# INFO 151
# Web Systems and Services

Week 6 (T1)

Dr Philip Moore

Dr Zhili Zhao

## review

- In this tutorial we will introduce:
  - Making changes to arrays and array elements
  - Arrays and strings
  - Sorting arrays
  - A brief overview of the algorithmic approach the array sorting with a worked example showing how the JavaScript elements combine to create a working program

## JavaScript

- In considering JavaScript arrays we have introduced JavaScript:
  - Operators / Operands / Properties / Methods / Functions
  - These JavaScript elements are used to work with arrays
- In this tutorial we will extend JavaScript arrays including:
  - Making changes to arrays and array elements
  - Arrays and strings
  - Sorting arrays
  - A brief overview of the algorithmic approach the array sorting with a worked example showing how the JavaScript elements combine to create a working 'real-world' program

## Making Changes to Array Elements

## Working with Arrays

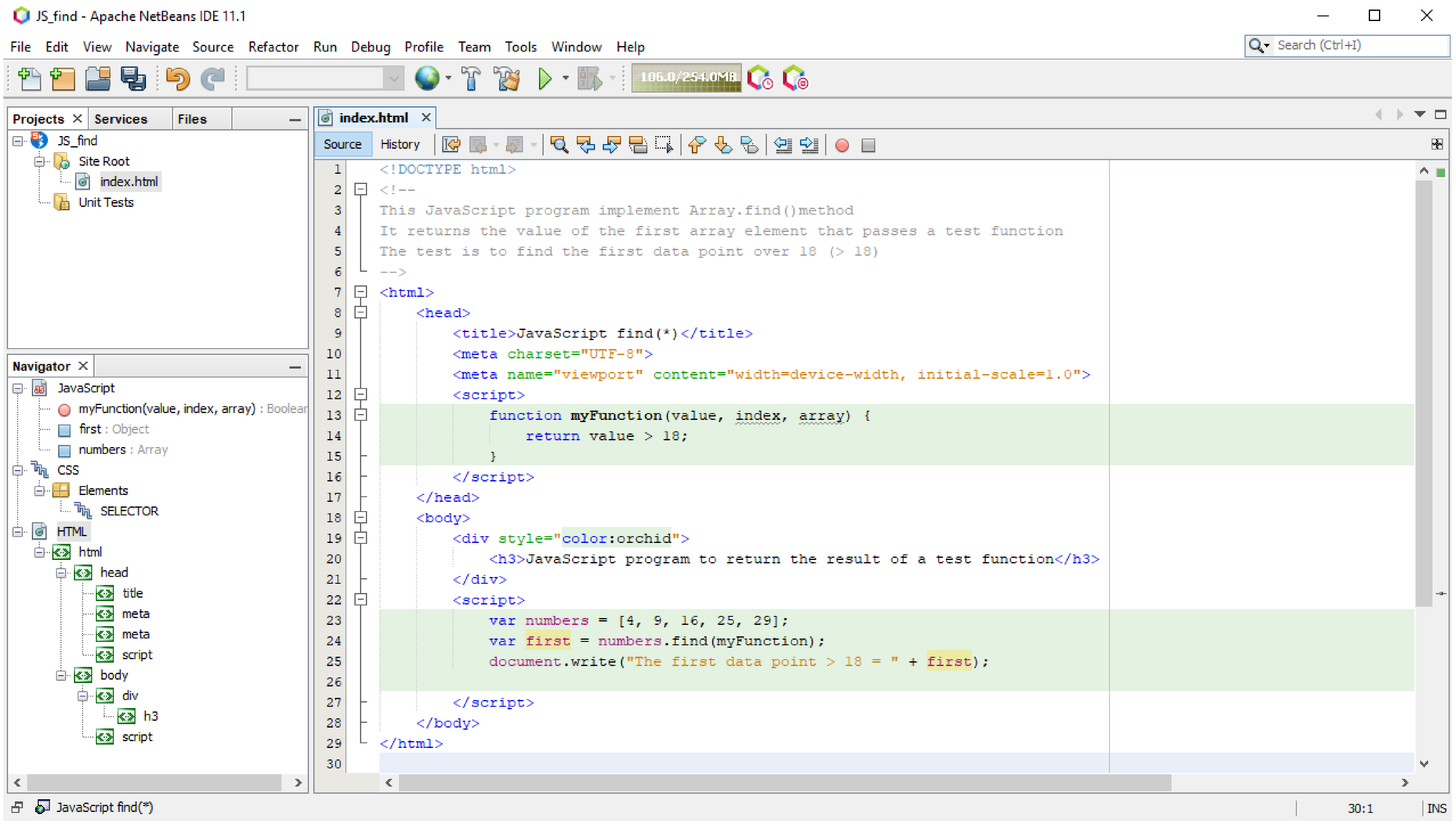
- Testing an array involves searching an array for it's properties which commonly include:
  - The nature of the object: is it an *array* or an *object*
  - The *length* of the array
  - The *data values* in an array (remember in JavaScript arrays are *untyped*)
  - The data value of the *first* element
- In this tutorial we will introduce JavaScript array methods
- For example: The following slide shows (**`find()`**) how to find the first value in an array that passes a test function

## Arrays Find

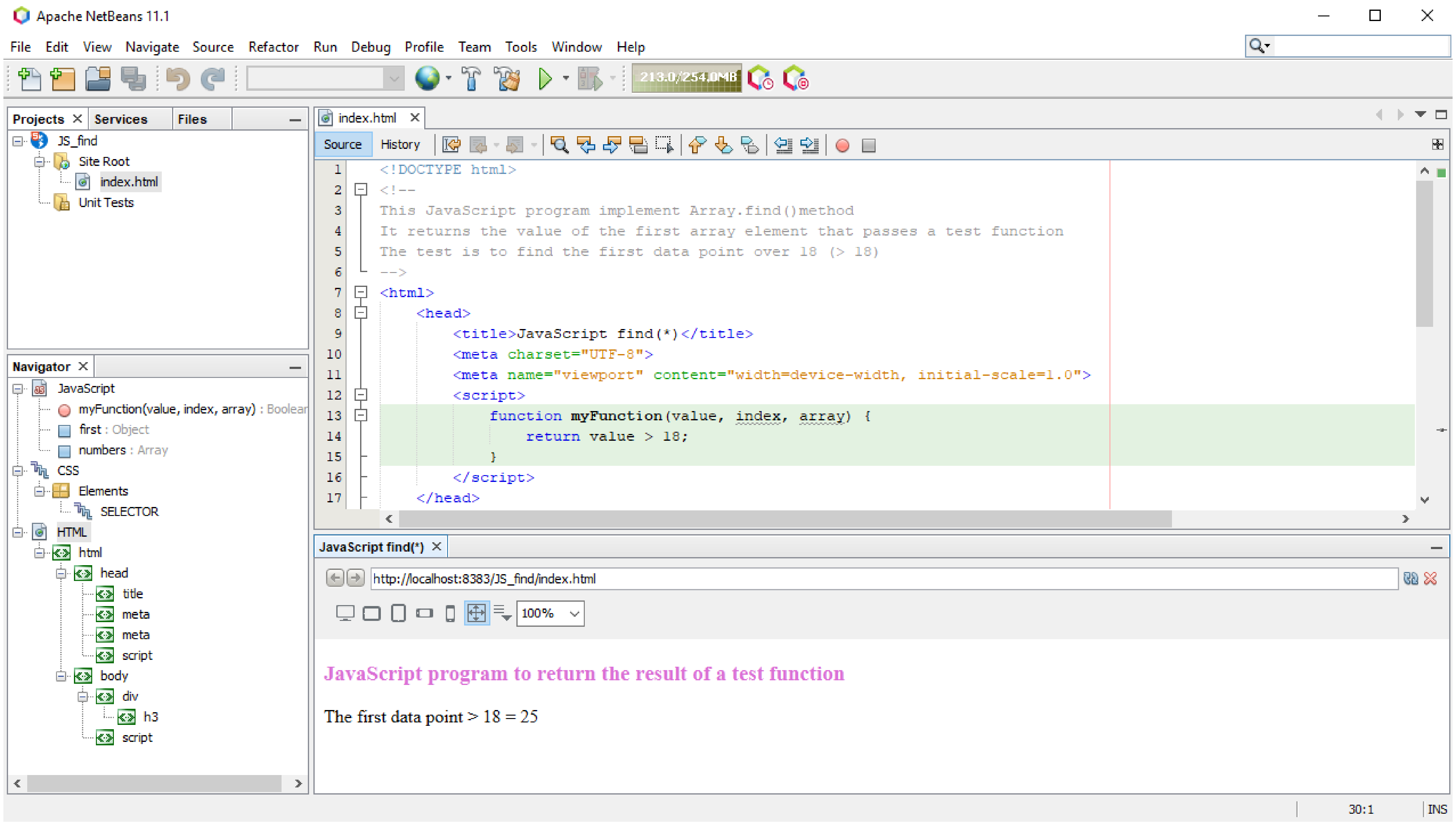
- The **`Array.find()`** method returns the value of the first array element that passes a test function
- This example finds (returns the value of ) the first element that is larger than 18

```
var numbers = [4, 9, 16, 25, 29];
var first = numbers.find(myFunction);
function myFunction(value, index, array) {
return value > 18;

}
```

- The **`find()`** method returns “25”



```html
<!DOCTYPE html>
<!--
This JavaScript program implement Array.find()method
It returns the value of the first array element that passes a test function
The test is to find the first data point over 18 (> 18)
-->
<html>
    <head>
        <title>JavaScript find(*)</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script>
            function myFunction(value, index, array) {
                return value > 18;
            }
        </script>
    </head>
    <body>
        <div style="color:orchid">
            <h3>JavaScript program to return the result of a test function</h3>
        </div>
        <script>
            var numbers = [4, 9, 16, 25, 29];
            var first = numbers.find(myFunction);
            document.write("The first data point > 18 = " + first);

        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
This JavaScript program implement Array.find()method
It returns the value of the first array element that passes a test function
The test is to find the first data point over 18 (> 18)
-->
<html>
    <head>
        <title>JavaScript find(*)</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script>
            function myFunction(value, index, array) {
                return value > 18;
            }
        </script>
    </head>
```

JavaScript program to return the result of a test function

The first data point > 18 = 25

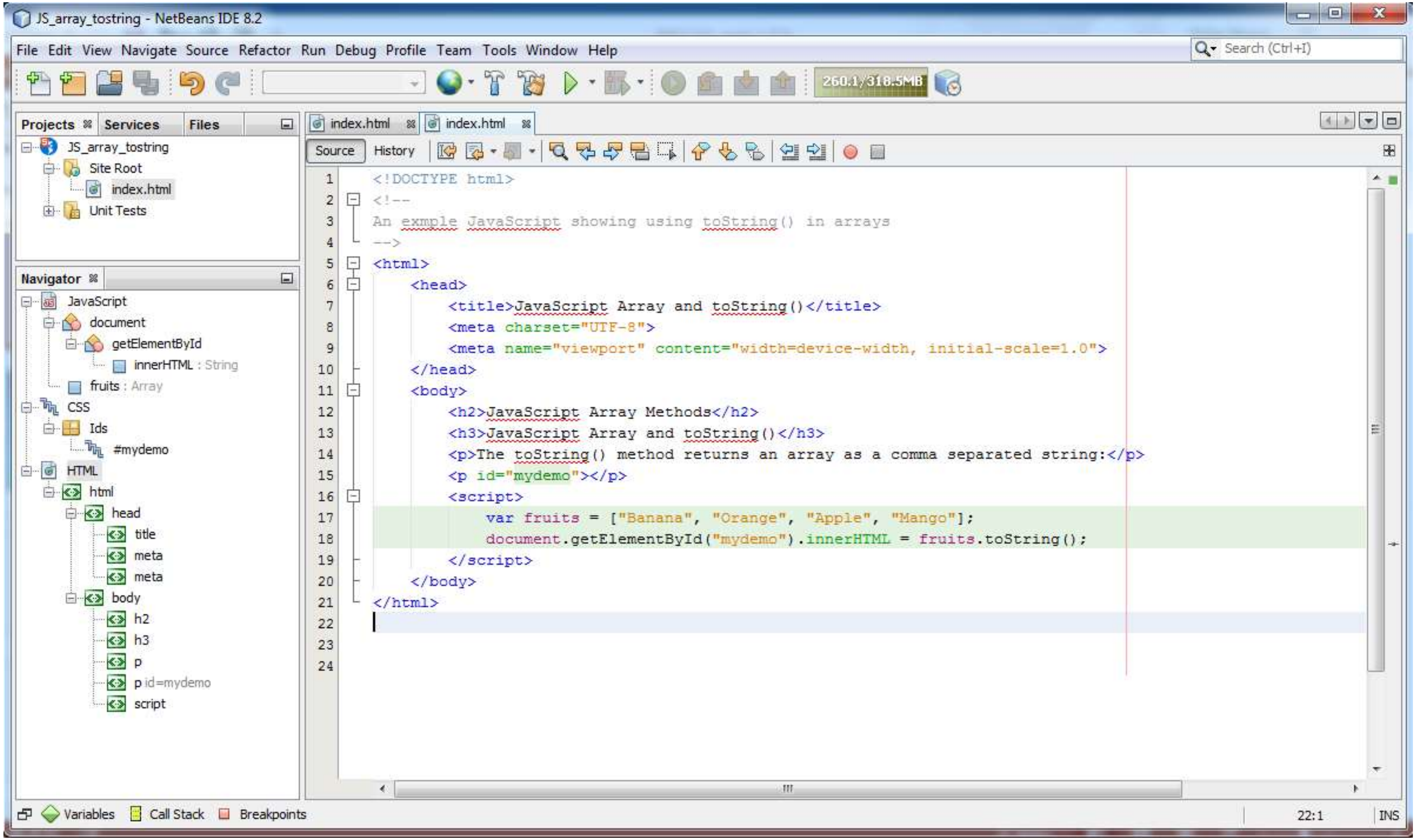
## Arrays and Strings

# Convert an Array to a Comma Separated String

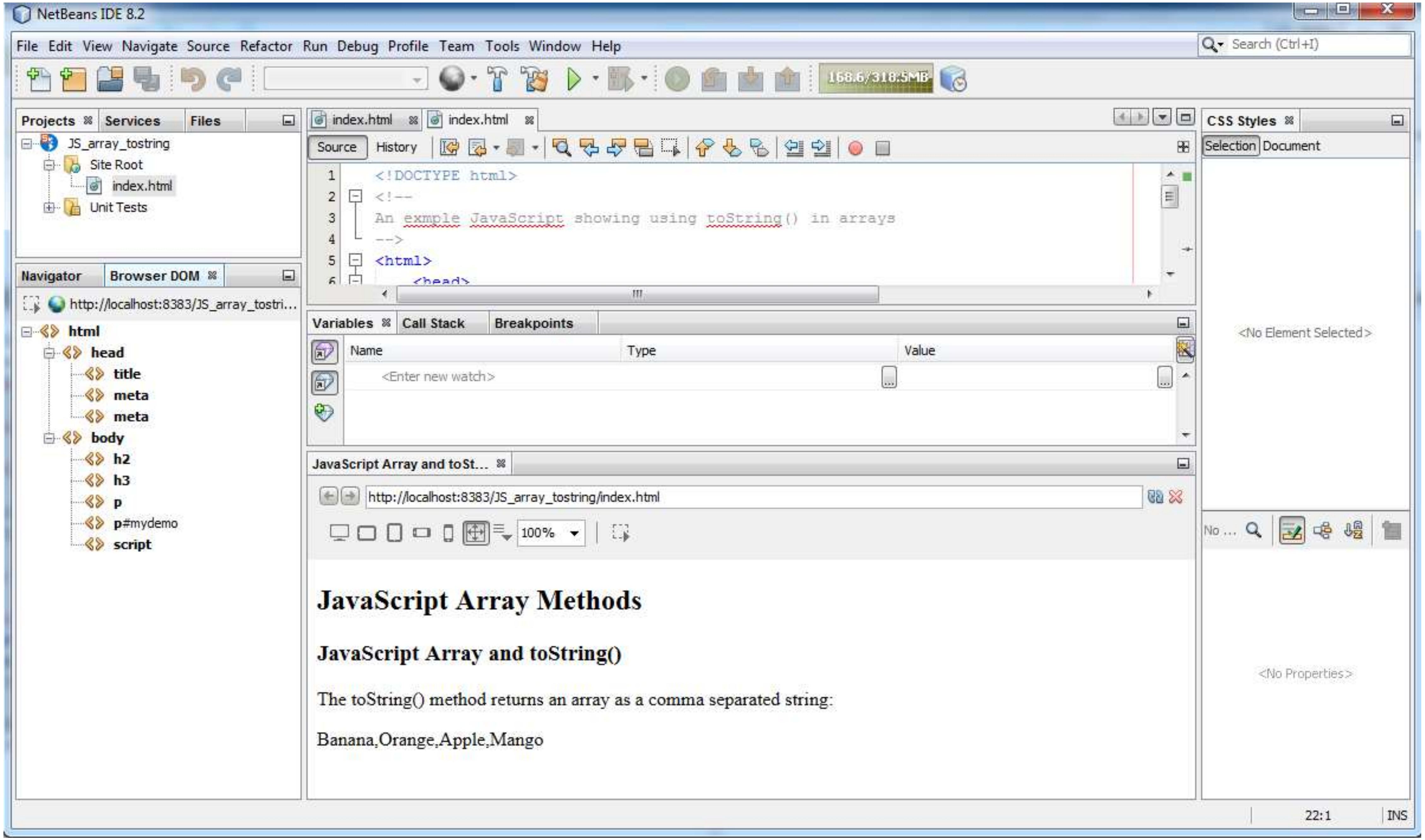
- JavaScript automatically converts an array to a comma separated string when a primitive value is expected – this is always the case for array output
- The following two examples will produce the same result (output):
    - “Banana,Orange,Apple,Mango” (a comma separated string)

