# Introduction To Full-Stack Web Development

**CS 386**

**Michael Kremer**

Last updated: 10/1/2023 6:33:10 PM

- 11.1 Introduction to Objects
- 11.2 Arrays
- 11.3 Functions
- 11.4 CRUD Operations

# Class 11

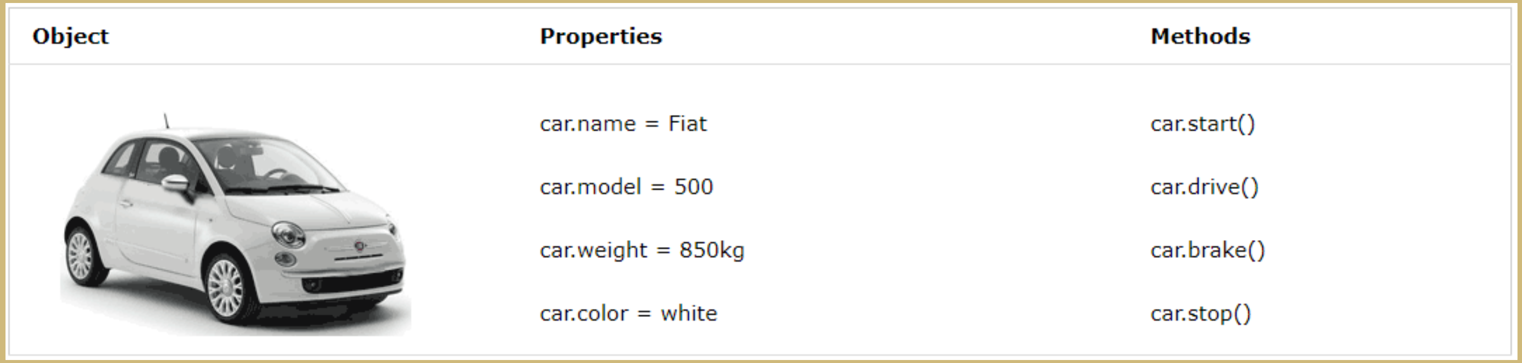
## 11.1 Introduction to Objects

- What is an Object?

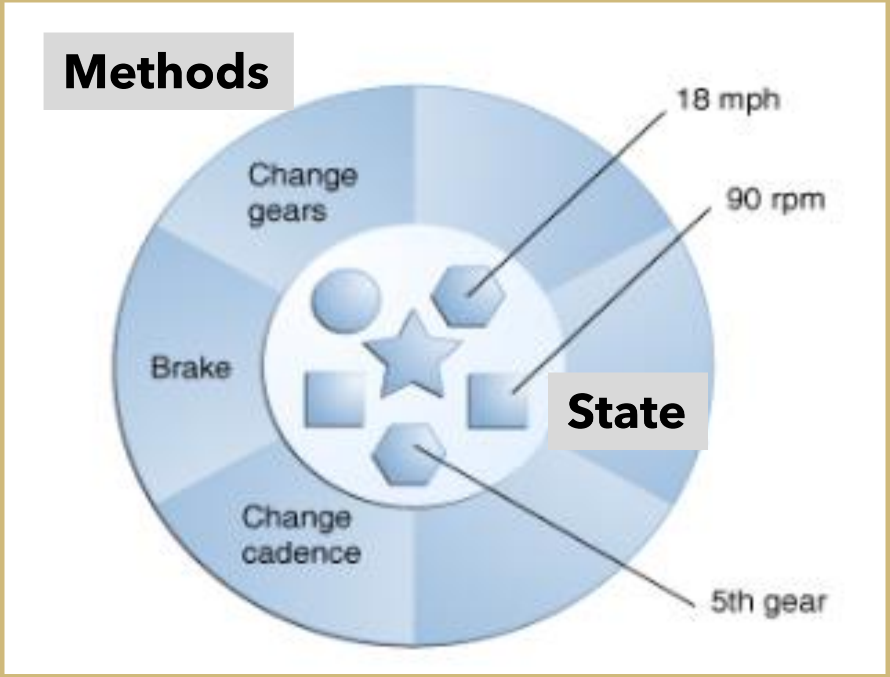
  - Real-world objects share two characteristics:
    - They all have state and behavior
  - Object stores its state in fields (variables, properties in some programming languages)
  - Exposes its behavior through methods (functions in some programming languages)

# 11.1 Introduction to Objects

- <u>Examples:</u>
- Bicycles
  - They have state (current gear, current pedal cadence, current speed)
  - They exhibit behavior (changing gear, changing pedal cadence, applying brakes)
- Cars
  - They have state (brand, model, weight, color)
  - The exhibit behavior (start, stop, drive, brake)



| Object | Properties | Methods |
|---|---|---|
| | car.name = Fiat<br>car.model = 500<br>car.weight = 850kg<br>car.color = white | car.start()<br>car.drive()<br>car.brake()<br>car.stop() |



## 11.1 Introduction to Objects

- To store state and behavior in objects, use properties
- Every object contains one or more property/value pairs:
  - For state, value is static (primitive value, or another object reference)
  - For behavior, value is function (covered next), executes and acts on state
- Arrays (covered next) are specialized objects
- Objects:
  - let obj = { prop1: val1, prop2: val2 ..propn: valn}
- Arrays:
  - let arr = [ val1, val2, … valn]
  - Values are called elements and are indexed starting with 0
  - Index 0 = val1, index 1 = val2 ..
  - Indices are “properties” (automatically maintained), elements are values

## 11.2 Arrays

- Array is object type in JavaScript:
  - Array is list of zero or more expressions
  - Each of which represents an array element
  - Elements are indexed starting at 0
  - Enclosed in square brackets ([])
- Array variable is initialized with specified values as its elements
  - Called literal syntax
  - Can also use variables as elements

**Syntax:**
**let** *var_array* **= [** *el1, el2, el3, ..eln* **]**

- Length is set to number of elements specified
- Example:
  - Coffees array with three elements, length of three, indices 0, 1, 2:
  - let coffees = ['French Roast', 'Colombian', 'Kona'];

# 11.2 Arrays

- Reading/Writing Array Elements
  - Access elements of array using [] operator

    **Syntax:**
    *var_array*[*index*]

  - Reference to array (variable) should appear on left side of brackets
  - Non-negative integer or variable/expression should be inside brackets
  - Reading array element:
    - Element appears on right side of assignment expression
    - **Remember:** Assignment is right-associative!!

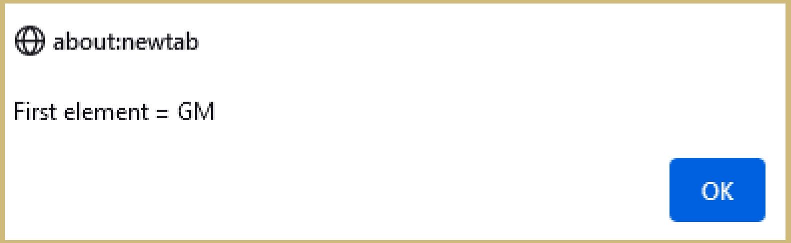
    **Syntax:**
    **let** *var* **=** *var_array*[*index*]

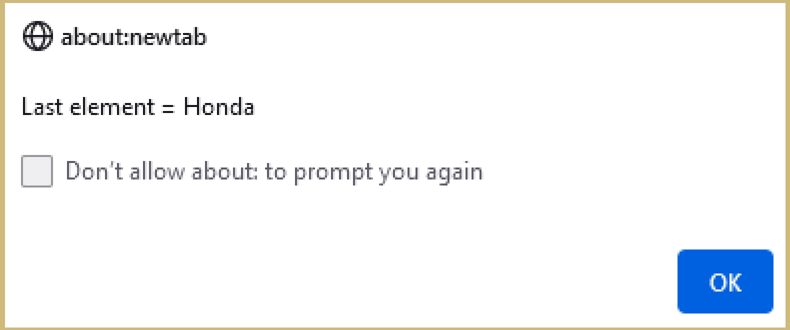
  - Writing array element:
    - Element appears on left side of assignment expression
    - New value or expression on right side

