INFO 151
Web Systems and Services

JavaScript

Programming

Debugging

Overview

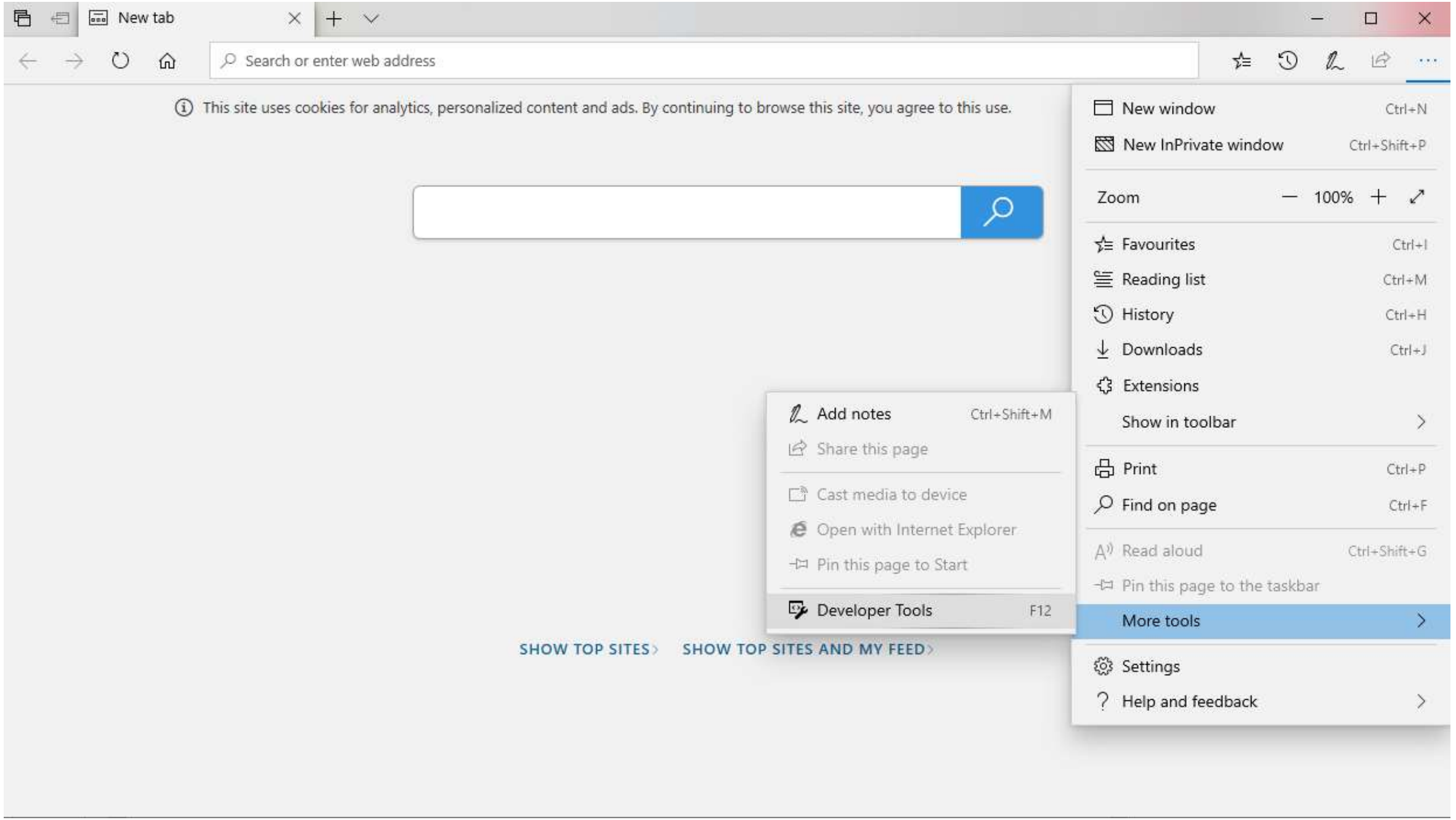
- In this tutorial I will introduce:
  - JavaScript programming including using:
    - The developer tools console in a web browser
    - A simple text editor
  - String output to the browser
  - The normal and `"strict"` modes
- I will introduce:
  - Documenting program code
  - Debugging program code
- In other tutorials I will introduce the new JavaScript ES6 language specification and the principles of program design

