INFO 151
Web Systems and Services

Week 8 (T2)

Dr Philip Moore

Dr Zhili Zhao

Overview

- In this tutorial we will introduce:
  - Iteration (loops) in PHP and compare the JavaScript and PHP syntax
  - The use of **`goto`** in place of a multi-level **`break`** in PHP
- There are similarities and differences between JavaScript and PHP
  - The differences are mainly in the language syntax
  - Using the incorrect syntax will produce errors which are hard to find

Iteration (loops)

Iteration (Loops)

- In JavaScript there are four loop types
- `while` loops
- `do` … `while` loops
- `for` loops
- `for` … `in` loops
  - Loop through the properties of an object

- In PHP there are four loop types
- `while` loops
- `do` … `while` loops
- `for` loops
- `for` … `each` loops
  - Works only on arrays
  - Used to loop through each key/value pair in an array

Comparing Loop Syntax

- From the previous slide we can see that
  - Both JavaScript and PHP implement four loop types
- Both languages implement
  - A `for` loop
  - A `while` loop
  - A `do … while` loop
- However
  - PHP implements a `for … each` loop (arrays)
  - JavaScript implements a `for … in` loop (objects)
- The following examples demonstrate the different loop syntax

`for` Loops

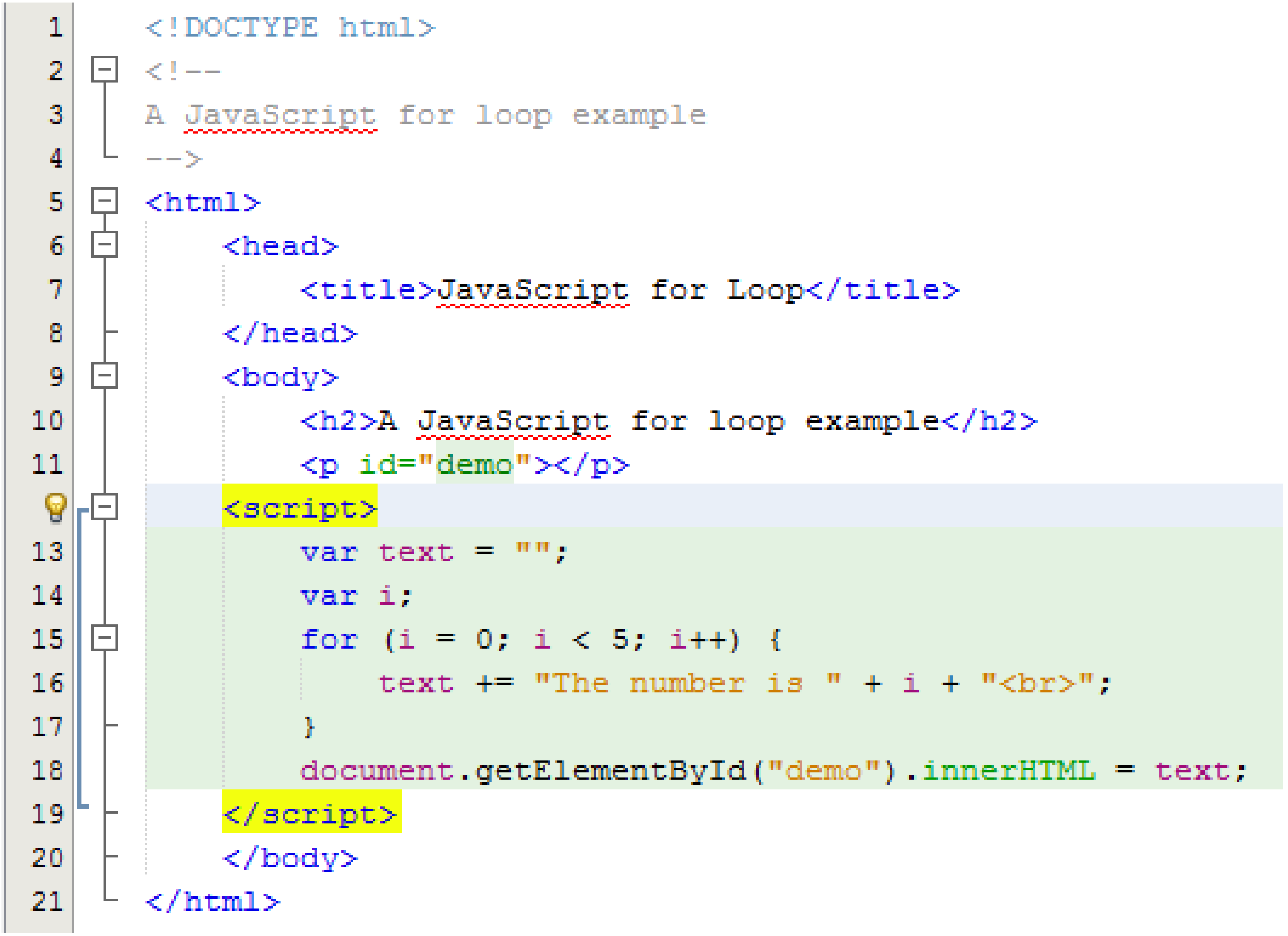
`for` loop

- The syntax for a `for` loop is as follows

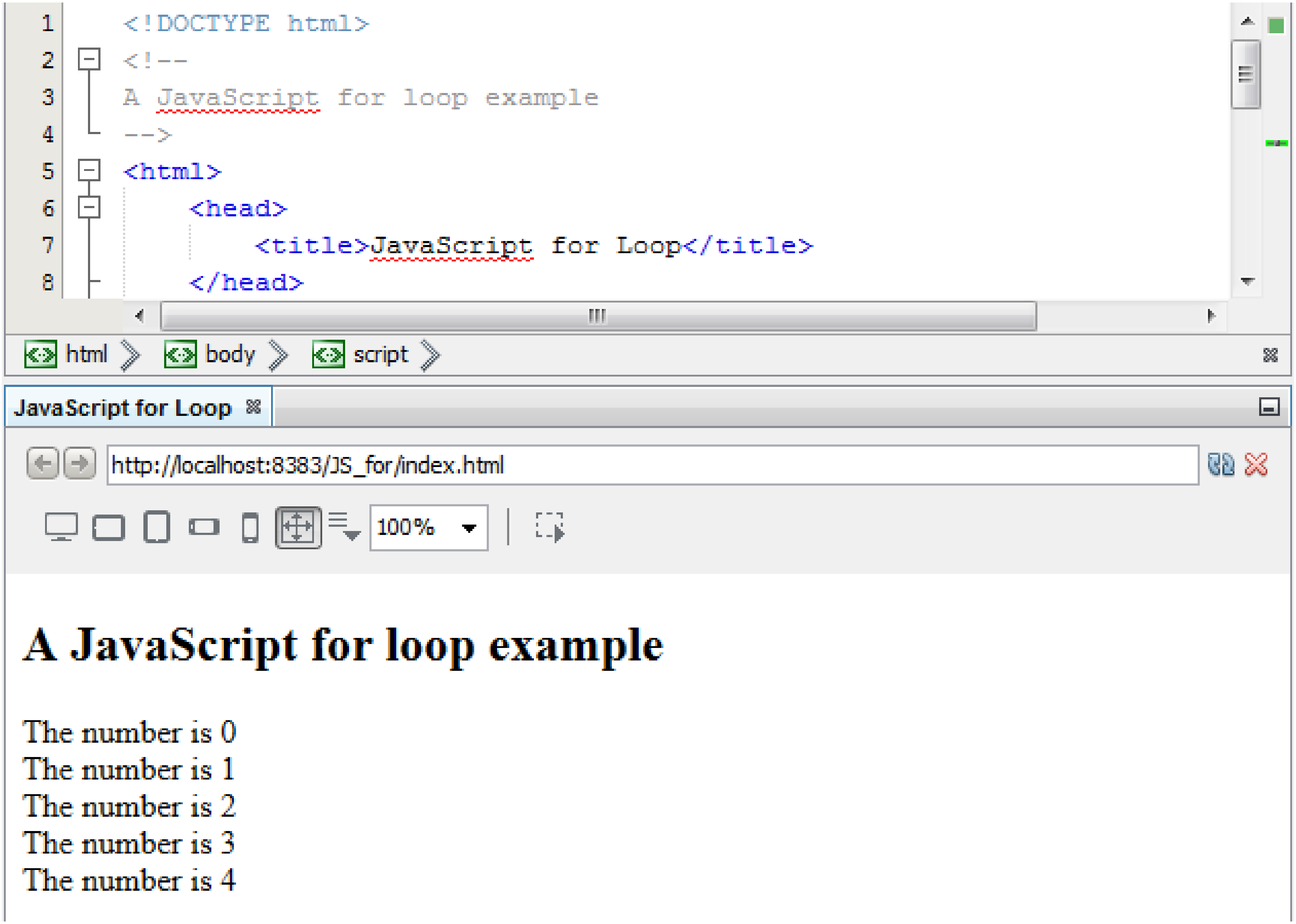
```
for ( initialise ; test ; update){
    statement
}
```

- The loop:
  - Repeatedly executes the ***statement*** while the ***test*** expression is `true`
  - Evaluates the ***initialise*** expression once before starting the loop run
  - Then evaluates the ***update*** expression at the termination of each iteration

JavaScript `for` Loop



```
<!DOCTYPE html>
<!--
A JavaScript for loop example
-->
<html>
    <head>
        <title>JavaScript for Loop</title>
    </head>
    <body>
        <h2>A JavaScript for loop example</h2>
        <p id="demo"></p>
    <script>
        var text = "";
        var i;
        for (i = 0; i < 5; i++) {
            text += "The number is " + i + "<br>";
        }
        document.getElementById("demo").innerHTML = text;
    </script>
    </body>
</html>
```



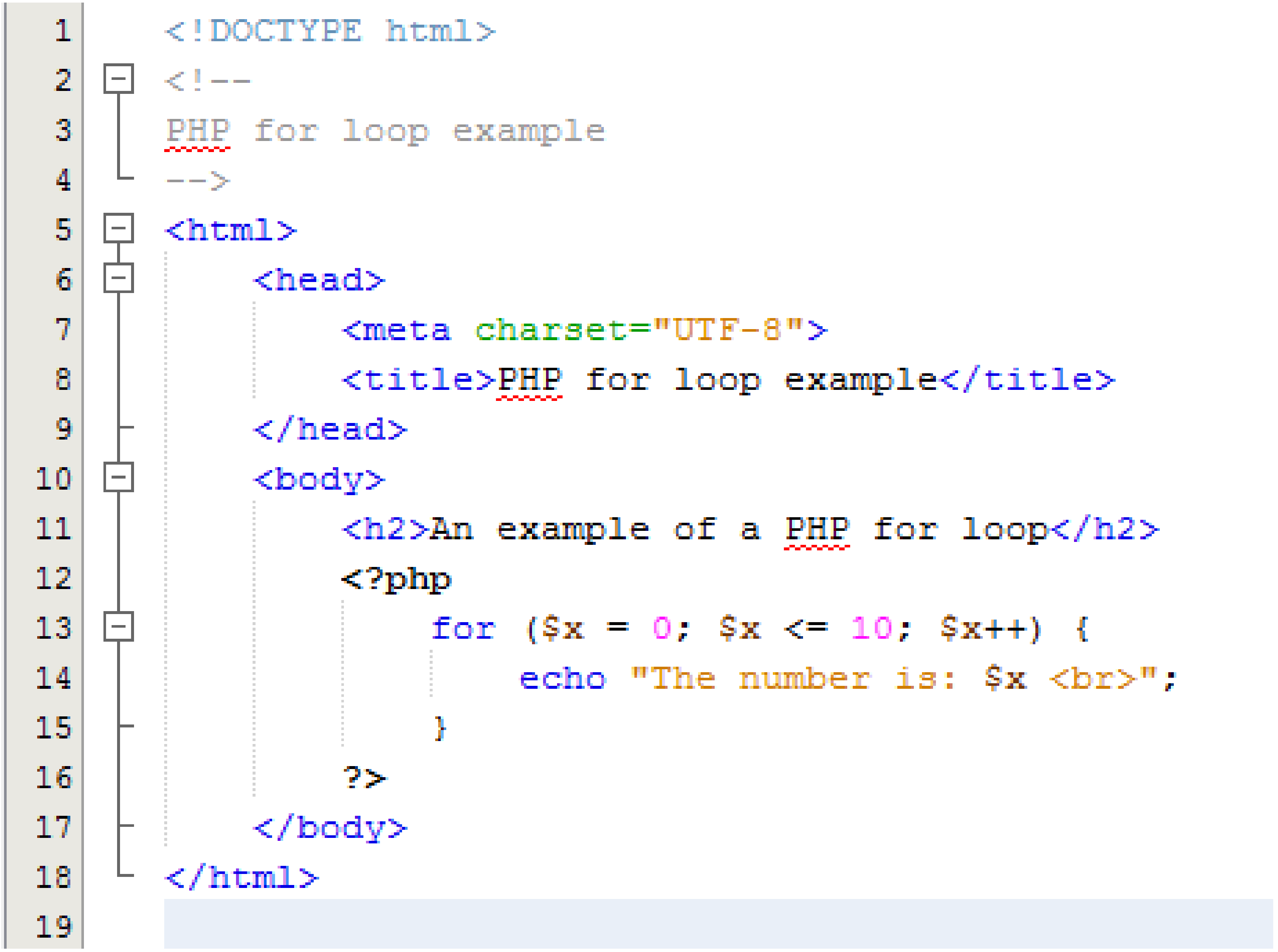
```
<!DOCTYPE html>
<!--
A JavaScript for loop example
-->
<html>
    <head>
        <title>JavaScript for Loop</title>
    </head>
```

http://localhost:8383/JS_for/index.html

A JavaScript for loop example

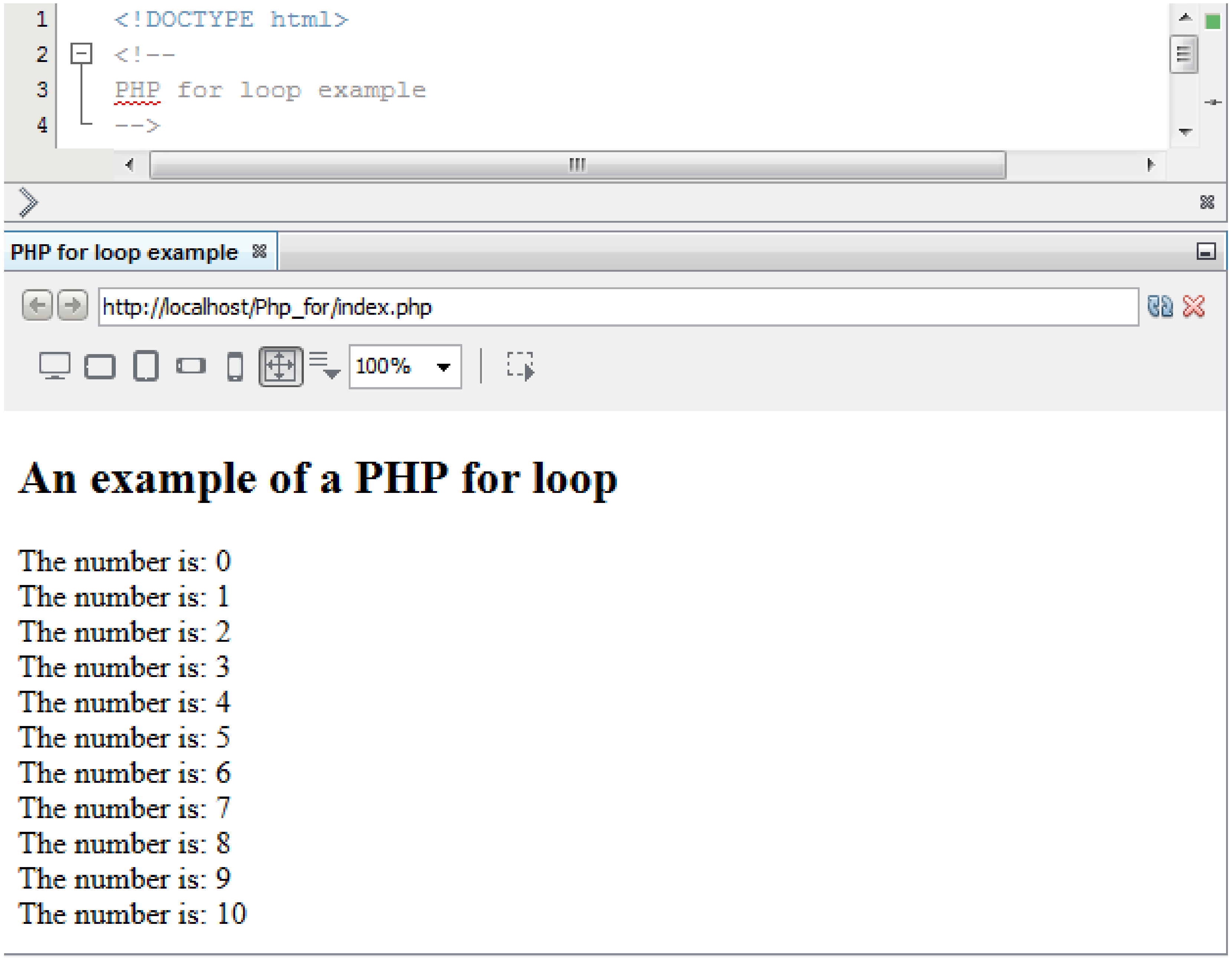
```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

PHP `for` Loop



```
<!DOCTYPE html>
<!--
PHP for loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP for loop example</title>
    </head>
    <body>
        <h2>An example of a PHP for loop</h2>
        <?php
            for ($x = 0; $x <= 10; $x++) {
                echo "The number is: $x <br>";
            }
        ?>
    </body>
</html>

```



