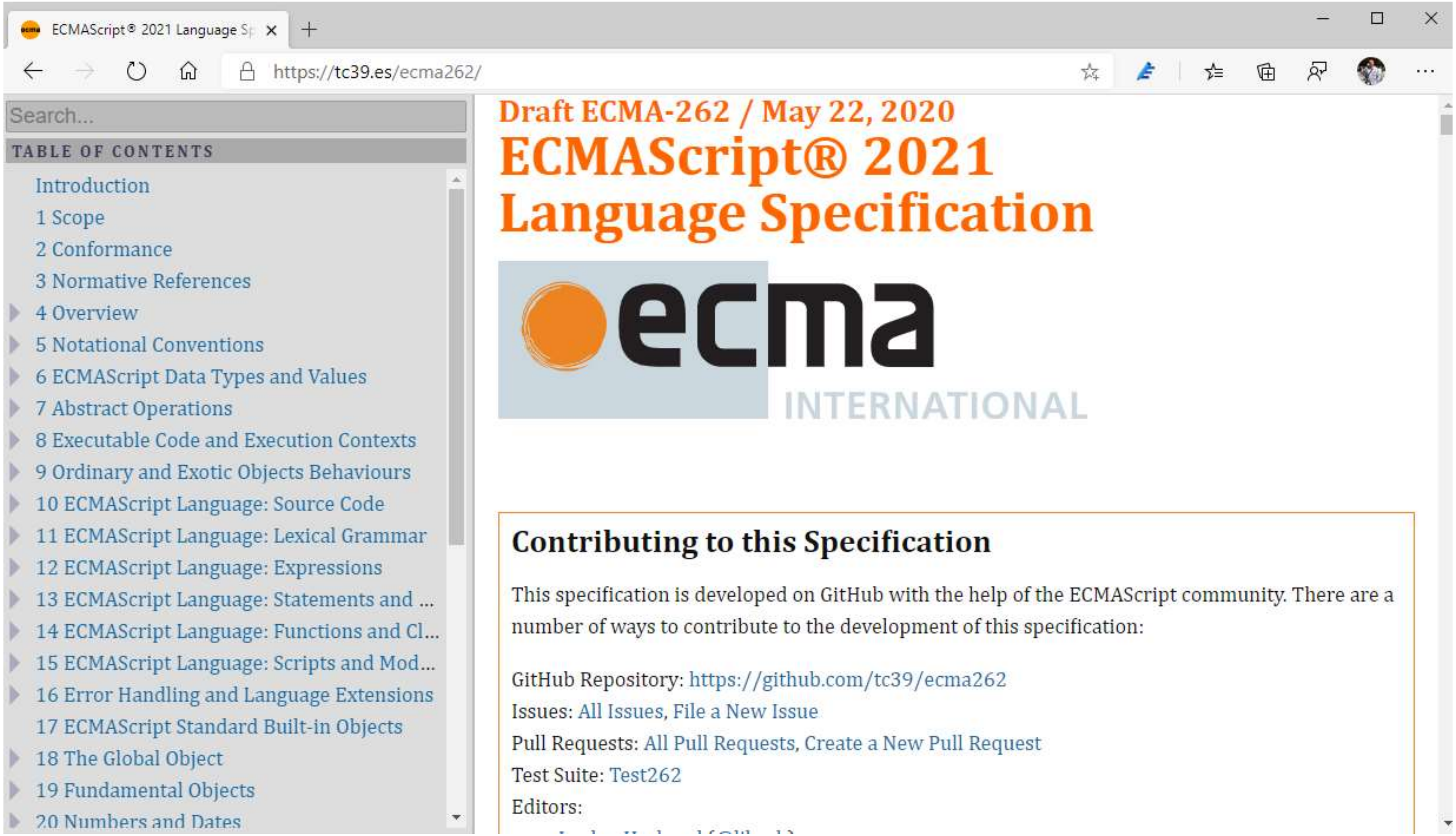
INFO 151
Web Systems and Services

JavaScript ES6

JavaScript ES6 Language Specification



ECMAScript® 2021 Language Sp

https://tc39.es/ecma262/

Search...

TABLE OF CONTENTS

Draft ECMA-262 / May 22, 2020

ECMAScript® 2021 Language Specification

ecma INTERNATIONAL

Contributing to this Specification

This specification is developed on GitHub with the help of the ECMAScript community. There are a number of ways to contribute to the development of this specification:

GitHub Repository: https://github.com/tc39/ecma262
Issues: All Issues, File a New Issue
Pull Requests: All Pull Requests, Create a New Pull Request
Test Suite: Test262
Editors:

JavaScript ES6 Changes

- The latest edition of the JavaScript standard (at the time of writing) is ECMASctipt 6 (known as ES6)
- Many new features are introduced including JavaScript:
  - **`let`**
  - **`symbol`**
  - **`const`**
  - *Arrow* Functions
  - *Classes*
  - Default *parameter* values
  - *array.find()*
  - *array.findIndex()*
  - *exponentiation* (**) (EcmaScript 2016)

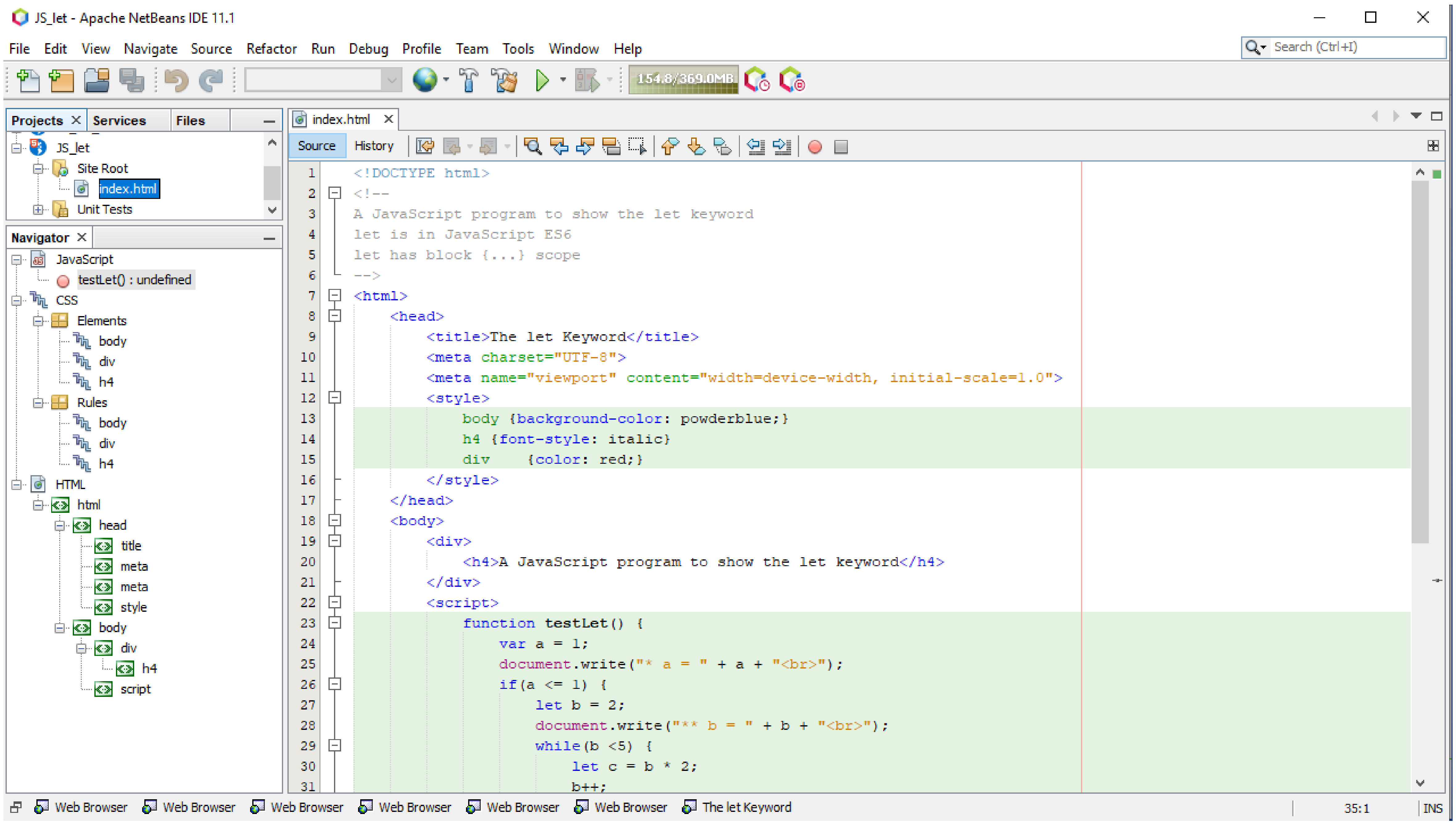
Overview

- In this brief overview of the ES6 version of the JavaScript
- We will introduced the following features and approaches to coding the JavaScript to try to address browser incompatibility (ES5 *vs* ES6) and show worked examples for:
  - **`const`**
  - **`symbol`**
  - **`transpiling`**
  - **`Polyfilling`**

