INFO 151
Web Systems and Services

Week 5 (T2)

Dr Philip Moore

Dr Zhili Zhao

Overview

- In this tutorial we will introduce:
  - JavaScript arrays including:
    - What is an array?
    - The nature of arrays and objects in JavaScript with associative arrays
    - Creating and working with arrays, array elements, and array methods
    - Merging (concatenating) two (or more) arrays into a single array
    - Array iteration (looping through arrays)

Sources of Resources

- The sources of information and resources for JavaScript may be found at:
  - The w3schools.com web-site
    - https://www.quanzhanketang.com/
  - The recommended course text book:
    - Sams Teach Yourself PHP, MySQL & JavaScript All in One SIXTH EDITION

What is an Array?

What is an Array?

- In computer science an array is a collection of same type data items that can be selected by indices computed at run-time, including:
  - Array *data structure*: an arrangement of items at equally spaced addresses in computer memory
  - Array *data type*: used in a programming language to specify a variable that can be indexed
  - *Associative array*: an abstract data structure model composed of key-value pairs, often implemented as a hash table or search tree
- Arrays can be very fast data structures:
  - High-level programming languages (e.g., Java) implement arrays and array methods

Array Variables

- In computer programming:
  - In general: a *variable* is a *memory location* that can store a *value*
  - The value held in memory can be changed
  - Each variable can only hold one item of data
- As we have seen in high-level programming languages (e.g., Java):
  - Variables are generally *strongly typed*
  - In such languages: array elements must be of the *same data type*
- As we have seen in JavaScript:
  - Variables are *untyped*
  - In an array in JavaScript elements can be of *any datatype*

JavaScript Methods

- In considering JavaScript arrays we have introduced
  - JavaScript *operators* / *operands* / *properties*/ *methods* / *functions*
- In JavaScript operators, methods, and functions can be used to manipulate *arrays* and the data held in *arrays*
- Comprehensive details of the available operators, methods, and functions can be found from the course resources
- The following slides show JavaScript methods with worked examples showing the program code and the results

JavaScript Arrays and Datatypes

- In JavaScript *variables* and *arrays* are *NOT* typed
- In the following example we can see that the elements of a single JavaScript array can hold different data types:

```
myArray[0] = Date.now;        //a method
myArray[1] = myFunction;      //a function
myArray[2] = myCars;          //an array
myArray[3] = "Philip";        //a string
myArray[4] = 9;               //a number
```

- A JavaScript arrays and elements can hold:
  - *Primitive* data types, *functions*, and *objects*
  - Any *legal* datatype can be held in array elements

JavaScript Arrays

- While we refer to *arrays* JavaScript does not incorporate arrays
  - For example: in JavaScript *variable* and *arrays* are *NOT* typed
- In JavaScript
  - An array is a special type of **`object`**
  - JavaScript automatically creates an *associative array* for each *object*
  - There are differences between JavaScript and PHP!
- When investigating arrays
  - The **`typeof`** operator in JavaScript returns **`object`** for an array

JavaScript Indexing

- It is important to note
  - While *arrays* are *objects* there are differences between JavaScript arrays and JavaScript objects
- The differences Between Arrays and Objects are:
  - JavaScript *arrays* use *numbered indexes*
  - JavaScript objects use *named indexes*
- Arrays are a special kind of objects with numbered indexes
- The following slide shows examples of numbered and named indexes

JavaScript Indexing

- In JavaScript *arrays* are *objects*
  - A specialized type of object with a length property and array methods
  - The array methods are useful when working with array elements
  - Arbitrary data values are associated with arbitrary names

- Named index (an *object*)

```
myArray["x"] = 1;
myArray["y"] = 2;
myArray["y"] = "Philip";
```

- Numbered index (an *array*)

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
myArray[3] = "Philip";
myArray[4] = 9;
```

JavaScript Arrays and Named Indexes

- Many high-level programming languages support arrays with named indexes
  - Arrays with *named* indexes are called *associative* arrays (or *hashes*)
- JavaScript does not support arrays with *named* indexes
  - In JavaScript arrays always use *numbered* indexes
- If *named* indexing is used JavaScript will *redefine* the array as a standard *object*
  - The result will be that some array methods and properties will produce incorrect results

```
var person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
var x = person.length;          (person.length will return 0)
var y = person[0];              (person.length will return undefined)
```

JavaScript Indexing

- In designing a JavaScript program
  - There is a choice to be made
    - When to Use Arrays
    - When to use Objects
- The factors to consider are:
  - JavaScript does not support associative arrays
  - You should use *objects* when you want the *element names* to be *strings* (text)
  - You should use *arrays* when you want the *element names* to be *numbers*

Why use an Object or an Array?

- To choose between an *object* and an *array*
  - We need to identify the purpose of each structure
  - JavaScript arrays model the way books store information
  - Objects model the way that newspapers store information
- *Arrays* are used when a defined order (of information) is the most important factor for organizing the information
  - For example: in a book which has a chapter structure
- *Objects* are used when information is organize based on data labels
  - For example: newspaper pages may be read in a random order

Identifying a JavaScript Array Object

Identifying an Array Object

- In a JavaScript program is often necessary to test the type of object
- The test uses the **`typeof`** JavaScript operator
- Unfortunately there is a problem
  - The **`typeof`** JavaScript operator returns **`object`**
- For example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
typeof fruits;
```

  - Returns **`object`**

The Problem

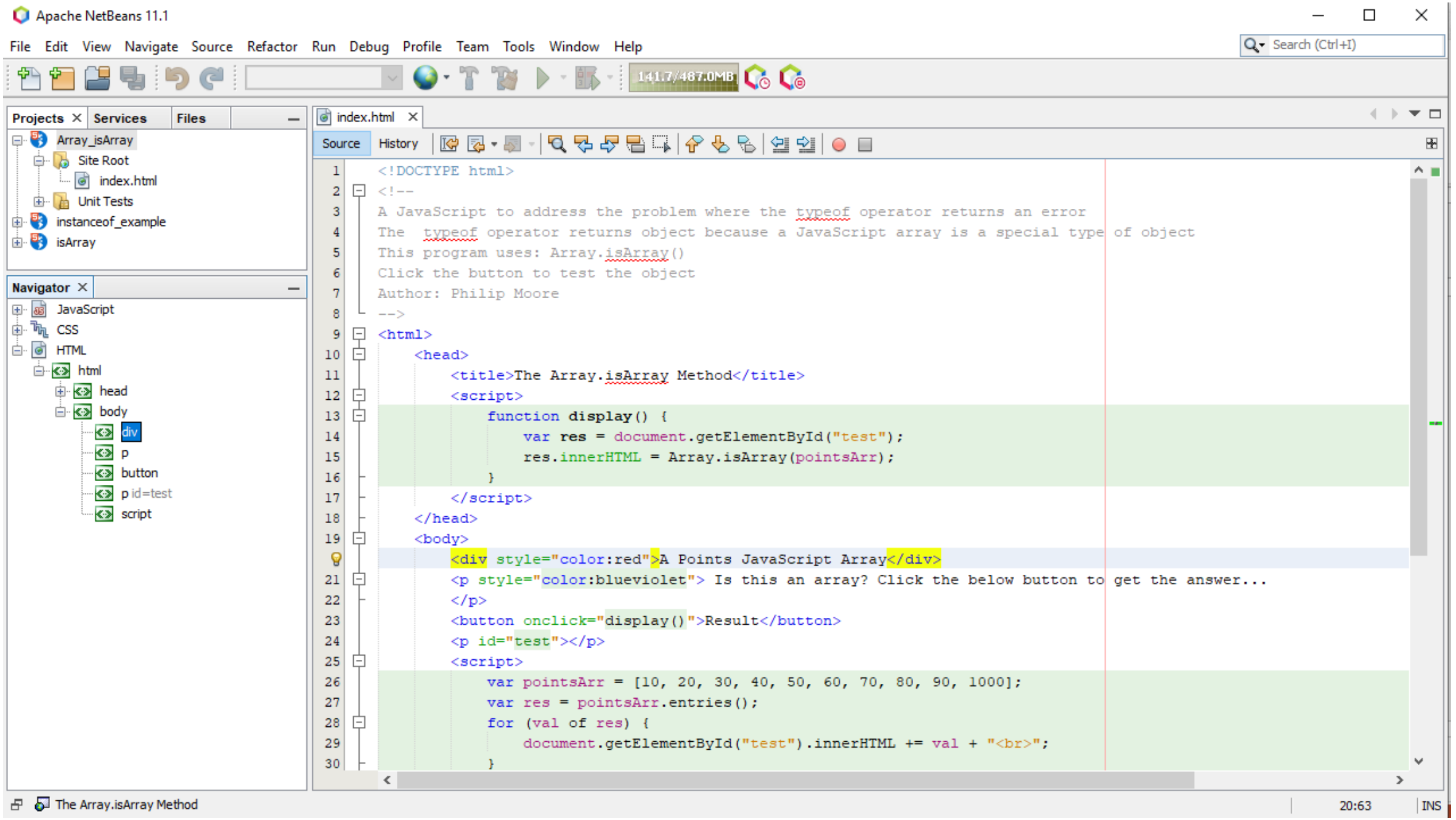
- The problem is created in JavaScript because:
  - The **`typeof`** operator returns **`object`** because a JavaScript *array* is a special type of **`object`**
- There are three potential solutions
  - The ECMAScript 5 standard method: **`Array.isArray()`**
  - Use of a user defined function: create an **`isArray()`**
  - To use the **`instanceof`** operator
- Each of the potential solutions has benefits and problems as shown in the following slides with worked examples