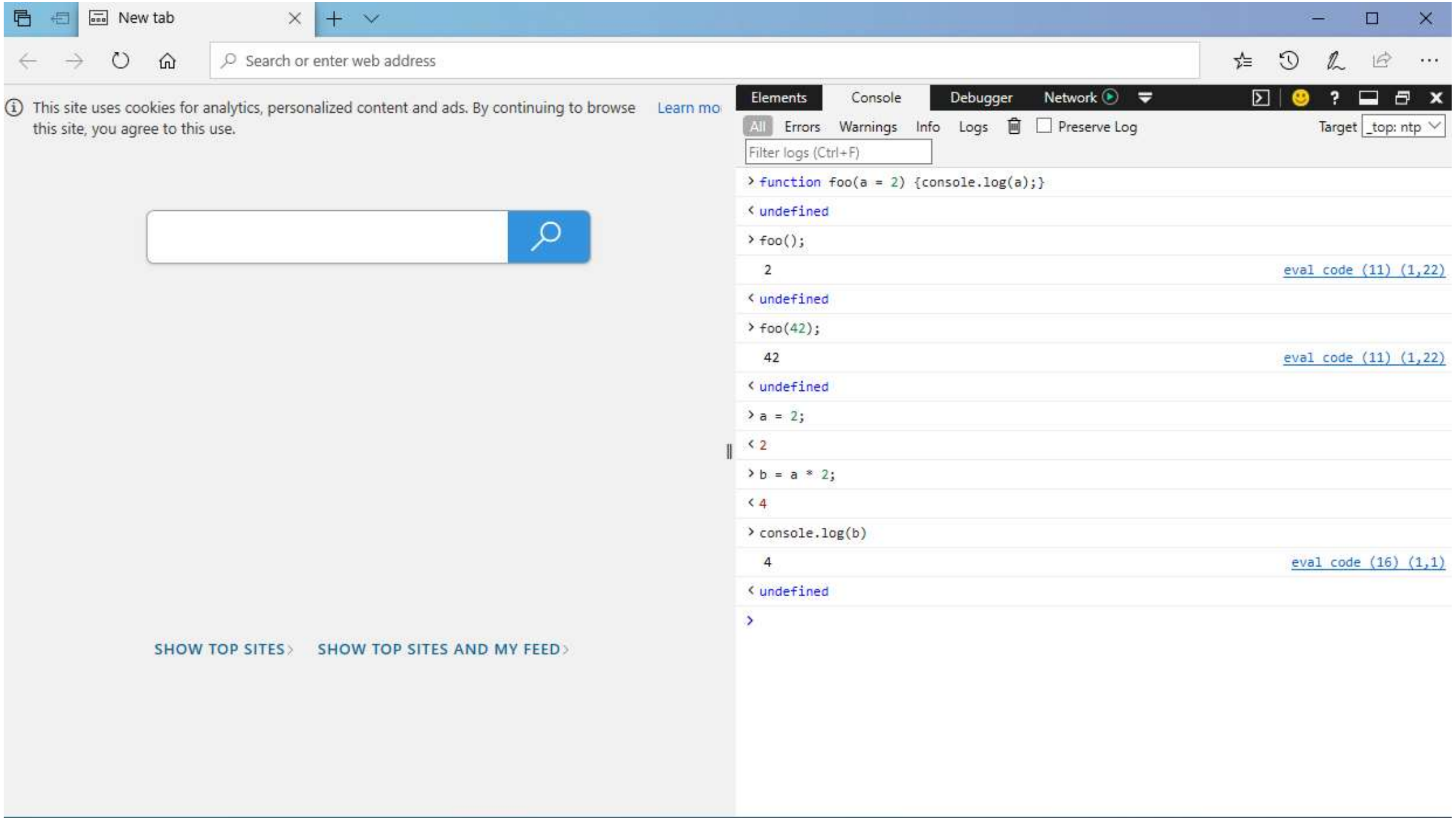
JavaScript Programming

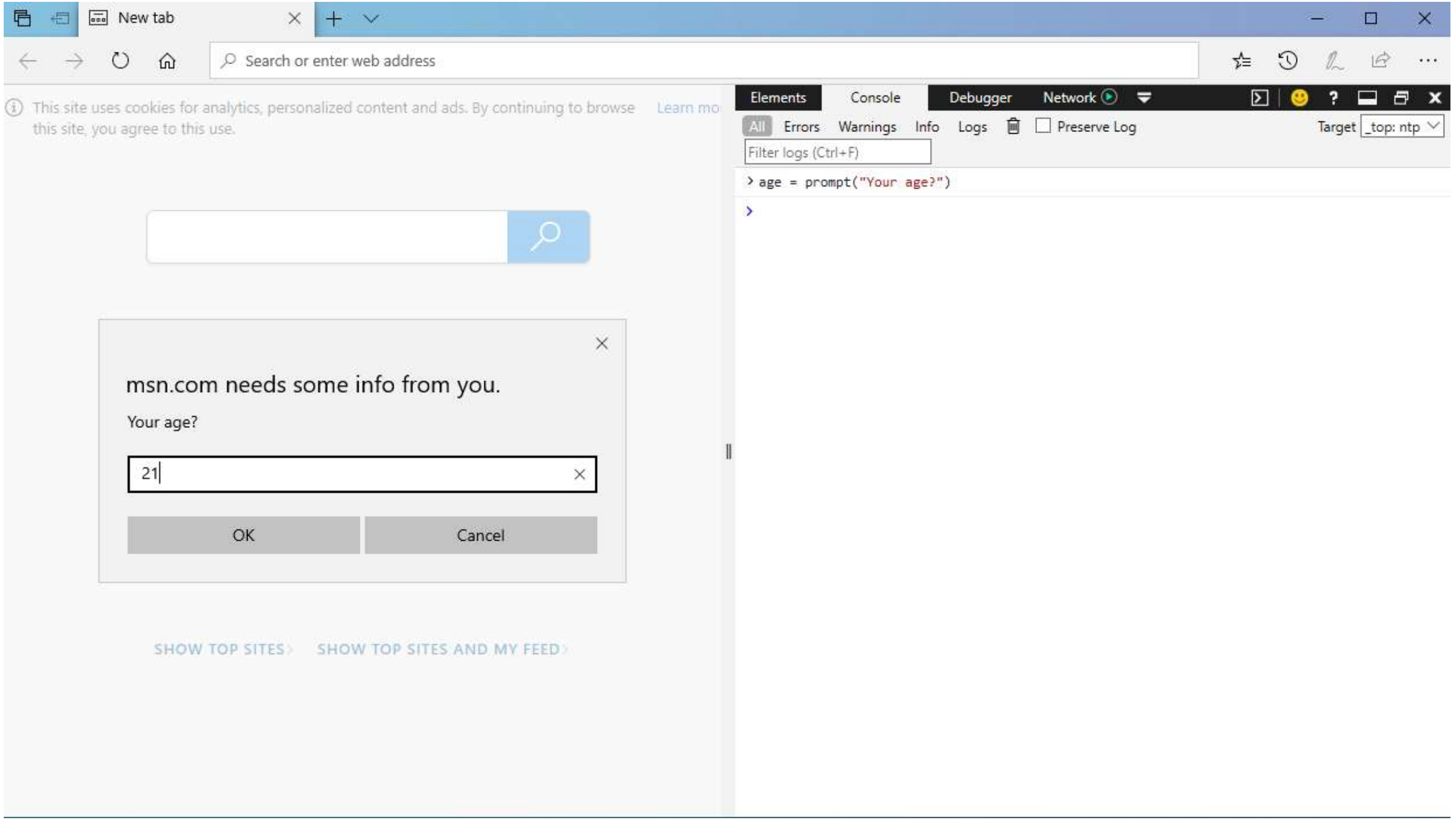
Programming JavaScript

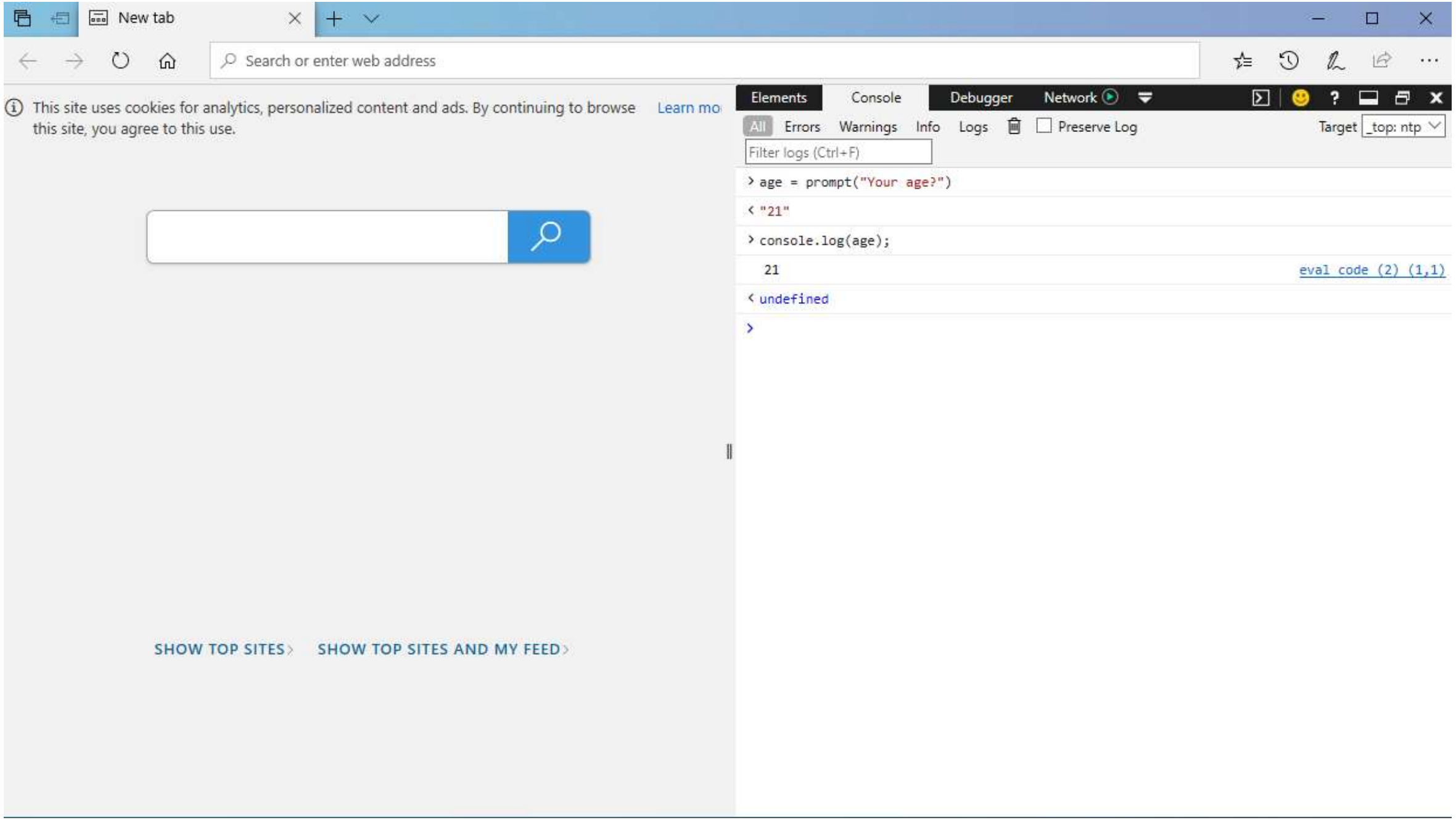
- In this course we are using the NetBeans IDE to program:
  - HTML (files stored in the NetBeans Projects folder)
  - JavaScript (files stored in the NetBeans Projects folder)
  - PHP and MySQL (files stored in the XAMPP htdocs Projects folder)
- We may also write HTML and JavaScript using:
  - The developer tools console in your web browser
  - A simple text editor such as Notepad or an application such as EMACS
- The following slides show the use of Notepad and developer tools

Developer Tools

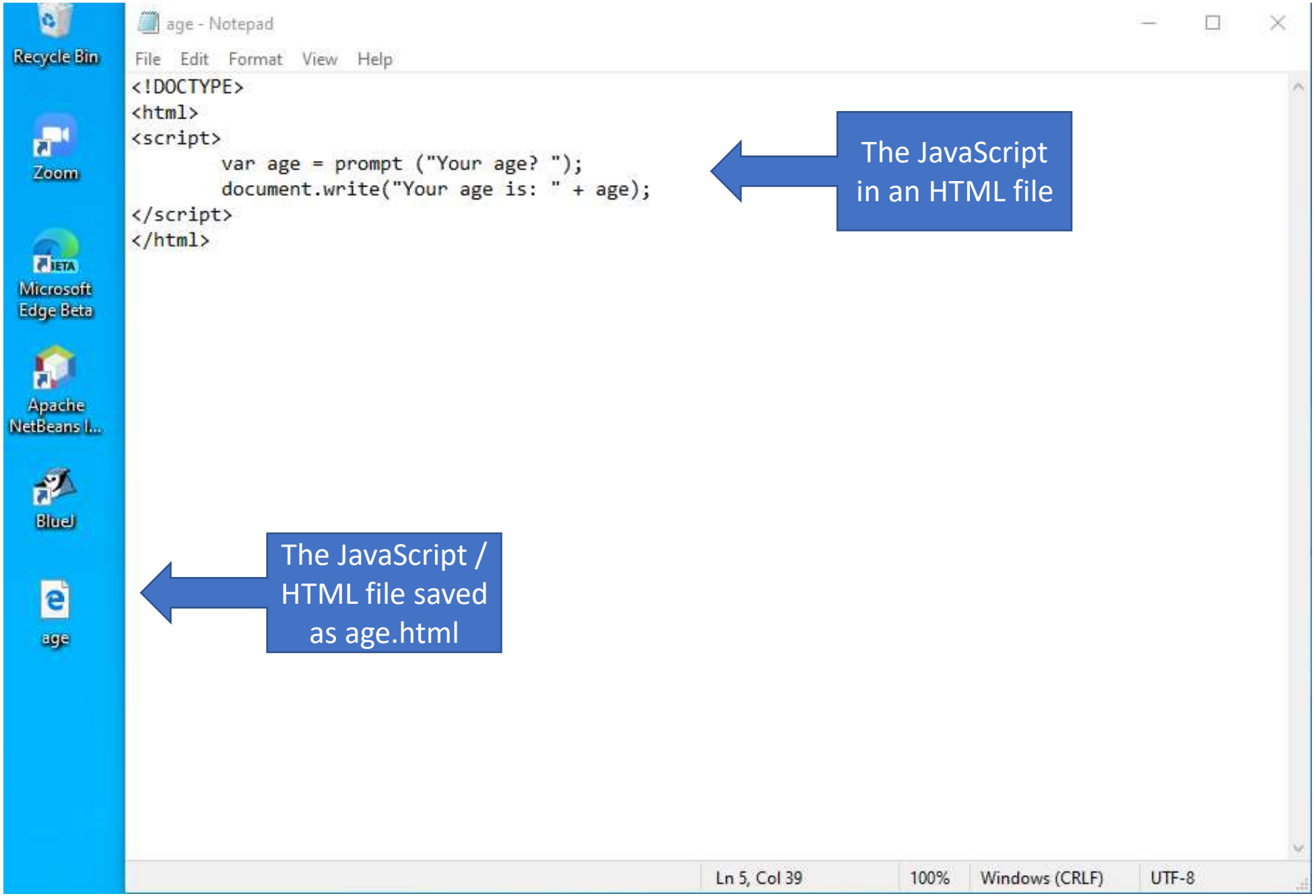

New tab
Search or enter web address
This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use.
New window Ctrl+N
New InPrivate window Ctrl+Shift+P
Zoom 100%
Favourites Ctrl+I
Reading list Ctrl+M
History Ctrl+H
Downloads Ctrl+J
Extensions
Show in toolbar
Print Ctrl+P
Find on page Ctrl+F
Read aloud Ctrl+Shift+G
Pin this page to the taskbar
More tools
Settings
Help and feedback
Add notes Ctrl+Shift+M
Share this page
Cast media to device
Open with Internet Explorer
Pin this page to Start
Developer Tools F12
SHOW TOP SITES
SHOW TOP SITES AND MY FEED

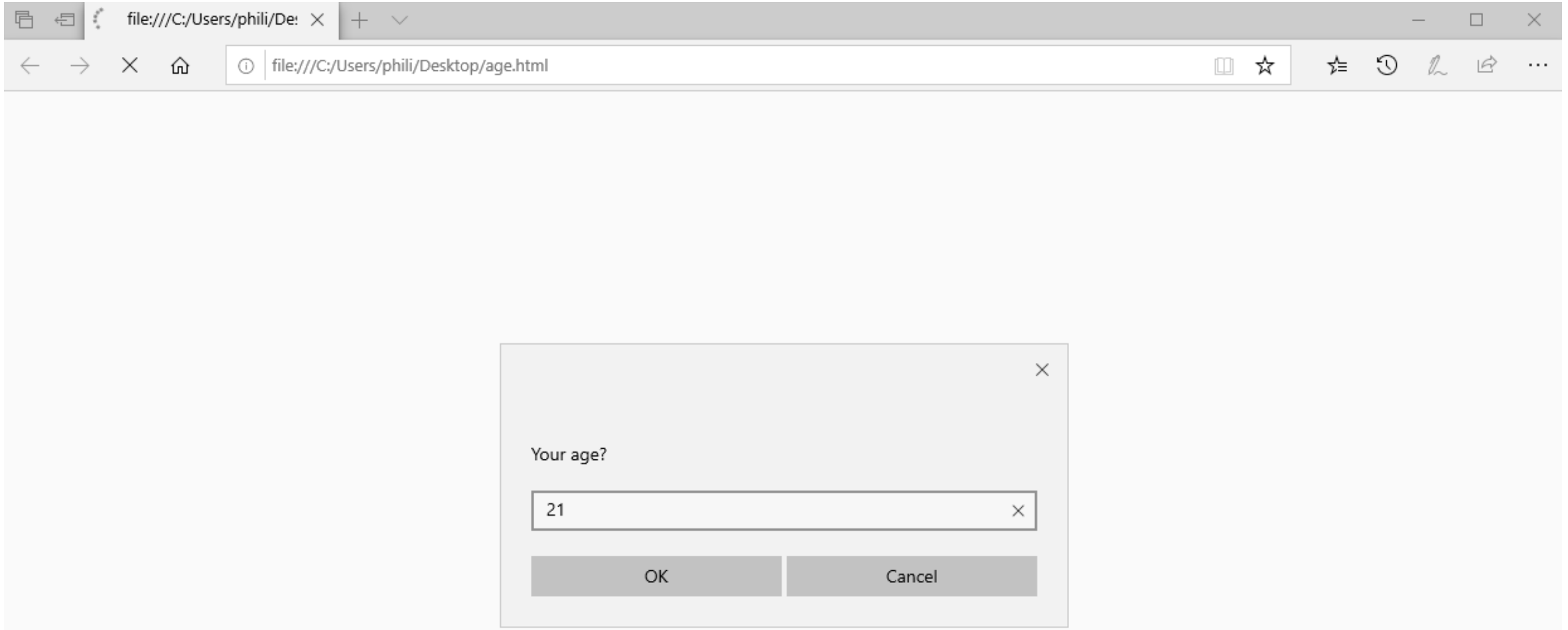

New tab
Search or enter web address
This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use.
Learn mo
SHOW TOP SITES
SHOW TOP SITES AND MY FEED
Elements
Console
Debugger
Network
All
Errors
Warnings
Info
Logs
Preserve Log
Target
_top: ntp
Filter logs (Ctrl+F)
> function foo(a = 2) {console.log(a);}
< undefined
> foo();
2
eval code (11) (1,22)
< undefined
> foo(42);
42
eval code (11) (1,22)
< undefined
> a = 2;
< 2
> b = a * 2;
< 4
> console.log(b)
4
eval code (16) (1,1)
< undefined
>


New tab
Search or enter web address
This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use.
Learn mo
msn.com needs some info from you.
Your age?
21
OK
Cancel
SHOW TOP SITES
SHOW TOP SITES AND MY FEED
Elements
Console
Debugger
Network
All
Errors
Warnings
Info
Logs
Preserve Log
Target
_top: ntp
Filter logs (Ctrl+F)
> age = prompt("Your age?")
>


New tab
Search or enter web address
This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use.
Learn mo
SHOW TOP SITES
SHOW TOP SITES AND MY FEED
Elements
Console
Debugger
Network
All
Errors
Warnings
Info
Logs
Preserve Log
Target
_top: ntp
Filter logs (Ctrl+F)
> age = prompt("Your age?")
< "21"
> console.log(age);
21
eval code (2) (1,1)
< undefined
>

