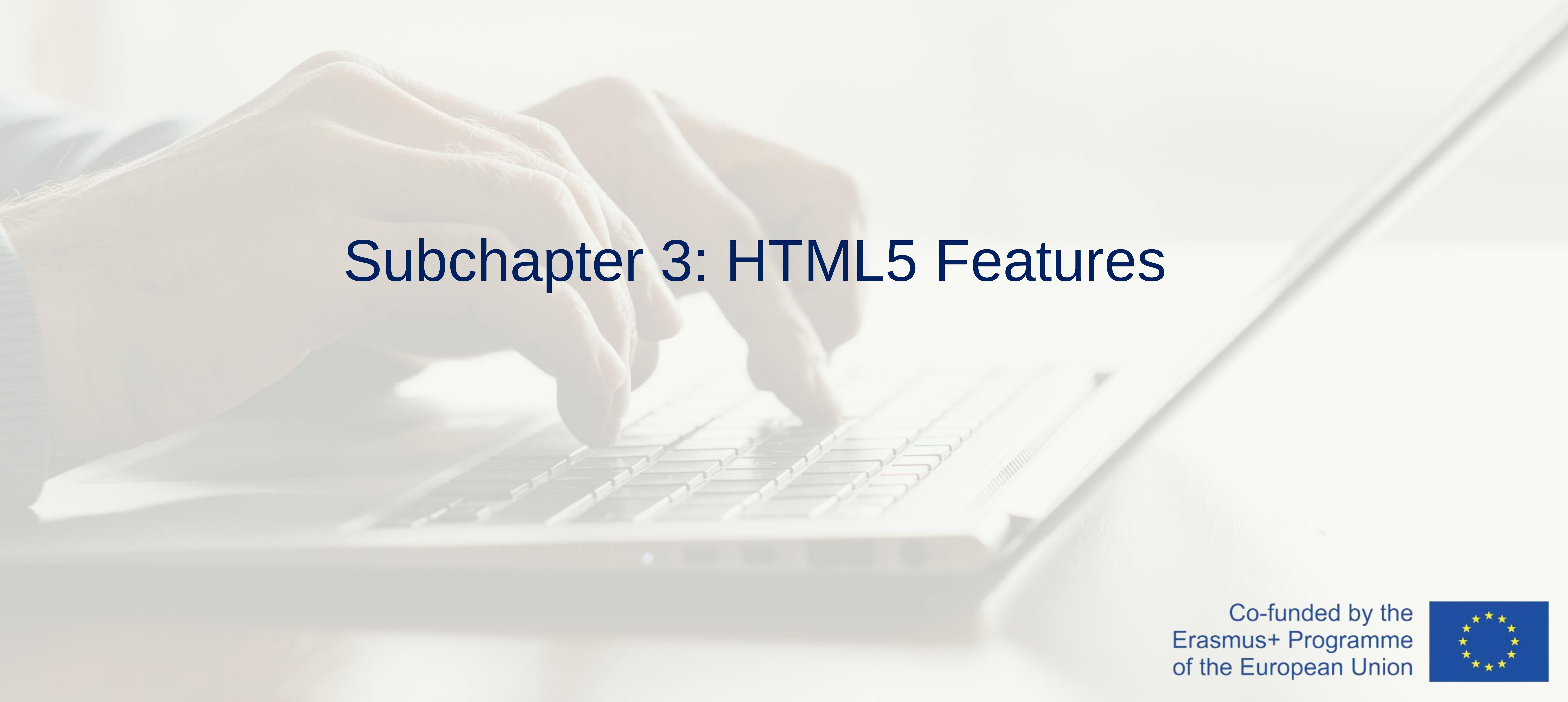
Co-funded by the
Erasmus+ Programme
of the European Union

code4sp
coding for social promotion

HTML Trainer Materials
Subchapter 3 – HTML 5 Features

WP3: Code4SP Training Materials

Prepared by:



Subchapter 3: HTML5 Features

Co-funded by the Erasmus+ Programme of the European Union

HTML5 Features

In this subchapter, the features of the fifth (and last) major HTML version will be explained.

HTML5 New Input Types

- HTML5 introduces several new <input> types like email, date, time, color, range, and so on, with the goal of improving the user experience and to make the forms more interactive.
- Nevertheless, if a browser failed to recognize these new input types, it will consider them a normal text box.

HTML5 New Input Types

The following are some new input types:

- color
- date
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

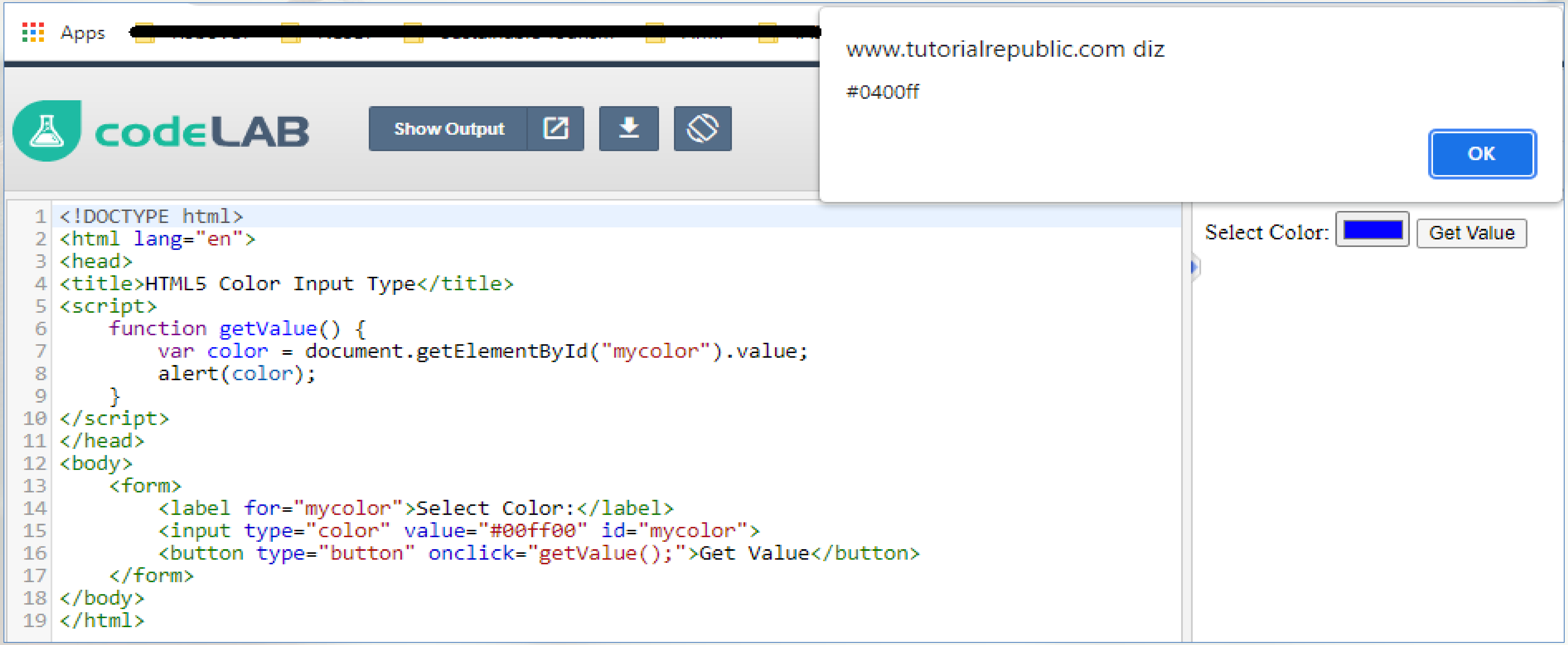
HTML5 New Input Types

The Color Input

It allows to select a colour from a colour picker and gives information about the colour value in hexadecimal format (e.g., #000000, which is black, the default colour if the user does not specify a value), as verified in the next page’s figure. It should be noted that the color input is supported in all major modern web browsers (Firefox, Chrome, Opera, Safari (12.1+), Edge (14+)), but it is not supported by the Microsoft Internet Explorer and older versions of Apple Safari browsers.

HTML5 New Input Types

The Color Input



[redacted]

www.tutorialrepublic.com diz

#0400ff

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Color Input Type</title>
<script>
    function getValue() {
        var color = document.getElementById("mycolor").value;
        alert(color);
    }
</script>
</head>
<body>
    <form>
        <label for="mycolor">Select Color:</label>
        <input type="color" value="#00ff00" id="mycolor">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

Select Color: 🟦 Get Value

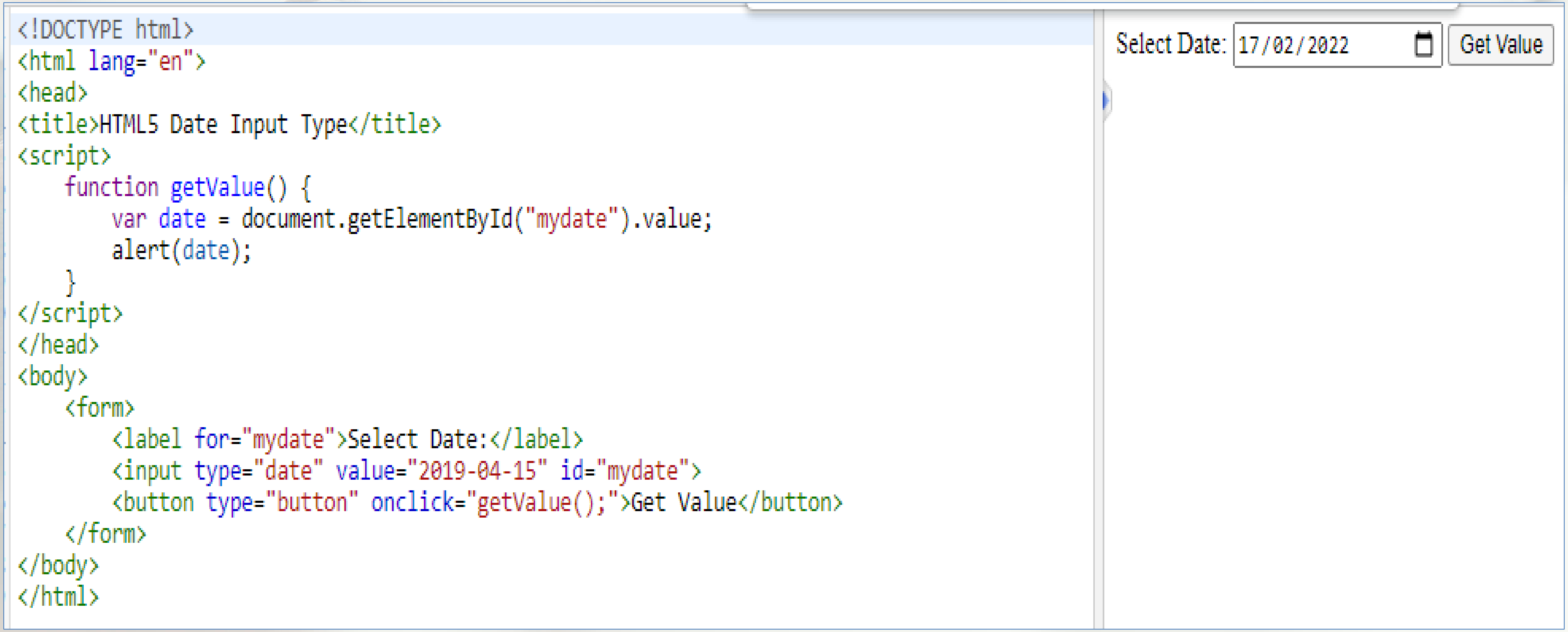
HTML5 New Input Types

The Date Input

It allows the user to select a date from a drop-down calendar, in which he/she may choose the year, month and day (but not time). This feature is also supported by most browsers, except for Internet Explorer and Safari.

HTML5 New Input Types

The Date Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Date Input Type</title>
<script>
    function getValue() {
        var date = document.getElementById("mydate").value;
        alert(date);
    }
</script>
</head>
<body>
    <form>
        <label for="mydate">Select Date:</label>
        <input type="date" value="2019-04-15" id="mydate">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

Select Date: 17/02/2022 Get Value

code4sp
coding for social promotion

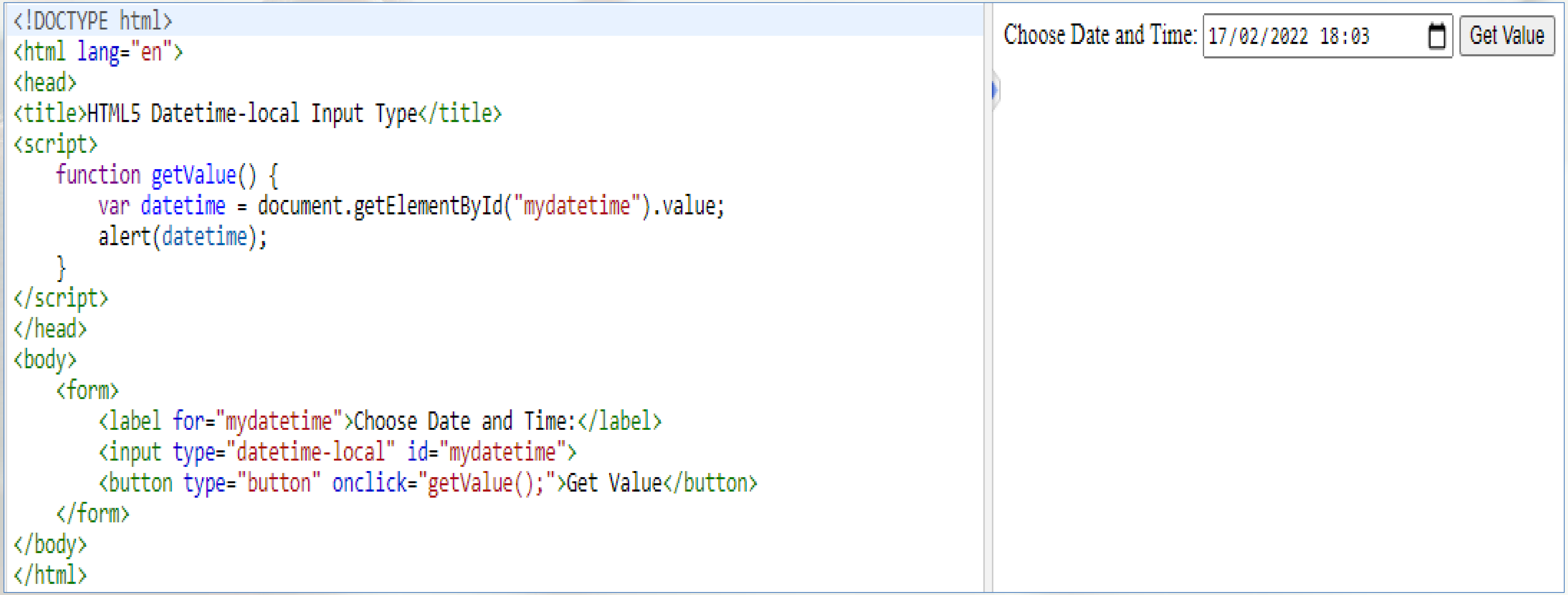
HTML5 New Input Types

The datetime-local Input

It makes available to the user to select both local date and time, including the year, month, and day including the time in hours and minutes. This input is supported by Safari, Firefox and IE, but not by Chrome, Edge and Opera.

HTML5 New Input Types

The datetime-local Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Datetime-local Input Type</title>
<script>
    function getValue() {
        var datetime = document.getElementById("mydatetime").value;
        alert(datetime);
    }
</script>
</head>
<body>
    <form>
        <label for="mydatetime">Choose Date and Time:</label>
        <input type="datetime-local" id="mydatetime">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

Choose Date and Time: 17/02/2022 18:03 Get Value

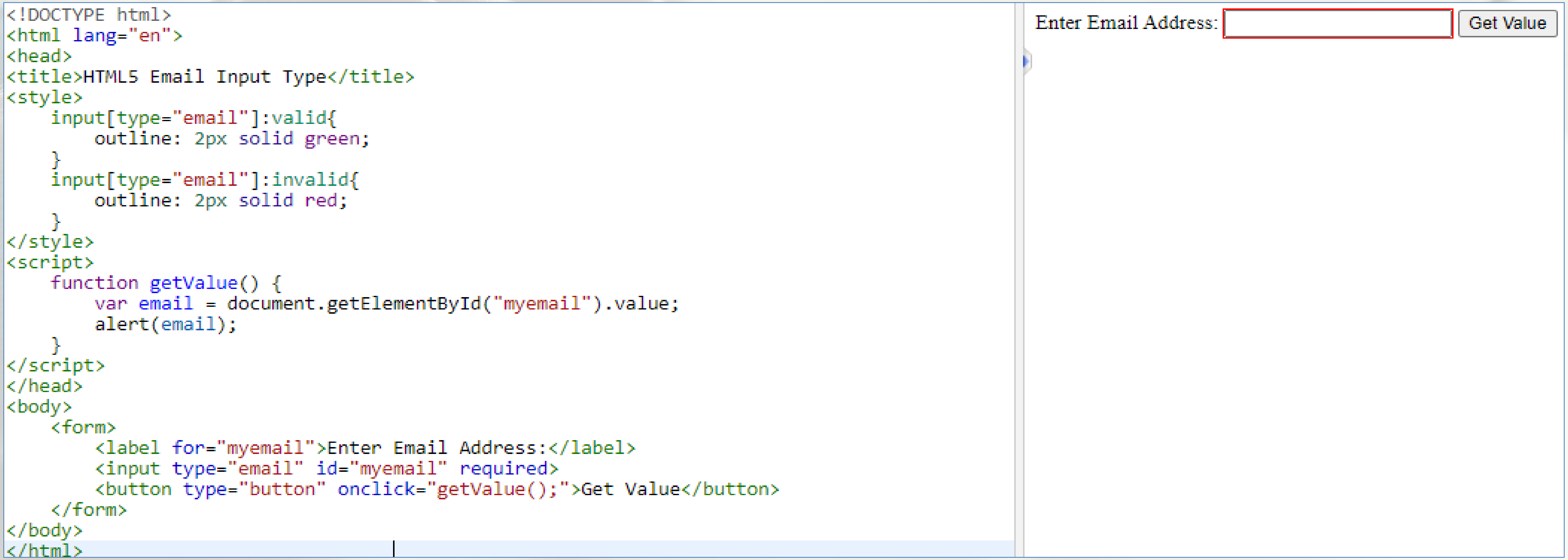
HTML5 New Input Types

The email Input

To allow the user to enter his/her e-mail address, email is the preferred input type. It is quite like a standard text input type, but if it is applied in combination with the required attribute, browsers may search for the patterns to guarantee a properly formatted e-mail address should be entered. The email input field can be styled for different validation states, when a value is entered using the :valid, :invalid or :required pseudo-classes. This input type is supported by all major browsers.

HTML5 New Input Types

The email Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Email Input Type</title>
<style>
    input[type="email"]:valid{
        outline: 2px solid green;
    }
    input[type="email"]:invalid{
        outline: 2px solid red;
    }
</style>
<script>
    function getValue() {
        var email = document.getElementById("myemail").value;
        alert(email);
    }
</script>
</head>
<body>
    <form>
        <label for="myemail">Enter Email Address:</label>
        <input type="email" id="myemail" required>
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

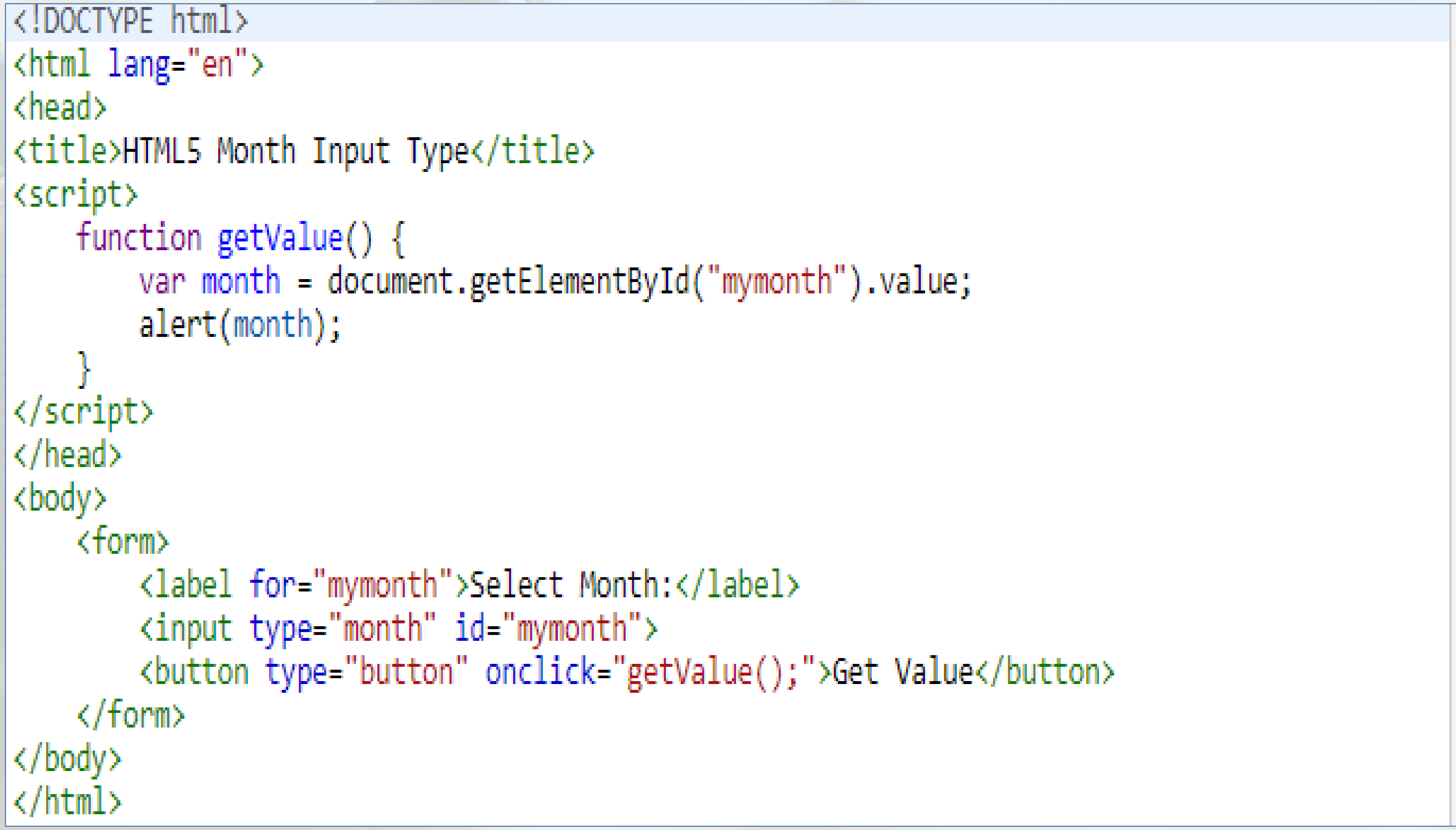
HTML5 New Input Types

The month Input

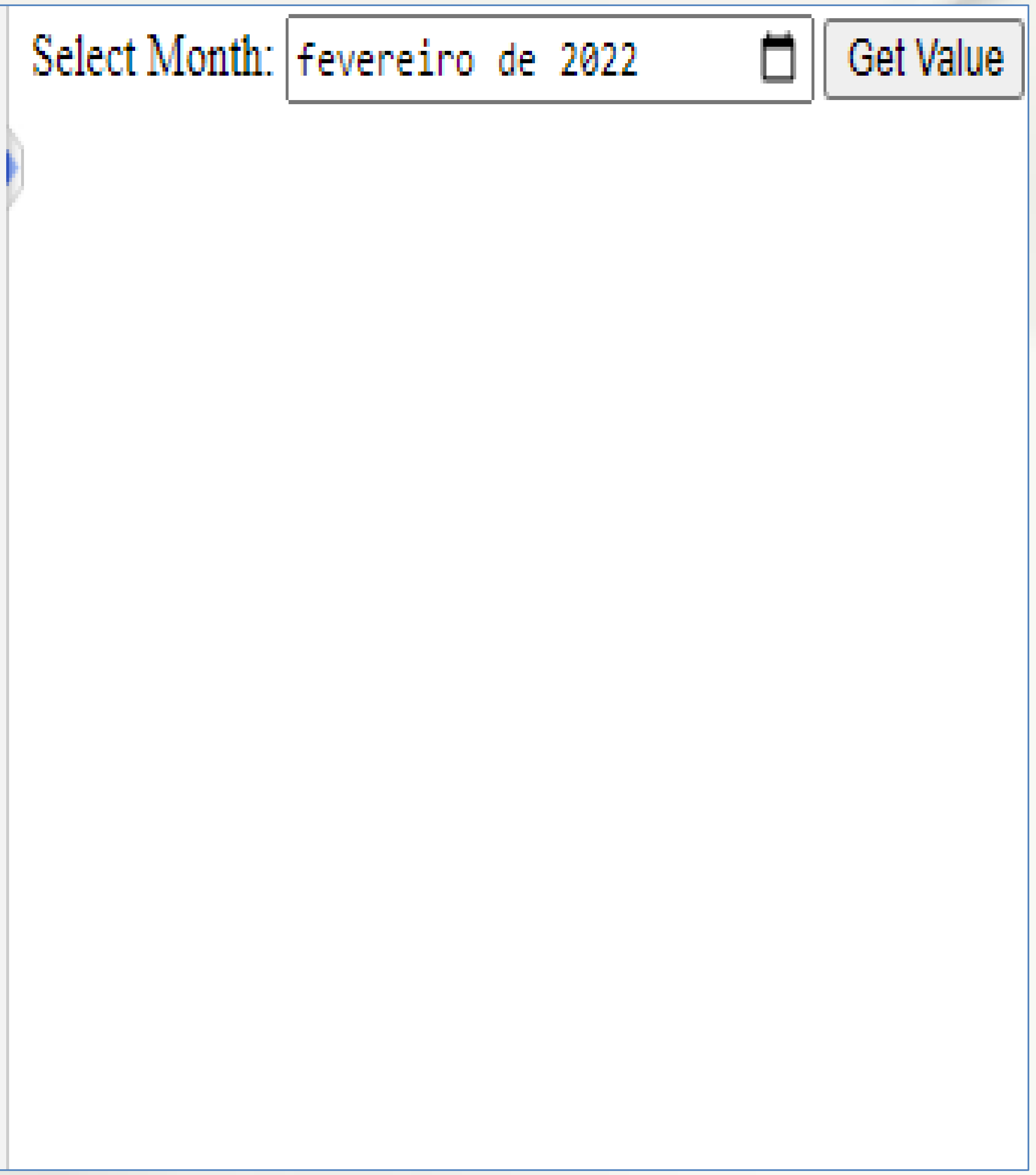
this is a very similar feature to the previous time ones since it allows to select a **month** and year from a drop-down calendar (being ‘YYYY’ the year and ‘’MM’’ the month. It should be noted that this is not supported by Firefox, Safari and Internet Explorer. Only Chrome, Edge and Opera browsers support it.

HTML5 New Input Types

The month Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Month Input Type</title>
<script>
    function getValue() {
        var month = document.getElementById("mymonth").value;
        alert(month);
    }
</script>
</head>
<body>
    <form>
        <label for="mymonth">Select Month:</label>
        <input type="month" id="mymonth">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```



Select Month: fevereiro de 2022

HTML5 New Input Types

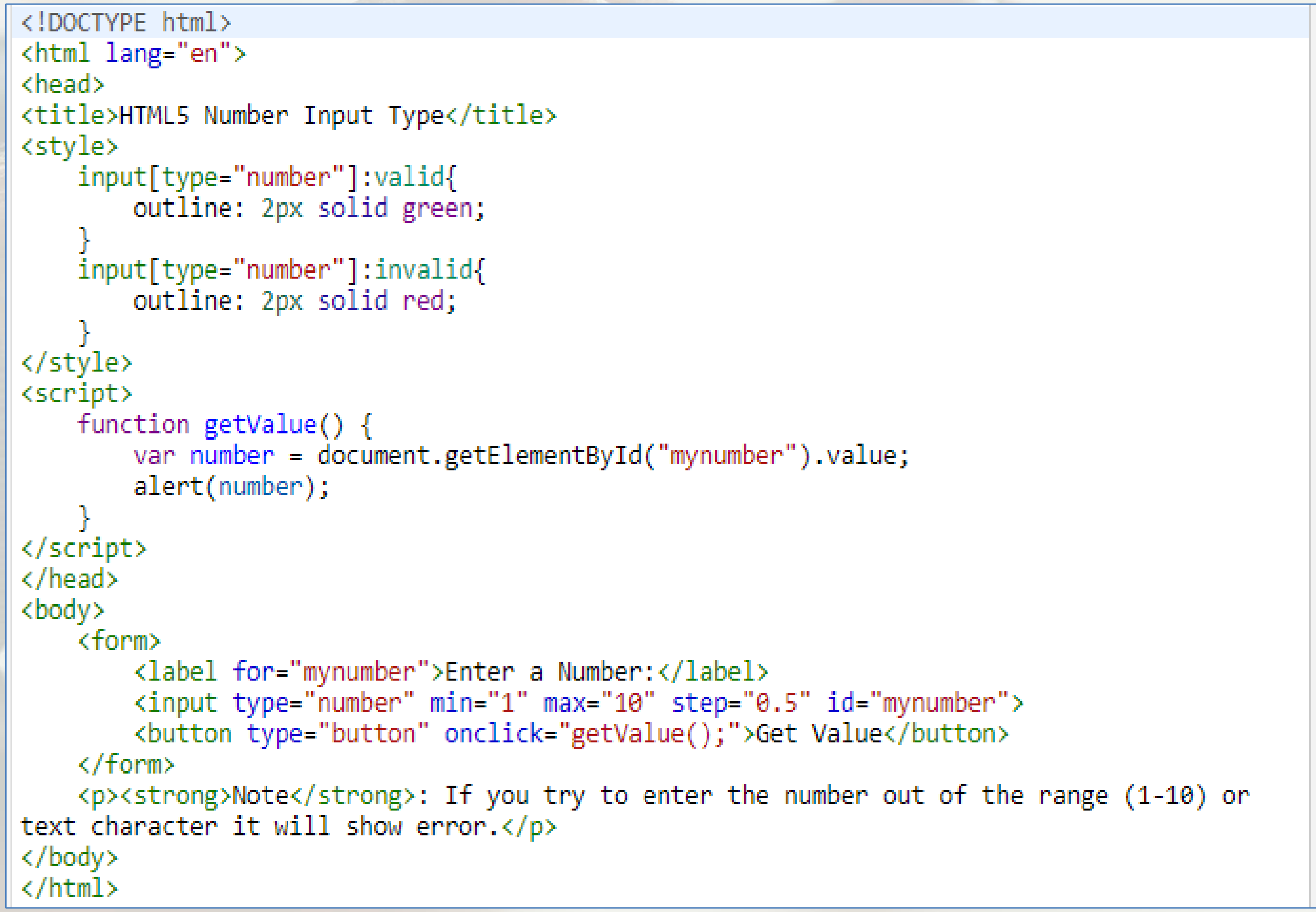
The number Input

Concerning the number input type, it is used for inserting a numerical value. The web designer can also limit the user to enter only acceptable values. For this to happen, the additional attributes min, max, and step should be used. This feature is supported by all major web browsers.