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write (fruits.toString());
```

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write (fruits);
```



```
<!DOCTYPE html>
<!--
An exmple JavaScript showing using toString() in arrays
-->
<html>
    <head>
        <title>JavaScript Array and toString()</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h2>JavaScript Array Methods</h2>
        <h3>JavaScript Array and toString()</h3>
        <p>The toString() method returns an array as a comma separated string:</p>
        <p id="mydemo"></p>
        <script>
            var fruits = ["Banana", "Orange", "Apple", "Mango"];
            document.getElementById("mydemo").innerHTML = fruits.toString();
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
An exmple JavaScript showing using toString() in arrays
-->
<html>
    <head>
```

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

**JavaScript Array Methods**

**JavaScript Array and toString()**

The toString() method returns an array as a comma separated string:

Banana,Orange,Apple,Mango

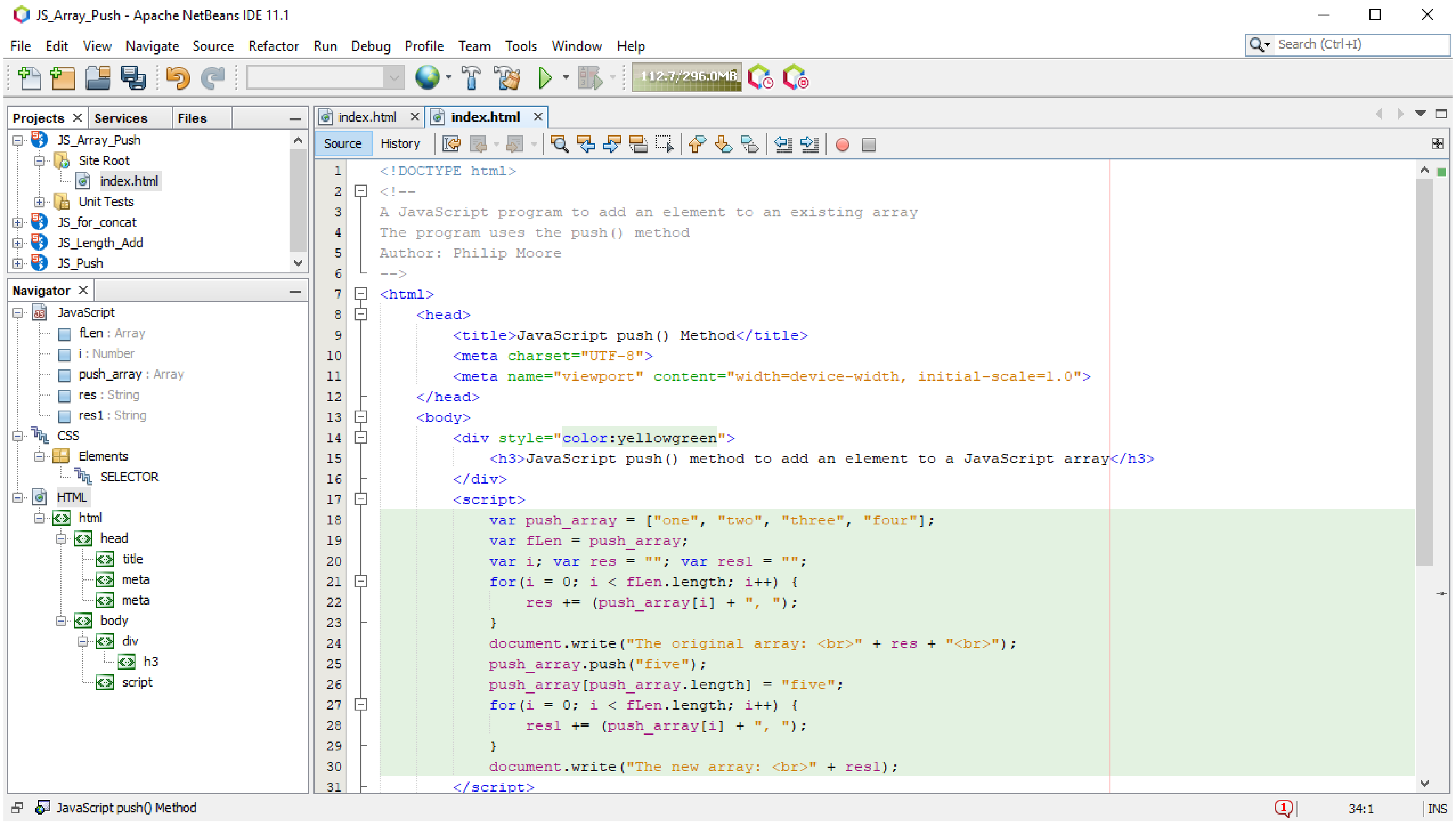
## Add

## Adding Elements using the **push** Property

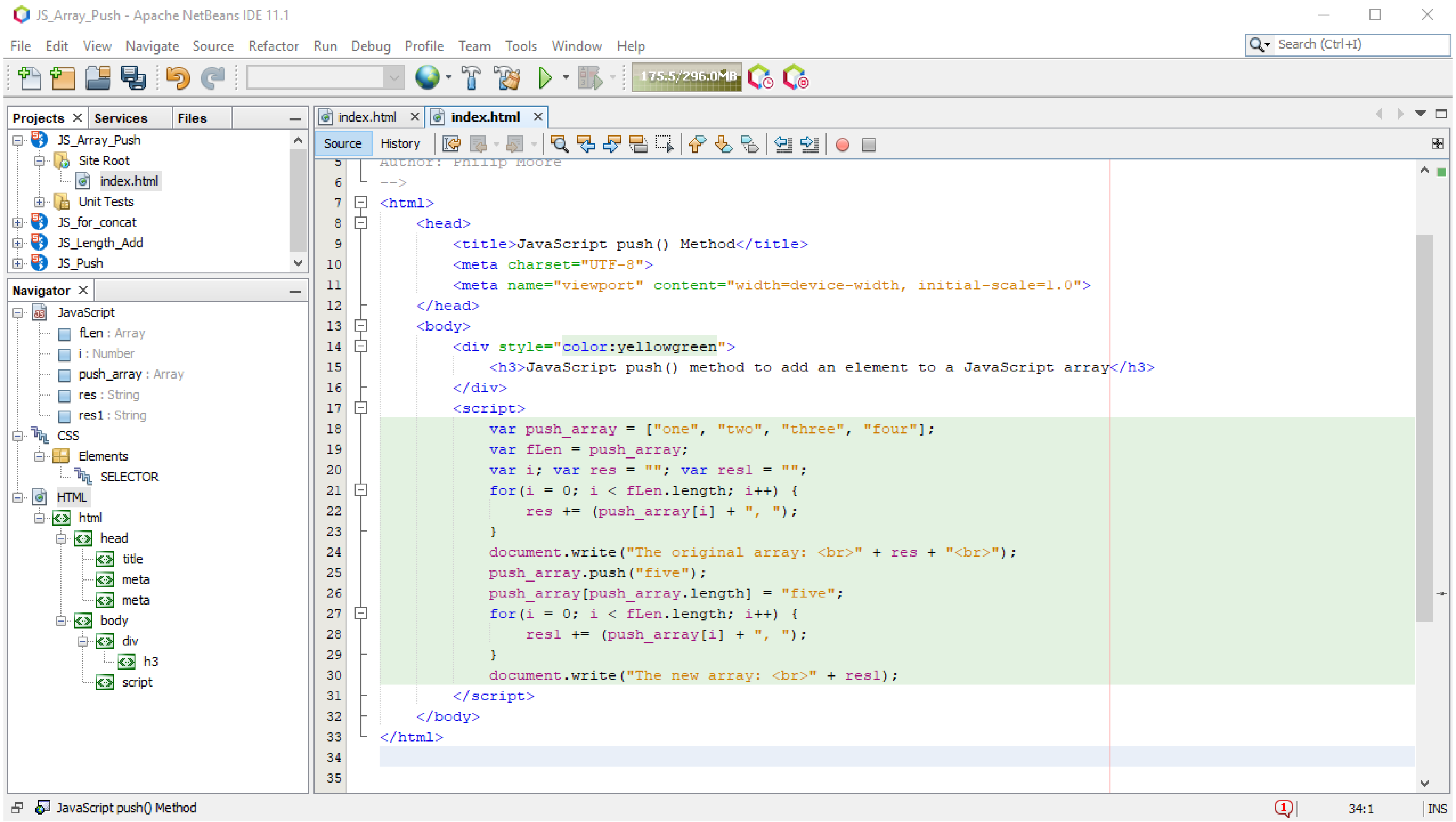
- To add a new element to the end of an existing array
  - We can use the **push** method:
- For example

```
var push_array = [“one", “two", “three”, “four”];
push_array.push()[“five”];
```

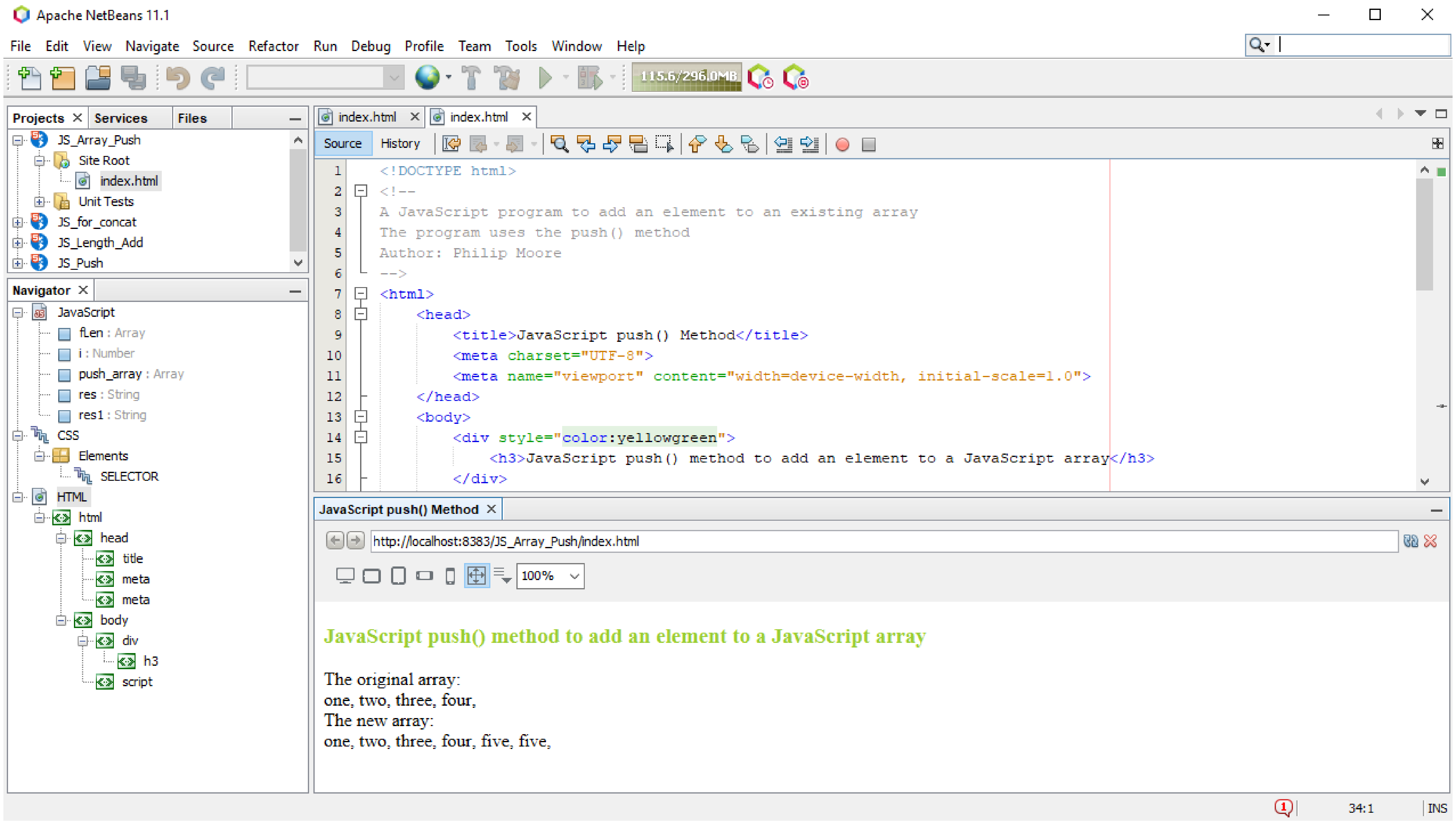
- This method adds a new element(s) to the person array
  - We can add multiple elements to an array with this method
  - The following worked example shows the **push** method and output



```html
<!DOCTYPE html>
<!--
A JavaScript program to add an element to an existing array
The program uses the push() method
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array.push("five");
            push_array[push_array.length] = "five";
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
            document.write("The new array: <br>" + res1);
        </script>
```



```
    Author: Philip Moore
    -->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array.push("five");
            push_array[push_array.length] = "five";
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
            document.write("The new array: <br>" + res1);
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
A JavaScript program to add an element to an existing array
The program uses the push() method
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
```

