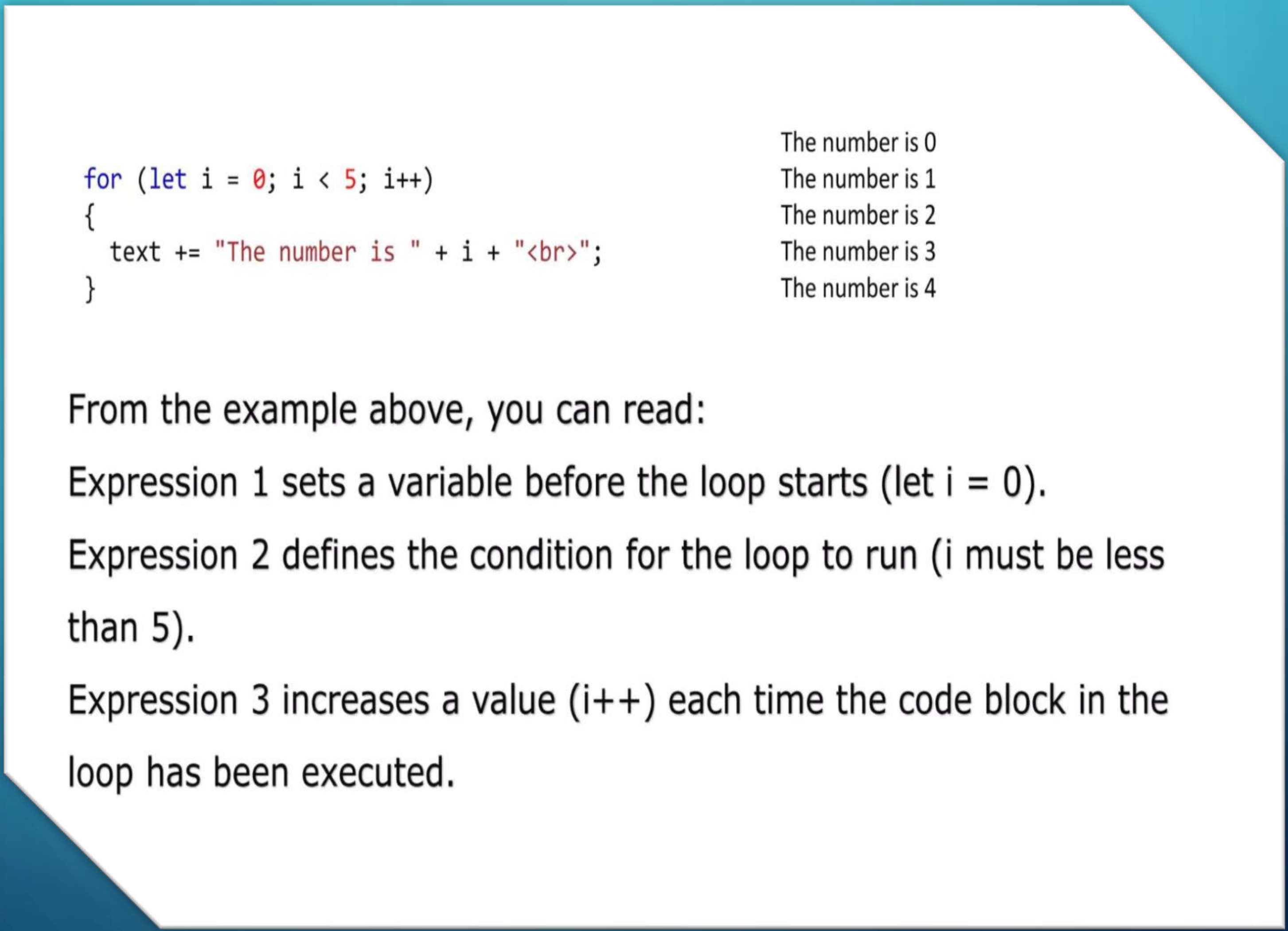


The For Loop

The for statement creates a loop with 3 optional expressions:

```
for (expression 1; expression 2; expression 3)
{
  // code block to be executed
}
```