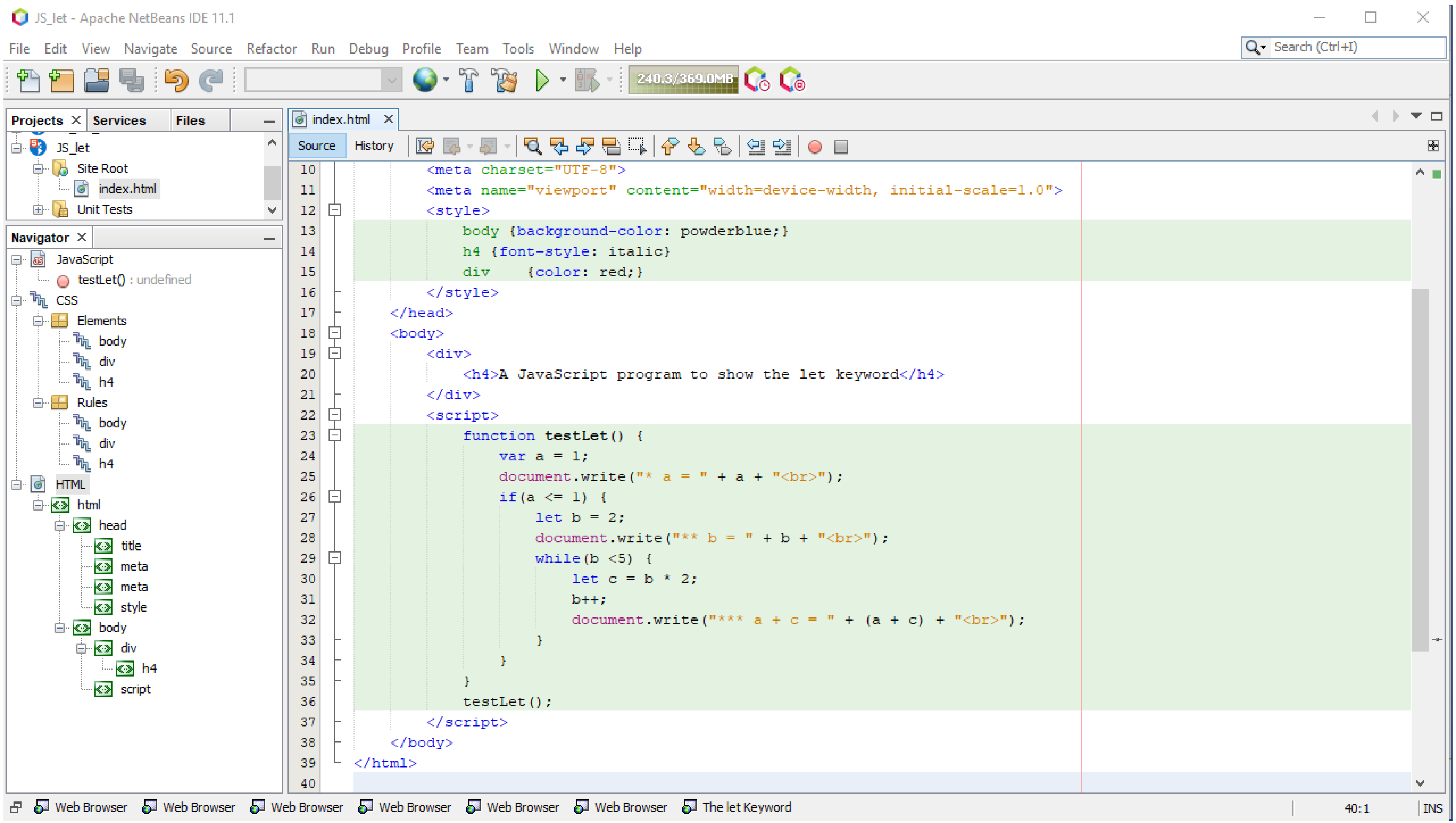
`let`

`let` keyword

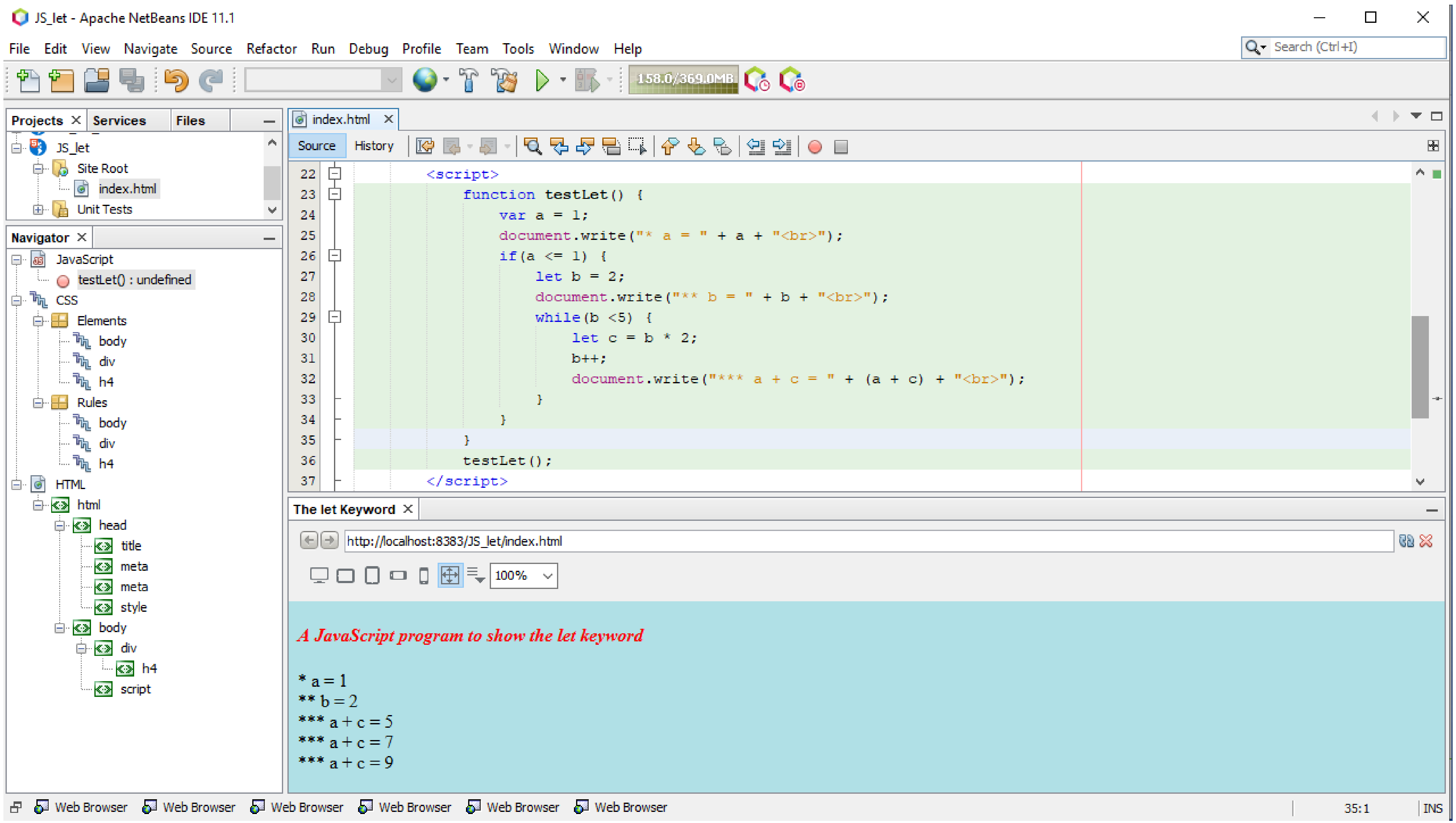
- In ES5 a variable id declared using the `var` keyword
- In ES6 variables can be declared with block `{…}` scope using the `let` keyword
- The following slides show the use of `let` in an `if` statement and `while` loop where variable:
  - `a` has `function` scope and lifetime
  - `b` has `if` statement scope and lifetime
  - `c` has `while` scope and lifetime



```
<!DOCTYPE html>
<!--
A JavaScript program to show the let keyword
let is in JavaScript ES6
let has block {...} scope
-->
<html>
    <head>
        <title>The let Keyword</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            body {background-color: powderblue;}
            h4 {font-style: italic}
            div    {color: red;}
        </style>
    </head>
    <body>
        <div>
            <h4>A JavaScript program to show the let keyword</h4>
        </div>
        <script>
            function testLet() {
                var a = 1;
                document.write("* a = " + a + "<br>");
                if(a <= 1) {
                    let b = 2;
                    document.write("** b = " + b + "<br>");
                    while(b <5) {
                        let c = b * 2;
                        b++;
```



```
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <style>
            body {background-color: powderblue;}
            h4 {font-style: italic}
            div    {color: red;}
        </style>
    </head>
    <body>
        <div>
            <h4>A JavaScript program to show the let keyword</h4>
        </div>
        <script>
            function testLet() {
                var a = 1;
                document.write("* a = " + a + "<br>");
                if(a <= 1) {
                    let b = 2;
                    document.write("** b = " + b + "<br>");
                    while(b <5) {
                        let c = b * 2;
                        b++;
                        document.write("*** a + c = " + (a + c) + "<br>");
                    }
                }
            }
            testLet();
        </script>
    </body>
</html>
```



```
    <script>
        function testLet() {
            var a = 1;
            document.write("* a = " + a + "<br>");
            if(a <= 1) {
                let b = 2;
                document.write("** b = " + b + "<br>");
                while(b <5) {
                    let c = b * 2;
                    b++;
                    document.write("*** a + c = " + (a + c) + "<br>");
                }
            }
        }
        testLet();
    </script>
```

A JavaScript program to show the let keyword

```
* a = 1
** b = 2
*** a + c = 5
*** a + c = 7
*** a + c = 9
```

`const`

GLOBAL Variables and Constants

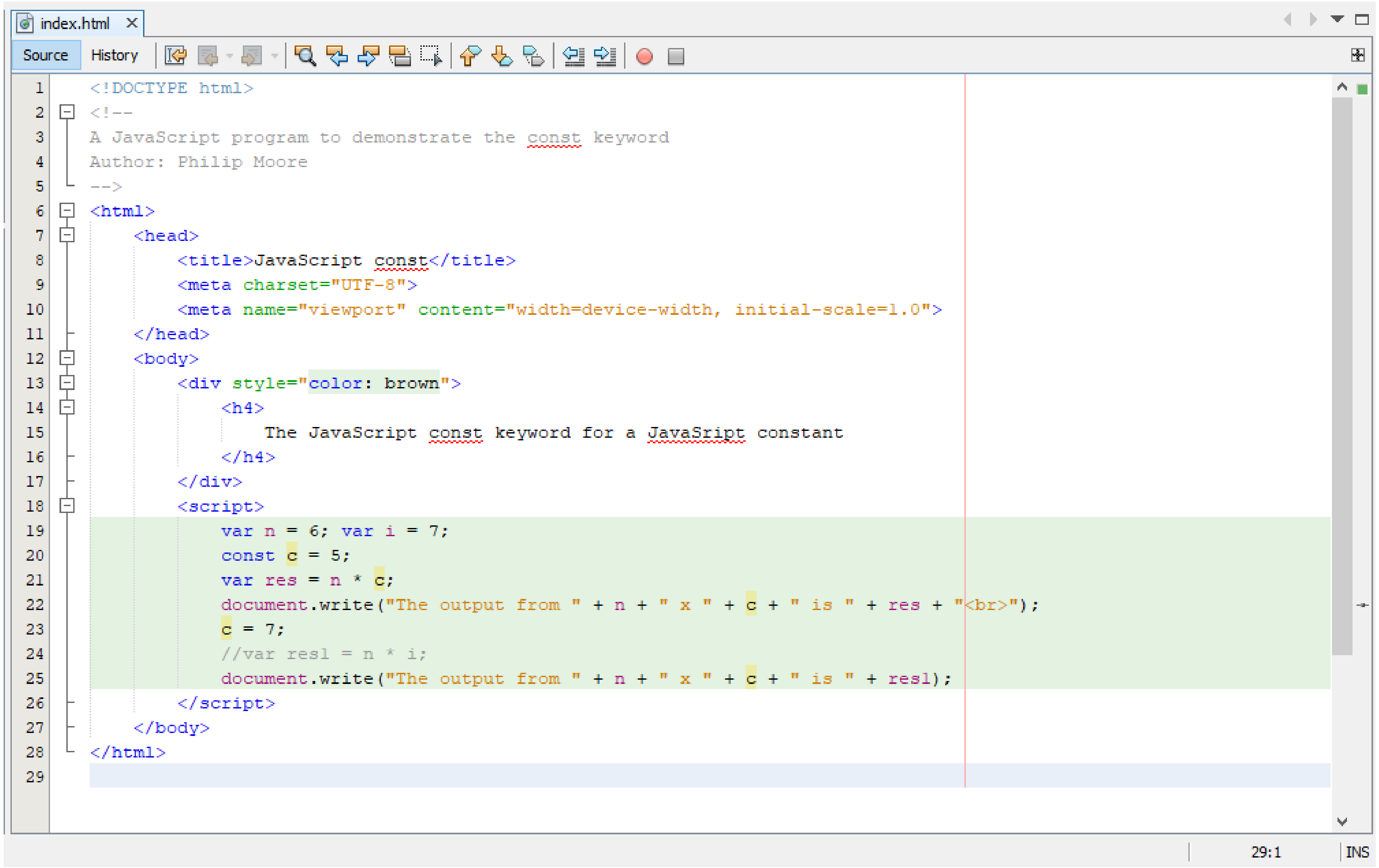
- As we have seen JavaScript global variables are part of a *global object*
- A useful approach to JavaScript programming (indeed all programming) is to create a constant for values used many times
  - In JavaScript the convention is to CAPITALISE constants
    - For example: a tax rate may be named TAX_RATE
  - However, the variable TAX_RATE is still a GLOBAL variable which can be changes anywhere in the program