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

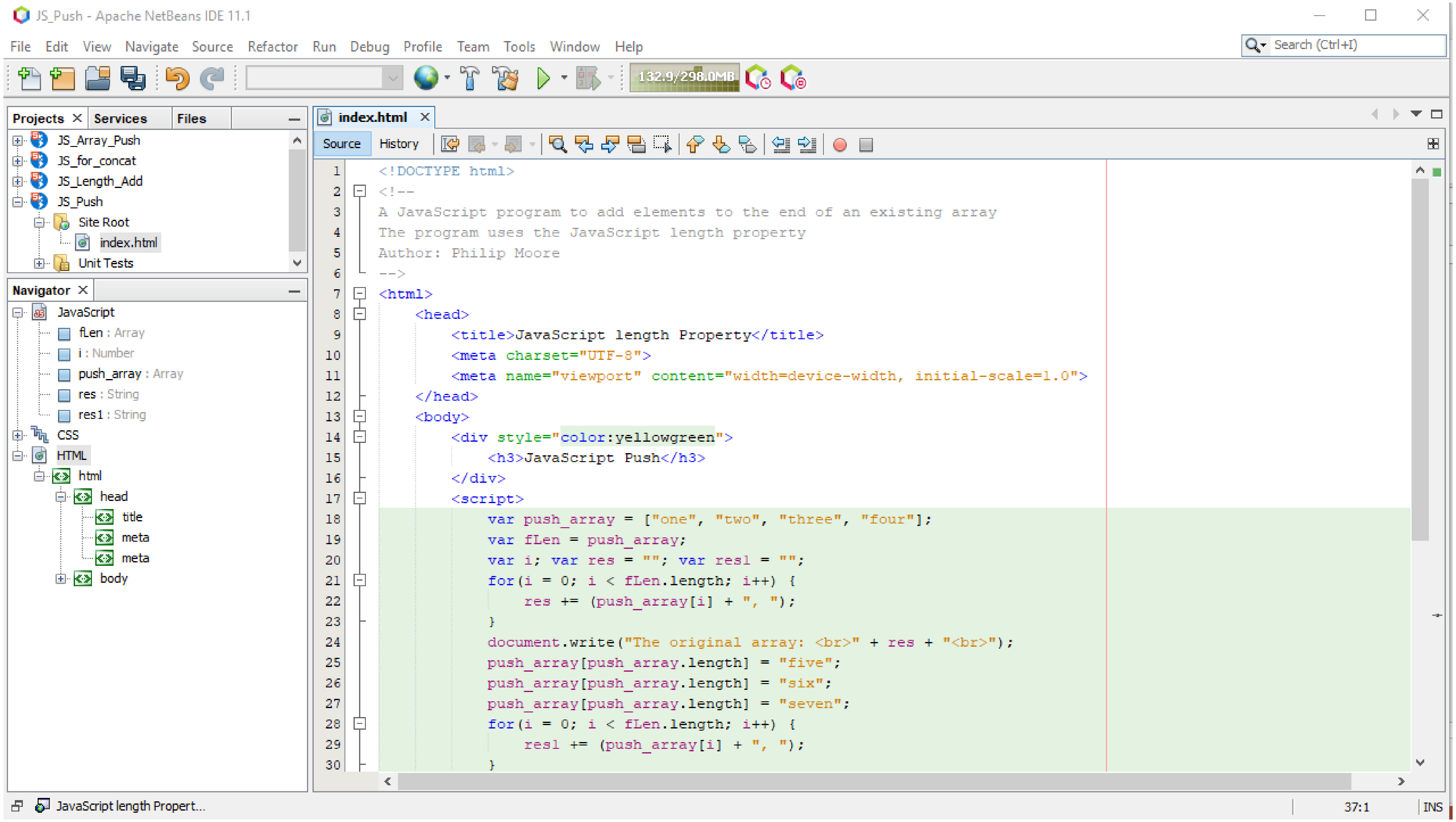
```
JavaScript push() method to add an element to a JavaScript array

The original array:
one, two, three, four,
The new array:
one, two, three, four, five, five,
```

# Adding Elements using the **`length`** Property

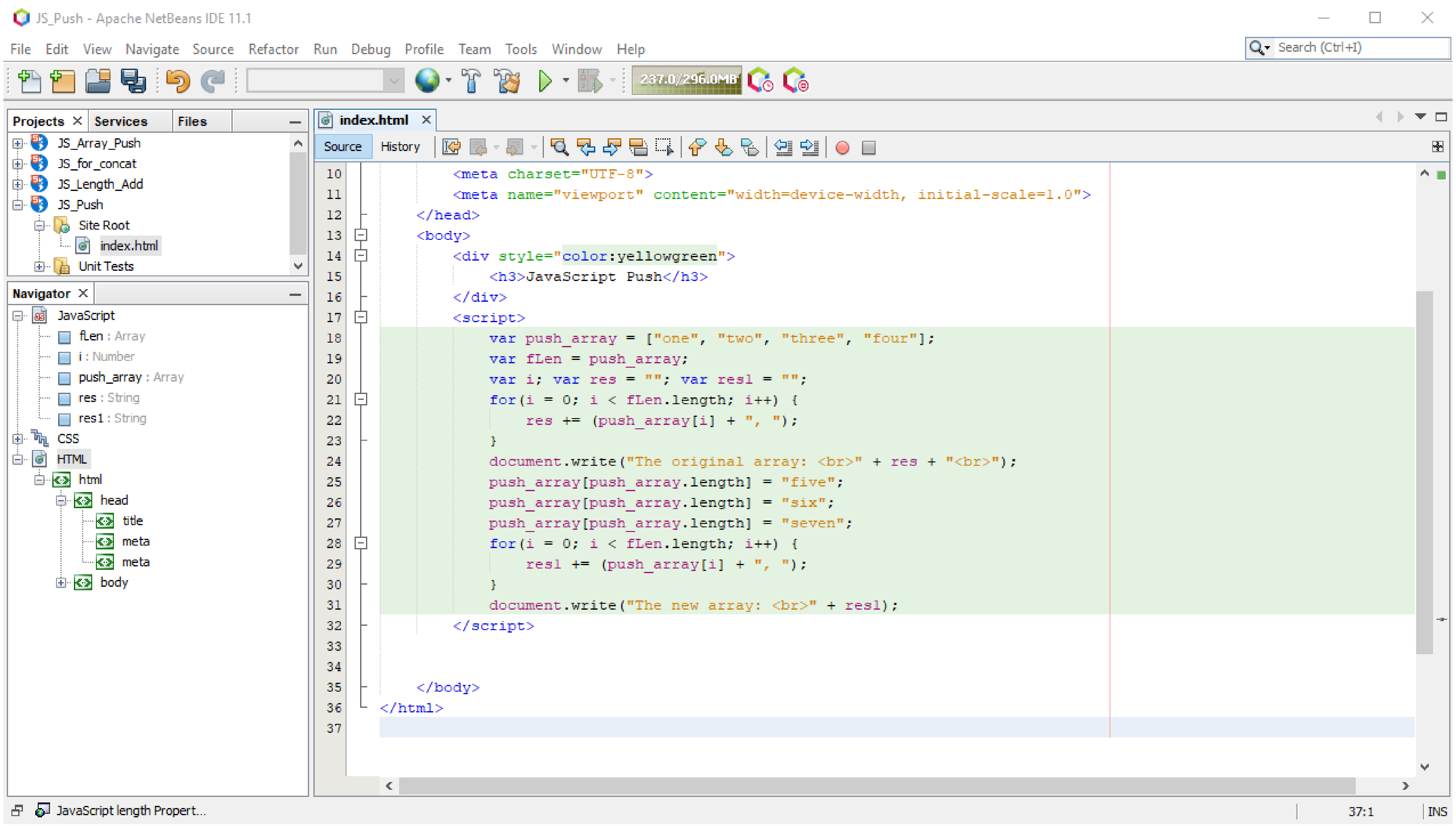
- The **`length`** property provides an easy way to append a new element(s) to an array
- For example:

```
var push_array = [“one", “two", “three", "four"];
push_array[push_array.length]=“five";
push_array[push_array.length]=“six";
push_array[push_array.length]=“seven";
```

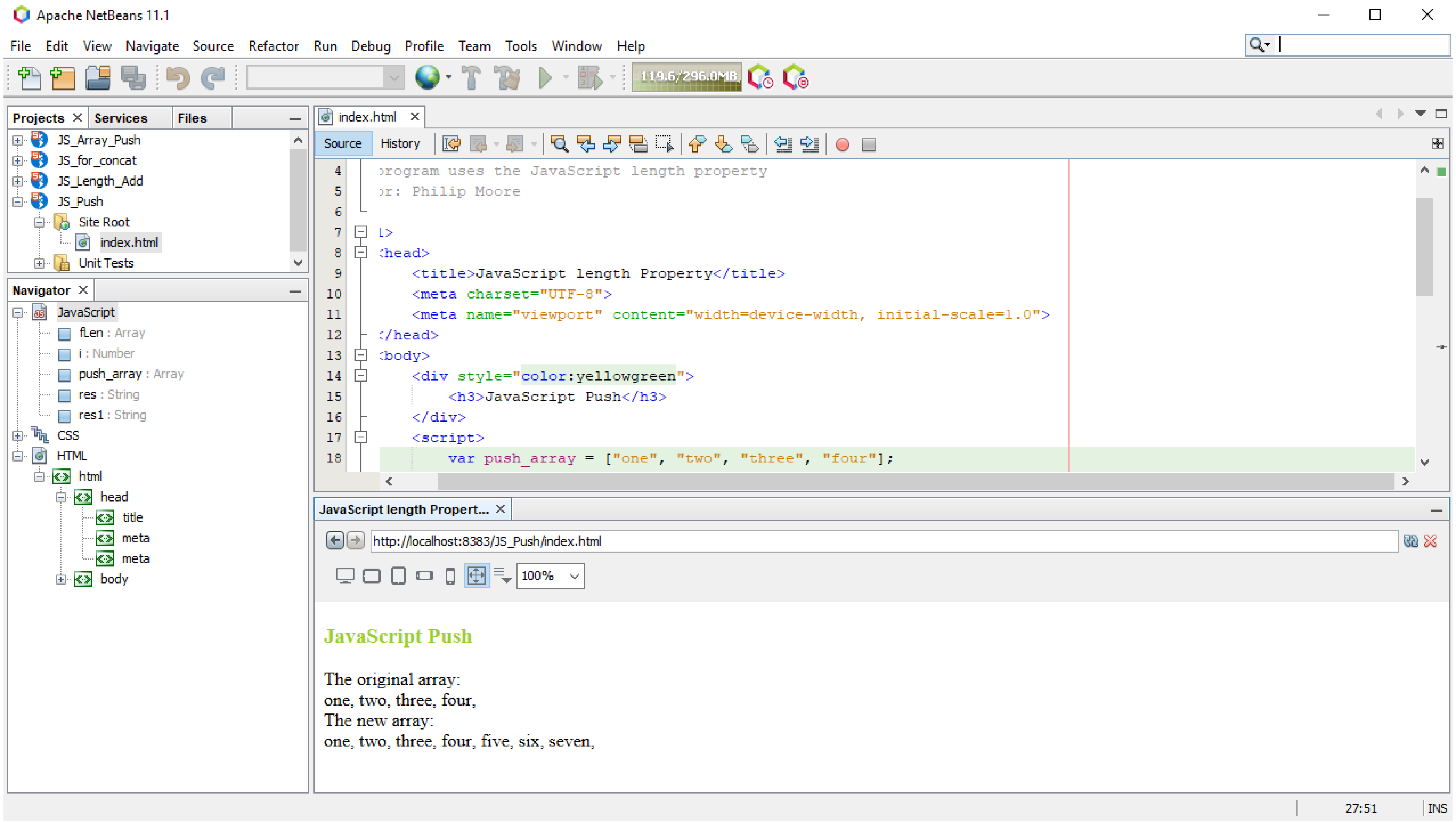
- The JavaScript appends “five” “six” and “seven” to the end of the array



```
<!DOCTYPE html>
<!--
A JavaScript program to add elements to the end of an existing array
The program uses the JavaScript length property
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript length Property</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript Push</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array[push_array.length] = "five";
            push_array[push_array.length] = "six";
            push_array[push_array.length] = "seven";
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
```



```
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript Push</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array[push_array.length] = "five";
            push_array[push_array.length] = "six";
            push_array[push_array.length] = "seven";
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
            document.write("The new array: <br>" + res1);
        </script>


    </body>
</html>
```



```
program uses the JavaScript length property
or: Philip Moore

l>
<head>
    <title>JavaScript length Property</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <div style="color:yellowgreen">
        <h3>JavaScript Push</h3>
    </div>
    <script>
        var push_array = ["one", "two", "three", "four"];
```

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

**JavaScript Push**

The original array:
one, two, three, four,
The new array:
one, two, three, four, five, six, seven,

# Problems in Adding Array Elements

- Adding elements with high indexes can create undefined *holes* in an array:
  - For example the following JavaScript code adds a new element [6] and the data value “six” to the numbers array

```
var numbers = [“one", “two", “three", “four"];
numbers[6] = “six";
```

  - However: the existing array has 4 elements
  - We have assigned “six” to numbers[6] (it should be numbers[5])
- In high-level programming languages (such as Java)
  - This would ‘throw an array out of bounds exception’
  - In JavaScript it will not identify the error but the program logic will be wrong

## Delete

## Delete and Array Element

- As JavaScript arrays are objects, elements can be deleted by using the JavaScript operator **`delete`**
- For example

```
var numbers = [“one", “two", “three", “four"];
delete numbers[0];
```

- This changes the first element in **`numbers`** to **`undefined`**
- Using **`delete`** can result in errors because:
    - The **`delete`** method is designed to free up system memory rather than to adjust array sizes

# Delete and Array Element

- There are better alternatives:
  - **`pop()`**
  - **`splice()`**
  - **`List.splice()`**
  - **`Shift()`**
  - **`Filter()`**
  - The alternative methods all address different requirements
  - I have provided supplementary a resource setting out details of the alternative methods and their design uses

# Delete and Array Element

- The alternative methods and their uses are:
  - **`splice()`** remove specific elements)
  - **`pop()`** remove elements from the end of an array
  - **`shift()`** remove elements from the start of an array
  - Use **`delete`** to remove individual array objects
- We may also:
  - Find and remove an element of a specific value
  - Find and remove multiple elements with the same value
  - Remove elements by filtering an array