Expression 1 is executed (one time) before the execution of the code block.
Expression 2 defines the condition for executing the code block.
Expression 3 is executed (every time) after the code block has been executed.



```
for (let i = 0; i < 5; i++)
{
  text += "The number is " + i + "<br>";
}
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

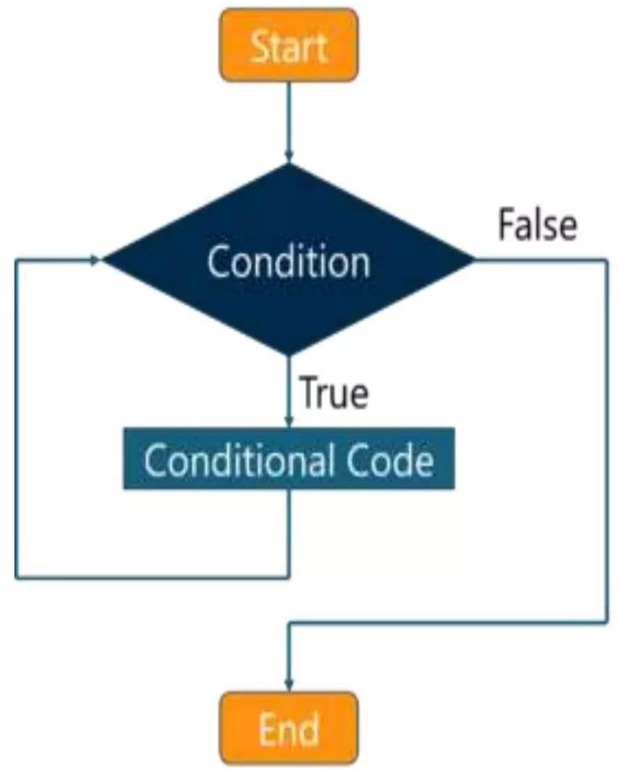
From the example above, you can read:

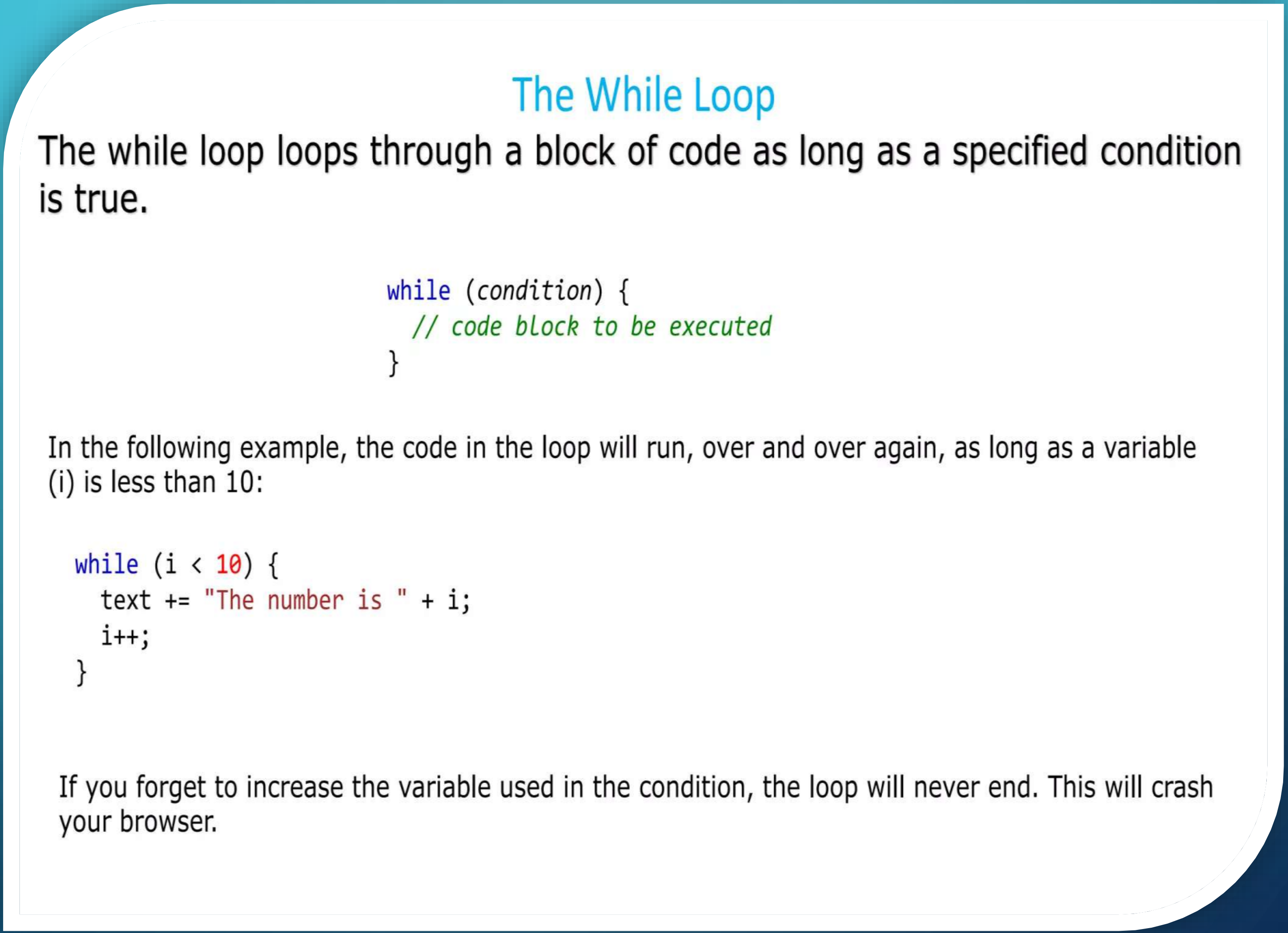
Expression 1 sets a variable before the loop starts (let i = 0).

Expression 2 defines the condition for the loop to run (i must be less than 5).

Expression 3 increases a value (i++) each time the code block in the loop has been executed.

The While Loop





The While Loop

The while loop loops through a block of code as long as a specified condition is true.

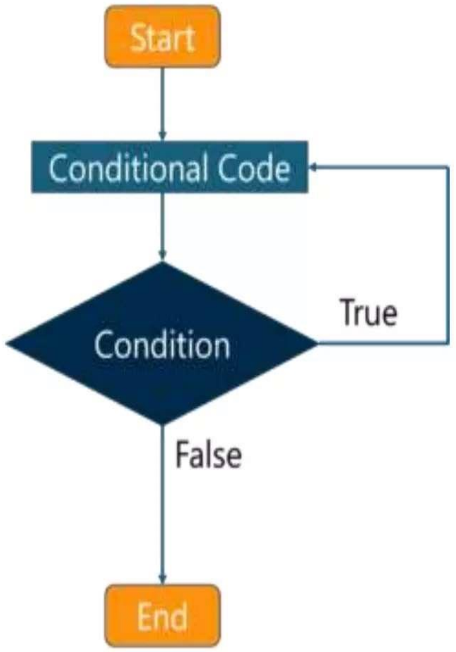
```
while (condition) {
  // code block to be executed
}
```

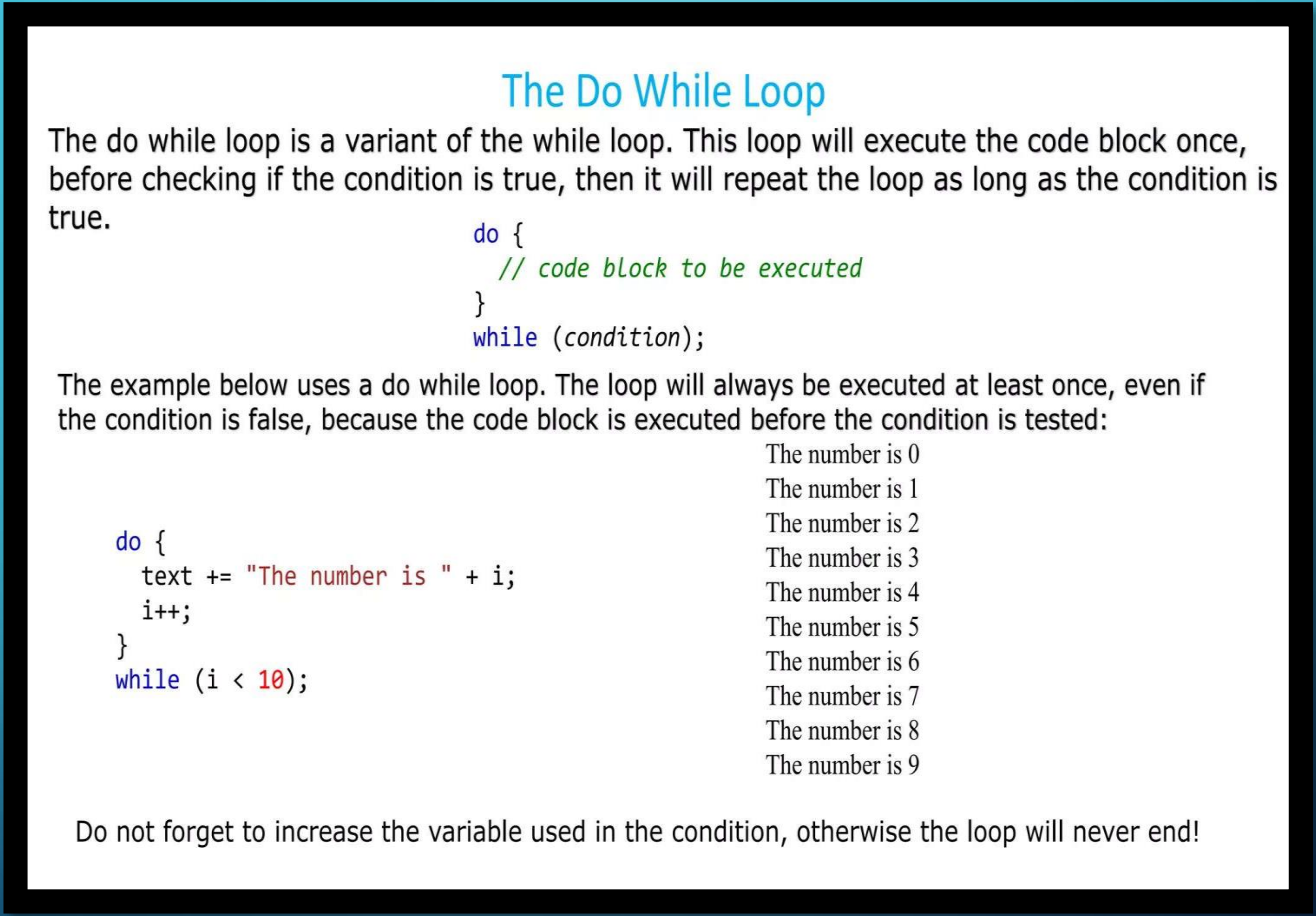
In the following example, the code in the loop will run, over and over again, as long as a variable (i) is less than 10:

```
while (i < 10) {
  text += "The number is " + i;
  i++;
}
```

If you forget to increase the variable used in the condition, the loop will never end. This will crash your browser.

The Do While Loop





The Do While Loop

The do while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
do {
  // code block to be executed
}
while (condition);
```

The example below uses a do while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

```
do {
  text += "The number is " + i;
  i++;
}
while (i < 10);
```

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

Do not forget to increase the variable used in the condition, otherwise the loop will never end!