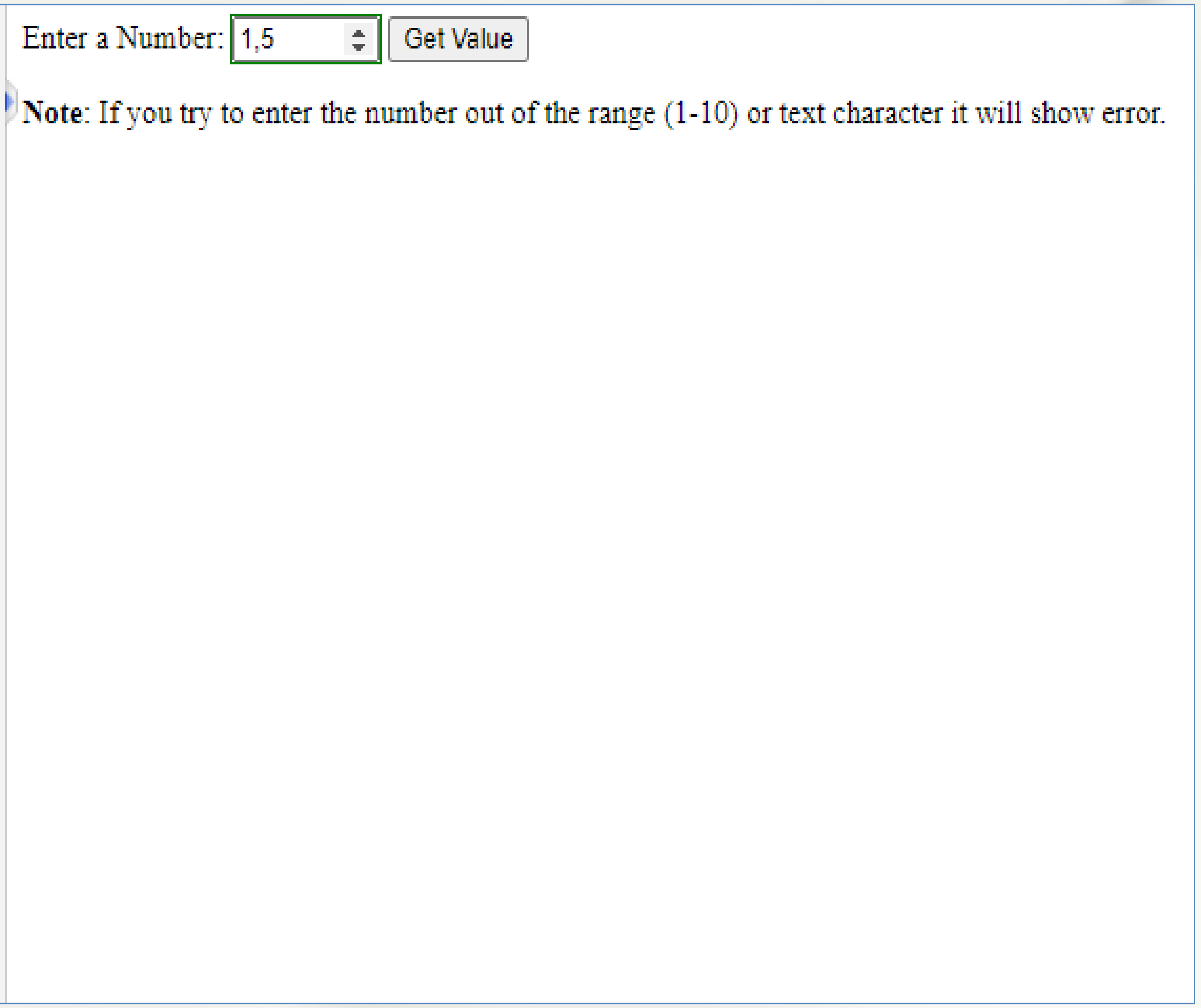
code4sp
coding for social promotion

HTML5 New Input Types

The number Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Number Input Type</title>
<style>
    input[type="number"]:valid{
        outline: 2px solid green;
    }
    input[type="number"]:invalid{
        outline: 2px solid red;
    }
</style>
<script>
    function getValue() {
        var number = document.getElementById("mynumber").value;
        alert(number);
    }
</script>
</head>
<body>
    <form>
        <label for="mynumber">Enter a Number:</label>
        <input type="number" min="1" max="10" step="0.5" id="mynumber">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
    <p><strong>Note</strong>: If you try to enter the number out of the range (1-10) or
text character it will show error.</p>
</body>
</html>
```



Enter a Number: 1,5 Get Value

Note: If you try to enter the number out of the range (1-10) or text character it will show error.

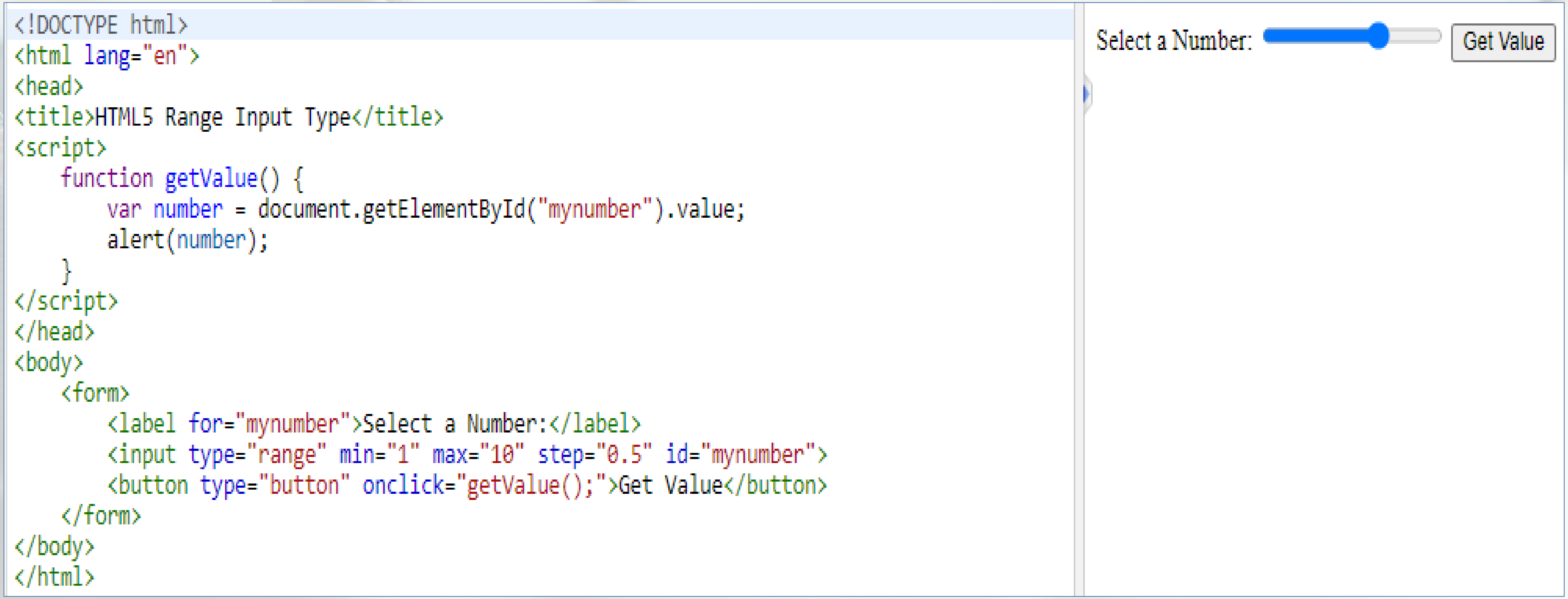
HTML5 New Input Types

The range Input

It can be applied for registering a numerical value within a specific range. It works very similar to number input above, although it presents an easier way for inserting a number. This input type is supported by all major web browsers.

HTML5 New Input Types

The range Input



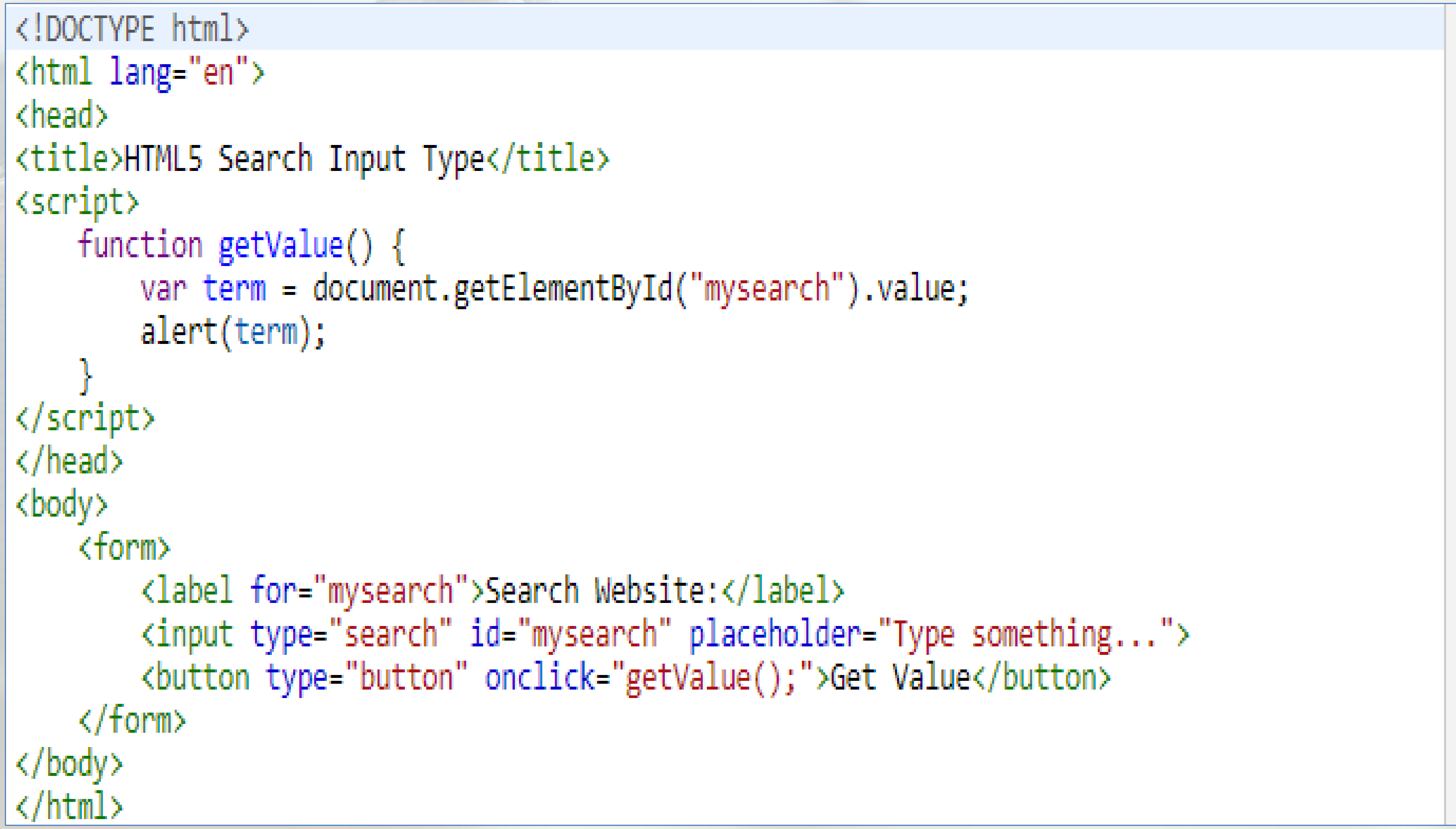
```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Range Input Type</title>
<script>
    function getValue() {
        var number = document.getElementById("mynumber").value;
        alert(number);
    }
</script>
</head>
<body>
    <form>
        <label for="mynumber">Select a Number:</label>
        <input type="range" min="1" max="10" step="0.5" id="mynumber">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

HTML5 New Input Types

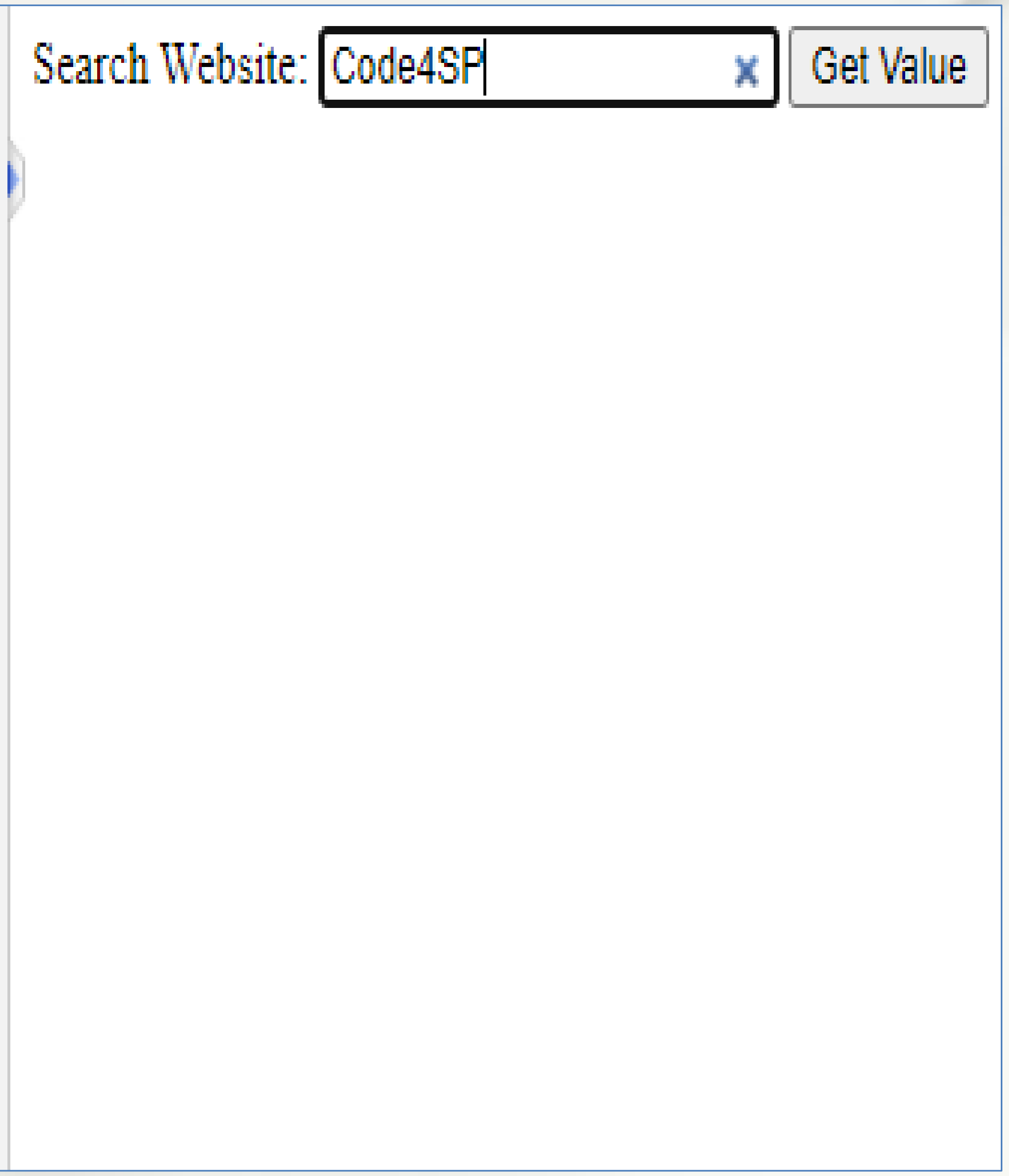
The search Input

It is suitable for generating search input fields. It must be stated that, in some browsers (i.e., Chrome and Safari), as soon as the user begins to type in the search box, a tiny cross emerges on the right side of the field, which can be useful for clearing the entire search field. It is supported by all major browsers.

The search Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Search Input Type</title>
<script>
    function getValue() {
        var term = document.getElementById("mysearch").value;
        alert(term);
    }
</script>
</head>
<body>
    <form>
        <label for="mysearch">Search Website:</label>
        <input type="search" id="mysearch" placeholder="Type something...">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```



HTML5 New Input Types

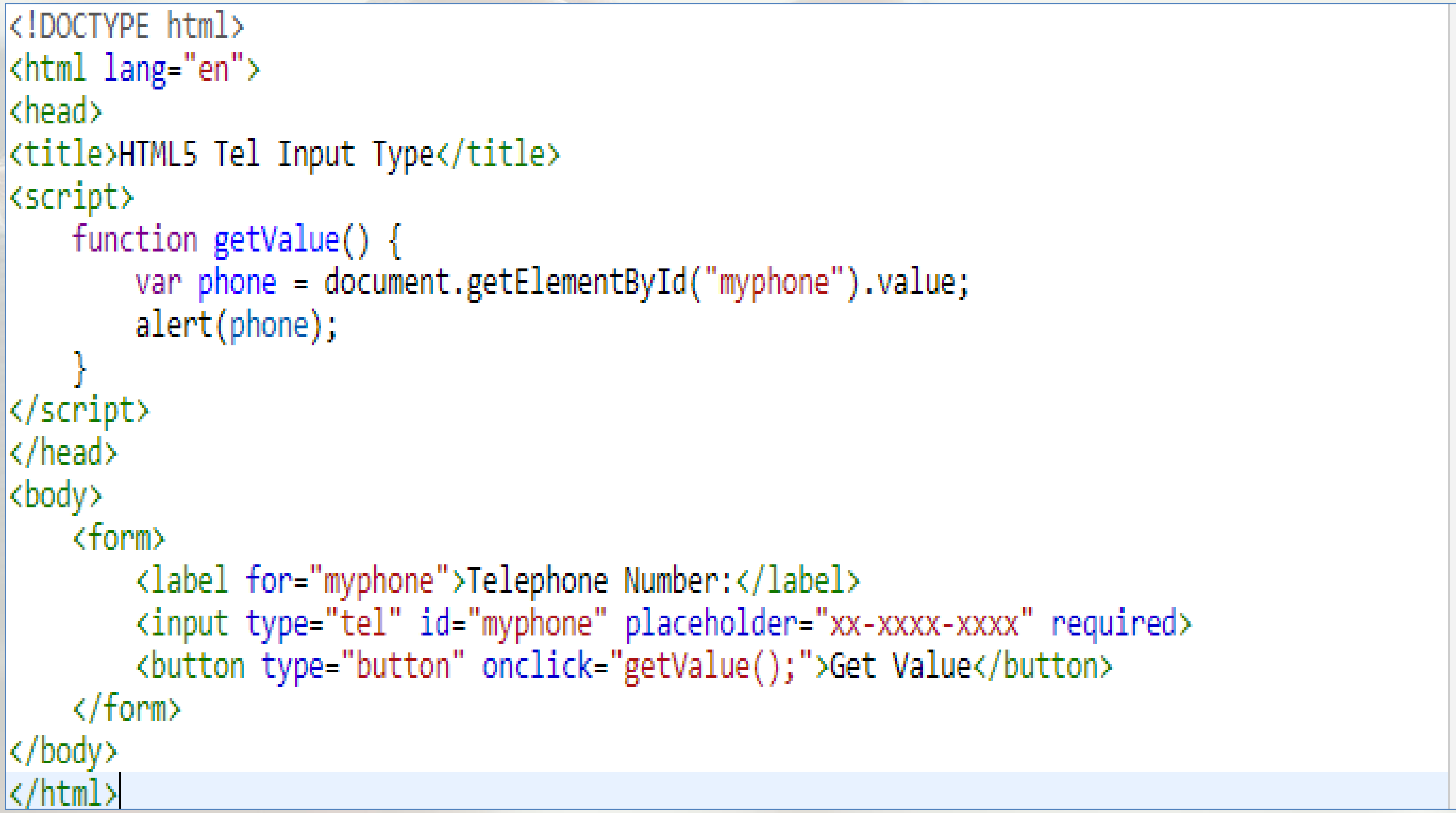
The tel Input

It is especially useful for inserting a phone number. As browsers do not support tel input validation by default, the placeholder attribute can be used to help users inserting the correct format for their phone number or indicate a regular expression to validate the user's input by applying the pattern attribute. This feature is not supported by any browser, as phone numbers vary a lot worldwide.

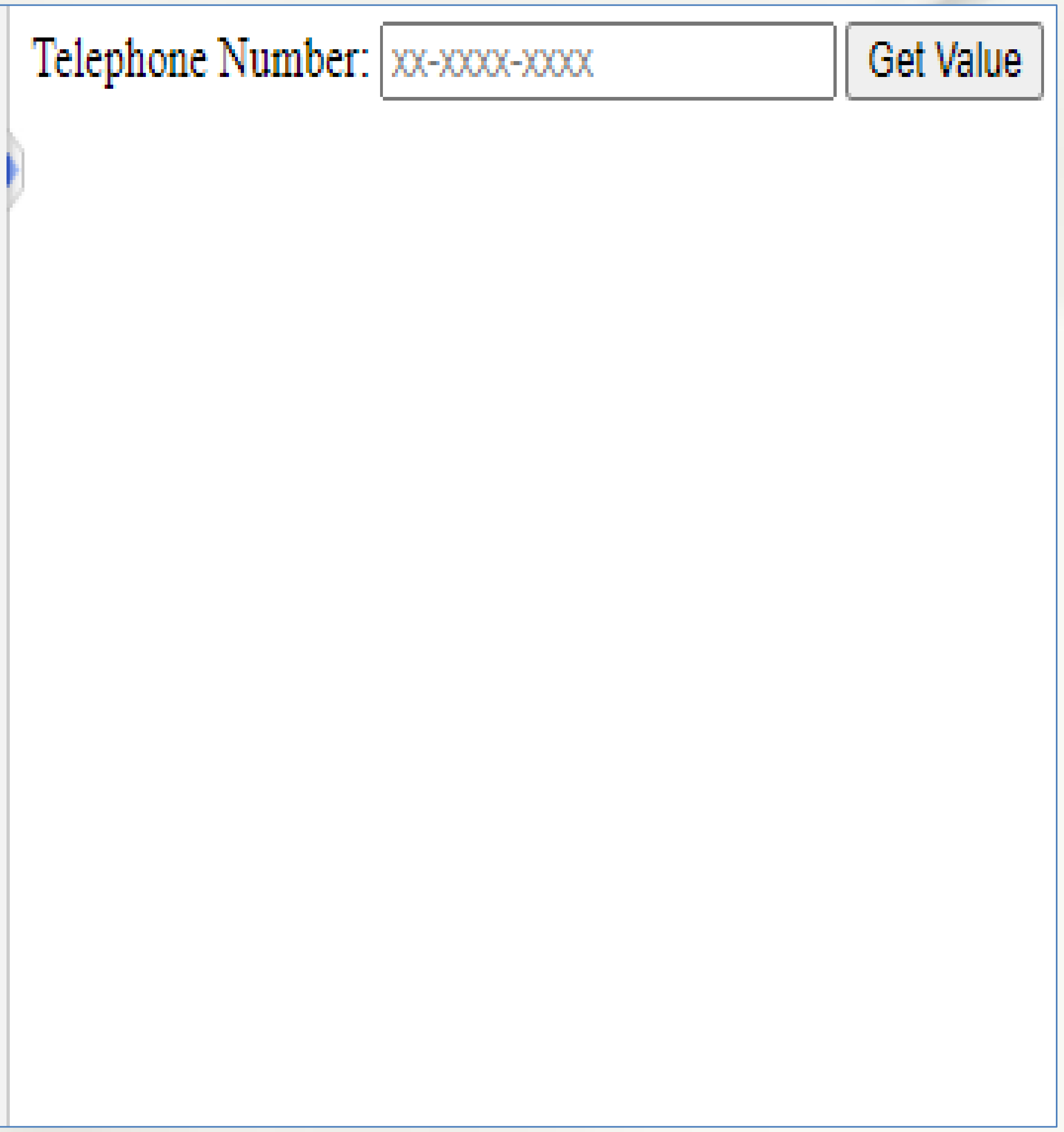
code4sp
coding for social promotion

HTML5 New Input Types

The tel Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Tel Input Type</title>
<script>
    function getValue() {
        var phone = document.getElementById("myphone").value;
        alert(phone);
    }
</script>
</head>
<body>
    <form>
        <label for="myphone">Telephone Number:</label>
        <input type="tel" id="myphone" placeholder="xx-xxxx-xxxx" required>
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```



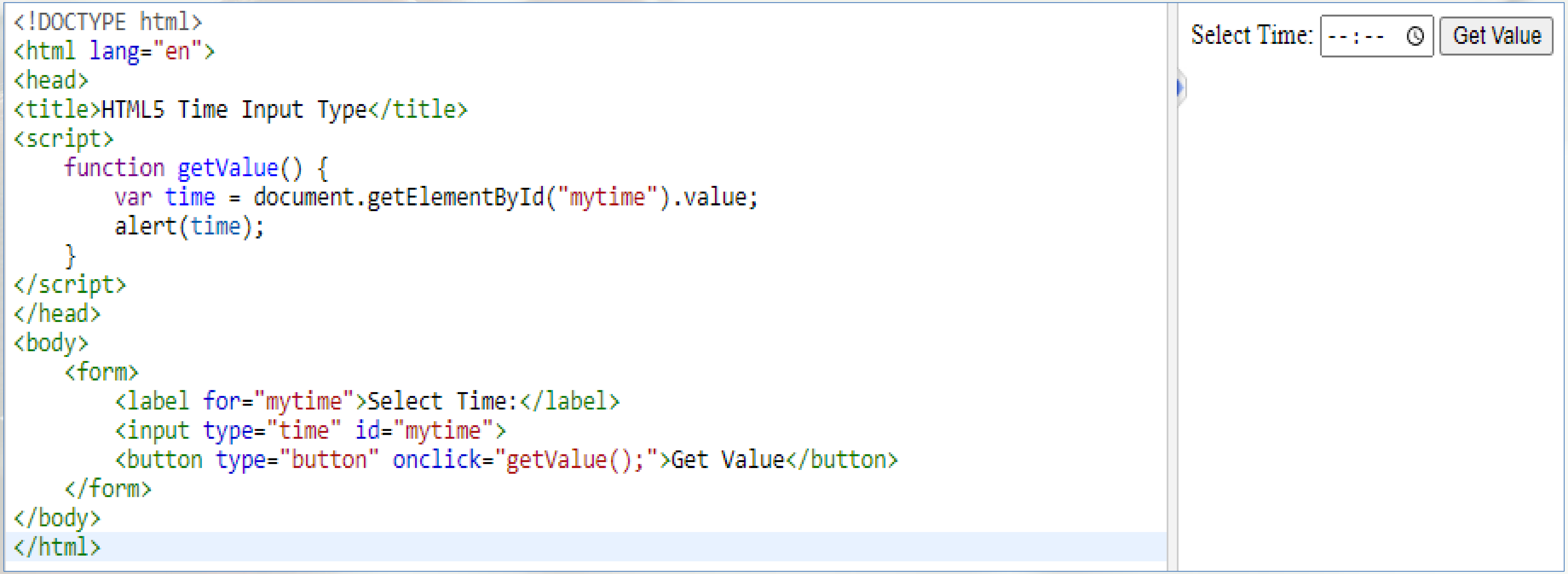
HTML5 New Input Types

The time Input

Regarding the time input type, it can be used for entering any given time (hours and minutes), and the browser can use both hour formats (12 or 24-hour) for inserting times, depending on the region. This input is not supported by IE and Safari browsers.