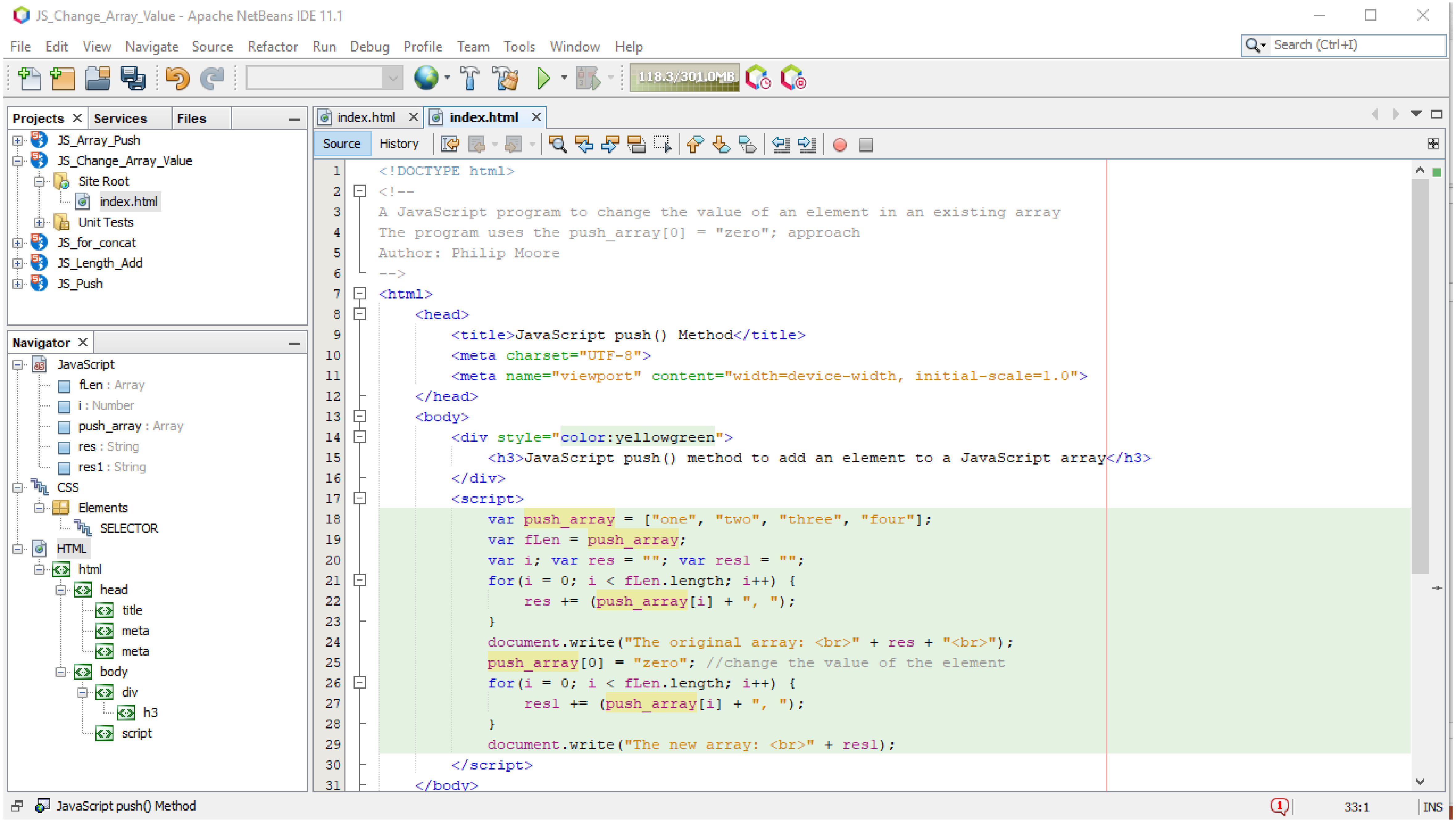
## Changing Values

# Changing Array Elements

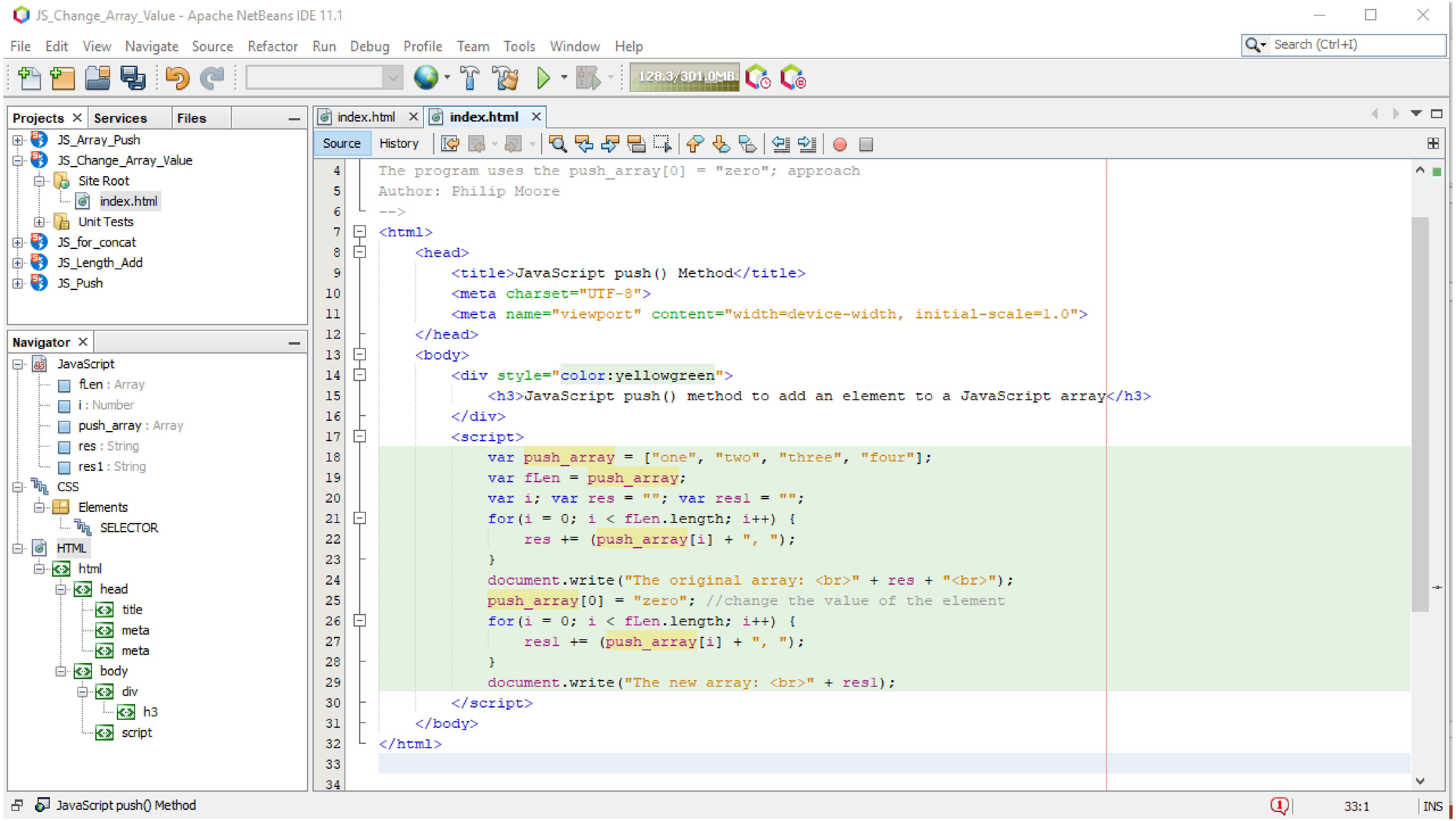
- A frequent activity is to update array element value(s)
- Note: remember that JavaScript is not typed so the datatype may change resulting in a possible logic error
- The following example changes the first element of numbers[0]) from “one” to “zero“:

```
var numbers = [“one", “two", “three", “four"];
numbers[0] = “zero";
```

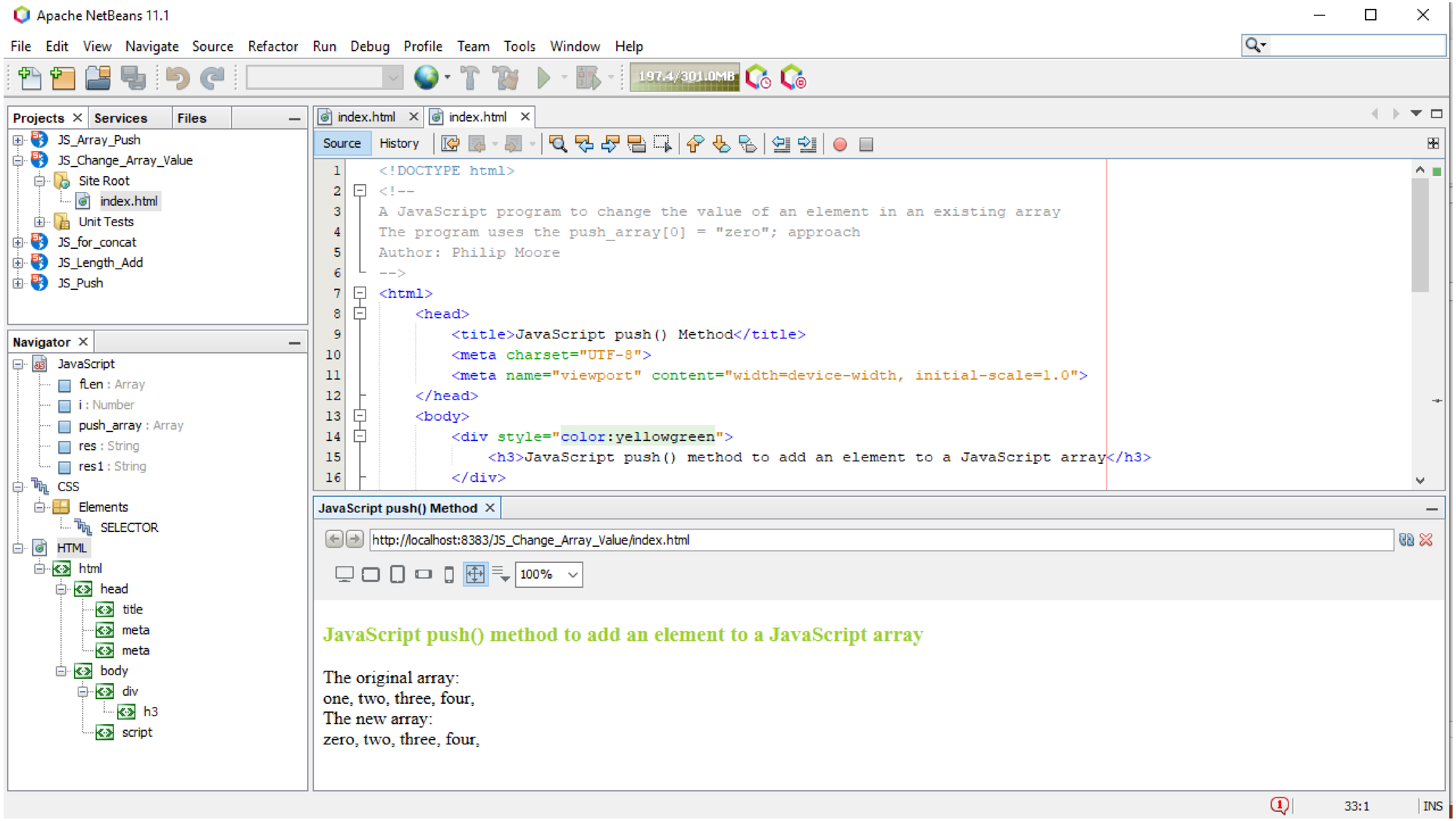
- The following worked example shows the changes



```
<!DOCTYPE html>
<!--
A JavaScript program to change the value of an element in an existing array
The program uses the push_array[0] = "zero"; approach
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array[0] = "zero"; //change the value of the element
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
            document.write("The new array: <br>" + res1);
        </script>
    </body>
```



```
The program uses the push_array[0] = "zero"; approach
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
        <script>
            var push_array = ["one", "two", "three", "four"];
            var fLen = push_array;
            var i; var res = ""; var res1 = "";
            for(i = 0; i < fLen.length; i++) {
                res += (push_array[i] + ", ");
            }
            document.write("The original array: <br>" + res + "<br>");
            push_array[0] = "zero"; //change the value of the element
            for(i = 0; i < fLen.length; i++) {
                res1 += (push_array[i] + ", ");
            }
            document.write("The new array: <br>" + res1);
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
A JavaScript program to change the value of an element in an existing array
The program uses the push_array[0] = "zero"; approach
Author: Philip Moore
-->
<html>
    <head>
        <title>JavaScript push() Method</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div style="color:yellowgreen">
            <h3>JavaScript push() method to add an element to a JavaScript array</h3>
        </div>
```

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

```
JavaScript push() method to add an element to a JavaScript array

The original array:
one, two, three, four,
The new array:
zero, two, three, four,
```

## JavaScript Program Code Structure

- We have introduced the methods to manage arrays by:
  - Adding elements and data values to arrays
  - Deleting elements and data values to arrays
  - Changing the data values in array elements
- The JavaScript program code structure:
  - Will follow be similar for all the tasks as will be the method of accessing arrays as shown in the previous worked examples
  - However: you must use the appropriate methods to make additions, deletions, and changes to arrays and array elements

## Sorting Arrays

## Sorting Arrays

- Sorting the elements of a JavaScript array is a frequent operation
- Arrays can be sorted in a number of ways which include:
    - *Numeric* and *alphabetical* sort
    - *Ascending* and *descending* sort
    - *Object* array sort
    - Sorting an array in *Random* order and *reversing* the order
    - Find the *highest* (max) (or) *lowest*) array element value
- We will show the basic approach with the following examples
- Comprehensive details for array sort methods may be found in the course resources

## JavaScript Array Properties and Methods

- In week 1 we introduced programming ‘frameworks’
- A strength of JavaScript (and Java) is the built in pre-defined API’s which implement properties and methods
- JavaScript arrays have such properties and methods
- For example:

```
var y = cars.sort();
```

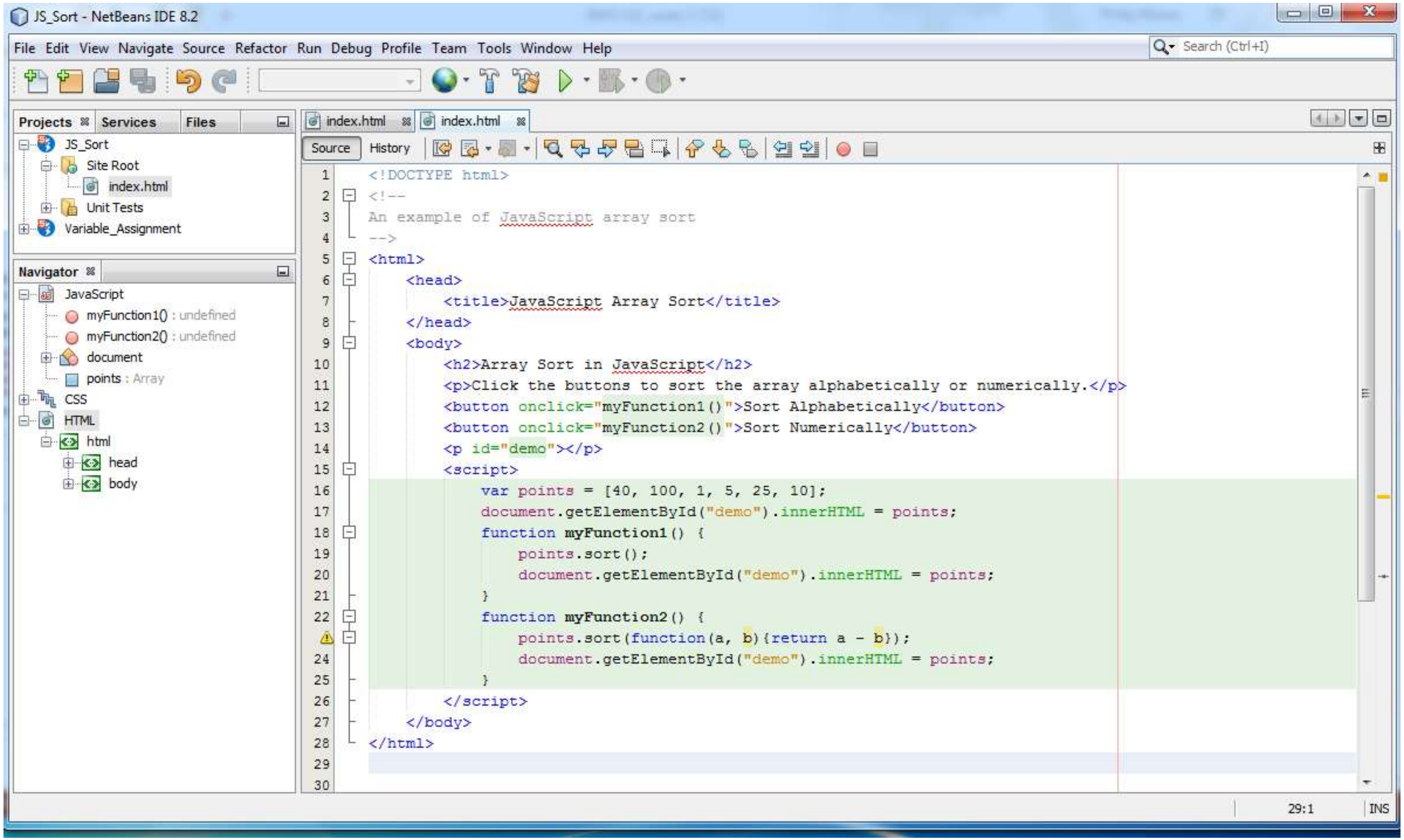
- The **`sort()`** method sorts arrays

## Numeric Sort

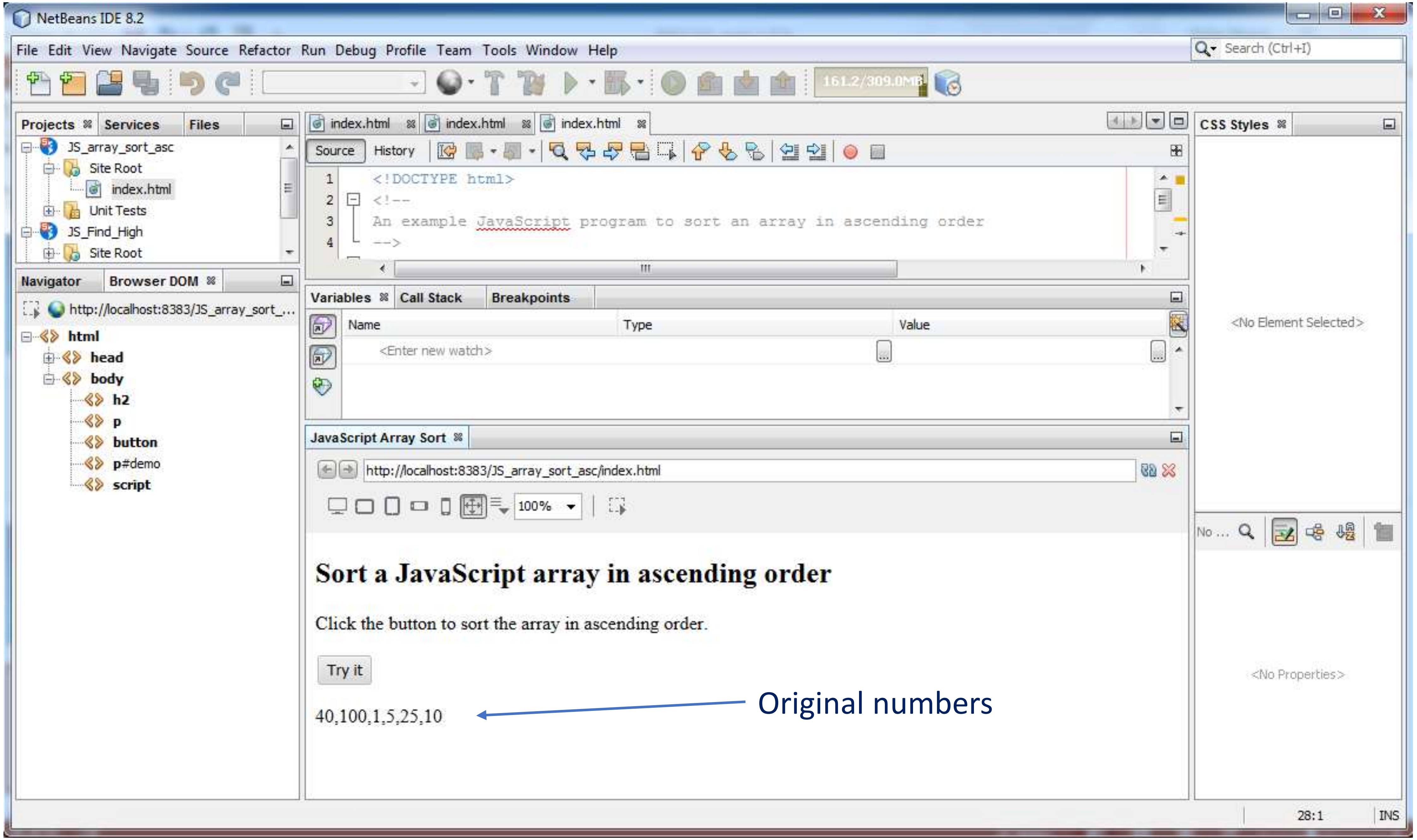
## Numeric Sort

- By default, the **`sort()`** function sorts values as **`strings`**
- This works well for strings ("Apple" comes before "Banana")
  - However, if numbers are sorted as **`strings`**, "25" is bigger than "100" because "2" is larger than "1".
- Because of this the **`sort()`** method will produce incorrect result when sorting numbers (the result is an alphabetical sort)
  - This can be corrected by providing a **`compare function`** as follows:

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```



```
<!DOCTYPE html>
<!--
An example of JavaScript array sort
-->
<html>
    <head>
        <title>JavaScript Array Sort</title>
    </head>
    <body>
        <h2>Array Sort in JavaScript</h2>
        <p>Click the buttons to sort the array alphabetically or numerically.</p>
        <button onclick="myFunction1()">Sort Alphabetically</button>
        <button onclick="myFunction2()">Sort Numerically</button>
        <p id="demo"></p>
        <script>
            var points = [40, 100, 1, 5, 25, 10];
            document.getElementById("demo").innerHTML = points;
            function myFunction1() {
                points.sort();
                document.getElementById("demo").innerHTML = points;
            }
            function myFunction2() {
                points.sort(function(a, b){return a - b});
                document.getElementById("demo").innerHTML = points;
            }
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
An example JavaScript program to sort an array in ascending order
-->
```

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

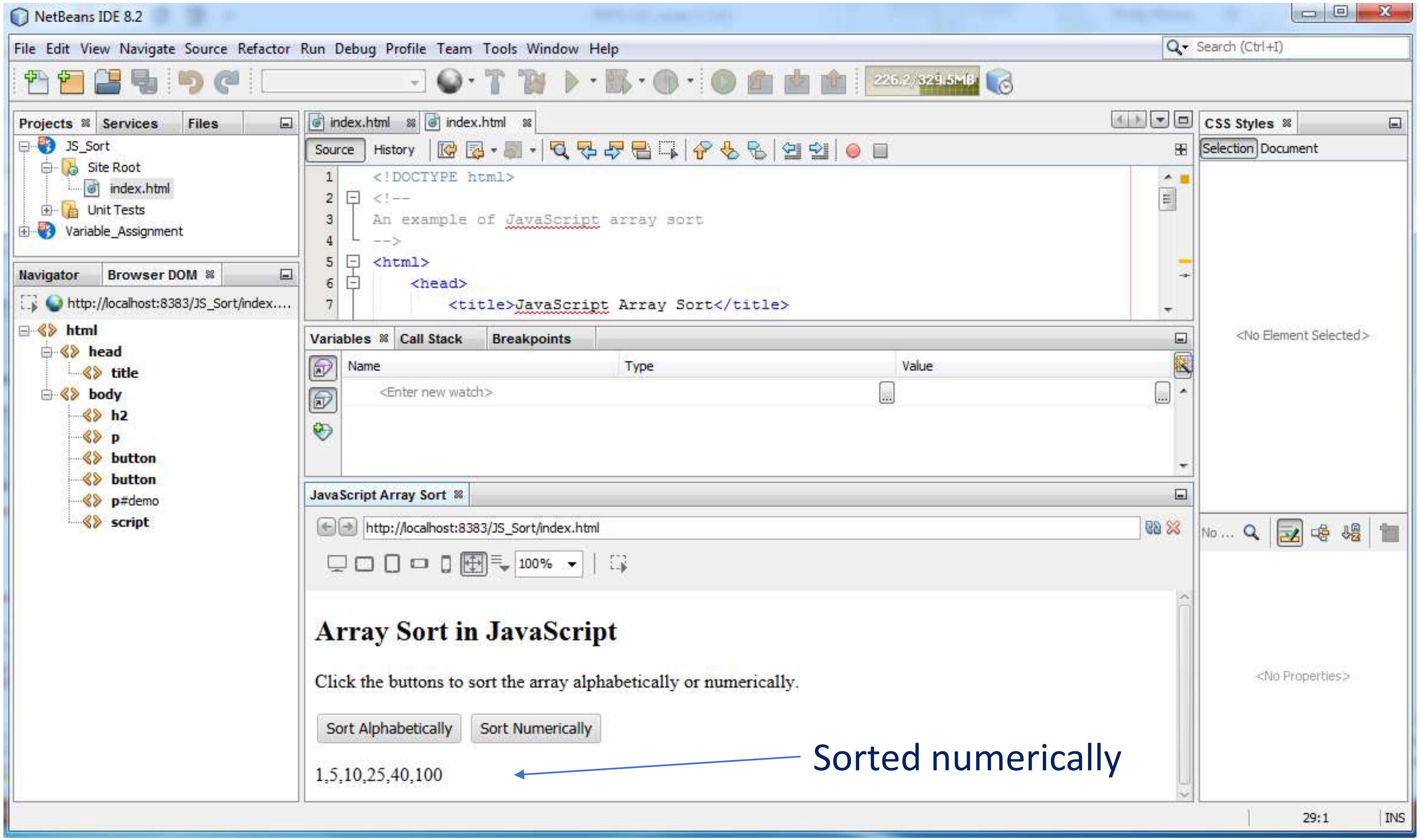
**Sort a JavaScript array in ascending order**

Click the button to sort the array in ascending order.

Try it

40,100,1,5,25,10

Original numbers



```
<!DOCTYPE html>
<!--
An example of JavaScript array sort
-->
<html>
    <head>
        <title>JavaScript Array Sort</title>
```

**Array Sort in JavaScript**

Click the buttons to sort the array alphabetically or numerically.

Sort Alphabetically Sort Numerically

```
1,5,10,25,40,100
```

Sorted numerically

## Alphabetical Sort

## Alphabetical Sort (strings)

- The **`sort()`** method sorts an array alphabetically:
- For example

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.sort();
```

- Sorts the elements of fruits as follows

```
Apple,Banana,Mango,Orange
```

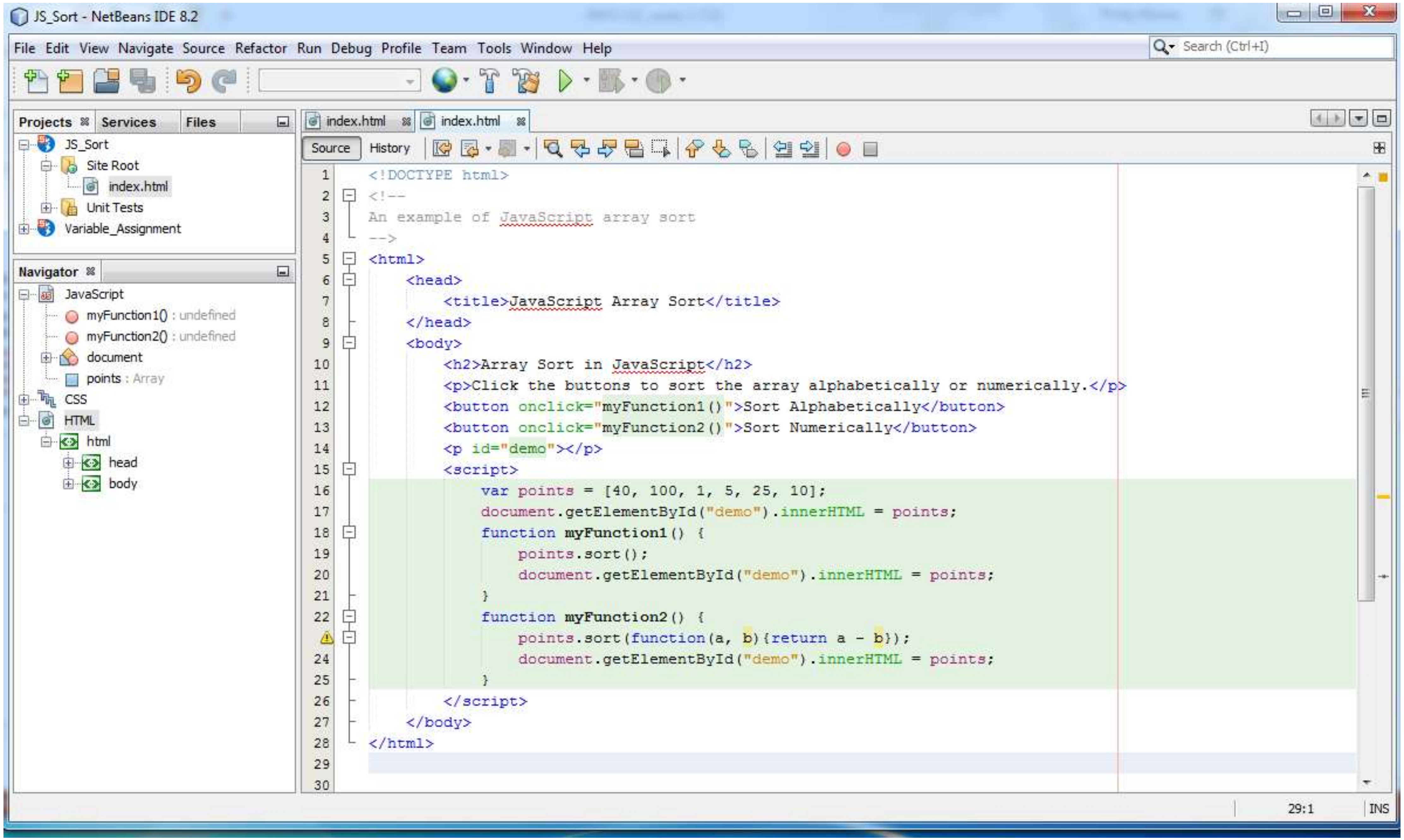
- The sort process is as expected (i.e., a, b, c, etc)
  - For numbers the process can present problems

## Alphabetical Sort (numbers)

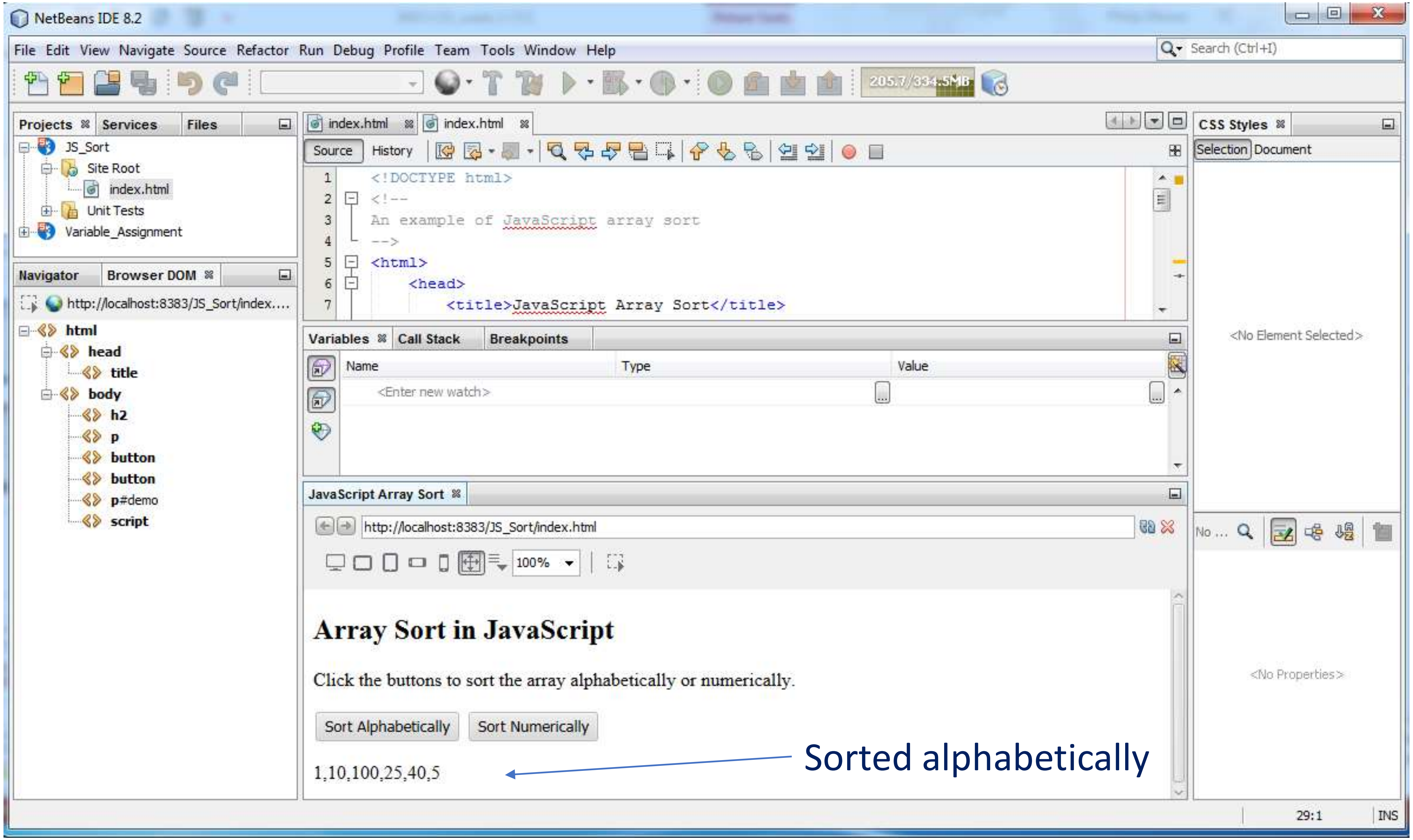
- The **`sort()`** method sorts an array alphabetically:
- For example consider the points array:

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

- Sorts the elements of the points array as follows
  - 1, 10, 100, 25, 40, 5
- The following example shows a worked an example to sort an array alphabetically



```
<!DOCTYPE html>
<!--
An example of JavaScript array sort
-->
<html>
    <head>
        <title>JavaScript Array Sort</title>
    </head>
    <body>
        <h2>Array Sort in JavaScript</h2>
        <p>Click the buttons to sort the array alphabetically or numerically.</p>
        <button onclick="myFunction1()">Sort Alphabetically</button>
        <button onclick="myFunction2()">Sort Numerically</button>
        <p id="demo"></p>
        <script>
            var points = [40, 100, 1, 5, 25, 10];
            document.getElementById("demo").innerHTML = points;
            function myFunction1() {
                points.sort();
                document.getElementById("demo").innerHTML = points;
            }
            function myFunction2() {
                points.sort(function(a, b){return a - b});
                document.getElementById("demo").innerHTML = points;
            }
        </script>
    </body>
</html>
```