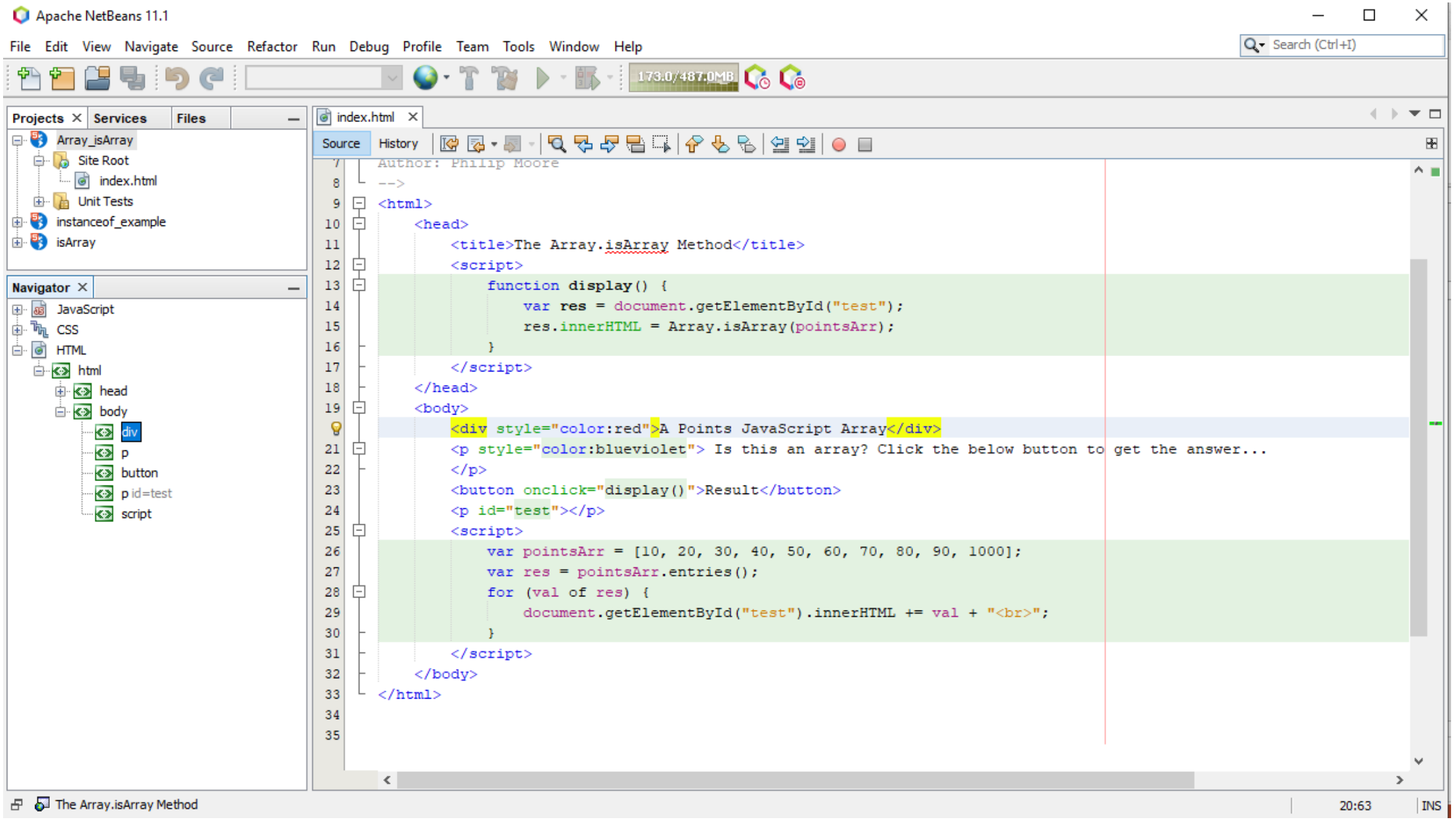
Solution One

- The ECMAScript 5 standard defines a new method:

```
Array.isArray()
Array.isArray(fruits); //returns true
```

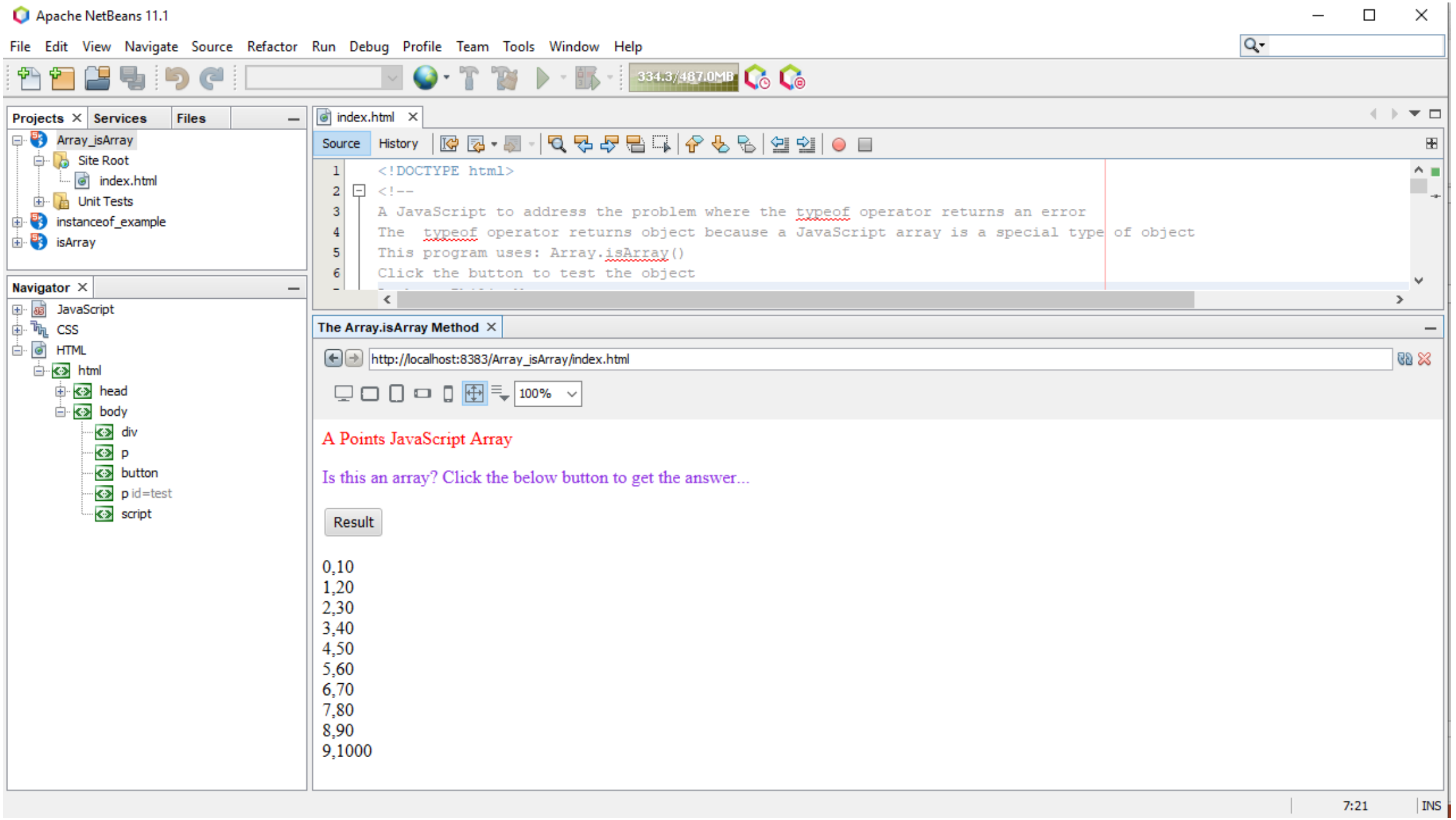
- There is however a problem with this solution
  - The ECMAScript 5 standard is not supported in older browsers
  - As older browsers are replaced this will become less of a problem
- The following slides implement this solution in the NetBeans embedded web kit and Microsoft Edge browser