GLOBAL Variables and Constants

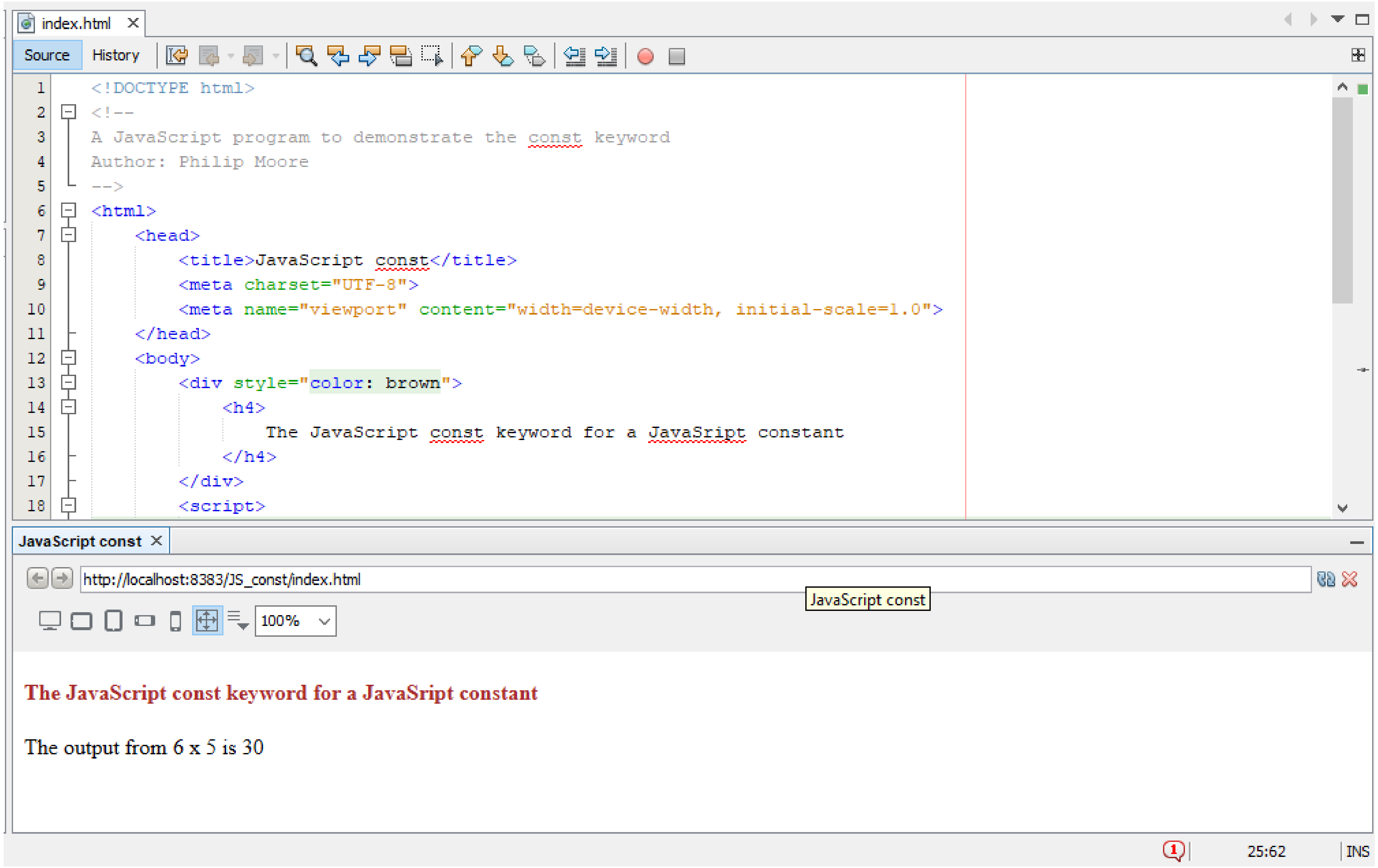
- Details of the ES6 standard can be found at the following link:
  - https://tc39.es/ecma262/
  - My interest here is in the new **`const`** keyword which can define a variable constant value as follows:

```
    Const tax rate = 0.20  //starting with es6
```

- Following the first declaration the program would reject any changes to the constant
- When run in *strict mode* the program run would stop and an error will be reported



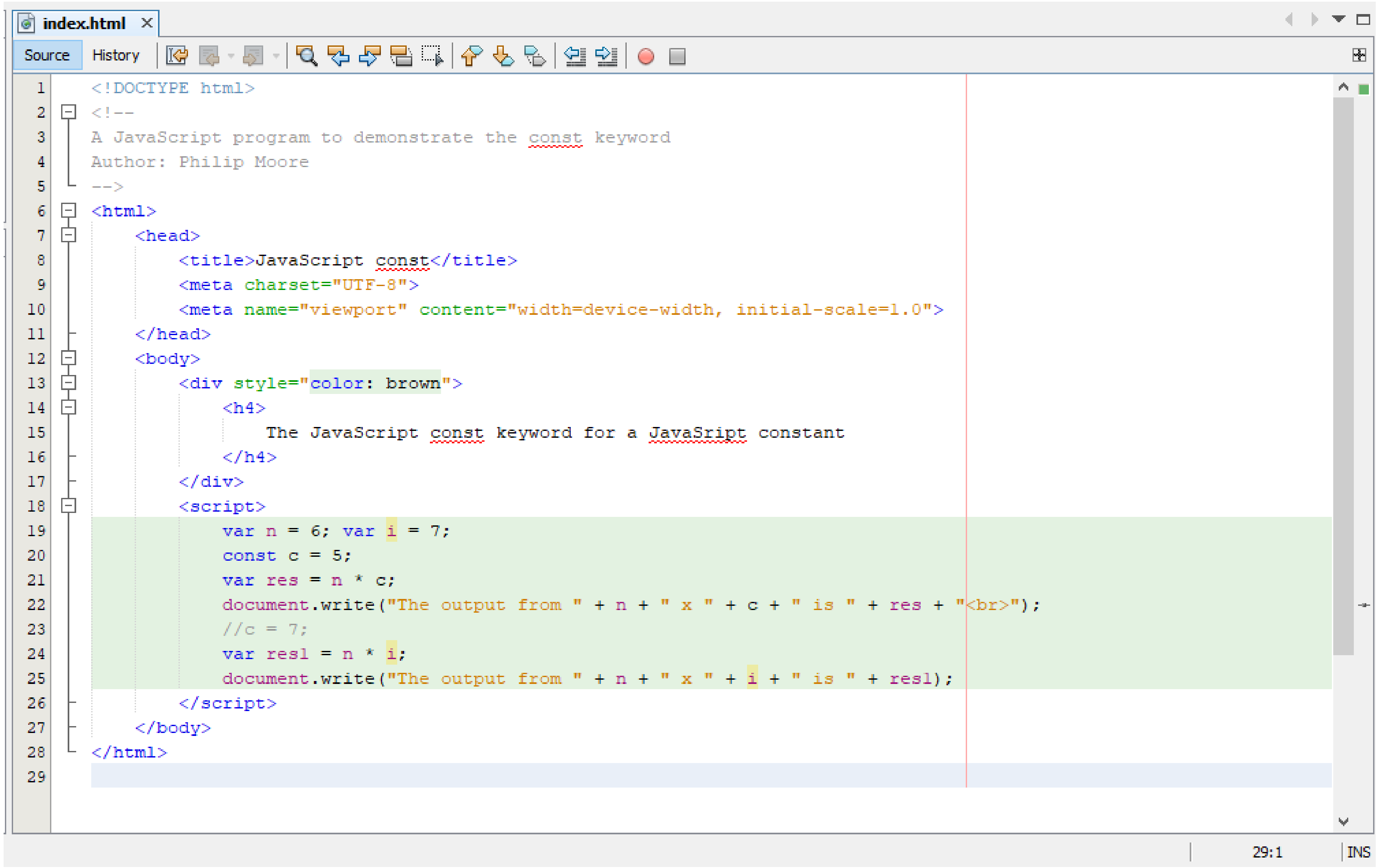
```
<!DOCTYPE html>
<!--
A JavaScript program to demonstrate the const keyword
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript const</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: brown">
            <h4>
                The JavaScript const keyword for a JavaSript constant
            </h4>
        </div>
        <script>
            var n = 6; var i = 7;
            const c = 5;
            var res = n * c;
            document.write("The output from " + n + " x " + c + " is " + res + "<br>");
            c = 7;
            //var res1 = n * i;
            document.write("The output from " + n + " x " + c + " is " + res1);
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
A JavaScript program to demonstrate the const keyword
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript const</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: brown">
            <h4>
                The JavaScript const keyword for a JavaSript constant
            </h4>
        </div>
        <script>
```

The JavaScript const keyword for a JavaSript constant

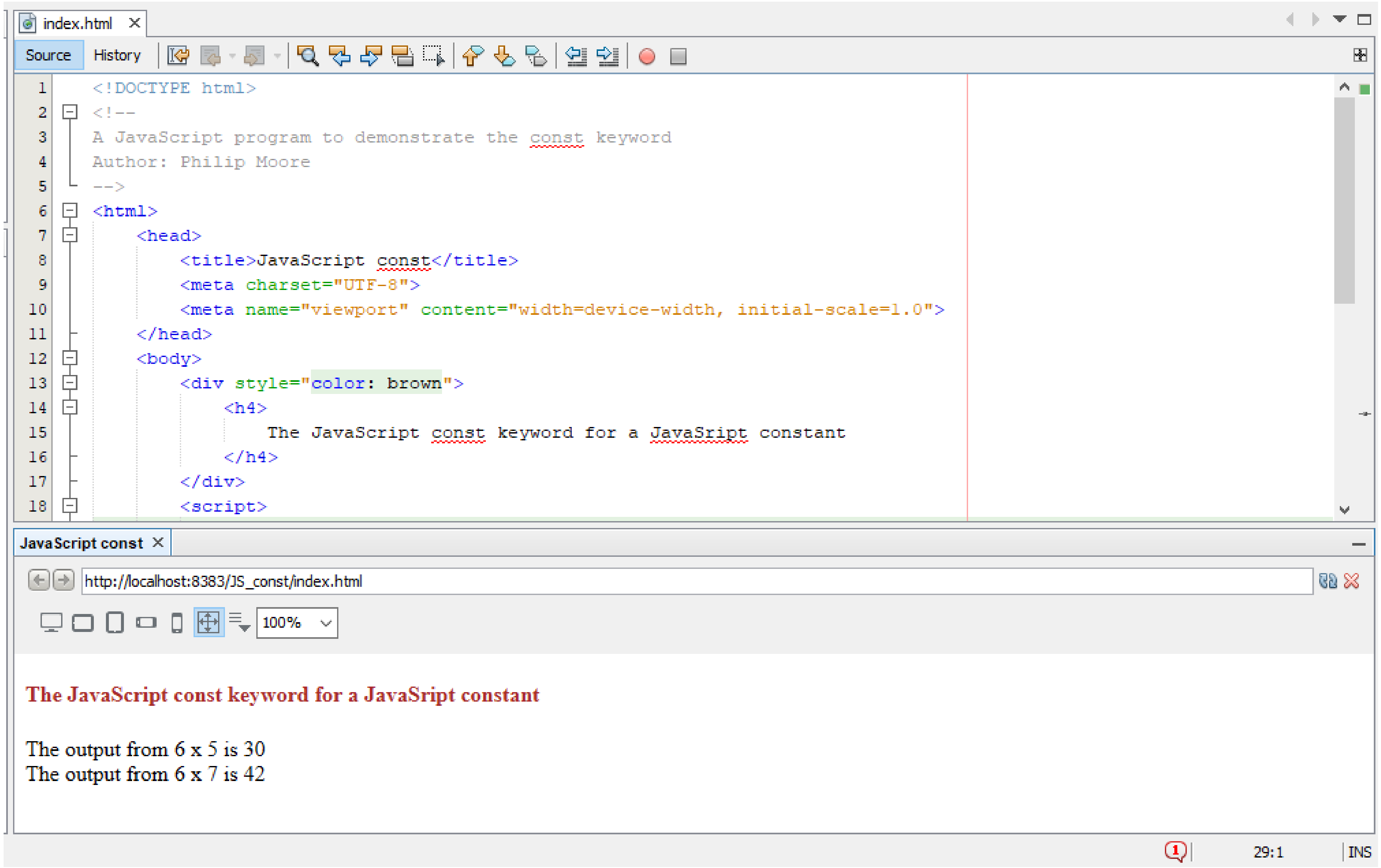
The output from 6 x 5 is 30


index.html

Source History

```
<!DOCTYPE html>
<!--
A JavaScript program to demonstrate the const keyword
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript const</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: brown">
            <h4>
                The JavaScript const keyword for a JavaSript constant
            </h4>
        </div>
        <script>
            var n = 6; var i = 7;
            const c = 5;
            var res = n * c;
            document.write("The output from " + n + " x " + c + " is " + res + "<br>");
            //c = 7;
            var res1 = n * i;
            document.write("The output from " + n + " x " + i + " is " + res1);
        </script>
    </body>
</html>
```