    **Syntax:**
    *var_array*[*index*] **=** *val/exp*

# 11.2 Arrays

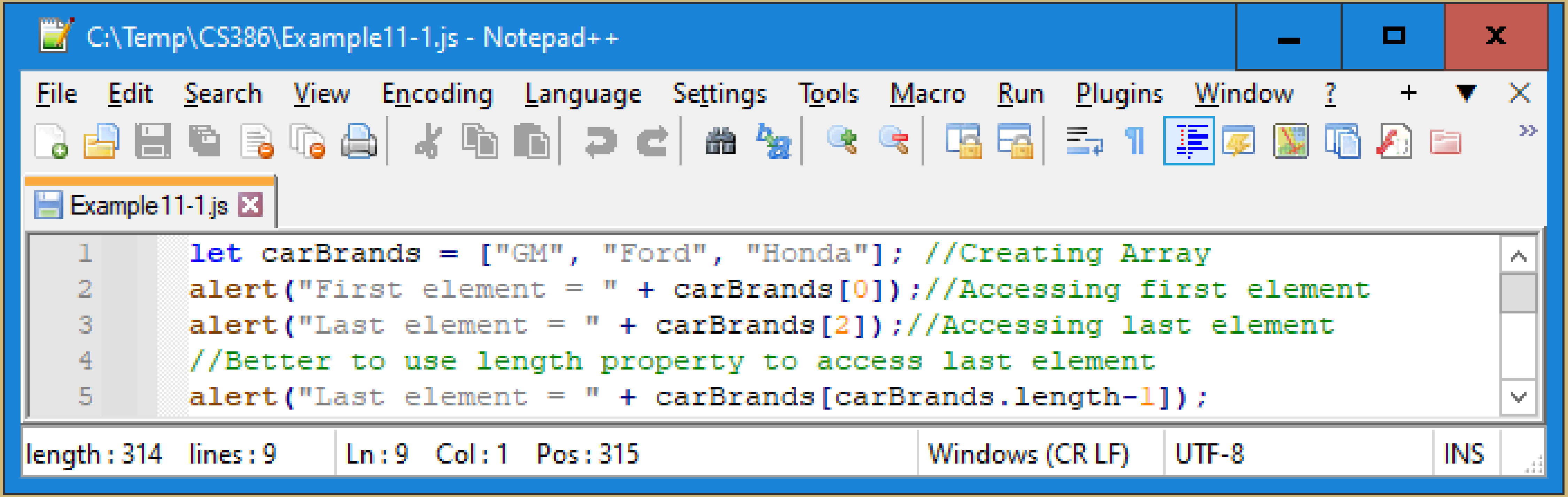
- **Example 11-1:**
  - ❑ Create array named carBrands and assign 3 elements
  - ❑ In alert/console, show first element
  - ❑ In second alert/console, show last element


```
about:newtab

First element = GM
```


```
about:newtab

Last element = Honda
```

## 11.2 Arrays

➢ **Example 11-1:**

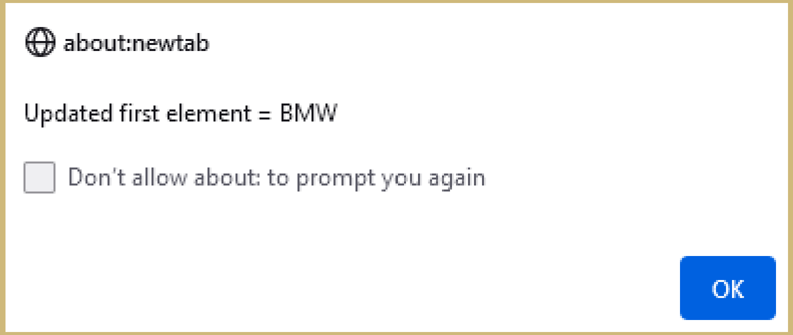

C:\Temp\CS386\Example11-1.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-1.js

```
let carBrands = ["GM", "Ford", "Honda"]; //Creating Array
alert("First element = " + carBrands[0]);//Accessing first element
alert("Last element = " + carBrands[2]);//Accessing last element
//Better to use length property to access last element
alert("Last element = " + carBrands[carBrands.length-1]);
```

length : 314 lines : 9 Ln : 9 Col : 1 Pos : 315 Windows (CR LF) UTF-8 INS

# 11.2 Arrays

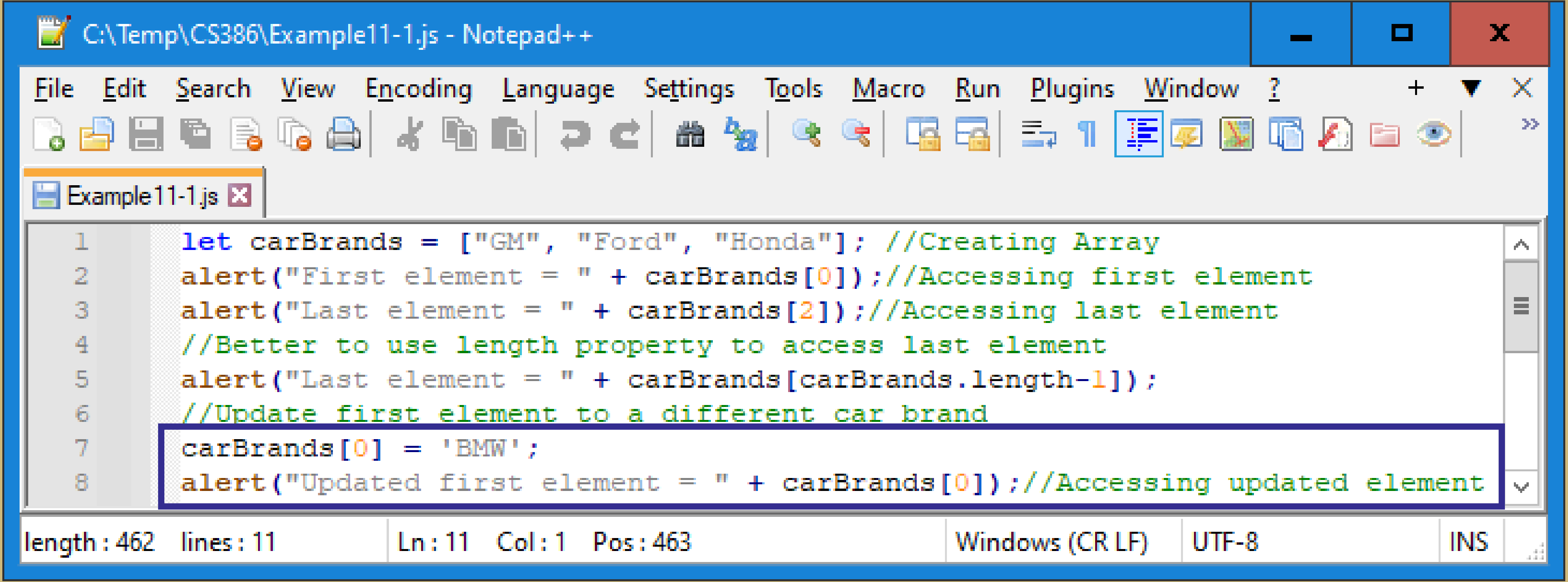
- **Example 11-1(continued):**
  - ❑ Update first element to different car brand
  - ❑ Display updated value in alert/console



```
about:newtab

Updated first element = BMW
```

# 11.2 Arrays

- **Example 11-1(continued):**


C:\Temp\CS386\Example11-1.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-1.js

```
let carBrands = ["GM", "Ford", "Honda"]; //Creating Array
alert("First element = " + carBrands[0]);//Accessing first element
alert("Last element = " + carBrands[2]);//Accessing last element
//Better to use length property to access last element
alert("Last element = " + carBrands[carBrands.length-1]);
//Update first element to a different car brand
carBrands[0] = 'BMW';
alert("Updated first element = " + carBrands[0]);//Accessing updated element
```

```
length : 462    lines : 11    Ln : 11    Col : 1    Pos : 463    Windows (CR LF)    UTF-8    INS
```

# 11.2 Arrays

- Adding/Deleting Array Elements
  - Simplest way to add elements to array:
    - Just assign values to new (consecutive) indices
  - Adding new element at end of array, use length property as new index
  - Length is always last index + 1 since index starts with 0
  - Example:
    - Array with 3 elements:
    - First index = 0, last index = 2
    - Length = 3
  - Also can use push() method to add value at end of array
  - Additional functionality:
    - Allows for more than element to be added
    - Comma separate multiple elements

**Syntax:**
*var_array***[**var_array***.length*****] =** *val/exp*

**Syntax:**
*var_array***.push(** *element[s]* **)**

# 11.2 Arrays

- Adding/Deleting Array Elements
  - Method unshift
    - Allows for adding one or more elements to beginning of array
    - Shifts existing elements to the "right"
    - Automatically updates indices
  - Method pop
    - Use pop method to delete last element at end of array
  - Method shift
    - Use shift method to delete first element at beginning of array
    - Shifting existing elements to "left"
    - Automatically updates indices
  - Operator delete
    - Deletes array elements, no shifting takes place
    - **Warning:** Leaves holes in array (sparse array), elements are undefined