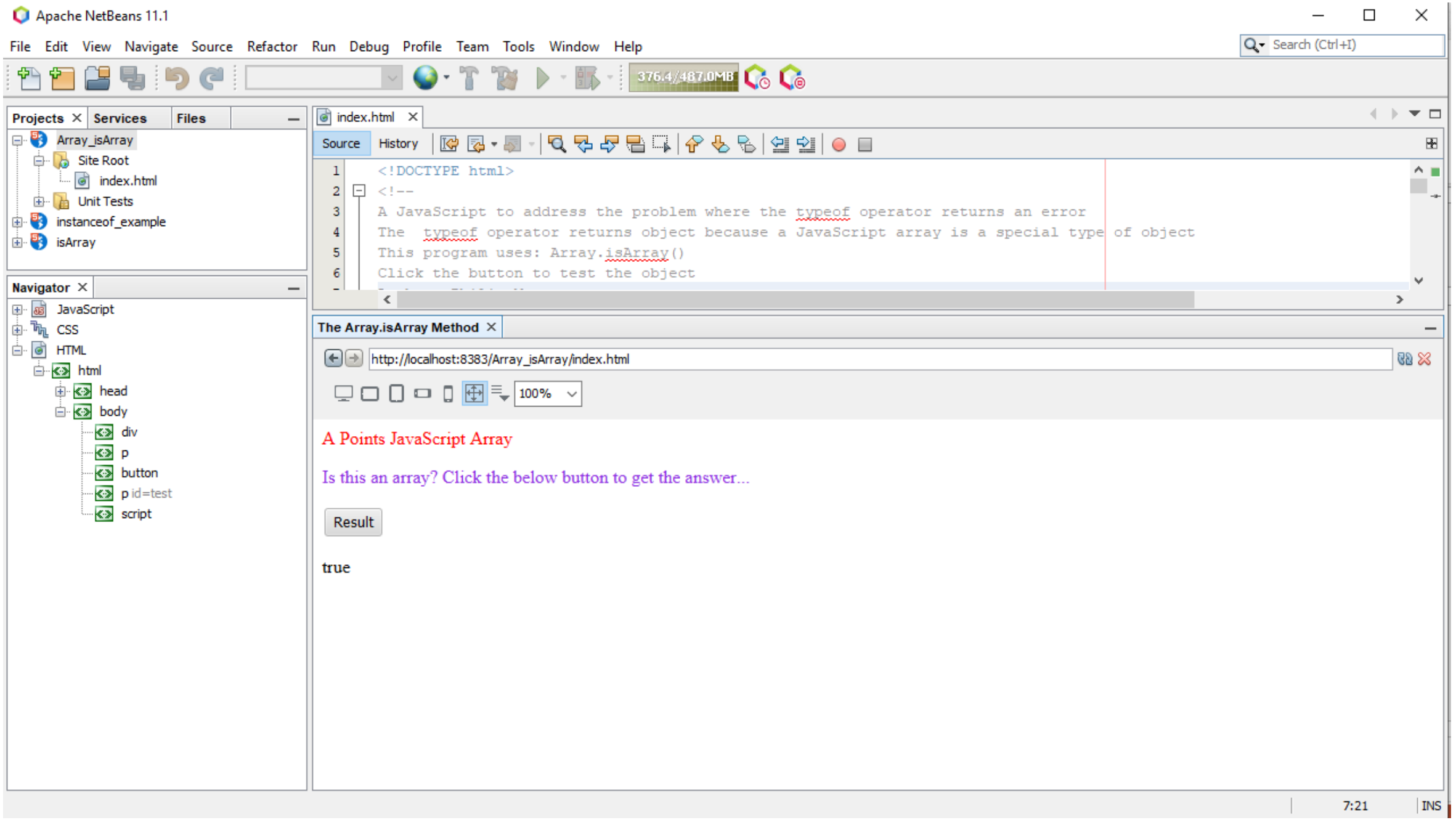

Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
141.7/487.0MB
Projects
Services
Files
Array_isArray
Site Root
index.html
Unit Tests
instanceof_example
isArray
Navigator
JavaScript
CSS
HTML
html
head
body
div
p
button
p id=test
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript to address the problem where the typeof operator returns an error
4 The typeof operator returns object because a JavaScript array is a special type of object
5 This program uses: Array.isArray()
6 Click the button to test the object
7 Author: Philip Moore
8 -->
9 <html>
10 <head>
11 <title>The Array.isArray Method</title>
12 <script>
13 function display() {
14 var res = document.getElementById("test");
15 res.innerHTML = Array.isArray(pointsArr);
16 }
17 </script>
18 </head>
19 <body>
A Points JavaScript Array
21 Is this an array? Click the below button to get the answer...
22
23 <button onclick="display()">Result</button>
24
25 <script>
26 var pointsArr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 1000];
27 var res = pointsArr.entries();
28 for (val of res) {
29 document.getElementById("test").innerHTML += val + "
";
30 }
The Array.isArray Method
20:63
INS

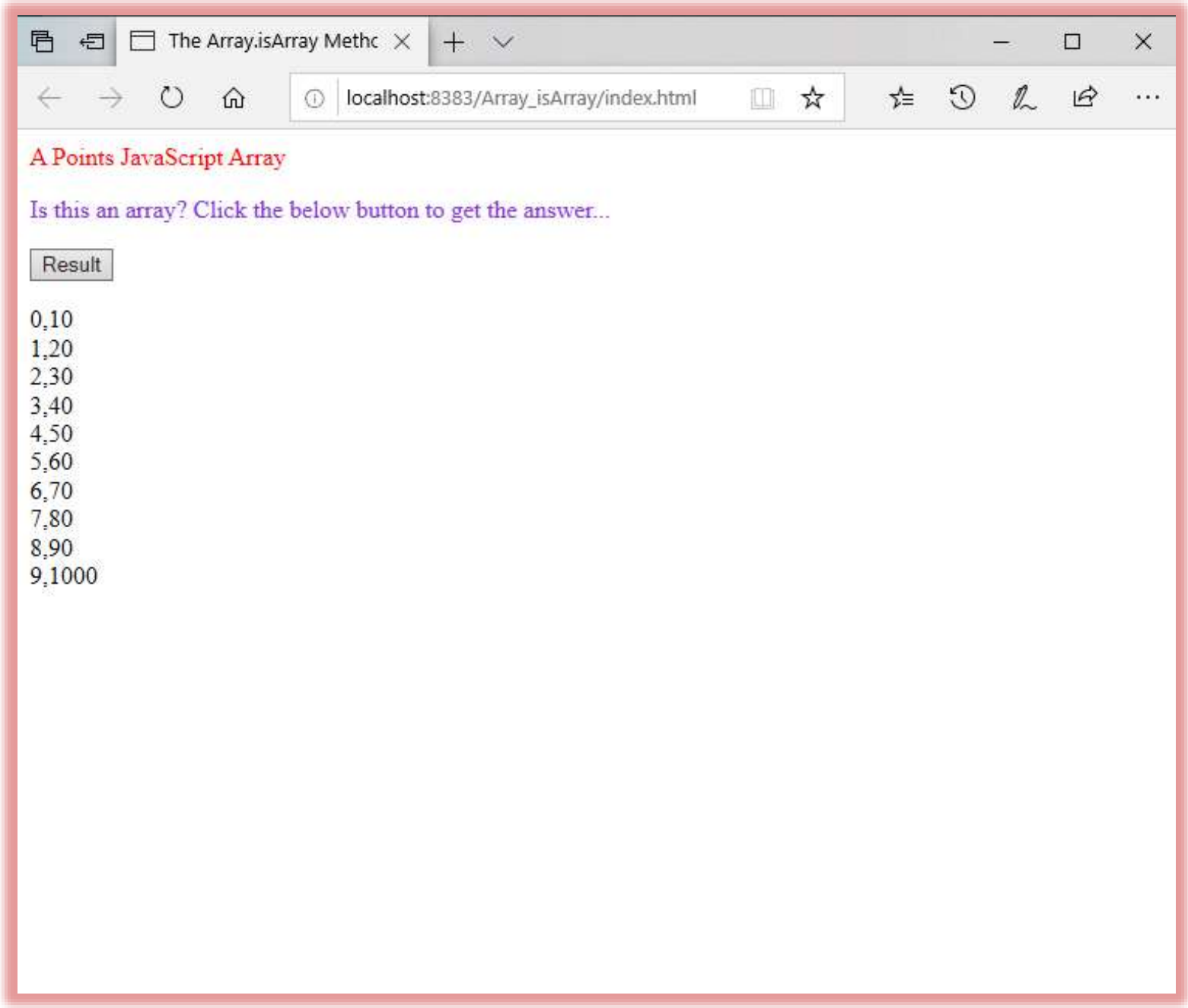

Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
173.0/487.0MB
Projects
Services
Files
Array_isArray
Site Root
index.html
Unit Tests
instanceof_example
isArray
Navigator
JavaScript
CSS
HTML
html
head
body
div
p
button
p id=test
script
index.html
Source
History
Author: Philip Moore
-->
<html>
<head>
<title>The Array.isArray Method</title>
<script>
function display() {
var res = document.getElementById("test");
res.innerHTML = Array.isArray(pointsArr);
}
</script>
</head>
<body>
A Points JavaScript Array
 Is this an array? Click the below button to get the answer...

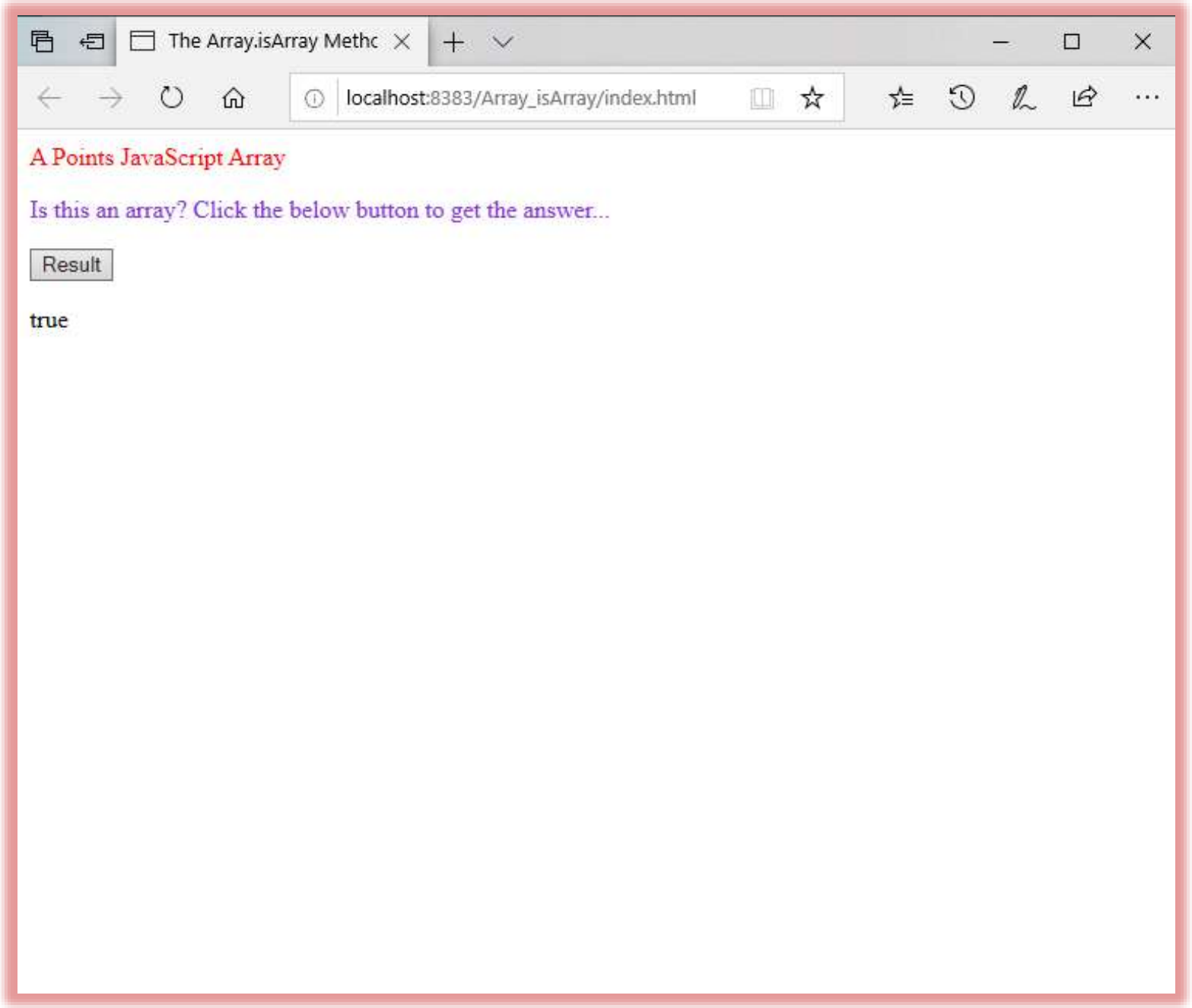
<button onclick="display()">Result</button>