```
<!DOCTYPE html>
<!--
PHP for loop example
-->
```

```
An example of a PHP for loop

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
The number is: 10
```

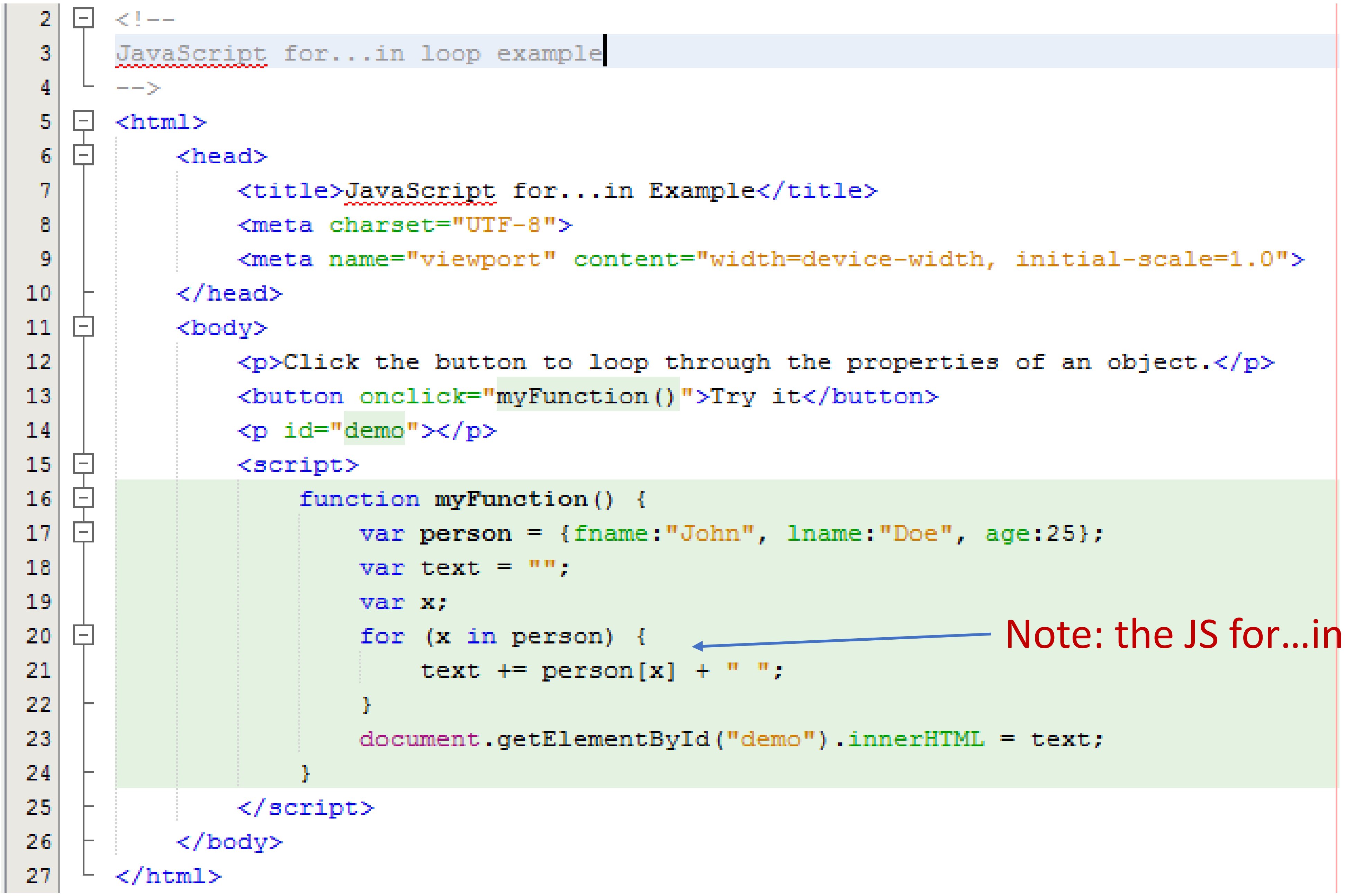
JavaScript
`for`…`in` Loop

`for…in` loop

- The syntax for a `for…in` loop is as follows

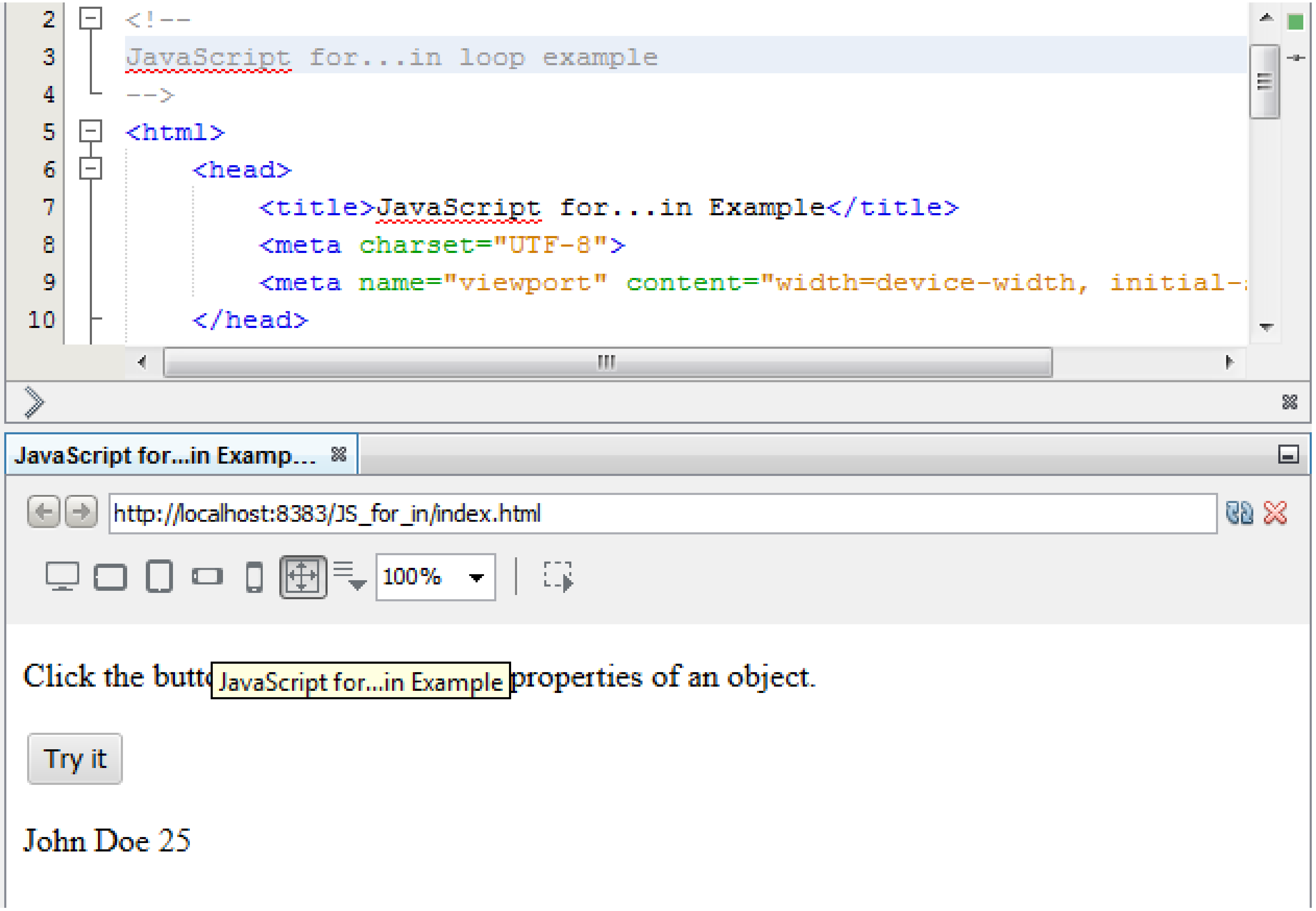
```
for ( variable in object ){
      statement
}
```

- The for…in loop executes a statement once for each property in an object
- Each time through the for…in loop is assigns the name of the current property to a specified variable
- Some properties of pre-defined JavaScript objects are not enumerated by the for…in loop
- User defined properties are always enumerated
- The following example demonstrates the for…in loop



```
<!--
JavaScript for...in loop example
-->
<html>
    <head>
        <title>JavaScript for...in Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p>Click the button to loop through the properties of an object.</p>
        <button onclick="myFunction()">Try it</button>
        <p id="demo"></p>
        <script>
            function myFunction() {
                var person = {fname:"John", lname:"Doe", age:25};
                var text = "";
                var x;
                for (x in person) {
                    text += person[x] + " ";
                }
                document.getElementById("demo").innerHTML = text;
            }
        </script>
    </body>
</html>
```

Note: the JS for…in loop



```
<!--
JavaScript for...in loop example
-->
<html>
    <head>
        <title>JavaScript for...in Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
    </head>
```

http://localhost:8383/JS_for_in/index.html

Click the butt JavaScript for...in Example properties of an object.

John Doe 25

PHP

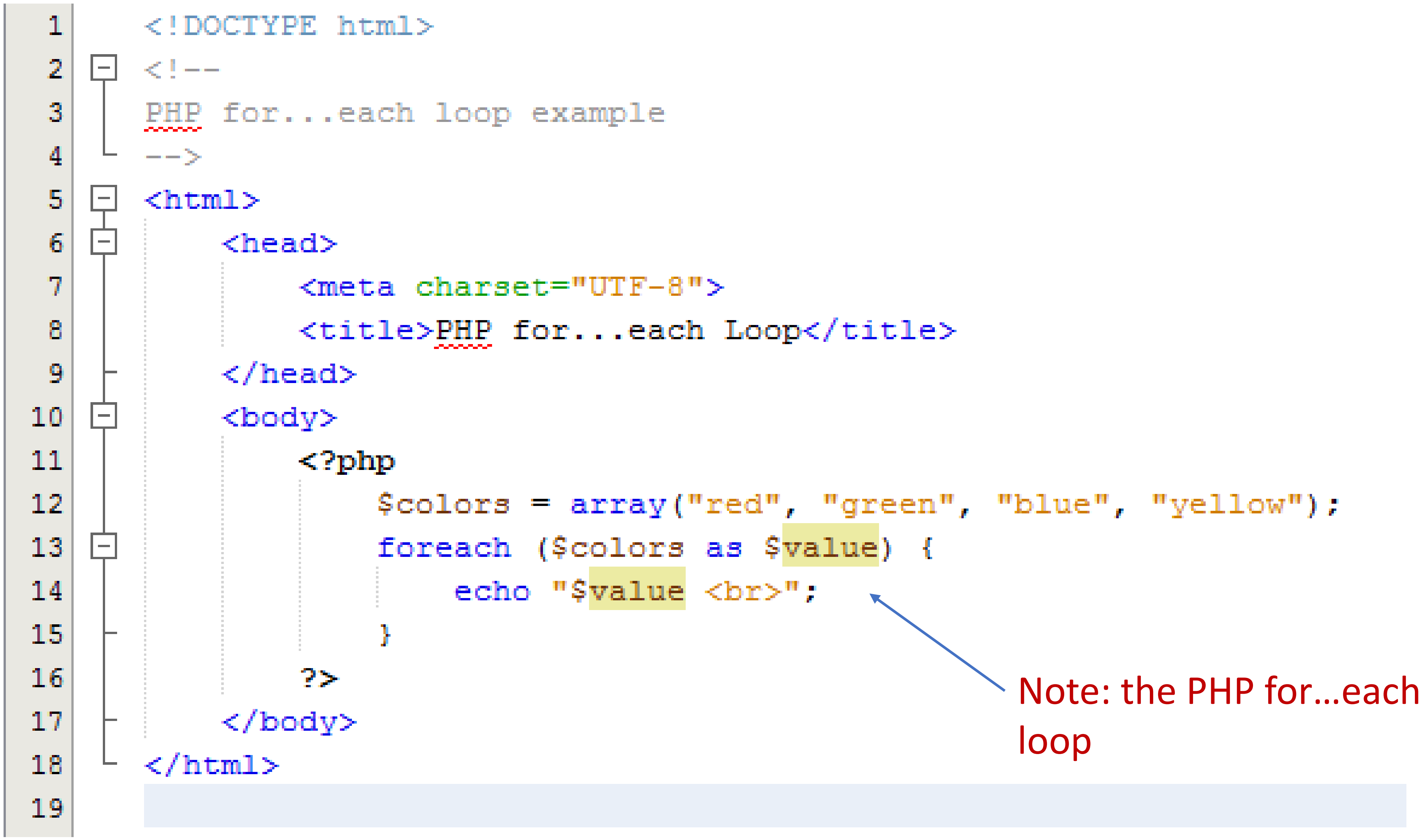
for…each Loop

`for…each` loop

- The syntax for a `for…each` loop is as follows

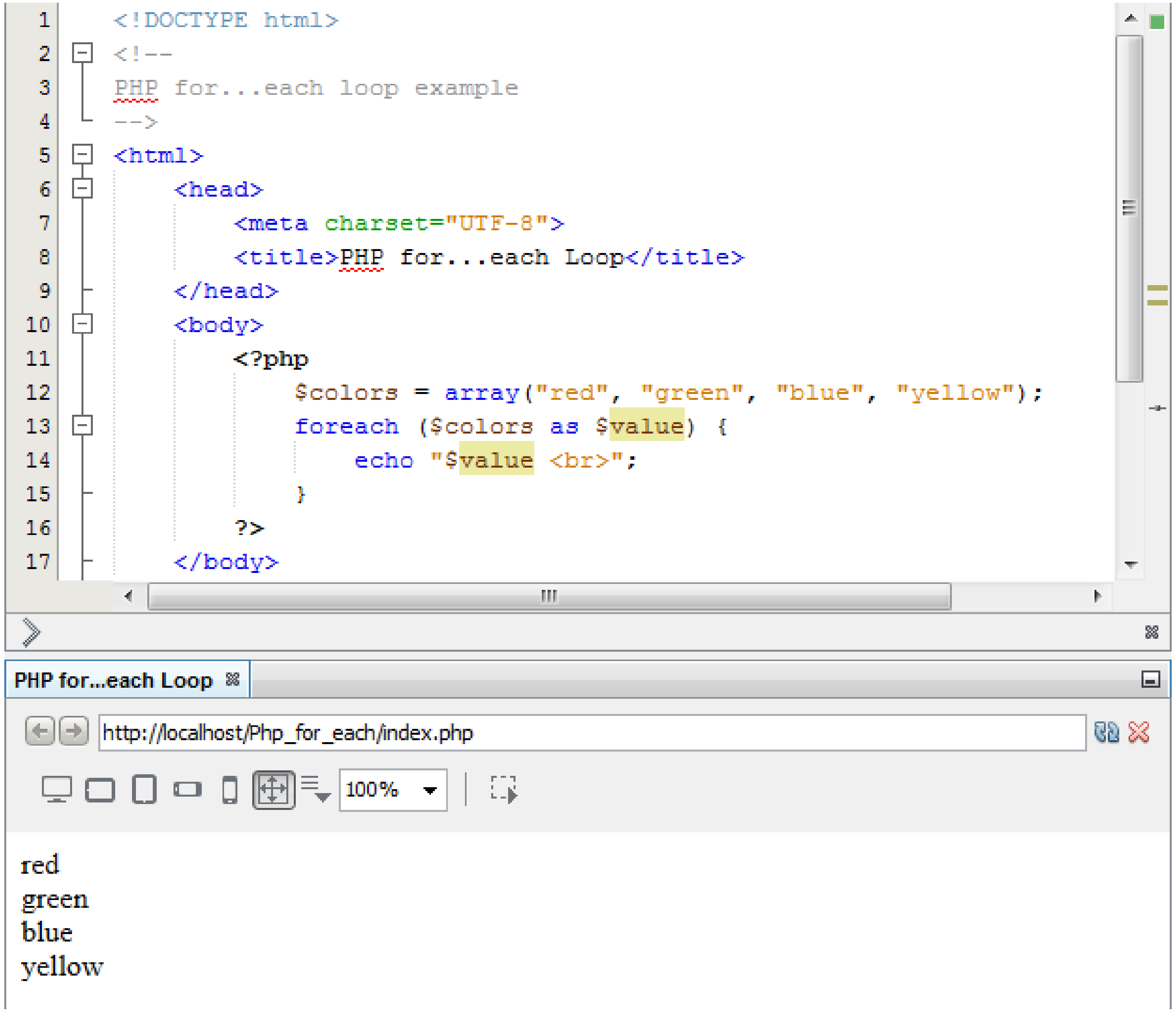
```
for ( array as value ){
        statement
}
```

- The for…each loop is used to iterate over a PHP array
- for…each loops over an array with each value for the current array element assigned to $value
- The array pointer is advanced by one to go the next element in the array
- The following example demonstrates the for…each loop



```
<!DOCTYPE html>
<!--
PHP for...each loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP for...each Loop</title>
    </head>
    <body>
        <?php
            $colors = array("red", "green", "blue", "yellow");
            foreach ($colors as $value) {
                echo "$value <br>";
            }
        ?>
    </body>
</html>
```

Note: the PHP for…each loop



```
<!DOCTYPE html>
<!--
PHP for...each loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP for...each Loop</title>
    </head>
    <body>
        <?php
            $colors = array("red", "green", "blue", "yellow");
            foreach ($colors as $value) {
                echo "$value <br>";
            }
        ?>
    </body>
```

PHP for...each Loop

http://localhost/Php_for_each/index.php

100%