```
<!DOCTYPE html>
<!--
An example of JavaScript array sort
-->
<html>
    <head>
        <title>JavaScript Array Sort</title>
```

**Array Sort in JavaScript**

Click the buttons to sort the array alphabetically or numerically.

Sort Alphabetically | Sort Numerically

1,10,100,25,40,5

Sorted alphabetically

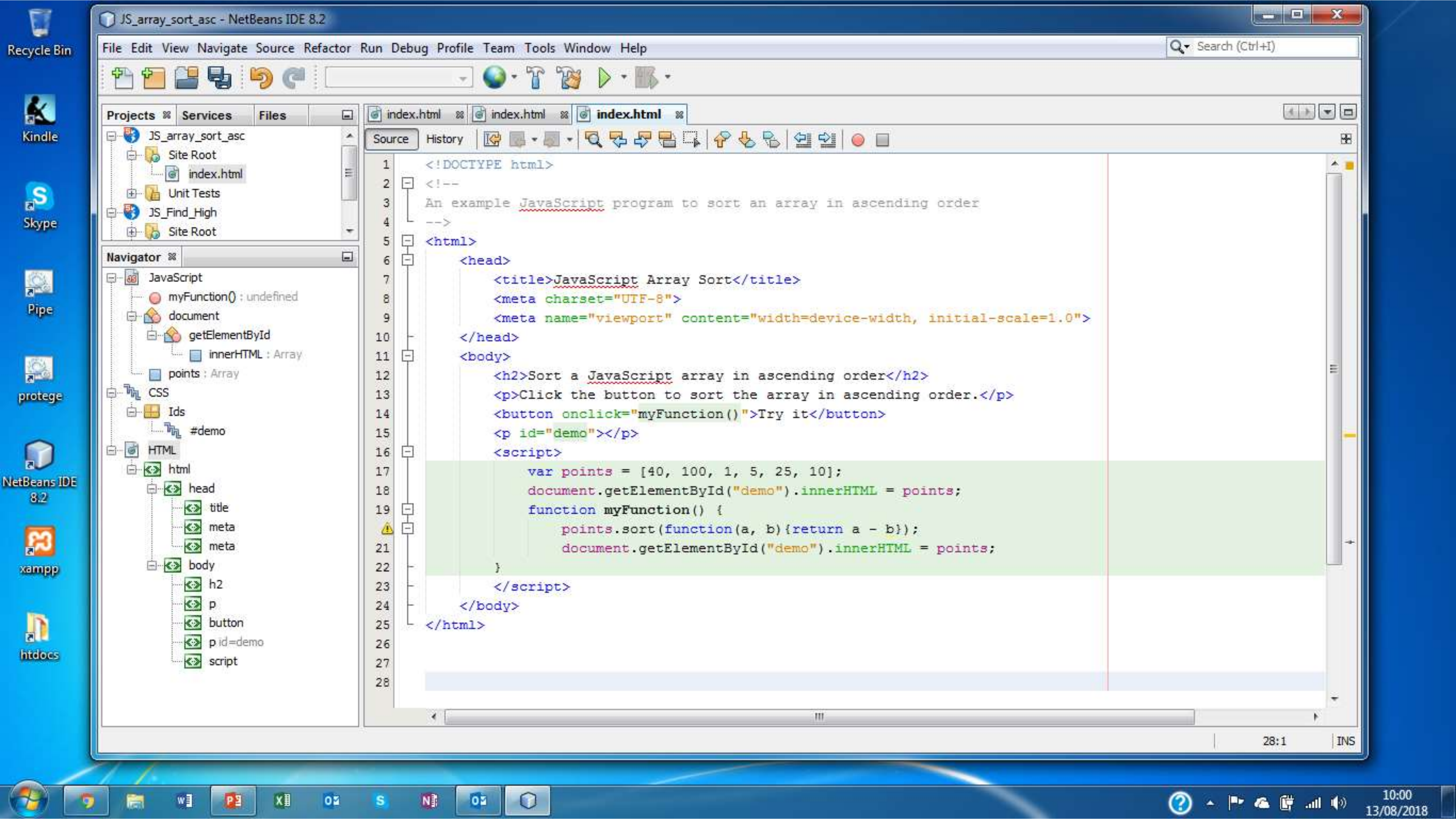
## Ascending Sort

# Sort in Ascending Order

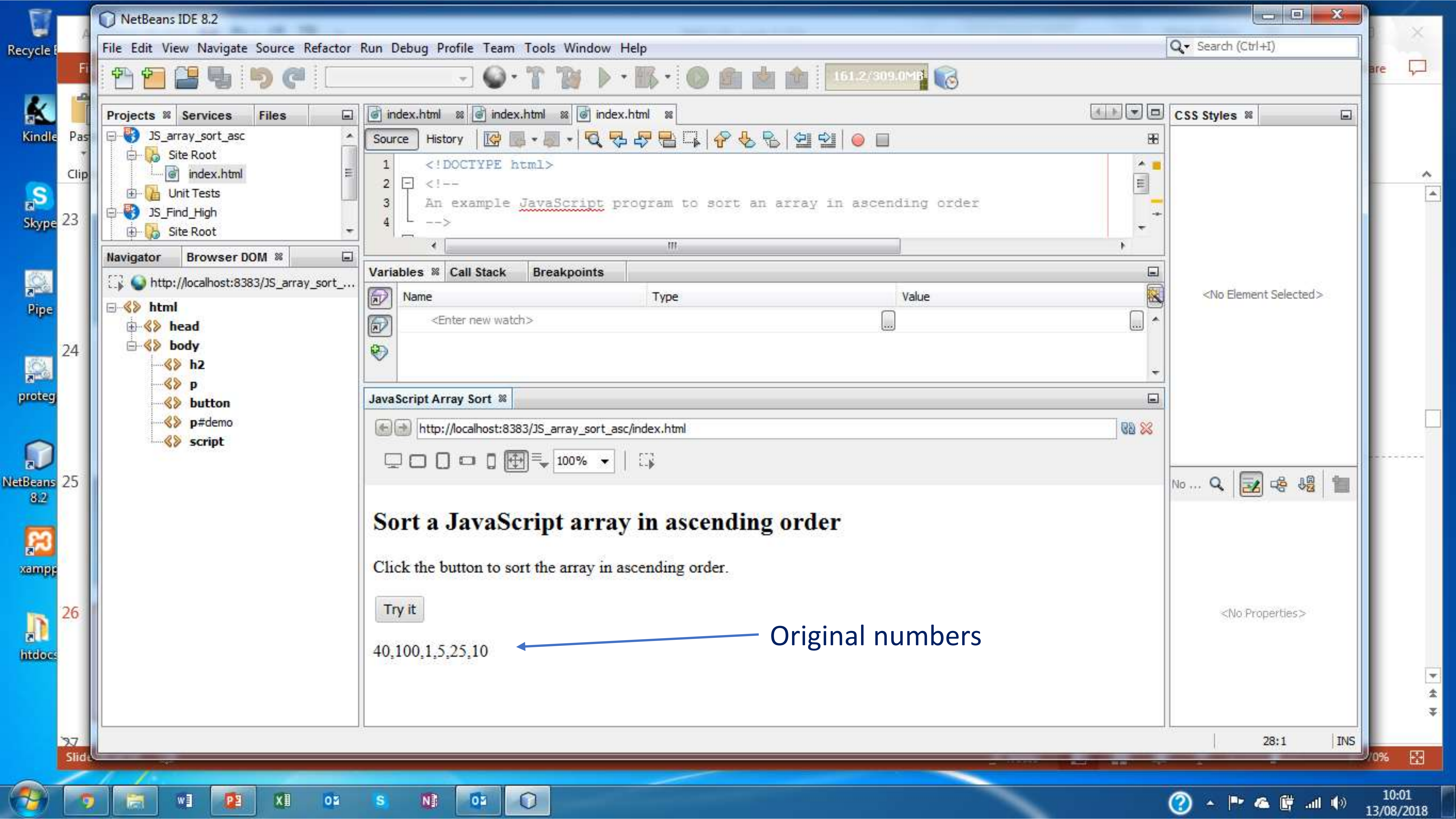
- Array sort in ascending order ()

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return a - b});
```

- Following the array sort **`points[0]`** contains the highest value
- The following slides show a worked example of array sort in ascending order



```
<!DOCTYPE html>
<!--
An example JavaScript program to sort an array in ascending order
-->
<html>
    <head>
        <title>JavaScript Array Sort</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h2>Sort a JavaScript array in ascending order</h2>
        <p>Click the button to sort the array in ascending order.</p>
        <button onclick="myFunction()">Try it</button>
        <p id="demo"></p>
        <script>
            var points = [40, 100, 1, 5, 25, 10];
            document.getElementById("demo").innerHTML = points;
            function myFunction() {
                points.sort(function(a, b){return a - b});
                document.getElementById("demo").innerHTML = points;
        }
        </script>
    </body>
</html>
```



```html
<!DOCTYPE html>
<!--
An example JavaScript program to sort an array in ascending order
-->
```

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

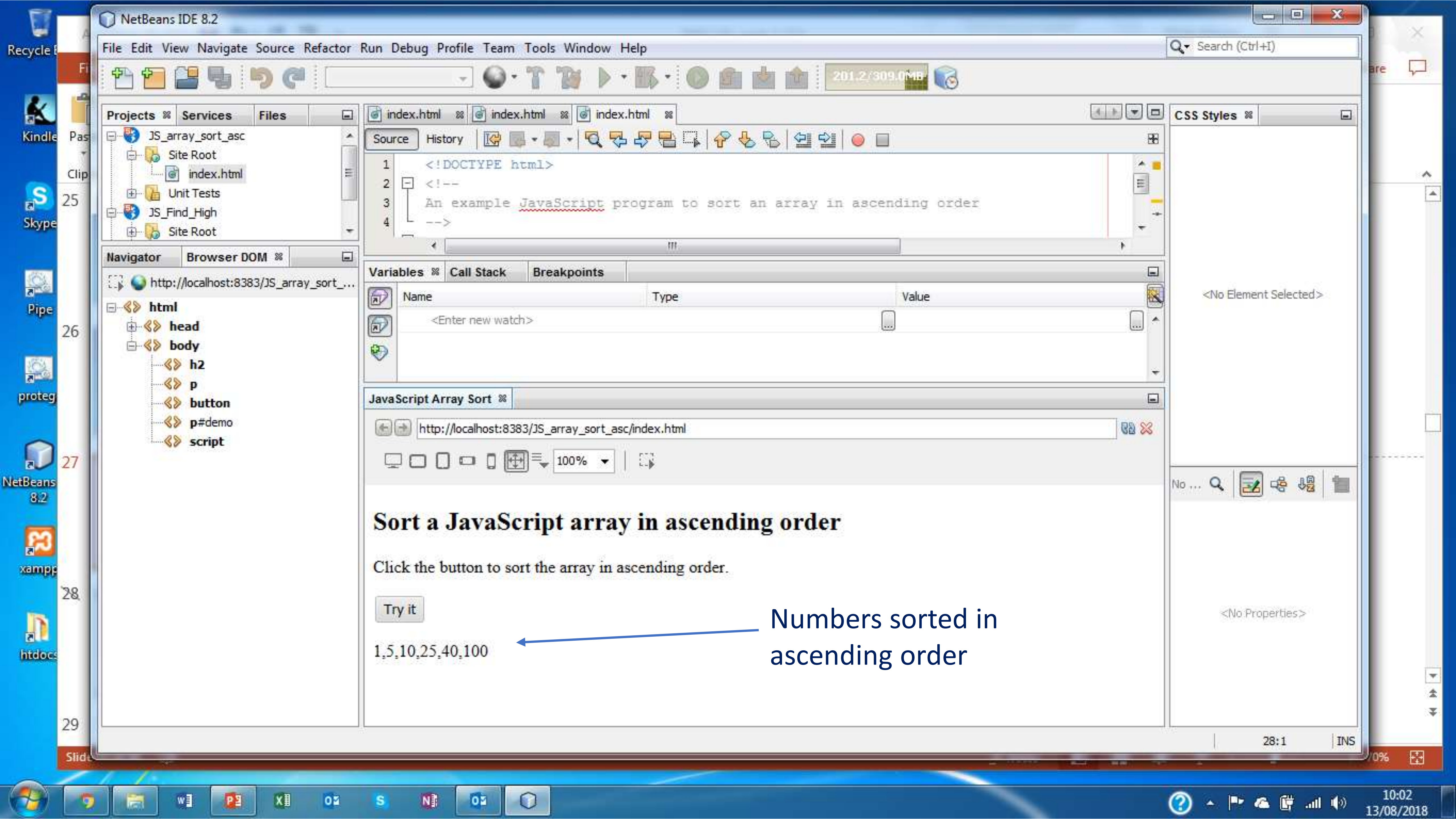
**Sort a JavaScript array in ascending order**

Click the button to sort the array in ascending order.

Try it

40,100,1,5,25,10

Original numbers



NetBeans IDE 8.2

```
<!DOCTYPE html>
<!--
An example JavaScript program to sort an array in ascending order
-->
```

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

**Sort a JavaScript array in ascending order**

Click the button to sort the array in ascending order.

Try it

```
1,5,10,25,40,100
```

Numbers sorted in ascending order

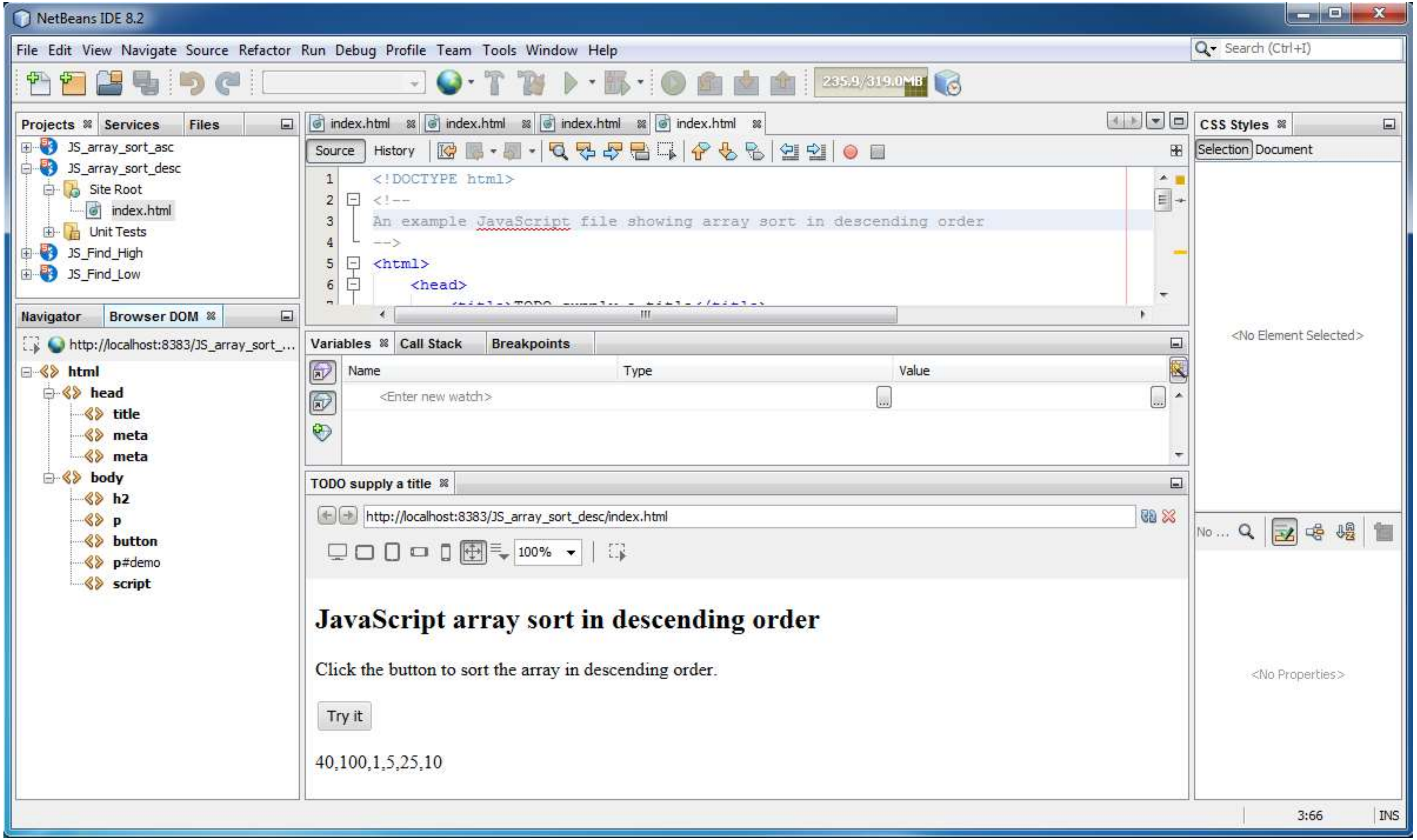
## Descending Sort

## Sort in Descending Order

- Array sort descending order ()

```
var points = [40, 100, 1, 5, 25, 10];
points.sort(function(a, b){return b - a});
```

