CSC-318
Web Technology
(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

JavaScript Assignment Operators

- Arithmetic operators are used to perform arithmetic on numbers

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

JavaScript Arithmetic Operators

- Addition and Subtraction

```
<script>
  var x = 5;
  var y = 2;
  var sum = x + y; // sum = 7
  var sub = x - y  // sub = 3
</script>
```

JavaScript Arithmetic Operators

- Multiplication

```
<script>
  var x = 5;
  var y = 2;
  var product = x * y; // product = 10
</script>
```

JavaScript Arithmetic Operators

- Division

```
<script>
  var x = 5;
  var y = 2;
  var division = x / y; // division = 2.5
</script>
```

JavaScript Arithmetic Operators

- Remainder (Modulo Division)

```
<script>
  var x = 5;
  var y = 2;
  var z = x % y; // z = 1
</script>
```

JavaScript Arithmetic Operators

- Increment and Decrement Operators

```
<script>
  var x = 5;
  x++; // x = 6

  /*****************************/

  var y = 5;
  y--; // y = 4
</script>
```

JavaScript Arithmetic Operators

- Exponentiation

```
<script>
  var x = 5;
  var y = x ** 2; // y = 25
</script>
```

x ** y produces the same result as `Math.pow(x,y)`:

```
var x = 5;
var z = Math.pow(x,2);    // result is 25
```

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

JavaScript Assignment Operators

- Addition Assignment Operator

```
<script>
    var x = 10;
    x += 5; // x = 15
</script>
```

- **Note : similar for other assignment operators**

JavaScript String Operators

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

```
<script>
  var txt1 = "John";
  var txt2 = "Doe";
  var name = txt1 + " " + txt2;
  // name = "John Doe"
</script>
```

JavaScript Comparison Operators

- Used to compare values

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

JavaScript Logical Operators

- Comparison and Logical operators are used to test for true or false

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

JavaScript Conditional Statements

- Conditional statements are used to perform different actions based on different conditions
- Use "if" to specify a block of code to be executed, if a specified condition is true
- Use "else" to specify a block of code to be executed, if the same condition is false
- Use "else if" to specify a new condition to test, if the first condition is false
- Use "switch" to specify many alternative blocks of code to be executed

JavaScript Conditional Statements

- "if" Statement

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

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

JavaScript Conditional Statements

- "else" Statement

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

JavaScript Conditional Statements

- "else if" Statement

```
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 < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

JavaScript Conditional Statements

- "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
}
```

JavaScript Conditional Statements

- switch Statement

```
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";
}
```

JavaScript Looping Statements

- Loops can execute a block of code a number of times
- JavaScript supports different kinds of loops:
  - for - loops through a block of code a number of times
  - while - loops through a block of code while a specified condition is true
  - do/while - also loops through a block of code while a specified condition is true

JavaScript Looping Statements

- for loop

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

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

JavaScript Looping Statements

- for loop : Example

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

JavaScript Looping Statements

- 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
}
```

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

JavaScript Looping Statements

- do/while loop
  - The do/while loop is a variant of the while loop
  - his 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);
```

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

JavaScript Arrays

- JavaScript arrays are used to store multiple values in a single variable
- An array is a special variable, which can hold more than one value at a time

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

Creating an Array

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

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

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

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

Creating an Array

- Using the JavaScript Keyword new

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Accessing the Elements of an Array

- You access an array element by referring to the index number
- This statement accesses the value of the first element in cars:

```
var name = cars[0];
```

```
var cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML = cars[0];
```

JavaScript Objects

- JavaScript objects are similar to javascript arrays but there are some differences
- JavaScript objects are containers for named values called properties or methods
- You define (and create) a JavaScript object with an object literal:

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

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

JavaScript Object Properties

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

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue

Accessing Object Properties

- You can access object properties in two ways:

```
objectName.propertyName
```

- Or

```
objectName["propertyName"]
```

Object Methods

- Objects can also have methods.
- Methods are actions that can be performed on objects.
- Methods are stored in properties as function definitions.

Property	Property Value
firstName	John
lastName	Doe
age	50
eyeColor	blue
fullName	function() {return this.firstName + " " + this.lastName;}

Object Methods

- A method is a function stored as a property.

```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

Arrays are Objects

- Arrays are a special type of objects. The typeof operator in JavaScript returns "object" for arrays.
- But, JavaScript arrays are best described as arrays.
- Arrays use numbers to access its "elements". In this example, person[0] returns John:

```
<script>
    var person = ["John", "Doe", 46];
    document.getElementById("demo").innerHTML = person[0];
</script>
```

Arrays are Objects

- Objects use names to access its "members". In this example, person.firstName returns John:

```
<script>
    var person = {firstName:"John", lastName:"Doe", age:46};
    var firstname = person["firstName"]; //firstname="John"
</script>
```