```
red
green
blue
yellow
```

`while` Loops

while loop and **do…while** loop

- The syntax for a **while** loop is as follows

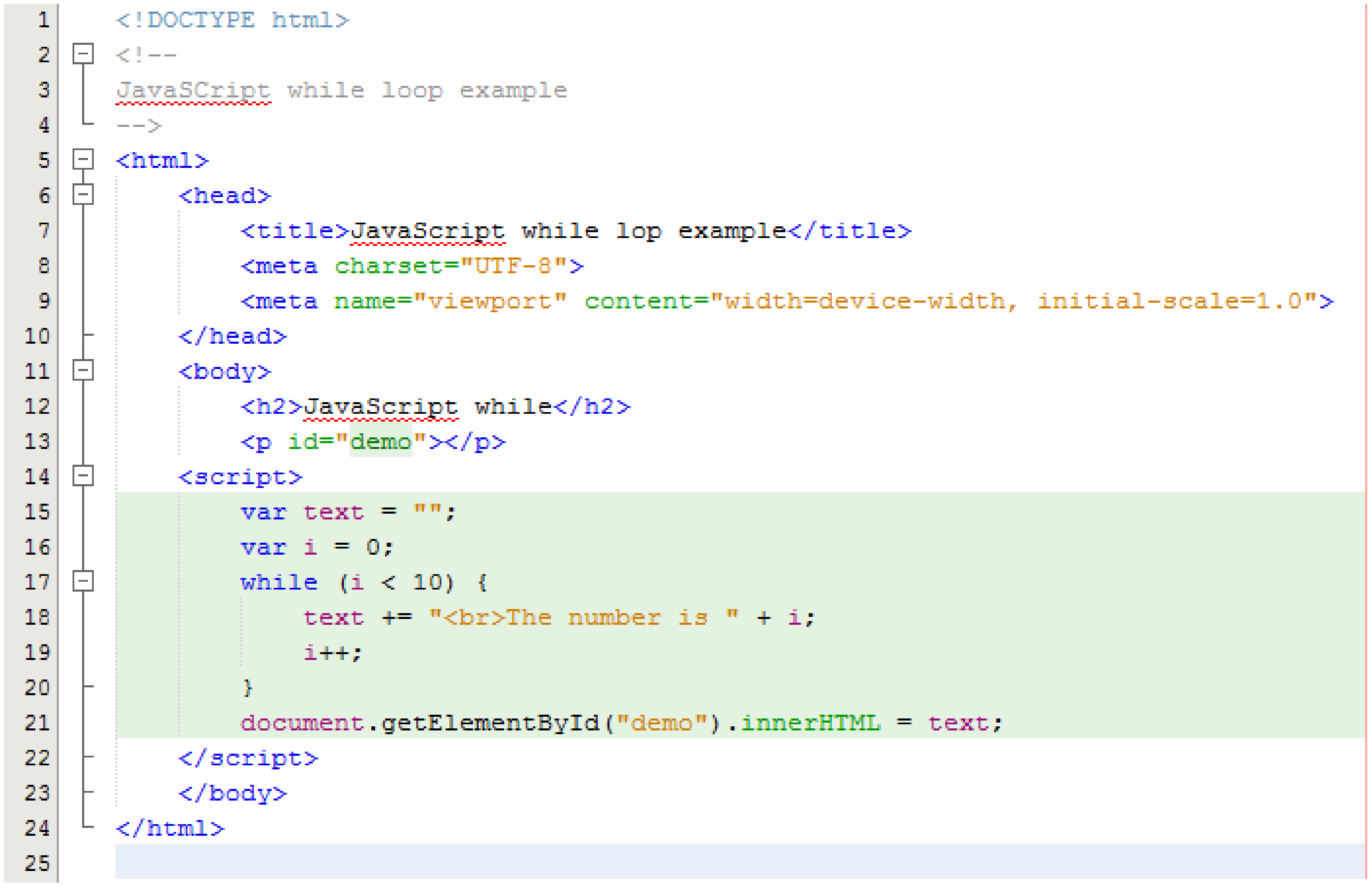
```
while ( expression ){
    statement
}
```

- The syntax for a **do…while** loop is as follows

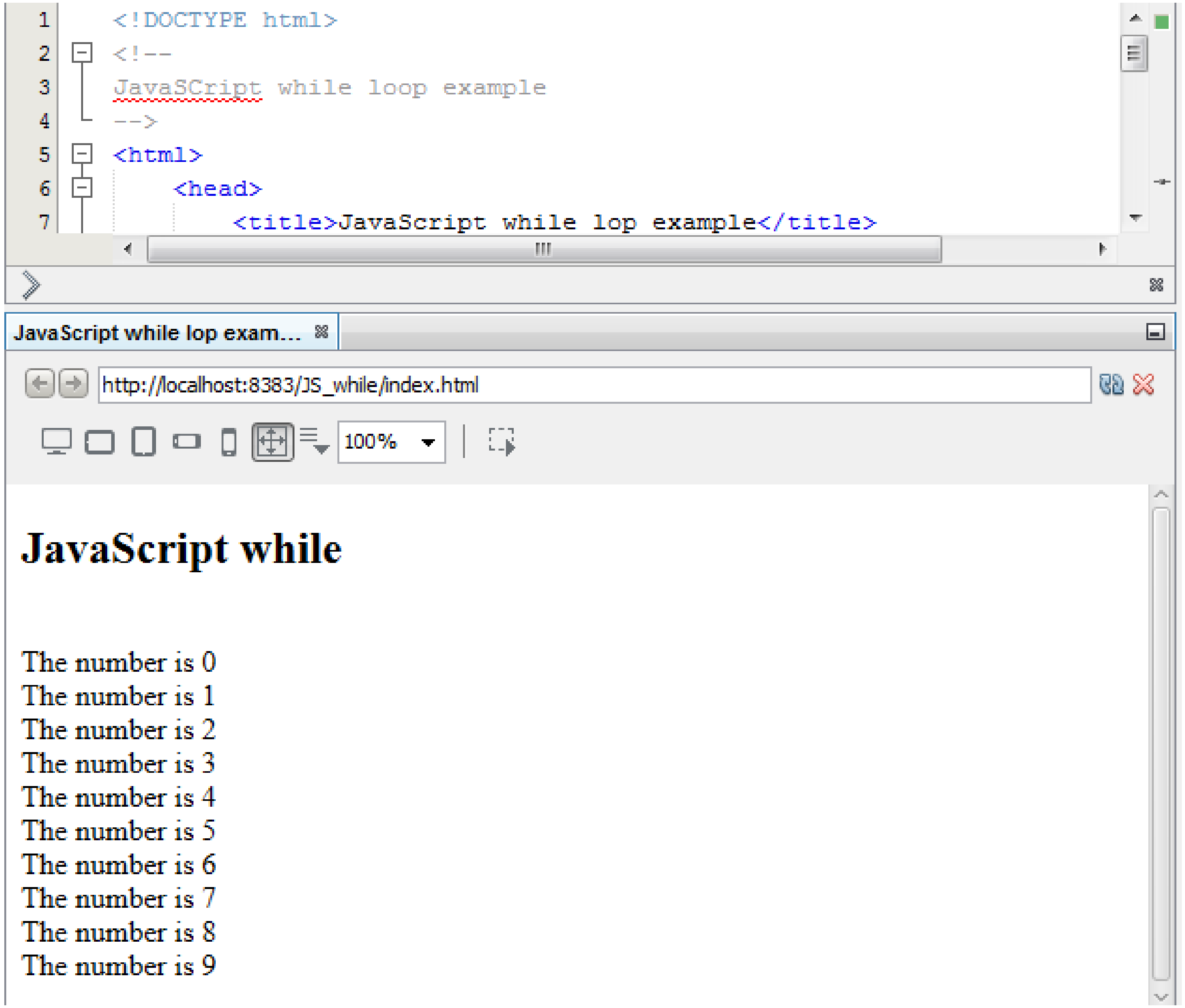
```
do{
    statement
}while ( expression );
```

- The following examples show the while and do…while loops

JavaScript **`while`** Loop



```html
<!DOCTYPE html>
<!--
JavaSCript while loop example
-->
<html>
    <head>
        <title>JavaScript while lop example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h2>JavaScript while</h2>
        <p id="demo"></p>
    <script>
        var text = "";
        var i = 0;
        while (i < 10) {
            text += "<br>The number is " + i;
            i++;
        }
        document.getElementById("demo").innerHTML = text;
    </script>
    </body>
</html>
```



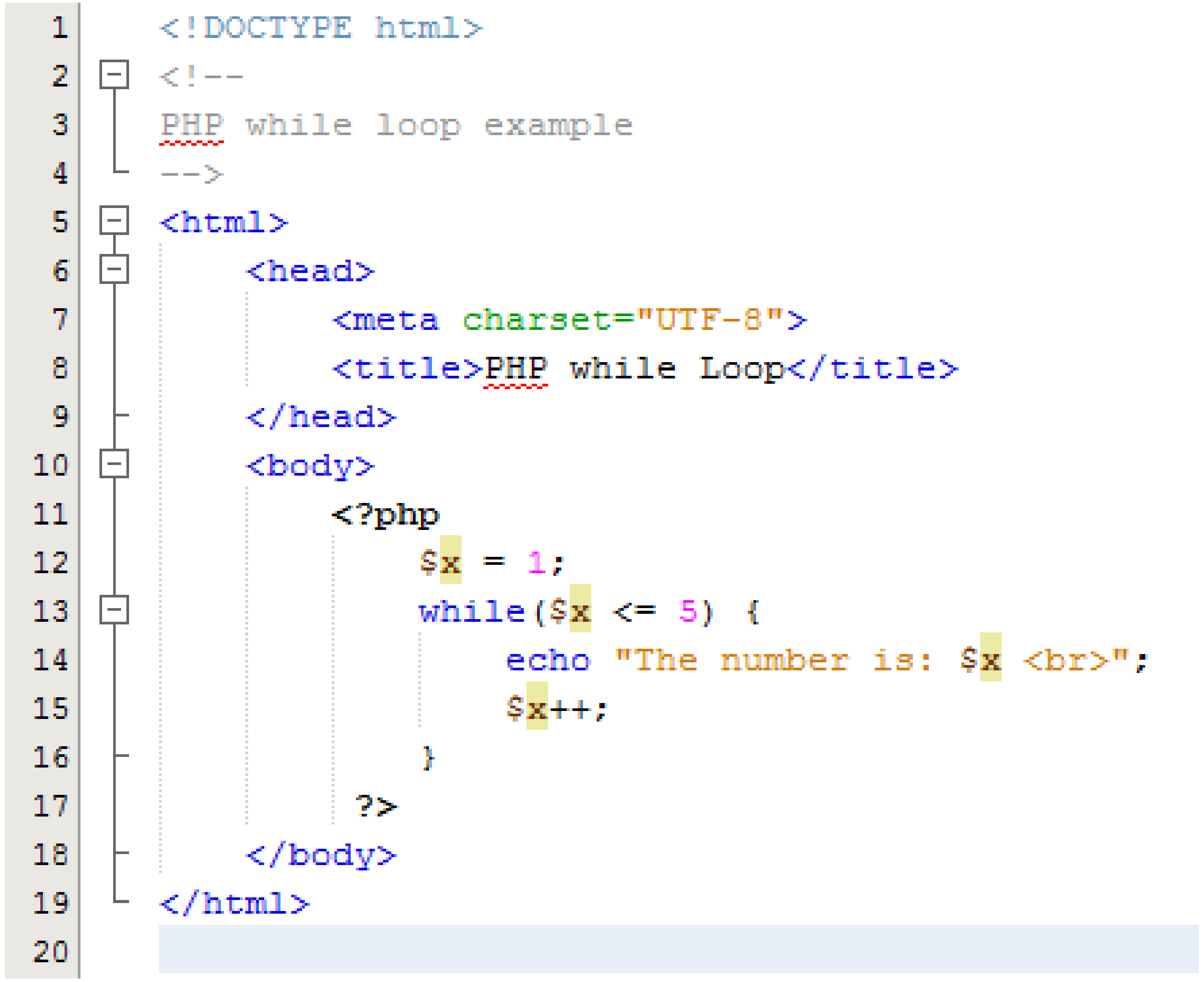
```
<!DOCTYPE html>
<!--
JavaSCript while loop example
-->
<html>
    <head>
        <title>JavaScript while lop example</title>
```

JavaScript while lop exam...
http://localhost:8383/JS_while/index.html
100%

JavaScript while

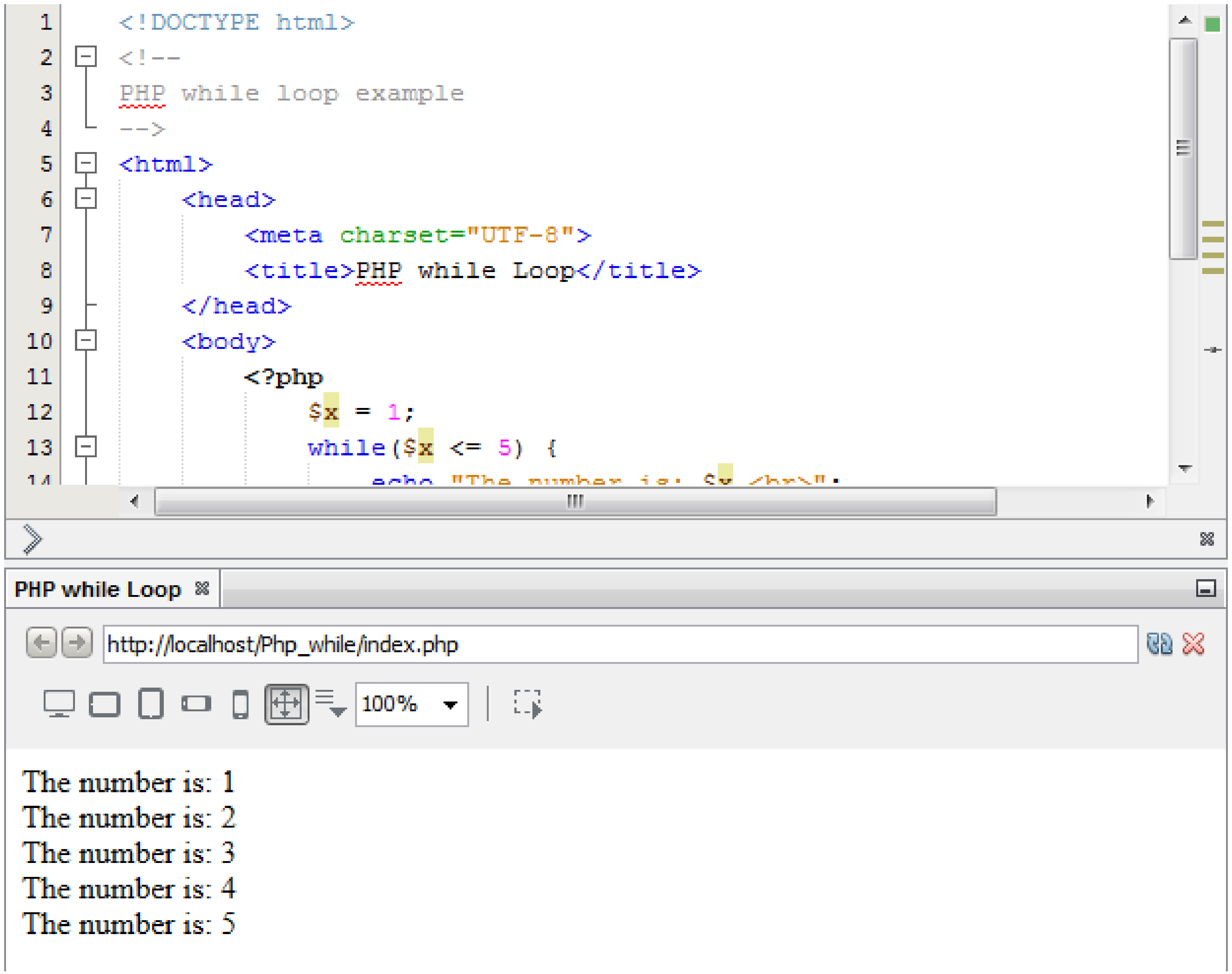
```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

PHP **`while`** Loop



```
<!DOCTYPE html>
<!--
PHP while loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP while Loop</title>
    </head>
    <body>
        <?php
            $x = 1;
            while($x <= 5) {
                echo "The number is: $x <br>";
                $x++;
            }
         ?>
    </body>
</html>

```

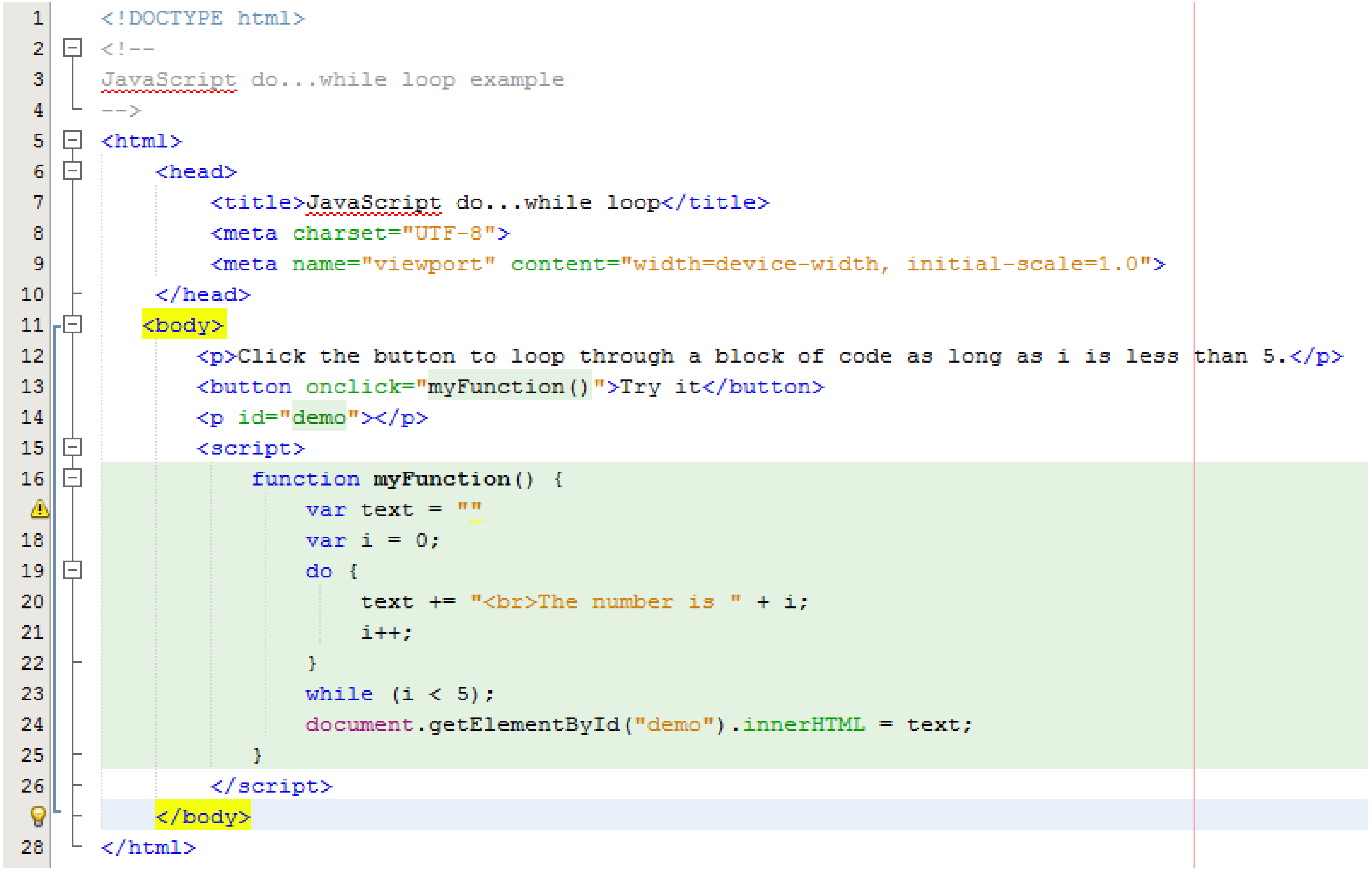


```
<!DOCTYPE html>
<!--
PHP while loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP while Loop</title>
    </head>
    <body>
        <?php
            $x = 1;
            while($x <= 5) {
                echo "The number is: $x <br>";
```