HTML5 New Input Types

The time Input



```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Time Input Type</title>
<script>
    function getValue() {
        var time = document.getElementById("mytime").value;
        alert(time);
    }
</script>
</head>
<body>
    <form>
        <label for="mytime">Select Time:</label>
        <input type="time" id="mytime">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

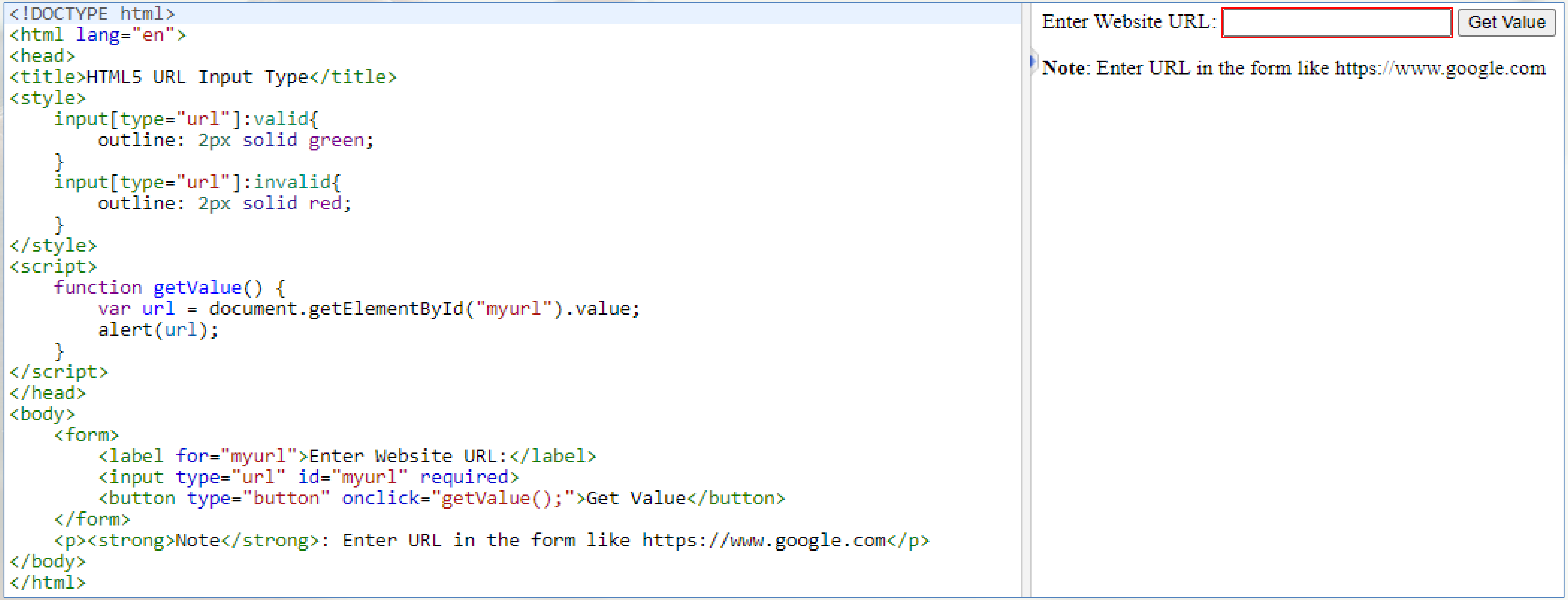
HTML5 New Input Types

The url Input

Concerning the url input type, it can be used for inserting URL’s or web addresses. The multiple attribute can be used for inserting more than one URL. Moreover, if required attribute is restricted, the browser will automatically perform validation to ensure that only text that meets the standard format for URLs goes into the input box. All major browsers support this input type.

HTML5 New Input Types

The url Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 URL Input Type</title>
<style>
    input[type="url"]:valid{
        outline: 2px solid green;
    }
    input[type="url"]:invalid{
        outline: 2px solid red;
    }
</style>
<script>
    function getValue() {
        var url = document.getElementById("myurl").value;
        alert(url);
    }
</script>
</head>
<body>
    <form>
        <label for="myurl">Enter Website URL:</label>
        <input type="url" id="myurl" required>
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
    <p><strong>Note</strong>: Enter URL in the form like https://www.google.com</p>
</body>
</html>
```

Enter Website URL: Get Value

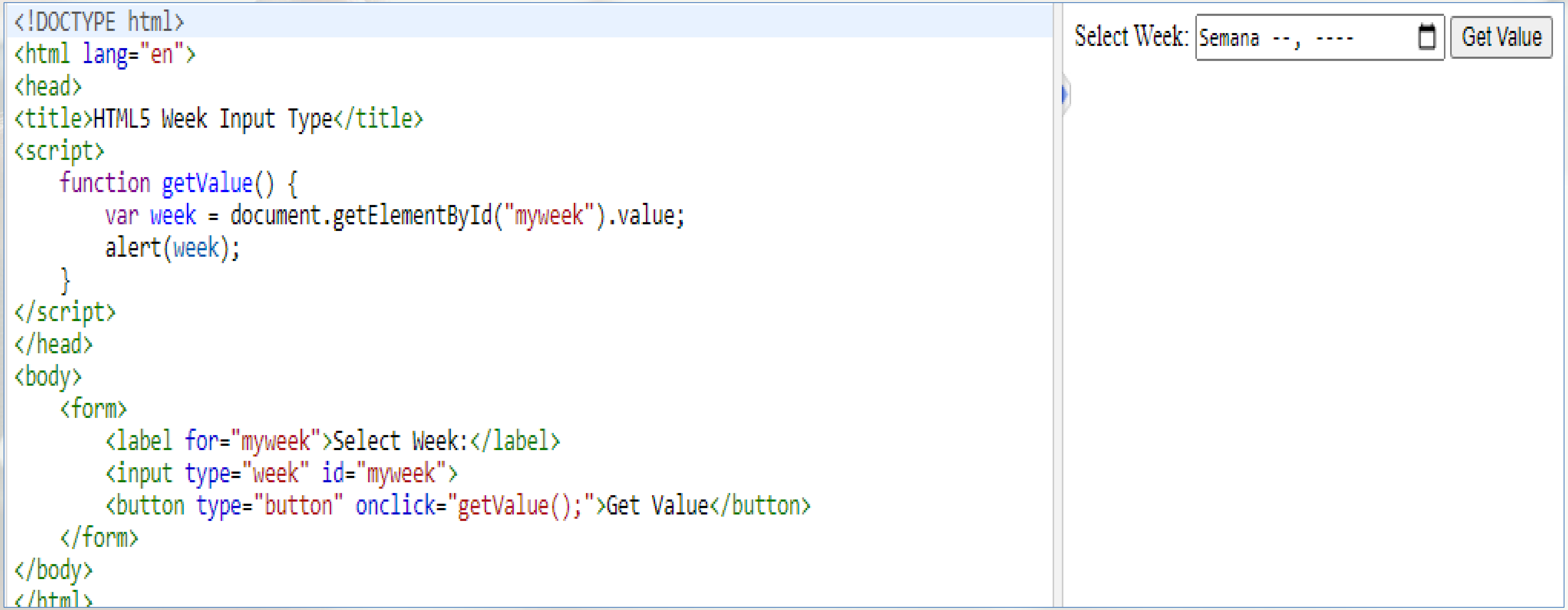
Note: Enter URL in the form like https://www.google.com

The week Input

Finally, the week input type enables the user to choose a week and year from a drop-down calendar. This feature is not supported by Firefox, Safari and IE, but it is currently supported by Chrome, Edge and Opera browsers.

HTML5 New Input Types

The week Input



```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Week Input Type</title>
<script>
    function getValue() {
        var week = document.getElementById("myweek").value;
        alert(week);
    }
</script>
</head>
<body>
    <form>
        <label for="myweek">Select Week:</label>
        <input type="week" id="myweek">
        <button type="button" onclick="getValue();">Get Value</button>
    </form>
</body>
</html>
```

HTML5 Canvas

- This section will be essentially useful for learning how to draw graphics using the HTML5 canvas element. It was originally created by Apple for the Mac OS dashboard widgets and to enable graphics in Safari. Moreover, it was adopted by other browsers, like Firefox, Google Chrome and Opera.

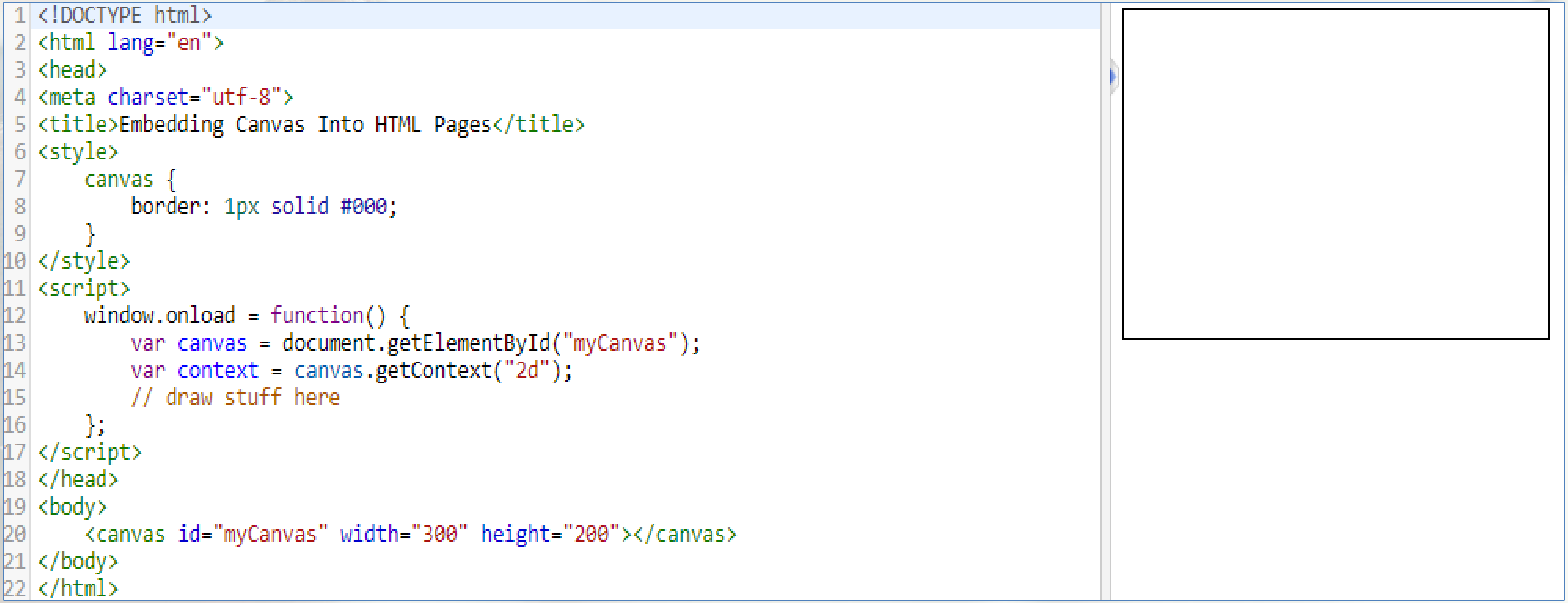
- By default, the <canvas> element has 300px of width and 150px of height without any border and content. Nevertheless, custom width and height can be specified using the CSS height and width feature.

- The canvas is a two-dimensional four-sided area. The coordinates of the top-left corner of the canvas are (0, 0), identified as origin, and the coordinates of the bottom-right corner are (*canvas width, canvas height*), as can be seen using the interactive tool available here.

- To draw basic paths and shapes utilizing the recently introduced HTML5 canvas element and JavaScript, it will be useful to take a look at several templates.

HTML5 Canvas

Firstly, the base template for drawing paths and shapes onto the 2D HTML5 canvas:



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Embedding Canvas Into HTML Pages</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        // draw stuff here
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

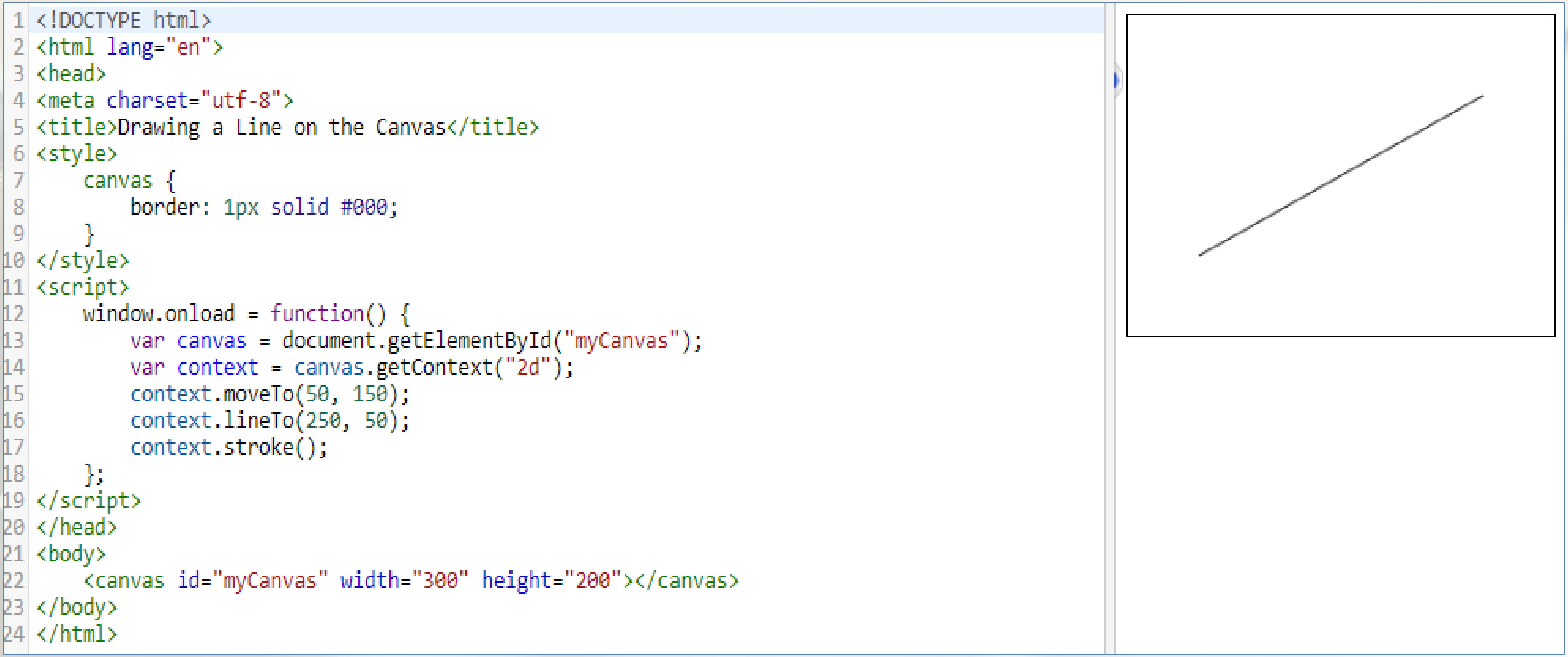
HTML5 Canvas

All the lines except those from 7 to 11 are quite straight forward and easy to interpret. The anonymous function affixed to the window.onload event will execute when the page loads. As soon as the page is loaded, one can access the canvas element with document.getElementById() method. Later, a 2D canvas context is defined by passing 2d into the getContext() method of the canvas object.

HTML5 Canvas

The initial step to draw on canvas is a **straight line**. The most important procedures used for this purpose are moveTo(), lineTo() and the stroke(). The moveTo() method identifies the position of drawing cursor onto the canvas, while the lineTo() method used to define the coordinates of the line's end point, and finally the stroke() method is used to make the line visible:

HTML5 Canvas



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Line on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

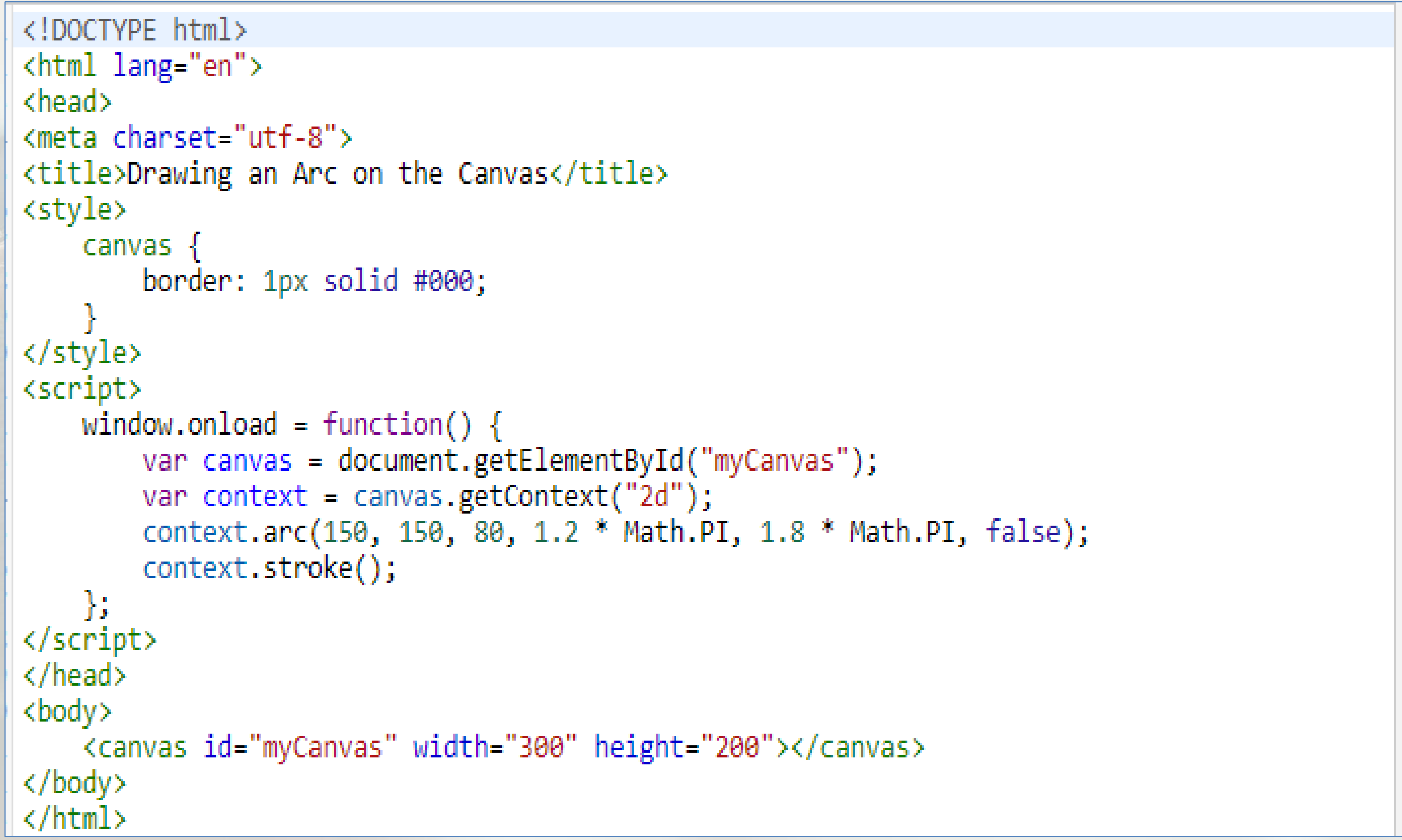
HTML5 Canvas

What about **drawing an arc**? It can be obtained by simply using the arc() method, which syntax is:

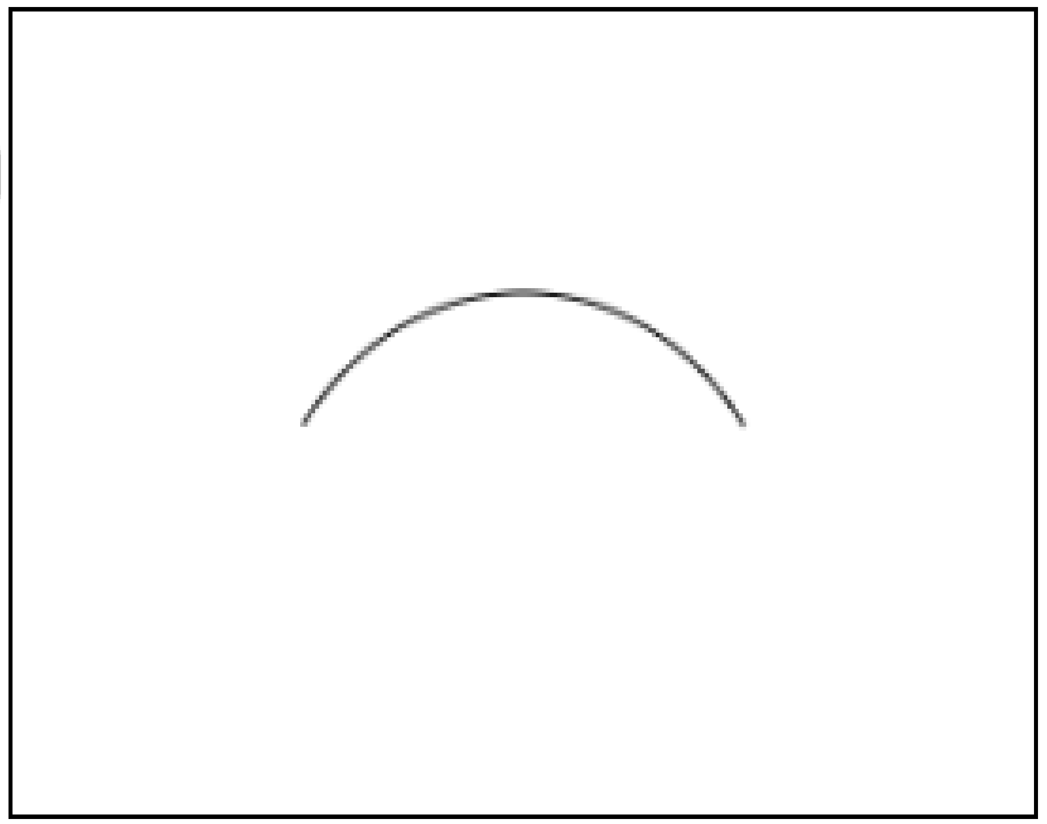
```
context.arc(centerX, centerY, radius, startingAngle,
endingAngle, counterclockwise);
```

In the next slide's example, an arc was drawn on canvas by inserting a JavaScript code:

HTML5 Canvas



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing an Arc on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



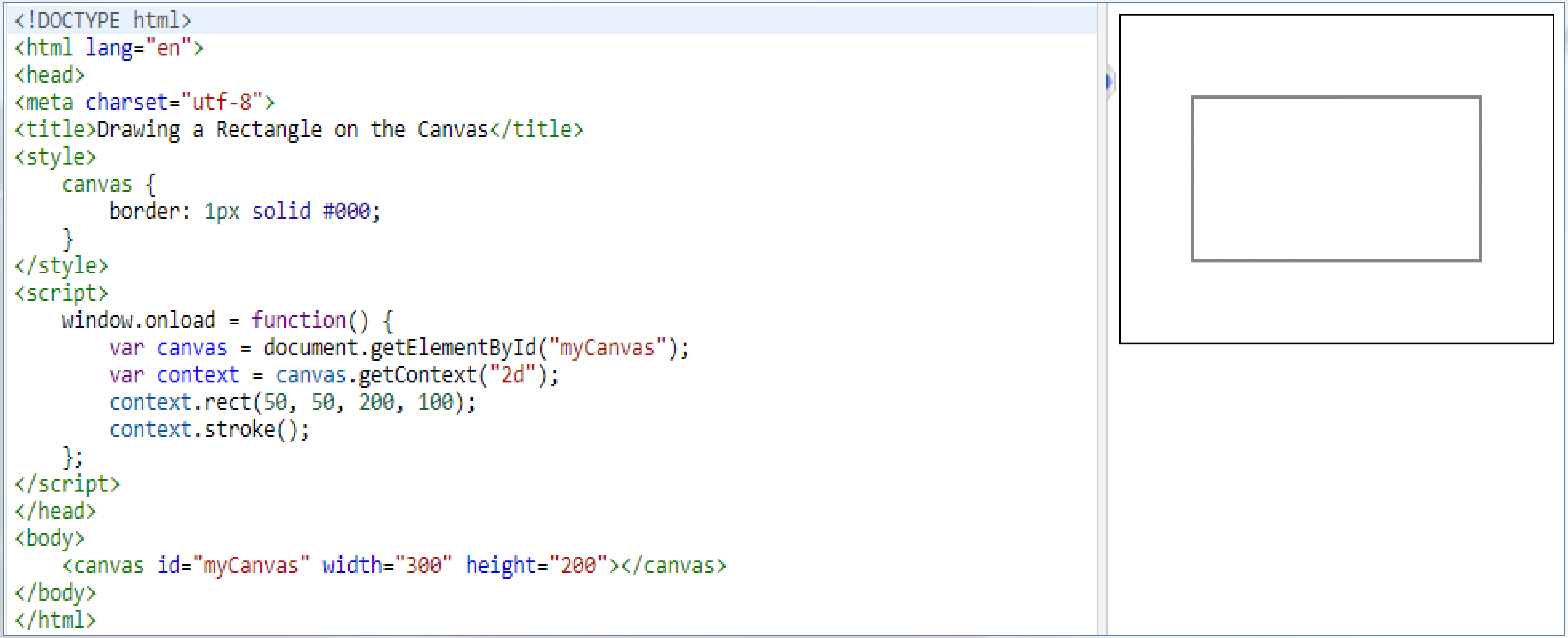
HTML5 Canvas

To draw a **rectangle and square shapes**, the rect() method is the way to go. It entails four parameters: x, y positions of the rectangle and its width and height. The basic syntax of the rect() method is the following:

```
context.rect(x, y, width, height);
```

To draw it using a JS code:

HTML5 Canvas



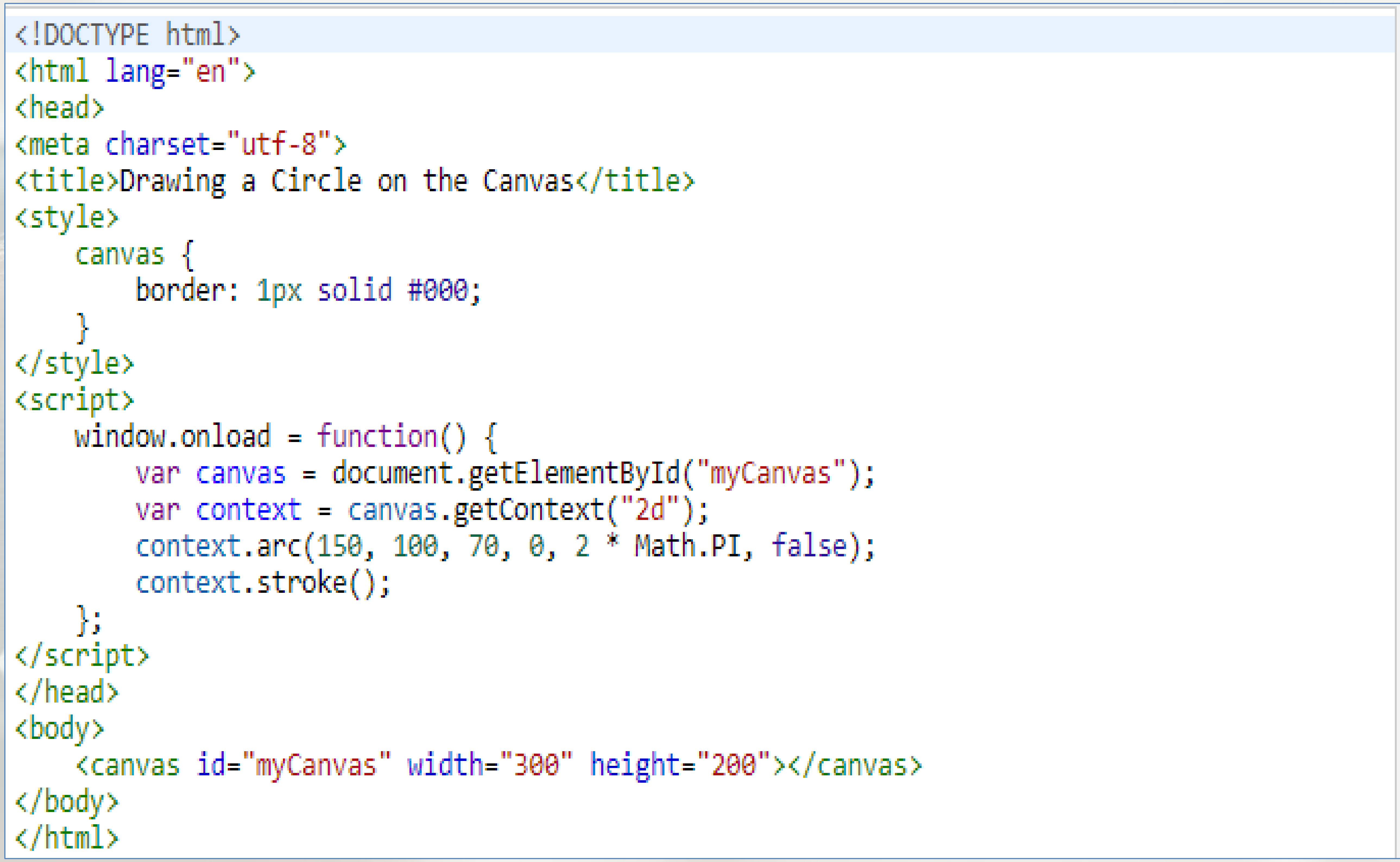
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Rectangle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

