ICT10013 Programming Concepts

Week 07

Manipulating Arrays.

Parallel Arrays.

Adding values to an array. push() method

Last week we discussed Arrays.

There are a number of ways to add data to an array

- **Creating and populating an array in a single statement**

```
var arrNames = ["Fred", "Sue", "Emma", "Dave"];
```

arrNames

0	Fred
1	Sue
2	Emma
3	Dave

The **.push() method** of an array **adds** a value to the **end** of the array.

```
var arrNames = ["Fred", "Sue", "Emma", "Dave"];
arrNames.push("Taylor");
arrNames.push("Liam");
arrNames.push("Ed","Clare");
```

Note: Round brackets are used, because it's a call to a function

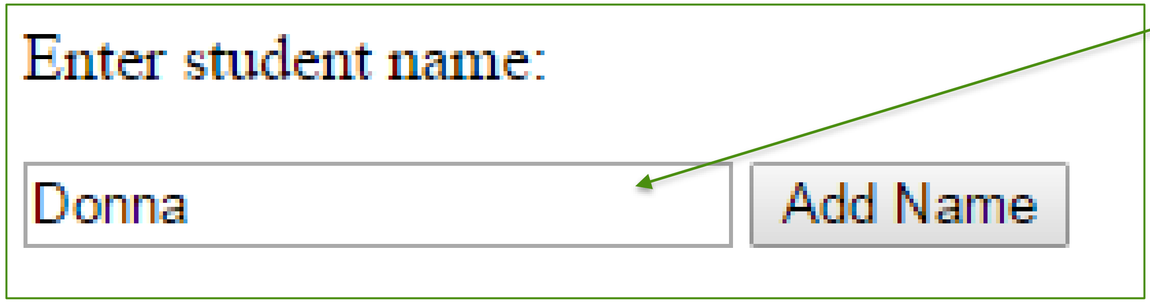
arrNames

0	Fred
1	Sue
2	Emma
3	Dave
4	Taylor
5	Liam
6	Ed
7	Clare

Adding values to an array via input box

- **Add an element(s) to an existing array using .push method & input box**

```
 var arrNames = [];   //creates an empty array
var arrNewName = document.getElementById("stuname").value;
arrNames.push(vNewName);
```



id of input box is stuname

Array contents: Donna (at position 0)   Length:1

Clearing an array

The most simple approach to clearing an array is:

- **Replace an array with a new empty array**

```
var arrValues = [10,20,30,40,50];
var arrValues = [];
```

`var arrValues = [10,20,30,40,50];`	Creates and populates the array
`var arrValues = [];`	Assigns an empty array to the arrValues variable

***Ignore** JavaScript forum discussions about, performance, garbage collections, JavaScript optimizations, memory allocation, etc.*

Note: To understand discussions about garbage collections and memory allocation you generally need to know about a few more advanced topics. We are learning basic programming concepts; at this stage you can ignore discussion about performance.

pop() method

The pop() method **removes** the **final** element from an array.

```
var arrNames = ["Ted", "Sue", "Dave", "Emma"];   Array contents: Ted, Sue, Dave, Emma   Length:4
arrNames.pop();                                  Array contents: Ted, Sue, Dave         Length:3
arrNames.pop();                                  Array contents: Ted, Sue               Length:2
```

shift() method

The shift() method **removes** the **first** element from an array.

```
var arrNames = ["Ted", "Sue", "Dave", "Emma"];   Array contents: Ted, Sue, Dave, Emma   Length:4
arrNames.shift();                                Array contents: Sue, Dave, Emma        Length:3
arrNames.shift();                                Array contents: Dave, Emma             Length:2
```

delete() method

The delete() method **replaces** an element value with an **undefined value** in the array.