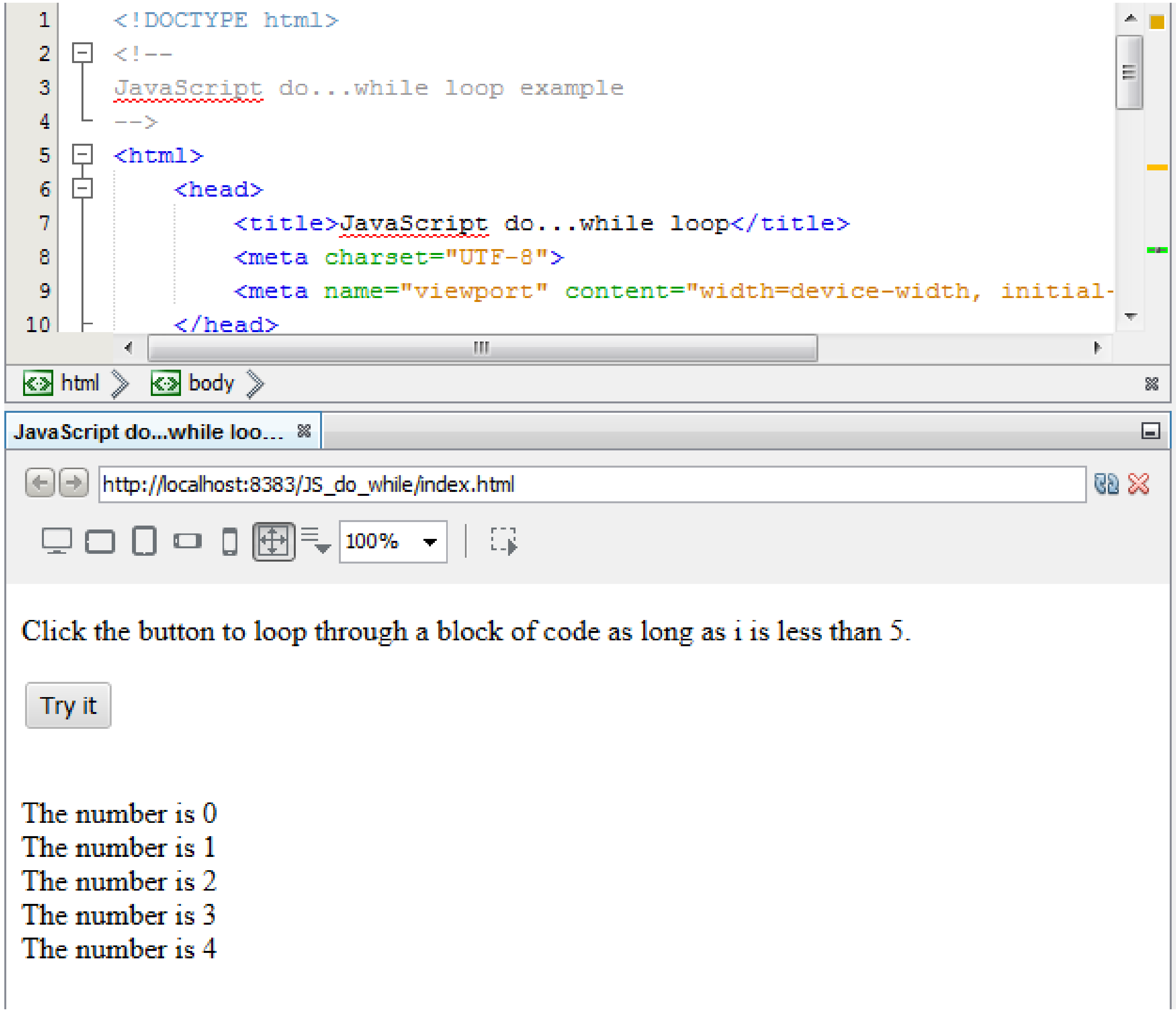
http://localhost/Php_while/index.php

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

JavaScript **`do…while`** Loop



```
<!DOCTYPE html>
<!--
JavaScript do...while loop example
-->
<html>
    <head>
        <title>JavaScript do...while loop</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
   <body>
       <p>Click the button to loop through a block of code as long as i is less than 5.</p>
       <button onclick="myFunction()">Try it</button>
       <p id="demo"></p>
       <script>
           function myFunction() {
               var text = ""
               var i = 0;
               do {
                   text += "<br>The number is " + i;
                   i++;
               }
               while (i < 5);
               document.getElementById("demo").innerHTML = text;
           }
        </script>
    </body>
</html>
```



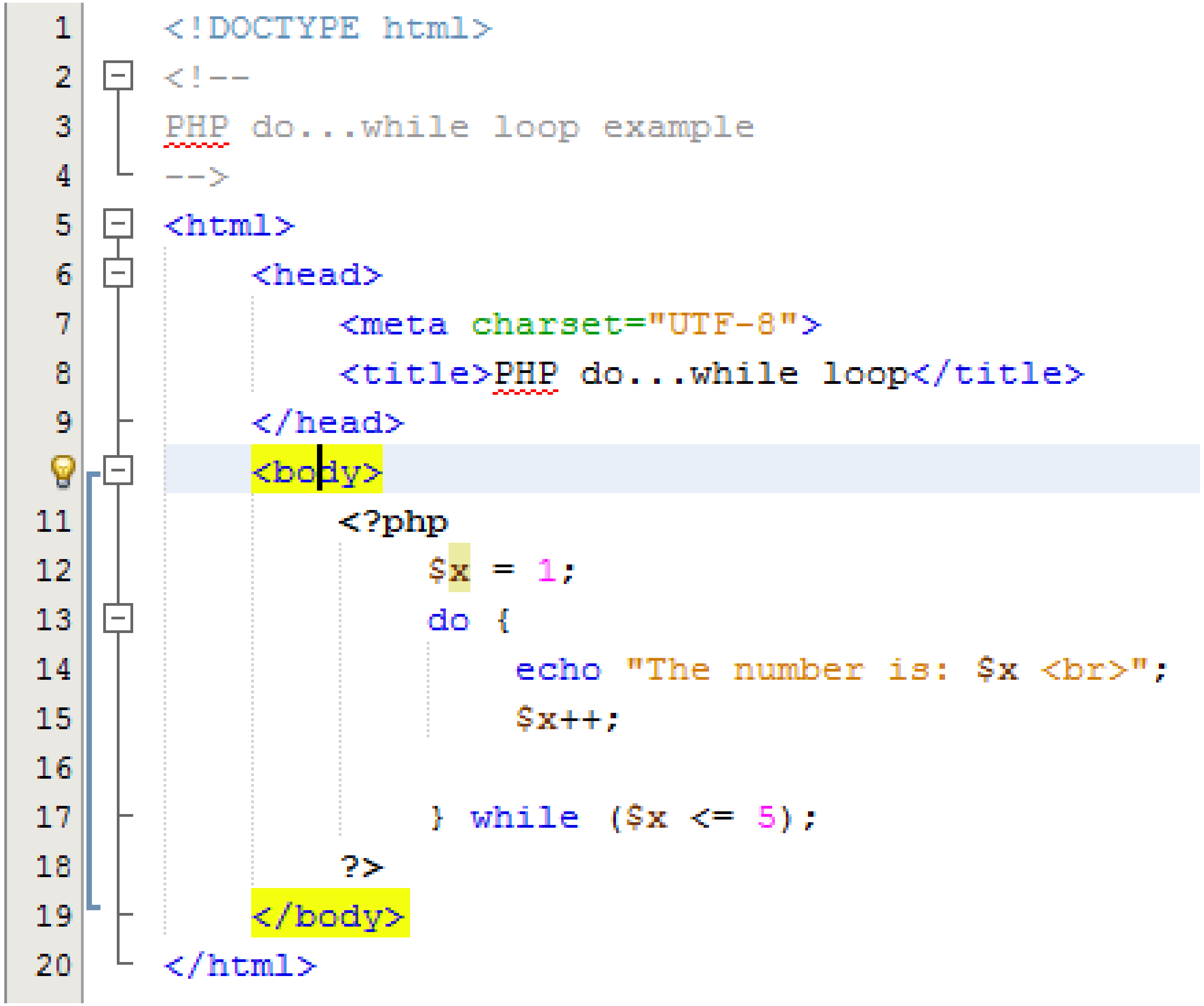
```
<!DOCTYPE html>
<!--
JavaScript do...while loop example
-->
<html>
    <head>
        <title>JavaScript do...while loop</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
    </head>
```

html
body
JavaScript do...while loo...
http://localhost:8383/JS_do_while/index.html
100%

Click the button to loop through a block of code as long as i is less than 5.

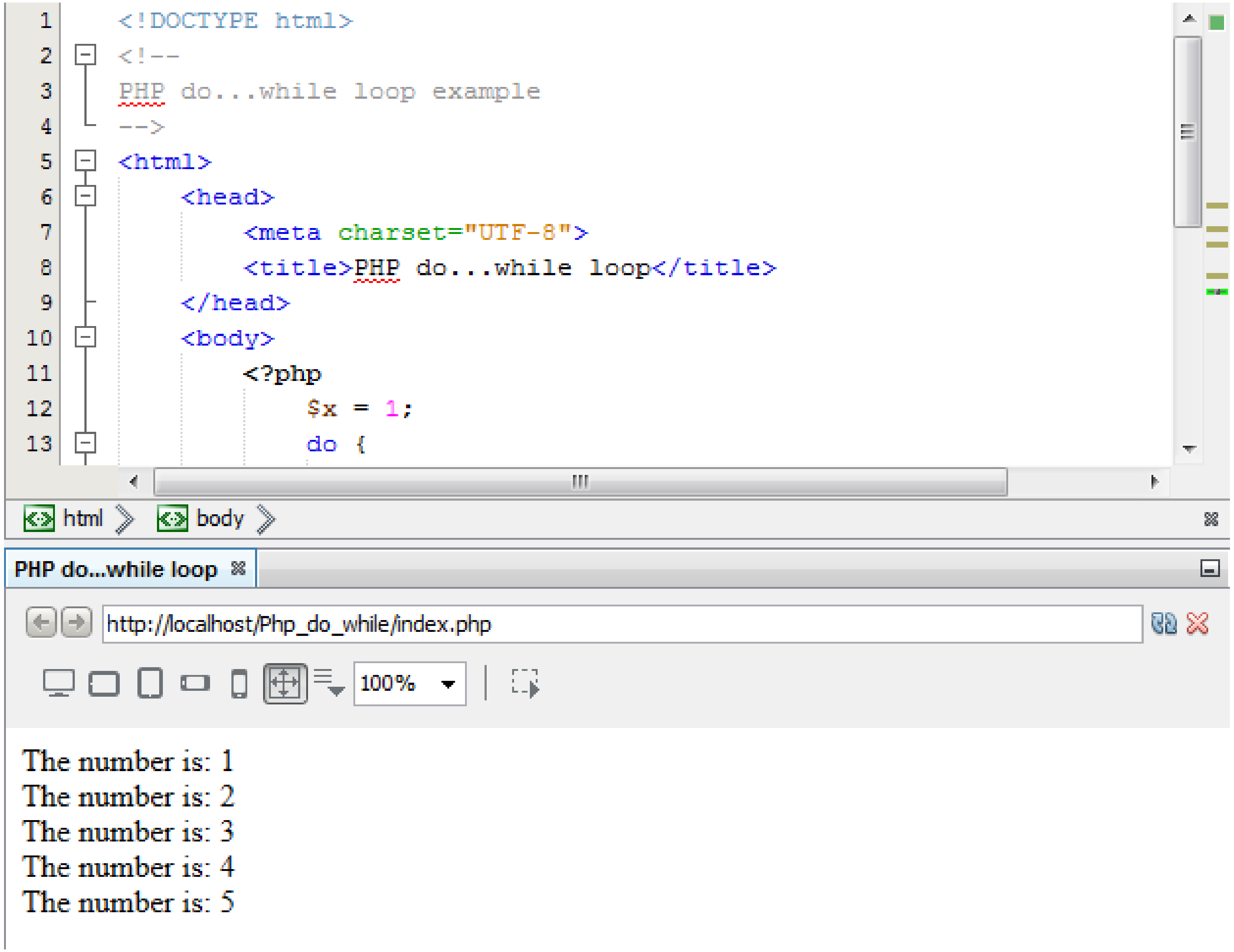
```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
```

PHP **`do…while`** Loop



```
<!DOCTYPE html>
<!--
PHP do...while loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP do...while loop</title>
    </head>
    <body>
        <?php
            $x = 1;
            do {
                echo "The number is: $x <br>";
                $x++;

            } while ($x <= 5);
        ?>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
PHP do...while loop example
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP do...while loop</title>
    </head>
    <body>
        <?php
            $x = 1;
            do {
```