**Syntax:**
*var_array*.**unshift(** *element[s]* **)**

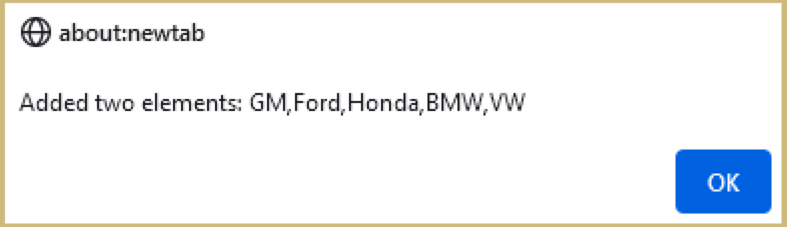
**Syntax:**
*var_array*.**pop()**

**Syntax:**
*var_array*.**shift()**

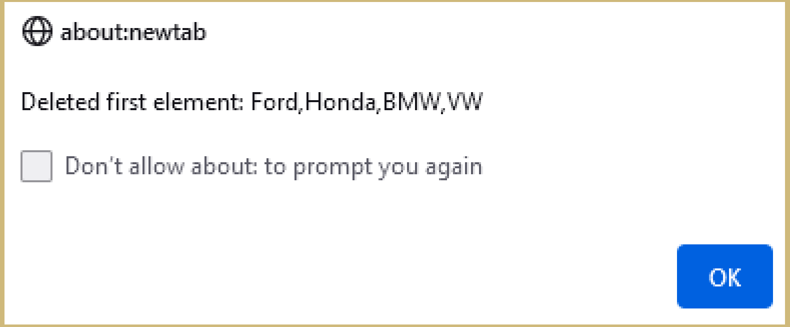
**Syntax:**
**delete** *var_array*[*index*]

# 11.2 Arrays

- **Example 11-2:**
  - ❑ Use array carBrands from previous example
  - ❑ Add two more brands to array carBrands at end of array
  - ❑ Display entire array in alert/console (use array variable in alert)
  - ❑ Delete first element
  - ❑ Display entire array in alert/console (use array variable in alert)
  - ❑ Delete last element
  - ❑ Display entire array in alert/console (use array variable in alert)

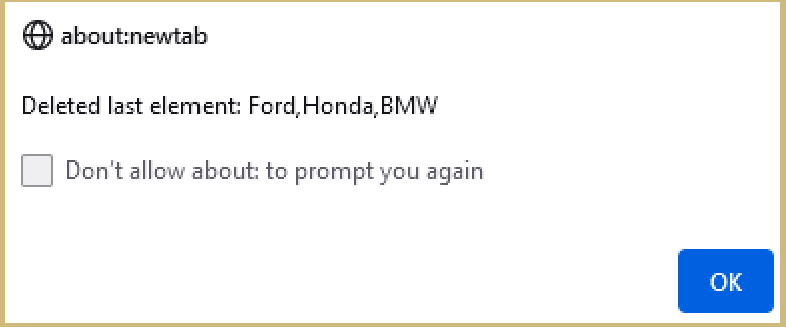

about:newtab

Added two elements: GM,Ford,Honda,BMW,VW


about:newtab

Deleted first element: Ford,Honda,BMW,VW

Don't allow about: to prompt you again

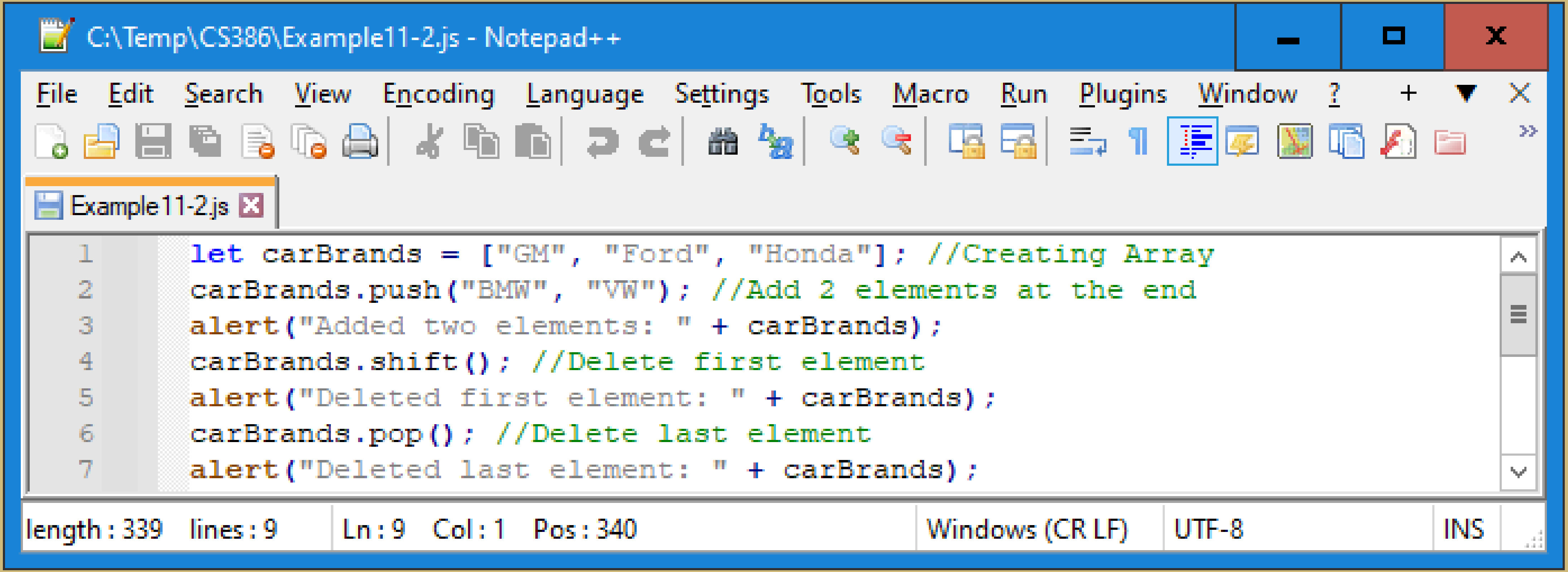

about:newtab

Deleted last element: Ford,Honda,BMW

Don't allow about: to prompt you again

# 11.2 Arrays

- **Example 11-2:**


C:\Temp\CS386\Example11-2.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-2.js

```
let carBrands = ["GM", "Ford", "Honda"]; //Creating Array
carBrands.push("BMW", "VW"); //Add 2 elements at the end
alert("Added two elements: " + carBrands);
carBrands.shift(); //Delete first element
alert("Deleted first element: " + carBrands);
carBrands.pop(); //Delete last element
alert("Deleted last element: " + carBrands);
```

length : 339 lines : 9 Ln : 9 Col : 1 Pos : 340 Windows (CR LF) UTF-8 INS

## 11.2 Arrays

- Iterating Arrays
  - Most common way to loop through elements of array is with for loop
  - Use for loop iterator variable to access elements by index

**Syntax:**

```
for ( let i = 0; i < var_array.length; i++) {
    var_array[i]; //Accessing array element
    statements
}
```

  - Condition expression (second part in for loop) is evaluated for every loop
  - For performance reason, evaluate length of array only once in initial expression of for loop

**Syntax:**

```
for (let i = 0, len = var_array.length; i < len; i++) {
    var_array[i]; //Accessing array element
     statements
}
```

# 11.2 Arrays

- Iterating Arrays
  - Two other **for** loops for arrays:
    - for .. in
      - Loops over indices
      - No need to specify completion condition
    - for .. of
      - Loops over element values instead of indices
      - No need to specify completion condition