HTML5 Canvas

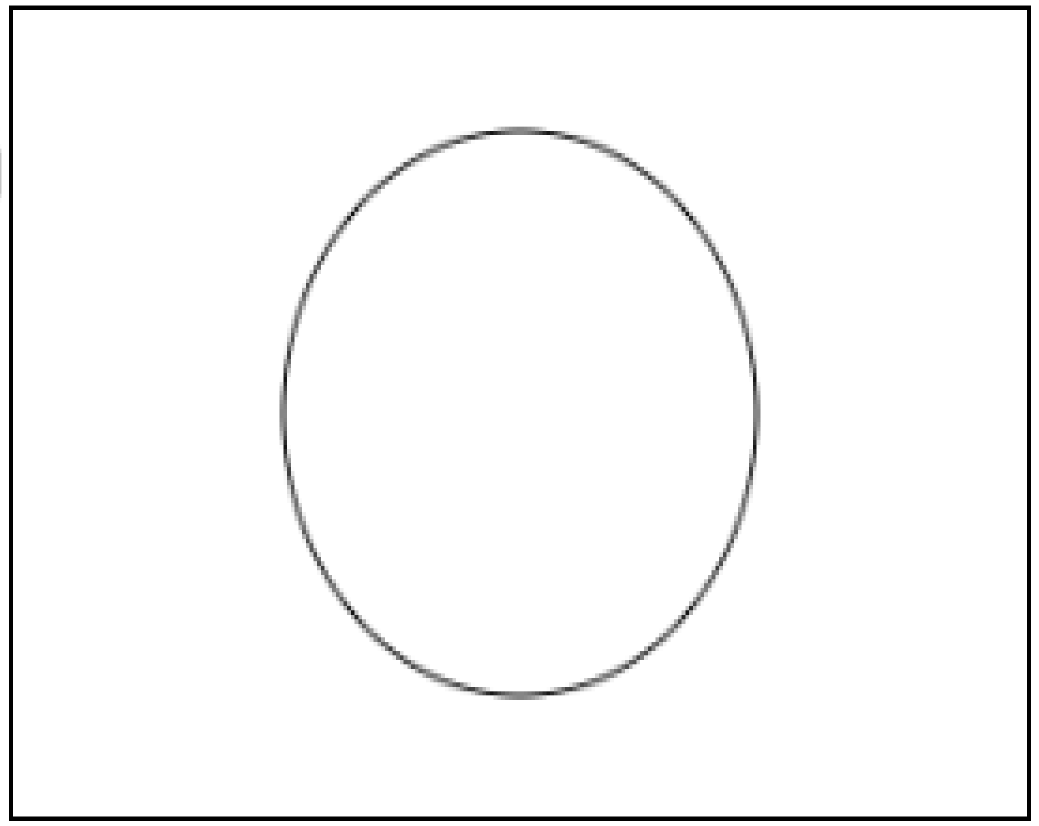
In opposite to the rect() method, there is no specific procedure for drawing a circle. However, the result can be obtained by creating a fully enclosed arc, by using the arc() method. The synthax for drawing a complete circle using the arc() method is the following:

```
context.arc(centerX, centerY, radius, 0, 2 * Math.PI,
false);
```

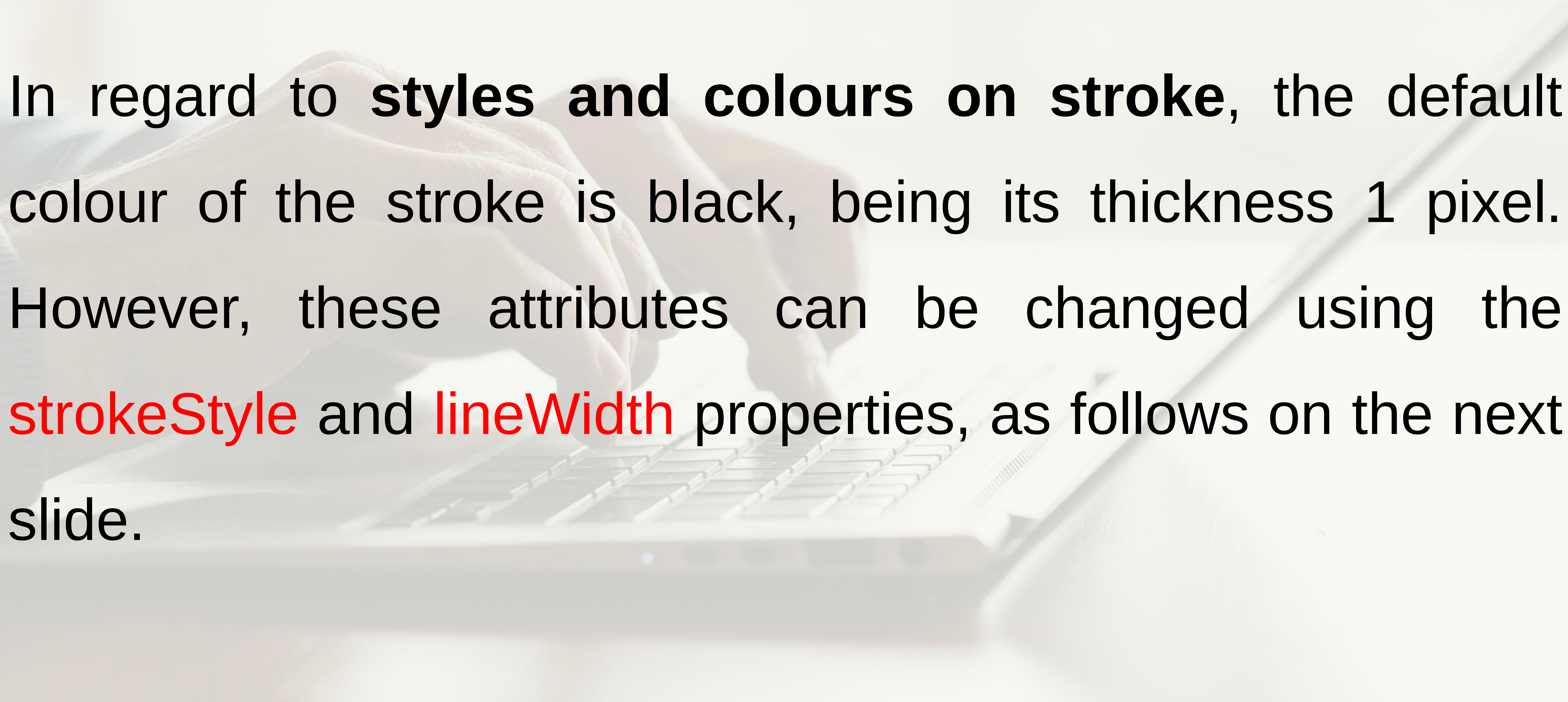
HTML5 Canvas



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Drawing a Circle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

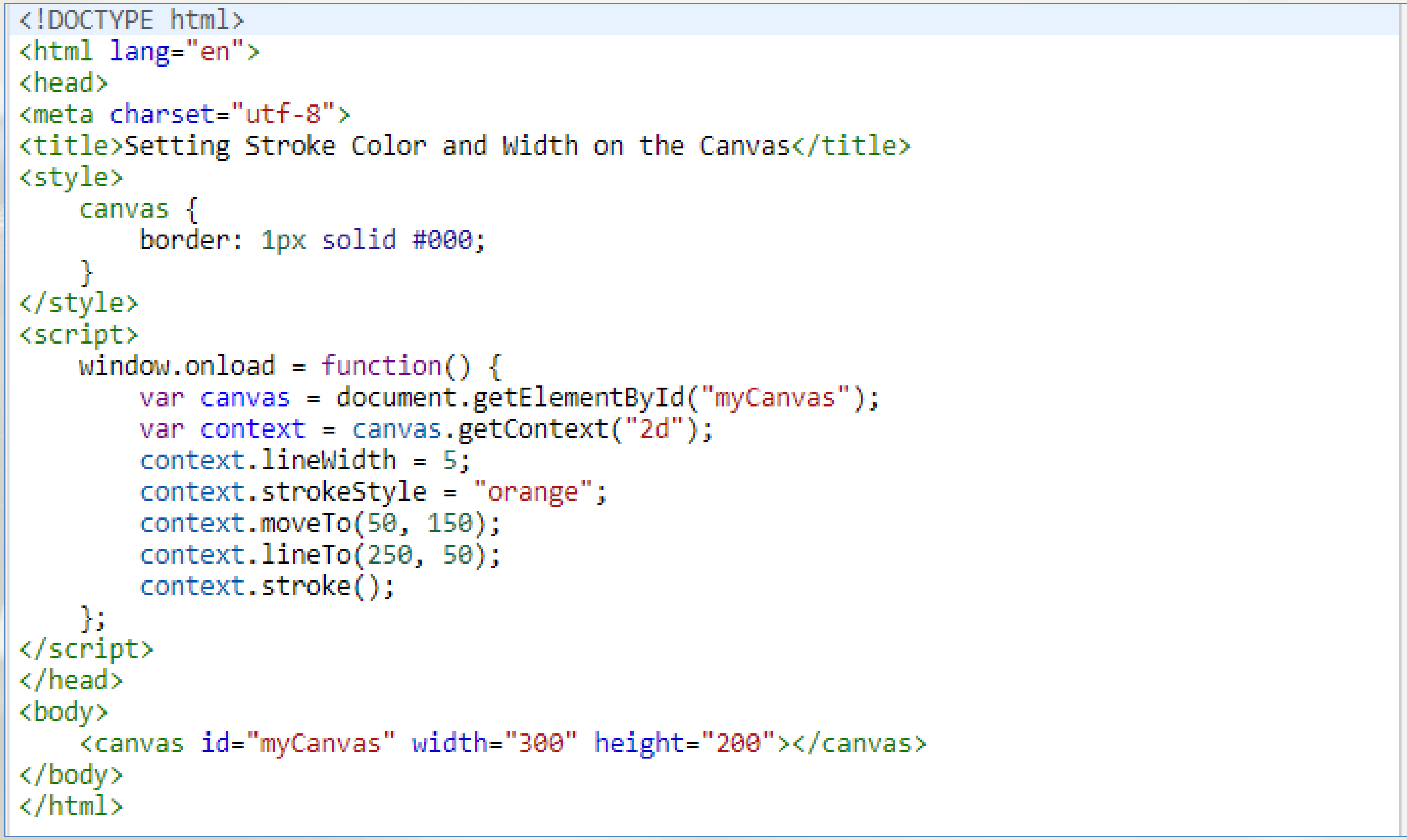


HTML5 Canvas

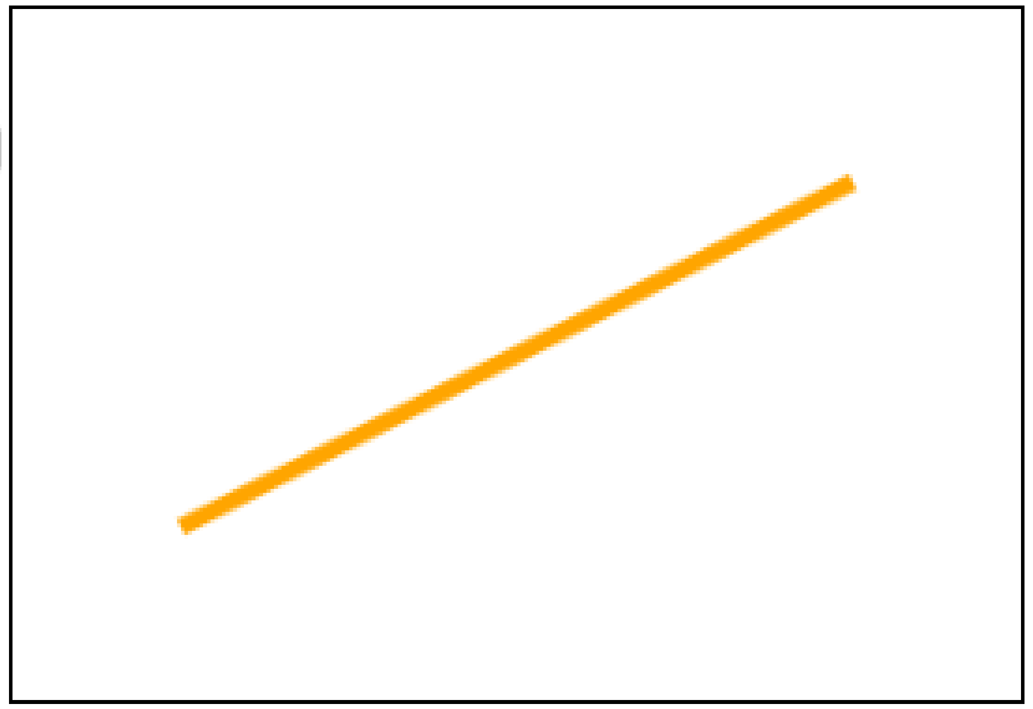


In regard to **styles and colours on stroke**, the default colour of the stroke is black, being its thickness 1 pixel. However, these attributes can be changed using the strokeStyle and lineWidth properties, as follows on the next slide.

HTML5 Canvas



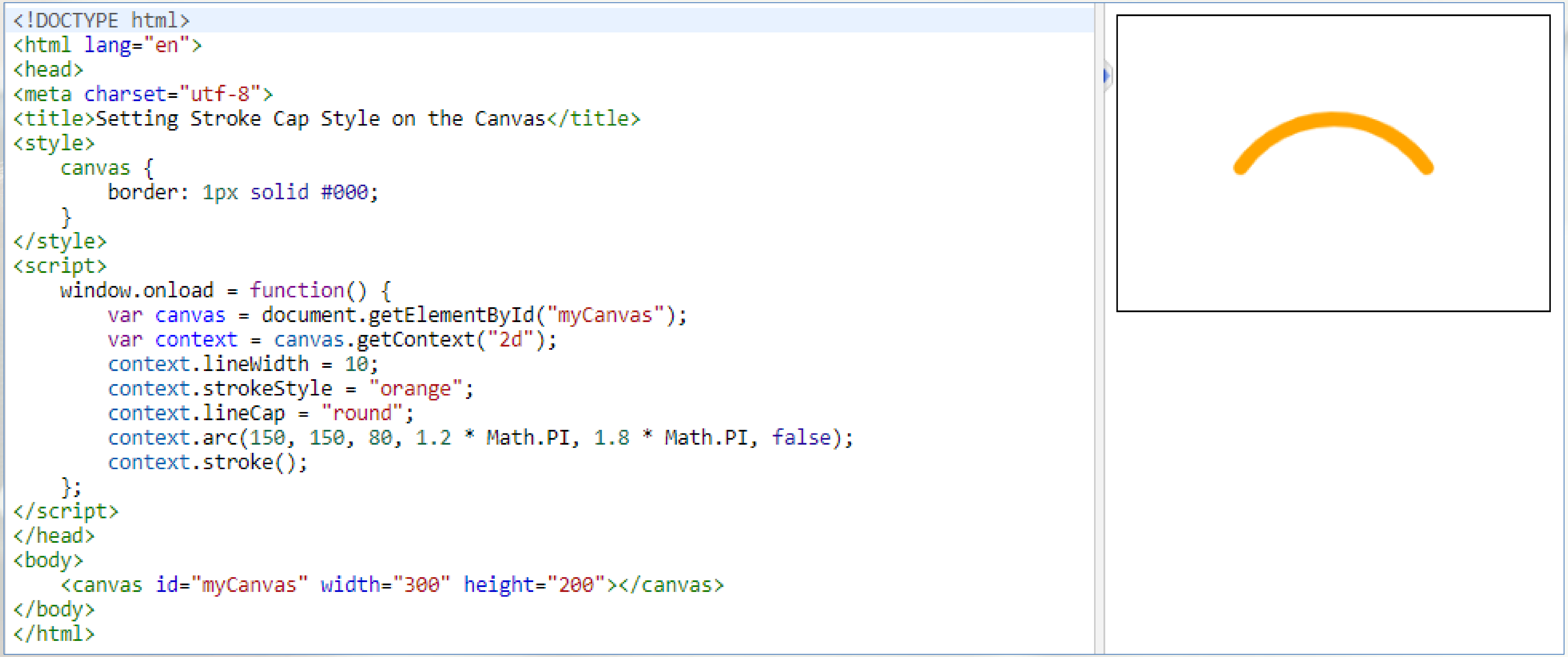
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Color and Width on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 5;
        context.strokeStyle = "orange";
        context.moveTo(50, 150);
        context.lineTo(250, 50);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



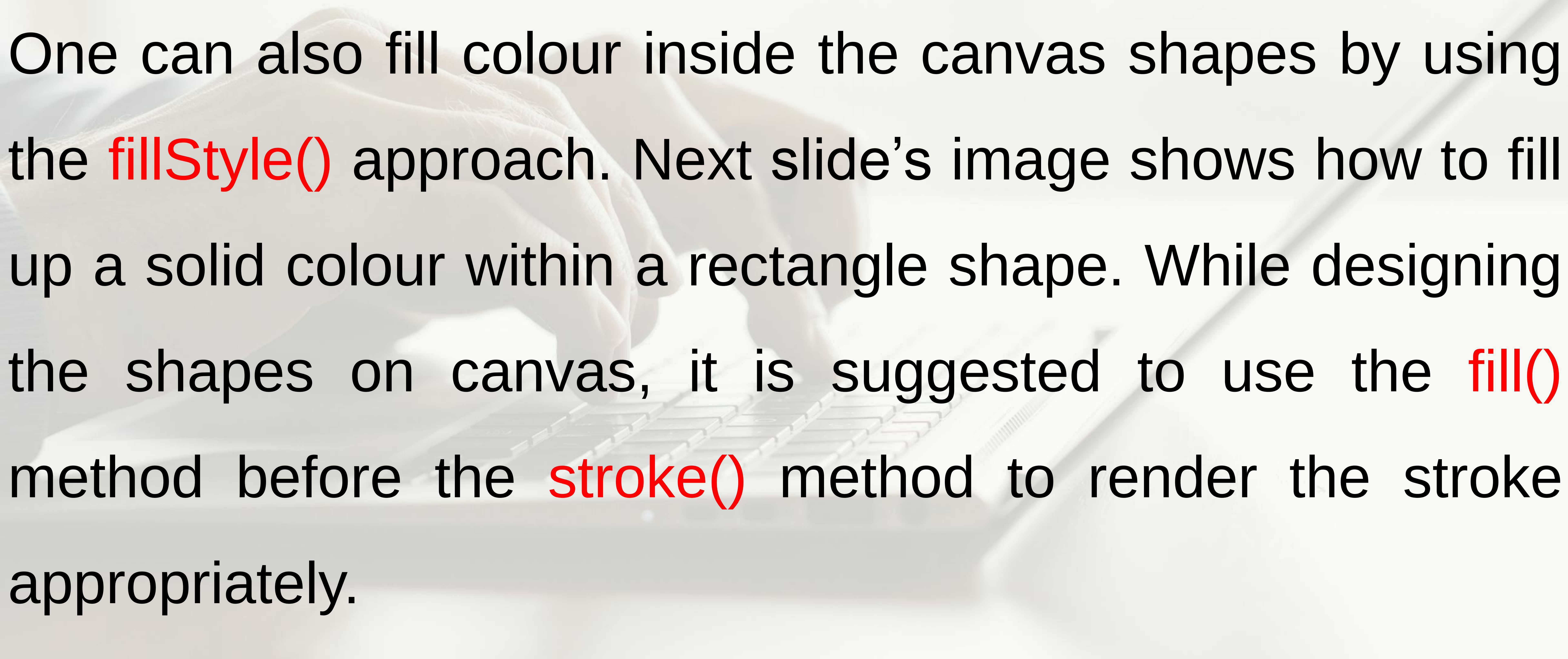
HTML5 Canvas

It is also possible to set the cap style for the lines by using the lineCap property, with three styles available: butt, round, and square. Check out next slide’s figure.

HTML5 Canvas



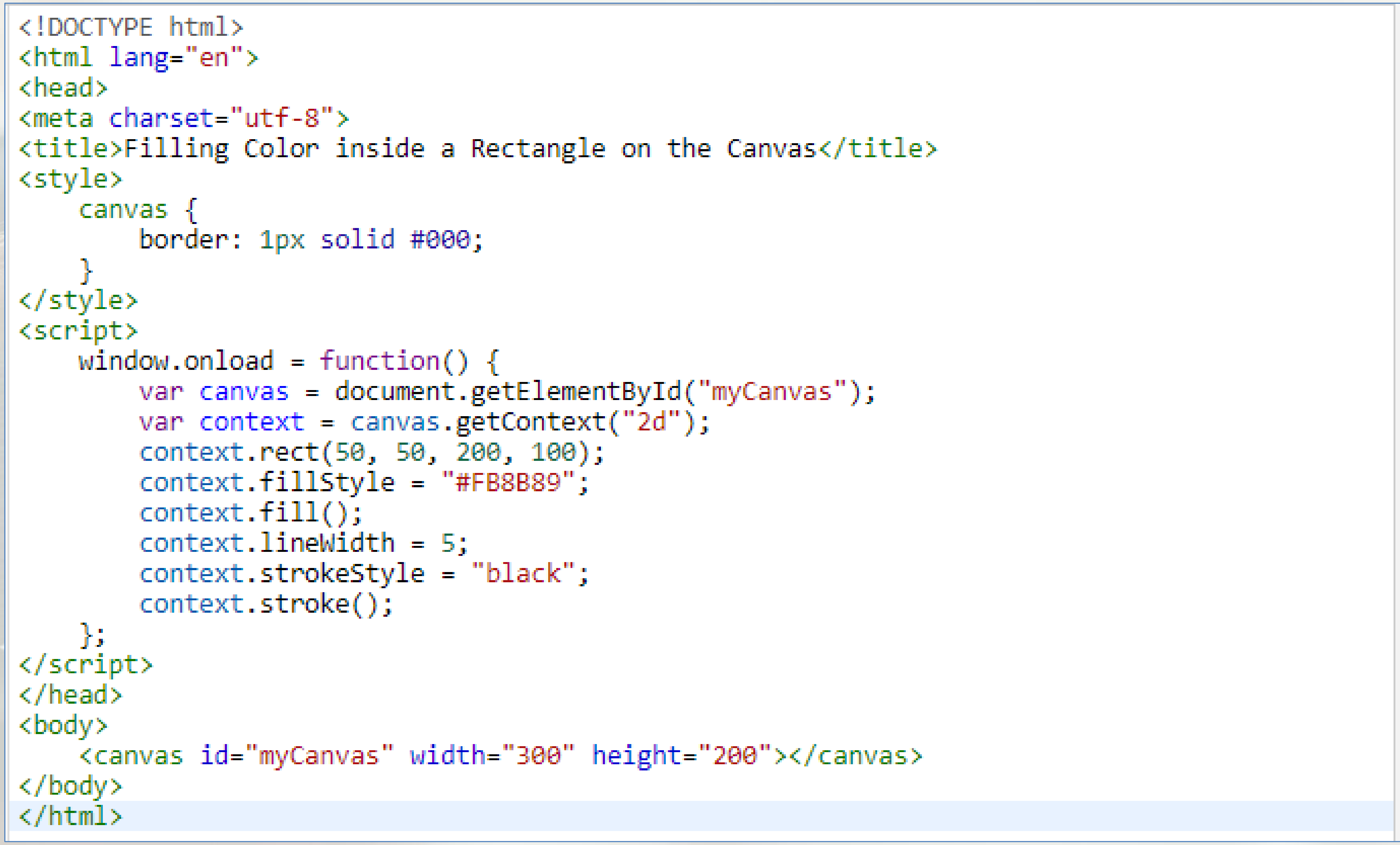
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Setting Stroke Cap Style on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.lineWidth = 10;
        context.strokeStyle = "orange";
        context.lineCap = "round";
        context.arc(150, 150, 80, 1.2 * Math.PI, 1.8 * Math.PI, false);
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



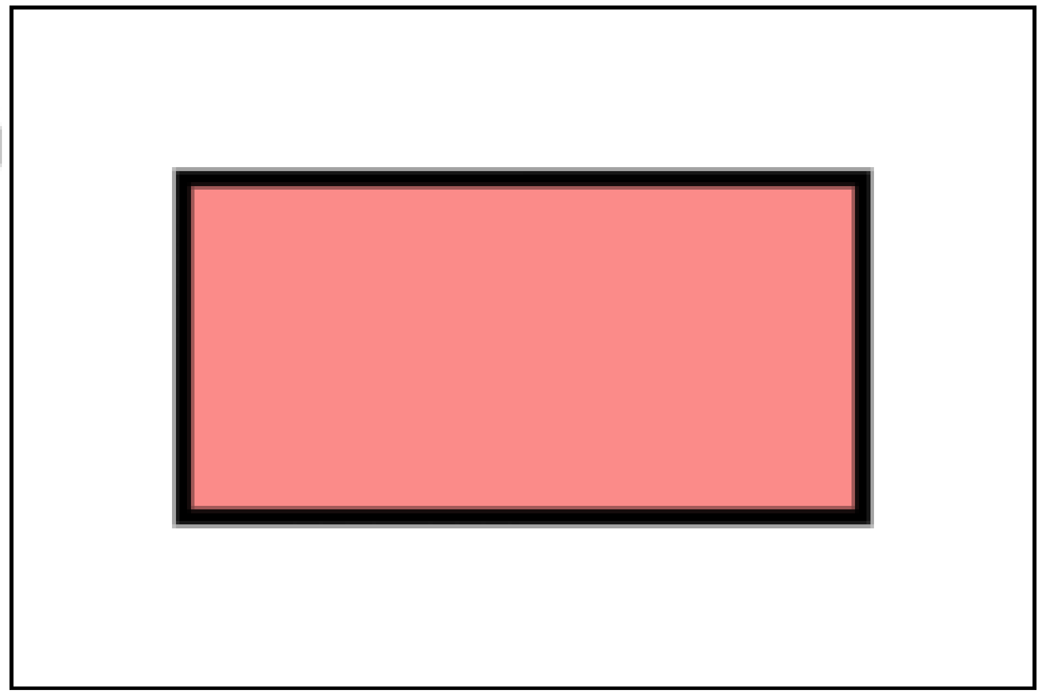
HTML5 Canvas

One can also fill colour inside the canvas shapes by using the fillStyle() approach. Next slide’s image shows how to fill up a solid colour within a rectangle shape. While designing the shapes on canvas, it is suggested to use the fill() method before the stroke() method to render the stroke appropriately.