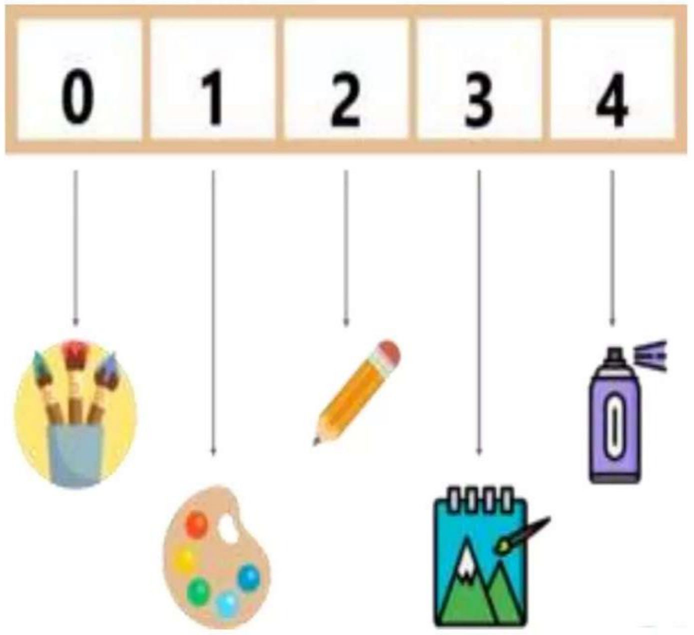
WEEK 12 & 13

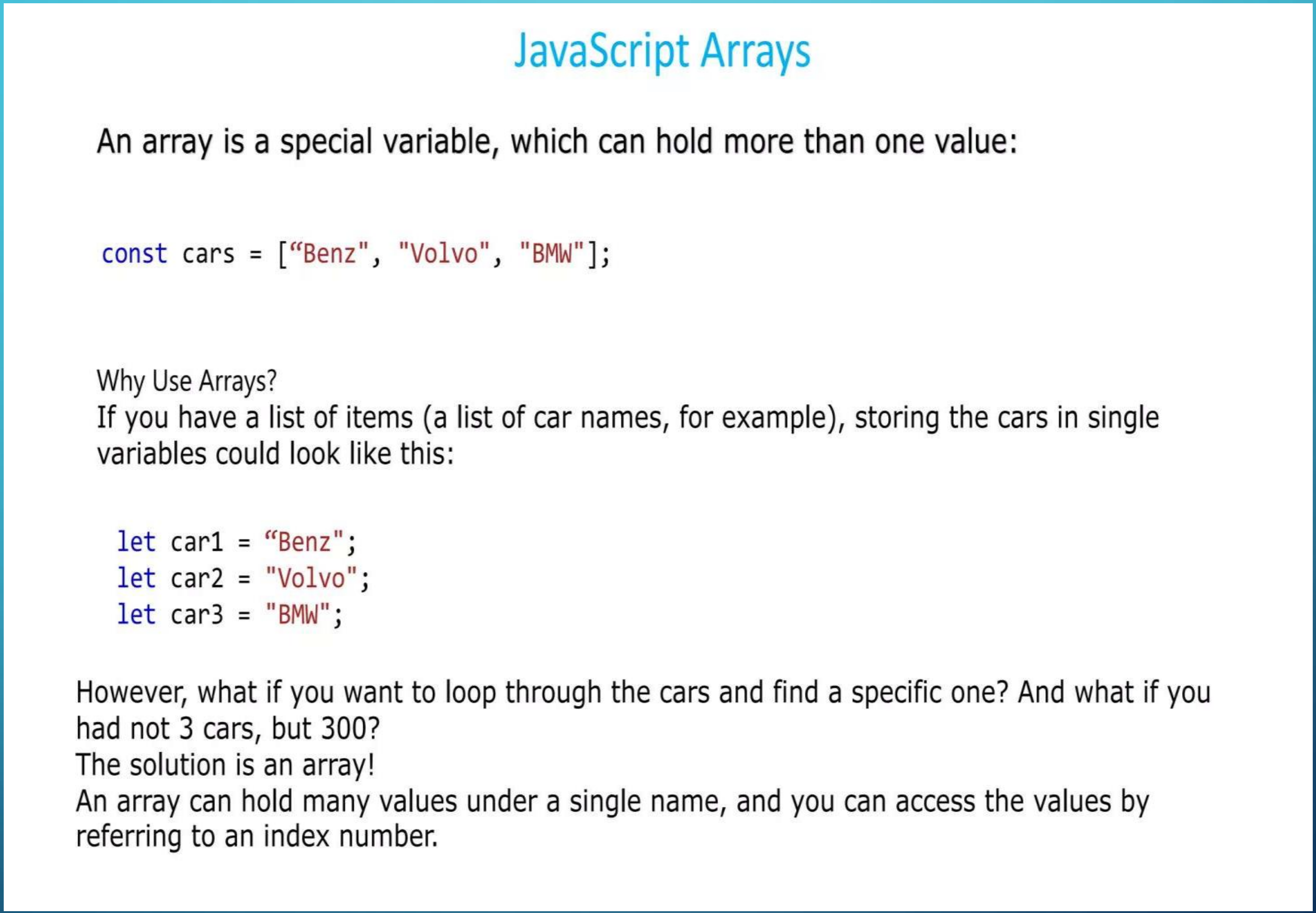
Page
67-71

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

Arrays

An array is a data structure that contains a list of elements which store multiple values in a single variable.