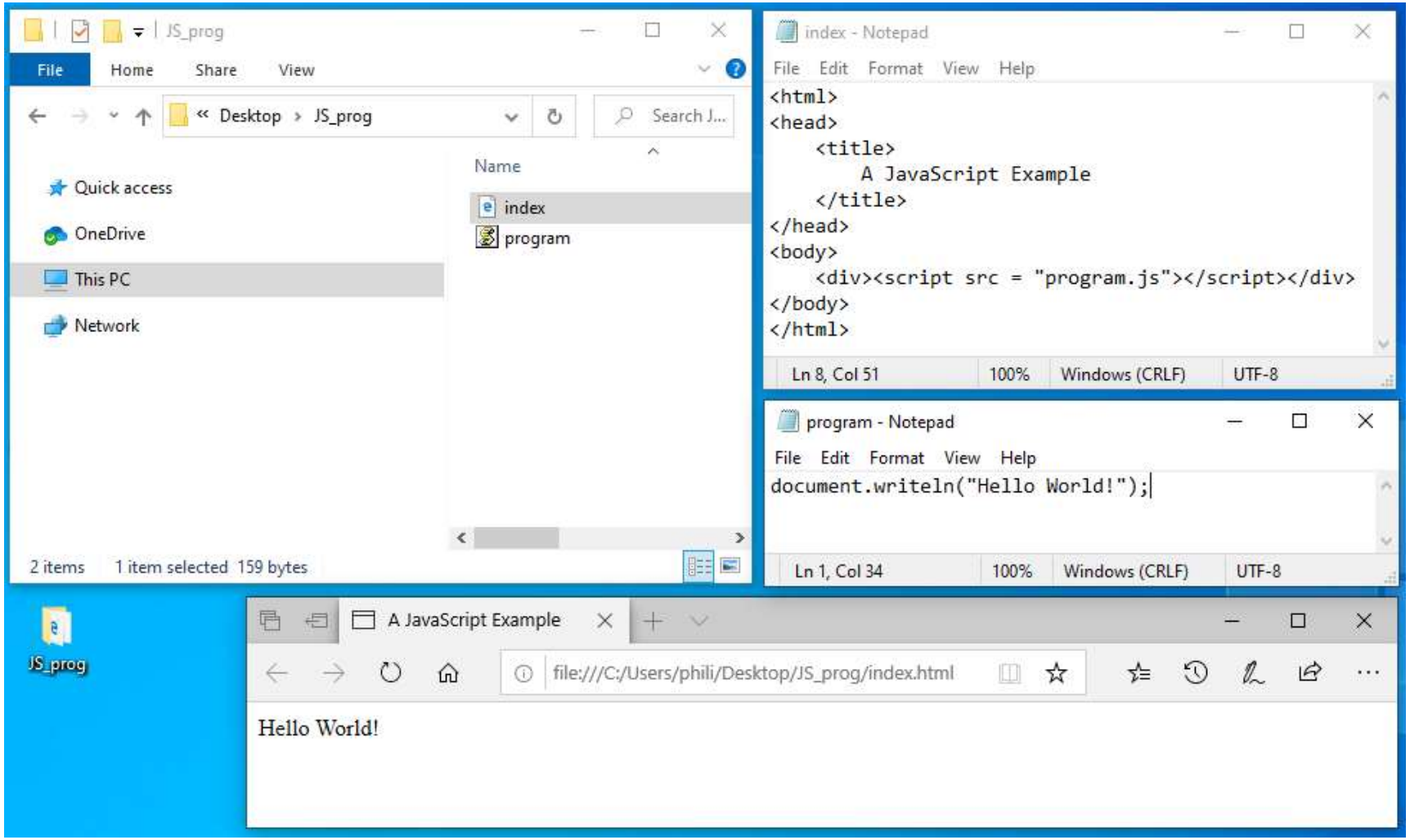
Notepad


Recycle Bin
Zoom
Microsoft Edge Beta
Apache NetBeans I...
BlueJ
age
age - Notepad
File Edit Format View Help
<!DOCTYPE>
<html>
<script>
var age = prompt ("Your age? ");
document.write("Your age is: " + age);
</script>
</html>
Ln 5, Col 39
100%
Windows (CRLF)
UTF-8
The JavaScript in an HTML file
The JavaScript / HTML file saved as age.html


file:///C:/Users/phili/De
file:///C:/Users/phili/Desktop/age.html
Your age?
21
OK
Cancel

Your age is: 21

JavaScript in Notepad with a Separate File



Programming JavaScript

- The developer tools and Notepad approaches:
  - Can be useful when learning to program HTML and JavaScript
- However:
  - Real programs will require multiple files
  - Notepad can accommodate multiple files in a single folder
  - Developer tools make complex programs difficult to code and run
- In the ‘real-world’:
  - IDE tools (such a NetBeans and other similar tools) will be used

String Output to Browser

JavaScript String Output

- To print JavaScript strings and variables there are two methods:
- One:

```
document.write("Hello JavaScript!");
```

- Two:

```
<p id="demo"></p>
document.getElementById("demo").innerHTML = ("Hello
JavaScript!");
```

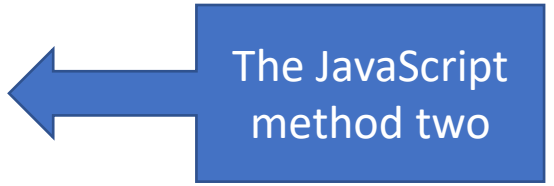
- Both methods produce the same output!

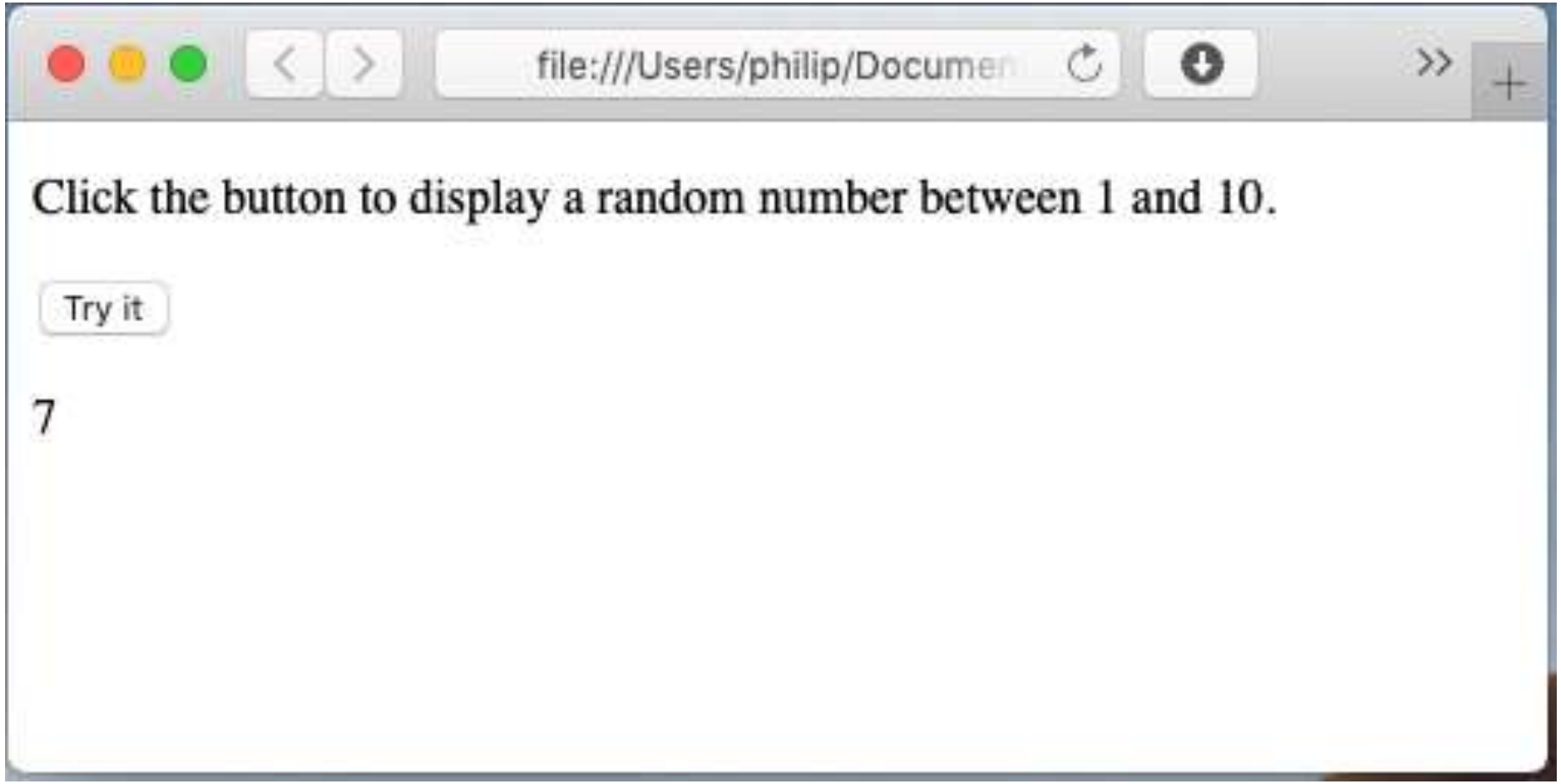
document Output

- Method one: `document.write()`:
  - Writes HTML expressions or JavaScript code to a document
  - The `write()` method is mostly used for testing
  - If it is used after an HTML document is fully loaded it will delete all existing HTML
- Method two: `document.getElementById()`:
  - The `documentElement` property returns the `documentElement` of the `document` as an `Element object` – for HTML documents the returned `object` is the <html> `element`

```
<!DOCTYPE html>
<!--
A JavaScript examples to create random integers between 1 and 10
-->
<html>
    <head>
        <title>Math.random() 1-10</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
   <body>
       <p>Click the button to display a random number between 1 and 10.</p>
       <button onclick="myFunction()">Try it</button>
       <p id="demo"></p>
       <script>
           function myFunction() {
               var n = Math.floor((Math.random() * 10) + 1);
               document.getElementById("demo").innerHTML = n;
            }
       </script>
    </body>
</html>

U:---  index.html      Top L1      (HTML+)
```




file:///Users/philip/Documer
Click the button to display a random number between 1 and 10.
Try it
7

document.getElementById():

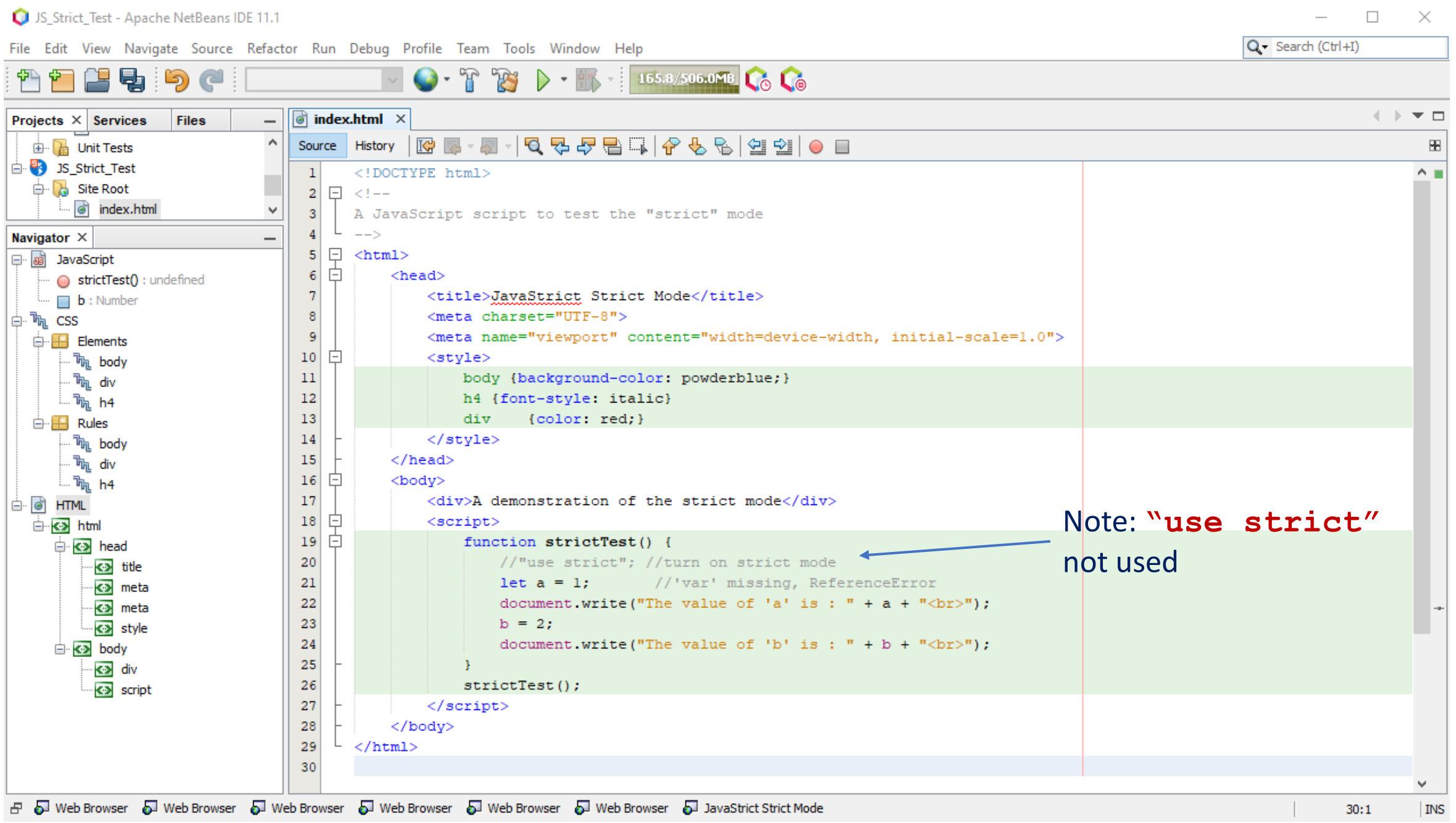
- The `document` variable exists as a GLOBAL variable when the code is running in a browser
- It is in fact a `non-JavaScript element` as it is controlled by the DOM API
- The `document` variable appears to be a normal JavaScript `object` while in practice it is a special `object`
- While `document.getElementById()` method seems to be a normal JavaScript method: it is in practice an interface to a built-in method in the Document Object Model (DOM) provided by the browser
- In some later browsers the layer may be also be in the JavaScript