http://localhost/Php_do_while/index.php

```
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

`break` and `continue`

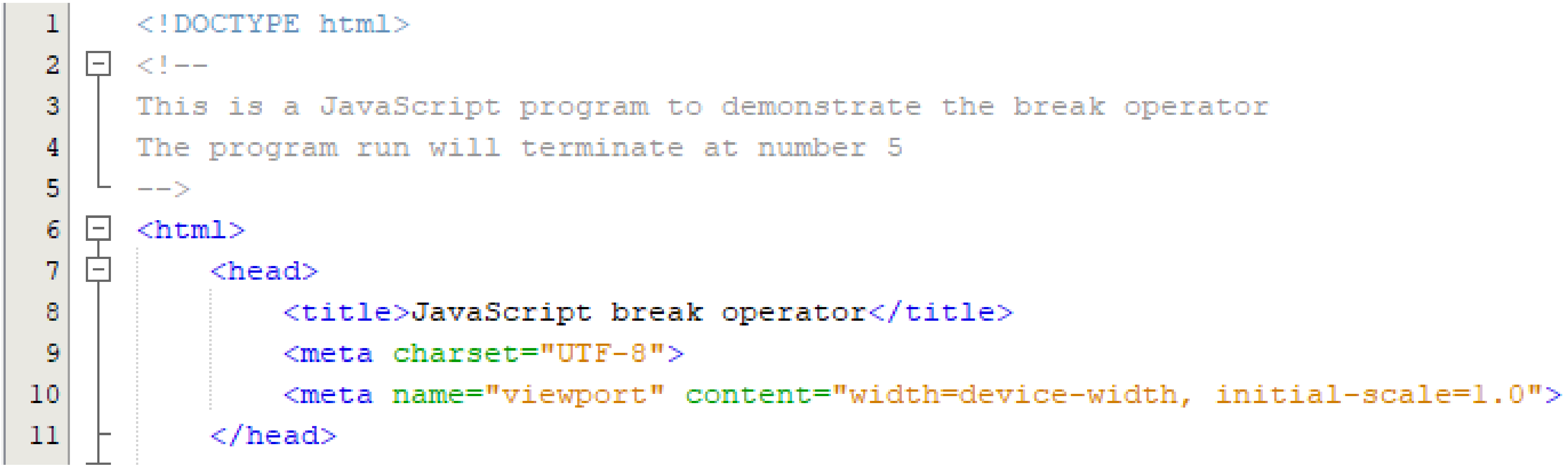
`Break` and `continue`

- PHP implements **break** , **continue** statements
- Sometimes you may want to let loops
  - Start without any control condition (and) allow the statements inside the brackets **{…}** to decide when to exit the loop
- There are two action statements that can be used inside loops (and also **`switch`** … **`case`** statements):
  - **`break`** **and** **`continue`**

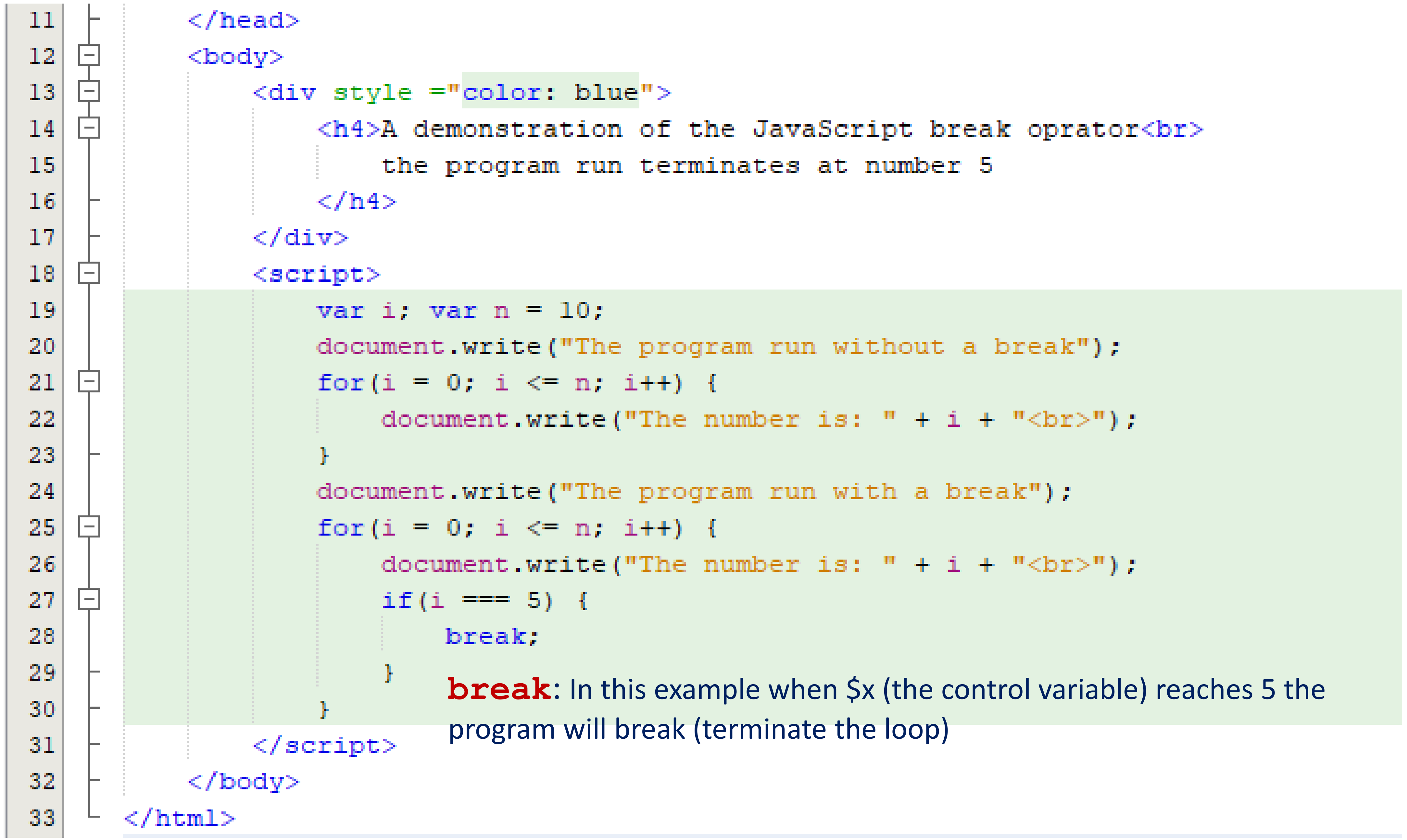
`break`

`break`

- The `break` statement:
  - Ends the program execution of the current structure and statement
  - Accepts an optional number argument
  - The numeric argument sets the parameters which tell the program when to interrupt the program flow
  - Sets how many execution of nested structures to be interrupted
  - It breaks the loop and continues executing the code after the loop

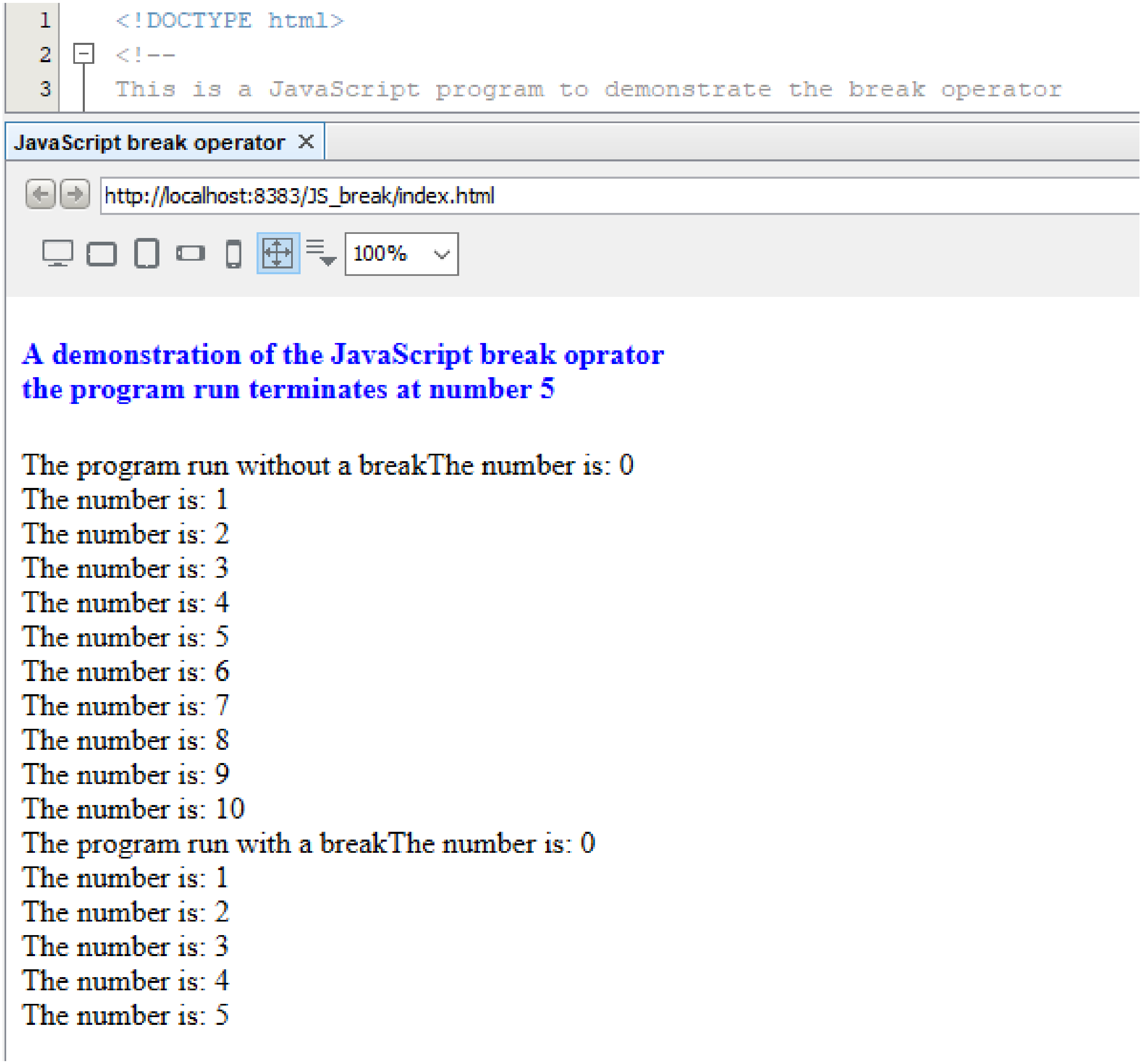


```html
<!DOCTYPE html>
<!--
This is a JavaScript program to demonstrate the break operator
The program run will terminate at number 5
-->
<html>
    <head>
        <title>JavaScript break operator</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
```



```
        </head>
        <body>
            <div style ="color: blue">
                <h4>A demonstration of the JavaScript break oprator<br>
                    the program run terminates at number 5
                </h4>
            </div>
            <script>
                var i; var n = 10;
                document.write("The program run without a break");
                for(i = 0; i <= n; i++) {
                    document.write("The number is: " + i + "<br>");
                }
                document.write("The program run with a break");
                for(i = 0; i <= n; i++) {
                    document.write("The number is: " + i + "<br>");
                    if(i === 5) {
                        break;
                    }
                }
            </script>
        </body>
    </html>
```

break: In this example when $x (the control variable) reaches 5 the program will break (terminate the loop)



```
<!DOCTYPE html>
<!--
This is a JavaScript program to demonstrate the break operator
```

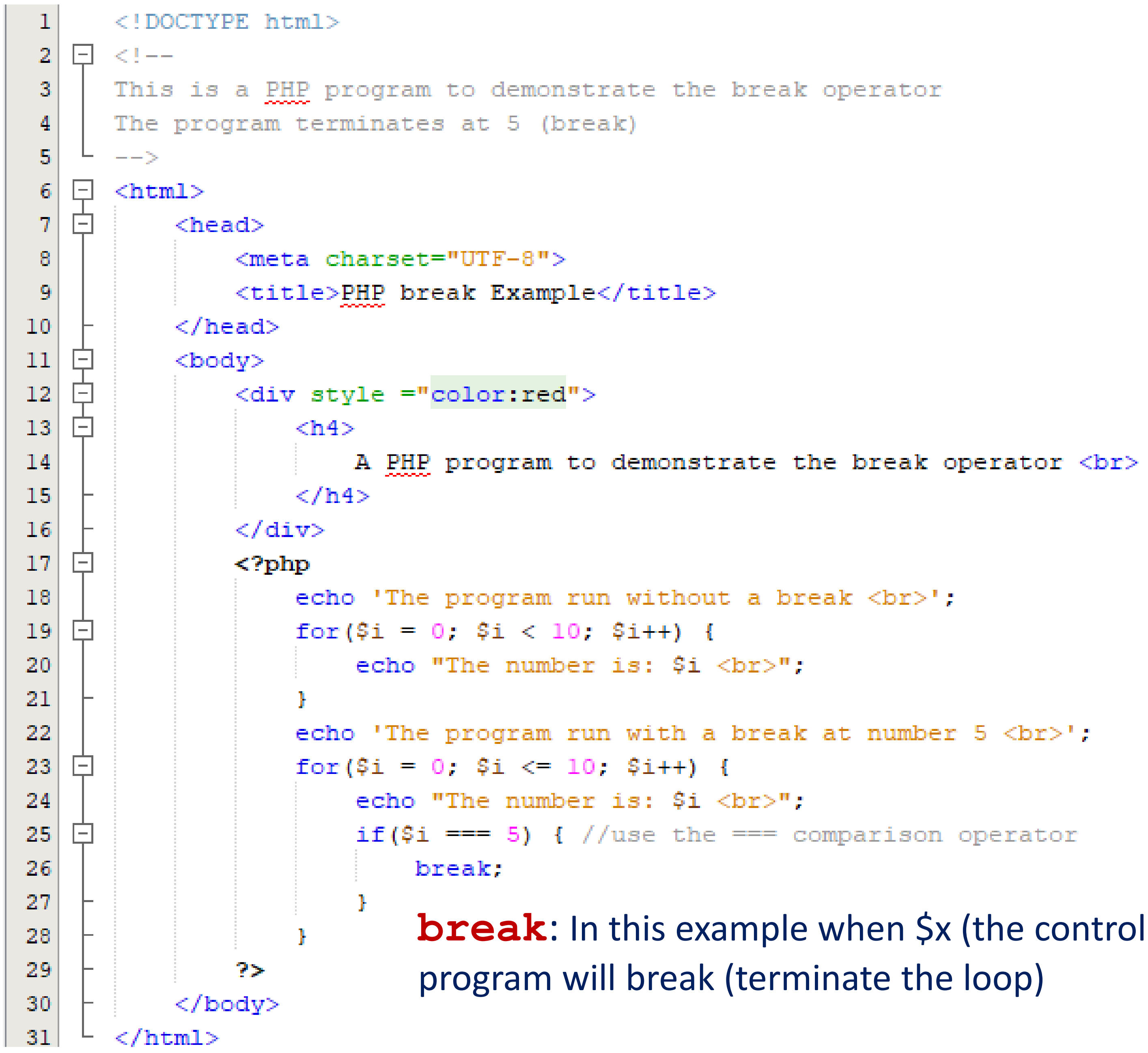
JavaScript break operator

http://localhost:8383/JS_break/index.html

100%

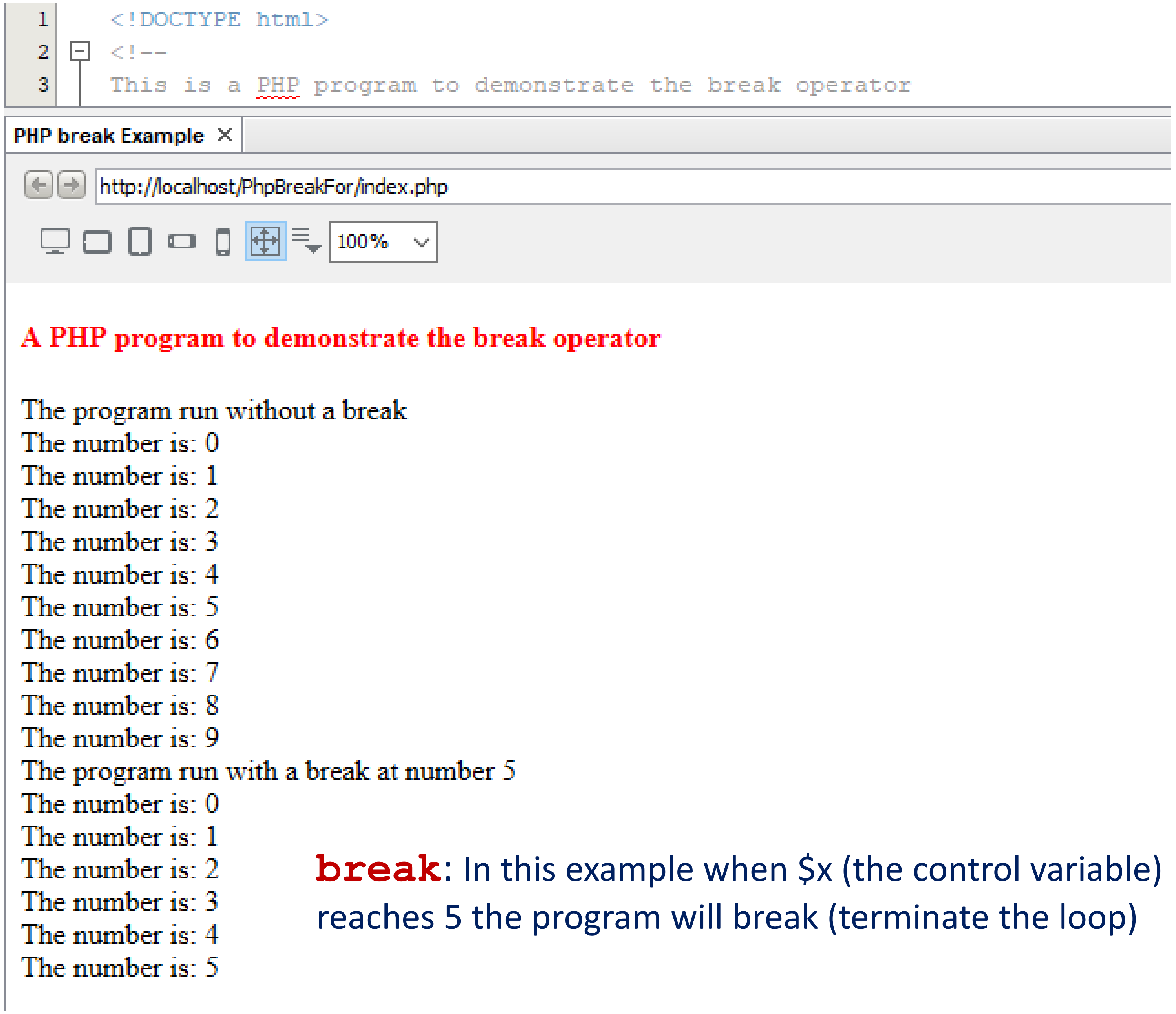
A demonstration of the JavaScript break oprator
the program run terminates at number 5

```
The program run without a breakThe 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
The number is: 10
The program run with a breakThe number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```



```
<!DOCTYPE html>
<!--
This is a PHP program to demonstrate the break operator
The program terminates at 5 (break)
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP break Example</title>
    </head>
    <body>
        <div style ="color:red">
            <h4>
                A PHP program to demonstrate the break operator <br>
            </h4>
        </div>
        <?php
            echo 'The program run without a break <br>';
            for($i = 0; $i < 10; $i++) {
                echo "The number is: $i <br>";
            }
            echo 'The program run with a break at number 5 <br>';
            for($i = 0; $i <= 10; $i++) {
                echo "The number is: $i <br>";
                if($i === 5) { //use the === comparison operator
                    break;
                }
            }
        ?>
    </body>
</html>
```

break: In this example when $x (the control variable) reaches 5 the program will break (terminate the loop)



```
<!DOCTYPE html>
<!--
This is a PHP program to demonstrate the break operator
```

PHP break Example

http://localhost/PhpBreakFor/index.php

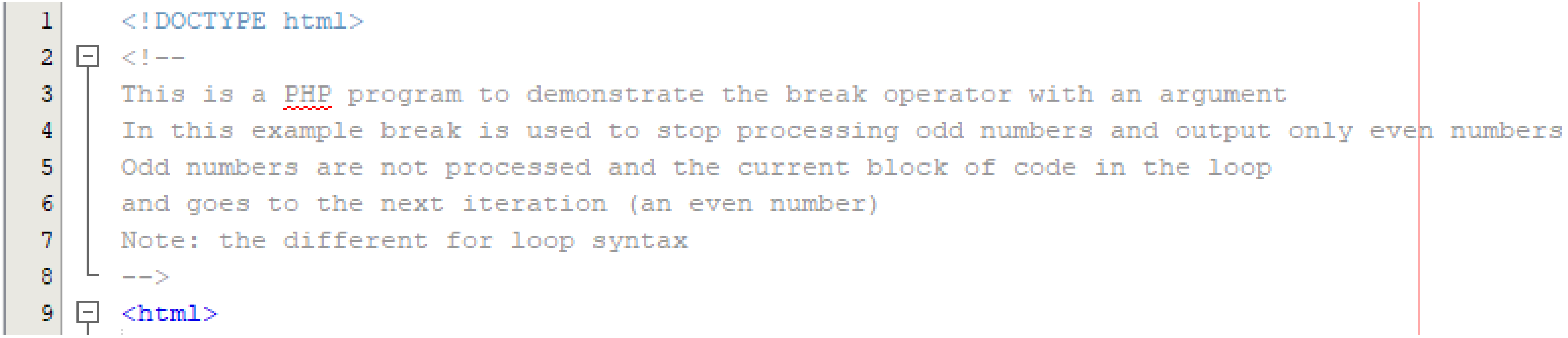
100%

```
A PHP program to demonstrate the break operator

The program run without a break
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
The program run with a break at number 5
The number is: 0
The number is: 1
The number is: 2
The number is: 3
The number is: 4
The number is: 5
```