29:1 INS



```
<!DOCTYPE html>
<!--
A JavaScript program to demonstrate the const keyword
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript const</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: brown">
            <h4>
                The JavaScript const keyword for a JavaSript constant
            </h4>
        </div>
        <script>
```

```
The JavaScript const keyword for a JavaSript constant

The output from 6 x 5 is 30
The output from 6 x 7 is 42
```

`symbol`

`Symbol`

- Details of the ES6 standard can be found at the following link:
  - https://tc39.es/ecma262/
- My interest here is in the new `Symbol` feature
  - In JavaScript `Symbol` is a primitive value
  - A value having the data type `Symbol` can be referred to as a Symbol value
  - In a JavaScript runtime environment a symbol value is created by invoking the function `Symbol()`

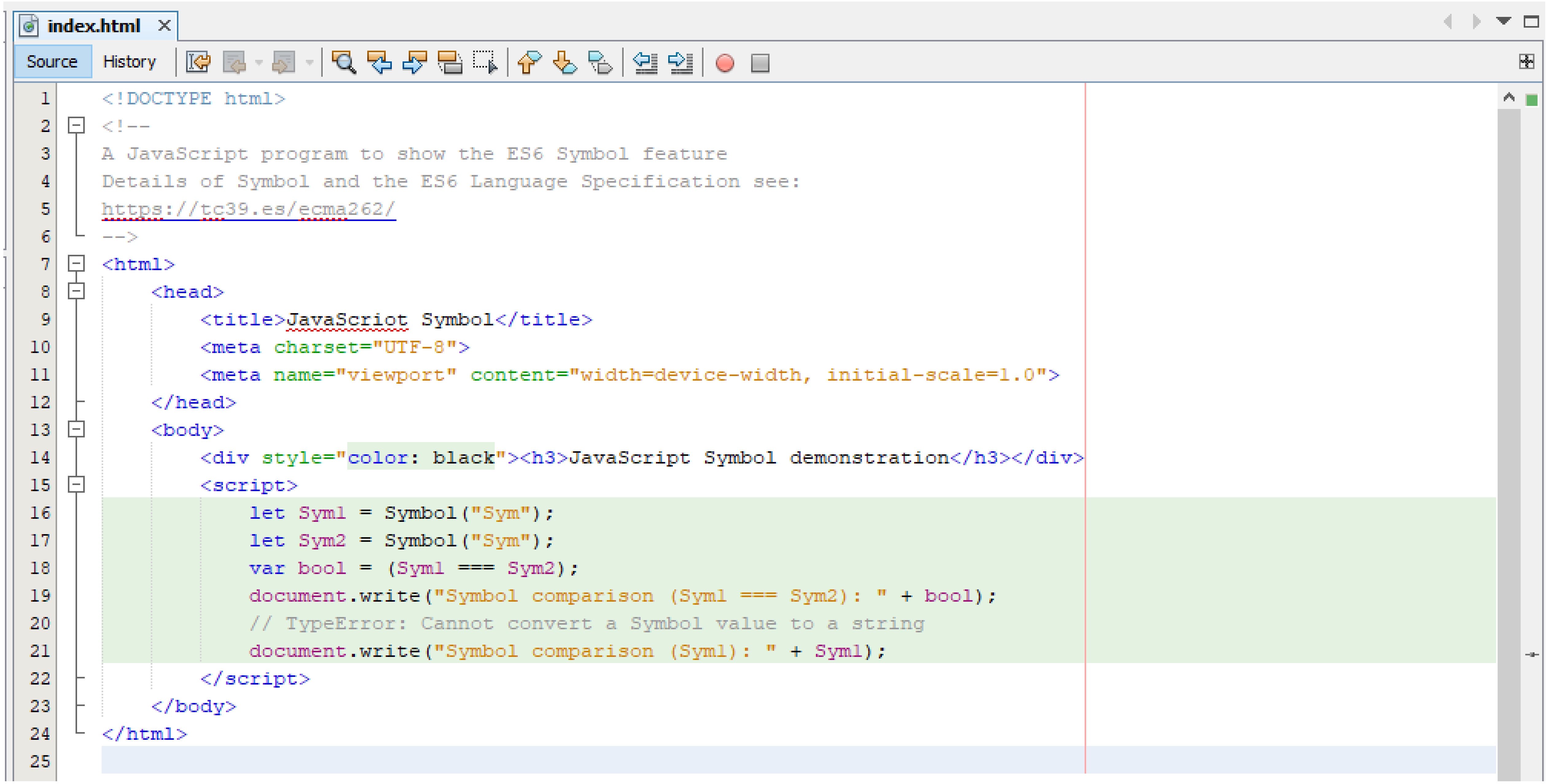
`Symbol`

- `Symbol` type is a new feature in ECMAScript 2015
  - There is no ECMAScript 5 equivalent for Symbol
  - In some programming languages, the symbol data type is referred to as an *atom*
  - `Symbols` do not automatically convert to strings
  - `Symbol` creates an anonymous unique value which represents a unique identifier
  - Examples of well-known symbols are:
    - `Symbol.iterator //for array-like objects`
    - `Symbol.search   //for string-like objects`

Do we really need symbols?

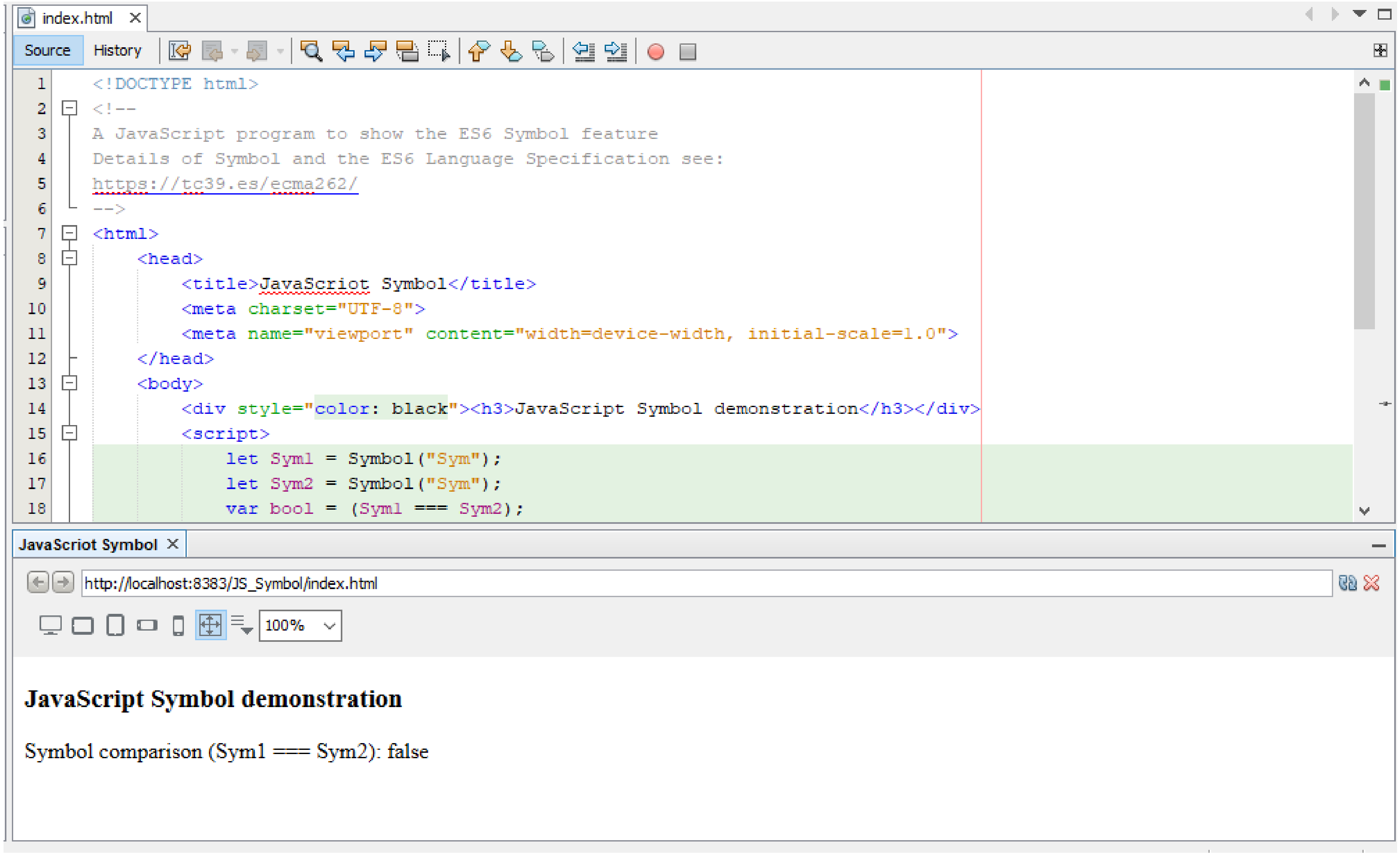
- Use symbols when your requirement is:
  - Enum: To allow you to define constants with semantic names and unique values – for example:

    ```
    const directions = {UP:Symbol('UP'), DOWN:Symbol('DOWN'),
    LEFT:Symbol('LEFT'), RIGHT:Symbol('RIGHT')}
    ```
  - Name Clashes: when you wanted to prevent collisions with keys in objects
  - Privacy: when you don't want your object properties to be enumerable
  - Protocols: To define how an object can be iterated
  - In addition to user-defined symbols JavaScript has some built-in symbols which represent internal language behaviours which were not exposed to developers in ES5 (see: https://developer.mozilla.org/)



```
<!DOCTYPE html>
<!--
A JavaScript program to show the ES6 Symbol feature
Details of Symbol and the ES6 Language Specification see:
https://tc39.es/ecma262/
-->
<html>
    <head>
        <title>JavaScriot Symbol</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: black"><h3>JavaScript Symbol demonstration</h3></div>
        <script>
            let Sym1 = Symbol("Sym");
            let Sym2 = Symbol("Sym");
            var bool = (Sym1 === Sym2);
            document.write("Symbol comparison (Sym1 === Sym2): " + bool);
            // TypeError: Cannot convert a Symbol value to a string
            document.write("Symbol comparison (Sym1): " + Sym1);
        </script>
    </body>
</html>

```



```
<!DOCTYPE html>
<!--
A JavaScript program to show the ES6 Symbol feature
Details of Symbol and the ES6 Language Specification see:
https://tc39.es/ecma262/
-->
<html>
    <head>
        <title>JavaScriot Symbol</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color: black"><h3>JavaScript Symbol demonstration</h3></div>
        <script>
            let Sym1 = Symbol("Sym");
            let Sym2 = Symbol("Sym");
            var bool = (Sym1 === Sym2);
```