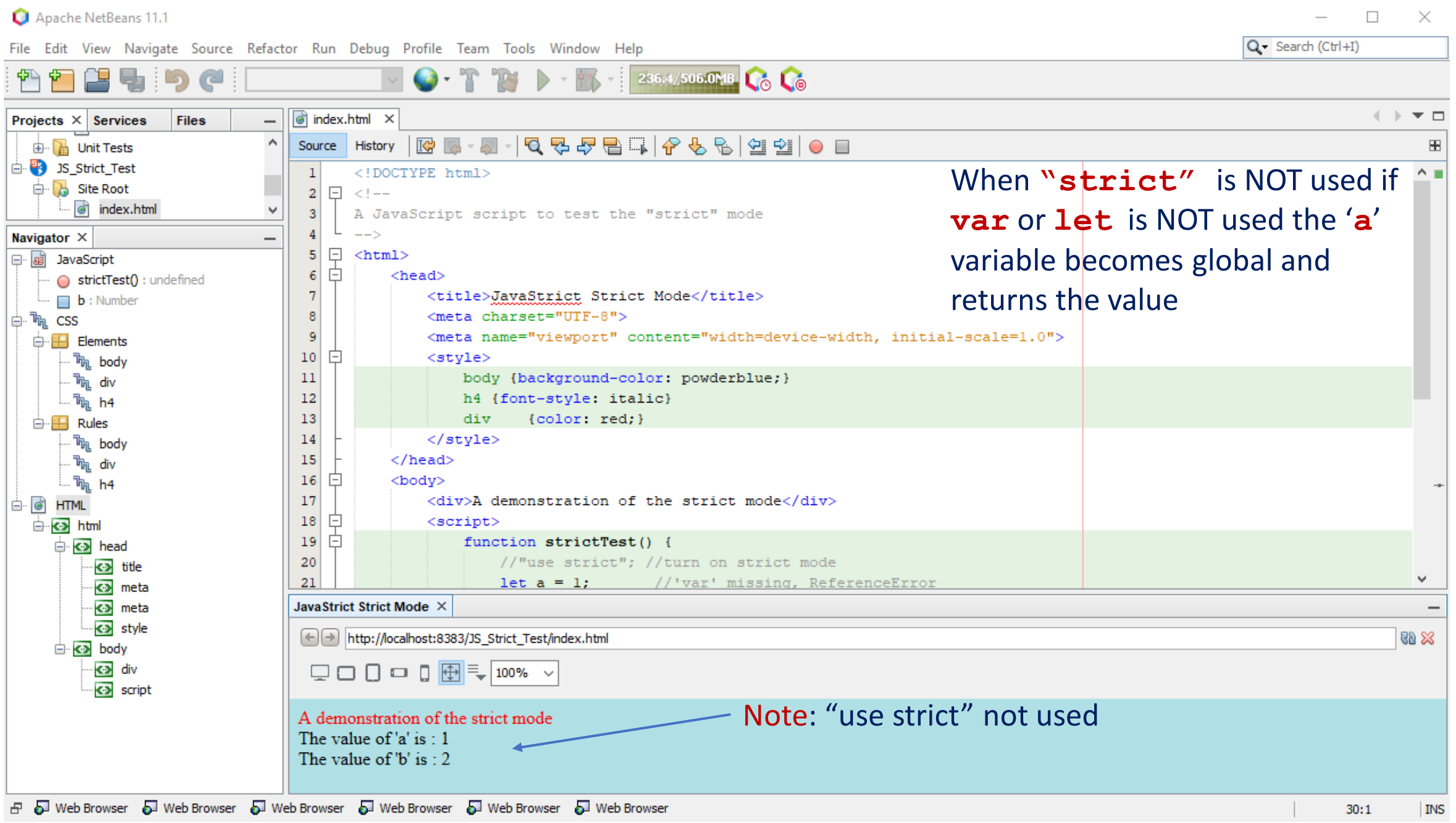
Running a JavaScript program

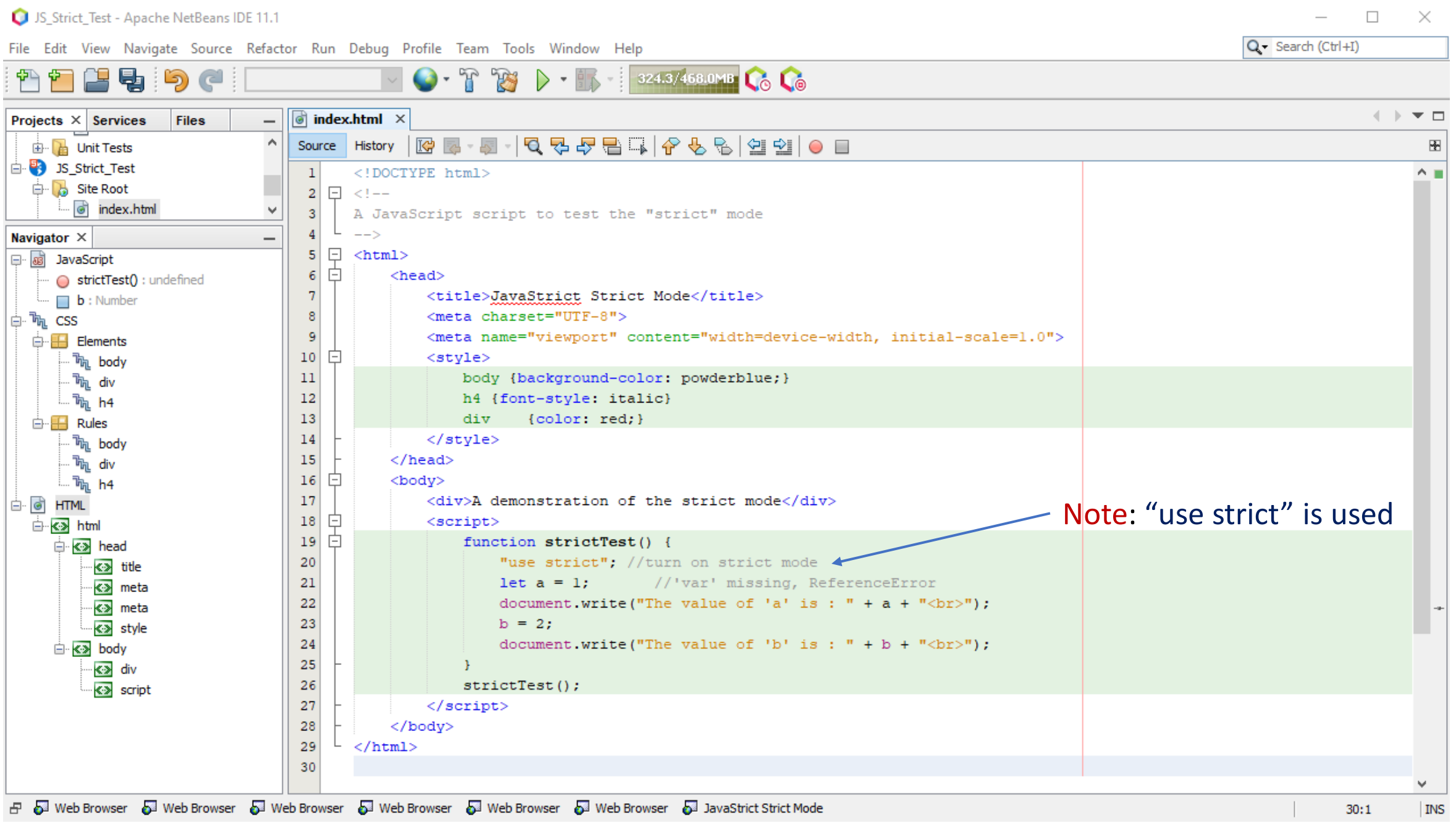
"strict"