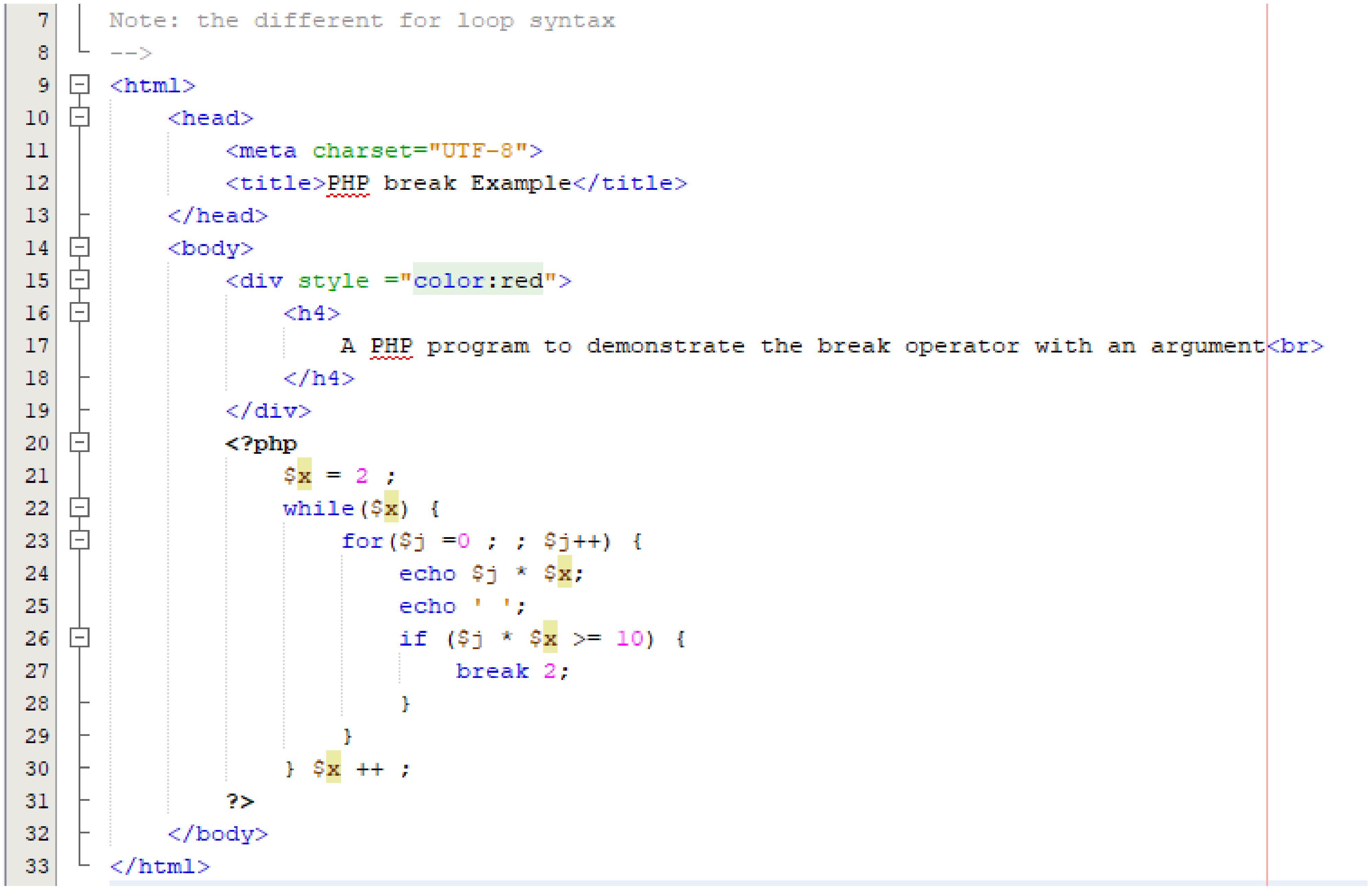
`break`: In this example when $x (the control variable) reaches 5 the program will break (terminate the loop)

`Break` with an argument

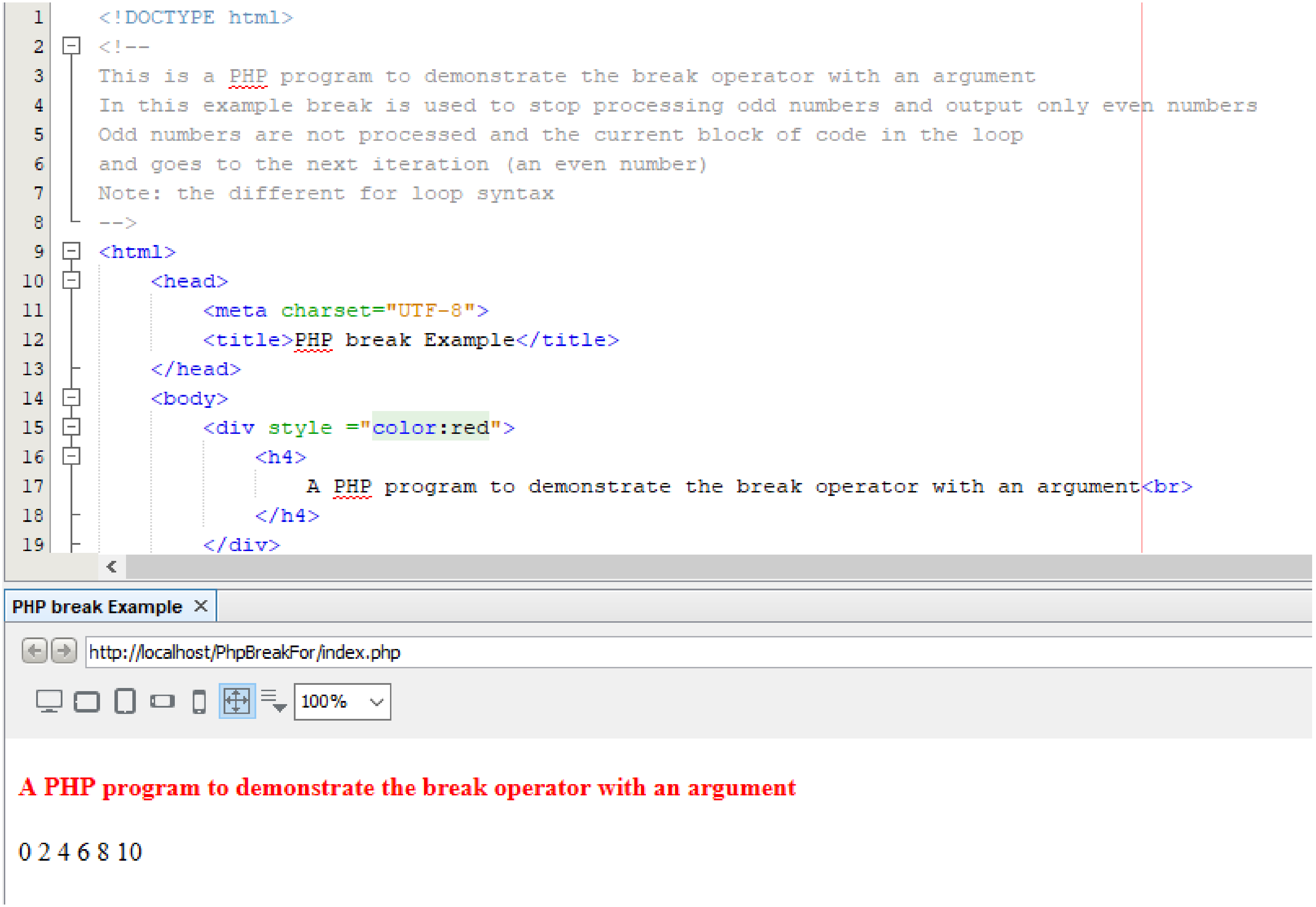


```
<!DOCTYPE html>
<!--
This is a PHP program to demonstrate the break operator with an argument
In this example break is used to stop processing odd numbers and output only even numbers
Odd numbers are not processed and the current block of code in the loop
and goes to the next iteration (an even number)
Note: the different for loop syntax
-->
<html>
```

- The program uses a while loop with a nested for loop:
- The output will be: 0 2 4 6 8 10
  - In this example break is used to stop processing odd numbers and output only even numbers.
  - Odd numbers are not processed and the current block of code in the loop and goes to the next iteration (an even number)
- Note: the different **`for`** loop syntax



```
Note: the different for loop syntax
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP break Example</title>
    </head>
    <body>
        <div style ="color:red">
            <h4>
                A PHP program to demonstrate the break operator with an argument<br>
            </h4>
        </div>
        <?php
            $x = 2 ;
            while($x) {
                for($j =0 ; ; $j++) {
                    echo $j * $x;
                    echo ' ';
                    if ($j * $x >= 10) {
                        break 2;
                    }
                }
            } $x ++ ;
        ?>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
This is a PHP program to demonstrate the break operator with an argument
In this example break is used to stop processing odd numbers and output only even numbers
Odd numbers are not processed and the current block of code in the loop
and goes to the next iteration (an even number)
Note: the different for loop syntax
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP break Example</title>
    </head>
    <body>
        <div style ="color:red">
            <h4>
                A PHP program to demonstrate the break operator with an argument<br>
            </h4>
        </div>
```

PHP break Example

http://localhost/PhpBreakFor/index.php

100%

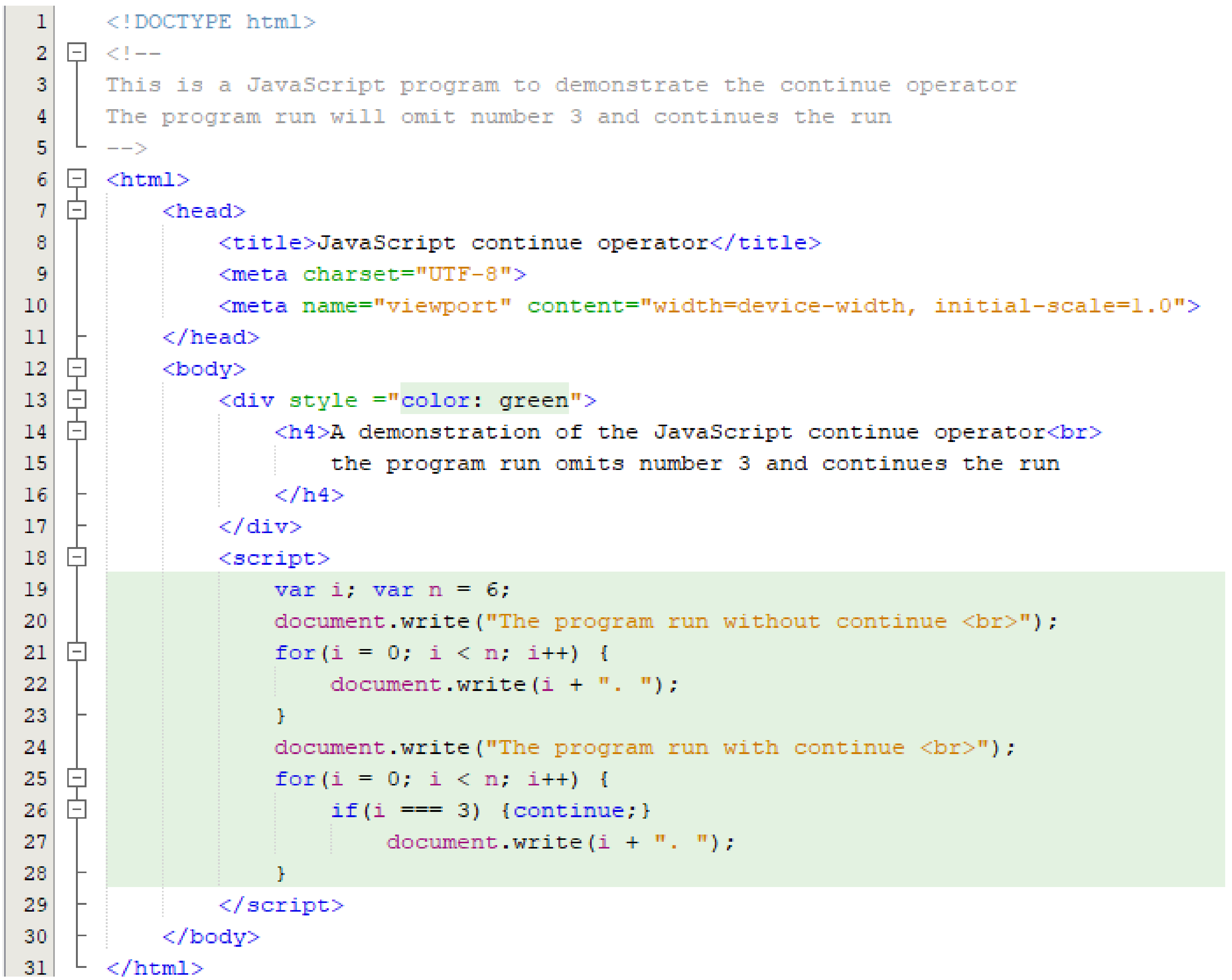
```
A PHP program to demonstrate the break operator with an argument

0 2 4 6 8 10
```

`continue`

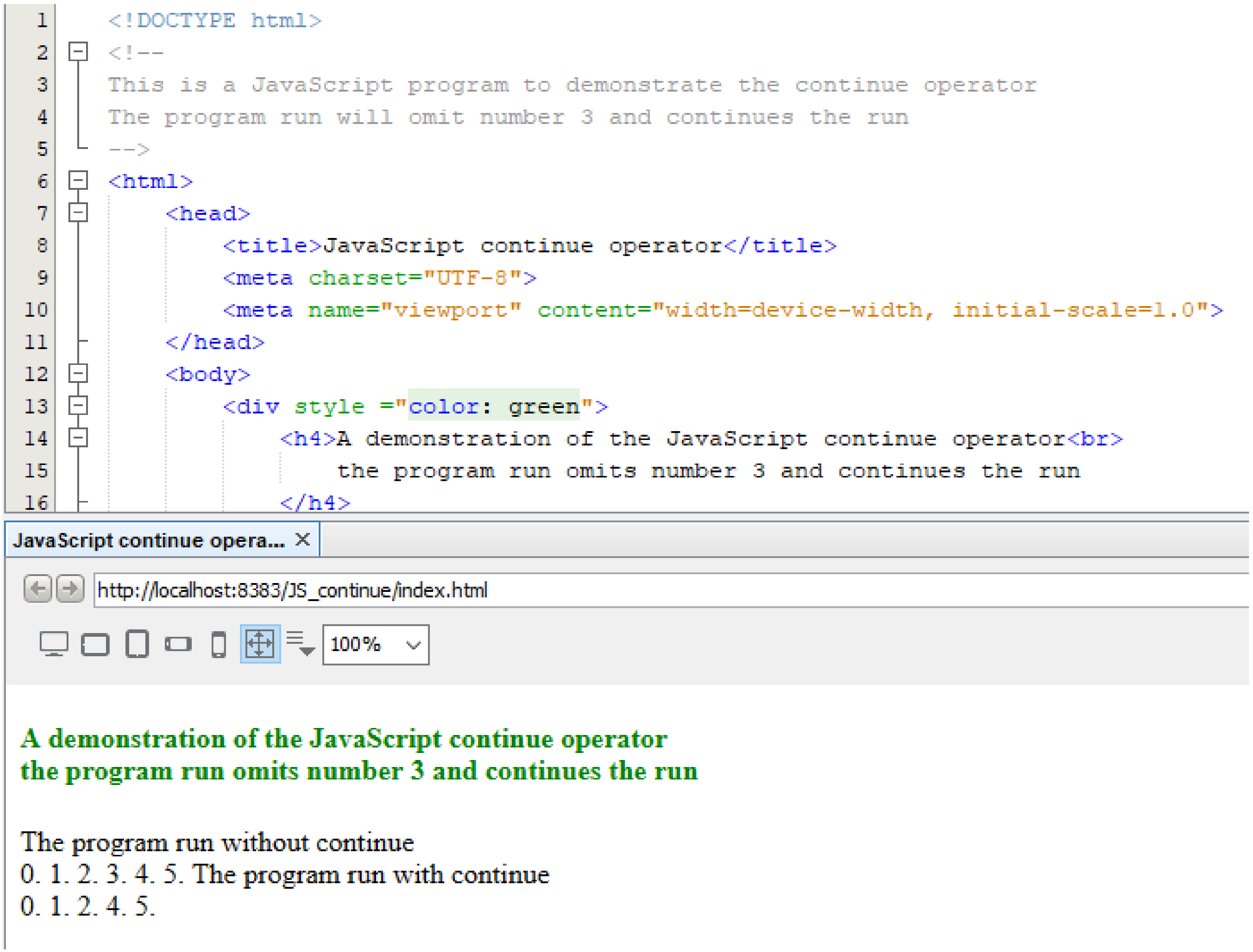
`continue`

- The `continue` statement:
  - Takes an optional numeric argument
  - The numeric argument sets the parameters which tell the program how many levels of enclosing loops it should skip to the end of
  - It breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop
  - The following JavaScript example omits number 3



```html
<!DOCTYPE html>
<!--
This is a JavaScript program to demonstrate the continue operator
The program run will omit number 3 and continues the run
-->
<html>
    <head>
        <title>JavaScript continue operator</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style ="color: green">
            <h4>A demonstration of the JavaScript continue operator<br>
                the program run omits number 3 and continues the run
            </h4>
        </div>
        <script>
            var i; var n = 6;
            document.write("The program run without continue <br>");
            for(i = 0; i < n; i++) {
                document.write(i + ". ");
            }
            document.write("The program run with continue <br>");
            for(i = 0; i < n; i++) {
                if(i === 3) {continue;}
                    document.write(i + ". ");
            }
        </script>
    </body>
</html>
```