<script>
var pointsArr = [10, 20, 30, 40, 50, 60, 70, 80, 90, 1000];
var res = pointsArr.entries();
for (val of res) {
document.getElementById("test").innerHTML += val + "
";
}
</script>
</body>
</html>
The Array.isArray Method
20:63
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
334.3/487.0MB
Projects
Services
Files
Array_isArray
Site Root
index.html
Unit Tests
instanceof_example
isArray
Navigator
JavaScript
CSS
HTML
html
head
body
div
p
button
p id=test
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript to address the problem where the typeof operator returns an error
4 The typeof operator returns object because a JavaScript array is a special type of object
5 This program uses: Array.isArray()
6 Click the button to test the object
The Array.isArray Method
http://localhost:8383/Array_isArray/index.html
100%
A Points JavaScript Array
Is this an array? Click the below button to get the answer...
Result
0,10
1,20
2,30
3,40
4,50
5,60
6,70
7,80
8,90
9,1000
7:21
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
376.4/487.0MB
Projects
Services
Files
Array_isArray
Site Root
index.html
Unit Tests
instanceof_example
isArray
Navigator
JavaScript
CSS
HTML
html
head
body
div
p
button
p id=test
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript to address the problem where the typeof operator returns an error
4 The typeof operator returns object because a JavaScript array is a special type of object
5 This program uses: Array.isArray()
6 Click the button to test the object
The Array.isArray Method
http://localhost:8383/Array_isArray/index.html
100%
A Points JavaScript Array
Is this an array? Click the below button to get the answer...
Result
true
7:21
INS


The Array.isArray Methc
localhost:8383/Array_isArray/index.html
A Points JavaScript Array
Is this an array? Click the below button to get the answer...
Result
0,10
1,20
2,30
3,40
4,50
5,60
6,70
7,80
8,90
9,1000

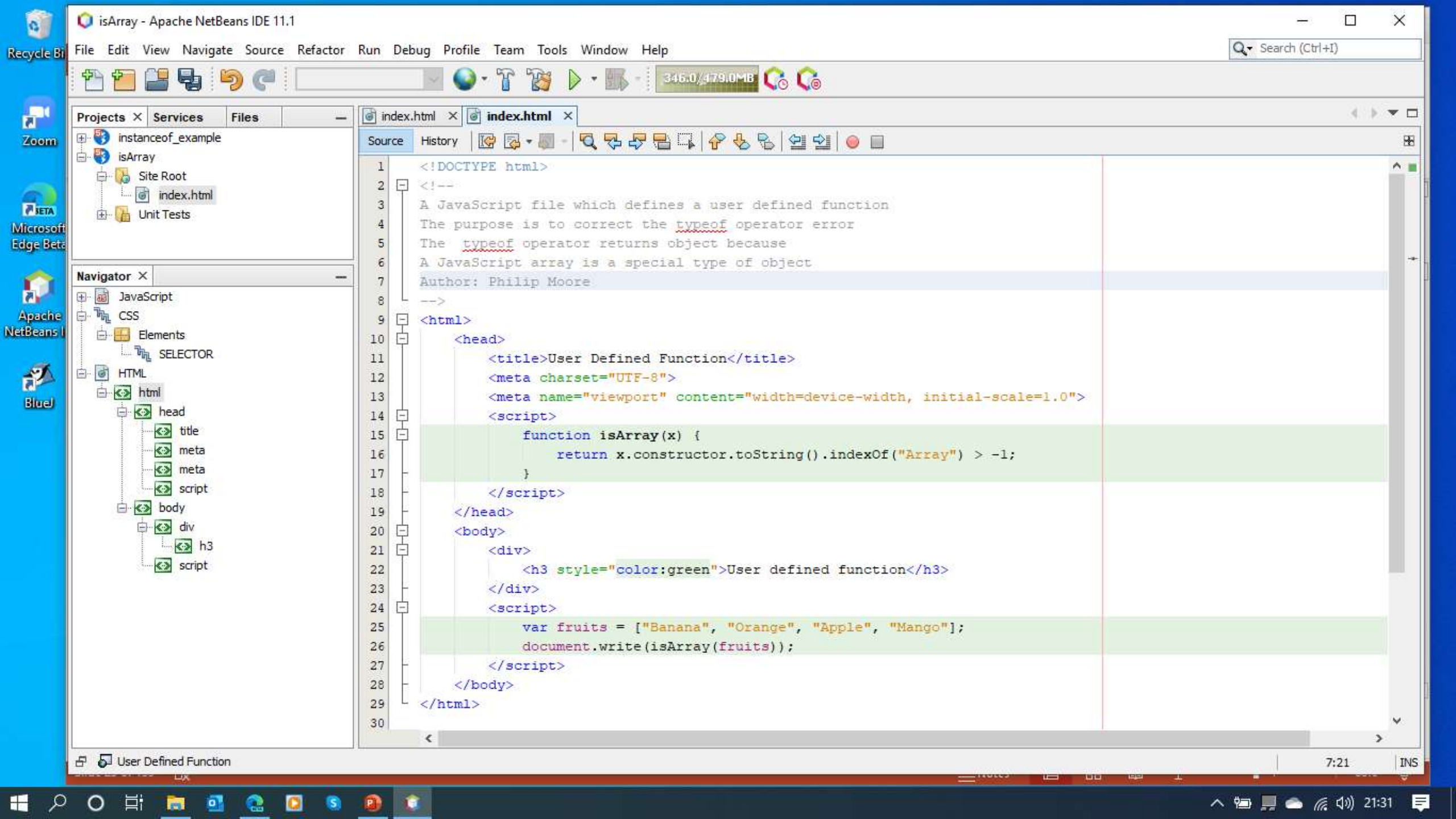

The Array.isArray Methc
localhost:8383/Array_isArray/index.html
A Points JavaScript Array
Is this an array? Click the below button to get the answer...
Result
true

Solution Two

- Create an **`isArray()`** user defined function as follows:

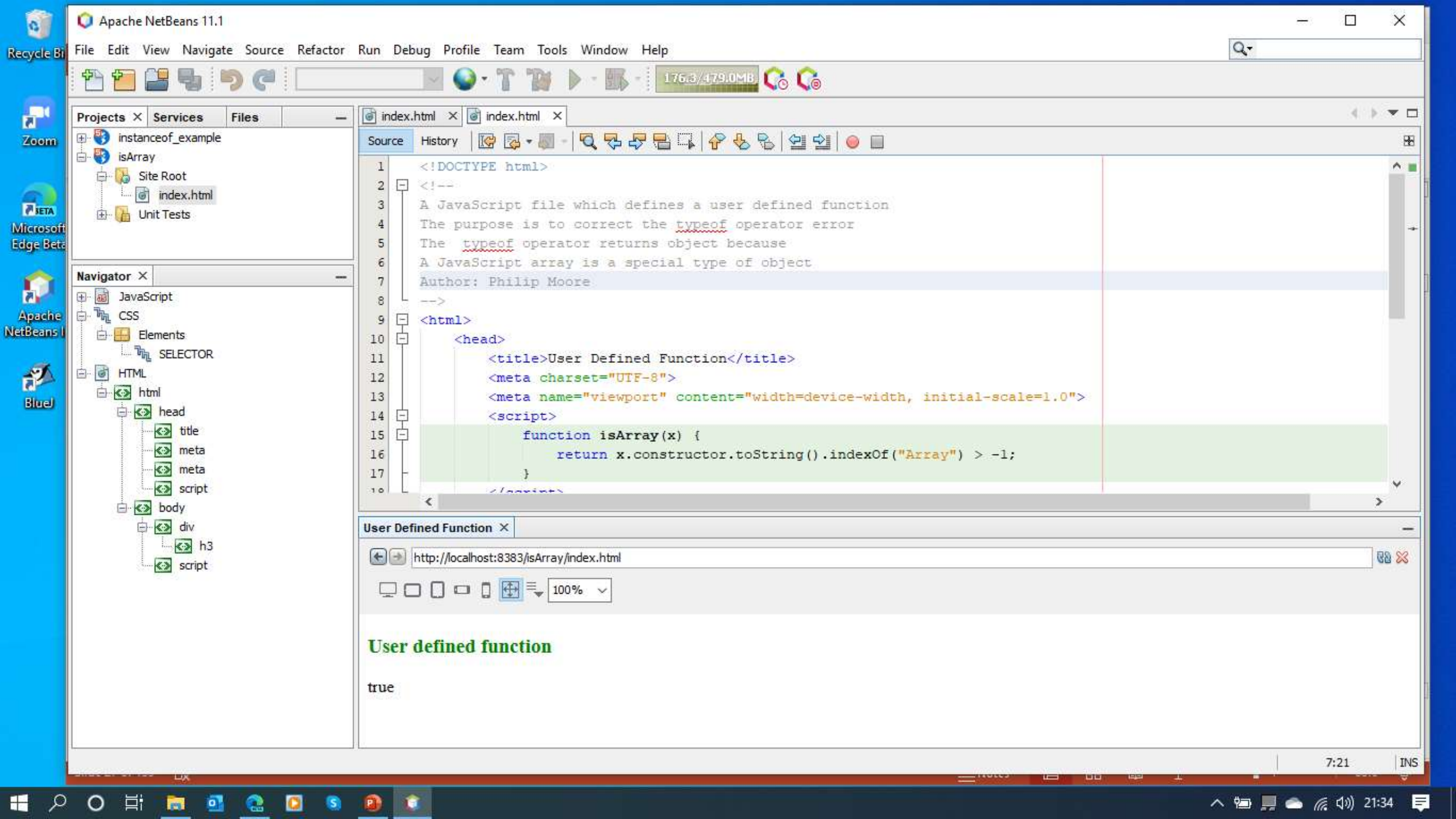
```
function isArray(x) {
    return x.constructor.toString().indexOf("Array") > -1;
}
```

- The user defined function above always returns **`true`** if the argument is an array
- Recall the concept of a **`function prototype`**
  - The user defined function returns **`true`** if the **`object prototype`** contains the word *Array*


isArray - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
346.0/479.0MB
Projects Services Files
instanceof_example
isArray
Site Root
index.html
Unit Tests
Navigator
JavaScript
CSS
Elements
SELECTOR
HTML
html
head
title
meta
meta
script
body
div
h3
script
index.html
index.html
Source History
<!DOCTYPE html>
<!--
A JavaScript file which defines a user defined function
The purpose is to correct the typeof operator error
The typeof operator returns object because
A JavaScript array is a special type of object
Author: Philip Moore
-->
<html>
<head>
<title>User Defined Function</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script>
function isArray(x) {
return x.constructor.toString().indexOf("Array") > -1;
}
</script>
</head>
<body>