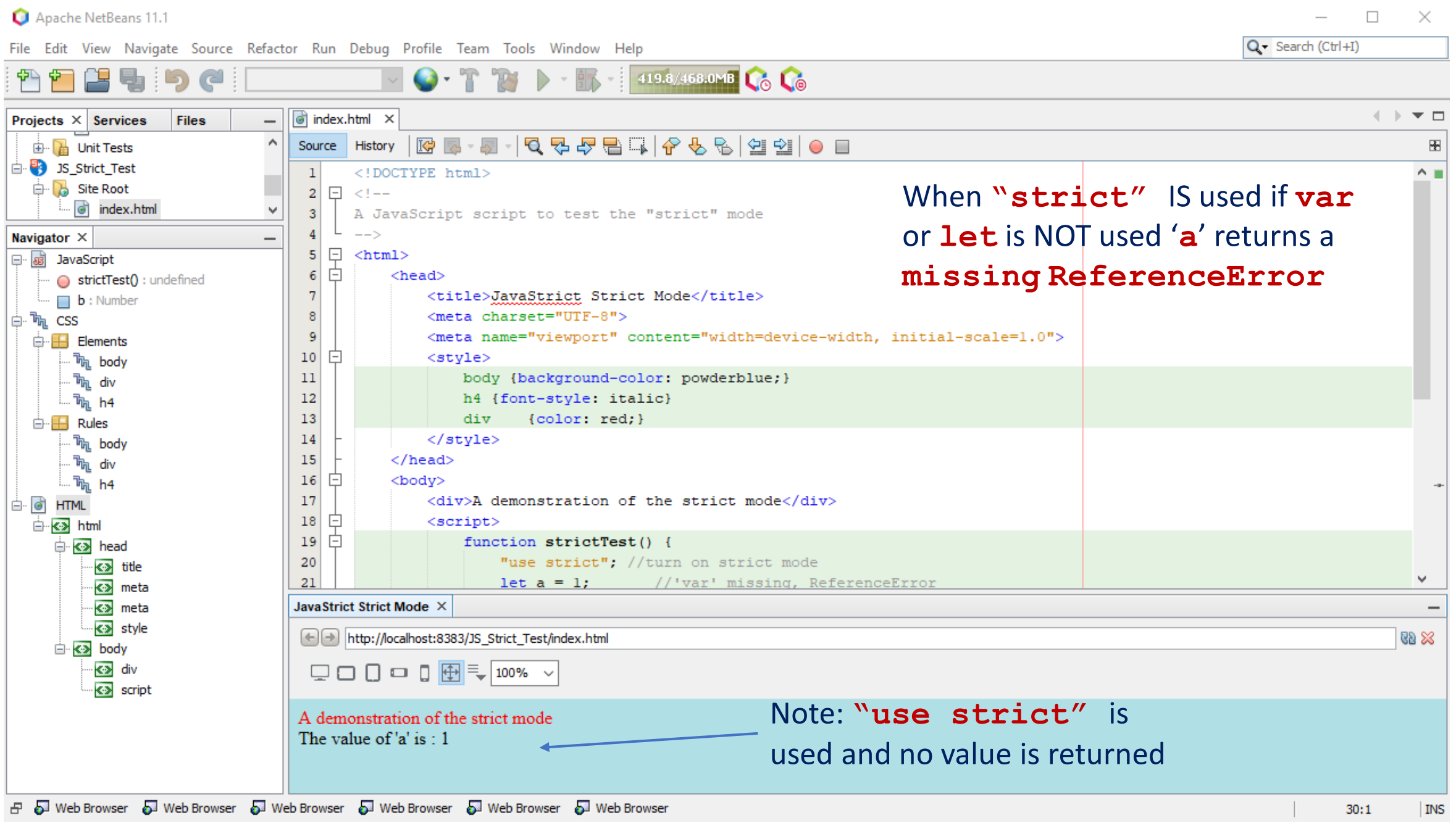
Running a JavaScript program

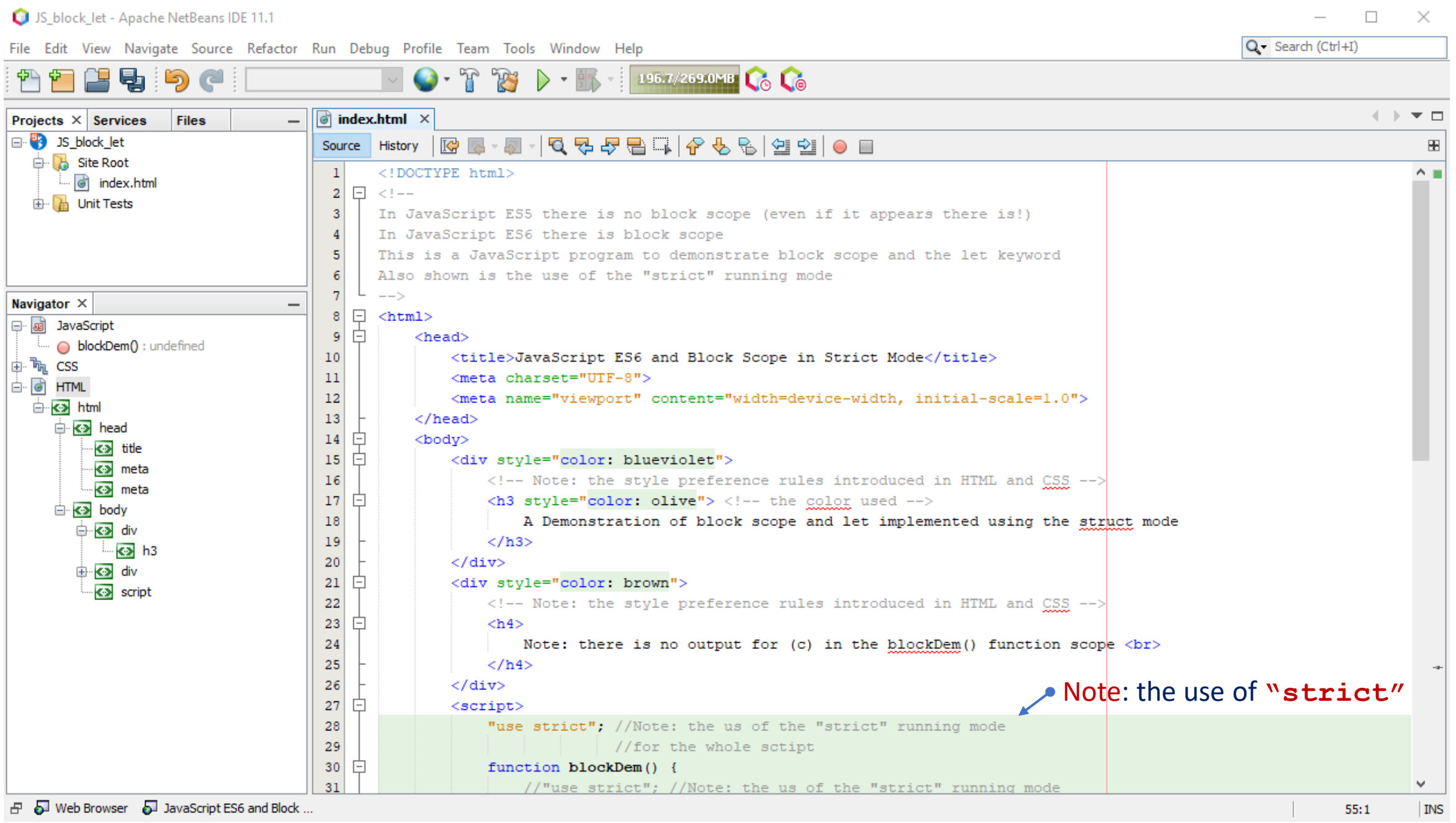
- To run a JavaScript program (the script either alone or when embedded in a HTML web page file) there are two run modes:
  - The normal mode (and)
  - The **"strict"** mode (added to the JavaScript language specification in ES5)
- The effect of running a JavaScript program as **"strict"** is:
  - To enforce the language rules for many behaviors
  - To make the code more easily optimized by the compiler
  - The downside is: if the code has syntax errors, they will be reported
- The following worked example demonstrates the **"strict"** mode

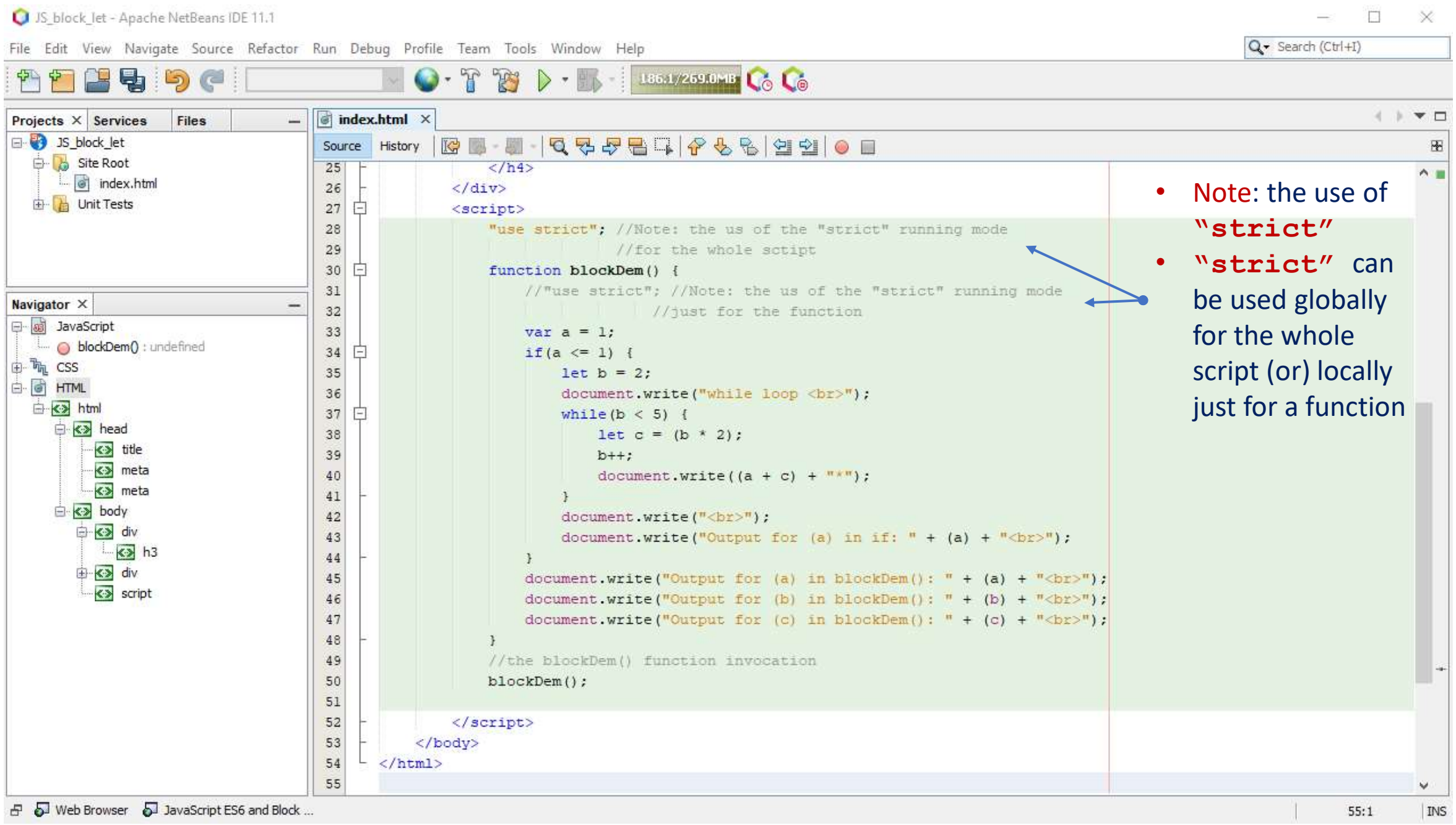

JS_Strict_Test - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
165.8/506.0MB
Projects
Services
Files
Unit Tests
JS_Strict_Test
Site Root
index.html
Navigator
JavaScript
strictTest() : undefined
b : Number
CSS
Elements
body
div
h4
Rules
body
div
h4
HTML
html
head
title
meta
meta
style
body
div
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript script to test the "strict" mode
4 -->
5 <html>
6 <head>
7 <title>JavaStrict Strict Mode</title>
8 <meta charset="UTF-8">
9 <meta name="viewport" content="width=device-width, initial-scale=1.0">
10 <style>
11 body {background-color: powderblue;}
12 h4 {font-style: italic}
13 div {color: red;}
14 </style>
15 </head>
16 <body>
17 A demonstration of the strict mode
18 <script>
19 function strictTest() {
20 //"use strict"; //turn on strict mode
21 let a = 1; //'var' missing, ReferenceError
22 document.write("The value of 'a' is : " + a + "
");
23 b = 2;
24 document.write("The value of 'b' is : " + b + "
");
25 }
26 strictTest();
27 </script>
28 </body>
29 </html>
30
Note: "use strict" not used
Web Browser Web Browser Web Browser Web Browser Web Browser Web Browser JavaStrict Strict Mode
30:1
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
236.4/506.0MB
Projects Services Files
Unit Tests
JS_Strict_Test
Site Root
index.html
Navigator
JavaScript
strictTest() : undefined
b : Number
CSS
Elements
body
div
h4
Rules
body
div
h4
HTML
html
head
title
meta
meta
style
body
div
script
index.html
Source History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript script to test the "strict" mode
4 -->
5 <html>
6 <head>
7 <title>JavaStrict Strict Mode</title>
8 <meta charset="UTF-8">
9 <meta name="viewport" content="width=device-width, initial-scale=1.0">
10 <style>
11 body {background-color: powderblue;}
12 h4 {font-style: italic}
13 div {color: red;}
14 </style>
15 </head>
16 <body>
17 A demonstration of the strict mode
18 <script>
19 function strictTest() {
20 //"use strict"; //turn on strict mode
21 let a = 1; //'var' missing, ReferenceError
When "strict" is NOT used if var or let is NOT used the 'a' variable becomes global and returns the value
JavaStrict Strict Mode
http://localhost:8383/JS_Strict_Test/index.html
100%
A demonstration of the strict mode
The value of 'a' is : 1
The value of 'b' is : 2
Note: "use strict" not used
Web Browser Web Browser Web Browser Web Browser Web Browser Web Browser
30:1
INS