- The output will be:
- 0. 1. 2. 4. 5.
- **`continue`**:
  - Is used to stop processing the current block of code in the loop and jump to the next iteration
  - In the example the number 3 is omitted



```
<!DOCTYPE html>
<!--
This is a JavaScript program to demonstrate the continue operator
The program run will omit number 3 and continues the run
-->
<html>
    <head>
        <title>JavaScript continue operator</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style ="color: green">
            <h4>A demonstration of the JavaScript continue operator<br>
                the program run omits number 3 and continues the run
            </h4>
```

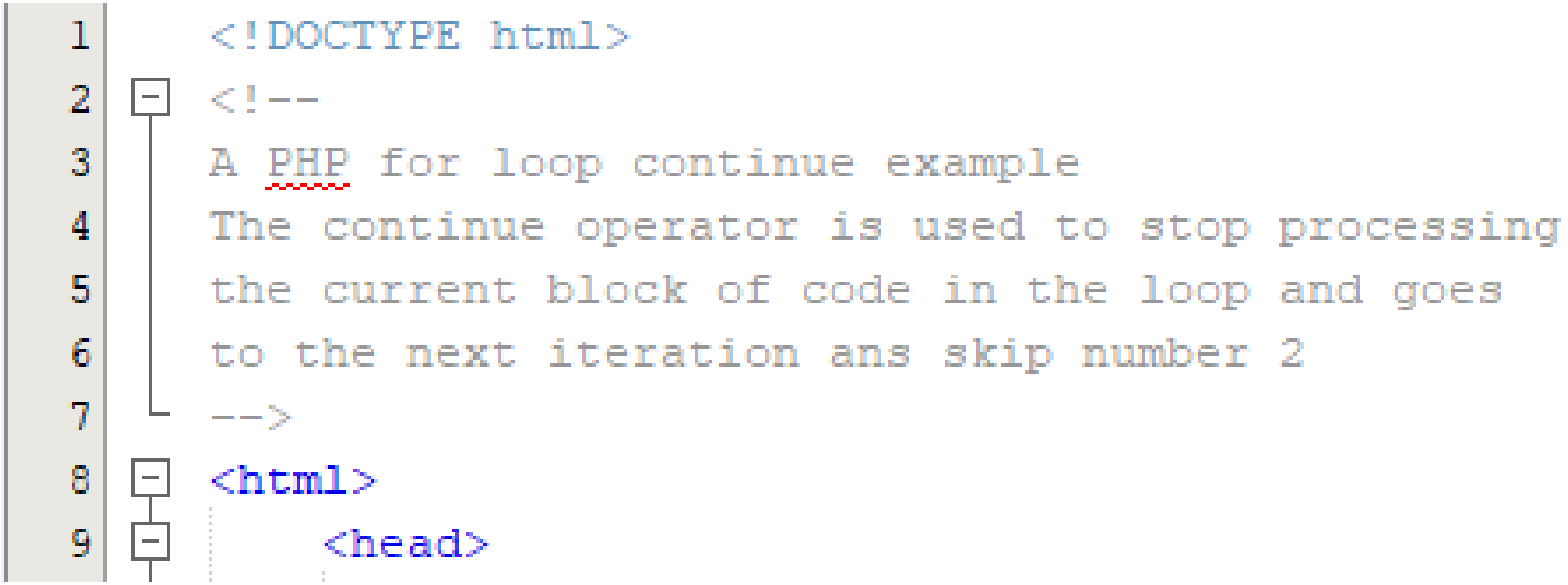
JavaScript continue opera...

http://localhost:8383/JS_continue/index.html

100%

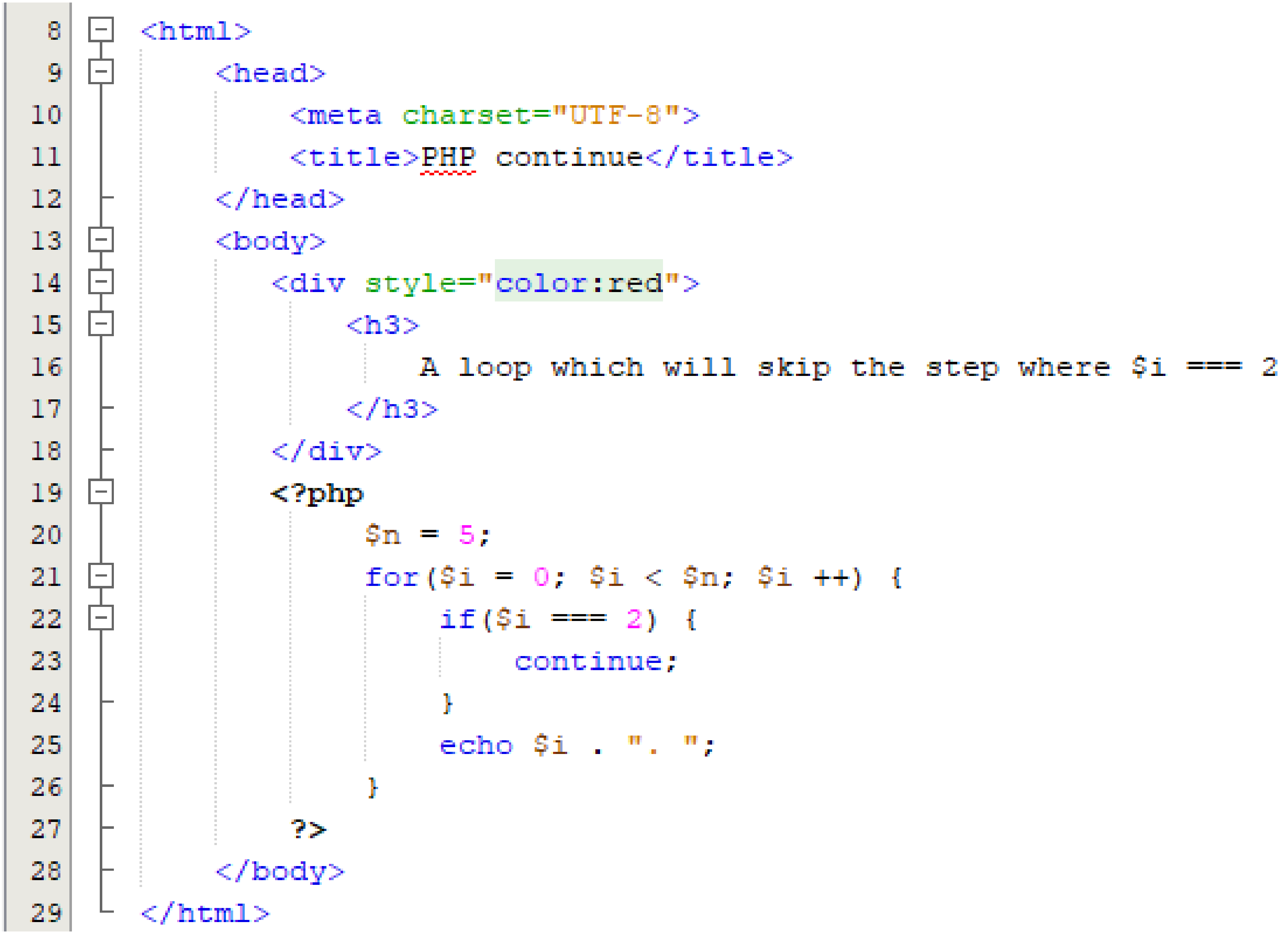
A demonstration of the JavaScript continue operator
the program run omits number 3 and continues the run

```
The program run without continue
0. 1. 2. 3. 4. 5. The program run with continue
0. 1. 2. 4. 5.
```

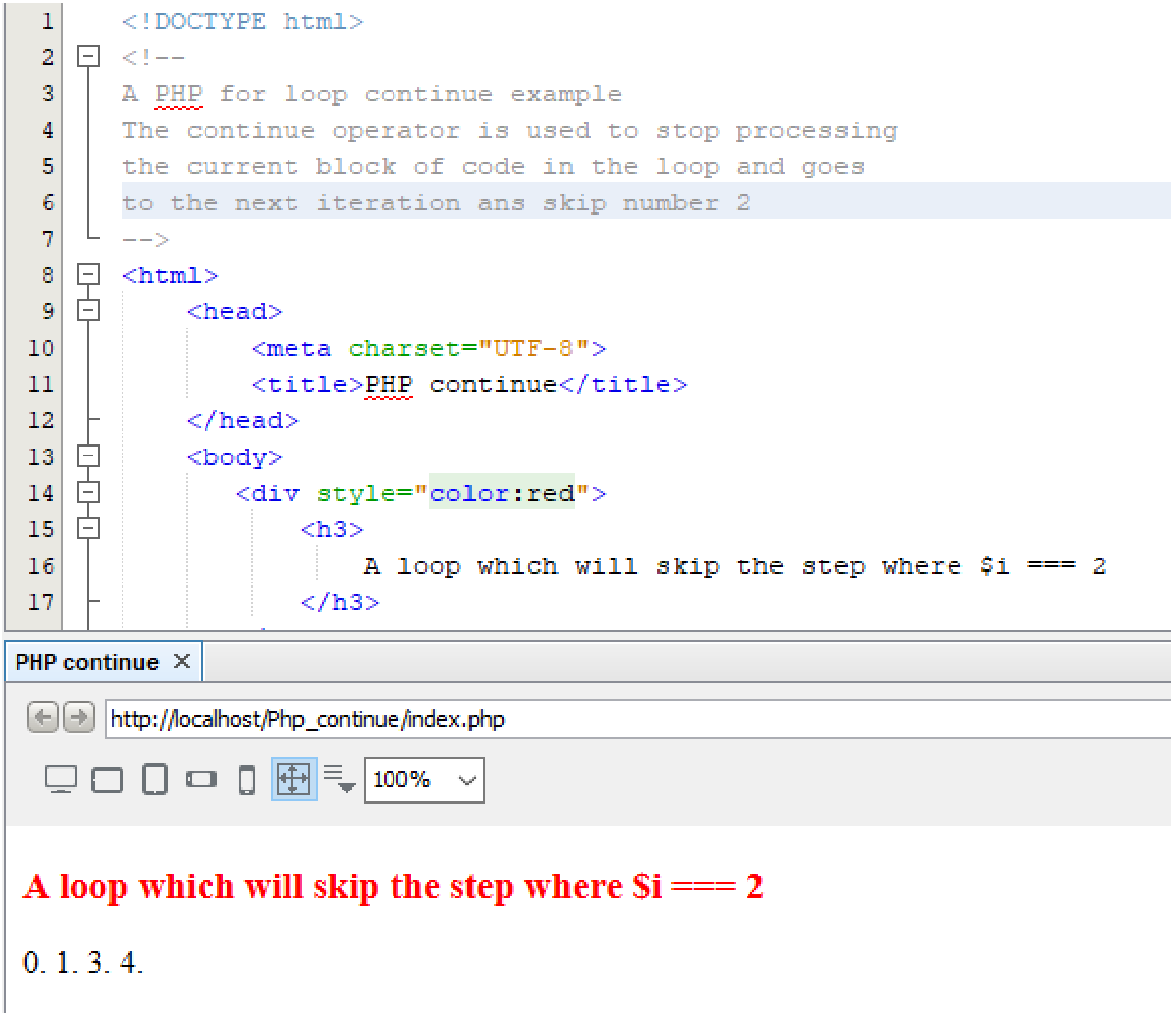


```
<!DOCTYPE html>
<!--
A PHP for loop continue example
The continue operator is used to stop processing
the current block of code in the loop and goes
to the next iteration ans skip number 2
-->
<html>
    <head>
```

- The output will be: 0. 1. 3. 4.
- **`continue`**: is used to stop processing the current block of code in the loop and goes to the next iteration - the program will omit number 2 and continue with number 3



```
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP continue</title>
    </head>
    <body>
       <div style="color:red">
           <h3>
               A loop which will skip the step where $i === 2
           </h3>
       </div>
       <?php
            $n = 5;
            for($i = 0; $i < $n; $i ++) {
                if($i === 2) {
                    continue;
                }
                echo $i . ". ";
            }
        ?>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
A PHP for loop continue example
The continue operator is used to stop processing
the current block of code in the loop and goes
to the next iteration ans skip number 2
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>PHP continue</title>
    </head>
    <body>
       <div style="color:red">
           <h3>
               A loop which will skip the step where $i === 2
           </h3>
```

A loop which will skip the step where $i === 2

```
0. 1. 3. 4.
```

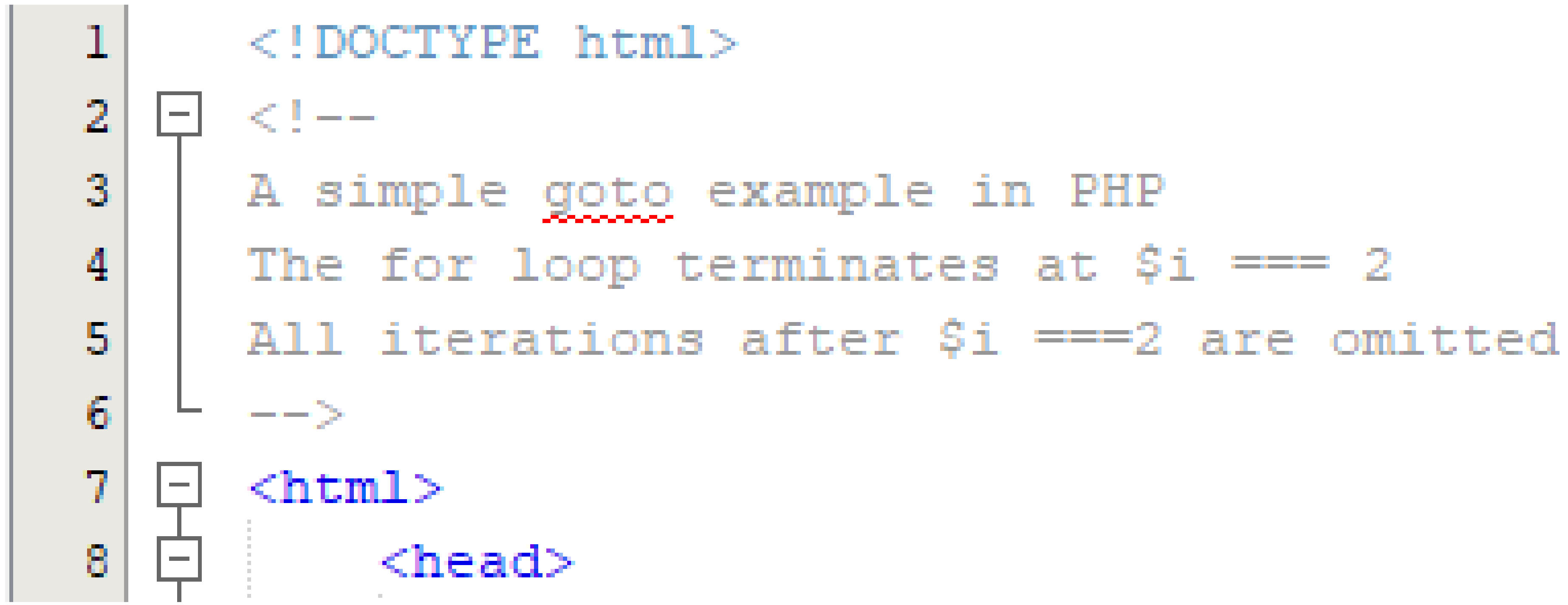
PHP
`goto` statement

`goto`

- The `goto` operator is a feature of the 'C' programming language but is NOT included in the 'Java' programming language
- JavaScript does have the `goto` keyword but it is only a reserved keyword
- In PHP **goto** is limited (it is not the full implementation as in 'C'):
  - The target label must be within the same file and context
  - you cannot jump out of (or into) a function or method
  - You cannot jump into any sort of loop or switch structure
  - You may jump out of any sort of loop or switch structure