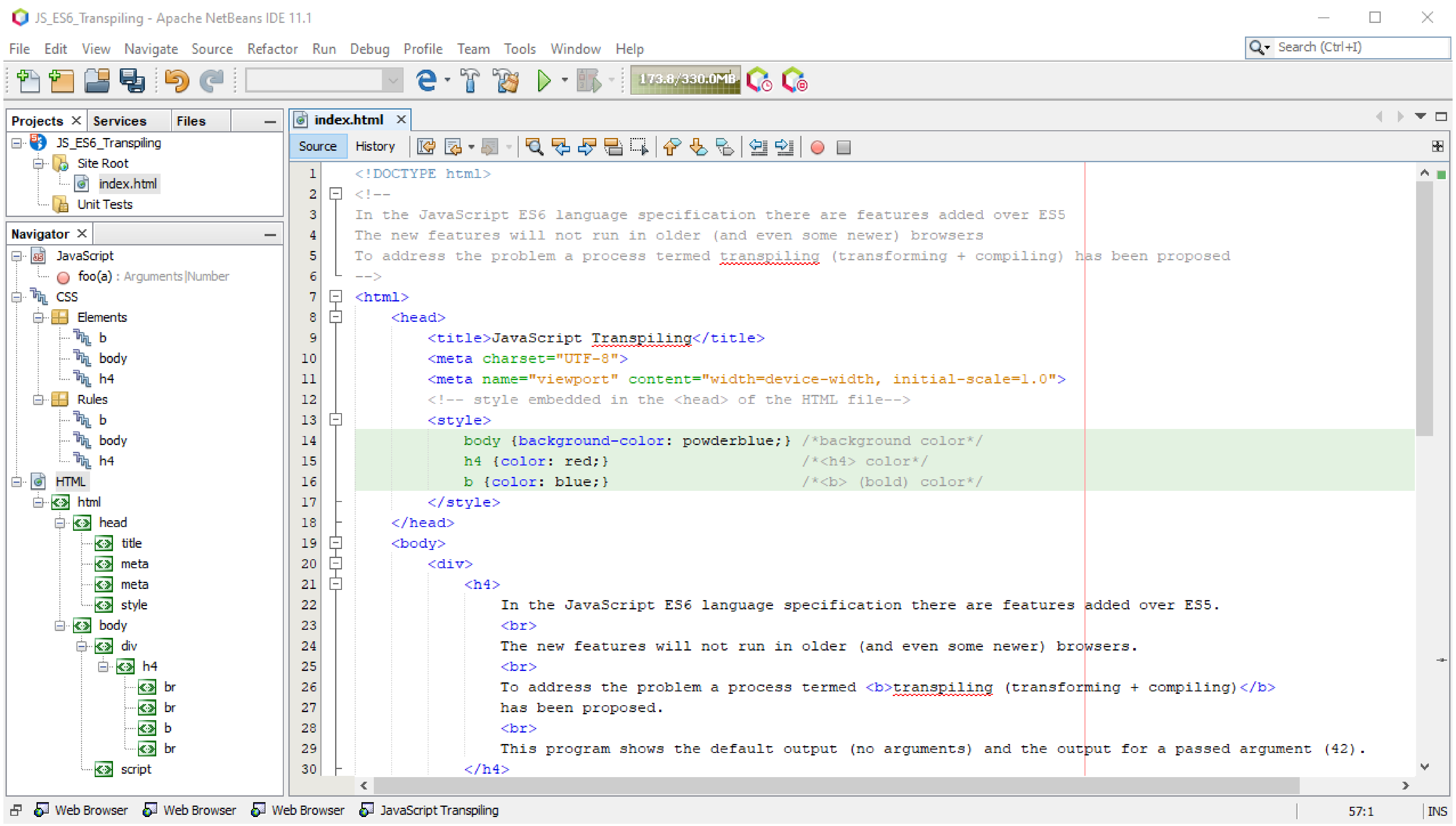
JavaScript Symbol demonstration

Symbol comparison (Sym1 === Sym2): false

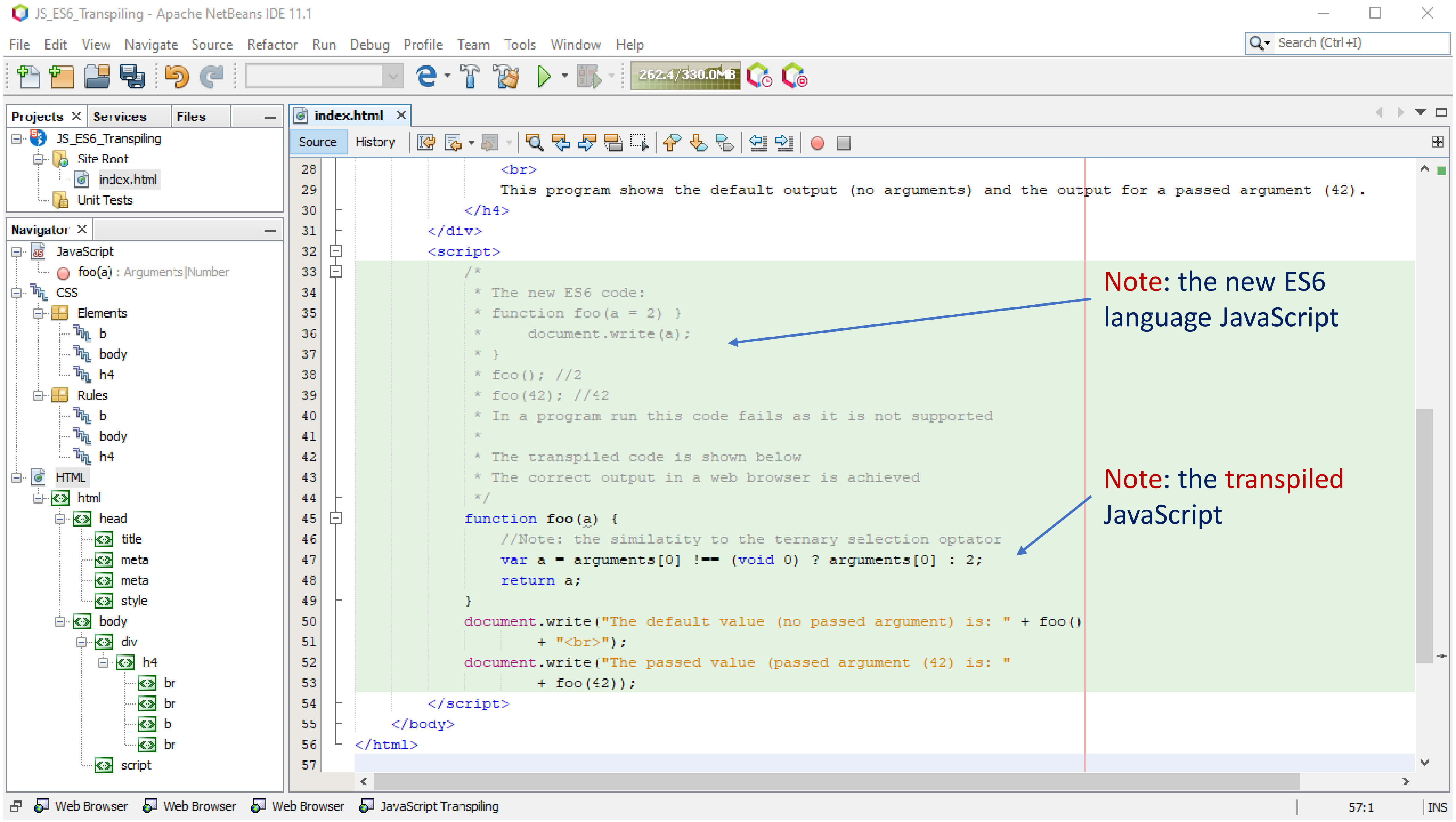
`transpiling`

`Transpiling` JavaScript

- In the JavaScript ES6 language specification there are features added over ES5
- My interest here lies in the issue that: many of the new features will not run in older (and even some newer) browsers
- To address the problem: a process termed **`transpiling`** (from: transforming + compiling) has been proposed
- The process has been conceived: to convert the newer code into the older equivalent code (to run on older and newer browsers)
- The following worked example demonstrates the new (failed) and **`transpiled`** (working) JavaScript code



```html
<!DOCTYPE html>
<!--
In the JavaScript ES6 language specification there are features added over ES5
The new features will not run in older (and even some newer) browsers
To address the problem a process termed transpiling (transforming + compiling) has been proposed
-->
<html>
    <head>
        <title>JavaScript Transpiling</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- style embedded in the <head> of the HTML file-->
        <style>
            body {background-color: powderblue;} /*background color*/
            h4 {color: red;}                     /*<h4> color*/
            b {color: blue;}                     /*<b> (bold) color*/
        </style>
    </head>
    <body>
        <div>
            <h4>
                In the JavaScript ES6 language specification there are features added over ES5.
                <br>
                The new features will not run in older (and even some newer) browsers.
                <br>
                To address the problem a process termed <b>transpiling (transforming + compiling)</b>
                has been proposed.
                <br>
                This program shows the default output (no arguments) and the output for a passed argument (42).
            </h4>
```

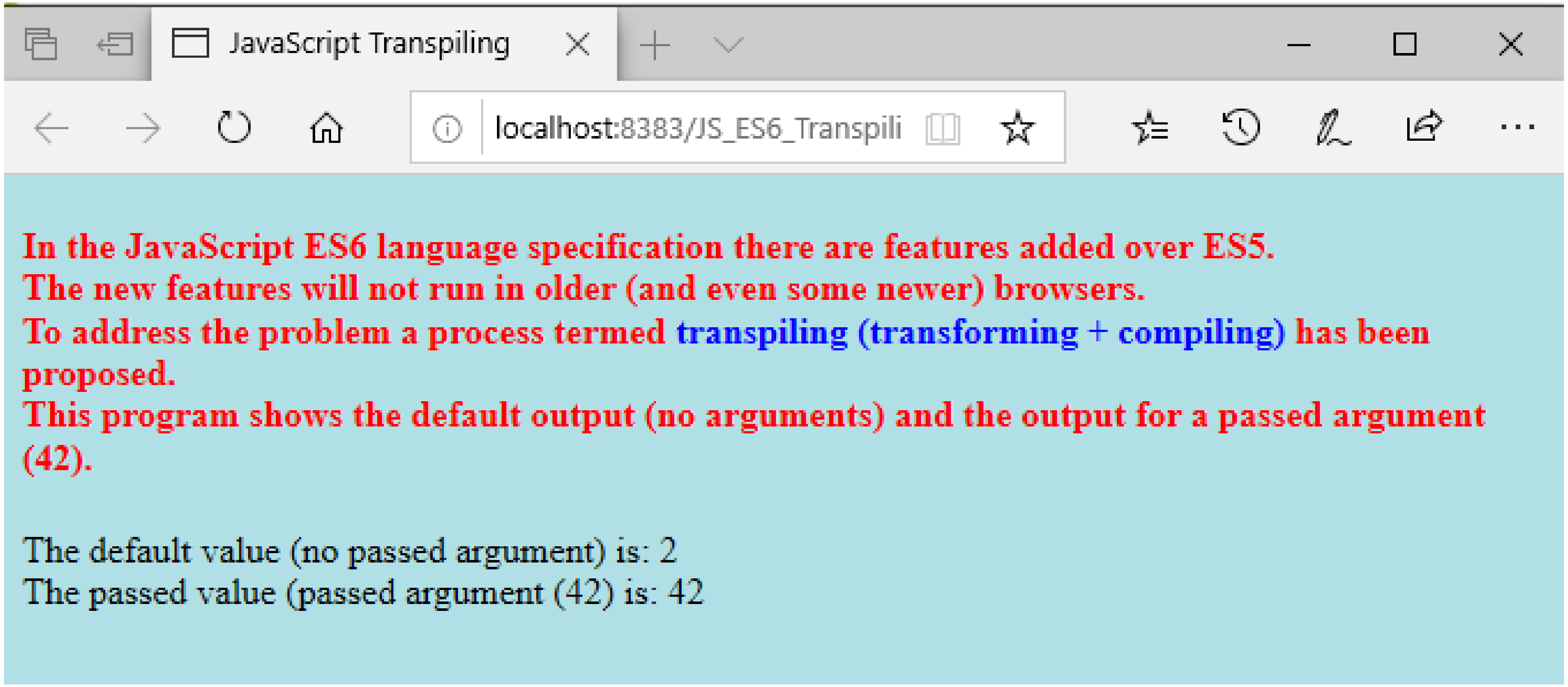


```
                <br>
                This program shows the default output (no arguments) and the output for a passed argument (42).
            </h4>
        </div>
        <script>
            /*
             * The new ES6 code:
             * function foo(a = 2) }
             *     document.write(a);
             * }
             * foo(); //2
             * foo(42); //42
             * In a program run this code fails as it is not supported
             *
             * The transpiled code is shown below
             * The correct output in a web browser is achieved
             */
            function foo(a) {
                //Note: the similatity to the ternary selection optator
                var a = arguments[0] !== (void 0) ? arguments[0] : 2;
                return a;
            }
            document.write("The default value (no passed argument) is: " + foo()
                    + "<br>");
            document.write("The passed value (passed argument (42) is: "
                    + foo(42));
        </script>
    </body>
</html>
```