JavaScript Arrays

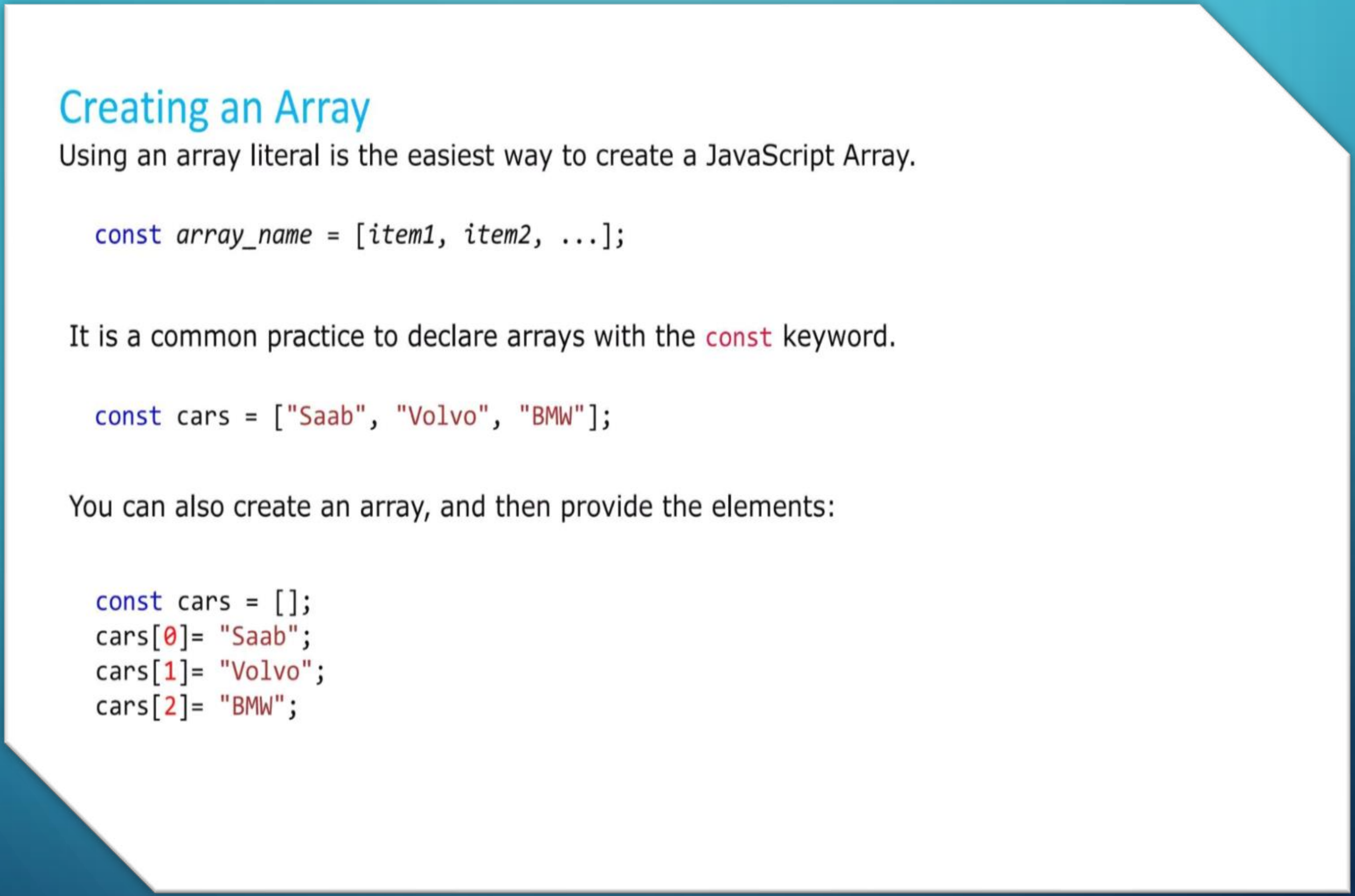
An array is a special variable, which can hold more than one value:

```
const cars = [“Benz", "Volvo", "BMW"];
```

Why Use Arrays?
If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
 let car1 = “Benz";
 let car2 = "Volvo";
 let car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?
The solution is an array!
An array can hold many values under a single name, and you can access the values by referring to an index number.



Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

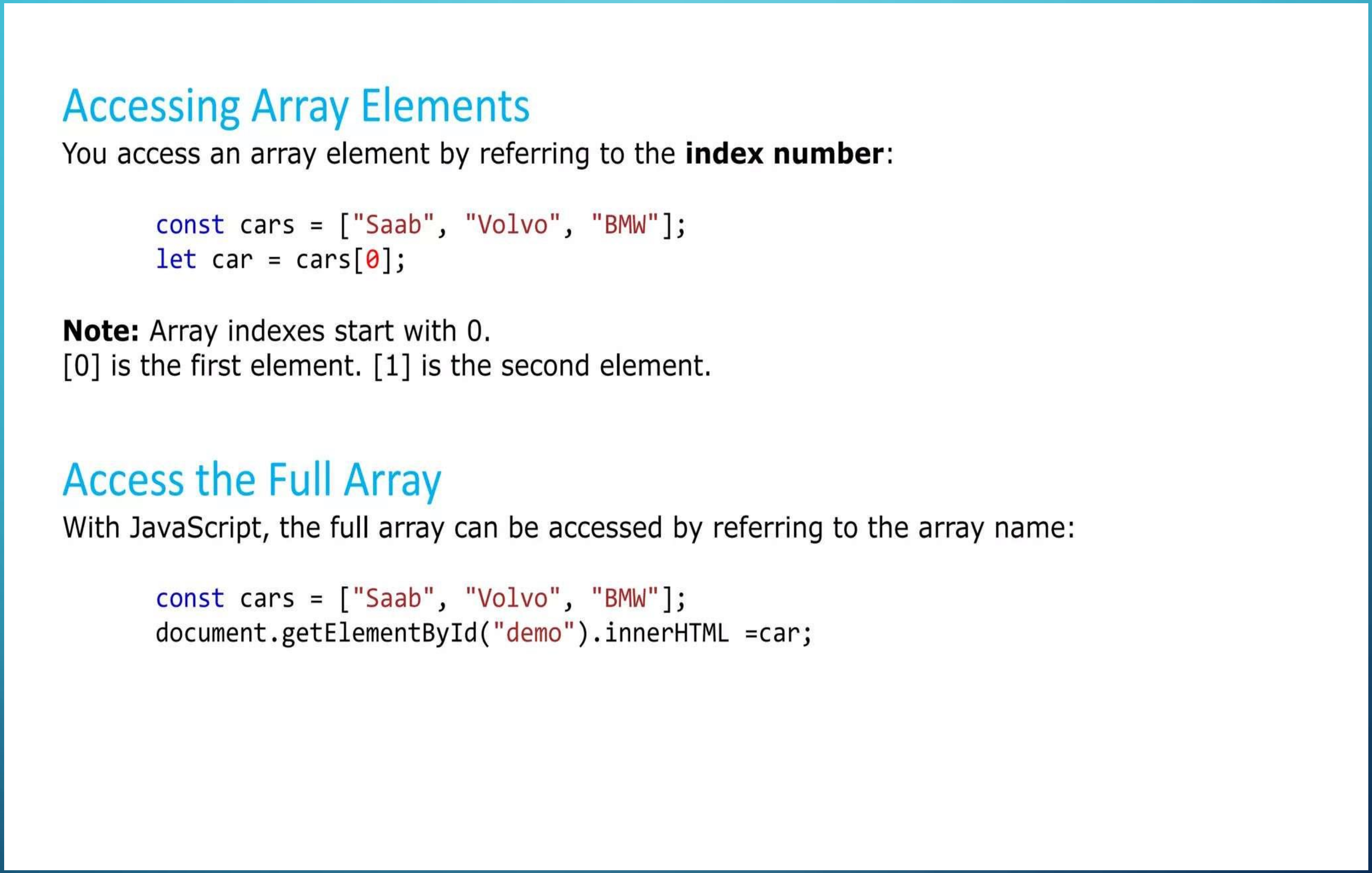
```
const array_name = [item1, item2, ...];
```

It is a common practice to declare arrays with the `const` keyword.

```
const cars = ["Saab", "Volvo", "BMW"];
```

You can also create an array, and then provide the elements:

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```



Accessing Array Elements

You access an array element by referring to the **index number**:

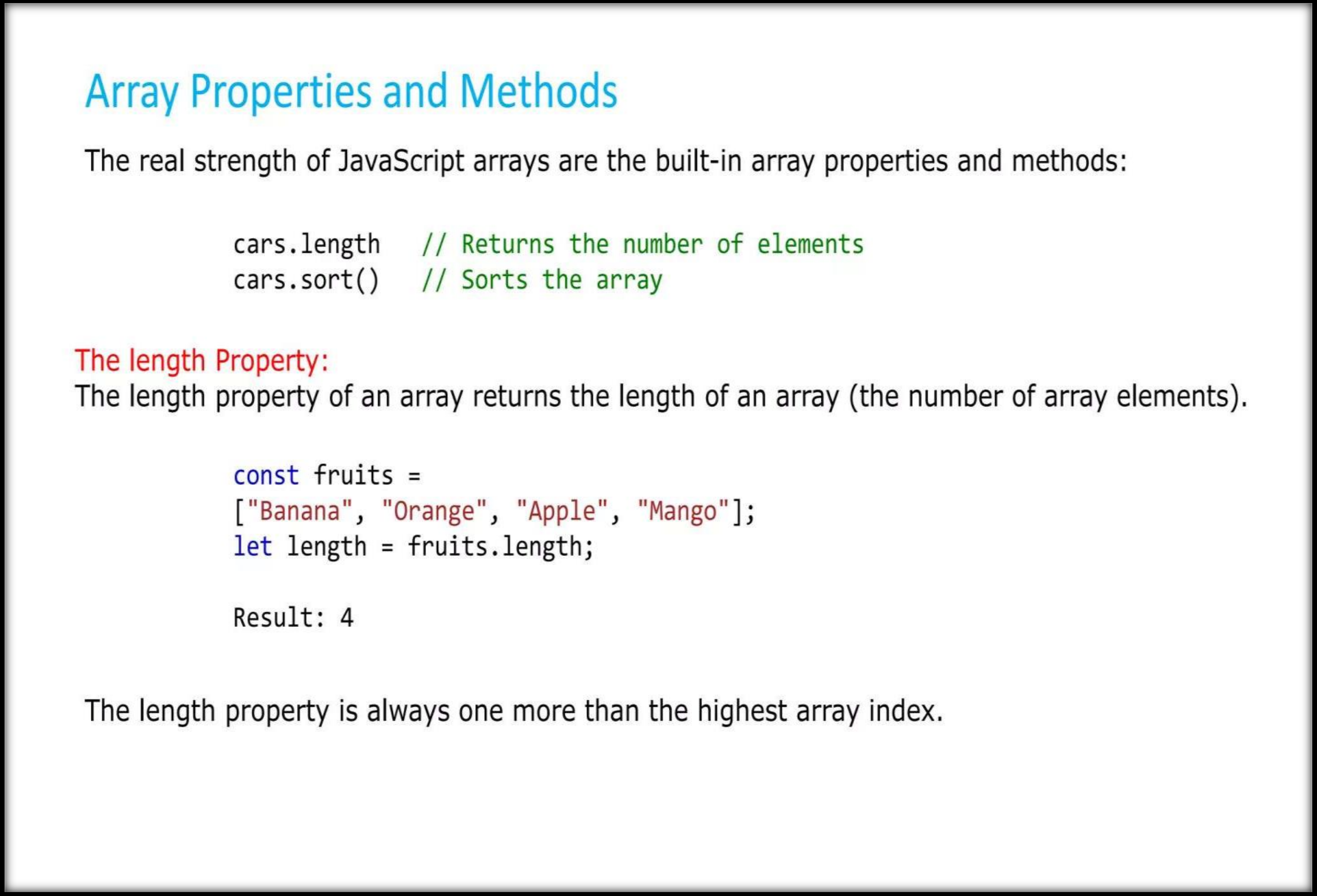
```
const cars = ["Saab", "Volvo", "BMW"];
let car = cars[0];
```

Note: Array indexes start with 0.
[0] is the first element. [1] is the second element.

Access the Full Array

With JavaScript, the full array can be accessed by referring to the array name:

```
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML =car;
```



Array Properties and Methods

The real strength of JavaScript arrays are the built-in array properties and methods:

```
cars.length   // Returns the number of elements
cars.sort()   // Sorts the array
```

The length Property:

The length property of an array returns the length of an array (the number of array elements).

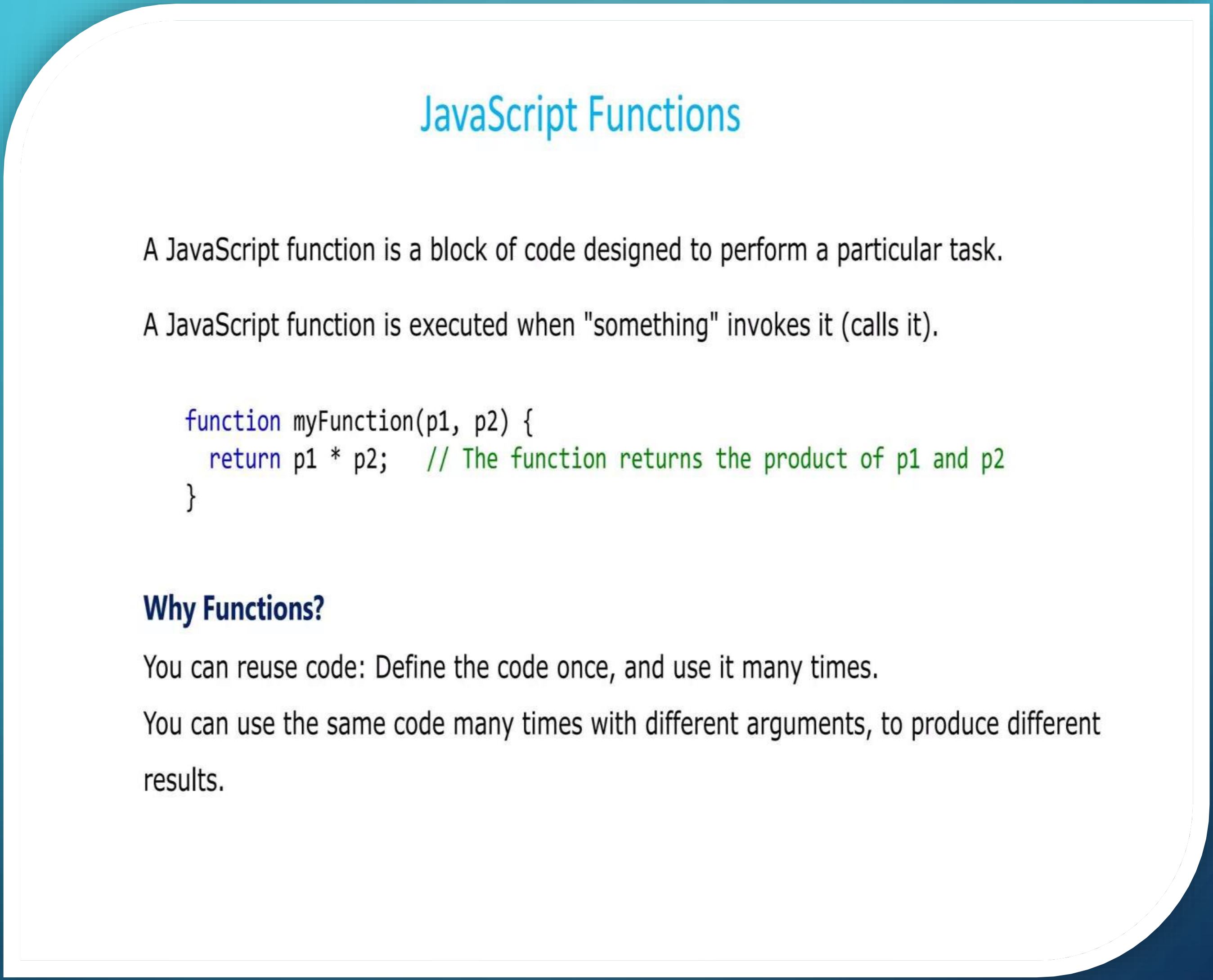
```
const fruits =
["Banana", "Orange", "Apple", "Mango"];
let length = fruits.length;

Result: 4
```

The length property is always one more than the highest array index.

WEEK 15

Page
73-78

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Functions

A JavaScript function is a block of code designed to perform a particular task.