  - Following the array sort **`points[0]`** contains the lowest value
  - and [**`points.length-1`**] contains the lowest value
- The following slides show a worked example of array sort in descending order



```
<!DOCTYPE html>
<!--
An example JavaScript file showing array sort in descending order
-->
<html>
    <head>
```

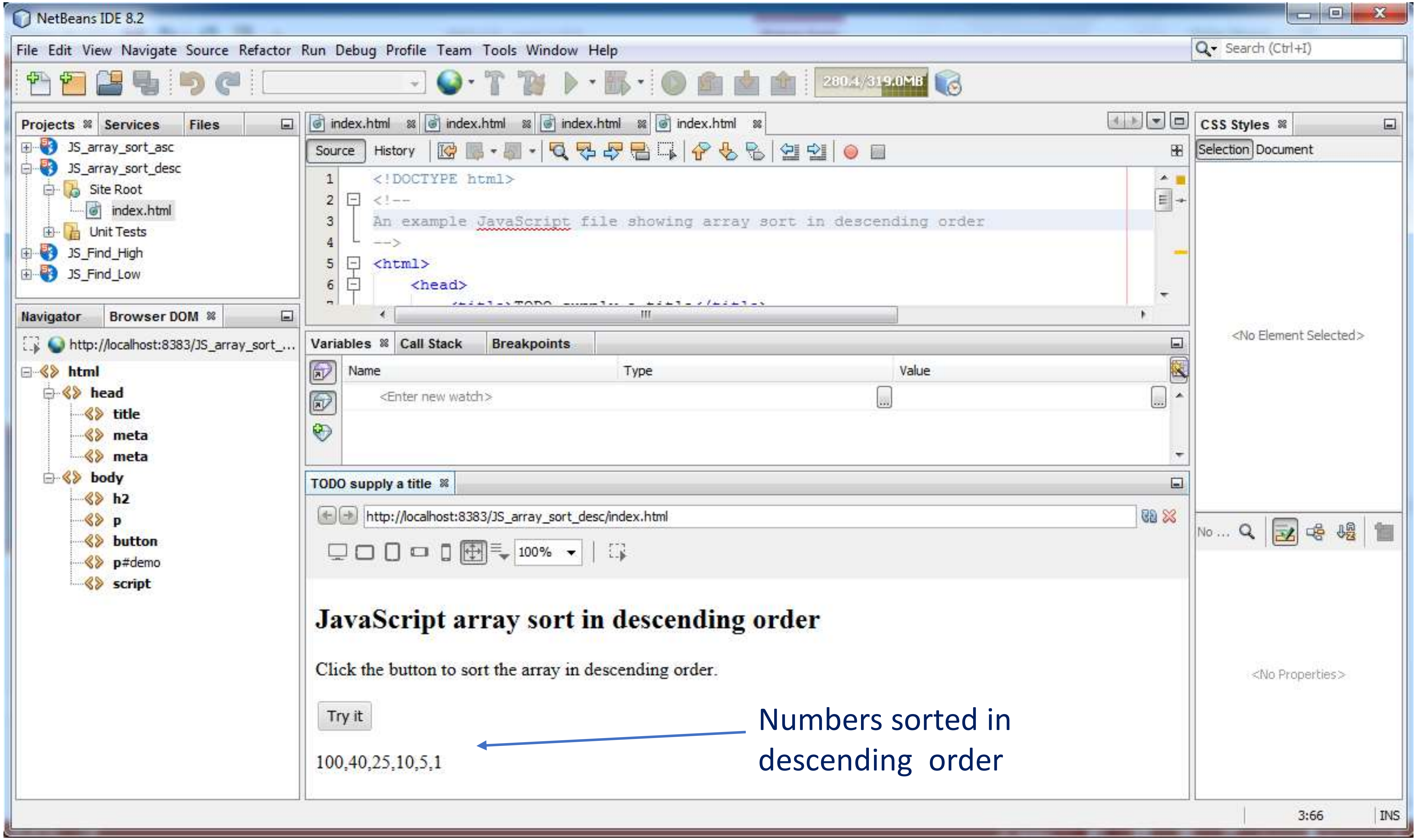
http://localhost:8383/JS_array_sort_desc/index.html

**JavaScript array sort in descending order**

Click the button to sort the array in descending order.

Try it

40,100,1,5,25,10



```
<!DOCTYPE html>
<!--
An example JavaScript file showing array sort in descending order
-->
<html>
    <head>
```

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

**JavaScript array sort in descending order**

Click the button to sort the array in descending order.

Try it

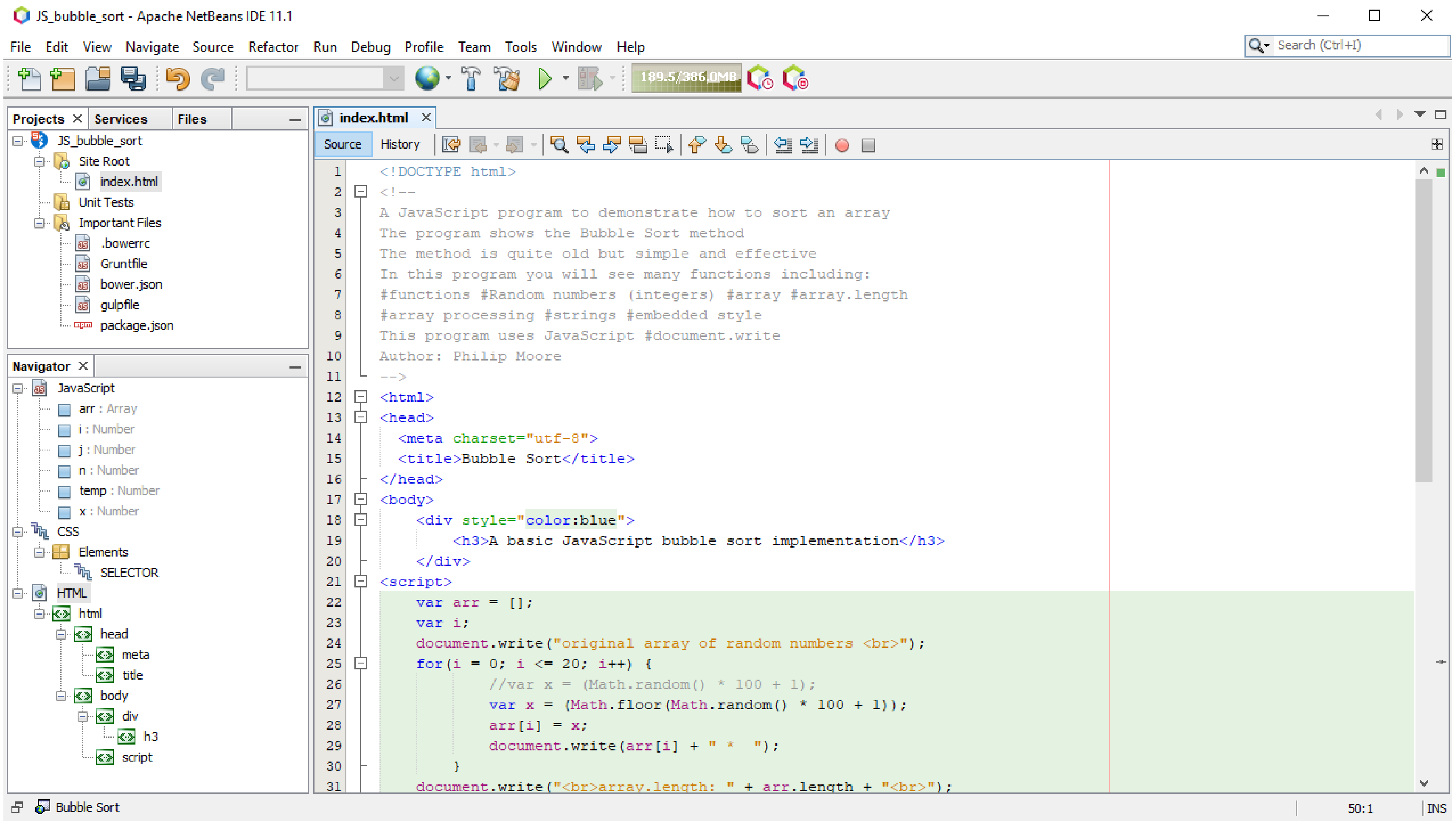
100,40,25,10,5,1

Numbers sorted in descending order

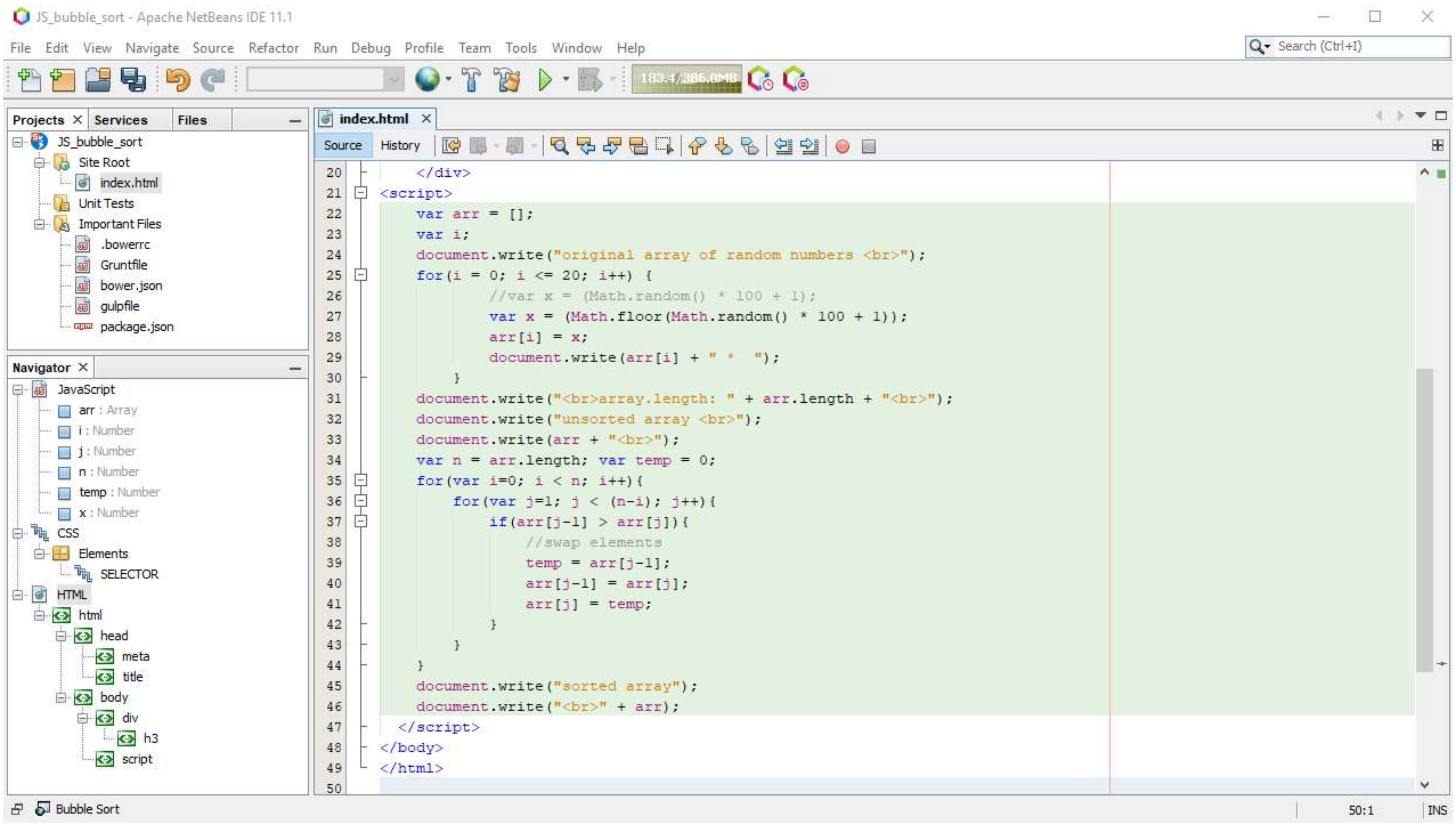
## Bubble Sort

## Bubble Sort

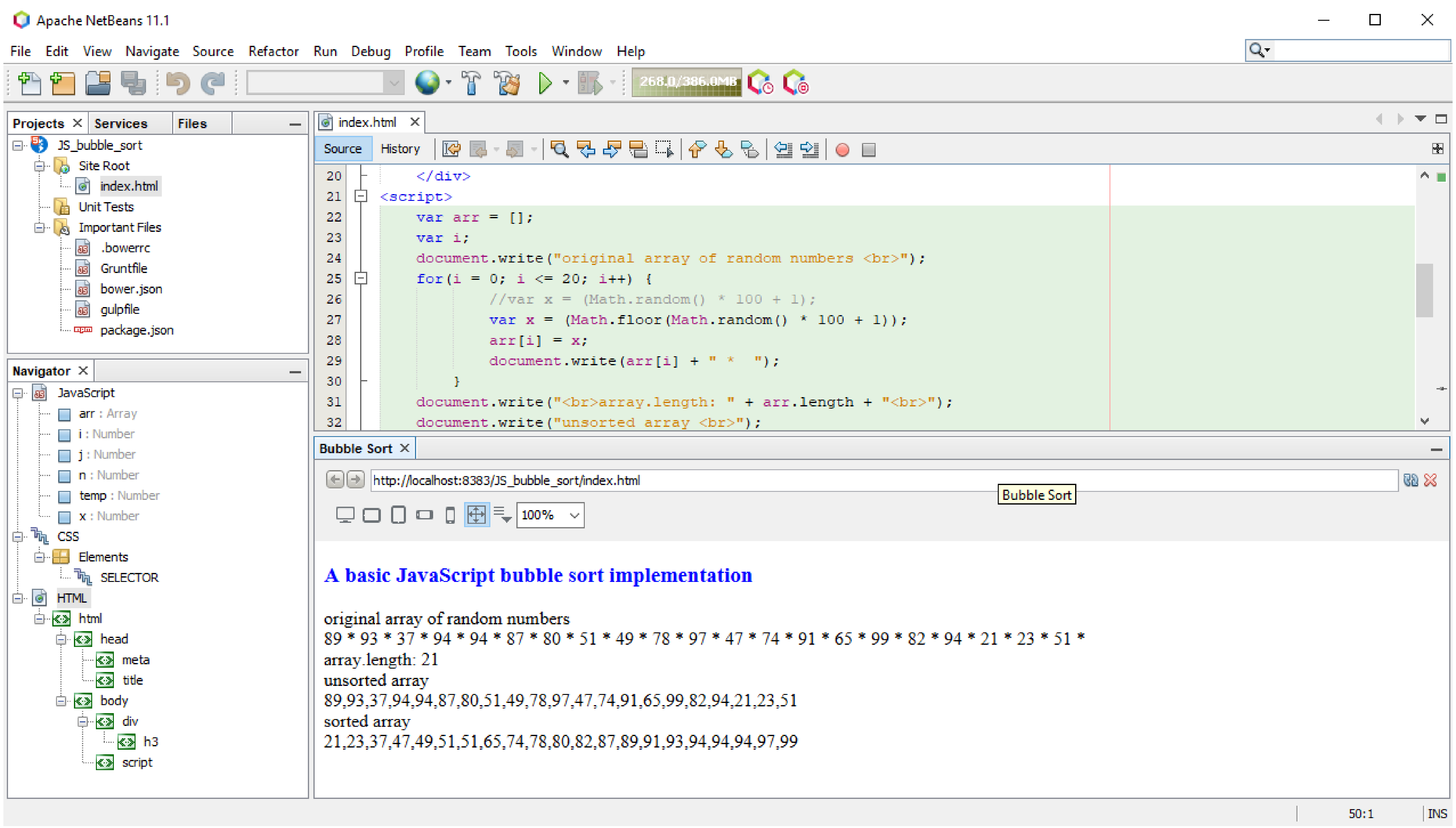
- The previous slides have shown the methods built into JavaScript to sort arrays
  - However: the worked example show the output is correct but the methodology `points.sort(function(a, b){return a - b});` is unclear
- In the following slides I provide:
  - A simple JavaScript Bubble Sort program to demonstrate how an array sort may work (implementations may vary between web browsers)
  - In this program you will see many functions including: #functions #Random numbers (integers) #arrays #nested arrays #array.length #array processing #strings. This program uses #embedded style and the JavaScript #document.write method



```html
<!DOCTYPE html>
<!--
A JavaScript program to demonstrate how to sort an array
The program shows the Bubble Sort method
The method is quite old but simple and effective
In this program you will see many functions including:
#functions #Random numbers (integers) #array #array.length
#array processing #strings #embedded style
This program uses JavaScript #document.write
Author: Philip Moore
-->
<html>
<head>
  <meta charset="utf-8">
  <title>Bubble Sort</title>
</head>
<body>
    <div style="color:blue">
        <h3>A basic JavaScript bubble sort implementation</h3>
    </div>
<script>
    var arr = [];
    var i;
    document.write("original array of random numbers <br>");
    for(i = 0; i <= 20; i++) {
            //var x = (Math.random() * 100 + 1);
            var x = (Math.floor(Math.random() * 100 + 1));
            arr[i] = x;
            document.write(arr[i] + " *  ");
        }
    document.write("<br>array.length: " + arr.length + "<br>");
```



```
    </div>
<script>
    var arr = [];
    var i;
    document.write("original array of random numbers <br>");
    for(i = 0; i <= 20; i++) {
            //var x = (Math.random() * 100 + 1);
            var x = (Math.floor(Math.random() * 100 + 1));
            arr[i] = x;
            document.write(arr[i] + " *  ");
        }
    document.write("<br>array.length: " + arr.length + "<br>");
    document.write("unsorted array <br>");
    document.write(arr + "<br>");
    var n = arr.length; var temp = 0;
    for(var i=0; i < n; i++){
        for(var j=1; j < (n-i); j++){
            if(arr[j-1] > arr[j]){
                //swap elements
                temp = arr[j-1];
                arr[j-1] = arr[j];
                arr[j] = temp;
            }
        }
    }
    document.write("sorted array");
    document.write("<br>" + arr);
  </script>
</body>
</html>
```



```
    </div>
<script>
    var arr = [];
    var i;
    document.write("original array of random numbers <br>");
    for(i = 0; i <= 20; i++) {
        //var x = (Math.random() * 100 + 1);
        var x = (Math.floor(Math.random() * 100 + 1));
        arr[i] = x;
        document.write(arr[i] + " *  ");
    }
    document.write("<br>array.length: " + arr.length + "<br>");
    document.write("unsorted array <br>");
```