**Syntax:**

**for ( let** *var_index* **in** *var_array***) {**
    *statements*
**}**

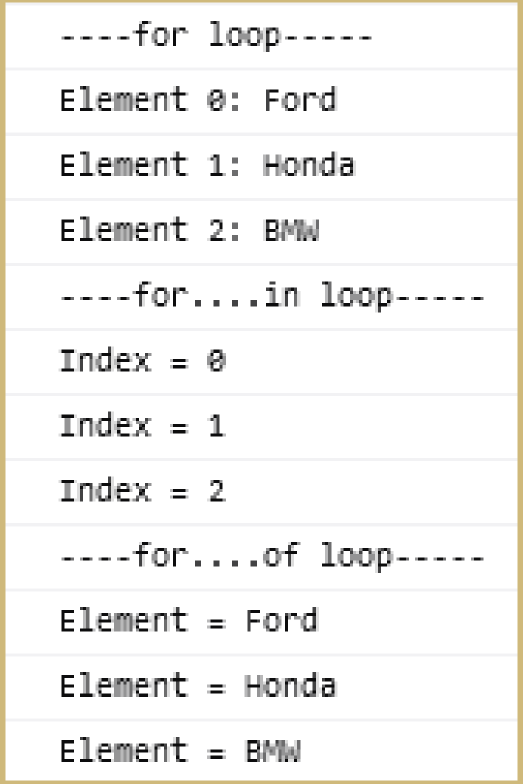
**Syntax:**

**for (let** *var_element* **of** *var_array***) {**
    *statements*
**}**

# 11.2 Arrays

➢ **<u>Example 11-3:</u>**

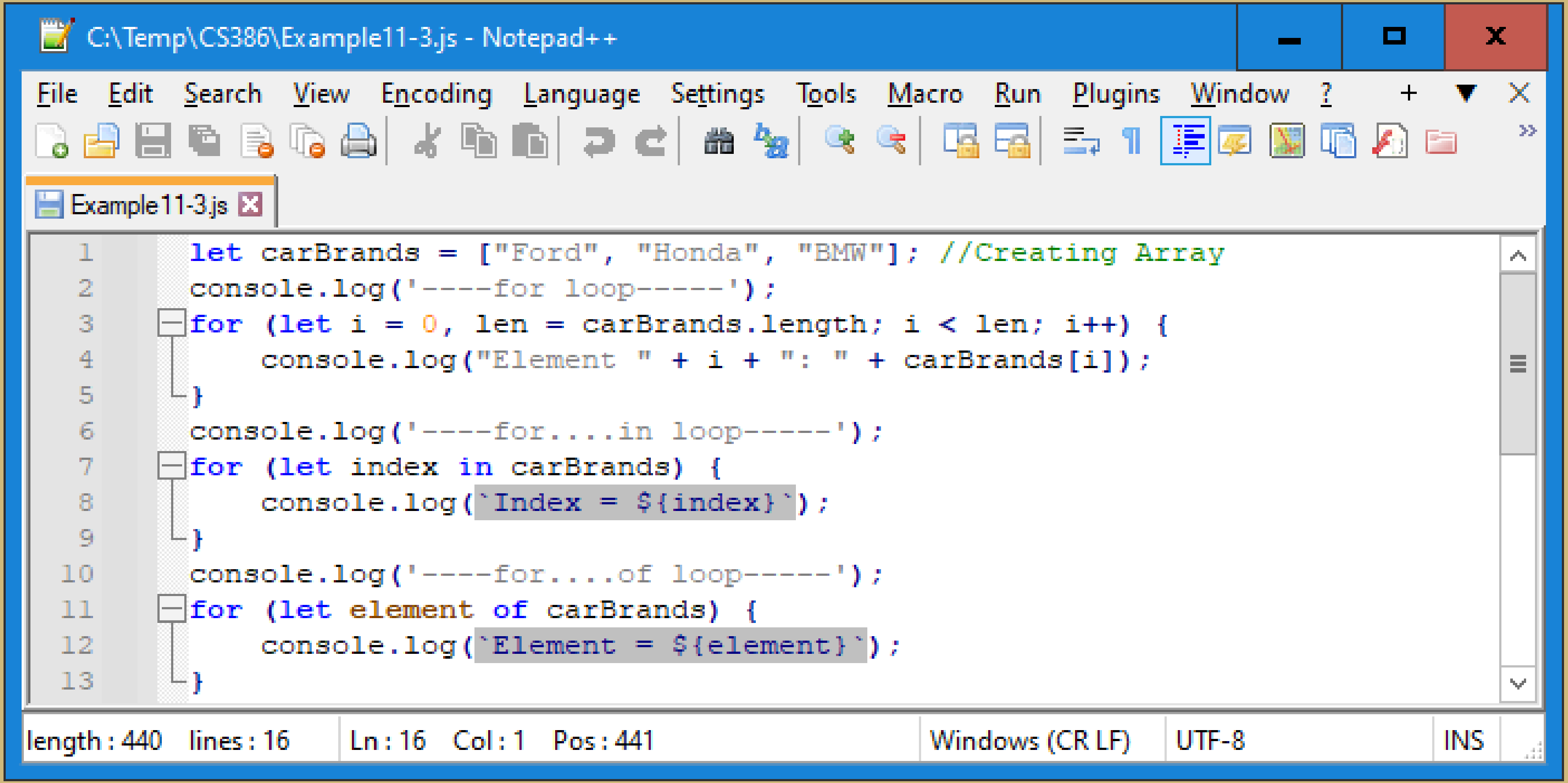
- ❑ Loop over array carBrands = ["Ford", "Honda", "BMW"] using for loop
- ❑ Display each element and corresponding index in console
- ❑ Then use for .. in loop and display index
- ❑ Finally use for .. of loop and display element value



```
----for loop-----
Element 0: Ford
Element 1: Honda
Element 2: BMW
----for....in loop-----
Index = 0
Index = 1
Index = 2
----for....of loop-----
Element = Ford
Element = Honda
Element = BMW
```

# 11.2 Arrays

➢ **Example 11-3:**


C:\Temp\CS386\Example11-3.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-3.js

```
let carBrands = ["Ford", "Honda", "BMW"]; //Creating Array
console.log('----for loop-----');
for (let i = 0, len = carBrands.length; i < len; i++) {
    console.log("Element " + i + ": " + carBrands[i]);
}
console.log('----for....in loop-----');
for (let index in carBrands) {
    console.log(`Index = ${index}`);
}
console.log('----for....of loop-----');
for (let element of carBrands) {
    console.log(`Element = ${element}`);
}
```

length : 440 lines : 16 Ln : 16 Col : 1 Pos : 441 Windows (CR LF) UTF-8 INS

## 11.2 Arrays

- Array Properties/Methods

| Name | Description |
|---|---|
| concat() | Joins arrays and returns an array with the joined arrays |
| forEach() | Calls a function for each array element |
| join([sep]) | Joins all elements of an array into a string, optional specify separator |
| length | Sets or returns the number of elements in an array |
| map() | Creates a new array with the result of calling a function for each array element |
| pop() | Removes the last element of an array, and returns that element |
| push() | Adds new elements to the end of an array, and returns the new length |
| reduce() | Reduce the values of an array to a single value (going left-to-right) |
| reduceRight() | Reduce the values of an array to a single value (going right-to-left) |
| reverse() | Reverses the order of the elements in an array |
| shift() | Removes the first element of an array, and returns that element |
| slice() | Selects a part of an array, and returns the new array |
| some() | Checks if any of the elements in an array pass a test |
| sort() | Sorts the elements of an array |
| splice() | Adds/Removes elements from an array |
| toString() | Converts an array to a string, and returns the result |
| unshift() | Adds new elements to the beginning of an array, and returns the new length |
| valueOf() | Returns the primitive value of an array |

# 11.2 Arrays

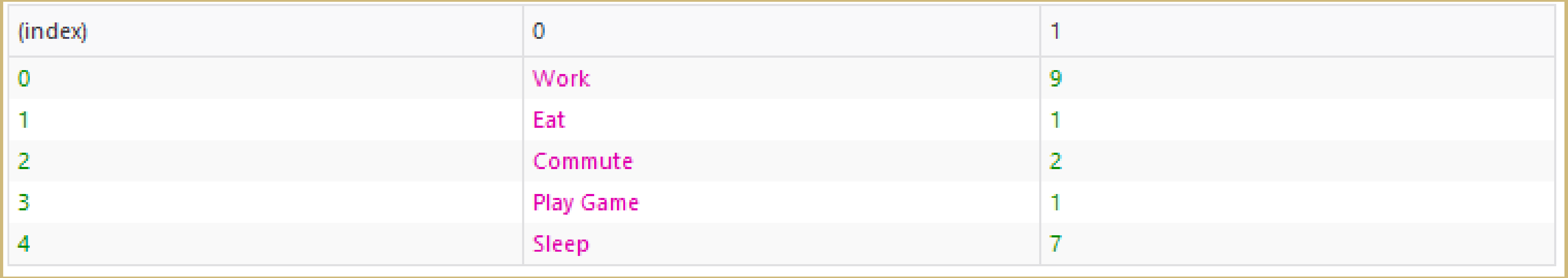
- Multi-Dimensional Arrays
  - JavaScript does not provide multidimensional array natively
  - Can create multidimensional array:
    - First, define "outer" array
    - Then nest in each element of outer array another array
  - Can say that JavaScript multidimensional array is array of arrays
  - Easiest way to define multidimensional array is to use array literal notation
  - Example:
    - Two-Dimensional Array
    - Each "row" is array of "column" values
    - Each element has two indices:
      - Row index (0 to n) (n rows)
      - Column index (0 to 1) (2 columns)

**Syntax:**

```
let var_array = [
                    [element00, element01],
                    [element10, element11],
                    .....
                    [elementn0, elementn1]
];
```