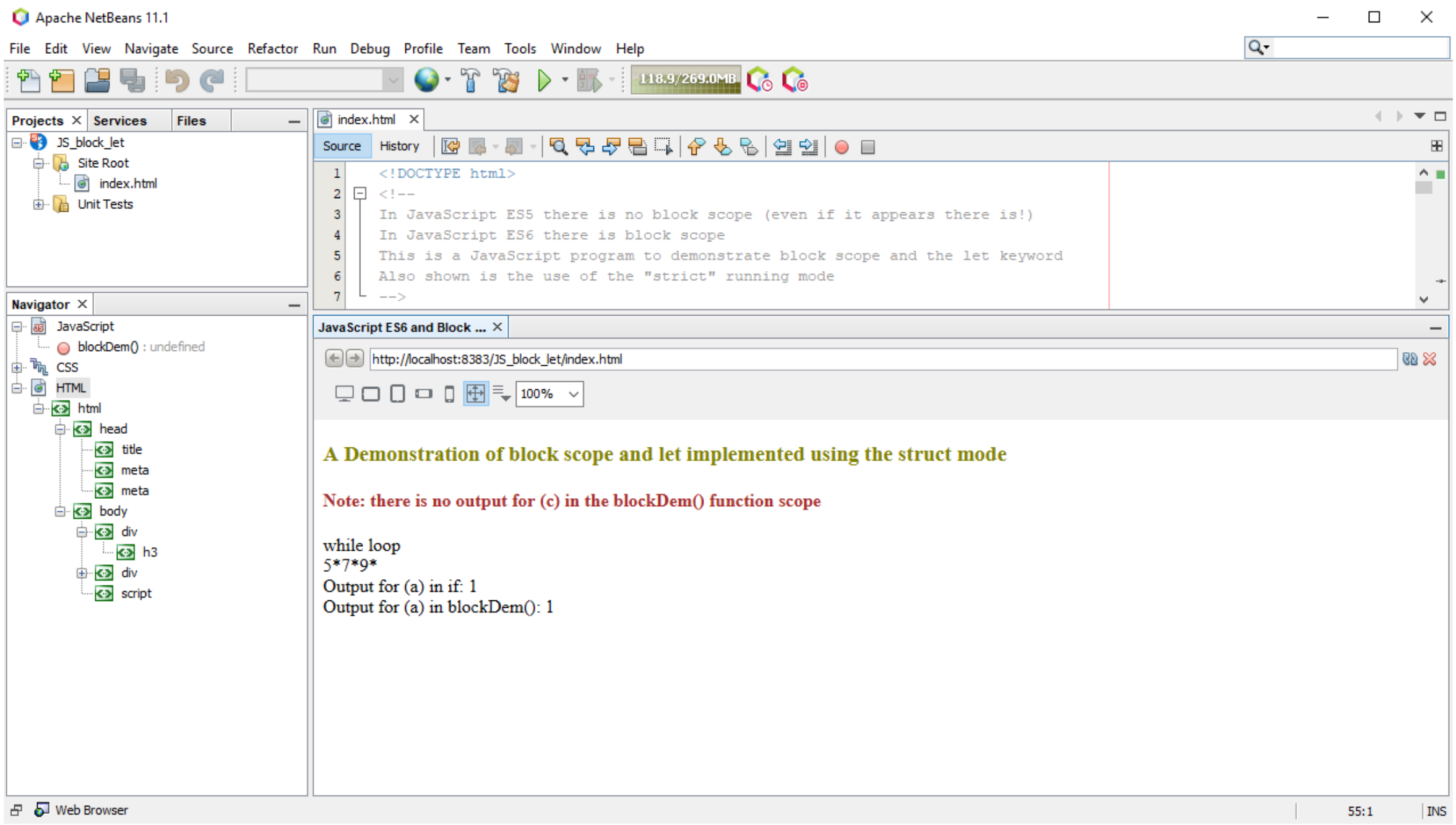

JS_Strict_Test - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
324.3/468.0MB
Projects
Services
Files
Unit Tests
JS_Strict_Test
Site Root
index.html
Navigator
JavaScript
strictTest() : undefined
b : Number
CSS
Elements
body
div
h4
Rules
body
div
h4
HTML
html
head
title
meta
meta
style
body
div
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript script to test the "strict" mode
4 -->
5 <html>
6 <head>
7 <title>JavaStrict Strict Mode</title>
8 <meta charset="UTF-8">
9 <meta name="viewport" content="width=device-width, initial-scale=1.0">
10 <style>
11 body {background-color: powderblue;}
12 h4 {font-style: italic}
13 div {color: red;}
14 </style>
15 </head>
16 <body>
17 A demonstration of the strict mode
18 <script>
19 function strictTest() {
20 "use strict"; //turn on strict mode
21 let a = 1; //'var' missing, ReferenceError
22 document.write("The value of 'a' is : " + a + "
");
23 b = 2;
24 document.write("The value of 'b' is : " + b + "
");
25 }
26 strictTest();
27 </script>
28 </body>
29 </html>
30
Note: “use strict” is used
Web Browser
Web Browser
Web Browser
Web Browser
Web Browser
Web Browser
JavaStrict Strict Mode
30:1
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
419.8/468.0MB
Projects Services Files
Unit Tests
JS_Strict_Test
Site Root
index.html
Navigator
JavaScript
strictTest() : undefined
b : Number
CSS
Elements
body
div
h4
Rules
body
div
h4
HTML
html
head
title
meta
meta
style
body
div
script
index.html
Source History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript script to test the "strict" mode
4 -->
5 <html>
6 <head>
7 <title>JavaStrict Strict Mode</title>
8 <meta charset="UTF-8">
9 <meta name="viewport" content="width=device-width, initial-scale=1.0">
10 <style>
11 body {background-color: powderblue;}
12 h4 {font-style: italic}
13 div {color: red;}
14 </style>
15 </head>
16 <body>
17 A demonstration of the strict mode
18 <script>
19 function strictTest() {
20 "use strict"; //turn on strict mode
21 let a = 1; //'var' missing, ReferenceError
When "strict" IS used if var or let is NOT used 'a' returns a missing ReferenceError
JavaStrict Strict Mode
http://localhost:8383/JS_Strict_Test/index.html
100%
A demonstration of the strict mode
The value of 'a' is : 1
Note: "use strict" is used and no value is returned
Web Browser Web Browser Web Browser Web Browser Web Browser Web Browser
30:1
INS


JS_block_let - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
196.7/269.0MB
Projects
Services
Files
JS_block_let
Site Root
index.html
Unit Tests
Navigator
JavaScript
blockDem() : undefined
CSS
HTML
html
head
title
meta
meta
body
div
h3
div
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 In JavaScript ES5 there is no block scope (even if it appears there is!)
4 In JavaScript ES6 there is block scope
5 This is a JavaScript program to demonstrate block scope and the let keyword
6 Also shown is the use of the "strict" running mode
7 -->
8 <html>
9 <head>
10 <title>JavaScript ES6 and Block Scope in Strict Mode</title>
11 <meta charset="UTF-8">
12 <meta name="viewport" content="width=device-width, initial-scale=1.0">
13 </head>
14 <body>
15
16 <!-- Note: the style preference rules introduced in HTML and CSS -->
17 <h3 style="color: olive"> <!-- the color used -->
18 A Demonstration of block scope and let implemented using the struct mode
19 </h3>
20
21
22 <!-- Note: the style preference rules introduced in HTML and CSS -->
23 <h4>
24 Note: there is no output for (c) in the blockDem() function scope
25 </h4>
26
27 <script>
28 "use strict"; //Note: the us of the "strict" running mode
29 //for the whole sctipt
30 function blockDem() {
31 //"use strict"; //Note: the us of the "strict" running mode
Note: the use of "strict"
Web Browser
JavaScript ES6 and Block ...
55:1
INS



- Note: the use of **"strict"**
- **"strict"** can be used globally for the whole script (or) locally just for a function


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
118.9/269.0MB
Projects Services Files
JS_block_let
Site Root
index.html
Unit Tests
Navigator
JavaScript
blockDem() : undefined
CSS
HTML
html
head
title
meta
meta
body
div
h3
div
script
index.html
Source History
1 <!DOCTYPE html>
2 <!--
3 In JavaScript ES5 there is no block scope (even if it appears there is!)
4 In JavaScript ES6 there is block scope
5 This is a JavaScript program to demonstrate block scope and the let keyword
6 Also shown is the use of the "strict" running mode
7 -->
JavaScript ES6 and Block ...
http://localhost:8383/JS_block_let/index.html
100%
A Demonstration of block scope and let implemented using the struct mode
Note: there is no output for (c) in the blockDem() function scope
while loop
5*7*9*
Output for (a) in if: 1
Output for (a) in blockDem(): 1
Web Browser
55:1
INS

Documenting JavaScript

Debugging JavaScript

JavaScript Debugging

- Writing and debugging JavaScript
  - Comment the program file and the code as it is written
  - Add regular test output test statements (they can be commented out) to track variable values and types
  - Keep a careful track of the variables and the variable assignments
  - Know the required output and test it against the actual output
- There are many debugging tools for all programming languages
  - Web browsers may include a debugging tool which can be used to debug a JavaScript script
- The starting point is to design the program and related algorithm
  - It is simpler to avoid bugs than it is to rectify errors

JavaScript Comments

- The following examples show the methods of inserting comments (documenting) JavaScript code

```
/*
 * This is a block multi-line Java, C, and C++ style comment
 * With a second line
 */

/* Another comment*/

// This is a single line Java, C, and C++ style comment
```

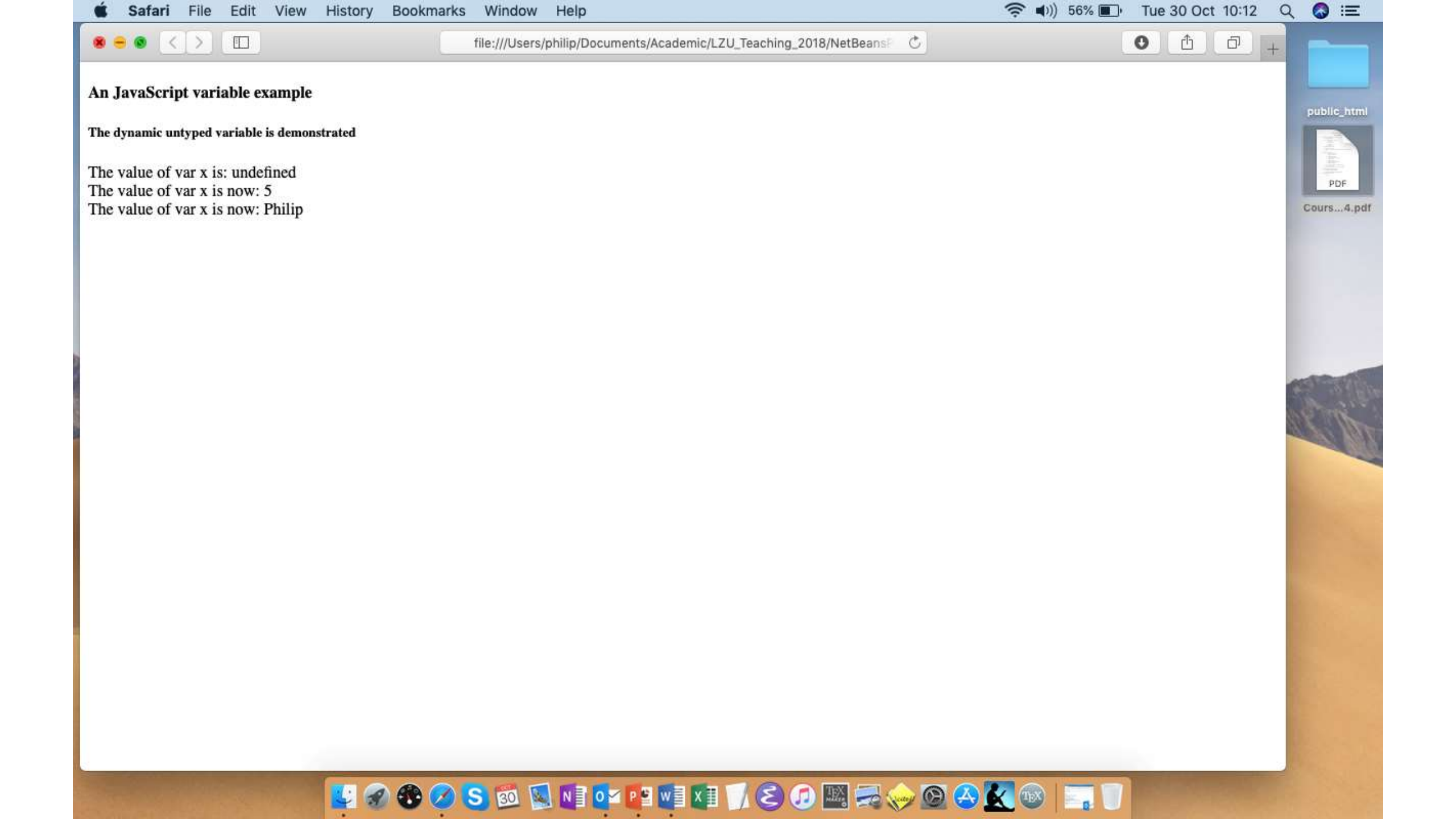
Testing Variable Typing with Test Statements

```
<script>
var x;
document.write("The value of var x is: " + x + "<br>" );  // test the value of x
var x = 5;
document.write("The value of var x is now: " + x + "<br>" );  // test the value
of x
var x = "Philip";
document.write("The value of var x is now: " + x + "<br>" );  // test the value
of x
</script>
```