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

**A basic JavaScript bubble sort implementation**

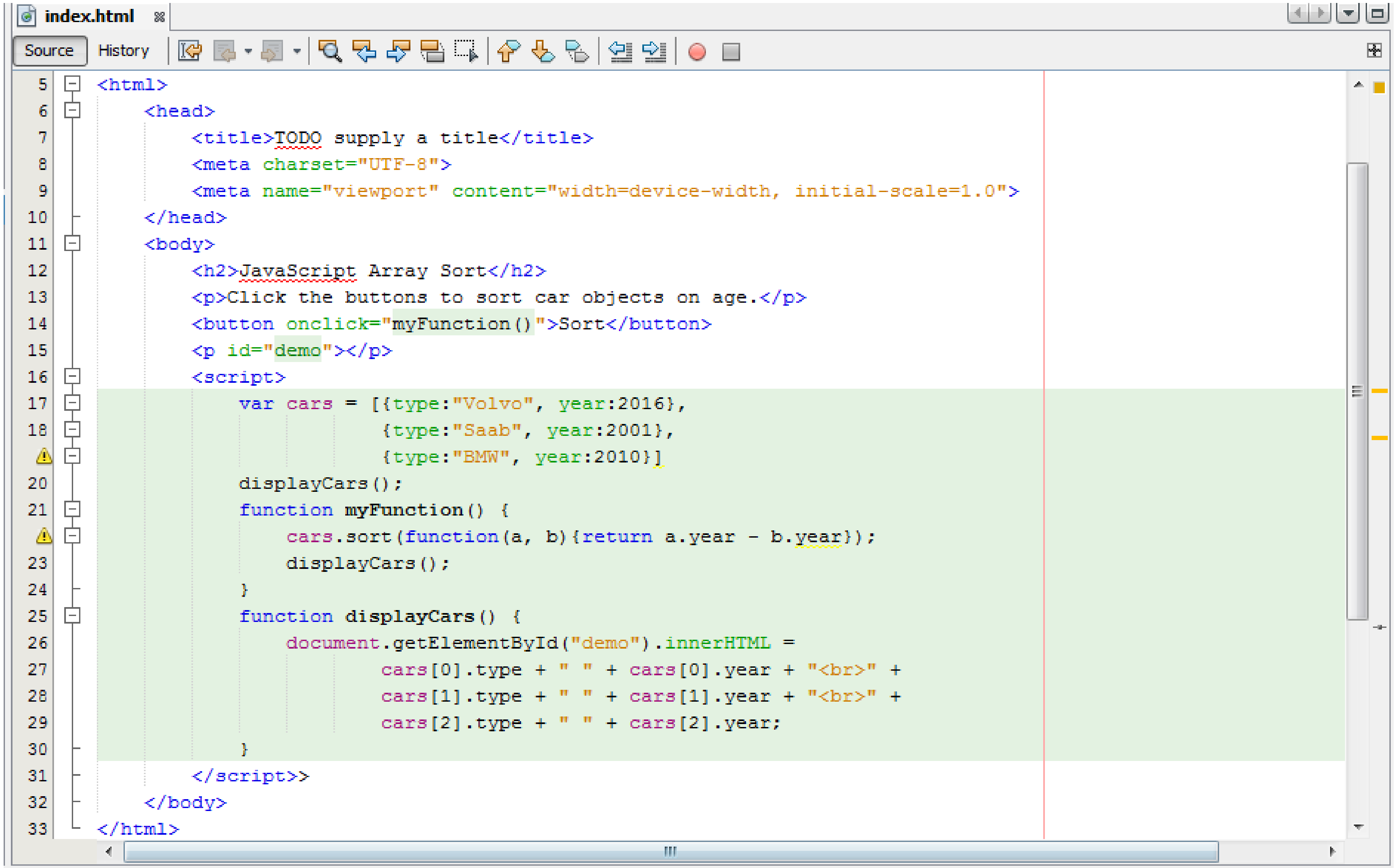
```
original array of random numbers
89 * 93 * 37 * 94 * 94 * 87 * 80 * 51 * 49 * 78 * 97 * 47 * 74 * 91 * 65 * 99 * 82 * 94 * 21 * 23 * 51 *
array.length: 21
unsorted array
89,93,37,94,94,87,80,51,49,78,97,47,74,91,65,99,82,94,21,23,51
sorted array
21,23,37,47,49,51,51,65,74,78,80,82,87,89,91,93,94,94,94,97,99
```

## Object Sort

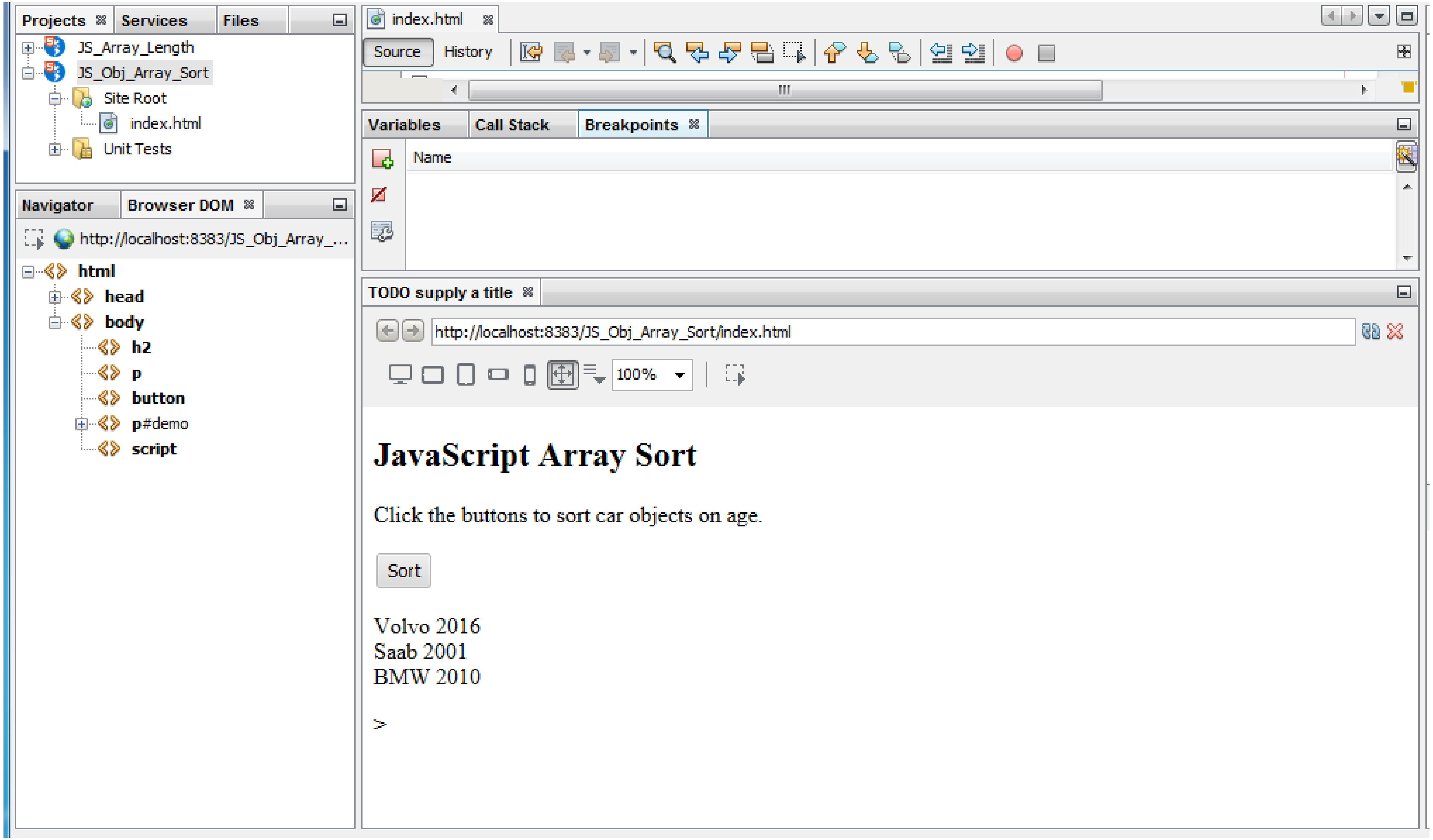
## Object Array Sort

- JavaScript arrays may hold a range of simple datatypes including objects
- For example

-
  ```
  var cars = [
      {type:"Volvo", year:2016},
      {type:"Saab", year:2001},
      {type:"BMW", year:2010}];
  ```
- The following slide shows the object array sort



```
<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>
        <h2>JavaScript Array Sort</h2>
        <p>Click the buttons to sort car objects on age.</p>
        <button onclick="myFunction()">Sort</button>
        <p id="demo"></p>
        <script>
            var cars = [{type:"Volvo", year:2016},
                        {type:"Saab", year:2001},
                        {type:"BMW", year:2010}]
            displayCars();
            function myFunction() {
                cars.sort(function(a, b){return a.year - b.year});
                displayCars();
            }
            function displayCars() {
                document.getElementById("demo").innerHTML =
                        cars[0].type + " " + cars[0].year + "<br>" +
                        cars[1].type + " " + cars[1].year + "<br>" +
                        cars[2].type + " " + cars[2].year;
            }
        </script>>
    </body>
</html>
```



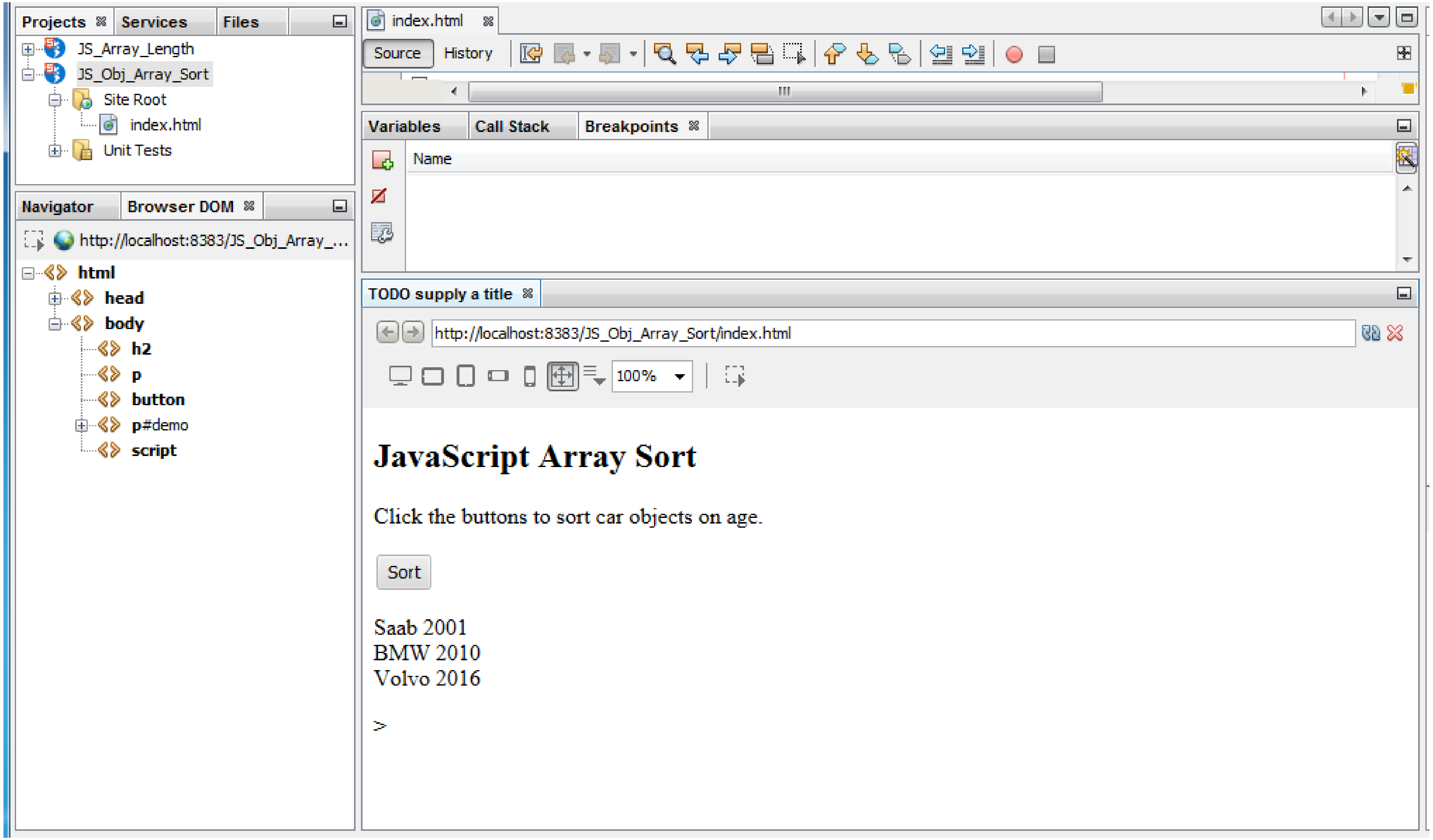
JavaScript Array Sort

Click the buttons to sort car objects on age.

Sort

Volvo 2016
Saab 2001
BMW 2010

>



JavaScript Array Sort

Click the buttons to sort car objects on age.

Sort

```
Saab 2001
BMW 2010
Volvo 2016

>
```

## review

- In this tutorial we have considered:
  - Making changes to arrays and array elements
  - Arrays and strings
  - Sorting arrays
  - A brief overview of the algorithmic approach the array sorting with a worked example showing how the JavaScript elements combine to create a working program