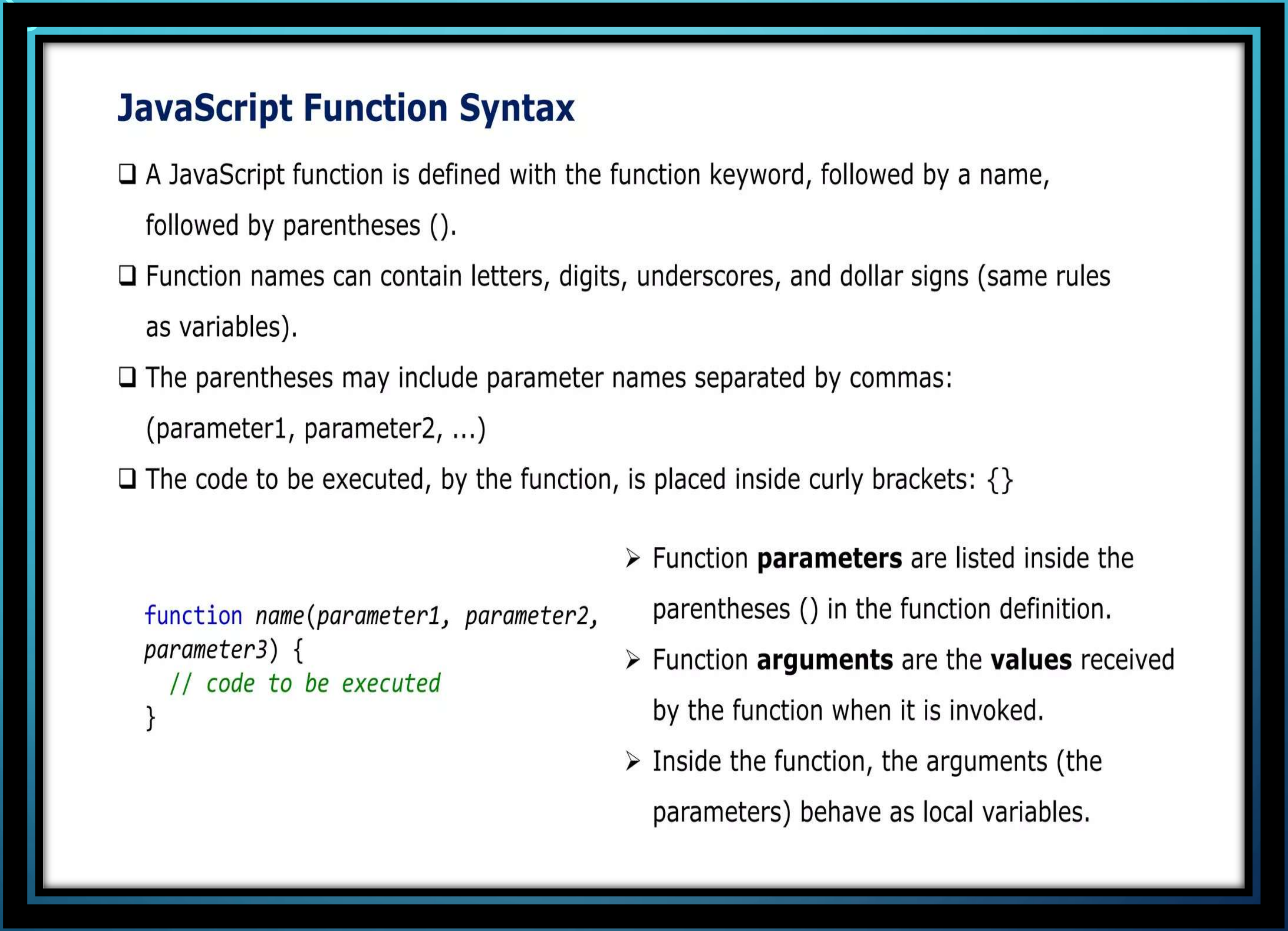
A JavaScript function is executed when "something" invokes it (calls it).

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}
```

Why Functions?

You can reuse code: Define the code once, and use it many times.

You can use the same code many times with different arguments, to produce different results.

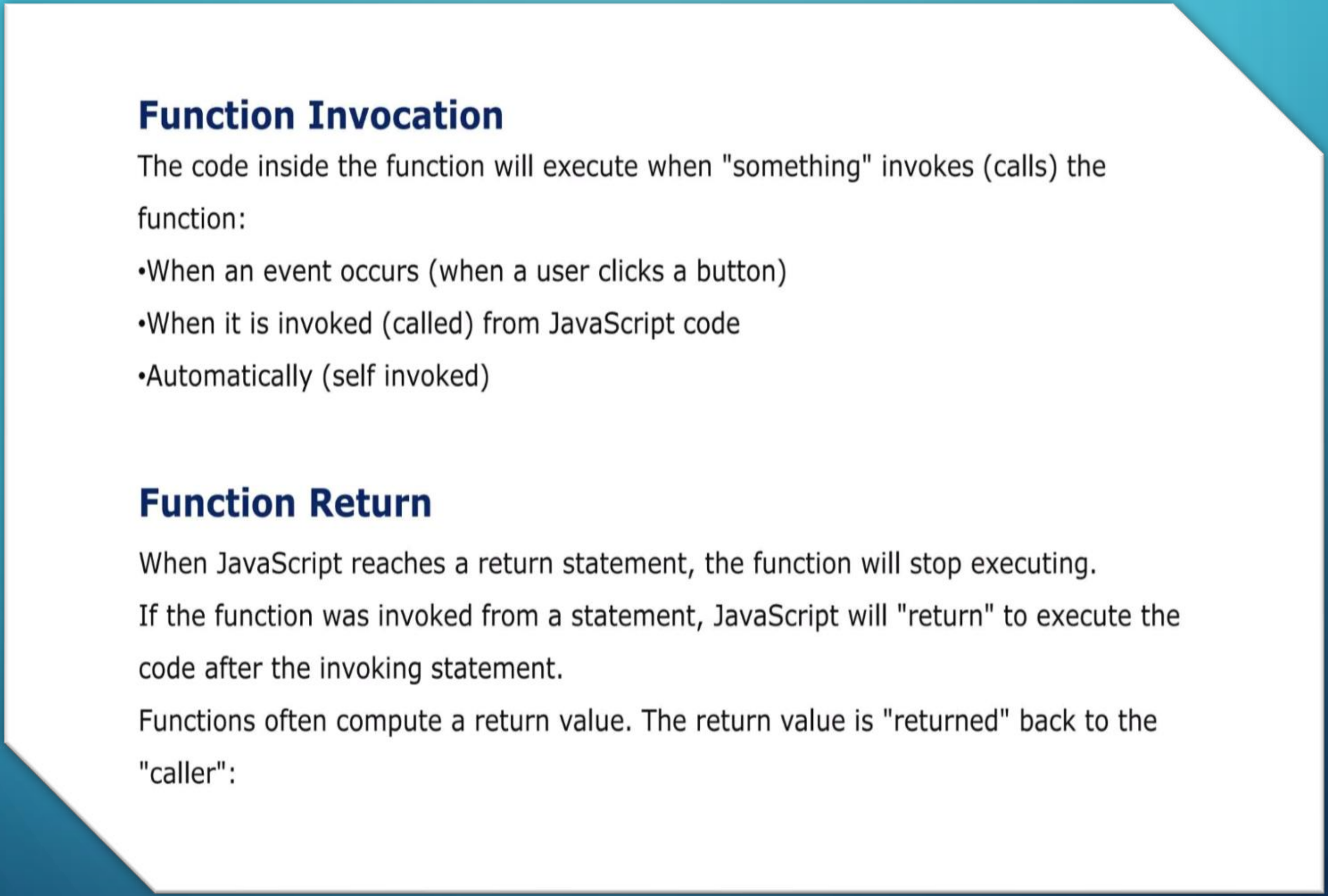


JavaScript Function Syntax

- ❑ A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- ❑ Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).
- ❑ The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)
- ❑ The code to be executed, by the function, is placed inside curly brackets: {}