# 11.2 Arrays

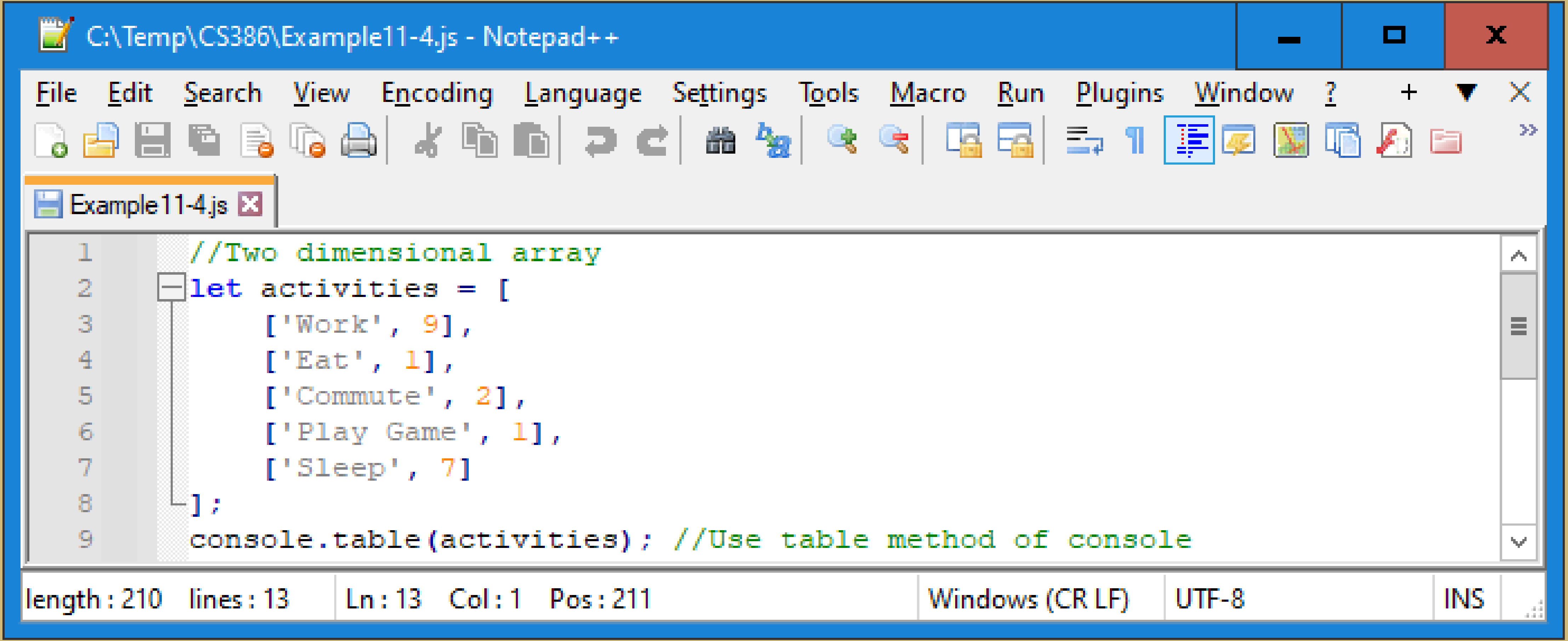
- **Example 11-4:**
  - Create two dimensional array of activities and daily hours spent on each activity
  - For example, one row would be ['Reading', 3]
  - Store array in variable arrActivities
  - Create 4 rows, any values
  - Display array in console using console.table(variable)



| (index) | 0 | 1 |
|---|---|---|
| 0 | Work | 9 |
| 1 | Eat | 1 |
| 2 | Commute | 2 |
| 3 | Play Game | 1 |
| 4 | Sleep | 7 |

## 11.2 Arrays

- **Example 11-4:**


C:\Temp\CS386\Example11-4.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-4.js

```
//Two dimensional array
let activities = [
    ['Work', 9],
    ['Eat', 1],
    ['Commute', 2],
    ['Play Game', 1],
    ['Sleep', 7]
];
console.table(activities); //Use table method of console
```

length : 210 lines : 13
Ln : 13 Col : 1 Pos : 211
Windows (CR LF)
UTF-8
INS

# 11.2 Arrays

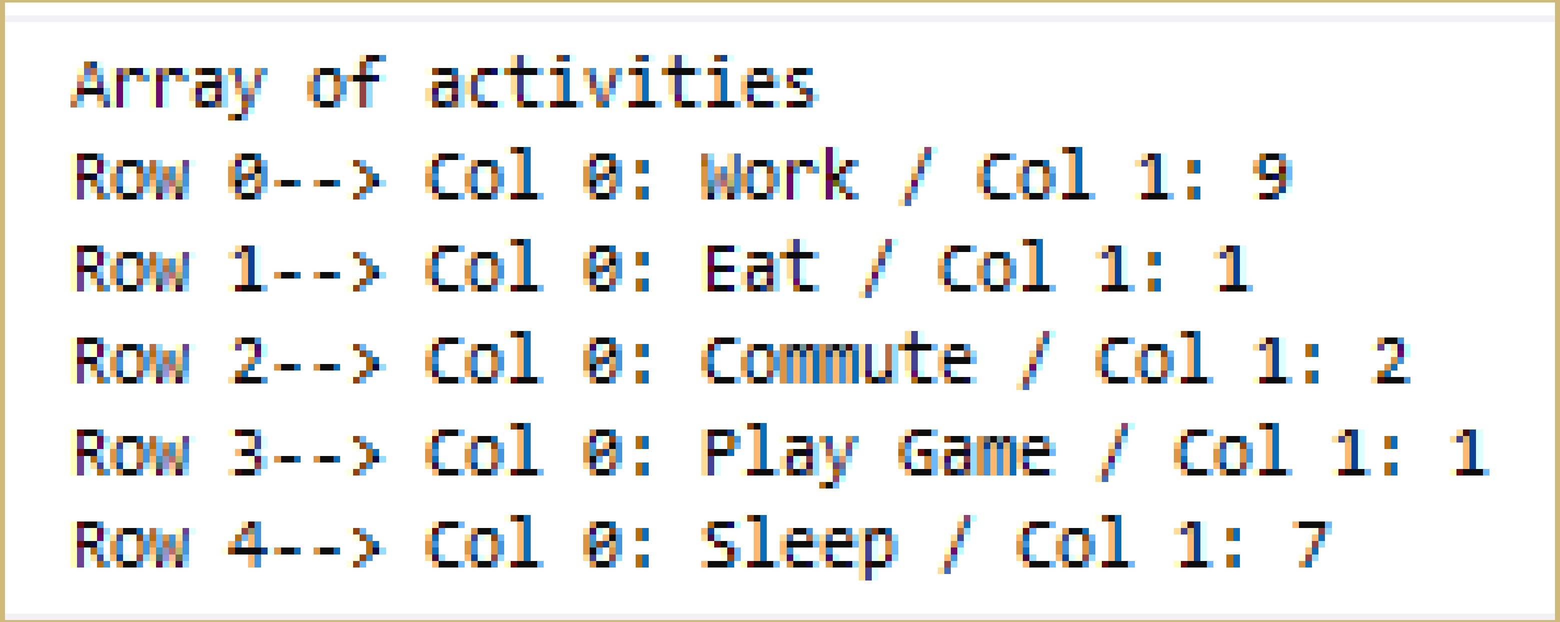
- Multi-Dimensional Arrays
  - To access elements of multidimensional array:
    - First use square brackets to access element of outer array → returns inner array
    - Then use another square bracket to access element of inner array
  - To loop over multi-dimensional arrays, use nested for loops:
    - Create outer loop variable → row index
    - Create inner loop variable → column index (using row index)
  - Within nested loop, access array elements by using outer and inner index

**Syntax:**
*var_array*[*i*][*j*]

**Syntax:**
```
for (let row=0; row < var_array.length; row++) {
    for (let col=0; col < var_array[row].length; col++) {
        var_array[row][col]
    }
}
```

## 11.2 Arrays

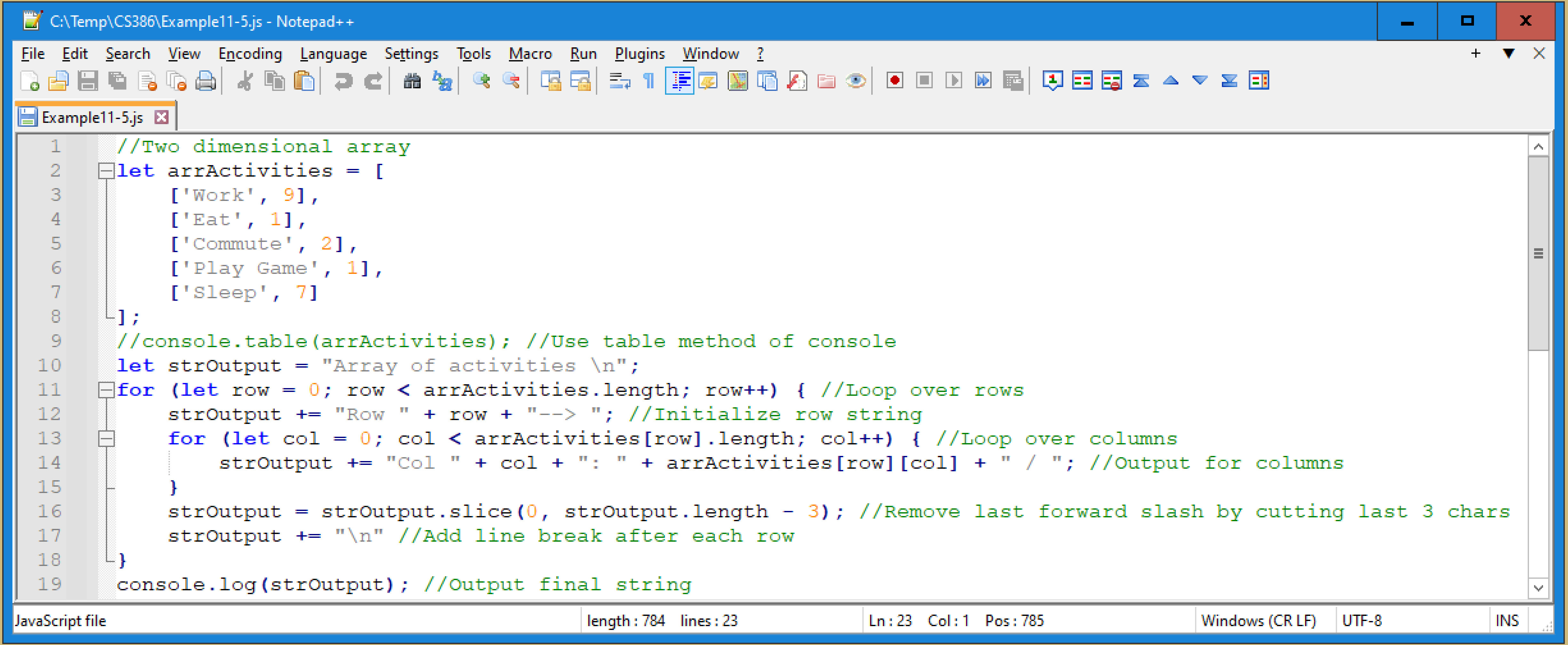
- **Example 11-5:**
  - Loop over array activities and output following string
  - Before loops, initialize strOutput with ‘Array of activities’ and line break
  - Within loops, accumulate strOutput to add the row and col values
  - After loops, display strOutput in console



```
Array of activities
Row 0--> Col 0: Work / Col 1: 9
Row 1--> Col 0: Eat / Col 1: 1
Row 2--> Col 0: Commute / Col 1: 2
Row 3--> Col 0: Play Game / Col 1: 1
Row 4--> Col 0: Sleep / Col 1: 7
```