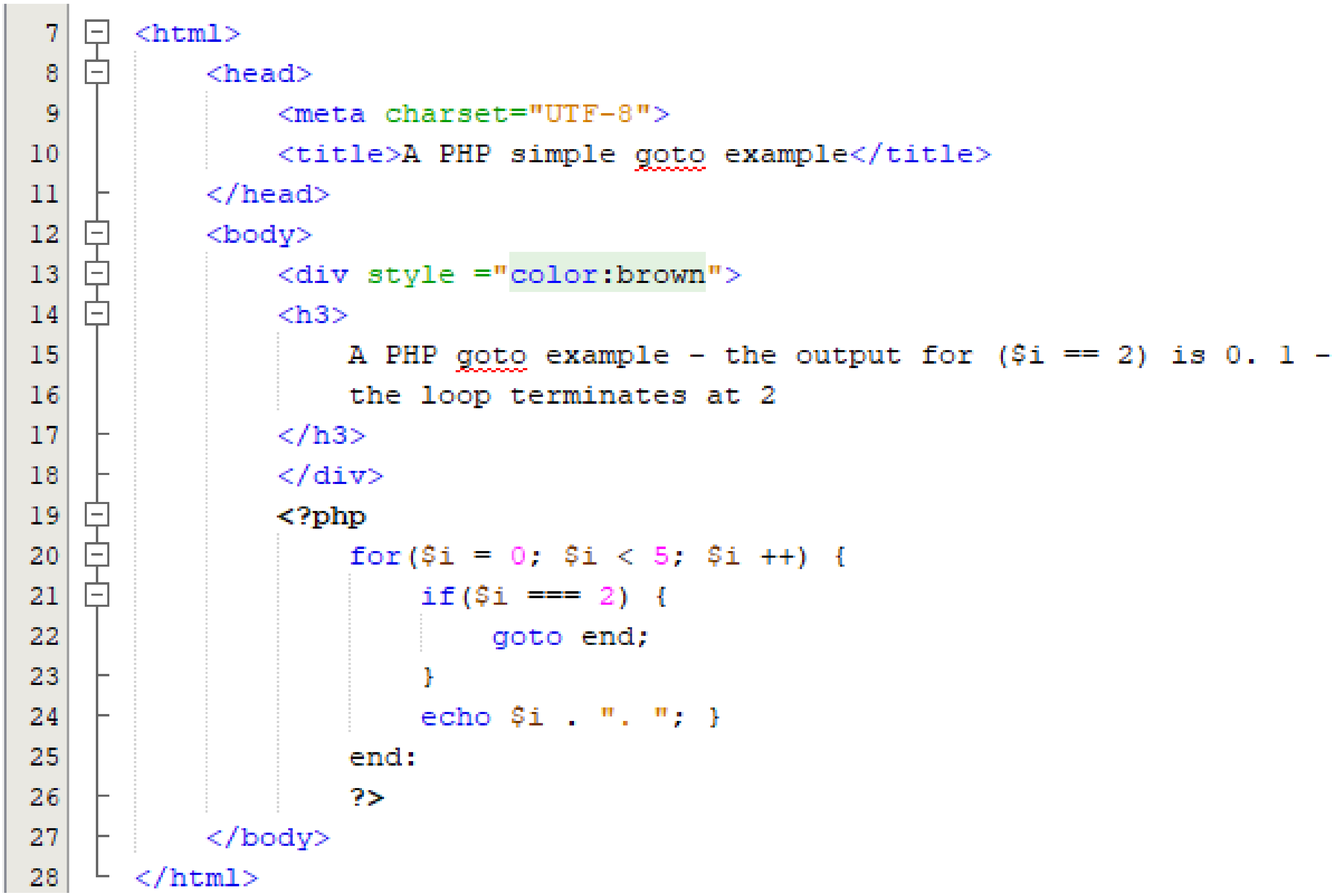
PHP `goto` statement

- The `goto` operator is designed to implement *unconditional transition*
  - It is used to force the program flow to *jump* to another location, area, or section within the program code (a location where you must go to in the current program)
  - It is indicated by the `goto` label (line 22 in the example PHP script)
- The intended location where the program must *jump* to is stated following the `goto` label

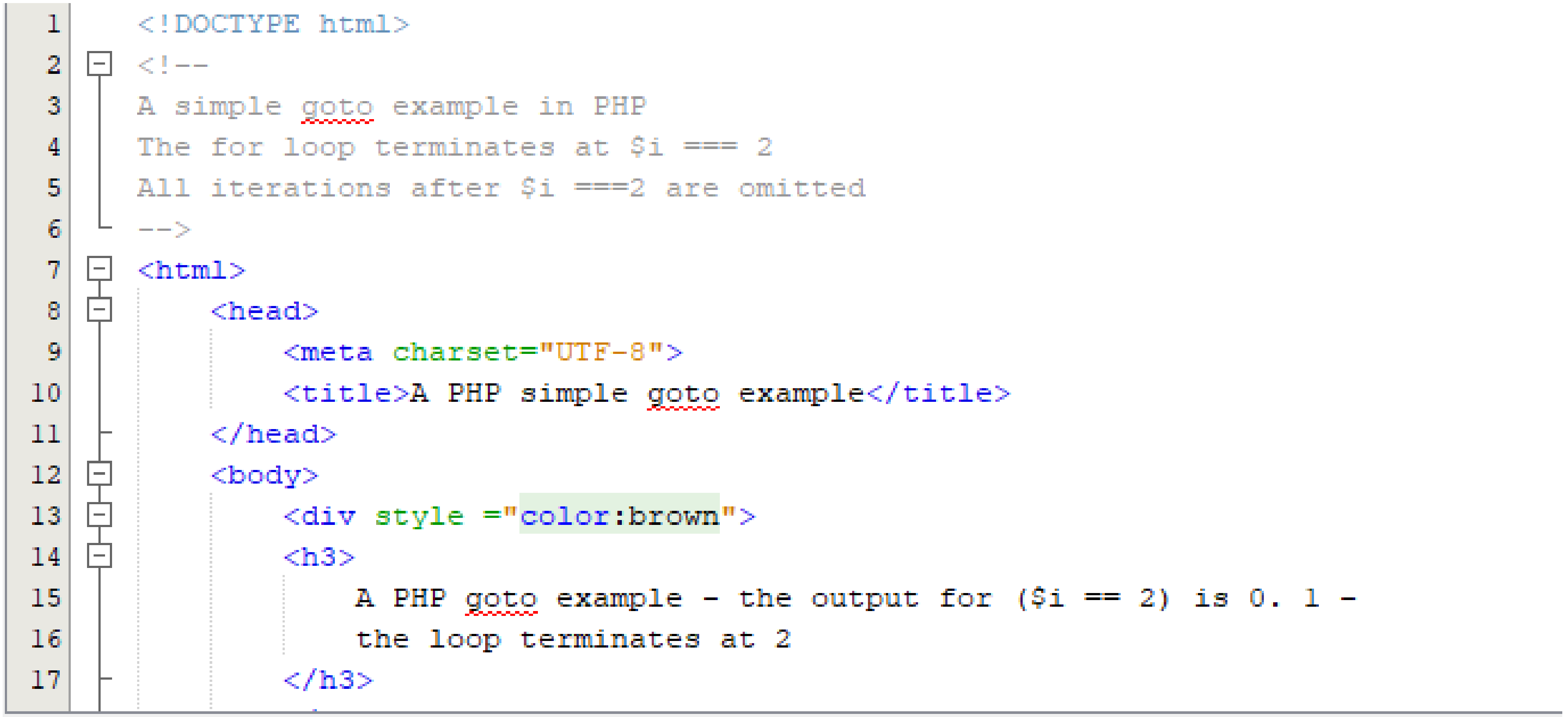


```
<!DOCTYPE html>
<!--
A simple goto example in PHP
The for loop terminates at $i === 2
All iterations after $i ===2 are omitted
-->
<html>
    <head>
```

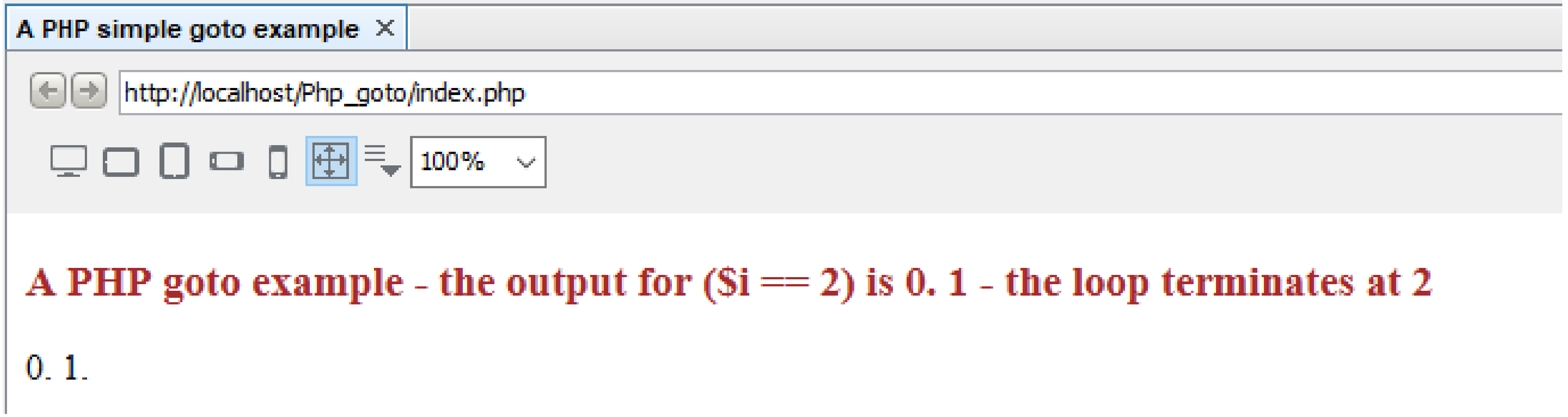
- The output will be: 0. 1.
- `goto`: is used to stop processing the current block of code and jumps to the `goto` location
  - This program will terminate at line 23
  - The code is: `goto` end;



```
<html>
    <head>
        <meta charset="UTF-8">
        <title>A PHP simple goto example</title>
    </head>
    <body>
        <div style ="color:brown">
        <h3>
            A PHP goto example - the output for ($i == 2) is 0. 1 -
            the loop terminates at 2
        </h3>
        </div>
        <?php
            for($i = 0; $i < 5; $i ++) {
                if($i === 2) {
                    goto end;
                }
                echo $i . ". "; }
            end:
            ?>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
A simple goto example in PHP
The for loop terminates at $i === 2
All iterations after $i ===2 are omitted
-->
<html>
    <head>
        <meta charset="UTF-8">
        <title>A PHP simple goto example</title>
    </head>
    <body>
        <div style ="color:brown">
        <h3>
            A PHP goto example - the output for ($i == 2) is 0. 1 -
            the loop terminates at 2
        </h3>
```



A PHP simple goto example
http://localhost/Php_goto/index.php
100%

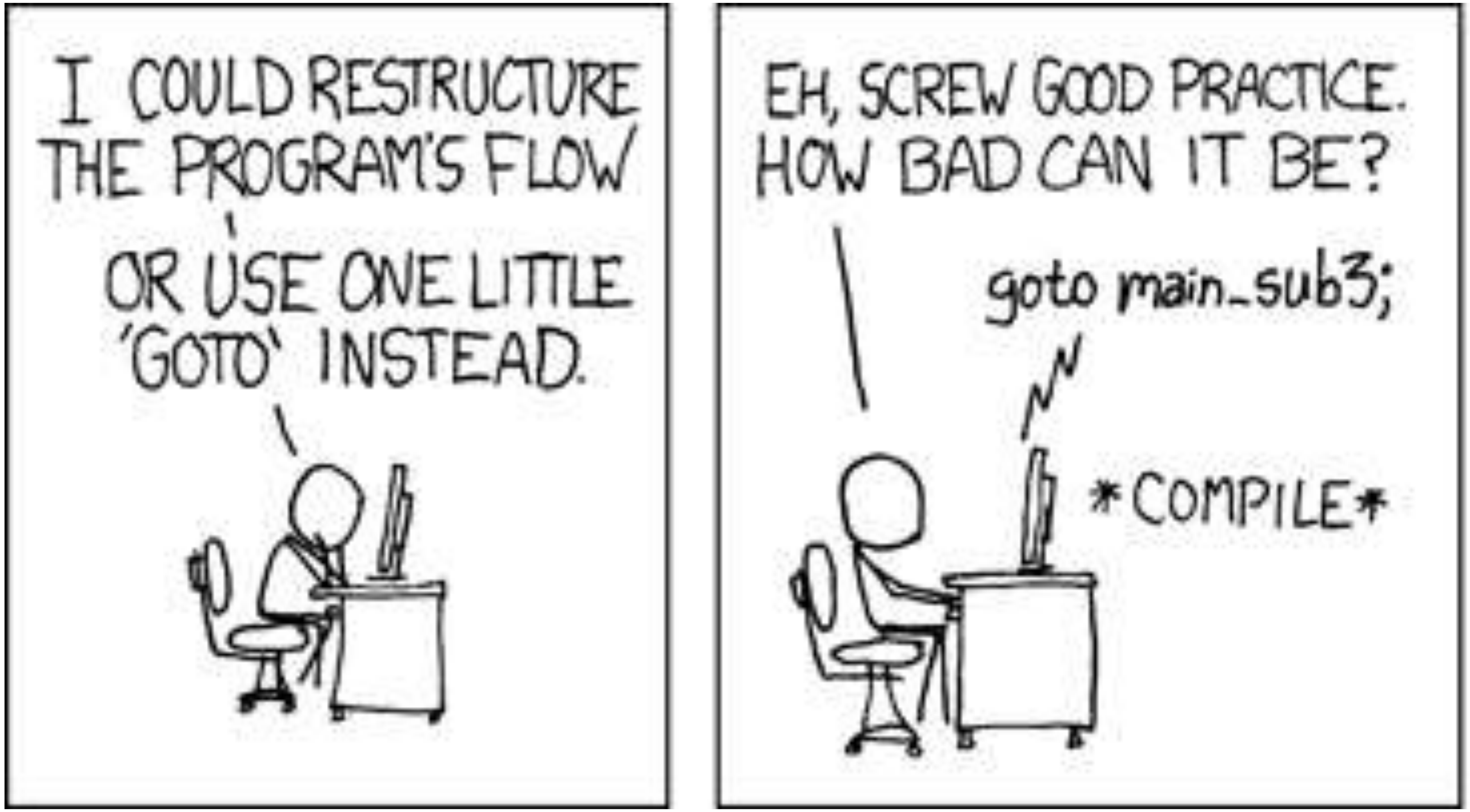
A PHP goto example - the output for ($i == 2) is 0. 1 - the loop terminates at 2

```
0. 1.
```

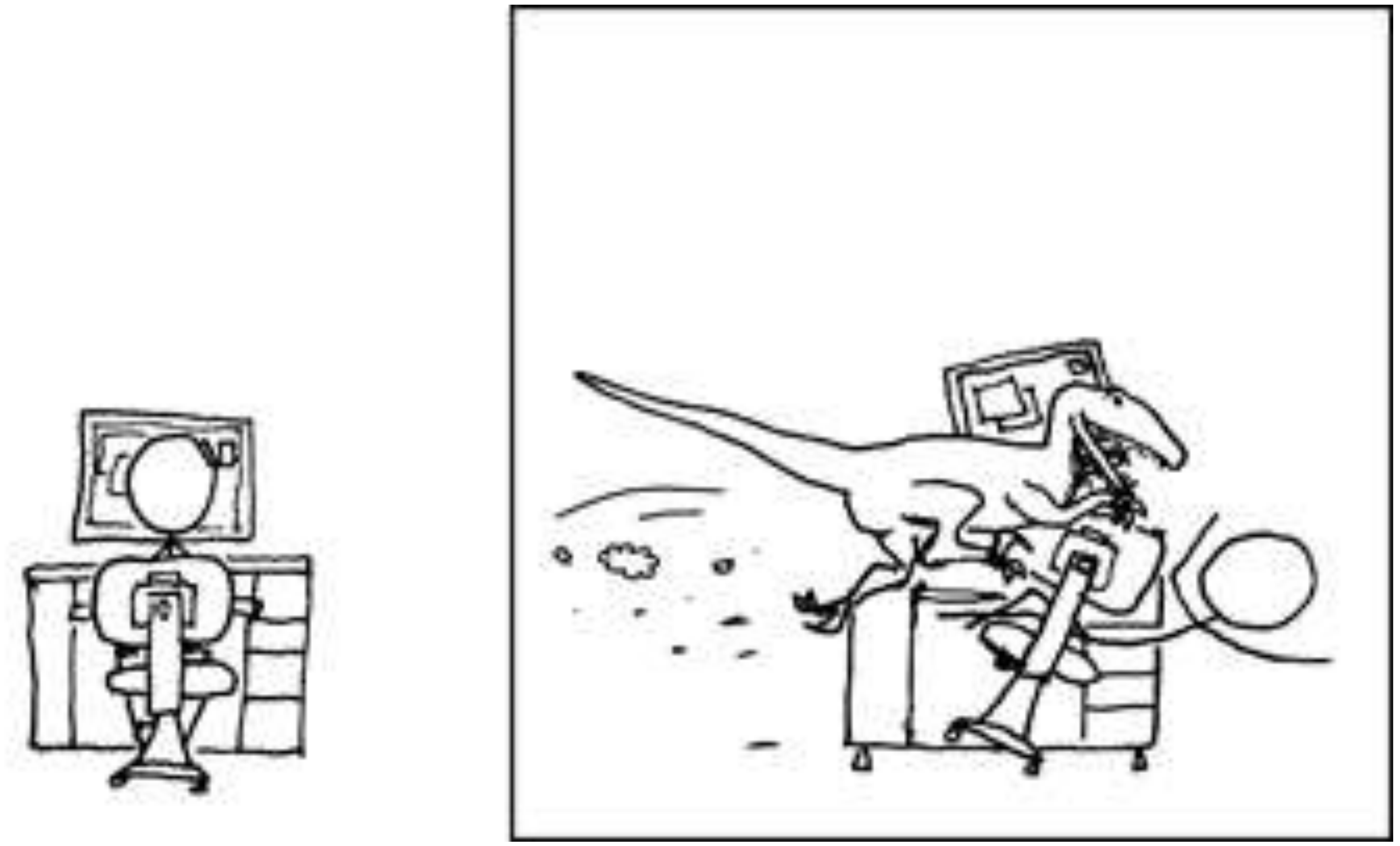
`goto`

- Implementation is simple and while `goto` may at be useful it is not generally recommended because in ‘C’:
  - It can result is unforeseen errors
  - It can result in program code that is not clear or understandable
  - Runtime errors can be very difficult to find in debugging
  - It can result in ‘*spaghetti*’ program code
  - In PHP a common use is to use a `goto` in place of a multi-level `break`

Goto?


I COULD RESTRUCTURE THE PROGRAM'S FLOW
OR USE ONE LITTLE 'GOTO' INSTEAD.
EH, SCREW GOOD PRACTICE. HOW BAD CAN IT BE?
goto main_sub3;
COMPILE

Goto?



Review

- In this session we have introduced:
  - Iteration (loops) in PHP
  - The use of **`goto`** in place of a multi-level **`break`** in PHP
- There are similarities and differences between JavaScript and PHP
  - The differences are mainly in the language syntax
  - Using the incorrect syntax will produce errors which are hard to find
- In the following tutorials we will introduce:
  - Working with MySQL server in a PHP script
  - Database basics, SQL, and MySql server in the NetBeans IDE