```
function name(parameter1, parameter2,
parameter3) {
  // code to be executed
}
```

- ➢ Function **parameters** are listed inside the parentheses () in the function definition.
- ➢ Function **arguments** are the **values** received by the function when it is invoked.
- ➢ Inside the function, the arguments (the parameters) behave as local variables.



Function Invocation

The code inside the function will execute when "something" invokes (calls) the function:

•When an event occurs (when a user clicks a button)

•When it is invoked (called) from JavaScript code

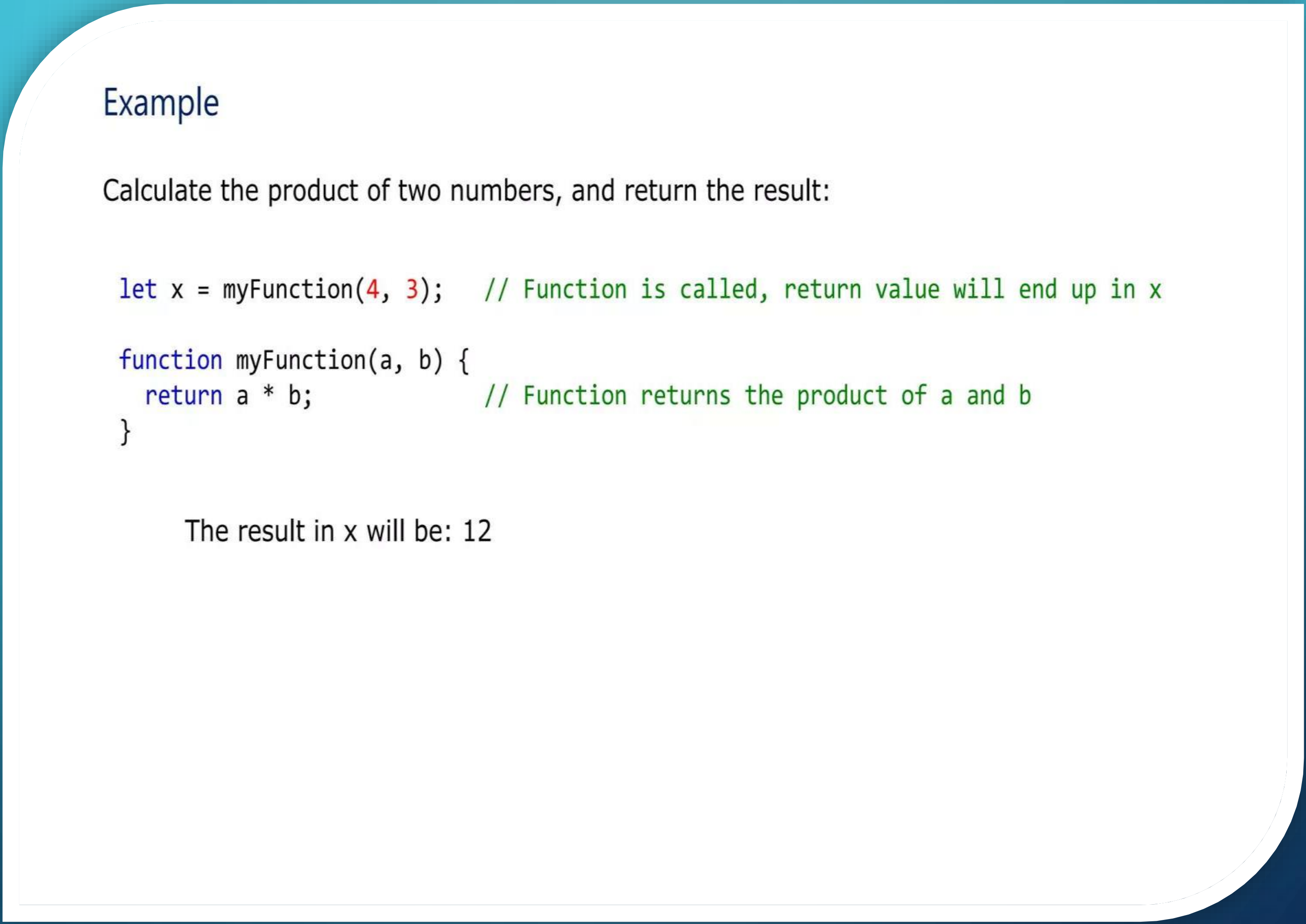
•Automatically (self invoked)

Function Return

When JavaScript reaches a return statement, the function will stop executing.

If the function was invoked from a statement, JavaScript will "return" to execute the code after the invoking statement.

Functions often compute a return value. The return value is "returned" back to the "caller":



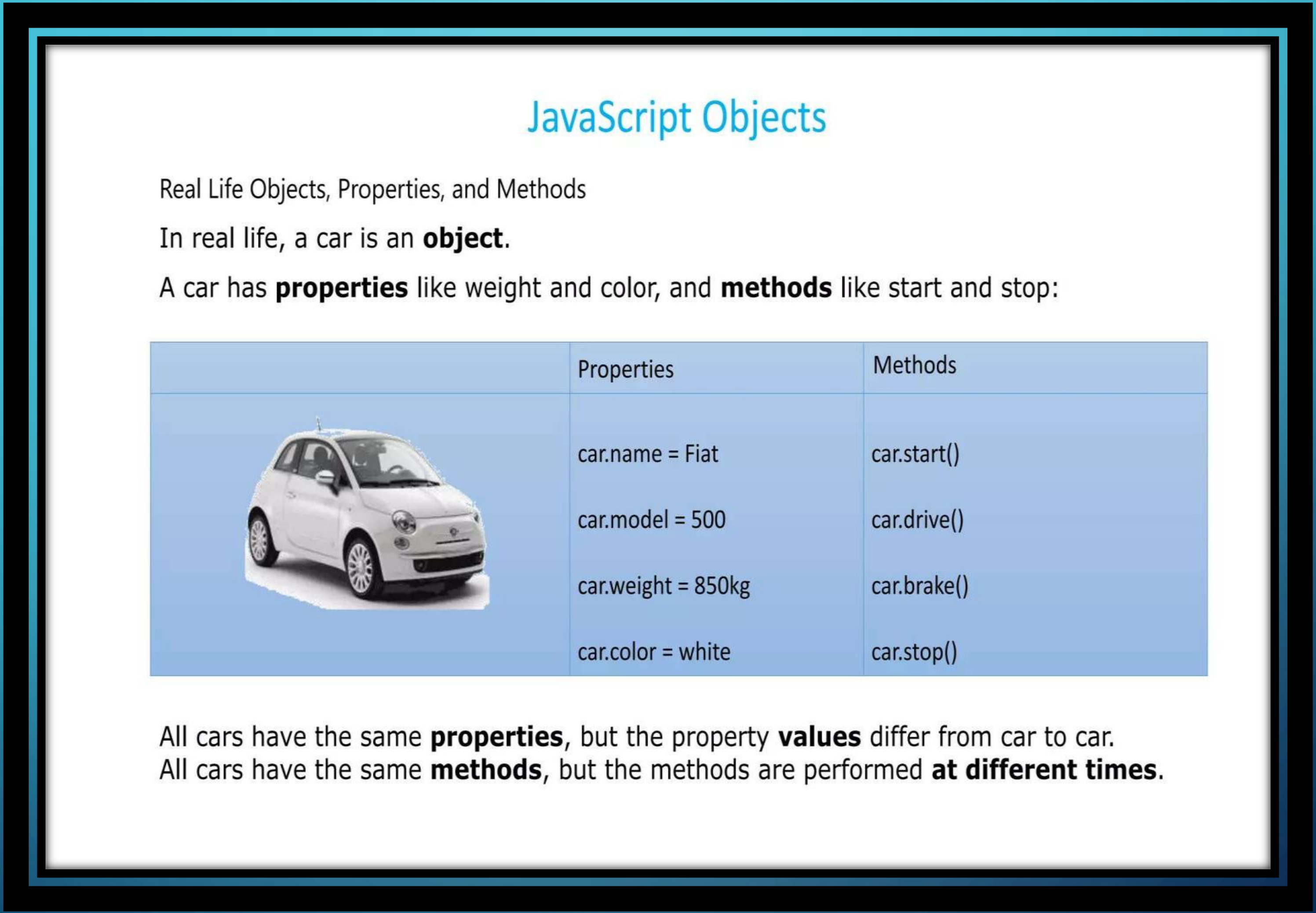
Example

Calculate the product of two numbers, and return the result:

```
let x = myFunction(4, 3);   // Function is called, return value will end up in x

function myFunction(a, b) {
  return a * b;             // Function returns the product of a and b
}
```

The result in x will be: 12

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal



JavaScript Objects

Real Life Objects, Properties, and Methods

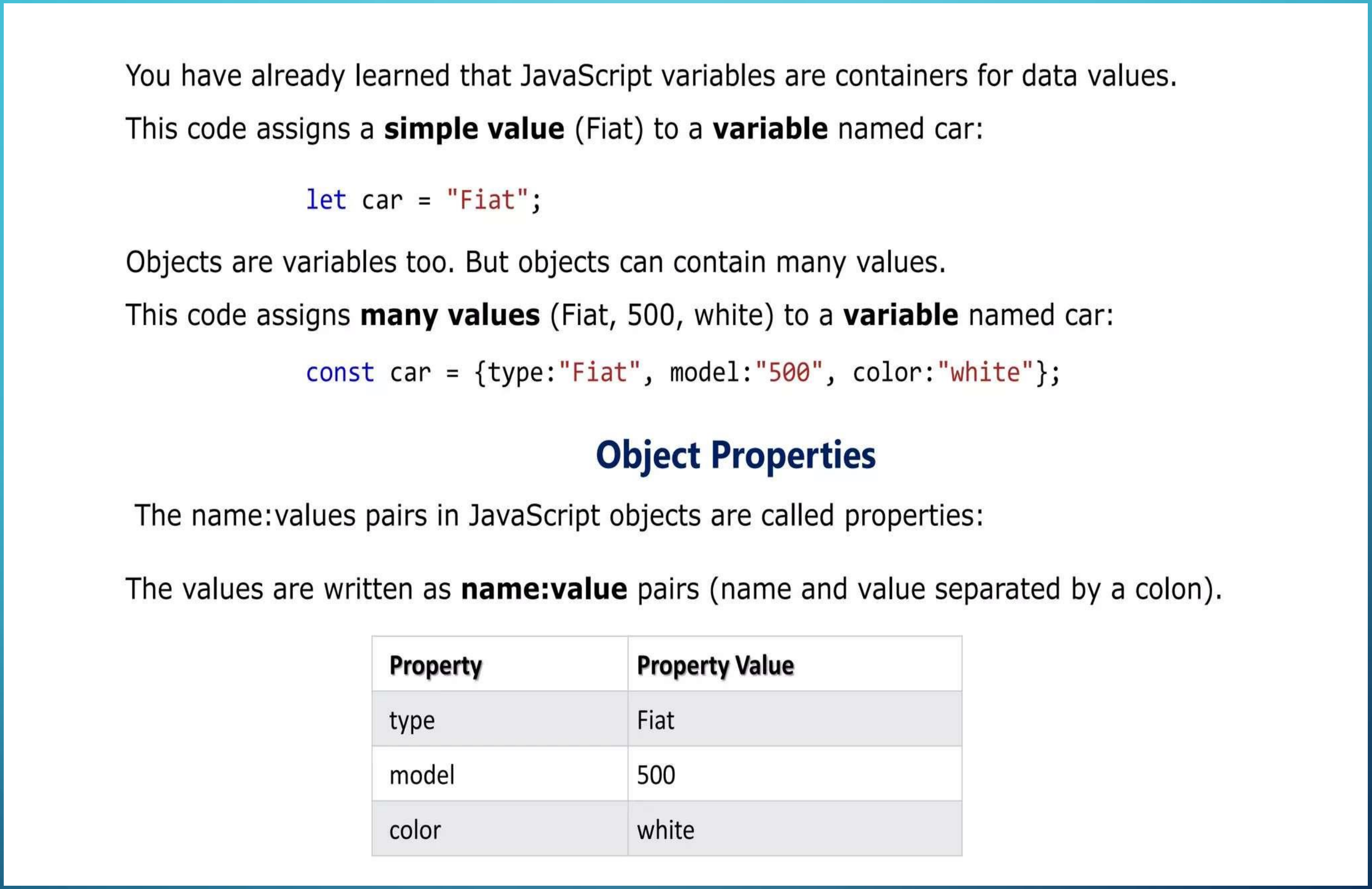
In real life, a car is an **object**.

A car has **properties** like weight and color, and **methods** like start and stop:

	Properties	Methods

	car.name = Fiat
	car.model = 500
	car.weight = 850kg
	car.color = white
		car.start()
		car.drive()
		car.brake()
		car.stop()

All cars have the same **properties**, but the property **values** differ from car to car.
All cars have the same **methods**, but the methods are performed **at different times**.



You have already learned that JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

```
let car = "Fiat";
```

Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to a **variable** named car:

```
const car = {type:"Fiat", model:"500", color:"white"};
```

Object Properties

The name:values pairs in JavaScript objects are called properties:

The values are written as **name:value** pairs (name and value separated by a colon).

Property	Property Value
type	Fiat
model	500
color	white

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

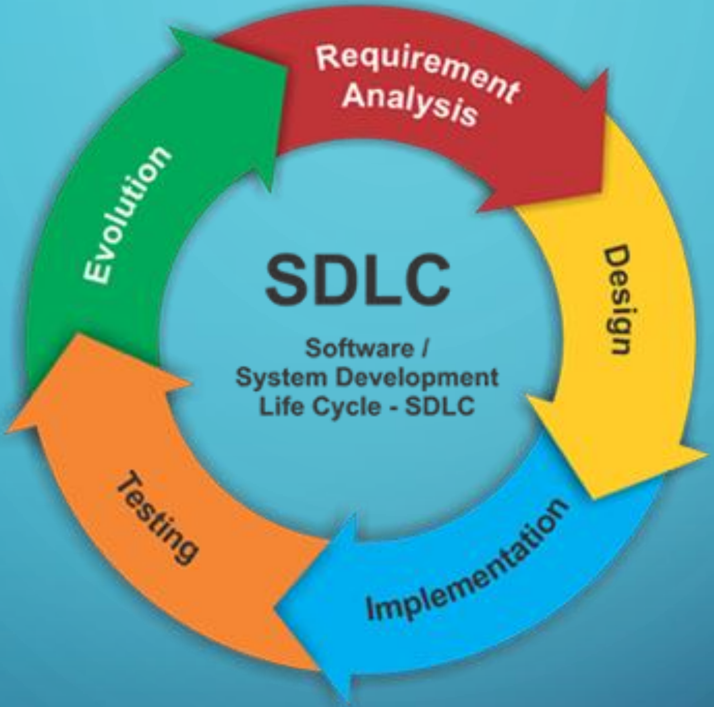
WEEK 16 & 17

Page
80-80

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

PROJECT PLANNING, PRESENTATION & PROJECT SHOW

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal


Requirement Analysis
Design
Implementation
Testing
Evolution
SDLC
Software / System Development Life Cycle - SDLC

Prepared By: Sohag Mollik
Lecturer, Dept. Of CSE
University Of Global Village (UGV), Barishal

THANK YOU