<h3 style="color:green">User defined function</h3>

<script>
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(isArray(fruits));
</script>
</body>
</html>
User Defined Function
7:21
INS
21:31

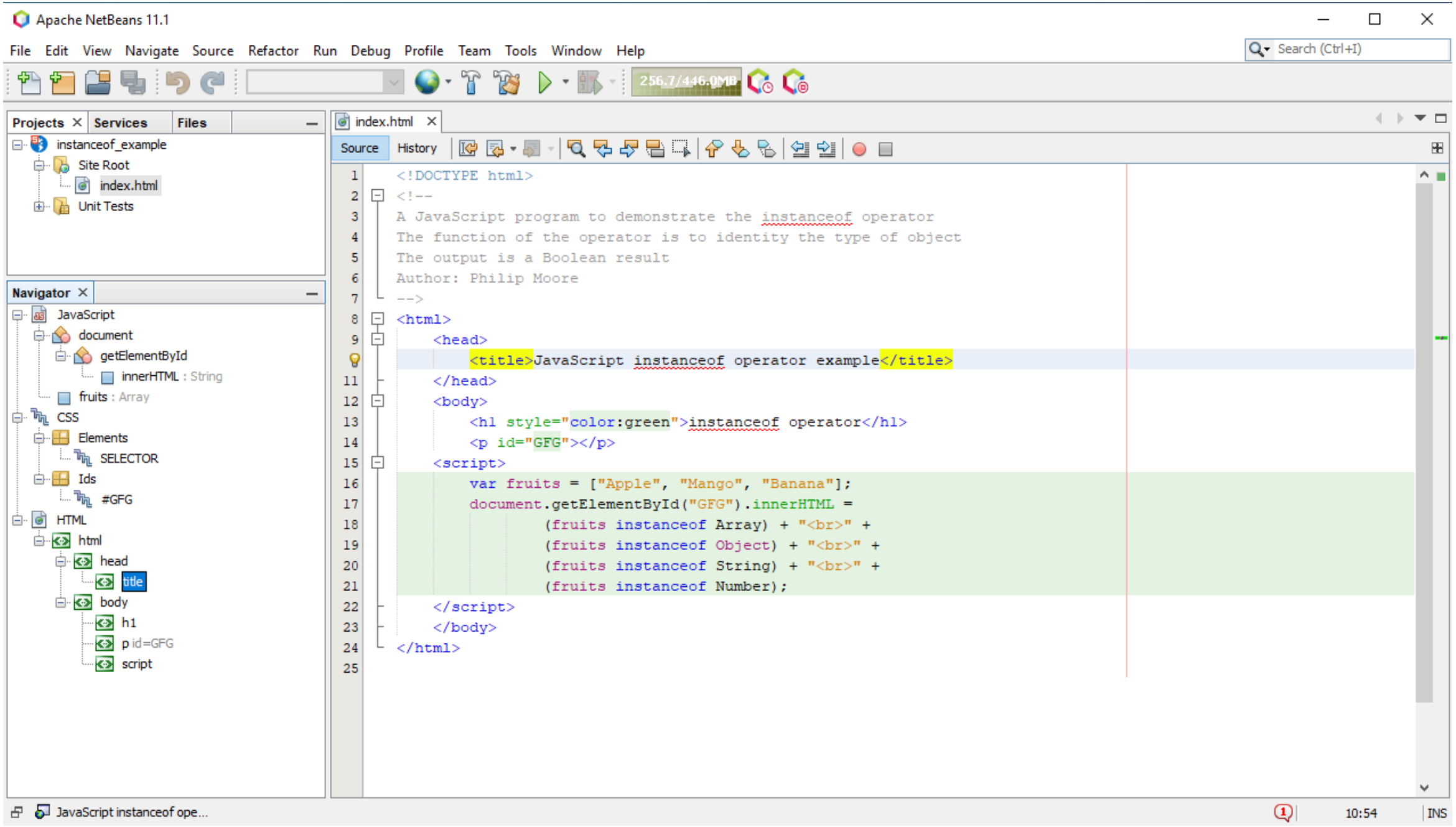

Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
instanceof_example
isArray
Site Root
index.html
Unit Tests
Navigator
JavaScript
CSS
Elements
SELECTOR
HTML
html
head
title
meta
meta
script
body
div
h3
script
index.html
index.html
Source History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript file which defines a user defined function
4 The purpose is to correct the typeof operator error
5 The typeof operator returns object because
6 A JavaScript array is a special type of object
7 Author: Philip Moore
8 -->
9 <html>
10 <head>
11 <title>User Defined Function</title>
12 <meta charset="UTF-8">
13 <meta name="viewport" content="width=device-width, initial-scale=1.0">
14 <script>
15 function isArray(x) {
16 return x.constructor.toString().indexOf("Array") > -1;
17 }
User Defined Function
http://localhost:8383/isArray/index.html
100%
User defined function
true
7:21
INS
21:34