- The following slide shows the JavaScript in a worked example in EMACS

```
<!DOCTYPE html>
<!--
An example of the dynamic nature
The untyped issue for global variables is shown
-->
<html>
    <head>
        <title>JavaSript Variable Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>
            <h4>An JavaScript variable example</h4>
            <h5>The dynamic untyped variable is demonstrated</h5>

            <script>
                var x;
                document.write("The value of var x is: " + x + "<br>" );
                var x = 5;
                document.write("The value of var x is now: " + x + "<br>" );
                var x = "Philip";
                document.write("The value of var x is now: " + x + "<br>" );
            </script>
        </div>
    </body>
</html>
```


Safari File Edit View History Bookmarks Window Help
56% Tue 30 Oct 10:12
file:///Users/philip/Documents/Academic/LZU_Teaching_2018/NetBeans
An JavaScript variable example
The dynamic untyped variable is demonstrated
The value of var x is: undefined
The value of var x is now: 5
The value of var x is now: Philip
public_html
PDF
Cours...4.pdf

Practical Examples

Design a Testing Strategy

- A typical approach the test computer programs is to:
  1. Develop a suitable testing strategy
  2. Develop and document test scenarios
  3. Develop tests to evaluate all true and false values
  4. Develop a randomly generated test dataset
  5. Run tests and evaluate the results against the test scenarios
- When the evaluation has been completed and any ‘bugs’ in the program have been eliminated:
  - Run the program with actual datasets
- The following example demonstrates the process

Control Context Processing Scenario: C_CP_8

Context Properties	Data Type	Functional Property	Literal Value	Validity
#event_Type	Integer	True	[2]	True
#action_Type	Integer	True	[1]	True
#user_Id	String	True		True
#user_Policy	Integer	True	[2]	False
#res_Type	Integer	True	[1, 2]	True

Exceptions Caught:	Invalid: #user_Policy
Anticipated (design) result:	False: in this scenario the user policy only permits a collaboration query input therefore the system terminates the context processing
Actual (experimental) result:	False: the anticipated design result is validated
Scenario testing result:	Correct
Design action:	Terminate context processing + error message
Actual action:	Terminate context processing + error message
Action result:	Correct

Notes:

- #event_Type [2] == a user input such as a resource or a collaboration query
- #action_Type [1] == a resource (file) input
- #user_Policy [1, 2] == an academic policy. [1] allows both a resource or collaboration input, [2] is a restricted level of rights and only allows collaboration queries
- The #user_Policy property is derived using a Sparql query using the #user_Id
- #res_Type [1, 2] == a type of resource input. [1] relates to a file input while [2] relates to a link input. Different context properties are used for a file and a link.

Generate Random Test Sets

Random Test Set: 1

property	test value [1]/[0]	true / false
coSub	1	true
modSub	1	true
acadQual	0	false
vocQual	1	true
acadExp	1	true
vocExp	1	true
acadInt	0	false
vocInt	1	true
resFHEQ	1	true

Random Test Set: 6

property	test value [1]/[0]	true / false
coSub	1	true
modSub	0	true
acadQual	1	false
vocQual	1	true
acadExp	1	true
vocExp	0	true
acadInt	0	false
vocInt	0	true
resFHEQ	1	true

Test set all True Literal Values

```
run:
Begin Program Run
 INFO [main] (SetupTDB.java:678) - Statistics-based BGP optimizer
Error: null
InData.inResType: 2
*loginAction: 2
cp_R4: 2
*begin context matching
*eval_CoSub(iCS): cpg1 (oCS): cpg1
*(e): 1 *(W): 0.9 *(av): 0.9 *(sv): 0.9 *(mv): 0.9
*eval_MoSub(iMS): m1cpg1 (oMS): m1cpg1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 1.7 *(mv): 1.7
*eval_AcadQual(iAQ): aq1 (oAQ): aq1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 2.5 *(mv): 2.5
*eval_VocQual(iVQ): vq1 (oVQ): vq1
*(e): 1 *(W): 0.5 *(av): 0.5 *(sv): 3.0 *(mv): 3.0
*eval_AcadExp(iAE): ae1 (oAE): ae1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 3.7 *(mv): 3.7
*eval_VocExp(iVE): ve1 (oVE): ve1
*(e): 1 *(W): 0.6 *(av): 0.6 *(sv): 4.3 *(mv): 4.3
*eval_AcadInt(iAI): ai1 (oAI): ai1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 5.0 *(mv): 5.0
*eval_VocInt(iVI): vi1 (oVI): vi1
*(e): 1 *(W): 0.4 *(av): 0.4 *(sv): 5.4 *(mv): 5.4
*eval_FHEQ(resFHEQ): 7 (outFHEQ): 7
*(e): 1 *(W): 1.0 *(av): 1.0 *(sv): 6.4 *(mv): 6.4
*getResultantValue (sv / mv): 1.0
*semanticMatch.outPrecision: GQ
*semanticMatch: 1.0
*semanticMatch (q): 0.01
*semanticMatch: HQ
*applyPrecision(semPrec)(outPrec): HQ : GQ
*CM result: true
BUILD SUCCESSFUL (total time: 3 seconds)
```

Test set all False Literal Values

```
run:
Begin Program Run
 INFO [main] (SetupTDB.java:678) - Statistics-based BGP optimizer
Error: null
InData.inResType: 2
*loginAction: 2
cp_R4: 2
*begin context matching
*eval_CoSub(iCS): cpg2 (oCS): cpg1
*(e): 0 *(W): 0.9 *(av): 0.0 *(sv): 0.0 *(mv): 0.9
*eval_MoSub(iMS): m1cpg2 (oMS): m1cpg1
*(e): 0 *(W): 0.8 *(av): 0.0 *(sv): 0.0 *(mv): 1.7
*eval_AcadQual(iAQ): aq2 (oAQ): aq1
*(e): 0 *(W): 0.8 *(av): 0.0 *(sv): 0.0 *(mv): 2.5
*eval_VocQual(iVQ): vq2 (oVQ): vq1
*(e): 0 *(W): 0.5 *(av): 0.0 *(sv): 0.0 *(mv): 3.0
*eval_AcadExp(iAE): ae2 (oAE): ae1
*(e): 0 *(W): 0.7 *(av): 0.0 *(sv): 0.0 *(mv): 3.7
*eval_VocExp(iVE): ve2 (oVE): ve1
*(e): 0 *(W): 0.6 *(av): 0.0 *(sv): 0.0 *(mv): 4.3
*eval_AcadInt(iAI): ai2 (oAI): ai1
*(e): 0 *(W): 0.7 *(av): 0.0 *(sv): 0.0 *(mv): 5.0
*eval_VocInt(iVI): vi2 (oVI): vi1
*(e): 0 *(W): 0.4 *(av): 0.0 *(sv): 0.0 *(mv): 5.4
*eval_FHEQ(resFHEQ): 8 (outFHEQ): 7
*(e): 0 *(W): 1.0 *(av): 0.0 *(sv): 0.0 *(mv): 6.4
*getResultantValue (sv / mv): 0.0
*semanticMatch.outPrecision: GQ
*semanticMatch: 0.0
*semanticMatch (q): 0.01
*semanticMatch: LQ
*applyPrecision(semPrec)(outPrec): LQ : GQ
*CM result: false
BUILD SUCCESSFUL (total time: 3 seconds)
```

Random Test Set: 1

```
run:
INFO [main] (SetupTDB.java:678) - Statistics-based BGP optimizer
Error: null
InData.inResType: 2
*loginAction: 2
cp_R4: 2
*begin context matching
*eval_CoSub(iCS): cug1 (oCS): cug1
*(e): 1 *(W): 0.9 *(av): 0.9 *(sv): 0.9 *(mv): 0.9
*eval_MoSub(iMS): m1cug1 (oMS): m1cug1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 1.7 *(mv): 1.7
*eval_AcadQual(iAQ): aq2 (oAQ): aq1
*(e): 0 *(W): 0.8 *(av): 0.0 *(sv): 1.7 *(mv): 2.5
*eval_VocQual(iVQ): vq1 (oVQ): vq1
*(e): 1 *(W): 0.5 *(av): 0.5 *(sv): 2.2 *(mv): 3.0
*eval_AcadExp(iAE): ae1 (oAE): ae1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 2.9 *(mv): 3.7
*eval_VocExp(iVE): ve1 (oVE): ve1
*(e): 1 *(W): 0.6 *(av): 0.6 *(sv): 3.5 *(mv): 4.3
*eval_AcadInt(iAI): ai2 (oAI): ai1
*(e): 0 *(W): 0.7 *(av): 0.0 *(sv): 3.5 *(mv): 5.0
*eval_VocInt(iVI): vi1 (oVI): vi1
*(e): 1 *(W): 0.4 *(av): 0.4 *(sv): 3.9 *(mv): 5.4
*eval_FHEQ(resFHEQ): 6 (outFHEQ): 6
*(e): 1 *(W): 1.0 *(av): 1.0 *(sv): 4.9 *(mv): 6.4
*getResultantValue (sv / mv): 0.765625
*semanticMatch.outPrecision: GQ
*semanticMatch: 0.765625
*semanticMatch (q): 0.01
*semanticMatch: GQ
*applyPrecision(semPrec)(outPrec): GQ : GQ
*CM result: true
BUILD SUCCESSFUL (total time: 3 seconds)
```

Test Set (1/1)

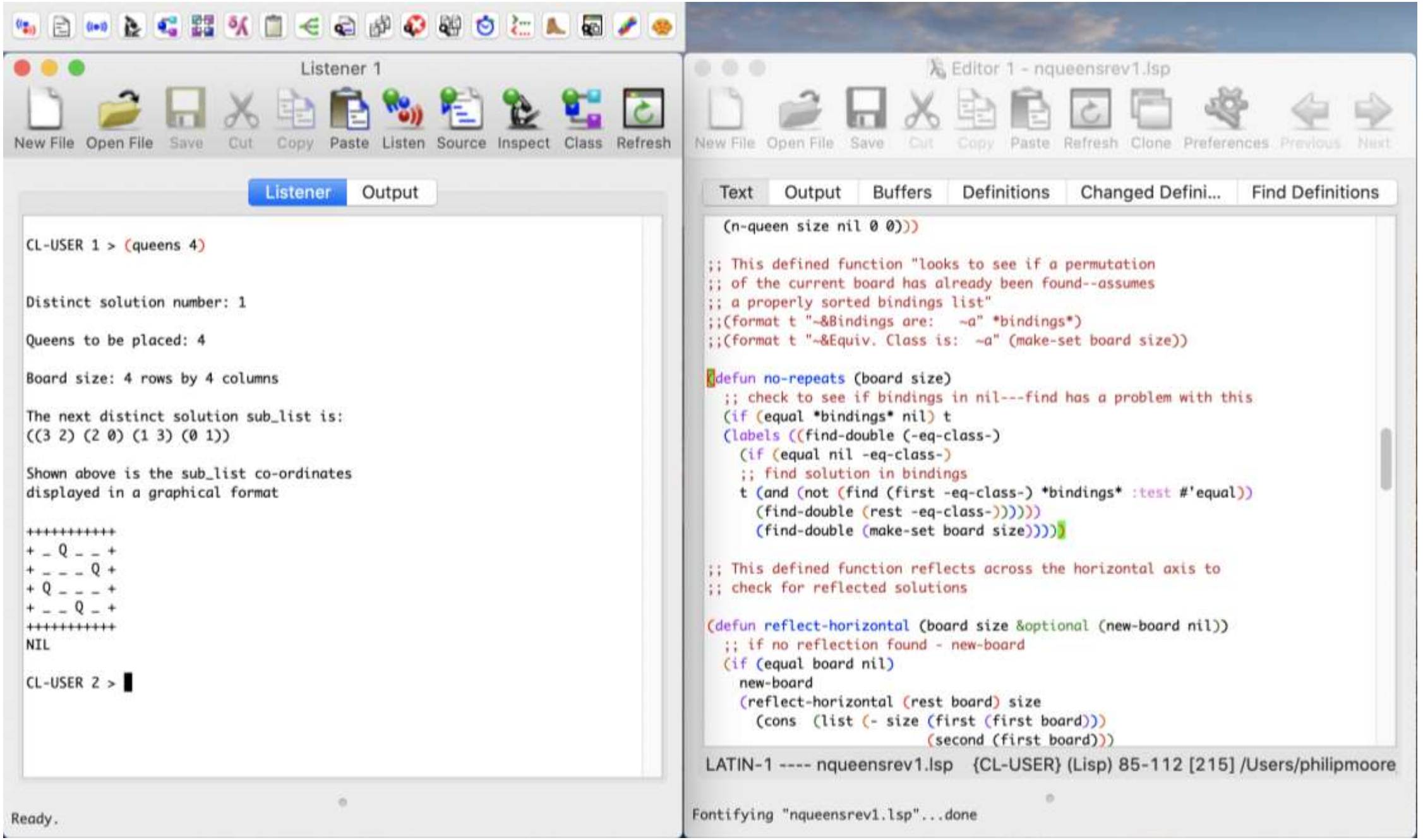
```
run:
Begin Program Run
 INFO [main] (SetupTDB.java:678) - Statistics-based BGP
optimizer
Error: null
InData.inResType: 2
*loginAction: 2
cp_R4: 2
*begin context matching
*eval_CoSub(iCS): cug1 (oCS): cug1
*(e): 1 *(W): 0.9 *(av): 0.9 *(sv): 0.9 *(mv): 0.9
*eval_MoSub(iMS): m1cug1 (oMS): m1cug1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 1.7 *(mv): 1.7
*eval_AcadQual(iAQ): aq1 (oAQ): aq1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 2.5 *(mv): 2.5
*eval_VocQual(iVQ): vq1 (oVQ): vq1
*(e): 1 *(W): 0.5 *(av): 0.5 *(sv): 3.0 *(mv): 3.0
*eval_AcadExp(iAE): ae1 (oAE): ae1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 3.7 *(mv): 3.7
*eval_VocExp(iVE): ve1 (oVE): ve1
*(e): 1 *(W): 0.6 *(av): 0.6 *(sv): 4.3 *(mv): 4.3
*eval_AcadInt(iAI): ai2 (oAI): ai1
*(e): 0 *(W): 0.7 *(av): 0.0 *(sv): 4.3 *(mv): 5.0
*eval_VocInt(iVI): vi2 (oVI): vi1
*(e): 0 *(W): 0.4 *(av): 0.0 *(sv): 4.3 *(mv): 5.4
*eval_FHEQ(resFHEQ): 8 (outFHEQ): 6
*(e): 0 *(W): 1.0 *(av): 0.0 *(sv): 4.3 *(mv): 6.4
*getResultantValue (sv / mv): 0.6718749999999999
*semanticMatch.outPrecision: GQ
*semanticMatch: 0.6718749999999999
*semanticMatch (q): 0.01
*semanticMatch: GQ
*applyPrecision(semPrec)(outPrec): GQ : GQ
*CM result: true
BUILD SUCCESSFUL (total time: 3 seconds)
```