Note: the new ES6 language JavaScript

Note: the transpiled JavaScript

Output in MS Edge Browser



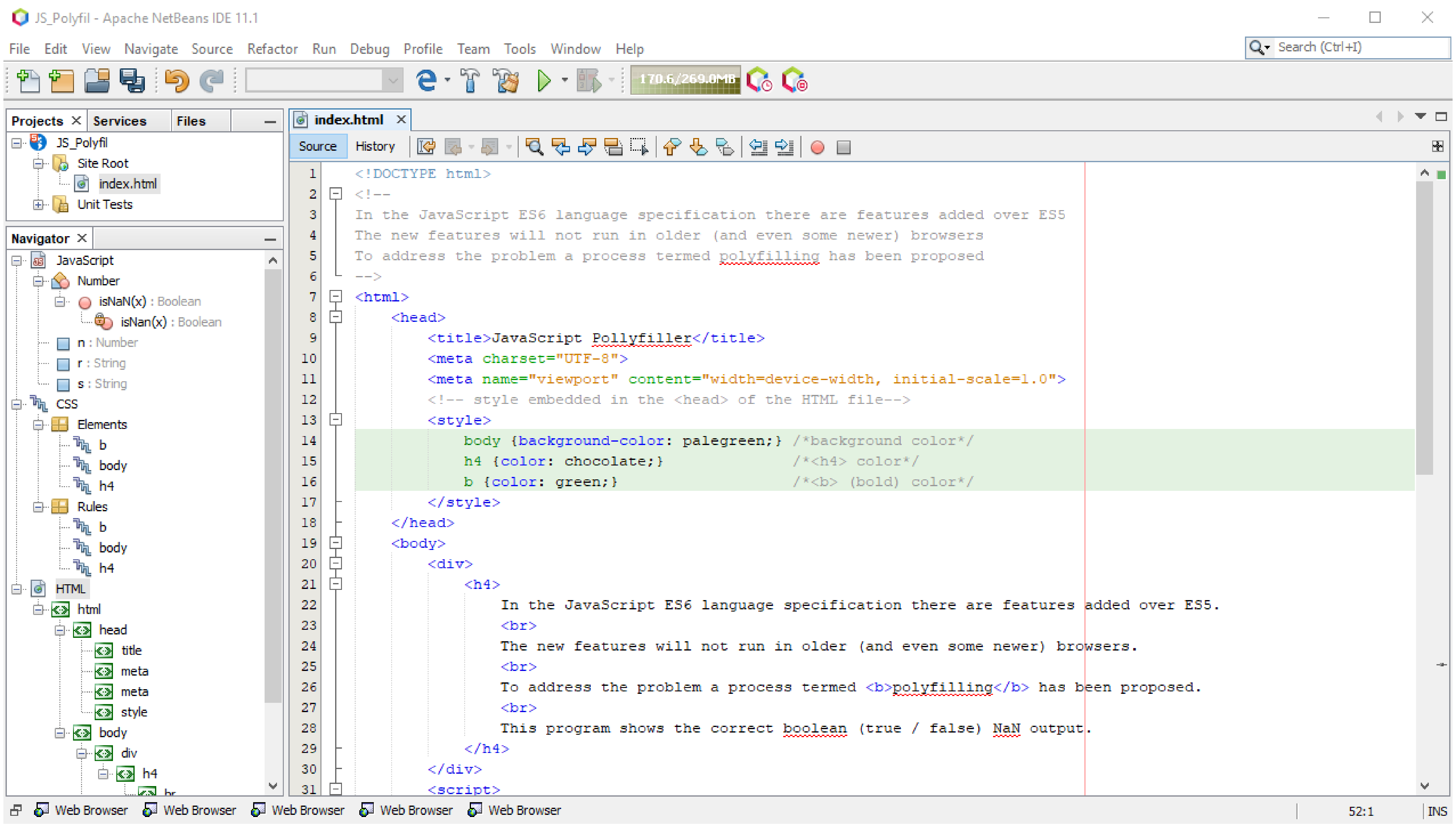
In the JavaScript ES6 language specification there are features added over ES5.
The new features will not run in older (and even some newer) browsers.
To address the problem a process termed transpiling (transforming + compiling) has been proposed.
This program shows the default output (no arguments) and the output for a passed argument (42).

The default value (no passed argument) is: 2
The passed value (passed argument (42) is: 42

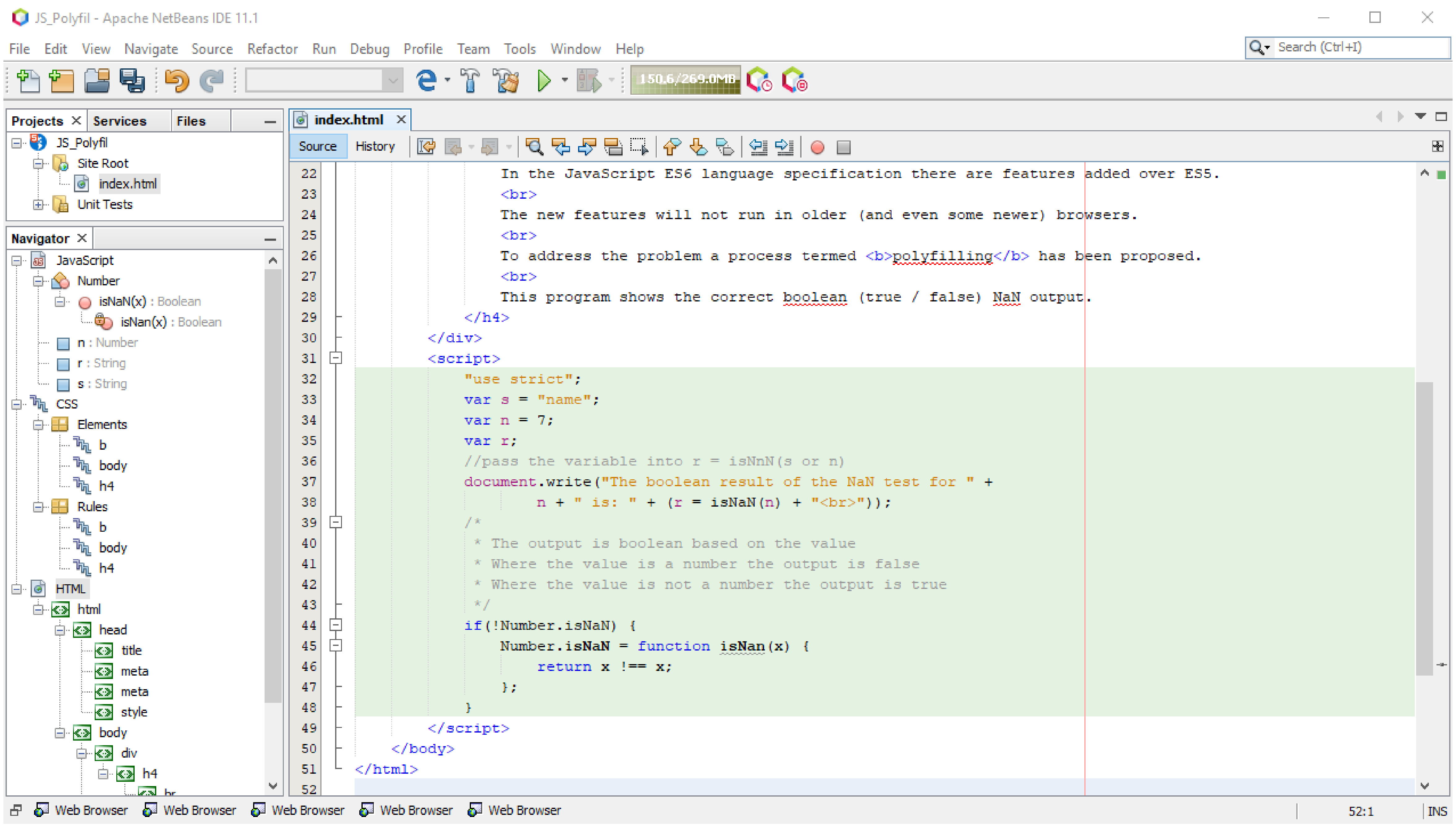
`polyfilling`

`Polyfilling` JavaScript

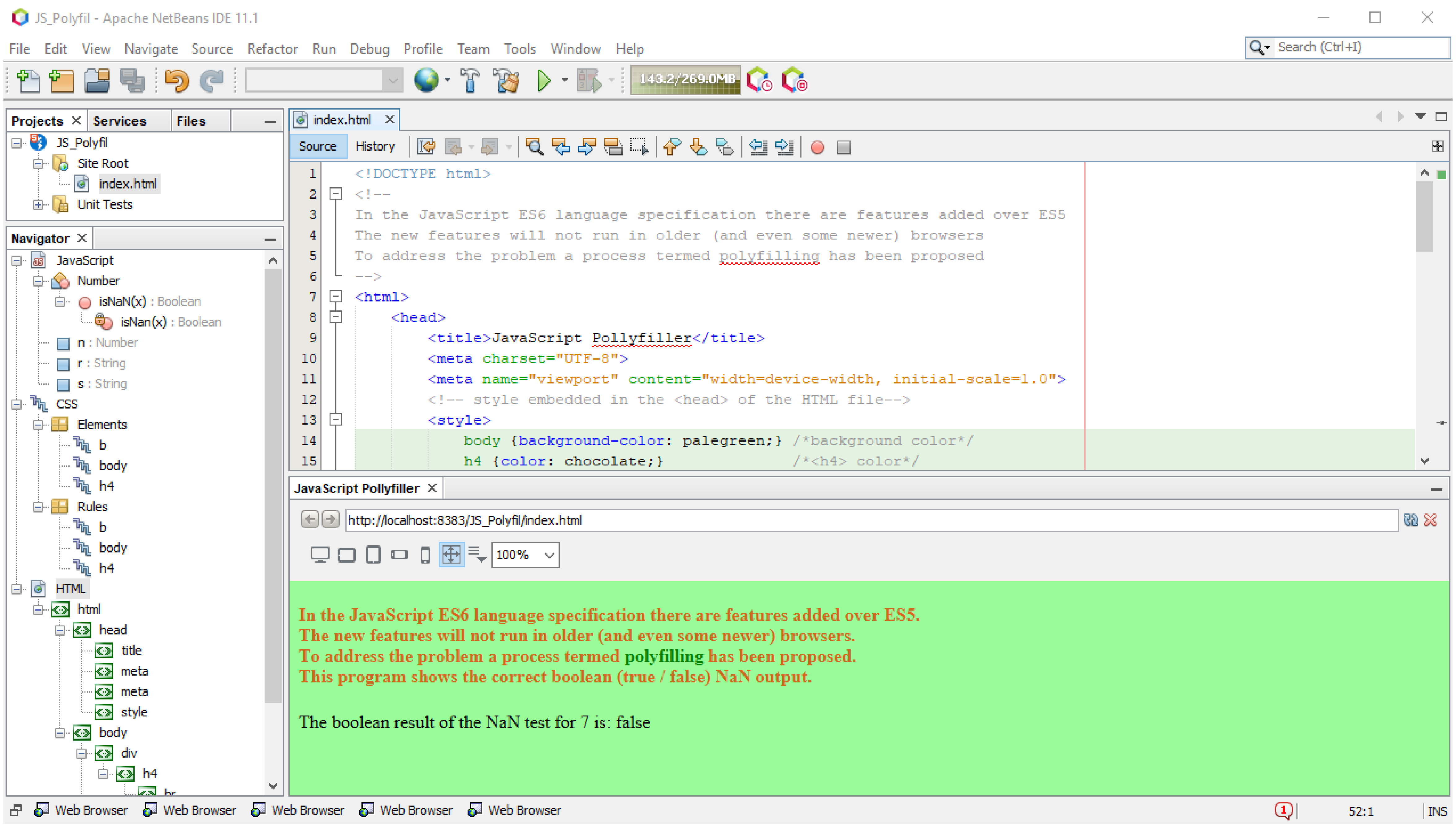
- In the JavaScript ES6 language specification there are features added over ES5
- My interest here lies in the issue that: many of the new features will not run in older (and even some newer) browsers
- To address the problem: a process termed **`polyfilling`** has been proposed
- The process has been conceived: to convert the newer code into the older equivalent code (to run on older and newer browsers)
- The following worked example demonstrates the new **`polyfiller`** JavaScript code with the Boolean output for a NaN test



```
<!DOCTYPE html>
<!--
In the JavaScript ES6 language specification there are features added over ES5
The new features will not run in older (and even some newer) browsers
To address the problem a process termed polyfilling has been proposed
-->
<html>
    <head>
        <title>JavaScript Pollyfiller</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- style embedded in the <head> of the HTML file-->
        <style>
            body {background-color: palegreen;} /*background color*/
            h4 {color: chocolate;}              /*<h4> color*/
            b {color: green;}                   /*<b> (bold) color*/
        </style>
    </head>
    <body>
        <div>
            <h4>
                In the JavaScript ES6 language specification there are features added over ES5.
                <br>
                The new features will not run in older (and even some newer) browsers.
                <br>
                To address the problem a process termed <b>polyfilling</b> has been proposed.
                <br>
                This program shows the correct boolean (true / false) NaN output.
            </h4>
        </div>
        <script>
```



```
        In the JavaScript ES6 language specification there are features added over ES5.
        <br>
        The new features will not run in older (and even some newer) browsers.
        <br>
        To address the problem a process termed <b>polyfilling</b> has been proposed.
        <br>
        This program shows the correct boolean (true / false) NaN output.
      </h4>
    </div>
    <script>
      "use strict";
      var s = "name";
      var n = 7;
      var r;
      //pass the variable into r = isNnN(s or n)
      document.write("The boolean result of the NaN test for " +
          n + " is: " + (r = isNaN(n) + "<br>"));
      /*
       * The output is boolean based on the value
       * Where the value is a number the output is false
       * Where the value is not a number the output is true
       */
      if(!Number.isNaN) {
        Number.isNaN = function isNan(x) {
          return x !== x;
        };
      }
    </script>
  </body>
</html>
```