HTML5 Canvas



```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Color inside a Rectangle on the Canvas</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        context.fillStyle = "#FB8B89";
        context.fill();
        context.lineWidth = 5;
        context.strokeStyle = "black";
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```



HTML5 Canvas

It is also possible to fill gradient colour (a smooth visual transition from one colour to another) within the canvas shapes. There are two types of gradients here: linear and radial.

The basic syntax for creating a **linear gradient** is:

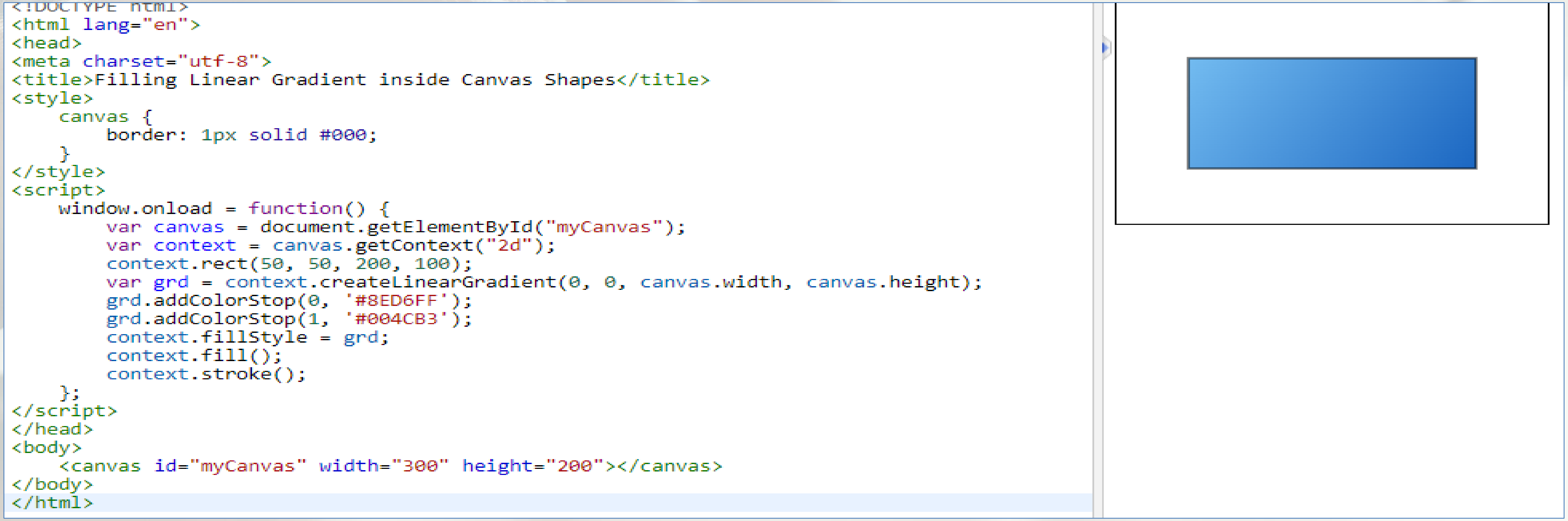
```
var grd = context.createLinearGradient(startX, startY, endX, endY);
```

The basic syntax for creating a **radial gradient** is:

```
var grd = context.createRadialGradient(startX, startY, startRadius,
endX, endY, endRadius);
```

HTML5 Canvas

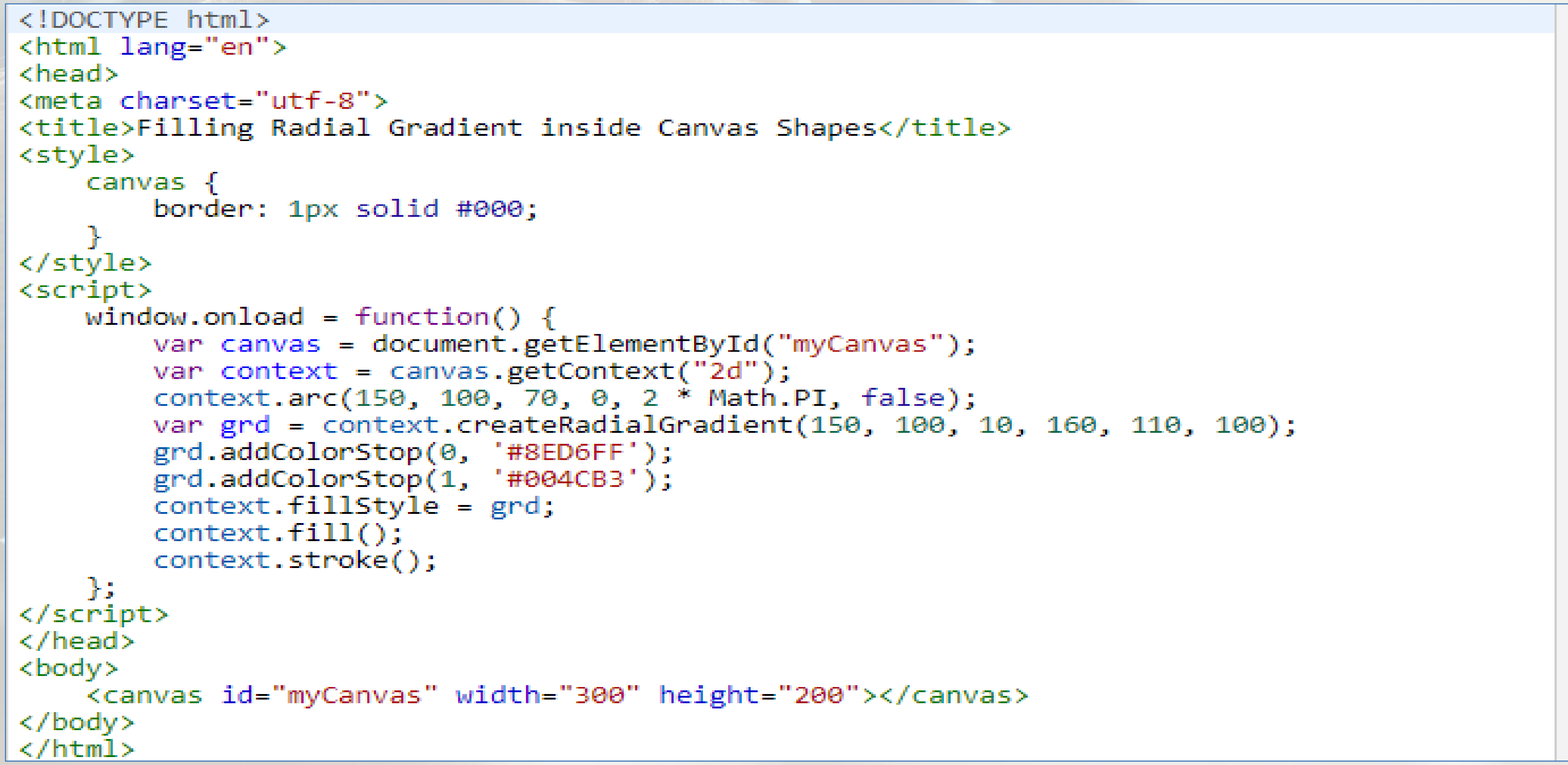
The picture below shows how to fill a **linear gradient** colour inside a rectangle using the createLinearGradient() method:



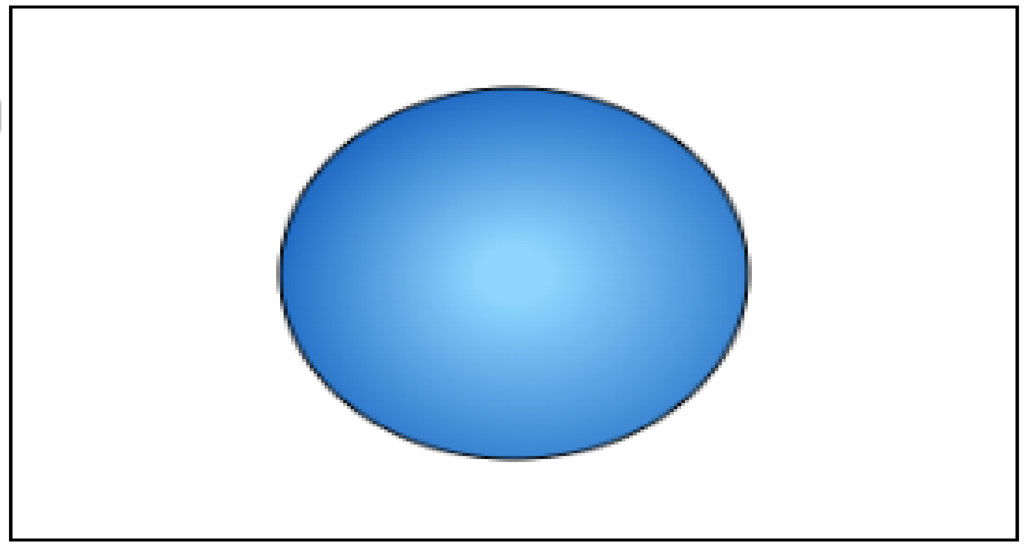
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Linear Gradient inside Canvas Shapes</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.rect(50, 50, 200, 100);
        var grd = context.createLinearGradient(0, 0, canvas.width, canvas.height);
        grd.addColorStop(0, '#8ED6FF');
        grd.addColorStop(1, '#004CB3');
        context.fillStyle = grd;
        context.fill();
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

HTML5 Canvas

The figure below demonstrates how to fill a radial gradient colour inside a circle through the createRadialGradient() method:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Filling Radial Gradient inside Canvas Shapes</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.arc(150, 100, 70, 0, 2 * Math.PI, false);
        var grd = context.createRadialGradient(150, 100, 10, 160, 110, 100);
        grd.addColorStop(0, '#8ED6FF');
        grd.addColorStop(1, '#004CB3');
        context.fillStyle = grd;
        context.fill();
        context.stroke();
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

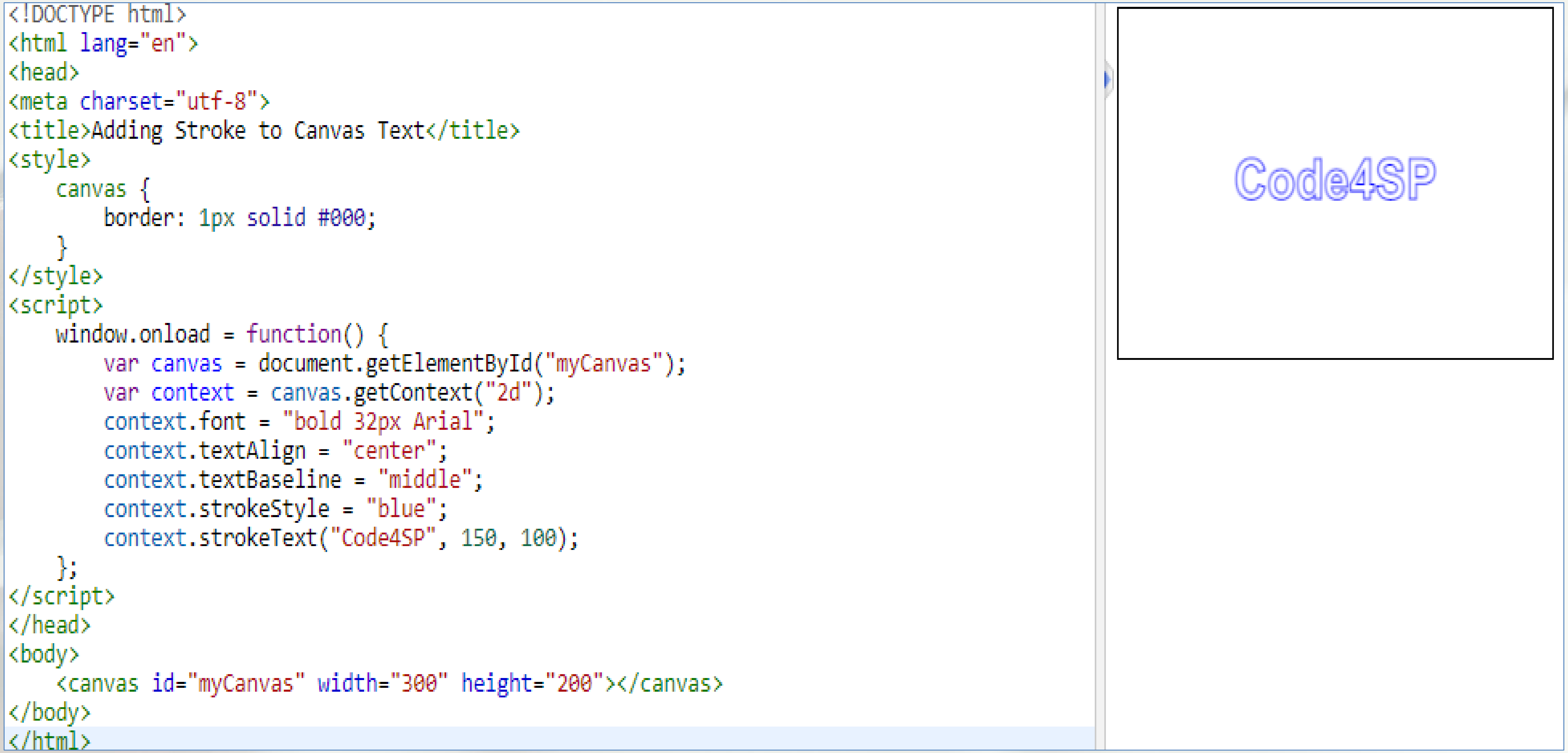


HTML5 Canvas

It is also possible to **draw text** on canvas (containing Unicode characters only), as well as adding it **colour and alignment**, and even apply stroke on text using the strokeText() method, which will colour the perimeter of the text instead of filling it.

Nonetheless, if the web designer intends to set both the fill and stroke on the text element, he/she can use the fillText() and the strokeText() methods together. it is suggested to use the fillText() method before the strokeText() method to render the stroke accurately.

HTML5 Canvas



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Adding Stroke to Canvas Text</title>
<style>
    canvas {
        border: 1px solid #000;
    }
</style>
<script>
    window.onload = function() {
        var canvas = document.getElementById("myCanvas");
        var context = canvas.getContext("2d");
        context.font = "bold 32px Arial";
        context.textAlign = "center";
        context.textBaseline = "middle";
        context.strokeStyle = "blue";
        context.strokeText("Code4SP", 150, 100);
    };
</script>
</head>
<body>
    <canvas id="myCanvas" width="300" height="200"></canvas>
</body>
</html>
```

HTML5 SVG

- This chapter is expected to be clear on how to use HTML5 svg elements to draw vector graphics on a web page. To do so, firstly it is important to define **SVG**.

- **SVG** stands for Scalable Vector Graphics, and it is an XML-based image format that is used to define two-dimensional vector-based graphics for the web. Distinct from raster image (e.g. .jpg, .gif, .png, and other two-dimensional formats), a vector image can be scaled up or down to any extent without losing the quality of the image. Vector images are comprised of a series of shapes defined by math, while raster images are composed of a fixed set of dots (pixels).

- Like other topics discussed along this chapter, SVG is also a W3C recommendation.

HTML5 SVG

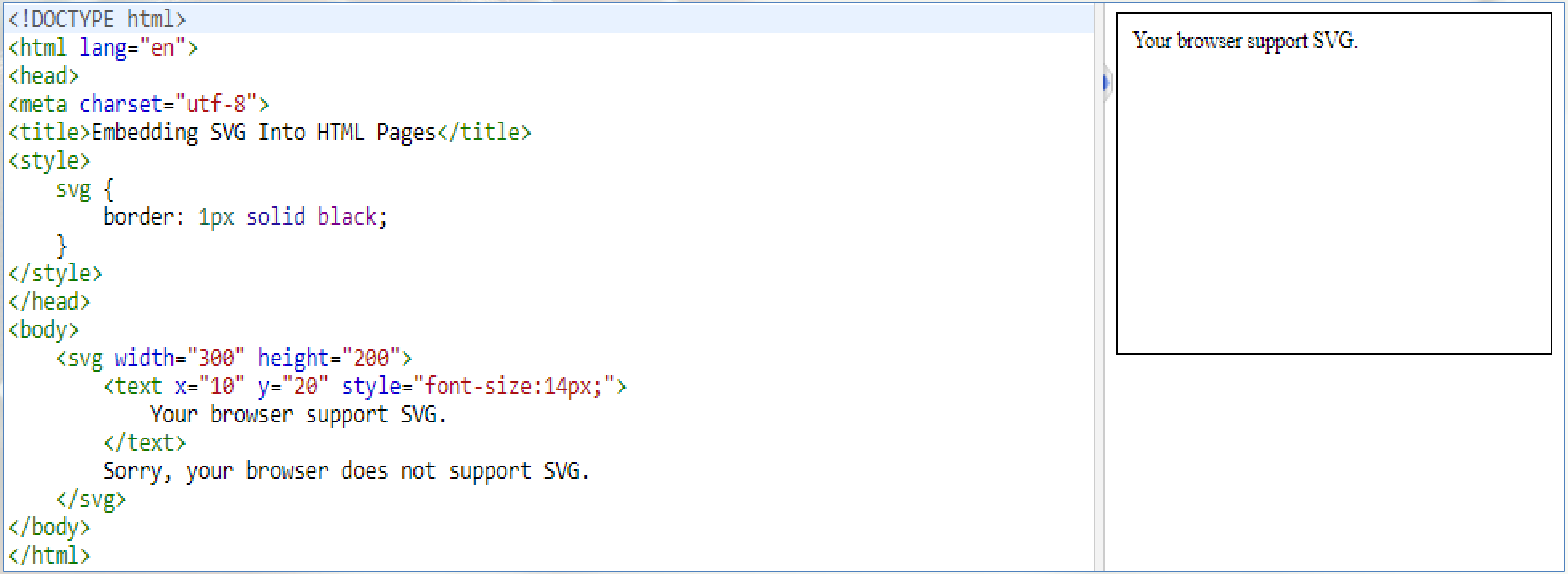
An SVG image is built by using a sequence of statements that go along with the XML schema, so, **SVG images** can be created and edited with a text editor like Notepad. There are numerous advantages of using SVG images rather than other image formats, as follows:

- They can be searched, indexed, scripted, and compressed.
- They can be created and modified using JavaScript in real time.
- They can be printed with high quality at any resolution.
- They can be animated using the built-in animation elements.
- They can contain hyperlinks to other documents.

All the major modern web browsers (Chrome, Firefox, Safari, and Opera), as well as Internet Explorer 9 and above are compatible with inline SVG rendering.

HTML5 SVG

SVG graphics can be embedded directly in a document by using the HTML5 <svg> element (as seen below):



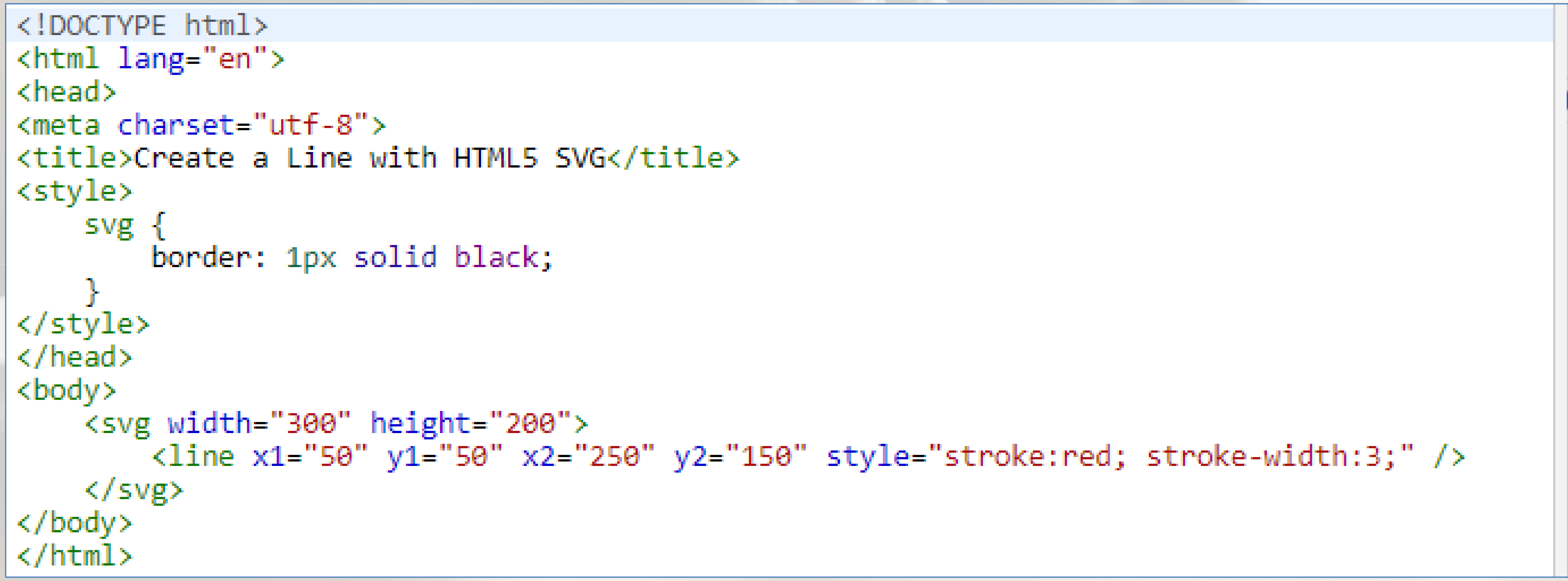
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Embedding SVG Into HTML Pages</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <text x="10" y="20" style="font-size:14px;">
            Your browser support SVG.
        </text>
        Sorry, your browser does not support SVG.
    </svg>
</body>
</html>
```

```
Your browser support SVG.
```

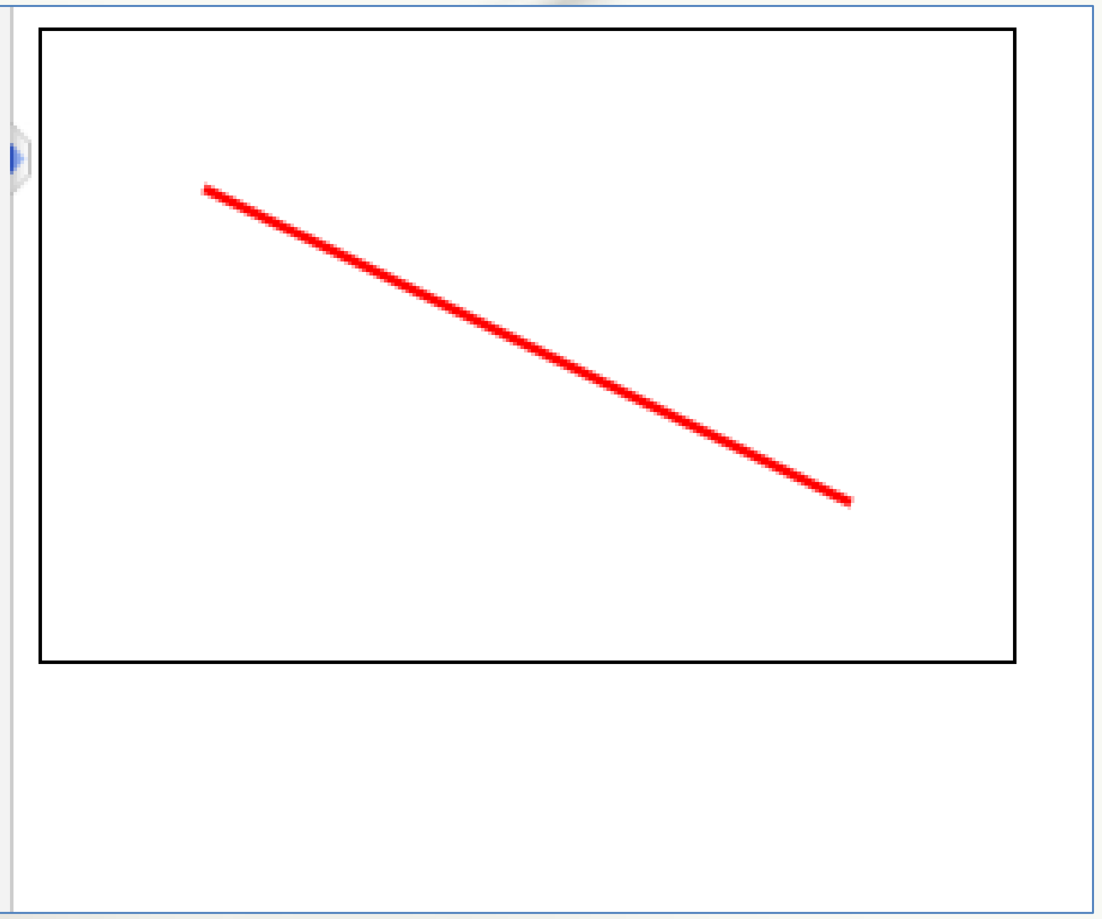
HTML5 SVG

Basic vector-based paths and shapes on the webpages can be drawn by utilizing the HTML5 <svg> element.

The most basic path to work with SVG is to **draw a straight line**. For that to happen, the SVG <line> element should be used. As can be seen in *Figure 93*, the attributes x1, x2, y1 and y2 of the <line> element draw a line from (x1,y1) to (x2,y2).

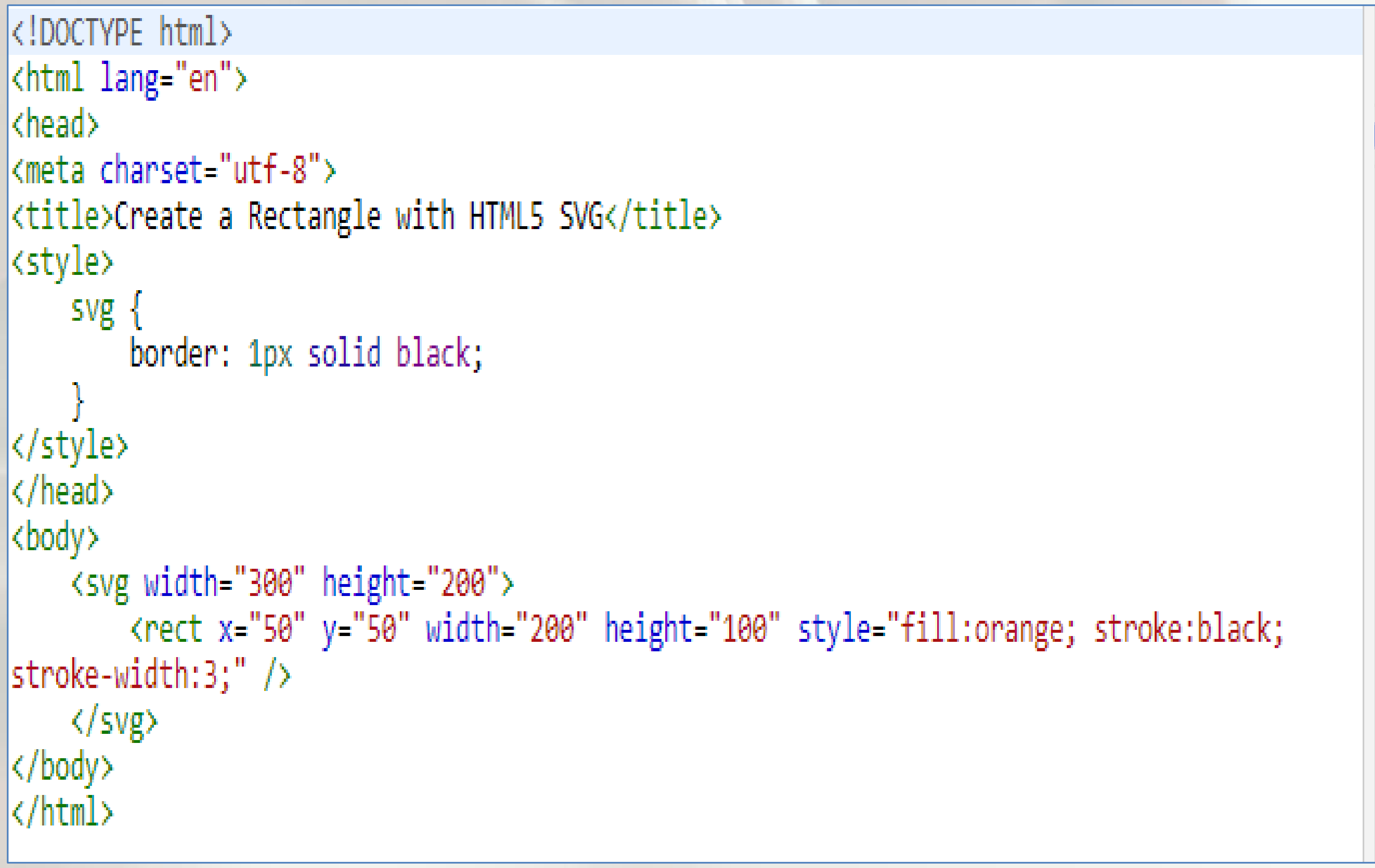


```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Line with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <line x1="50" y1="50" x2="250" y2="150" style="stroke:red; stroke-width:3;" />
    </svg>
</body>
</html>
```

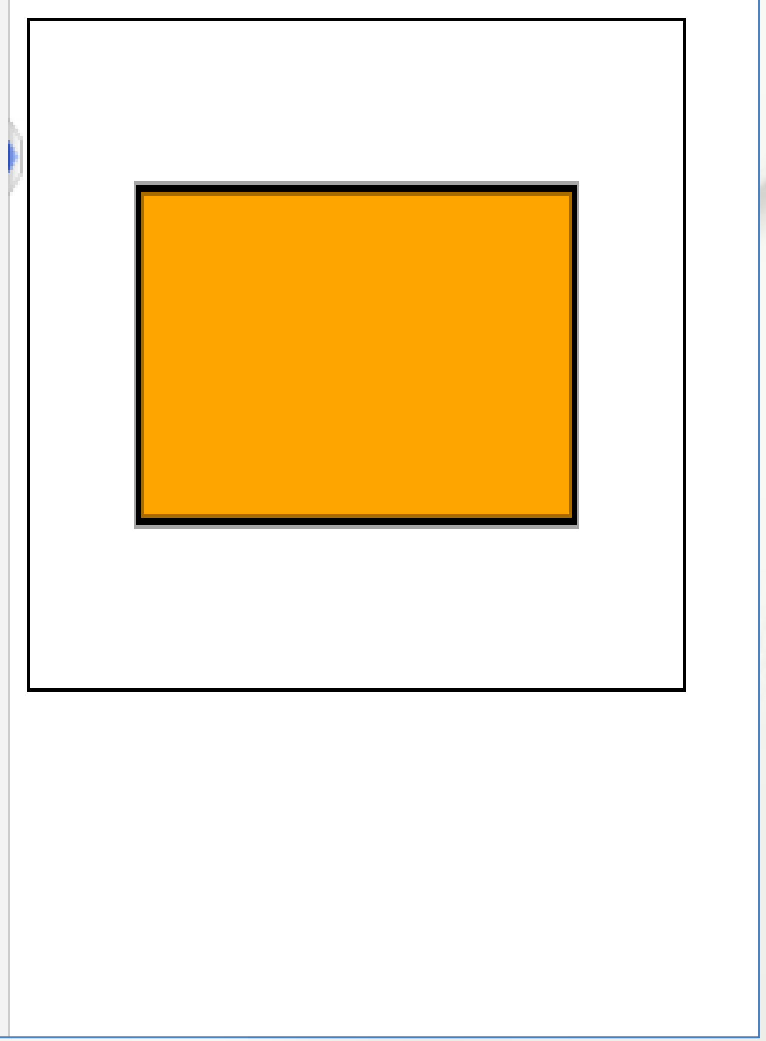


HTML5 SVG

For drawing a **rectangle** and squares, the <rect> SVG element is the most appropriate way. The attributes x and y of <rect> element specify the co-ordinates of the top-left corner of the rectangle. The attributes width and height specify the width and height of the shape.

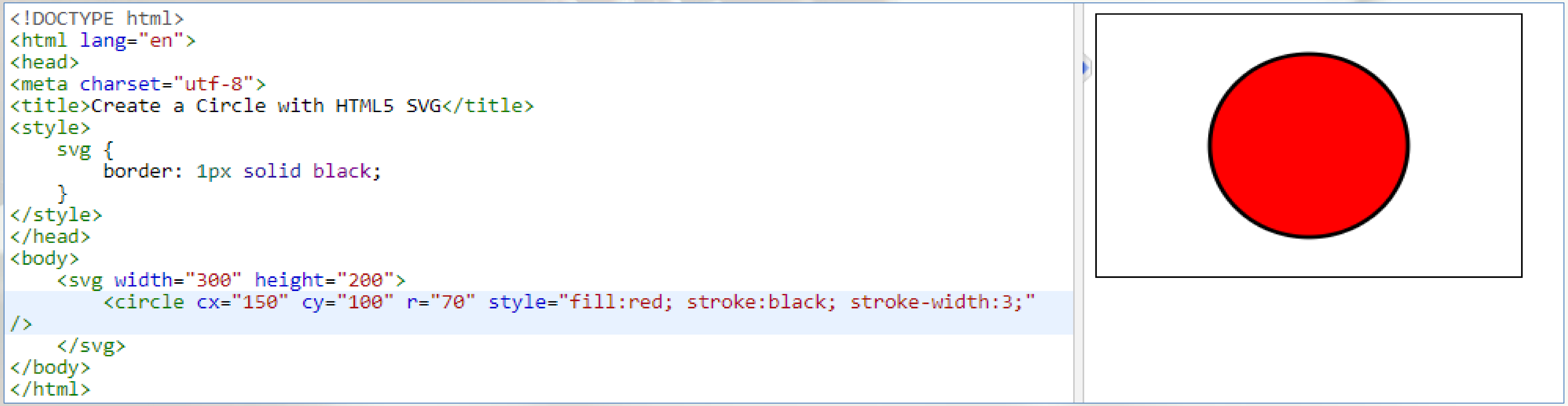