# 11.2 Arrays

➢ **Example 11-5:**


C:\Temp\CS386\Example11-5.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-5.js

```
//Two dimensional array
let arrActivities = [
    ['Work', 9],
    ['Eat', 1],
    ['Commute', 2],
    ['Play Game', 1],
    ['Sleep', 7]
];
//console.table(arrActivities); //Use table method of console
let strOutput = "Array of activities \n";
for (let row = 0; row < arrActivities.length; row++) { //Loop over rows
    strOutput += "Row " + row + "--> "; //Initialize row string
    for (let col = 0; col < arrActivities[row].length; col++) { //Loop over columns
        strOutput += "Col " + col + ": " + arrActivities[row][col] + " / "; //Output for columns
    }
    strOutput = strOutput.slice(0, strOutput.length - 3); //Remove last forward slash by cutting last 3 chars
    strOutput += "\n" //Add line break after each row
}
console.log(strOutput); //Output final string
```

JavaScript file | length : 784 lines : 23 | Ln : 23 Col : 1 Pos : 785 | Windows (CR LF) | UTF-8 | INS

# 11.3 Functions

- Functions are one of fundamental building blocks in JavaScript
- Functions in JavaScript are similar to procedures:
  - Set of statements that performs tasks or calculates a value
  - But for procedure to qualify as function → Take some input and return some output
- To use functions, must define it somewhere in scope from which to call it
- Hoisting behavior:
  - Functions can be called before they are declared
  - Function declaration is hoisted to top of code

## 11.3 Functions

- Declaring Functions
  - Function declaration begins with keyword function followed by these components:
    - Identifier that names function:
    - Pair of parentheses around optional, comma-separated list of zero or more identifiers:
      - Identifiers are parameter names for function
      - Behave like local variables within body of function
    - Pair of curly braces with zero or more JavaScript statements inside:
      - Statements are body of function, they are executed whenever function is invoked/called
      - Optional return statement returns primitive/object value

**Syntax:**

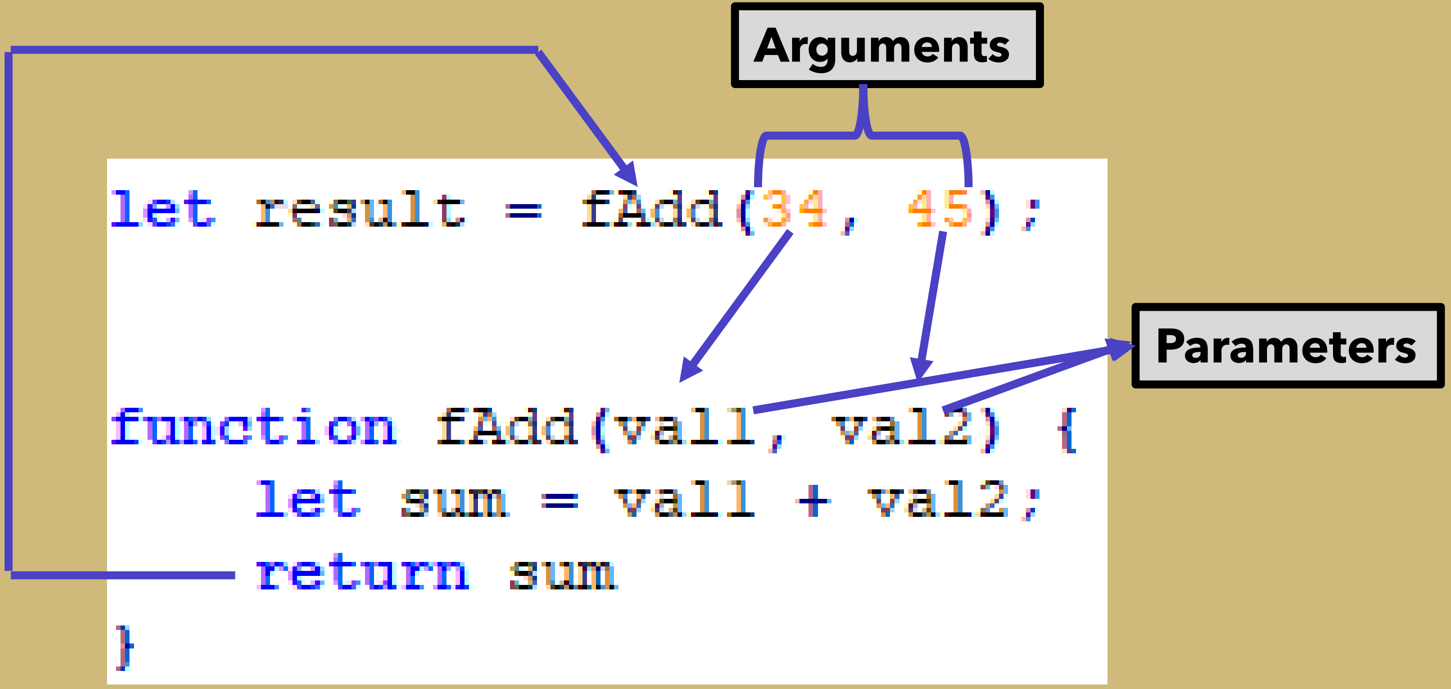
```
function func_name(parameters) {
    statements
    return value/object
}
```

## 11.3 Functions

- Declaring Functions
  - Function name is identifier, follows same naming rules as for variables
  - Function name should tell what function does → choosing good function names is important
  - Parameters are optional, when defined need to pass arguments when function is called
  - Return statement is optional:
    - If included, return either primitive or object value
    - Use one return statement at end of function (recommended)
    - If omitted, function returns undefined

## 11.3 Functions

- Executing Functions
  - To execute functions, use function name with parentheses and arguments (if required)
  - Arguments can be literal values, variables, or another function call (that returns values)
  - Consume returning value in expression or assignment