Test Set (8/1/1)

```
run:
Begin Program Run
 INFO [main] (SetupTDB.java:678) - Statistics-based BGP
optimizer
Error: null
InData.inResType: 2
*loginAction: 2
cp_R4: 2
*begin context matching
*eval_CoSub(iCS): cug1 (oCS): cug1
*(e): 1 *(W): 0.9 *(av): 0.9 *(sv): 0.9 *(mv): 0.9
*eval_MoSub(iMS): m1cug1 (oMS): m1cug1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 1.7 *(mv): 1.7
*eval_AcadQual(iAQ): aq1 (oAQ): aq1
*(e): 1 *(W): 0.8 *(av): 0.8 *(sv): 2.5 *(mv): 2.5
*eval_VocQual(iVQ): vq1 (oVQ): vq1
*(e): 1 *(W): 0.5 *(av): 0.5 *(sv): 3.0 *(mv): 3.0
*eval_AcadExp(iAE): ae1 (oAE): ae1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 3.7 *(mv): 3.7
*eval_VocExp(iVE): ve1 (oVE): ve1
*(e): 1 *(W): 0.6 *(av): 0.6 *(sv): 4.3 *(mv): 4.3
*eval_AcadInt(iAI): ai1 (oAI): ai1
*(e): 1 *(W): 0.7 *(av): 0.7 *(sv): 5.0 *(mv): 5.0
*eval_VocInt(iVI): vi1 (oVI): vi1
*(e): 1 *(W): 0.4 *(av): 0.4 *(sv): 5.4 *(mv): 5.4
*eval_FHEQ(resFHEQ): 8 (outFHEQ): 6
*(e): 0 *(W): 1.0 *(av): 0.0 *(sv): 5.4 *(mv): 6.4
*getResultantValue (sv / mv): 0.84375
*semanticMatch.outPrecision: GQ
*semanticMatch: 0.84375
*semanticMatch (q): 0.01
*semanticMatch: HQ
*applyPrecision(semPrec)(outPrec): HQ : GQ
*CM result: true
BUILD SUCCESSFUL (total time: 2 seconds)
```

A Lisp program

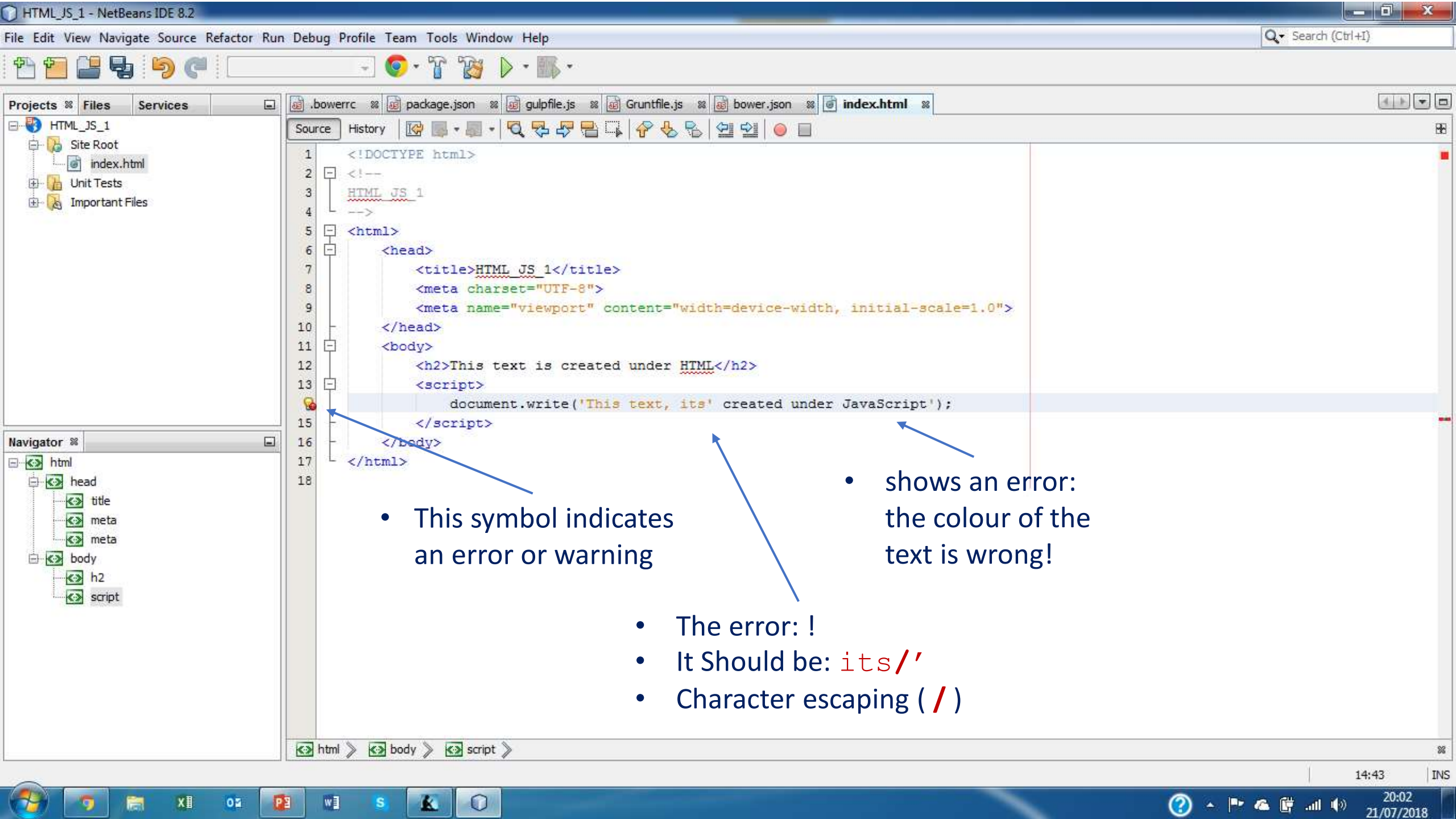
- The following slide shows a Lisp ‘8 queens’ program
- From the program code we can see:
  - The working program code
  - The extensive comments explaining: the purpose of the code, how specific functions operate, and and the output achieved
  - The output is shown for (queens 4) with the explanatory text
- The comments (in Lisp we use (;) are essential to
  - Provide an explanation of the code to remind the programmer (me!) and others! how the program works


Listener 1
New File
Open File
Save
Cut
Copy
Paste
Listen
Source
Inspect
Class
Refresh
Listener
Output
CL-USER 1 > (queens 4)
Distinct solution number: 1
Queens to be placed: 4
Board size: 4 rows by 4 columns
The next distinct solution sub_list is:
((3 2) (2 0) (1 3) (0 1))
Shown above is the sub_list co-ordinates
displayed in a graphical format
++++++++++
+ _ Q _ _ +
+ _ _ _ Q +
+ Q _ _ _ +
+ _ _ Q _ +
++++++++++
NIL
CL-USER 2 >
Ready.
Editor 1 - nqueensrev1.lsp
New File
Open File
Save
Cut
Copy
Paste
Refresh
Clone
Preferences
Previous
Next
Text
Output
Buffers
Definitions
Changed Defini...
Find Definitions
(n-queen size nil 0 0)))
;; This defined function "looks to see if a permutation
;; of the current board has already been found--assumes
;; a properly sorted bindings list"
;;(format t "~&Bindings are: ~a" *bindings*)
;;(format t "~&Equiv. Class is: ~a" (make-set board size))
(defun no-repeats (board size)
;; check to see if bindings in nil---find has a problem with this
(if (equal *bindings* nil) t
(labels ((find-double (-eq-class-)
(if (equal nil -eq-class-)
;; find solution in bindings
t (and (not (find (first -eq-class-) *bindings* :test #'equal))
(find-double (rest -eq-class-))))))
(find-double (make-set board size)))))
;; This defined function reflects across the horizontal axis to
;; check for reflected solutions
(defun reflect-horizontal (board size &optional (new-board nil))
;; if no reflection found - new-board
(if (equal board nil)
new-board
(reflect-horizontal (rest board) size
(cons (list (- size (first (first board)))
(second (first board)))
LATIN-1 ---- nqueensrev1.lsp {CL-USER} (Lisp) 85-112 [215] /Users/philipmoore
Fontifying "nqueensrev1.lsp"...done

Programming Errors

Types of Errors

- There are two types of error (of bug) in any computer program (including JavaScript):
  - *Syntax* errors (mistakes in writing the program code
  - *Logic* errors (where the output is not what is expected or required)
- *Syntax* errors WILL be shown in NetBeans (see the following slide)
  - Identifying such errors is relatively simple (the program will 'crash')
- *Logic* errors will NOT be shown in NetBeans (or any other tool)
  - Identifying such errors can be very difficult and requires extensive testing
  - An effective design and testing plan is required to avoid such errors and find and correct them



- This symbol indicates an error or warning
- shows an error: the colour of the text is wrong!
- The error: !
- It Should be: `its/'`
- Character escaping (/)

Program Design

Program Design (1)

- The design process for a computer program is as follows:
- Identify and document the requirements specification
  - This will set out the required *input* (data) and the required *output* (results)
- Create a block layout showing the program algorithm structure)
  - This will set out the sequence of operations (sequential / selection / iteration)
  - Write pseudo program code describing the algorithm
  - I will have more to say on these topics in another tutorial
- Creating a block layout and pseudo program code will be introduced in another tutorial addressing design principles

Program Design (2)

- Plot the path of the data through the program
- Design a test plan to:
  - Test and verify if the data values throughout the program are correct
  - Test to check if the result is consistent with the required output
- Testing may use both black-box and white-box testing:
  - Black-box testing: a method where the inner workings of the program are not known (only the input and the output)
  - White-box testing: a method where the data-flows within the program are checked and mapped to the output