```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Rectangle with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <rect x="50" y="50" width="200" height="100" style="fill:orange; stroke:black;
stroke-width:3;" />
    </svg>
</body>
</html>
```



HTML5 SVG

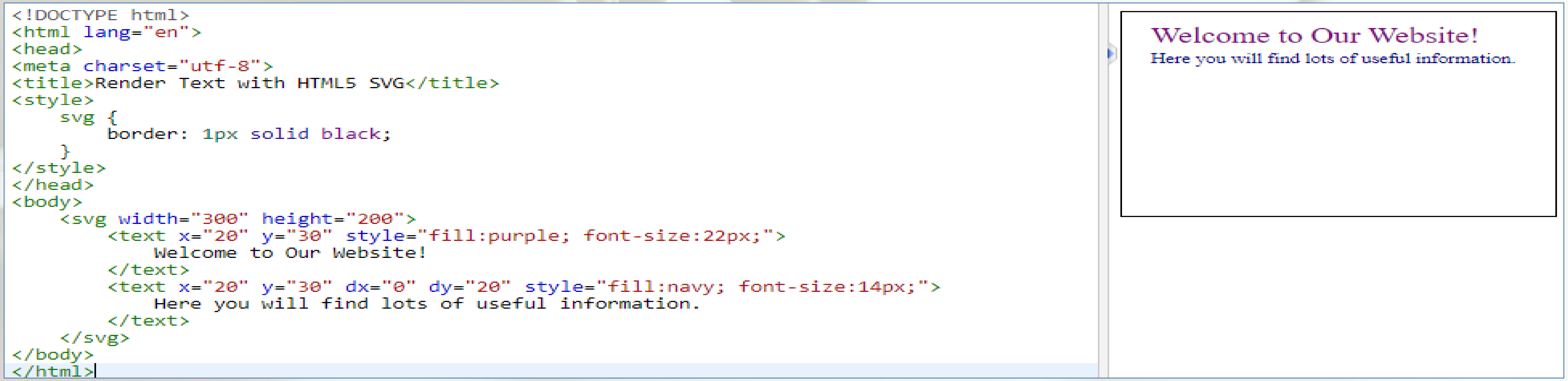
If a **circle** is the chosen shape, the SVG element <circle> is the most suitable. The attributes cx and cy of the <circle> element specifies the co-ordinates of the center of the circle and the attribute r identifies the radius of the circle. Still, if the attributes cx and cy are absent or not specified, the center of the circle is set to (0,0).



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Create a Circle with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <circle cx="150" cy="100" r="70" style="fill:red; stroke:black; stroke-width:3;"
/>
    </svg>
</body>
</html>
```

HTML5 SVG

It is also possible to **draw text** with SVG. The text in SVG is rendered as a graphic so the web designer can use all the graphic transformation to it, but it still performs as text, so it can be selected and copied as text by the user. The attributes x and y of the <text> element identify the location of the top-left corner in absolute terms although the attributes dx and dy indicates the relative location.

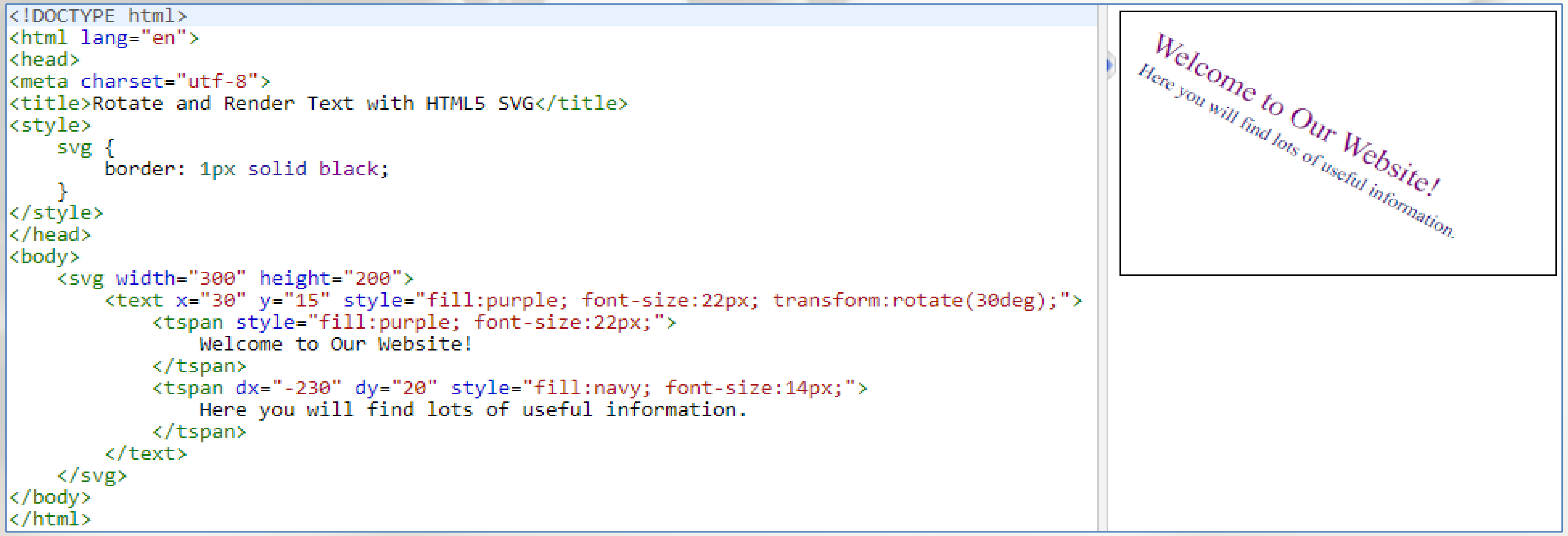


```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Render Text with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <text x="20" y="30" style="fill:purple; font-size:22px;">
            Welcome to Our Website!
        </text>
        <text x="20" y="30" dx="0" dy="20" style="fill:navy; font-size:14px;">
            Here you will find lots of useful information.
        </text>
    </svg>
</body>
</html>
```

Welcome to Our Website!
Here you will find lots of useful information.

HTML5 SVG

In alternative, web designers may use the <tspan> element to resize or relocate the span of text included within a <text> element. The text is included in separate tspans, but inside the same text element can all be selected at the same moment — when clicking and dragging to select the text. Yet, the text in separate text elements cannot be picked at the same time.



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Rotate and Render Text with HTML5 SVG</title>
<style>
    svg {
        border: 1px solid black;
    }
</style>
</head>
<body>
    <svg width="300" height="200">
        <text x="30" y="15" style="fill:purple; font-size:22px; transform:rotate(30deg);">
            <tspan style="fill:purple; font-size:22px;">
                Welcome to Our Website!
            </tspan>
            <tspan dx="-230" dy="20" style="fill:navy; font-size:14px;">
                Here you will find lots of useful information.
            </tspan>
        </text>
    </svg>
</body>
</html>
```

HTML5 SVG

Even though the new graphical elements <canvas> and <svg> have been introduced by HTML 5 in order to create high-quality graphics on the web, they differ quite a lot.

In the table below, the differences between both are summarized, and this will help learners on how to use them in an appropriate, effective way:

SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixel)
Multiple graphical elements, which become the part of the page's DOM tree	Single element similar to <img> in behavior. Canvas diagram can be saved to PNG or JPG format
Modified through script and CSS	Modified through script only
Good text rendering capabilities	Poor text rendering capabilities
Give better performance with smaller number of objects or larger surface, or both	Give better performance with larger number of objects or smaller surface, or both
Better scalability. Can be printed with high quality at any resolution. Pixelation does not occur	Poor scalability. Not suitable for printing on higher resolution. Pixelation may occur

HTML5 SVG

Even though the new graphical elements <canvas> and <svg> have been introduced by HTML 5 in order to create high-quality graphics on the web, they differ quite a lot.

In the table below, the differences between both are summarized, and this will help learners on how to use them in an appropriate, effective way:

SVG	Canvas
Vector based (composed of shapes)	Raster based (composed of pixel)
Multiple graphical elements, which become the part of the page's DOM tree	Single element similar to <img> in behavior. Canvas diagram can be saved to PNG or JPG format
Modified through script and CSS	Modified through script only
Good text rendering capabilities	Poor text rendering capabilities
Give better performance with smaller number of objects or larger surface, or both	Give better performance with larger number of objects or smaller surface, or both
Better scalability. Can be printed with high quality at any resolution. Pixelation does not occur	Poor scalability. Not suitable for printing on higher resolution. Pixelation may occur

HTML5 Audio

- This section is intended to explain how to embed audio in an HTML document.

- As web browsers did not have a uniform standard for embedding media files as audio, it was not an easy task to perform in the past. However, there are many ways to embed sound in a webpage, from simply use a simple link to using the HTML5 <audio> element. This element provides a standard way to insert audio in web pages. Since the audio element is somewhat new, it runs in most of the modern web browsers.

- There are many ways of inserting an audio into a HTML5 document. One of them is by using the browser’s default group of controls, with one source defined by the src attribute, as it can be verified in this code, in which it is possible to hear a group of birds singing.

HTML5 Audio

- Another way can be achieved by using the <object> element, which is used for embedding different types of media files. This example embeds an audio file into a web page following the aforementioned method. It should be stated that the <object> element is not supported broadly and it depends on the type of the object that is being implanted. Other methods like HTML5 <audio> element or third-party HTML5 audio players could be a better option in a lot of cases.

- Finally, the <embed> element can also be another way of inserting media in an HTML document, following this example. Even though the <embed> element is superbly supported in nowadays browsers and defined as standard in HTML5, the inserted audio might not be played due to absence of browser support for that file format or inaccessibility of plugins.

HTML5 Video

- It is now time to learn how to embed video content into an HTML document.

- Pretty much like sound, video contents were also difficult to insert into a web page, and for the same reason (web browsers did not have a uniform standard for defining embedded media files like video). In the next paragraphs, many ways of inserting these contents will be explained.

HTML5 Video

The newly introduced <video> element works in most of the modern browsers. This example explains how to simply insert a video into the HTML document, using the browser default set of controls, with one source defined by the src attribute.

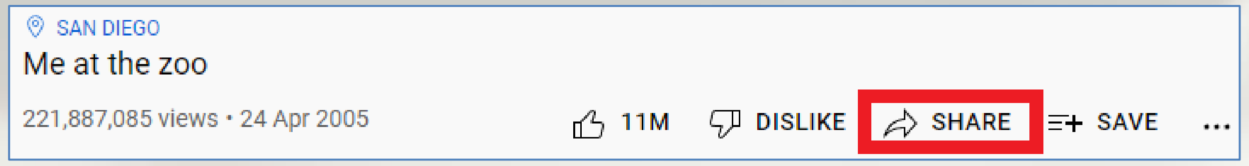
The <object> element is also used to embed different types of media files. Following this example, one can understand how to embed a Flash video into a webpage (only browsers/applications supporting Flash will be able to play it). It should be noted that the <object> element is not supported extensively and depends a lot on the type of the object embedded. Other methods could be a better choice in many cases as, for instance, iPad and iPhone device cannot display Flash videos.

HTML5 Video

And what about **embedding YouTube videos**? That is actually the most simple and common way of embedding video files in webpages nowadays. The web designer must simply upload the video on YouTube and insert HTML code to display the video on his/her webpage. Here is a step-by-step mini guide:

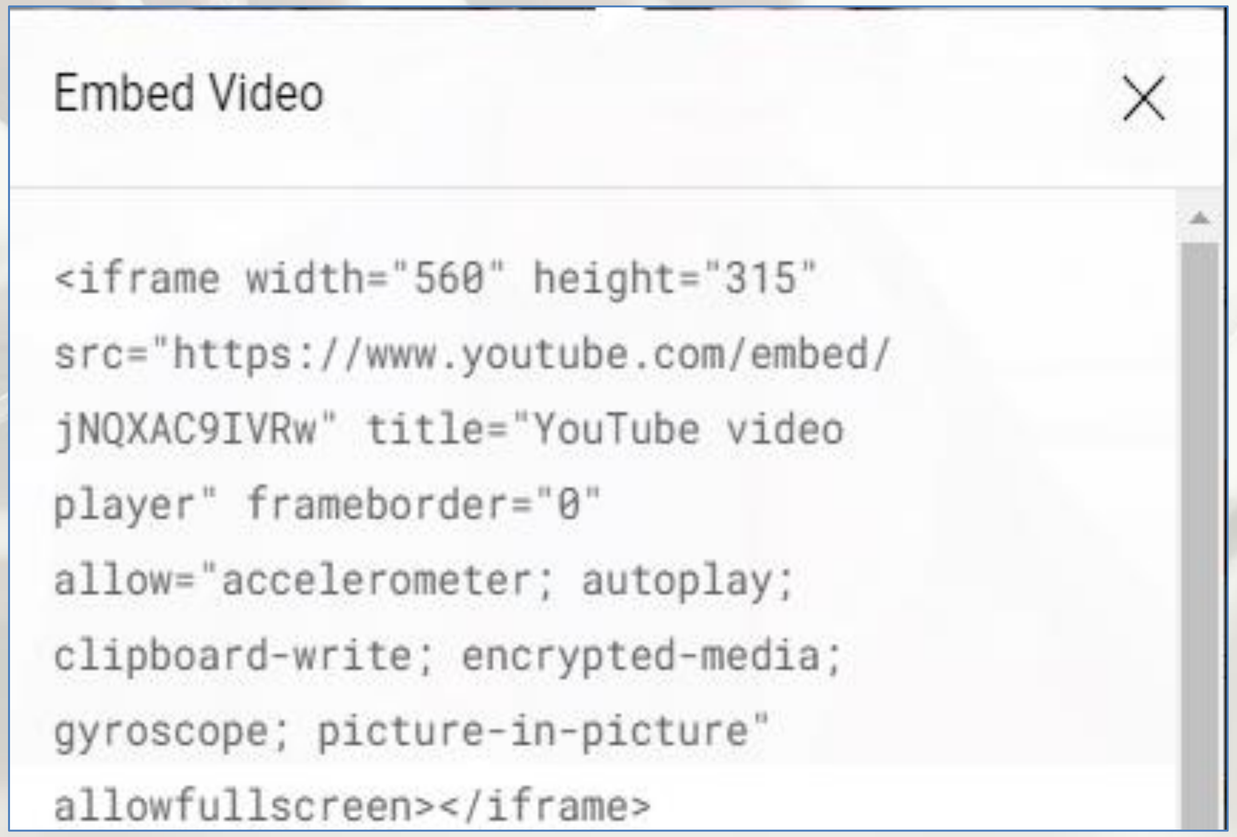
Step 1 – Upload a video on YouTube.

Step 2 – After uploading a video on YouTube, the web designer should look for the 'Share' button, which is located below a video running on the platform's video player, just like as follows:



HTML5 Video

When clicking the ‘Share’ button, a share panel will open exhibiting some more options. Now, the ‘**Embed**’ button should be clicked, as it will generate the HTML code to directly embed the video into the web page. For that to happen, the web designer should copy and paste that code into the HTML document.

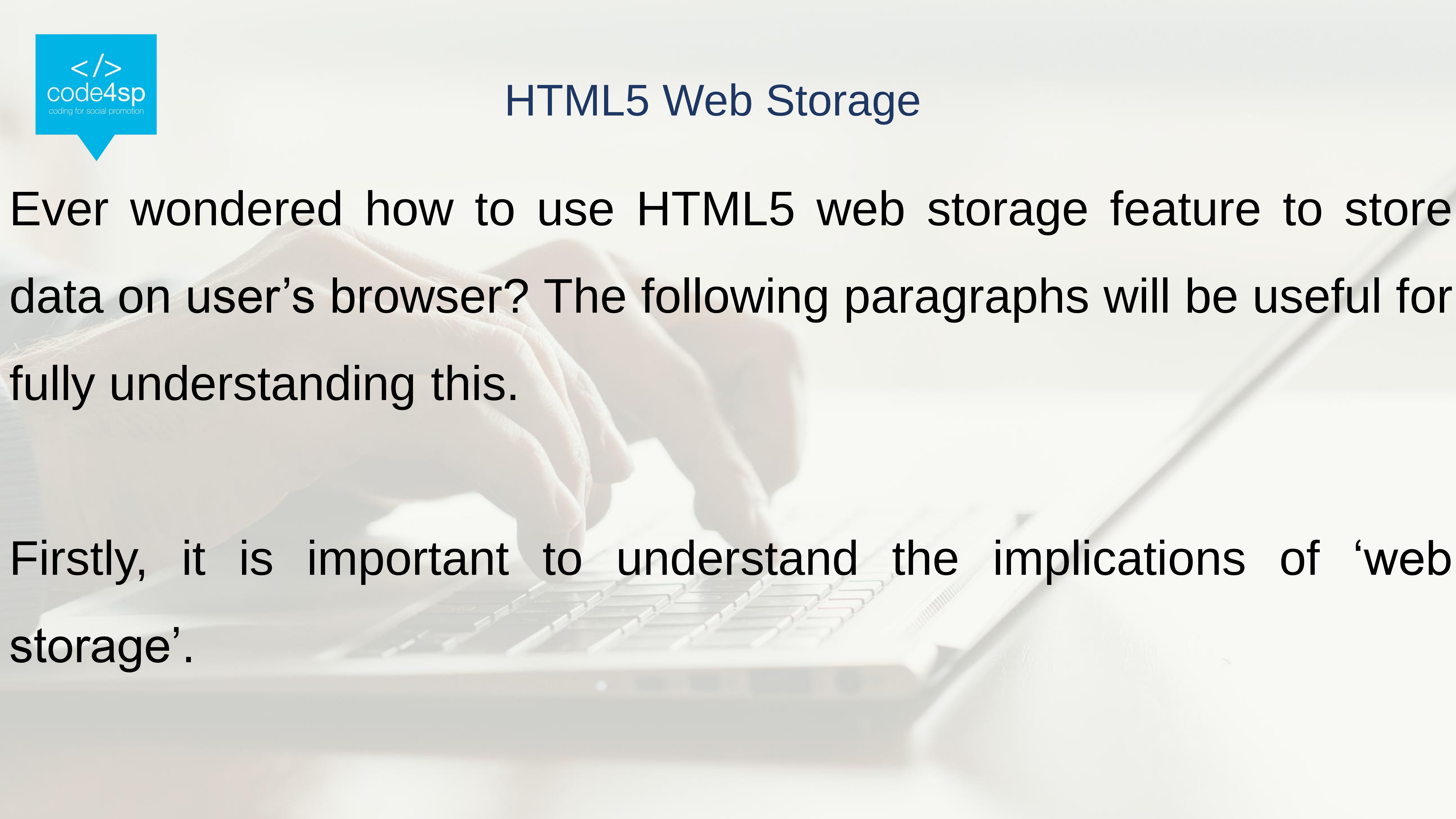


```
<iframe width="560" height="315"
src="https://www.youtube.com/embed/
jNQXAC9IVRw" title="YouTube video
player" frameborder="0"
allow="accelerometer; autoplay;
clipboard-write; encrypted-media;
gyroscope; picture-in-picture"
allowfullscreen></iframe>
```

HTML5 Video

This code can be furtherly customized, and for that to happen the web designer just needs to select the customization option given just below the embed-code input box.

The insertion of a YouTube video on a webpage is explained in this example.



HTML5 Web Storage

Ever wondered how to use HTML5 web storage feature to store data on user’s browser? The following paragraphs will be useful for fully understanding this.

Firstly, it is important to understand the implications of ‘web storage’.

HTML5 Web Storage

With web storage, web applications can store data locally within the user's browser. Prior to HTML5, application data had to be stored in cookies, incorporated in every server request. Web storage is more secure, and substantial amounts of data can be stored locally, without impacting website performance.

The information kept in the web storage is not sent to the web server as opposite to the cookies where data is delivered to the server with every request. Moreover, cookies only allow to store a small amount of data (nearly 4KB), while the web storage permits to store up to 5MB.

HTML5 Web Storage

There are two types of web storage:

Local storage — makes use of the localStorage object to store data for the entire website on a *permanent basis*. That being said, the stored local data will be available on the next day, the next week, or the next year unless it is removed.

Session storage — it uses the sessionStorage object to store data on a *temporary basis*, for a single browser window or tab. The data vanishes when session ends, for instance when the user shuts that browser window or tab.

HTML5 Web Storage

As regards the **local storage**, each piece of data is collected in a key/value pair. The key identifies the name of the information (i.e. 'first_name'), and the value is the value related to the same key (i.e. ‘Peter’). The following JS code expresses the following:

- localStorage.setItem (key, value) stores the value associated with a key.
- localStorage.getItem (key) saves the value associated with the key.

It is also possible to remove a particular item from the storage, by passing the key name to the removeItem() method,
i.e. localStorage.removeItem("first_name").

HTML5 Web Storage

It is also possible to remove a particular item from the storage, by passing the key name to the removeItem() method, i.e. localStorage.removeItem("first_name").

However, if the web designer intends to remove the complete storage, he/she should use the clear() method, i.e. localStorage.clear(). The clear() method simply clears all key/value pairs from localStorage at once, **so it must be used cautiously**. The web storage data will not be accessible between different browsers.

Finally, the sessionStorage object works similarly to localStorage, except that it stores the data only for one session. The following example is quite explanatory on how this works.

HTML5 Application Cache

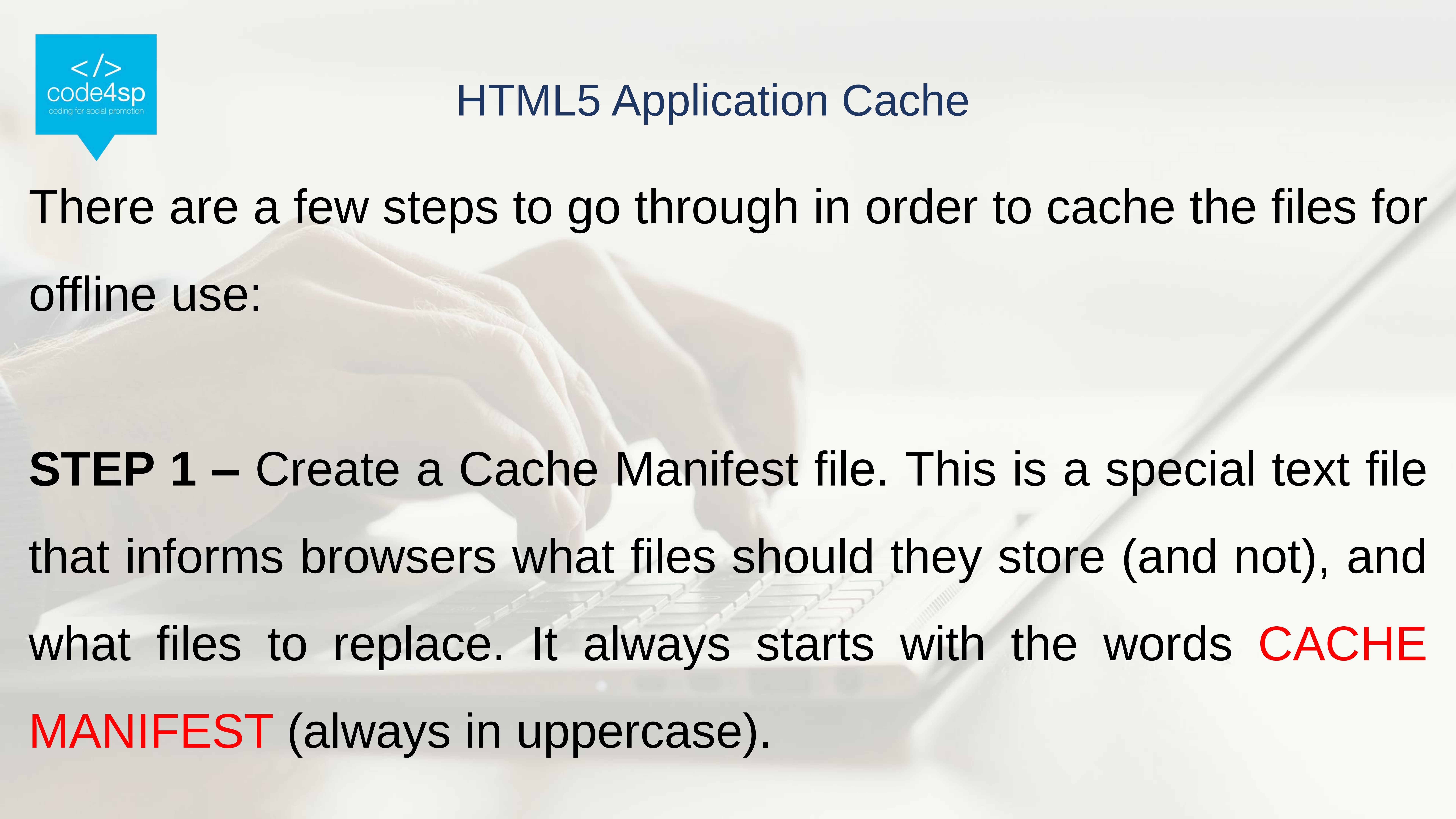
This section will be useful on how to create offline applications using **HTML5 caching feature**.

It is well-known that most web-based application will not work if the web designer is offline. However, HTML5 brings in an application cache mechanism that allows the browser to automatically save the HTML file and all the other resources that requires to display it properly on the local machine, that way the browser can still access the web page and its resources without establishing a connection to the internet. This is supported in all major modern web browsers (Firefox, Chrome, Opera, Safari, and Internet Explorer 10 and above.

HTML5 Application Cache

There are several advantages in using this feature:

- **Offline browsing** — Visitors can use the application even when they are not online or there are unexpected interruptions in the network connection.
- **Improve performance** — Cached resources load directly from the user's machine instead of the remote server so web pages load quicker and perform better.
- **Reduce HTTP request and server load** — The browser will only have to download the updated/changed resources from the remote server that reduce the HTTP requests and saves valuable bandwidth as well as decrease the load on the web server.

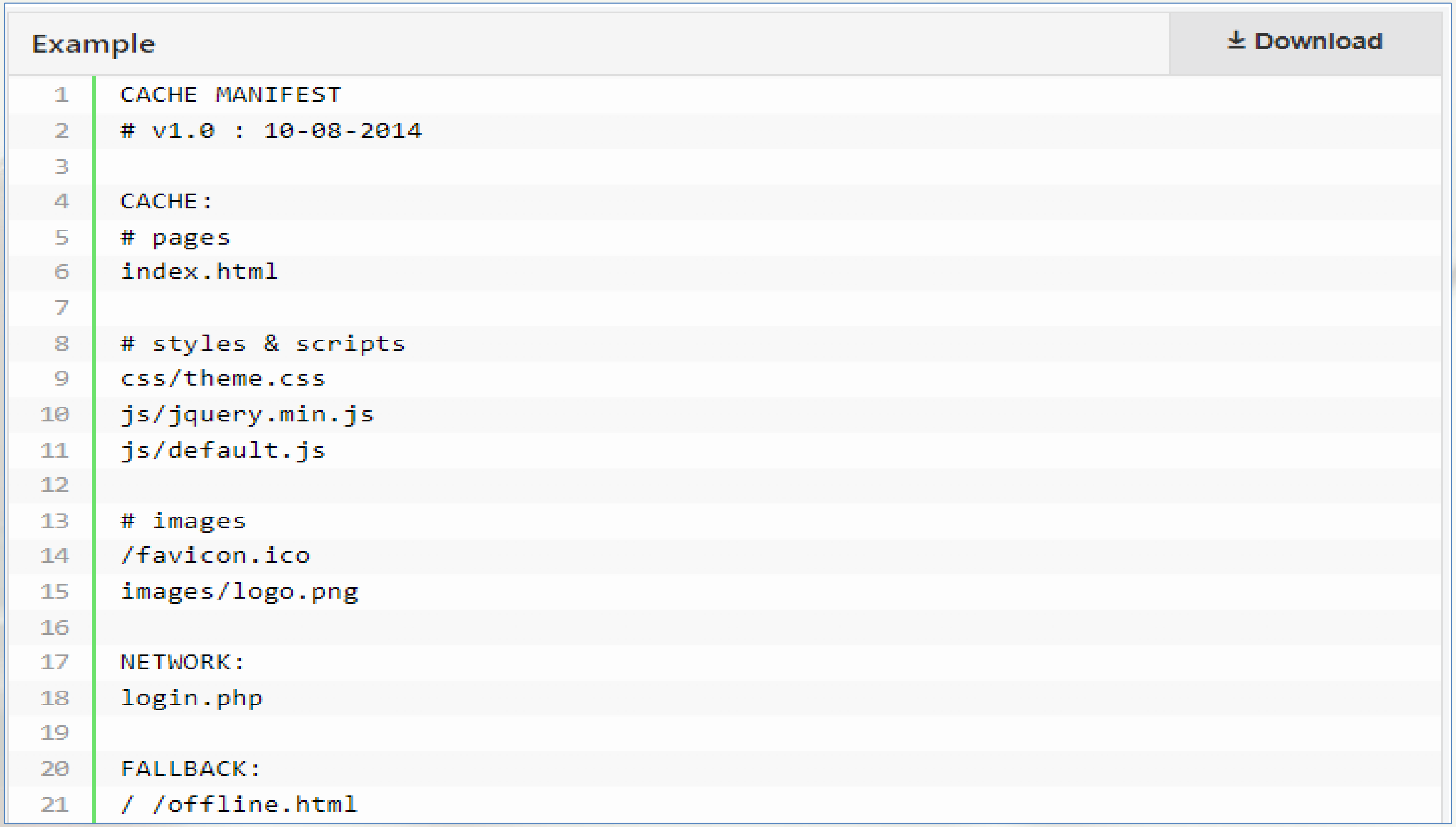


HTML5 Application Cache

There are a few steps to go through in order to cache the files for offline use:

STEP 1 – Create a Cache Manifest file. This is a special text file that informs browsers what files should they store (and not), and what files to replace. It always starts with the words CACHE MANIFEST (always in uppercase).

HTML5 Application Cache