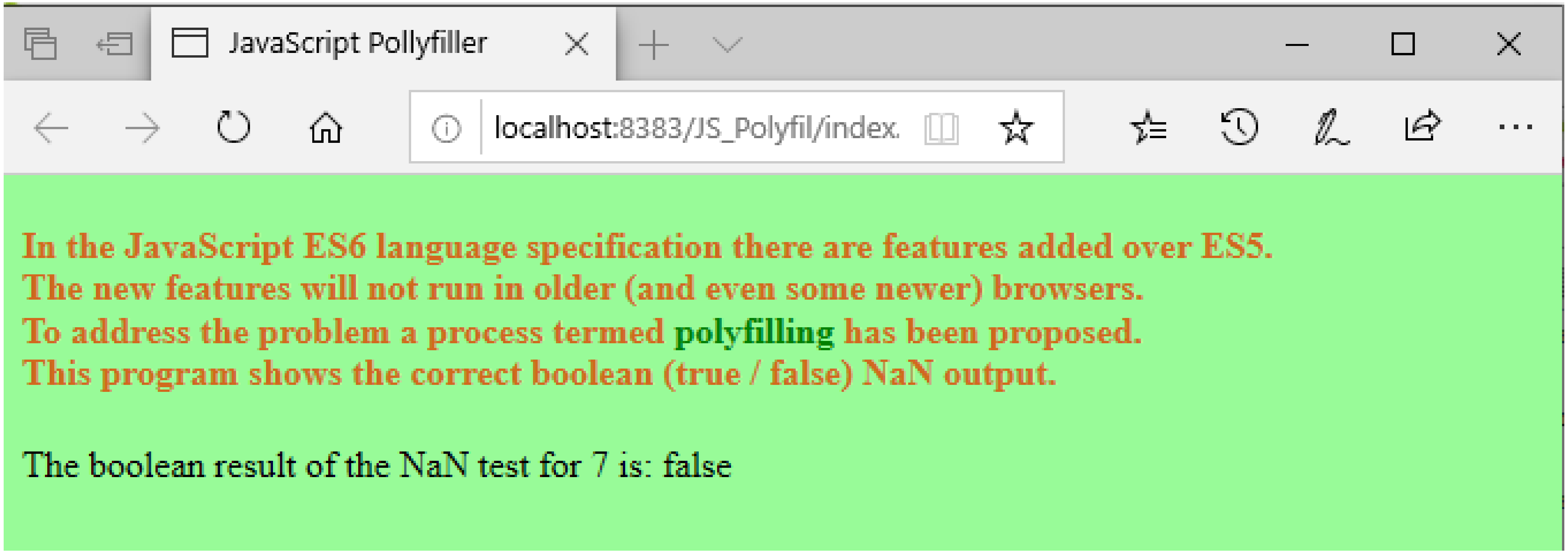
```
<!DOCTYPE html>
<!--
In the JavaScript ES6 language specification there are features added over ES5
The new features will not run in older (and even some newer) browsers
To address the problem a process termed polyfilling has been proposed
-->
<html>
    <head>
        <title>JavaScript Pollyfiller</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- style embedded in the <head> of the HTML file-->
        <style>
            body {background-color: palegreen;} /*background color*/
            h4 {color: chocolate;}              /*<h4> color*/
```

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

In the JavaScript ES6 language specification there are features added over ES5.
The new features will not run in older (and even some newer) browsers.
To address the problem a process termed polyfilling has been proposed.
This program shows the correct boolean (true / false) NaN output.

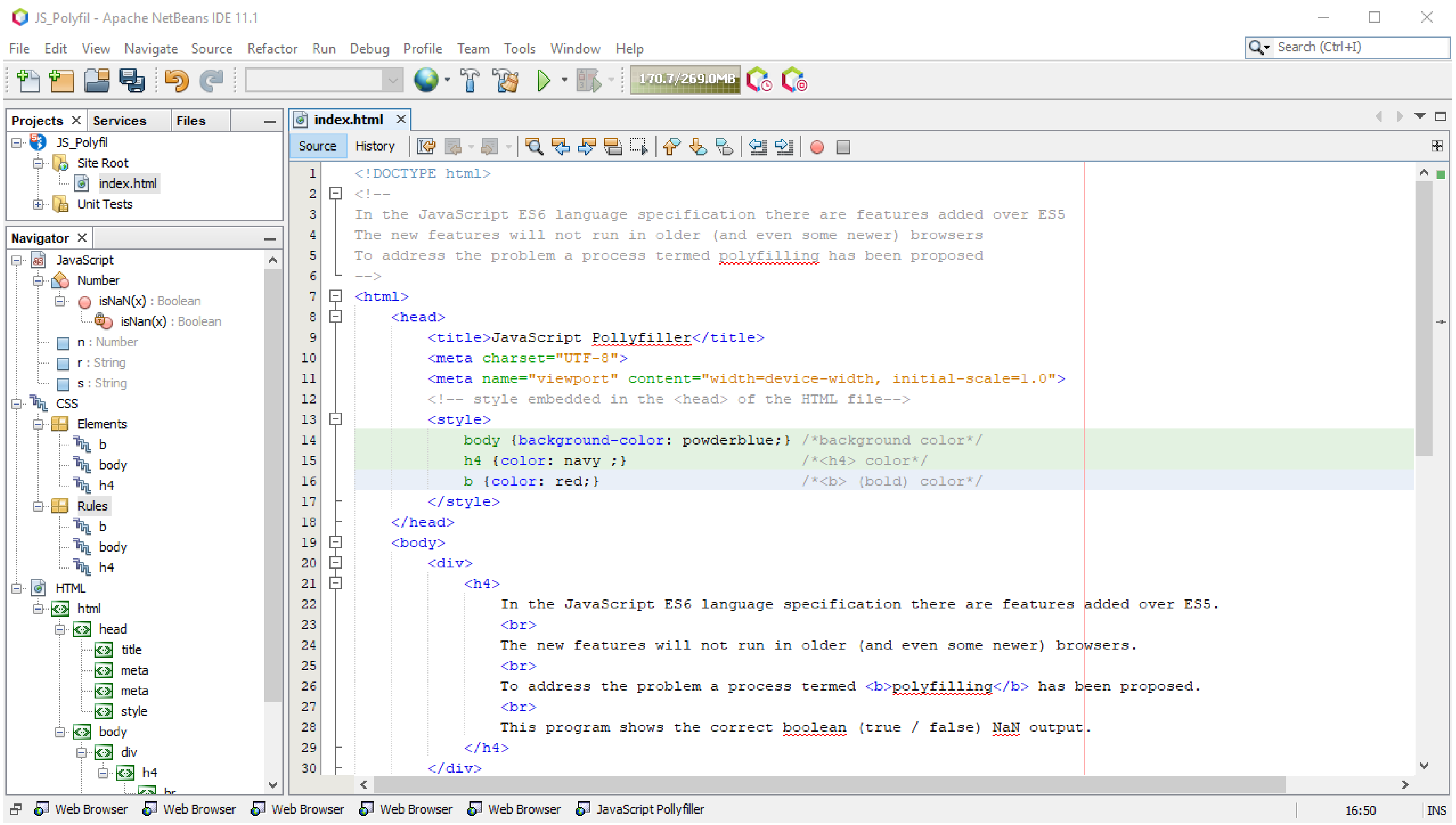
The boolean result of the NaN test for 7 is: false