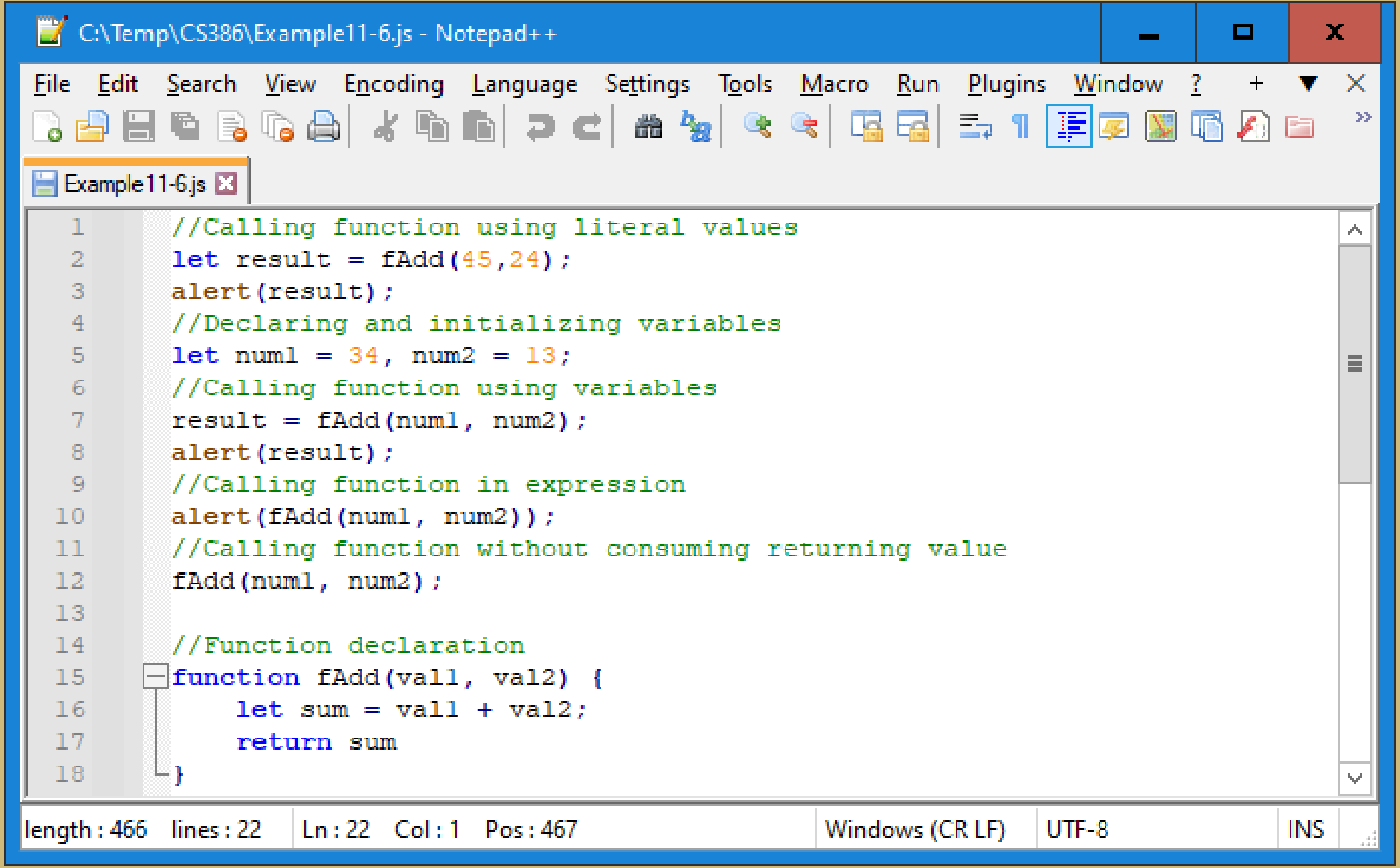
```
let result = fAdd(34, 45);


function fAdd(val1, val2) {
    let sum = val1 + val2;
    return sum
}
```

## 11.3 Functions

- **Example 11-6:**
  - ❑ Create function fAdd having two parameters val1 and val2:
    - Declare variable sum and assign sum of parameters val1 and val2
    - Return variable sum
  - ❑ Place function at end of code (to demonstrate hoisting behavior)
  - ❑ Declare variable result and assign fAdd return value using literal values
  - ❑ Display result in alert/console
  - ❑ Declare and initialize two variables num1 and num2 with some numbers
  - ❑ Assign fAdd returning value using num1 and num2 as arguments into variable result
  - ❑ Display result in alert/console
  - ❑ Place same function call as above into alert/console (nested functions)

# 11.3 Functions

- **Example 11-6:**


C:\Temp\CS386\Example11-6.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-6.js

```
//Calling function using literal values
let result = fAdd(45,24);
alert(result);
//Declaring and initializing variables
let num1 = 34, num2 = 13;
//Calling function using variables
result = fAdd(num1, num2);
alert(result);
//Calling function in expression
alert(fAdd(num1, num2));
//Calling function without consuming returning value
fAdd(num1, num2);

//Function declaration
function fAdd(val1, val2) {
    let sum = val1 + val2;
    return sum
}
```

length : 466 lines : 22 Ln : 22 Col : 1 Pos : 467 Windows (CR LF) UTF-8 INS

# 11.4 CRUD Operations

- CRUD Operations:
  - **C**reate data
  - **R**ead data
  - **U**pdate data
  - **D**elete data
- Store table like data (rows and columns)
- Use array containing element objects to store table data:

**Syntax:**
```
let var = [
   {col1: val1, col2: val2, ...coln: valn},
   {col1: val1, col2: val2, ...coln: valn},
    .....
]
```

# 11.4 CRUD Operations

- First column should be id column, unique identifier, primary key
- Use array find or findIndex method to query data in array

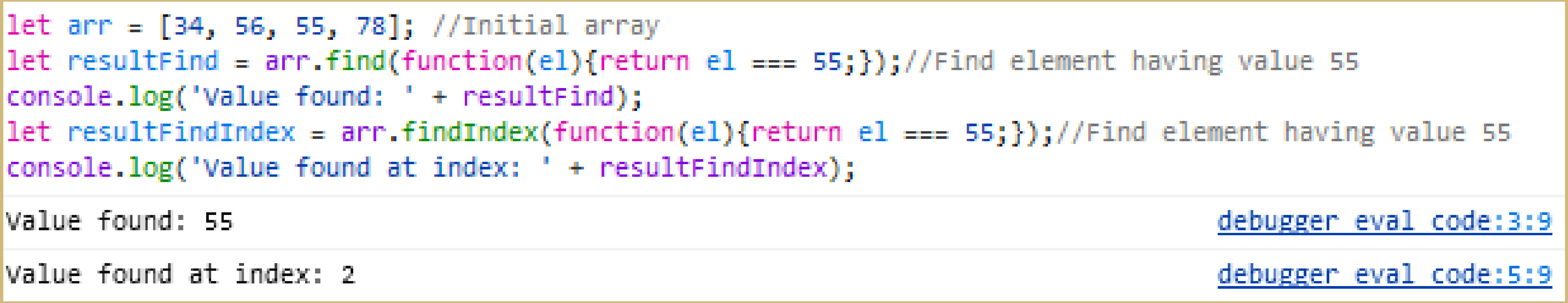
**Syntax:**
*var_arr*.**find(function(***currentValue*[**,** *index* [**,** arr]]),thisValue)
*var_arr*.**findIndex(function(***currentValue*[**,** *index* [**,** arr]]),thisValue)

| Parameter | Description |
|---|---|
| **function()** | Required. A function to run for each array element. |
| **currentValue** | Required. The value of the current element. |
| **index** | Optional. The index of the current element. |
| **arr** | Optional. The array of the current element. |
| **thisValue** | Optional. Default undefined. A value passed to the function as its this value. |

- Return value:
  - Find(): Value of first element if found
  - findIndex(): Index of first element if found
  - Otherwise returns undefined

# 11.4 CRUD Operations

- <u>Examples:</u>
- Initial array:
  - let arr = [34, 56, 55, 78]
- Find element having value of 55, also find index

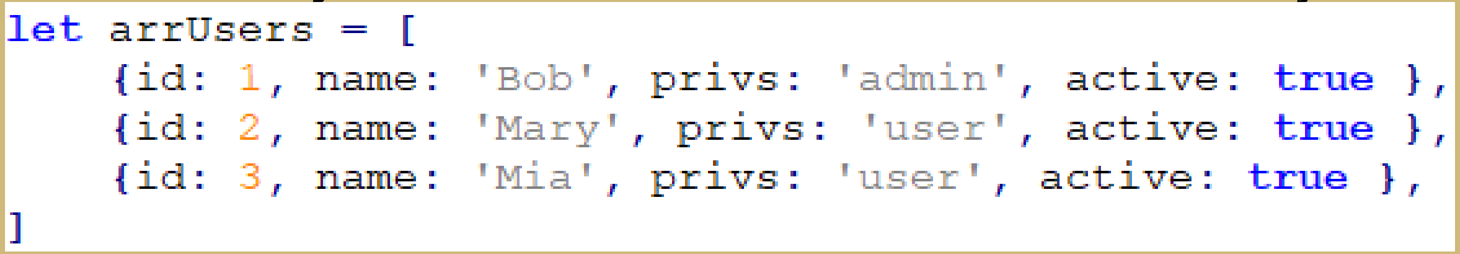


```
let arr = [34, 56, 55, 78]; //Initial array
let resultFind = arr.find(function(el){return el === 55;});//Find element having value 55
console.log('Value found: ' + resultFind);
let resultFindIndex = arr.findIndex(function(el){return el === 55;});//Find element having value 55
console.log('Value found at index: ' + resultFindIndex);
```

```
Value found: 55                                              debugger eval code:3:9
Value found at index: 2                                      debugger eval code:5:9
```

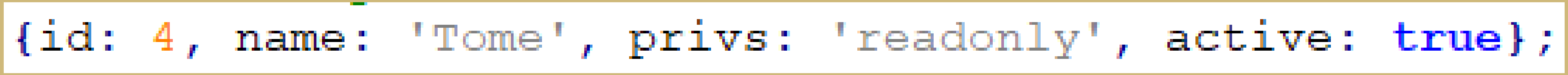
## 11.4 CRUD Operations

- **Example 11-7:**
- Create array arrUsers with three records as objects:



```
let arrUsers = [
    {id: 1, name: 'Bob', privs: 'admin', active: true },
    {id: 2, name: 'Mary', privs: 'user', active: true },
    {id: 3, name: 'Mia', privs: 'user', active: true },
]
```

- Add new record at the end of array:

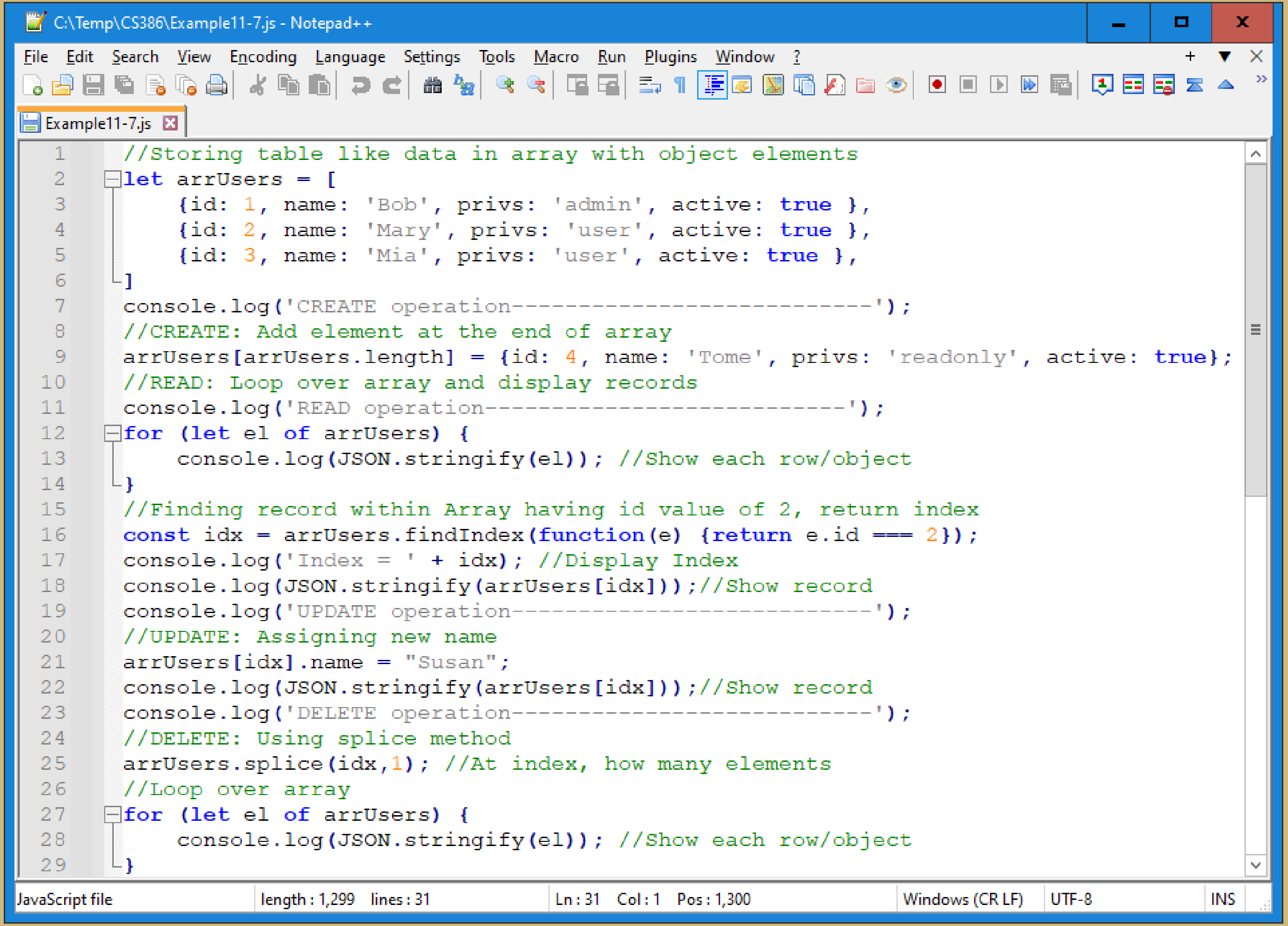


```
{id: 4, name: 'Tome', privs: 'readonly', active: true};
```

- READ: Loop over array, display each object using JSON.stringify
- For UPDATE and DELETE, find record having id value of 2 and get index
- UPDATE: Assign new name for record of interest using found index
- DELETE: Use splice() method to remove record of interest using found index

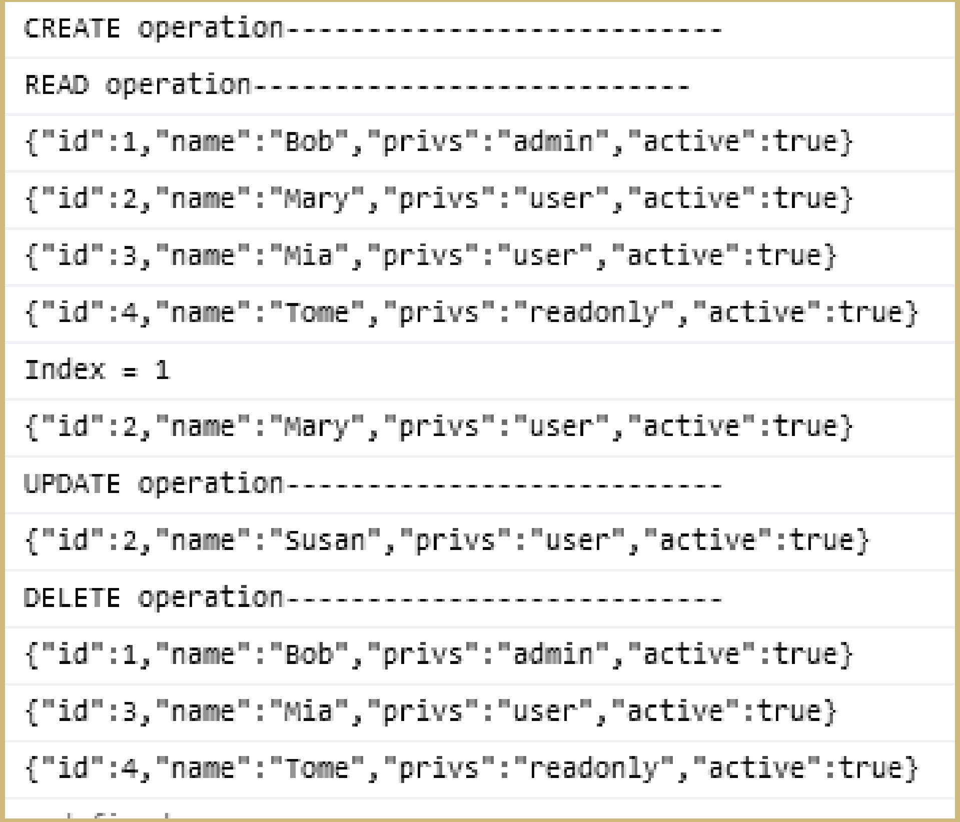
# 11.4 CRUD Operations

➢ **Example 11-7:**


C:\Temp\CS386\Example11-7.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example11-7.js

```javascript
//Storing table like data in array with object elements
let arrUsers = [
    {id: 1, name: 'Bob', privs: 'admin', active: true },
    {id: 2, name: 'Mary', privs: 'user', active: true },
    {id: 3, name: 'Mia', privs: 'user', active: true },
]
console.log('CREATE operation---------------------------');
//CREATE: Add element at the end of array
arrUsers[arrUsers.length] = {id: 4, name: 'Tome', privs: 'readonly', active: true};
//READ: Loop over array and display records
console.log('READ operation---------------------------');
for (let el of arrUsers) {
    console.log(JSON.stringify(el)); //Show each row/object
}
//Finding record within Array having id value of 2, return index
const idx = arrUsers.findIndex(function(e) {return e.id === 2});
console.log('Index = ' + idx); //Display Index
console.log(JSON.stringify(arrUsers[idx]));//Show record
console.log('UPDATE operation---------------------------');
//UPDATE: Assigning new name
arrUsers[idx].name = "Susan";
console.log(JSON.stringify(arrUsers[idx]));//Show record
console.log('DELETE operation---------------------------');
//DELETE: Using splice method
arrUsers.splice(idx,1); //At index, how many elements
//Loop over array
for (let el of arrUsers) {
    console.log(JSON.stringify(el)); //Show each row/object
}
```

JavaScript file | length : 1,299 lines : 31 | Ln : 31 Col : 1 Pos : 1,300 | Windows (CR LF) | UTF-8 | INS



```
CREATE operation---------------------------
READ operation---------------------------
{"id":1,"name":"Bob","privs":"admin","active":true}
{"id":2,"name":"Mary","privs":"user","active":true}
{"id":3,"name":"Mia","privs":"user","active":true}
{"id":4,"name":"Tome","privs":"readonly","active":true}
Index = 1
{"id":2,"name":"Mary","privs":"user","active":true}
UPDATE operation---------------------------
{"id":2,"name":"Susan","privs":"user","active":true}
DELETE operation---------------------------
{"id":1,"name":"Bob","privs":"admin","active":true}
{"id":3,"name":"Mia","privs":"user","active":true}
{"id":4,"name":"Tome","privs":"readonly","active":true}
```