```
CACHE MANIFEST
# v1.0 : 10-08-2014

CACHE:
# pages
index.html

# styles & scripts
css/theme.css
js/jquery.min.js
js/default.js

# images
/favicon.ico
images/logo.png

NETWORK:
login.php

FALLBACK:
/ /offline.html
```

HTML5 Application Cache

It is now time to explain the coding of last slide's picture.

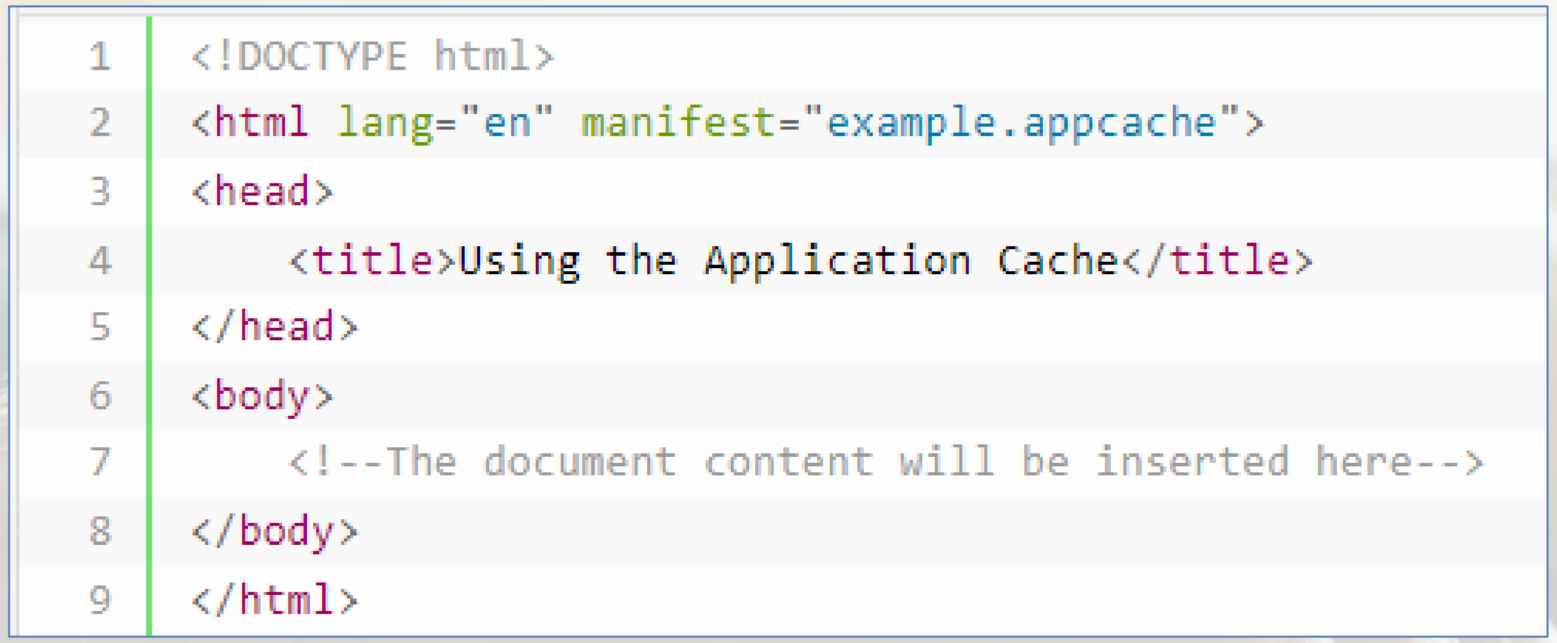
- Firstly, it is important to understand that manifest files can have three different sections: **CACHE, NETWORK**, and **FALLBACK**.
- The files listed under the CACHE: section header (or right after the CACHE MANIFEST line) are clearly cached after they are downloaded for the first time;
- Files under the NETWORK: section header are white-listed resources that are never cached and are only available online. It means users cannot never access login.php page when they're offline;
- The FALLBACK: section specifies alternative pages the browser should use in case the connection to the server cannot be done. Each entry in this section lists two URIs — first is the primary resource, the second is the fallback. For instance, in *Figure 98* case, offline.html page will be displayed if the user is offline. Also, both URIs must be from the same origin as the manifest file.
- It should be noted that lines starting with '#' (hash symbol) are comment lines.

HTML5 Application Cache

STEP 2 – Use the cache manifest file. After creating it, the web designer should upload the cache manifest file on the web server — making sure the web server is configured to serve the manifest files with the MIME type text/cache-manifest.

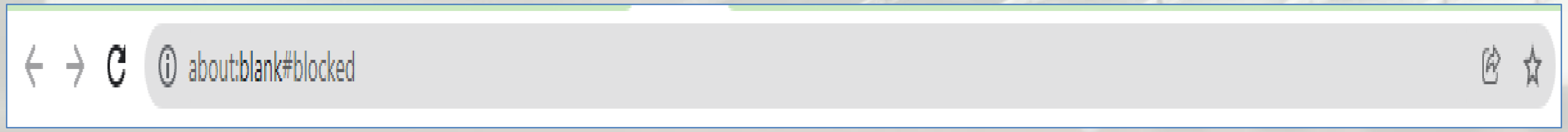
To make the cache manifest work, the web designer will need to enable it in the web pages, by adding the manifest attribute to the root <html> element, as shown by the picture on the next slide.

HTML5 Application Cache



```
<!DOCTYPE html>
<html lang="en" manifest="example.appcache">
<head>
    <title>Using the Application Cache</title>
</head>
<body>
    <!--The document content will be inserted here-->
</body>
</html>
```

If the user is online, the result for this code will be the following:



about:blank#blocked

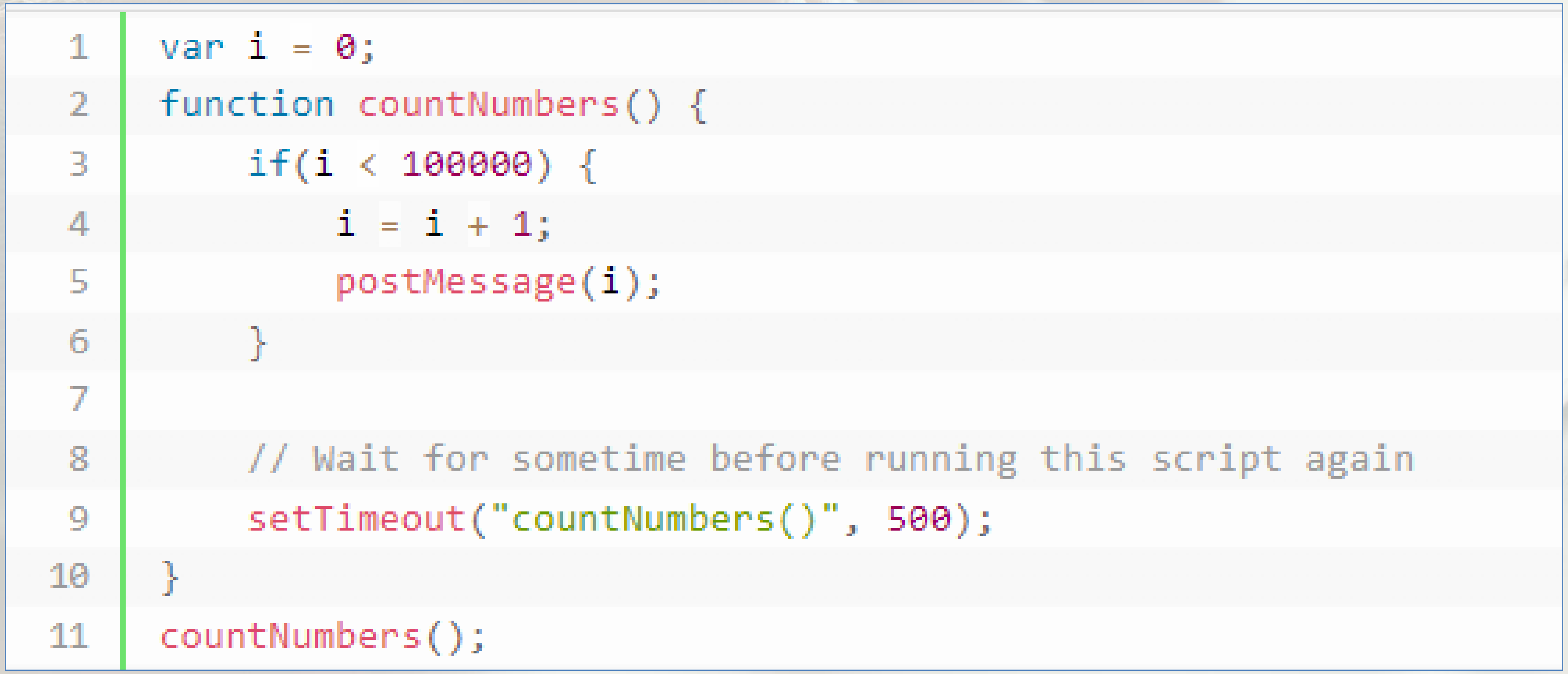
HTML5 Web Workers

This section will be essentially useful for further learning on JS topics, as it will teach how to use HTML5 web worker to run JS code in the background. So, learners must come back if they experience any difficulties.

If one tries to perform intensive, time-consuming, and heavy calculations demanding tasks with JavaScript, it probably will freeze up the webpages and will prevent users from doing anything until the job is done. Why? Well, because JS code always runs in the foreground. However, HTML5 has a new technology ('web worker') created to perform background work apart from other user-interface scripts, without impacting the performance of the page. Distinct from normal JS operations, web worker does not interrupt the user and the web page stays responsive because they are running the tasks in the background.

HTML5 Web Workers

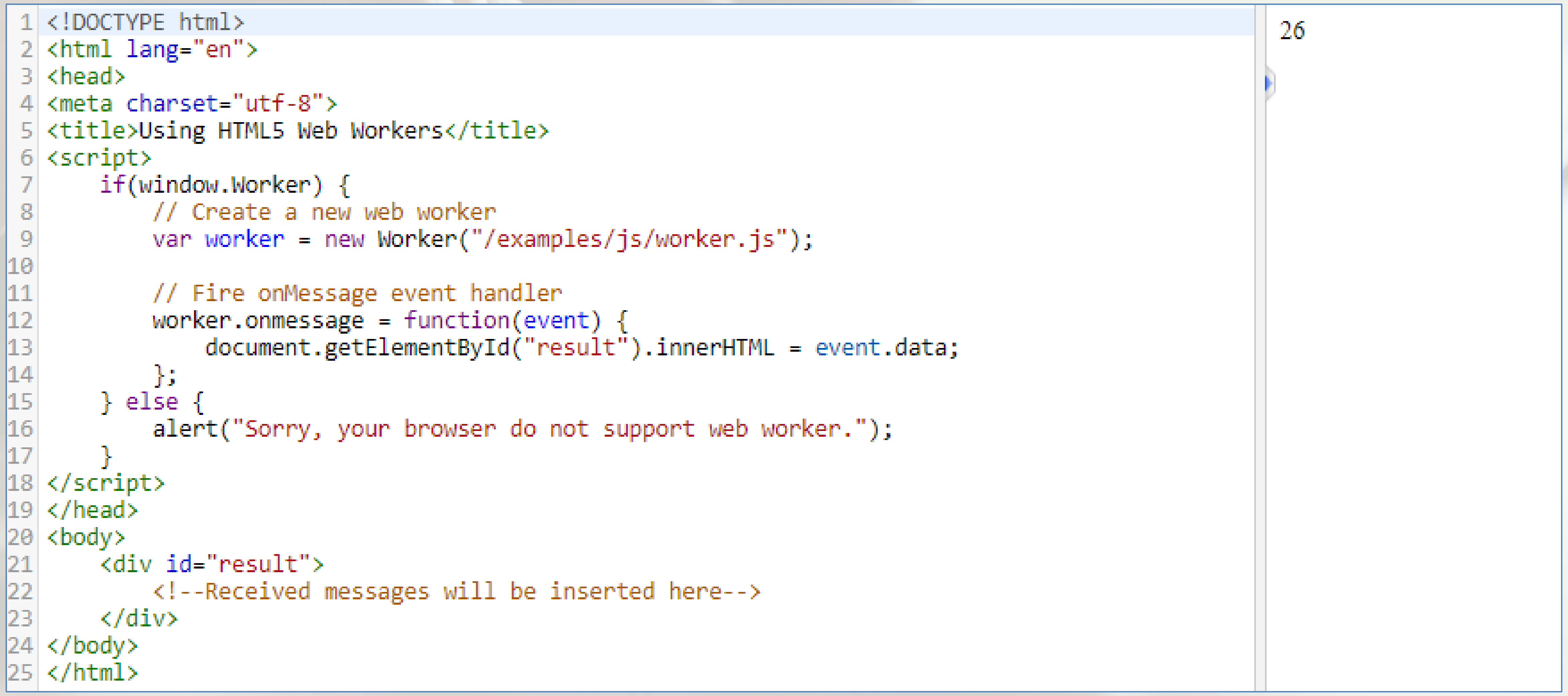
Web workers are specially useful for performing a time-consuming task. So, in the first, example, a simple JS task that counts from zero to 100 000 will be created (name of the file should be worker.js), as follows:



```
var i = 0;
function countNumbers() {
    if(i < 100000) {
        i = i + 1;
        postMessage(i);
    }

    // Wait for sometime before running this script again
    setTimeout("countNumbers()", 500);
}
countNumbers();
```

HTML5 Web Workers

So, now that the web worker file has been created, it is time to start the web worker from an HTML document that runs the code inside the file named "worker.js" in the background and gradually shows the result on the web page. It should be noted that the number in the right will always be growing until it reaches 100,000.



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Using HTML5 Web Workers</title>
<script>
    if(window.Worker) {
        // Create a new web worker
        var worker = new Worker("/examples/js/worker.js");

        // Fire onMessage event handler
        worker.onmessage = function(event) {
            document.getElementById("result").innerHTML = event.data;
        };
    } else {
        alert("Sorry, your browser do not support web worker.");
    }
</script>
</head>
<body>
    <div id="result">
        <!--Received messages will be inserted here-->
    </div>
</body>
</html>
```

```
26
```

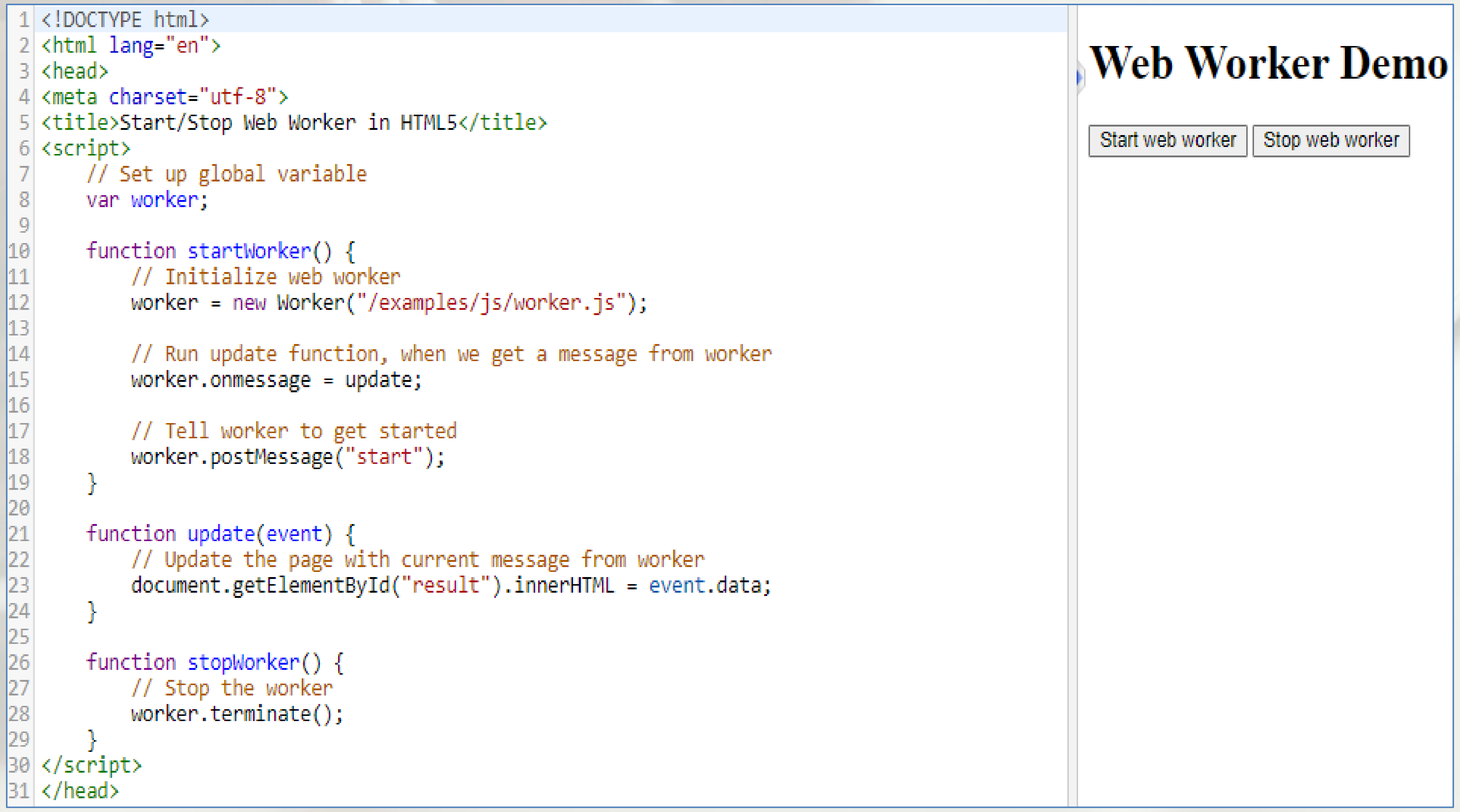
HTML5 Web Workers

- Now explaining what is going on the example from above, the statement **var worker = new Worker("worker.js");** creates a new web worker object, which is used to communicate with the web worker. When the worker posts a message, it triggers the **onmessage** event handler (line 14) that permits the code to receive messages from the web worker.

- The **event.data** element comprises the message sent from the web worker. For the record, the code that a worker runs is always stored in a separate JavaScript file to prevent web developer from writing the web worker code that makes an attempt to use global variables or directly open the elements on the web page.

HTML5 Web Workers

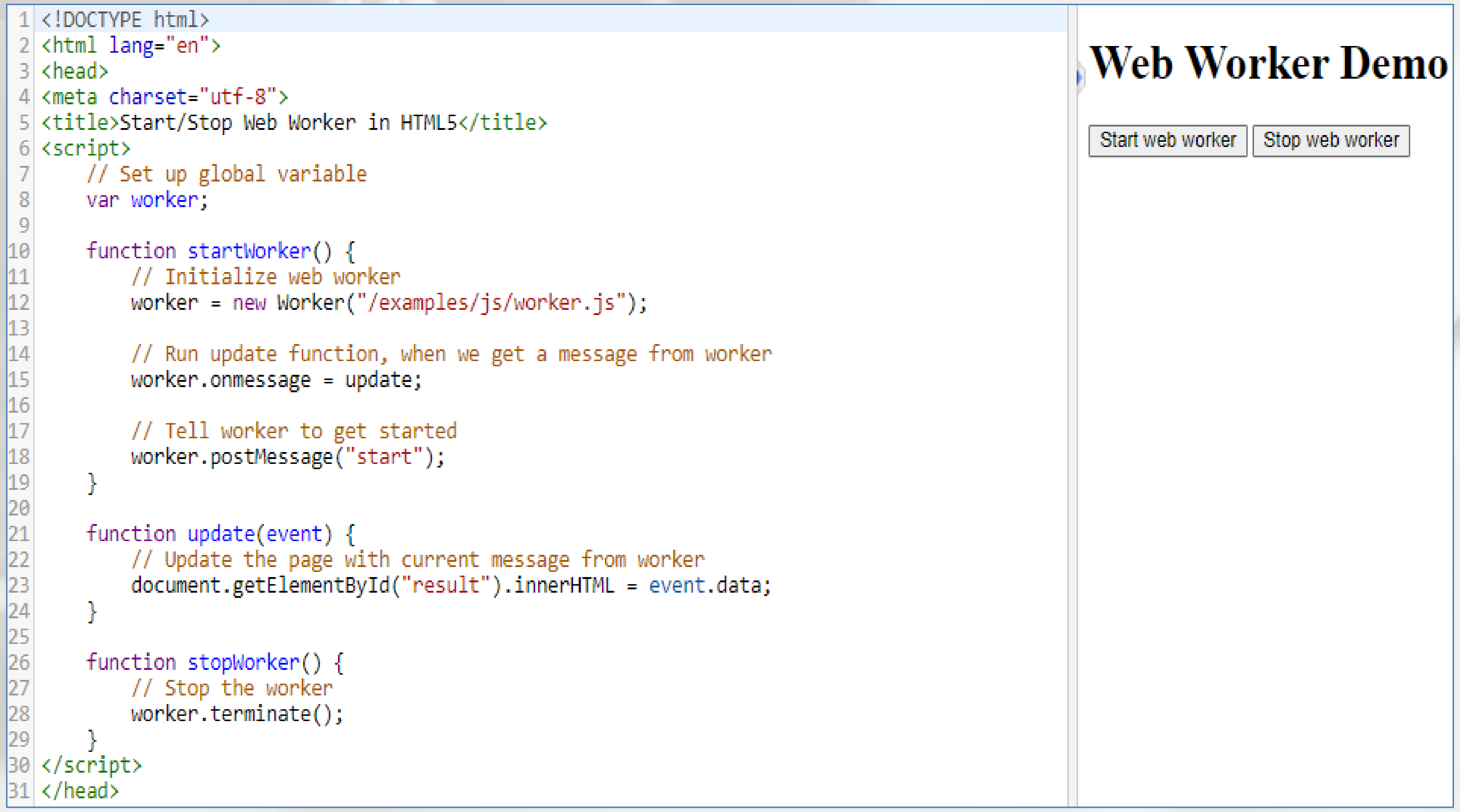
It is also possible to **put an end to a running worker** in the middle of the operation, following the example below:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Start/Stop Web Worker in HTML5</title>
<script>
    // Set up global variable
    var worker;

    function startWorker() {
        // Initialize web worker
        worker = new Worker("/examples/js/worker.js");

        // Run update function, when we get a message from worker
        worker.onmessage = update;

        // Tell worker to get started
        worker.postMessage("start");
    }

    function update(event) {
        // Update the page with current message from worker
        document.getElementById("result").innerHTML = event.data;
    }

    function stopWorker() {
        // Stop the worker
        worker.terminate();
    }
</script>
</head>
```

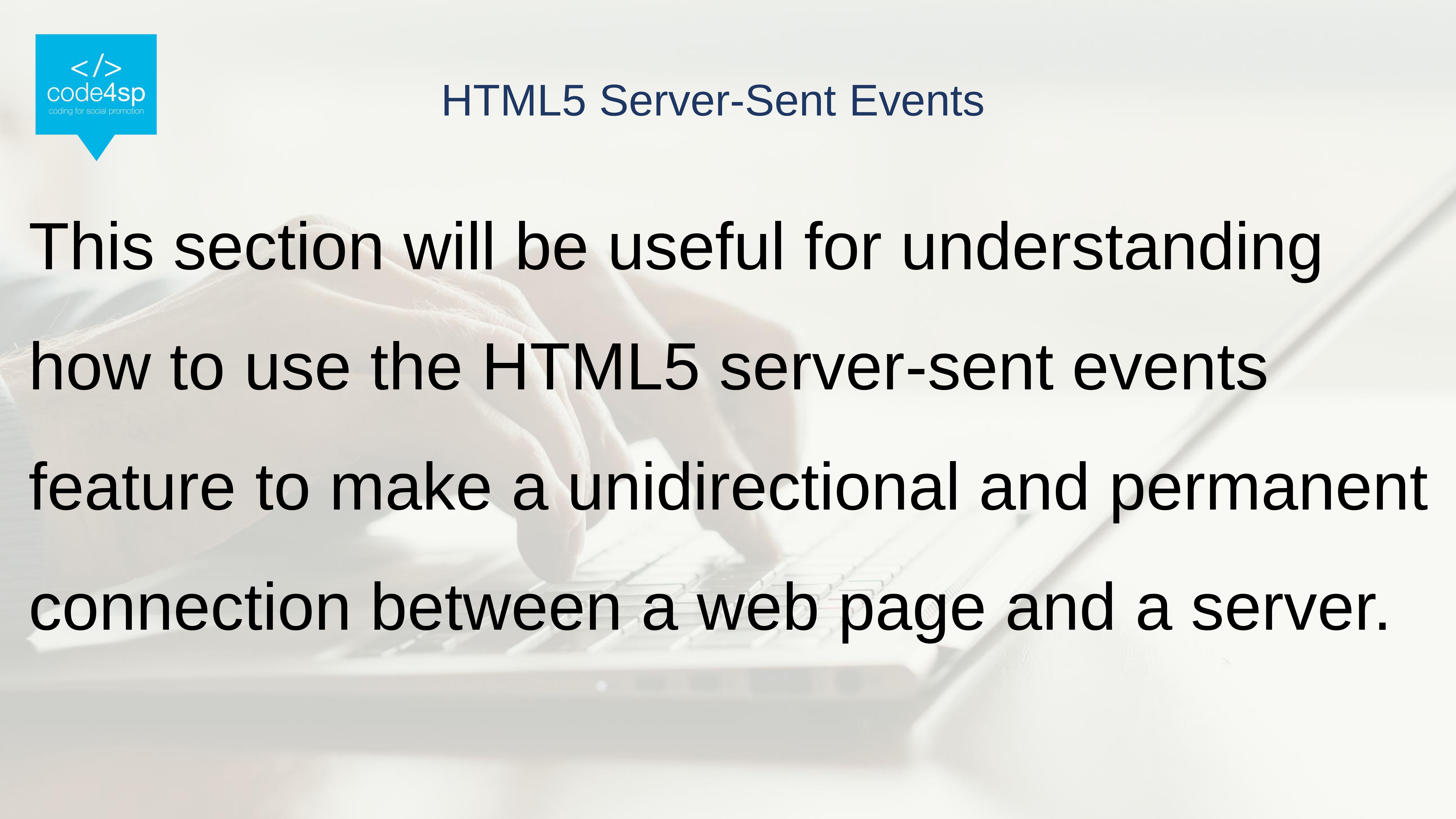
HTML5 Web Workers

It is also possible to **put an end to a running worker** in the middle of the operation, following the example below:



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Start/Stop Web Worker in HTML5</title>
<script>
    // Set up global variable
    var worker;

    function startWorker() {
        // Initialize web worker
        worker = new Worker("/examples/js/worker.js");

        // Run update function, when we get a message from worker
        worker.onmessage = update;

        // Tell worker to get started
        worker.postMessage("start");
    }

    function update(event) {
        // Update the page with current message from worker
        document.getElementById("result").innerHTML = event.data;
    }

    function stopWorker() {
        // Stop the worker
        worker.terminate();
    }
</script>
</head>
```

Web Worker Demo



HTML5 Server-Sent Events

This section will be useful for understanding how to use the HTML5 server-sent events feature to make a unidirectional and permanent connection between a web page and a server.