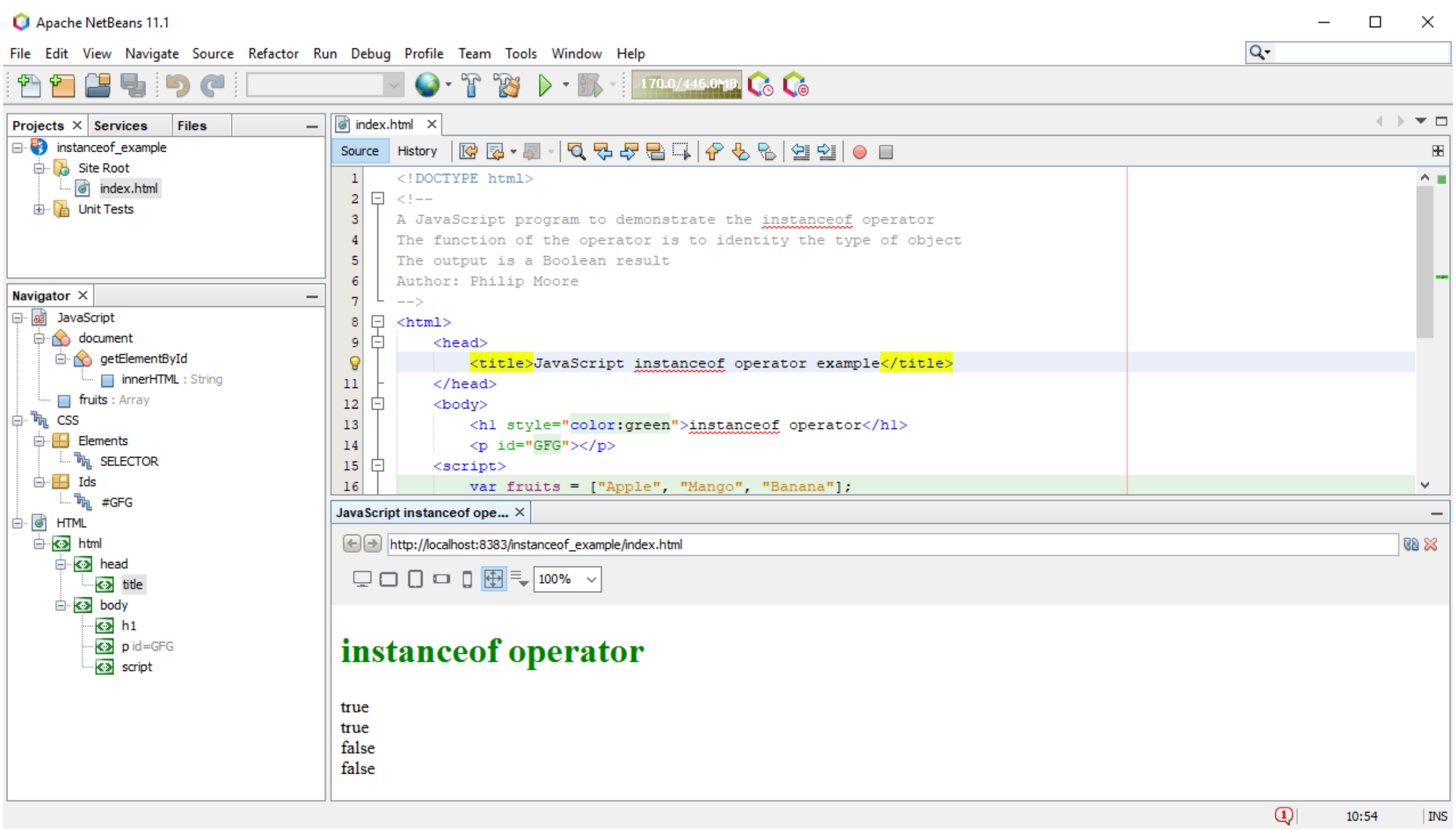
Solution Three

- And alternative solution is
  - To use the **`instanceof`** operator
  - the operator returns **`true`** if an object is created by a constructor
- For example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits instanceof Array;
```

- The **`instanceof`** operator returns **`true`**
- For an worked example see the following slides


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
256.7/446.0MB
Projects
Services
Files
instanceof_example
Site Root
index.html
Unit Tests
Navigator
JavaScript
document
getElementById
innerHTML : String
fruits : Array
CSS
Elements
SELECTOR
Ids
#GFG
HTML
html
head
title
body
h1
p id=GFG
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript program to demonstrate the instanceof operator
4 The function of the operator is to identity the type of object
5 The output is a Boolean result
6 Author: Philip Moore
7 -->
8 <html>
9 <head>
11 </head>
12 <body>
13 <h1 style="color:green">instanceof operator</h1>
14
15 <script>
16 var fruits = ["Apple", "Mango", "Banana"];
17 document.getElementById("GFG").innerHTML =
18 (fruits instanceof Array) + "
" +
19 (fruits instanceof Object) + "
" +
20 (fruits instanceof String) + "
" +
21 (fruits instanceof Number);
22 </script>
23 </body>
24 </html>
25
JavaScript instanceof ope...
10:54
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
instanceof_example
Site Root
index.html
Unit Tests
Navigator
JavaScript
document
getElementById
innerHTML : String
fruits : Array
CSS
Elements
SELECTOR
Ids
#GFG
HTML
html
head
title
body
h1
p id=GFG
script
index.html
Source History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript program to demonstrate the instanceof operator
4 The function of the operator is to identity the type of object
5 The output is a Boolean result
6 Author: Philip Moore
7 -->
8 <html>
9 <head>
<title>JavaScript instanceof operator example</title>
11 </head>
12 <body>
13 <h1 style="color:green">instanceof operator</h1>
14
15 <script>
16 var fruits = ["Apple", "Mango", "Banana"];
JavaScript instanceof ope...
http://localhost:8383/instanceof_example/index.html
100%
instanceof operator
true
true
false
false
10:54
INS

Working with Arrays

Array Methods

- We must enable actions on arrays using array methods including:
  - *Accessing* arrays
  - *Changing* and *Updating* arrays
  - *Adding* to array elements
  - *Deleting* from array elements
  - *Sorting* arrays
  - *Concatenating* (merging) arrays
  - *Converting* array elements to *strings*
  - Running *methods* on arrays
- In the following slides we introduce some of these actions with worked examples
- We will complete the introduction in subsequent tutorials

The JavaScript `length` Property

JavaScript `length` Property

- The **`length`** property:
  - Is used in many methods in JavaScript and general programming
  - For example: the following code snippet is used to test a **`for`** loop:

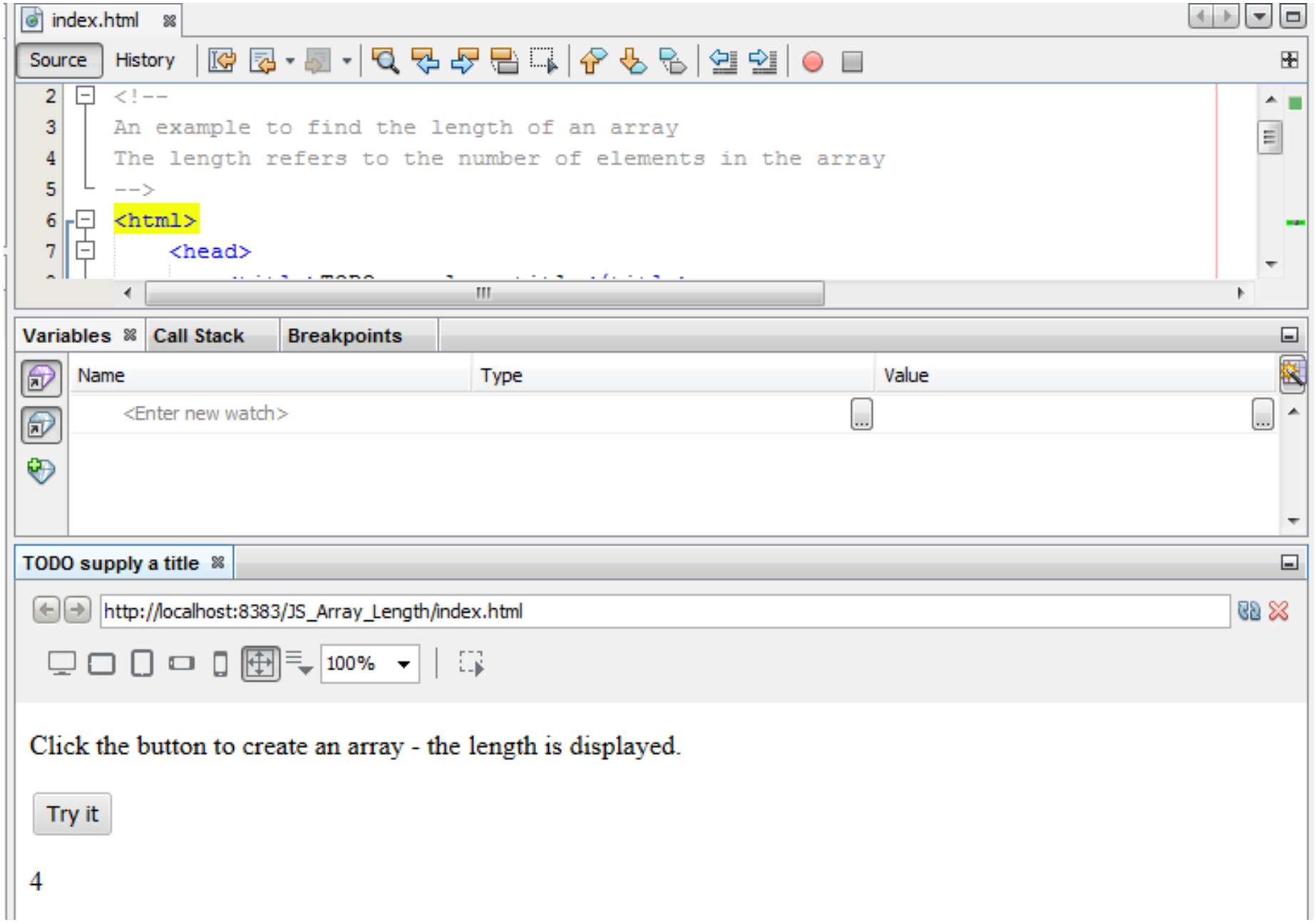
```
for (i = 0; i < fLen; i++) {
    text += "<li>" + fruits[i] + "</li>";
}
```

- The **`length`** property of an array returns the length of an array (the number of array elements)

index.html

Source | History

```
<!--
An example to find the length of an array
The length refers to the number of elements in the array
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <p>Click the button to create an array - the  length is displayed.</p>
        <button onclick="arrayLength()">Try it</button>
        <p id="demo"></p>
        <script>
            function arrayLength() {
                var fruits = ["Banana", "Orange", "Apple", "Mango"];
                document.getElementById("demo").innerHTML = fruits.length;
            }
        </script>
    </body>
</html>
```


index.html
Source
History
2 <!--
3 An example to find the length of an array
4 The length refers to the number of elements in the array
5 -->
6 <html>
7 <head>
Variables
Call Stack
Breakpoints
Name
Type
Value
<Enter new watch>
TODO supply a title
http://localhost:8383/JS_Array_Length/index.html
100%
Click the button to create an array - the length is displayed.
Try it
4

Creating a JavaScript Array?

Creating Arrays (1)

- There are 2 methods to create a new array

```
var cars = ["Saab",  "Volvo",  "BMW"];
var cars = new Array("Saab", "Volvo", "BMW");
```

- The general recommendation:
  - Avoid `new Array()`
  - There is no need to use the JavaScript's built-in array constructor `new Array()`
  - Use `[ … ]` instead

Creating new Arrays (2)

- The following two different statements both create a new empty array named **points**:

```
var points = new Array(); //not recommended
var points = [];          // recommended
```

- The two different statements both create a new **points** array containing 6 numbers:

```
var points = new Array(40, 100, 1, 5, 25, 10);
var points = [40, 100, 1, 5, 25, 10];
```

The **new** keyword

- In a high-level programming language (such as Java) the new keyword is used to create a new object – the **`new`** keyword performs the same function in JavaScript
- However - In JavaScript the **`new`** keyword only complicates the code
  - It can produce some unexpected results such as:

```
var points = new Array(40, 100);
```

    - Creates an array with two elements (40 and 100)
  - Removing one of the elements?

```
var points = new Array(40);
```

    - Creates an array with 40 undefined elements !!!!!

Accessing Arrays

Accessing Arrays

- To *access* JavaScript arrays we may use *iteration*
  - To *iterate* over *array elements* we may use a **`for`** loop
- For example:

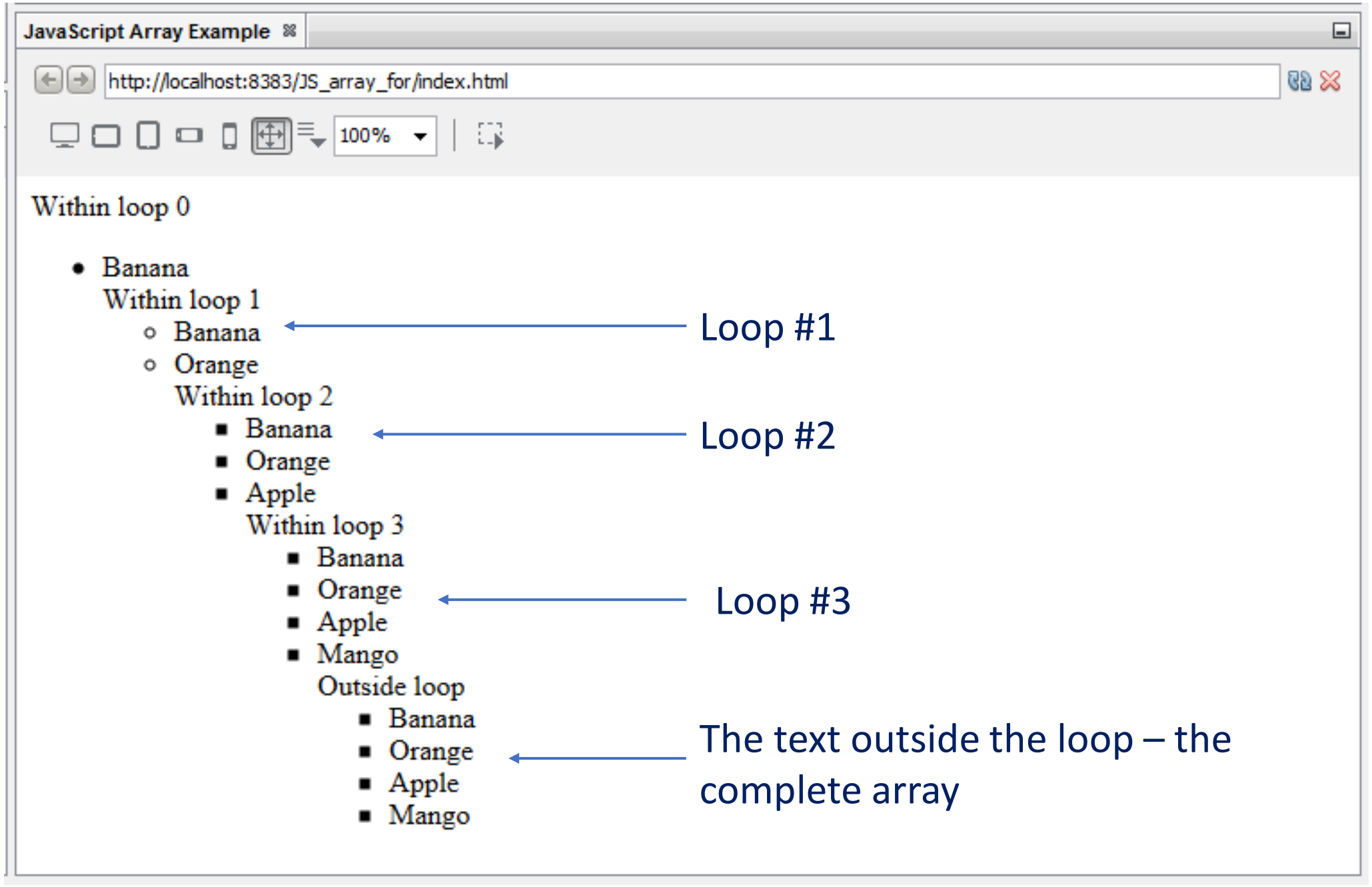
```
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length; //assign the method to the variable fLen
text = "<ul>"; //the list structure
for (i = 0; i < fLen; i++) {
      text += "<li>" + fruits[i] + "</li>";
}
```

- Note: the loop body { … } and the `i++` operator (shorthand for `i = i + 1`)