Output in MS Edge Browser

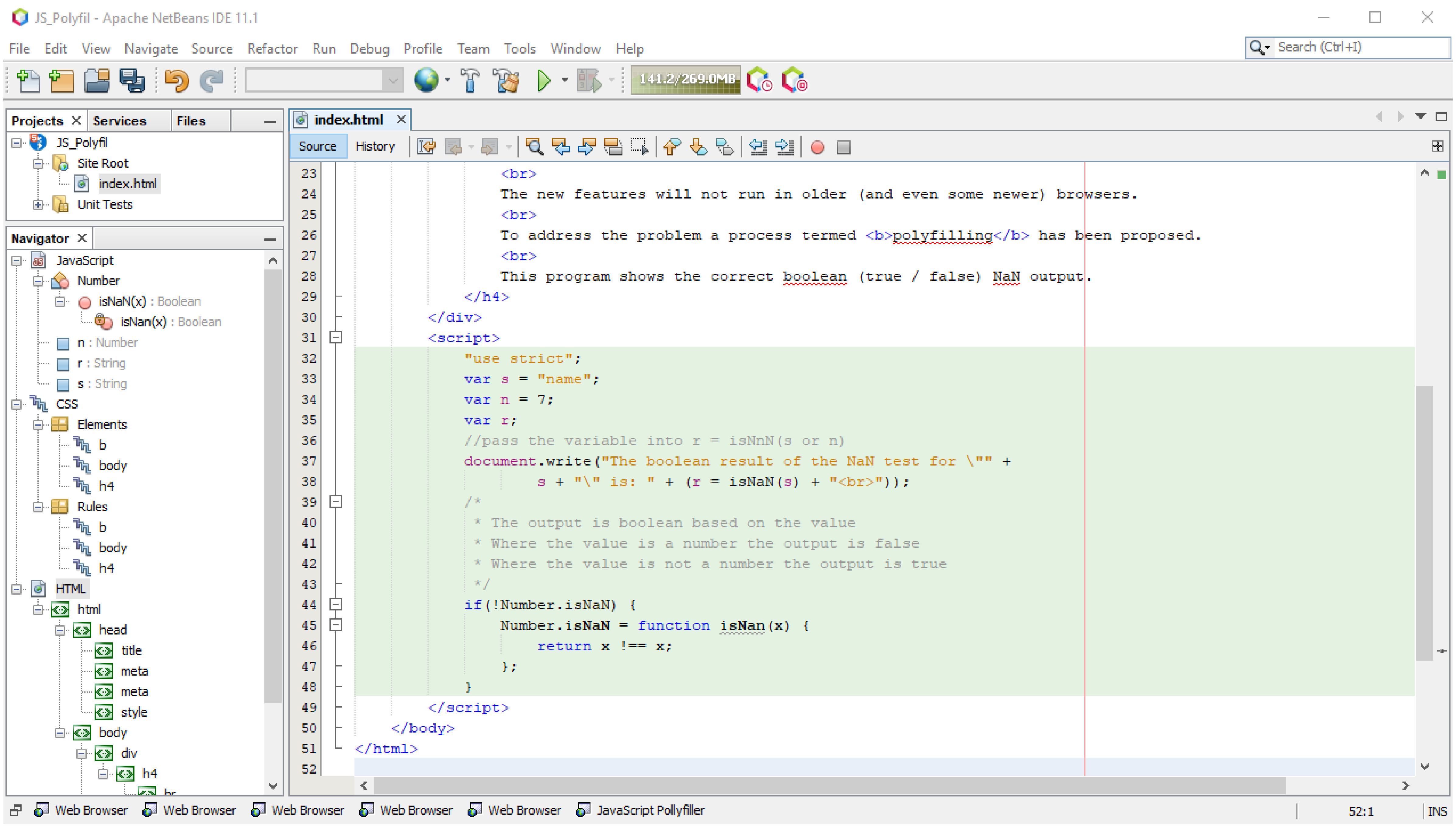


In the JavaScript ES6 language specification there are features added over ES5.
The new features will not run in older (and even some newer) browsers.
To address the problem a process termed polyfilling has been proposed.
This program shows the correct boolean (true / false) NaN output.

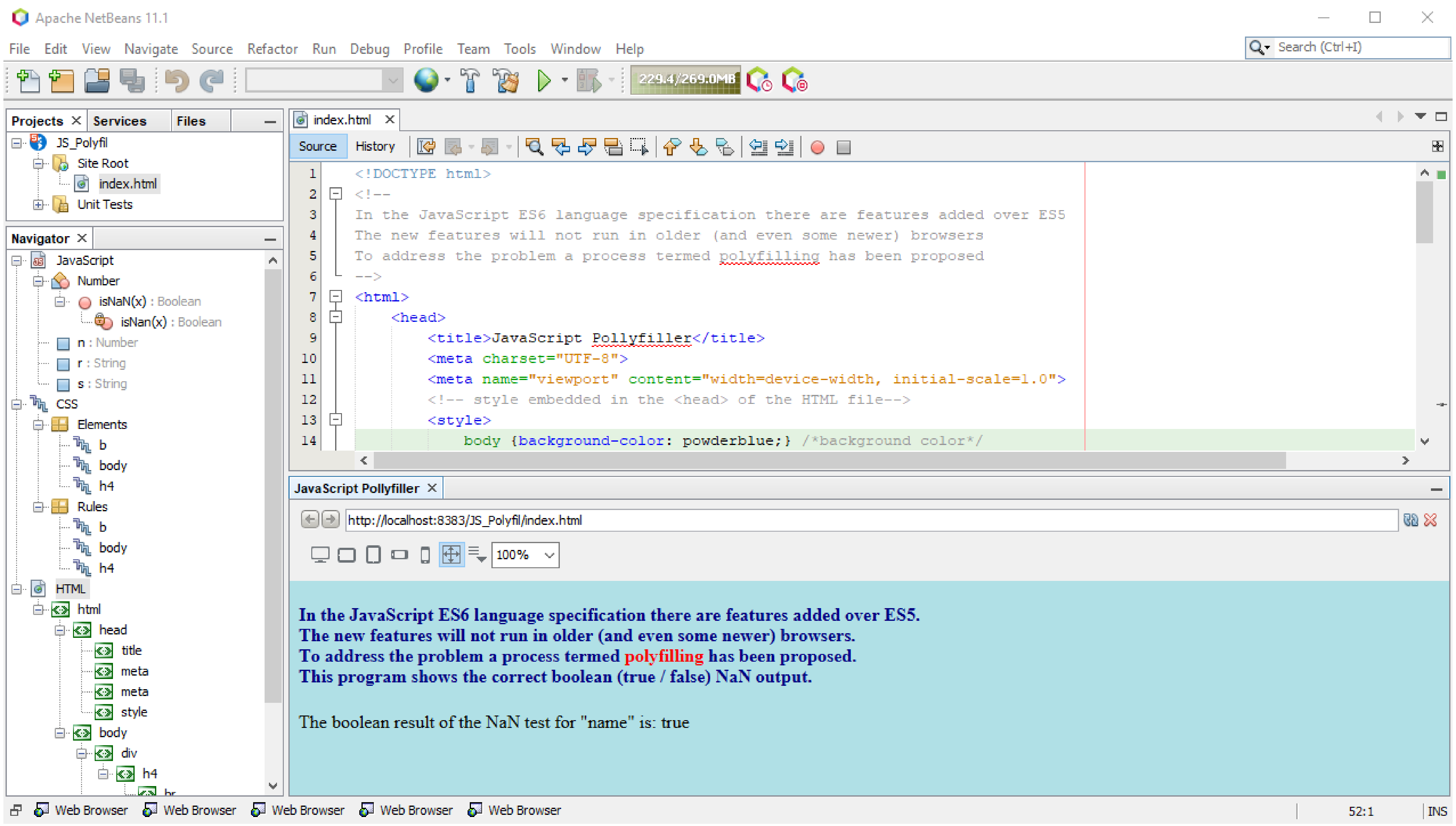
The boolean result of the NaN test for 7 is: false



```
<!DOCTYPE html>
<!--
In the JavaScript ES6 language specification there are features added over ES5
The new features will not run in older (and even some newer) browsers
To address the problem a process termed polyfilling has been proposed
-->
<html>
    <head>
        <title>JavaScript Pollyfiller</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- style embedded in the <head> of the HTML file-->
        <style>
            body {background-color: powderblue;} /*background color*/
            h4 {color: navy ;}                   /*<h4> color*/
            b {color: red;}                      /*<b> (bold) color*/
        </style>
    </head>
    <body>
        <div>
            <h4>
                In the JavaScript ES6 language specification there are features added over ES5.
                <br>
                The new features will not run in older (and even some newer) browsers.
                <br>
                To address the problem a process termed <b>polyfilling</b> has been proposed.
                <br>
                This program shows the correct boolean (true / false) NaN output.
            </h4>
        </div>
```



```
                <br>
                The new features will not run in older (and even some newer) browsers.
                <br>
                To address the problem a process termed <b>polyfilling</b> has been proposed.
                <br>
                This program shows the correct boolean (true / false) NaN output.
            </h4>
        </div>
        <script>
            "use strict";
            var s = "name";
            var n = 7;
            var r;
            //pass the variable into r = isNnN(s or n)
            document.write("The boolean result of the NaN test for \"" +
                    s + "\" is: " + (r = isNaN(s) + "<br>"));
            /*
             * The output is boolean based on the value
             * Where the value is a number the output is false
             * Where the value is not a number the output is true
             */
            if(!Number.isNaN) {
                Number.isNaN = function isNan(x) {
                    return x !== x;
                };
            }
        </script>
    </body>
</html>
```



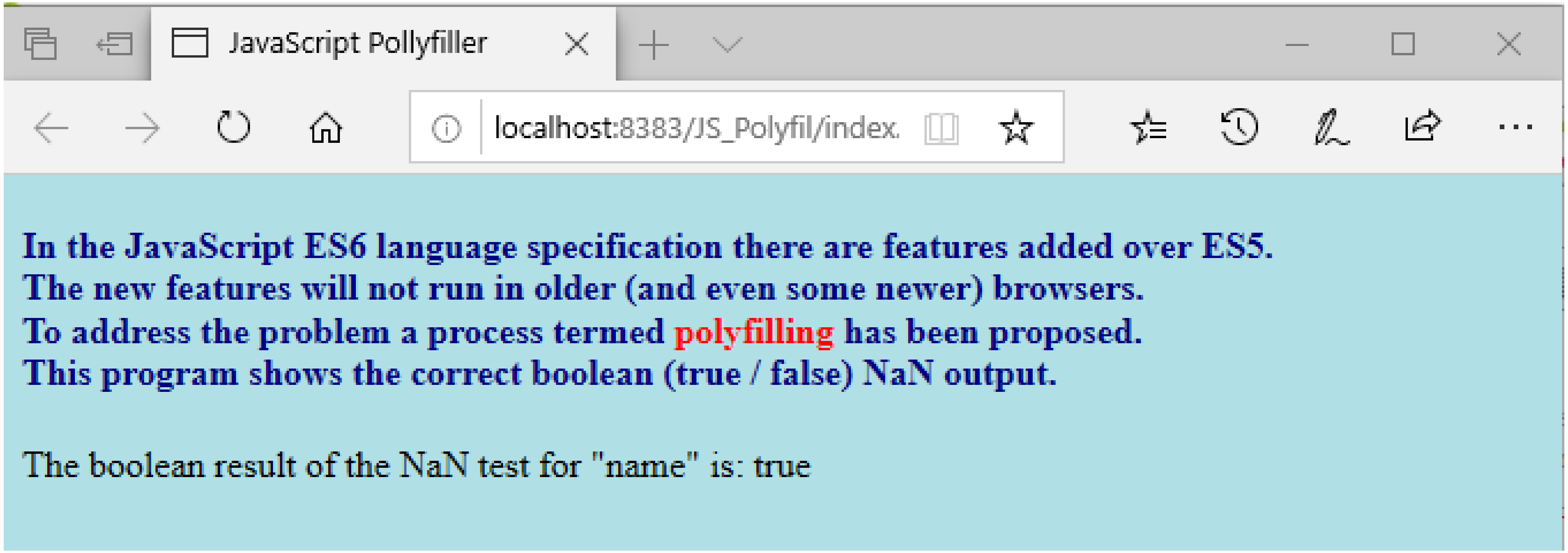
```
<!DOCTYPE html>
<!--
In the JavaScript ES6 language specification there are features added over ES5
The new features will not run in older (and even some newer) browsers
To address the problem a process termed polyfilling has been proposed
-->
<html>
    <head>
        <title>JavaScript Pollyfiller</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <!-- style embedded in the <head> of the HTML file-->
        <style>
            body {background-color: powderblue;} /*background color*/
```

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

In the JavaScript ES6 language specification there are features added over ES5.
The new features will not run in older (and even some newer) browsers.
To address the problem a process termed polyfilling has been proposed.
This program shows the correct boolean (true / false) NaN output.

The boolean result of the NaN test for "name" is: true

Output in MS Edge Browser



In the JavaScript ES6 language specification there are features added over ES5.
The new features will not run in older (and even some newer) browsers.
To address the problem a process termed polyfilling has been proposed.
This program shows the correct boolean (true / false) NaN output.

The boolean result of the NaN test for "name" is: true

`Polyfilling` JavaScript

- Not all new features in JavaScript ES6 are `polyfillable`
  - While most of the JavaScript behaviour can be `polyfilled` there are some deviations
  - Care should be exercised when implementing a `polyfill` yourself to ensure that you are complying with the ES5 and ES6 language specifications
- The worked example has shown how the new `polyfiller` JavaScript code is used in a `boolean` output for a `NaN` test for `string` and `number` values

Review

- In this brief overview of the ES6 version of the JavaScript
- We have introduced the following features and approaches to coding the JavaScript to try to address browser incompatibility
- We have shown worked examples for:
  - `const`
  - `symbol`
  - `transpiling`
  - `Polyfilling`
- The issues in the use of ES6 code demonstrates that extensive testing of the JavaScript and HTML code in multiple browsers and devices is critical