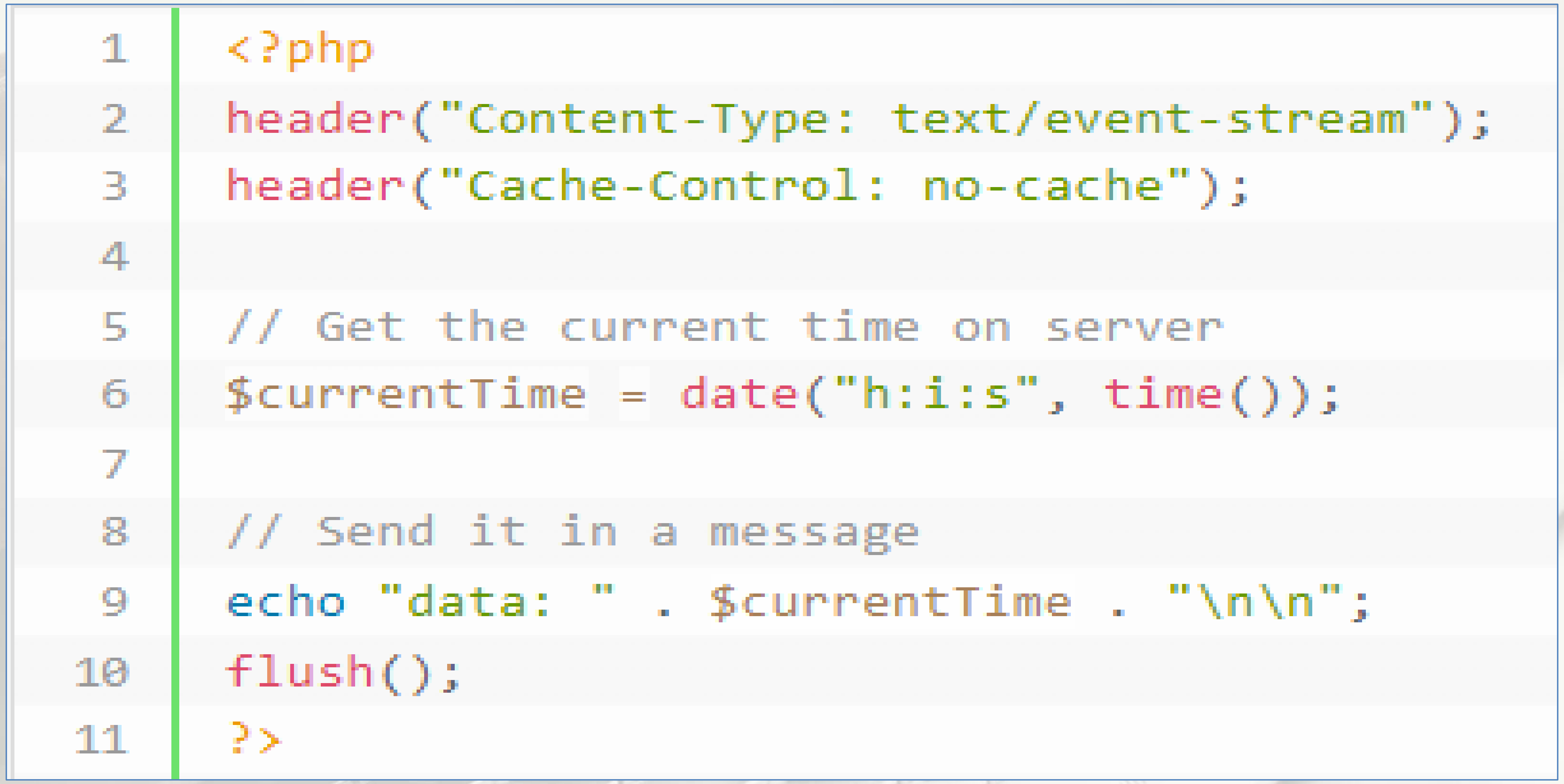
HTML5 Server-Sent Events

- HTML5 server-sent event is an innovative way for the web pages to communicate with a web server. Nevertheless, there are some circumstances where web pages need a longer-term connection to the web server, for example, on stock quotes on finance websites where price updated automatically or game tickers running on various sports websites.

- Such things are feasible with the HTML5 server-sent events, as it makes available for a web page to hold an open connection to a web server, in a way that the web server can send a new response mechanically at whatever time. At this point, there is no need to reconnect and run the same server script from the beginning repeatedly.

- For a better understanding of the aforementioned concepts, a PHP file named "server_time.php" and type the following script into it. This file will merely report the present time of the web server's built-in clock in regular intervals:

HTML5 Server-Sent Events



```php
<?php
header("Content-Type: text/event-stream");
header("Cache-Control: no-cache");

// Get the current time on server
$currentTime = date("h:i:s", time());

// Send it in a message
echo "data: " . $currentTime . "\n\n";
flush();
?>
```

NOTE: A PHP file is a plain-text file which contains code written in the PHP programming language. Since PHP is a server-side (back-end) scripting language, the code written on it is executed on the server. In fact, a PHP file may contain plain text, HTML tags, or code as per the PHP syntax. PHP is often used to develop web applications that are processed by a PHP engine on the web server.

HTML5 Server-Sent Events

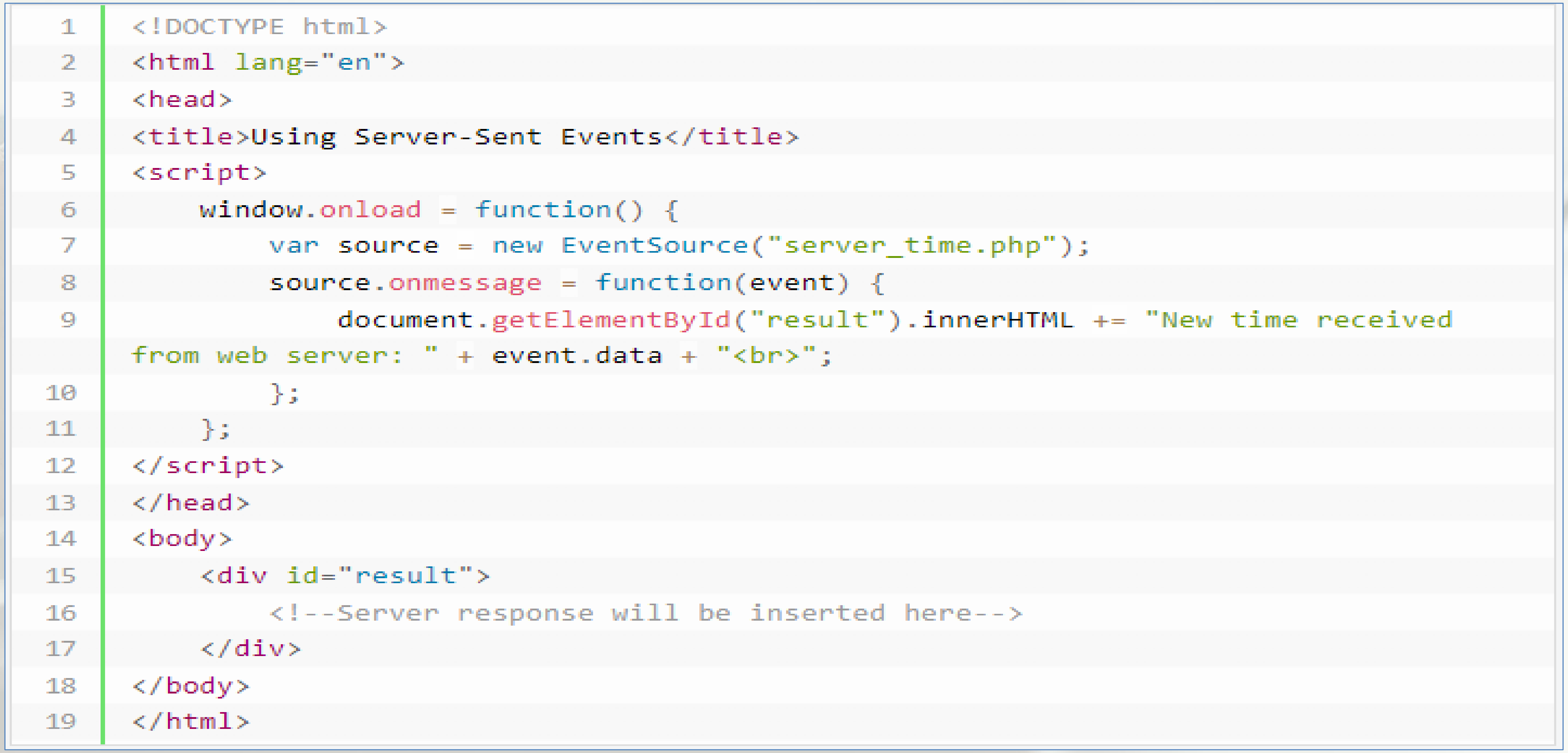
- The first two line of the PHP script sets two crucial headers. Number one, it sets the MIME type to text/event-stream, which is needed by the server-side event standard. The second line informs the web server to turn off caching or else the output of the script may be cached.

- Every message send through HTML5 server-sent events must start with the text data: followed by the actual message text and the new line character sequence (\n\n).

- And lastly, the PHP flush() function has been used to ensure that the data is sent immediately, rather than buffered until the PHP code is complete.

- Regarding on **how to process messages in a web page**, the EventSource object is used to receive server-sent event messages. In the example below, learners will see how a HTML document simply receives the current time reported by the web server and displays it to the web page visitors. For a better understanding, an HTML document named “demo_sse.htlm” will be created and furtherly placed in the same project directory where “server_time.php” is located.

HTML5 Server-Sent Events



```html
<!DOCTYPE html>
<html lang="en">
<head>
<title>Using Server-Sent Events</title>
<script>
    window.onload = function() {
        var source = new EventSource("server_time.php");
        source.onmessage = function(event) {
            document.getElementById("result").innerHTML += "New time received from web server: " + event.data + "<br>";
        };
    };
</script>
</head>
<body>
    <div id="result">
        <!--Server response will be inserted here-->
    </div>
</body>
</html>
```

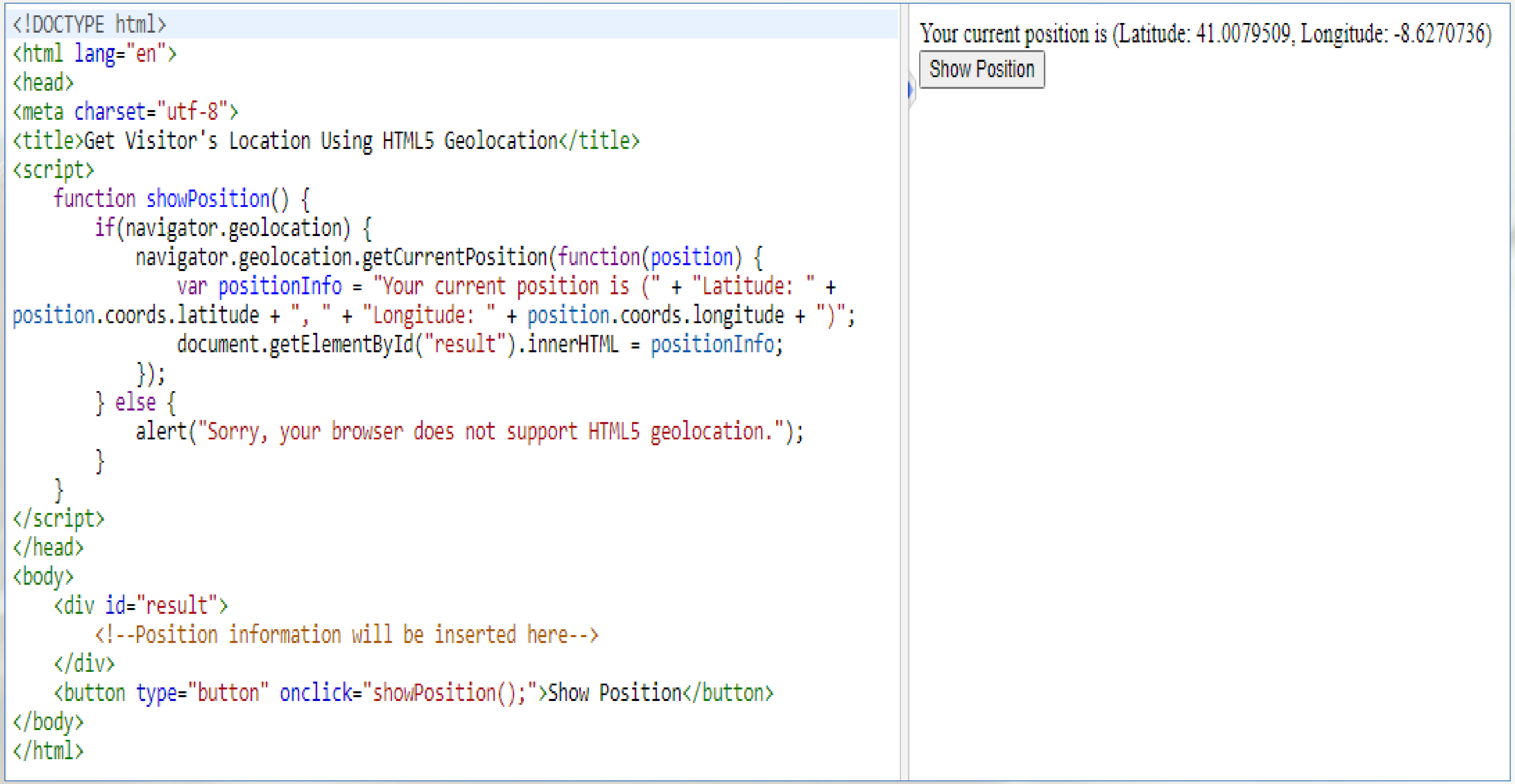
HTML5 Geolocation

- Through this subtopic, learners will get a few insights on how to use HTML5 geolocation feature for detecting user's location. This feature lets the programmer find out the geographic coordinates (latitude and longitude) of the website visitor's current location. It is especially useful for providing the best browsing experience for the visitor, since, for instance, this tool can show search results that are physically close to the user's location.

- Receiving the position information of the website visitor using the HTML5 geolocation API is not difficult. It exploits the three methods that are packed into the navigator.geolocation object — getCurrentPosition(), watchPosition() and clearWatch().

- After the user agrees to let the browser tell the web server about his/her position (web browsers will not share the visitor location with a webpage unless the user agrees to do so), the geolocation process should occur as in the image on the next slide.

HTML5 Geolocation



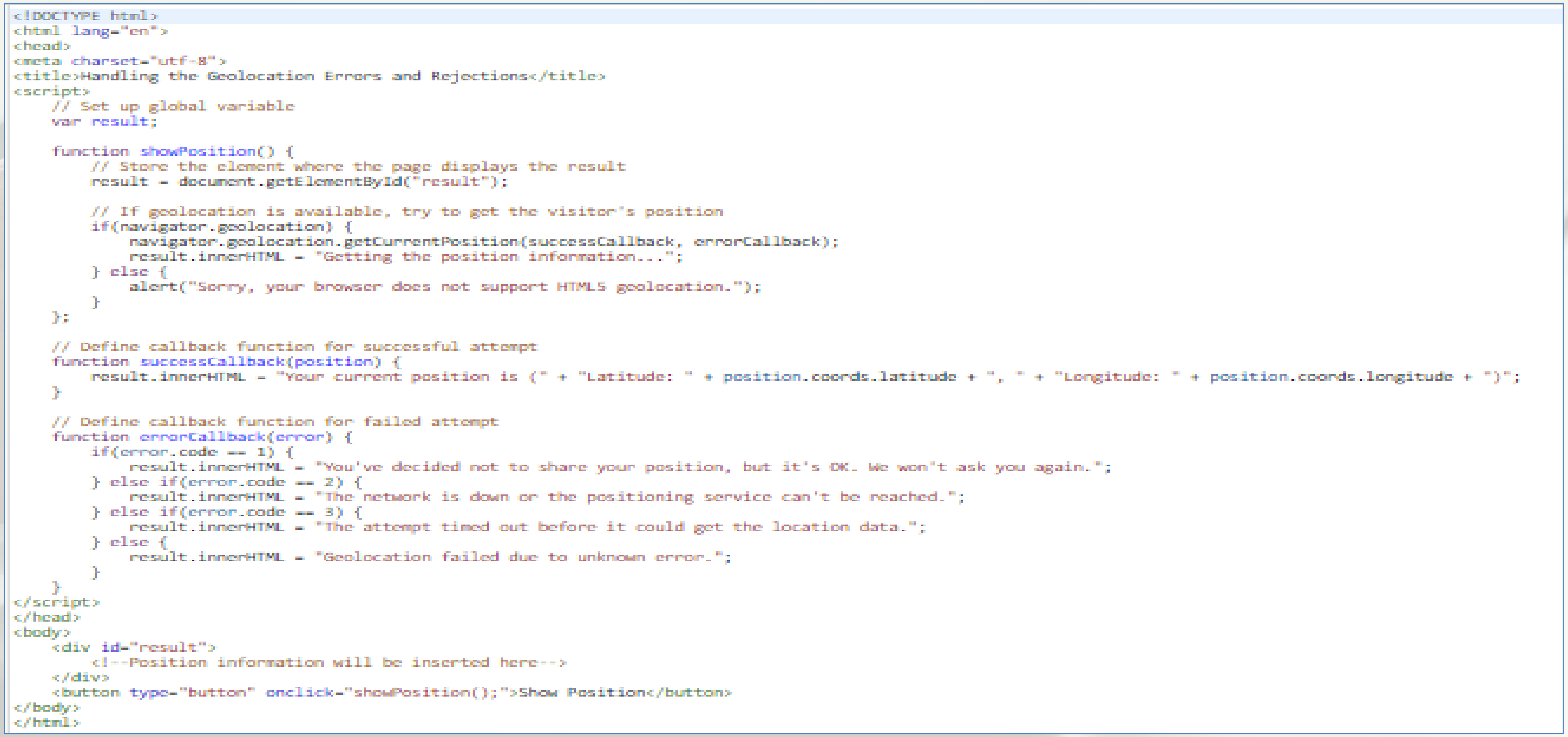
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Get Visitor's Location Using HTML5 Geolocation</title>
<script>
    function showPosition() {
        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(function(position) {
                var positionInfo = "Your current position is (" + "Latitude: " +
position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";
                document.getElementById("result").innerHTML = positionInfo;
            });
        } else {
            alert("Sorry, your browser does not support HTML5 geolocation.");
        }
    }
</script>
</head>
<body>
    <div id="result">
        <!--Position information will be inserted here-->
    </div>
    <button type="button" onclick="showPosition();">Show Position</button>
</body>
</html>
```

```
Your current position is (Latitude: 41.0079509, Longitude: -8.6270736)
```

HTML5 Geolocation

- In case a user does not intend to share his/her location data with the website, the programmer can supply two functions when calling the getCurrentLocation() function. First function is called in case geolocation attempt is successful, whereas the second is called if the geolocation attempt fails.

HTML5 Geolocation



```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Handling the Geolocation Errors and Rejections</title>
<script>
    // Set up global variable
    var result;

    function showPosition() {
        // Store the element where the page displays the result
        result = document.getElementById("result");

        // If geolocation is available, try to get the visitor's position
        if(navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
            result.innerHTML = "Getting the position information...";
        } else {
            alert("Sorry, your browser does not support HTML5 geolocation.");
        }
    };

    // Define callback function for successful attempt
    function successCallback(position) {
        result.innerHTML = "Your current position is (" + "Latitude: " + position.coords.latitude + ", " + "Longitude: " + position.coords.longitude + ")";
    }

    // Define callback function for failed attempt
    function errorCallback(error) {
        if(error.code == 1) {
            result.innerHTML = "You've decided not to share your position, but it's OK. We won't ask you again.";
        } else if(error.code == 2) {
            result.innerHTML = "The network is down or the positioning service can't be reached.";
        } else if(error.code == 3) {
            result.innerHTML = "The attempt timed out before it could get the location data.";
        } else {
            result.innerHTML = "Geolocation failed due to unknown error.";
        }
    }
</script>
</head>
<body>
    <div id="result">
        <!--Position information will be inserted here-->
    </div>
    <button type="button" onclick="showPosition();">Show Position</button>
</body>
</html>
```

HTML5 Geolocation

- There are many interesting functions to explore with geolocation data, as showing the user's location on Google Maps. Based on the latitude and longitude data retrieved through the HTML5 geolocation feature, this example shows the reader's current location. This simply shows a static image showing user's location, although an interactive Google maps with dragging, zoom in/out and other features, as this example shows.

- All the aforementioned examples have been based on the getCurrentPosition() method. Nevertheless, the geolocation function has another technique –watchPosition() that allows to track the visitor's movement by returning the updated position as the location changes. watchPosition() has the same input parameters as getCurrentPosition(). Yet, watchPosition() may activate the success function multiple times — when it gets the location for the first time, and again, every time it spots a new position, as this example suggests.

HTML5 Drag and Drop

- It is a common procedure in the online daily routine **to drag and drop an element to another location in a website**.
- The HTML5 Drag and Drop feature allows to do so, and any element can be dragged and dropped.
- This [example]() sets a simple drag and drop example learners may try to get more familiar with this concept. Even though the code seems difficult to understand, it is quite simple and logic:

HTML5 Drag and Drop

- Firstly, to make an element draggable, the draggable attribute must be true:

```
<img draggable="true">
```

- Then, it should be specified what should happen once the element is dragged. In the example given above, the ondragstart attribute calls a function (drag (event)) that specifies what data will be dragged. The dataTransfer.setData() process sets the data type and the value of the dragged data:

```
Functiondrag(ev){ev.dataTransfer.setData("text",ev.target.id);}
```

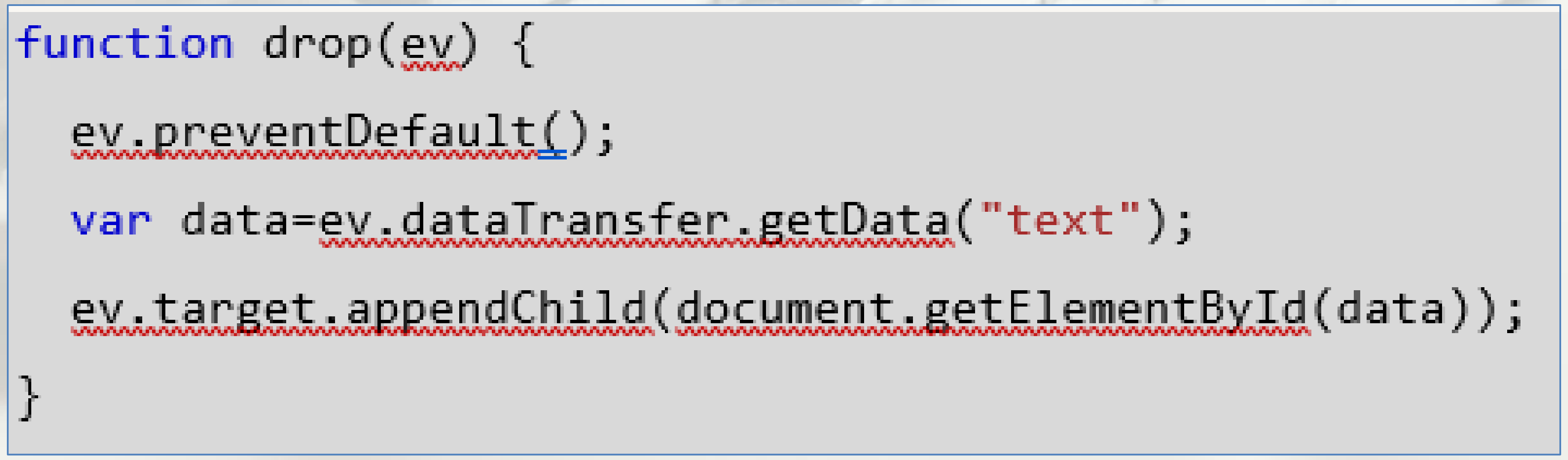
- In this example, the data type is “text”, being the value the id of the draggable element (“drag1”).

HTML5 Drag and Drop

- The ondragover event stipulates where the dragged data can be dropped. By default, data/elements are unable to be dropped in other elements. To permit a drop, the web designer should prevent the default handling of the element, by calling the event.preventDefault() technique for the ondragover event:

```
event.preventDefault()
```

- As soon as the dragged data is dropped, a drop event happens. In the example given previously, the ondrop attribute calls a function, drop(event):



```
function drop(ev) {
  ev.preventDefault();
  var data=ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
```

HTML5 Drag and Drop

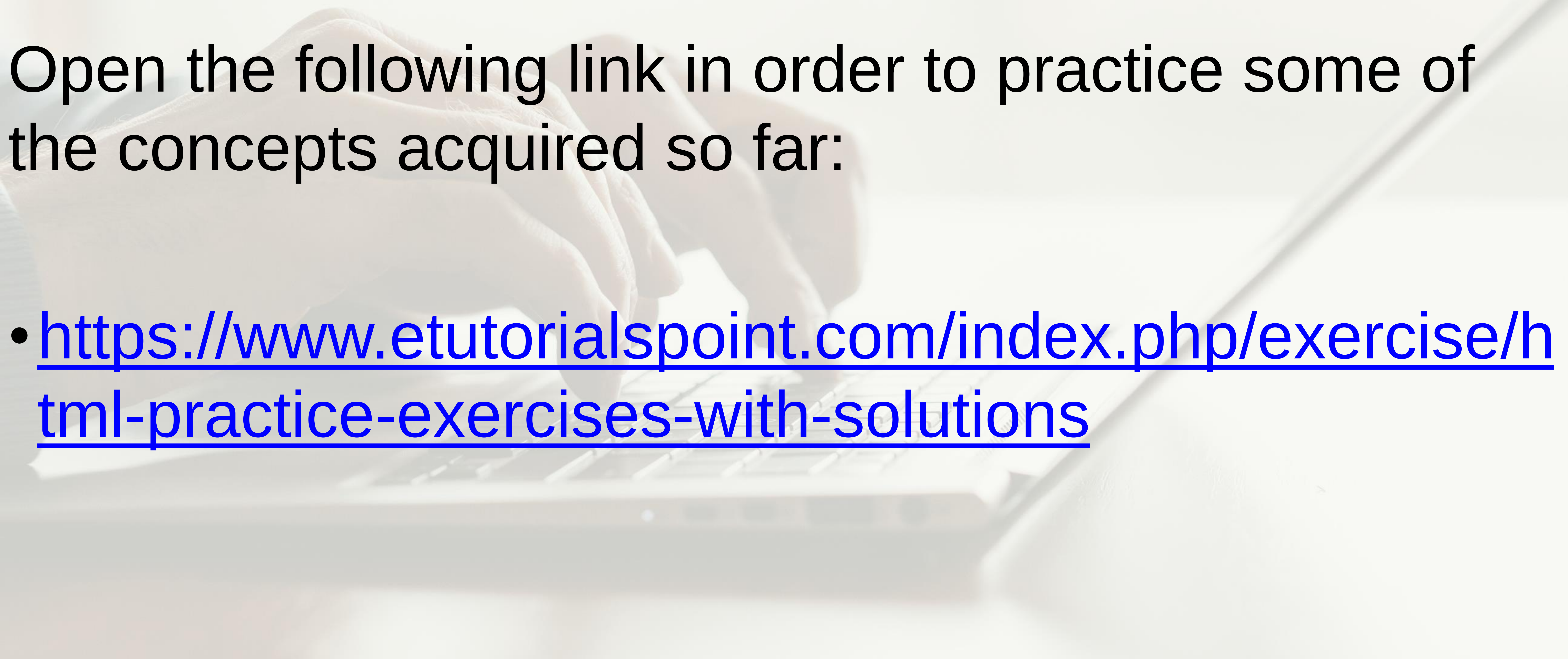
✓Call preventDefault() to prevent the browser default handling of the data (default is open as a link on drop);

✓The programmer gets the dragged data with the dataTransfer.getData() technique. This technique will return any data that was set to the same type in the setData() technique;

✓The dragged data is the id of the dragged element ("drag1")

✓The programmer should also attach the dragged element into the drop element.

LET’S PRACTICE!



Open the following link in order to practice some of the concepts acquired so far:

- https://www.etutorialspoint.com/index.php/exercise/html-practice-exercises-with-solutions

THANK YOU!

Co-funded by the
Erasmus+ Programme
of the European Union