```
<!DOCTYPE html>
<!--
A JavaScript Array Example using a for loop
-->
<html>
    <head>
        <title>JavaScript Array Example</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>
            <script>
                var fruits, text, fLen, i;
                fruits = ["Banana", "Orange", "Apple", "Mango"];
                fLen = fruits.length;
                text = "<ul>";
                for (i = 0; i < fLen; i++) {
                    text += "<li>" + fruits[i] + "</li>";
                    document.write("Within loop " + i + " " + text);
                }
                document.write("Outside loop" + text);
            </script>

        </div>
    </body>
</html>

```


JavaScript Array Example
http://localhost:8383/JS_array_for/index.html
100%
Within loop 0
Banana
Within loop 1
Banana
Orange
Within loop 2
Banana
Orange
Apple
Within loop 3
Banana
Orange
Apple
Mango
Outside loop
Banana
Orange
Apple
Mango
Loop #1
Loop #2
Loop #3
The text outside the loop – the complete array

Merging Arrays

Merging (Concatenating) Arrays

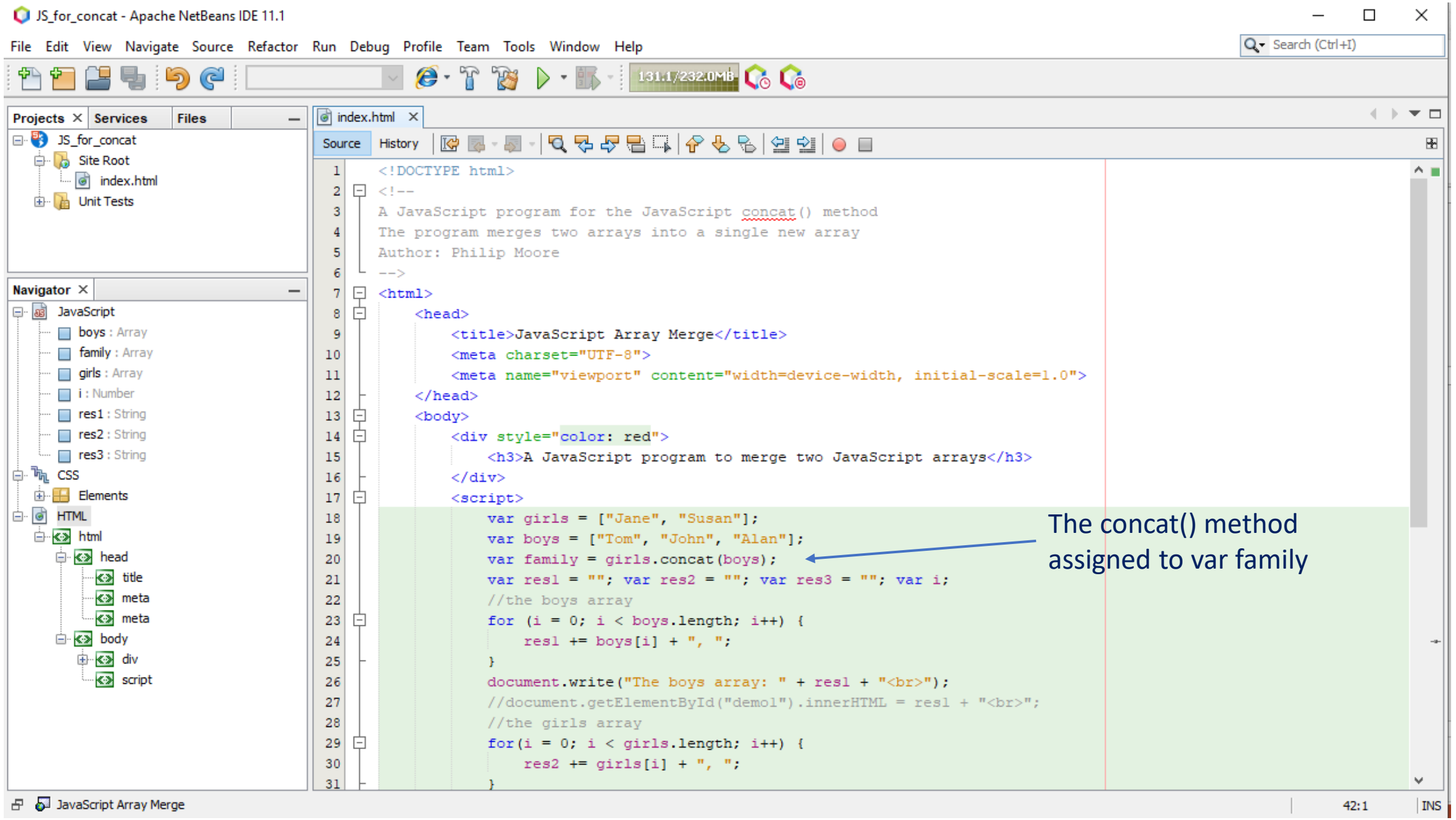
- The *concat()* method creates a new array by merging (concatenating) existing arrays
- For example to merge two arrays:

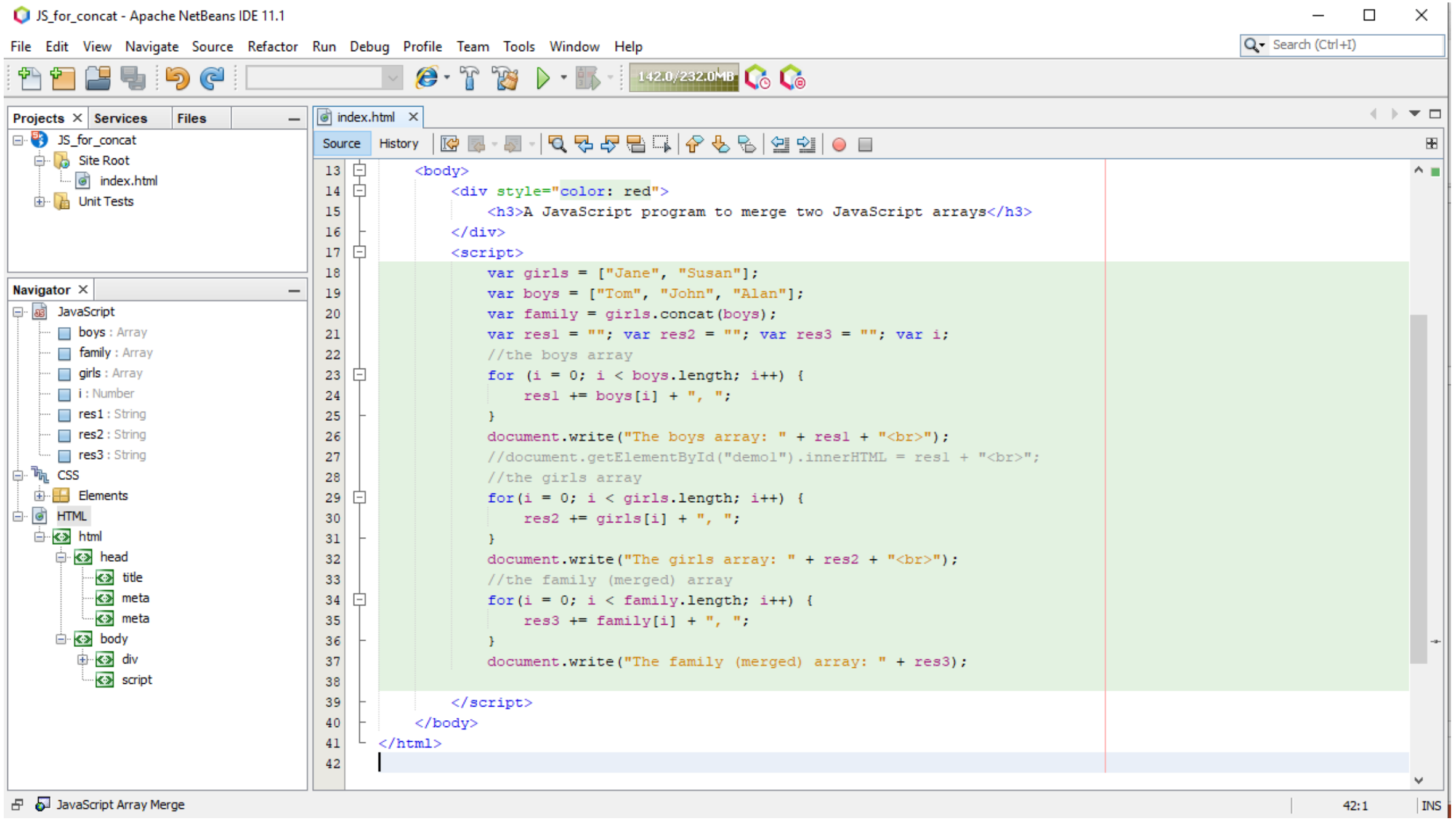
```
var myGirls = ["Cecilie", "Lone"];
var myBoys = ["Emil", "Tobias", "Linus"];
var myChildren = myGirls.concat(myBoys);
```

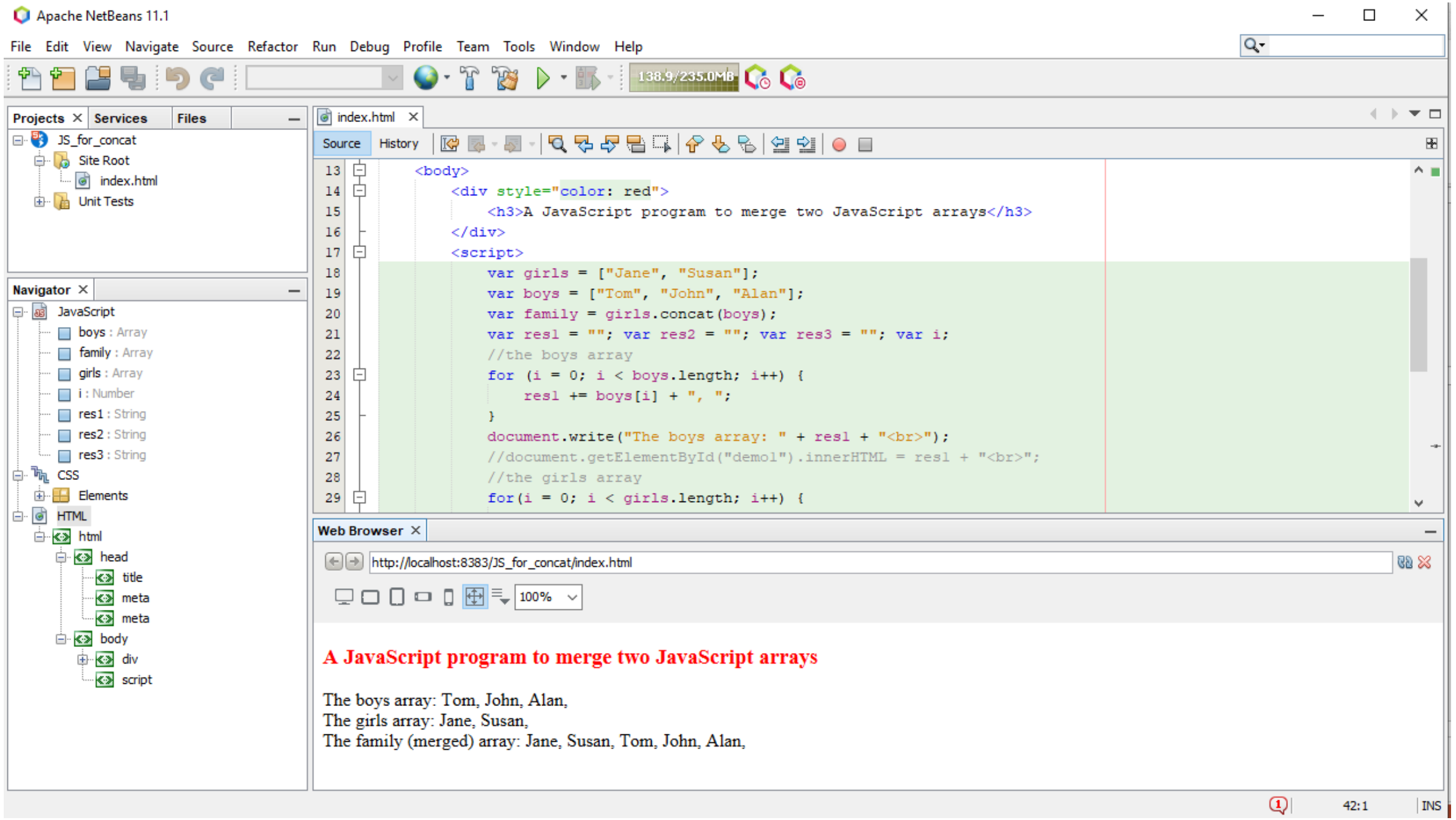
- The example JavaScript code concatenates (merges) `myGirls` and `myBoys` arrays into a single array `myChildren`

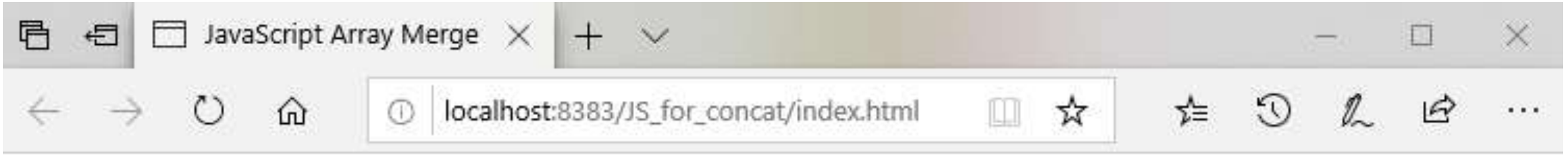
Merging (Concatenating) Arrays

- The concat() method:
  - Can take any number of array arguments
    - Multiple arrays can be merged into one single array
  - The method always returns a new array
- In operation:
  - The method does not delete or change the existing arrays
  - The method always returns a new array
  - `myGirls` and `myBoys` merge into `myChildren`


JS_for_concat - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
131.1/232.0MB
Projects
Services
Files
JS_for_concat
Site Root
index.html
Unit Tests
Navigator
JavaScript
boys : Array
family : Array
girls : Array
i : Number
res1 : String
res2 : String
res3 : String
CSS
Elements
HTML
html
head
title
meta
meta
body
div
script
index.html
Source
History
1 <!DOCTYPE html>
2 <!--
3 A JavaScript program for the JavaScript concat() method
4 The program merges two arrays into a single new array
5 Author: Philip Moore
6 -->
7 <html>
8 <head>
9 <title>JavaScript Array Merge</title>
10 <meta charset="UTF-8">
11 <meta name="viewport" content="width=device-width, initial-scale=1.0">
12 </head>
13 <body>
14
15 <h3>A JavaScript program to merge two JavaScript arrays</h3>
16
17 <script>
18 var girls = ["Jane", "Susan"];
19 var boys = ["Tom", "John", "Alan"];
20 var family = girls.concat(boys);
21 var res1 = ""; var res2 = ""; var res3 = ""; var i;
22 //the boys array
23 for (i = 0; i < boys.length; i++) {
24 res1 += boys[i] + ", ";
25 }
26 document.write("The boys array: " + res1 + "
");
27 //document.getElementById("demo1").innerHTML = res1 + "
";
28 //the girls array
29 for(i = 0; i < girls.length; i++) {
30 res2 += girls[i] + ", ";
31 }
The concat() method assigned to var family
JavaScript Array Merge
42:1
INS


JS_for_concat - Apache NetBeans IDE 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
142.0/232.0MB
Projects
Services
Files
JS_for_concat
Site Root
index.html
Unit Tests
Navigator
JavaScript
boys : Array
family : Array
girls : Array
i : Number
res1 : String
res2 : String
res3 : String
CSS
Elements
HTML
