PALAWAN TECHNOLOGICAL COLLEGE, INCORPORATED
245 Malvar Street, Puerto Princesa City

Computer Programming 2 (JavaScript)

BSIT 1 A & B

Lesson 3 – Conditional Statements

JavaScript Conditional statements allow you to execute specific blocks of code based on conditions. If the condition meets then a particular block of action will be executed otherwise it will execute another block of action that satisfies that particular condition.

There are several methods that can be used to perform Conditional Statements in JavaScript.

- **if statement**: Executes a block of code if a specified condition is true.
- **else statement**: Executes a block of code if the same condition of the preceding if statement is false.
- **else if statement**: Adds more conditions to the if statement, allowing for multiple alternative conditions to be tested.
- **switch statement**: Evaluates an expression, then executes the case statement that matches the expression's value.
- **ternary operator (conditional operator)**: Provides a concise way to write if-else statements in a single line.

A. The **if** Statement

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

Syntax
```
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.*

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

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

<h2>JavaScript if</h2>
```

```
<p>Display "Good day!" if the hour is less than 18:00:</p>

<p id="demo">Good Evening!</p>

<script>
if (new Date().getHours() < 18) {
  document.getElementById("demo").innerHTML = "Good day!";
}
</script>

</body>
</html>
```

B. 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
}
```

Example

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

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

<h2>JavaScript if .. else</h2>

<p>A time-based greeting:</p>

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

<script>
const hour = new Date().getHours();
let greeting;

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

document.getElementById("demo").innerHTML = greeting;
```

```
</script>

</body>
</html>
```

C. The **else if** Statement

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

Syntax

```
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
}
```

Example

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":

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

<h2>JavaScript else .. if</h2>

<p>A time-based greeting:</p>

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

<script>
const time = new Date().getHours();
let greeting;
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
document.getElementById("demo").innerHTML = greeting;
</script>

</body>
</html>
```

D. The **switch** statement

The switch statement is used to perform different actions based on different conditions. Use the switch statement to select one of many code blocks to be executed.

Syntax

```
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.

Example

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:

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

<h2>JavaScript switch</h2>

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

<script>
let day;
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";
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

</body>
</html>
```

D1. The break Keyword

When JavaScript reaches a break keyword, it breaks out of the switch block.

This will stop the execution inside the switch block.

It is not necessary to break the last case in a switch block. The block breaks (ends) there anyway.

Note: If you omit the break statement, the next case will be executed even if the evaluation does not match the case.

D2. The default Keyword

The default keyword specifies the code to run if there is no case match:

Example
The getDay() method returns the weekday as a number between 0 and 6.

If today is neither Saturday (6) nor Sunday (0), write a default message:

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

<h2>JavaScript switch</h2>
```

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

<script>
let text;
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

If default is not the last case in the switch block, remember to end the default case with a break.

D3. Common Code Blocks

Sometimes you will want different switch cases to use the same code.

In this example case 4 and 5 share the same code block, and 0 and 6 share another code block:

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

<h2>JavaScript switch</h2>

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

<script>
let text;
switch (new Date().getDay()) {
  case 4:
  case 5:
    text = "Soon it is Weekend";
    break;
  case 0:
  case 6:
    text = "It is Weekend";
    break;
  default:
    text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```

D4. Switching Details

If multiple cases matches a case value, the first case is selected.

If no matching cases are found, the program continues to the default label.

If no default label is found, the program continues to the statement(s) after the switch.

D5. Strict Comparison

Switch cases use strict comparison (===).

The values must be of the same type to match.

A strict comparison can only be true if the operands are of the same type.

In this example there will be no match for x:

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

<h2>JavaScript switch</h2>

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

<script>
let x = "0";

switch (x) {
  case 0:
    text = "Off";
    break;
  case 1:
    text = "On";
    break;
  default:
    text = "No value found";
}
```

```
document.getElementById("demo").innerHTML = text;
</script>

</body>
</html>
```