html
head
title
meta
meta
body
div
script
index.html
Source
History
13 <body>
14
15 <h3>A JavaScript program to merge two JavaScript arrays</h3>
16
17 <script>
18 var girls = ["Jane", "Susan"];
19 var boys = ["Tom", "John", "Alan"];
20 var family = girls.concat(boys);
21 var res1 = ""; var res2 = ""; var res3 = ""; var i;
22 //the boys array
23 for (i = 0; i < boys.length; i++) {
24 res1 += boys[i] + ", ";
25 }
26 document.write("The boys array: " + res1 + "
");
27 //document.getElementById("demo1").innerHTML = res1 + "
";
28 //the girls array
29 for(i = 0; i < girls.length; i++) {
30 res2 += girls[i] + ", ";
31 }
32 document.write("The girls array: " + res2 + "
");
33 //the family (merged) array
34 for(i = 0; i < family.length; i++) {
35 res3 += family[i] + ", ";
36 }
37 document.write("The family (merged) array: " + res3);
38
39 </script>
40 </body>
41 </html>
42
JavaScript Array Merge
42:1
INS


Apache NetBeans 11.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
138.9/235.0MB
Projects Services Files
JS_for_concat
Site Root
index.html
Unit Tests
Navigator
JavaScript
boys : Array
family : Array
girls : Array
i : Number
res1 : String
res2 : String
res3 : String
CSS
Elements
HTML
html
head
title
meta
meta
body
div
script
index.html
Source History
13 <body>
14
15 <h3>A JavaScript program to merge two JavaScript arrays</h3>
16
17 <script>
18 var girls = ["Jane", "Susan"];
19 var boys = ["Tom", "John", "Alan"];
20 var family = girls.concat(boys);
21 var res1 = ""; var res2 = ""; var res3 = ""; var i;
22 //the boys array
23 for (i = 0; i < boys.length; i++) {
24 res1 += boys[i] + ", ";
25 }
26 document.write("The boys array: " + res1 + "
");
27 //document.getElementById("demo1").innerHTML = res1 + "
";
28 //the girls array
29 for(i = 0; i < girls.length; i++) {
Web Browser
http://localhost:8383/JS_for_concat/index.html
100%
A JavaScript program to merge two JavaScript arrays
The boys array: Tom, John, Alan,
The girls array: Jane, Susan,
The family (merged) array: Jane, Susan, Tom, John, Alan,
42:1
INS



A JavaScript program to merge two JavaScript arrays

The boys array: Tom, John, Alan,
The girls array: Jane, Susan,
The family (merged) array: Jane, Susan, Tom, John, Alan,

Combining Arrays and Objects

- In JavaScript *strings* may be stored and processed in *arrays* and *objects*
- Other basic *data types* (numbers and Booleans) may be stored and processed in *arrays* and *objects*
- There are other available options:
  - Arrays may be used within objects
  - Objects may be used within arrays
  - Arrays may be used within other arrays
  - Objects may be used within other objects

Combining Arrays and Objects

- The three basic operations in computer programming are:
  - Sequential / selection / iteration
  - These three operation apply in JavaScript and working with arrays and objects
- While the basic concepts in computing and JavaScript arrays and objects and arrays are very simple:
  - Using the available options and combination of options is a JavaScript program can be very complex
- In ‘real-world’ JavaScript programs:
  - There is almost always need for a combination operations on arrays and objects to store data in a scalable and manageable way

Overview

- In this tutorial we have introduced:
  - JavaScript arrays including:
    - What is an array?
    - The nature of arrays and objects in JavaScript with associative arrays
    - Creating and working with arrays, array elements, and array methods
    - Merging (concatenating) two (or more) arrays into a single array
    - Array iteration (looping through arrays)