It creates 'holes' in the array

Creates confusion! Do NOT use.

var arrNames = ["Ted", "Sue", "Dave", "Emma"];	Array contents: Ted, Sue, Dave, Emma	Length:4
arrNames.delete(0);	Array contents: , Sue, Dave, Emma	Length:4
arrNames.delete(2);	Array contents: , Sue, , Emma	Length:4

Finding a maximum value in an array

Suppose we want to find the person with the longest name

var arrNames = ["Fred","Sue","Emma“,...];

Or we want to **find** the **maximum** value in an array, e.g. var arrVals = [4,8,5];

You may be able to use a function such as Math.max for numbers

e.g. alert("The largest value is " + Math.max(. . . arrVals));

The … (3 dots) is a JavaScript operator called a spread operator.
See https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Spread_operator

However, this doesn't teach you how to do it using your **programming skills.**

Also finding there are **no inbuilt** functions to find the **longest name,** etc.

Finding a maximum value in an array

var arrVals = [4,8,5];

How would you find the maximum value in this array **programmatically**?

You could use a **loop** to work through an array.

```
    Begin by storing a value from position 0 in a variable called vMaxVal
    Now loop through all elements one by one starting from position 1
        Does the current element have a larger value than vMaxVal ?
        If Yes, then remember it (i.e. replace the value of vMaxVal)
        If No, then ignore it
    Repeat
```

After processing the entire array, vMaxVal will contain the maximum value in the array

Step by Step

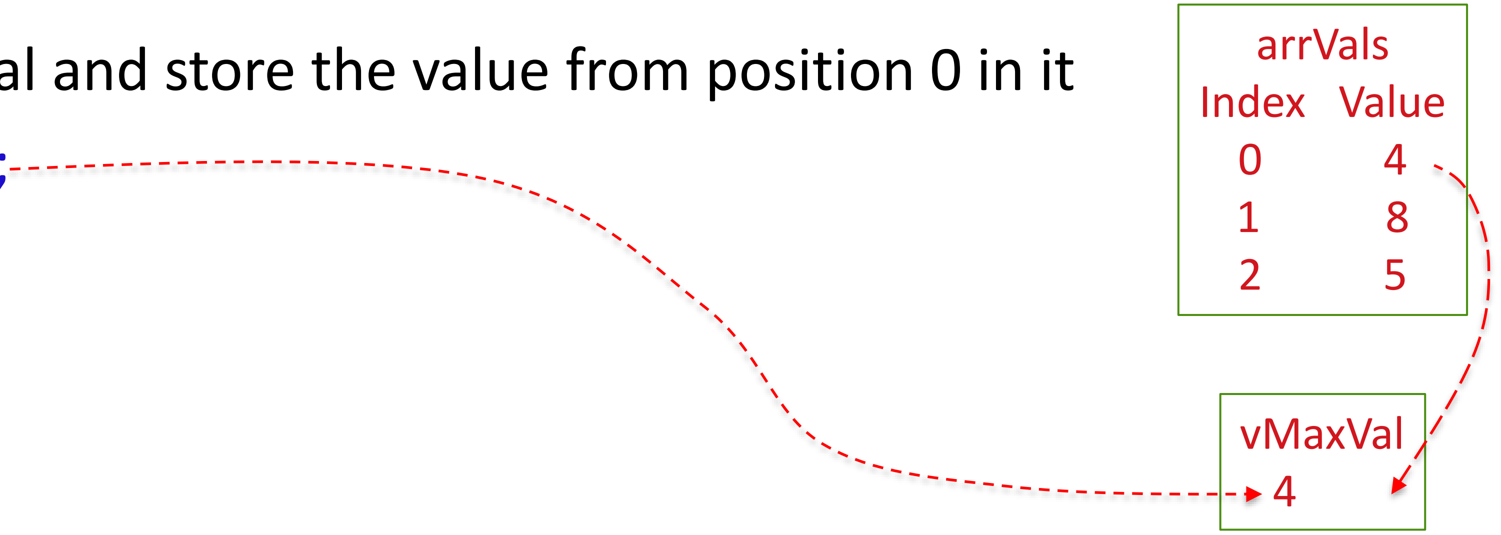
Create the array of values;

```
var arrVals = [4,8,5];
```

Create a variable vMaxVal and store the value from position 0 in it

```
var vMaxVal = arrVals[0];
```

Pictorial view of variables

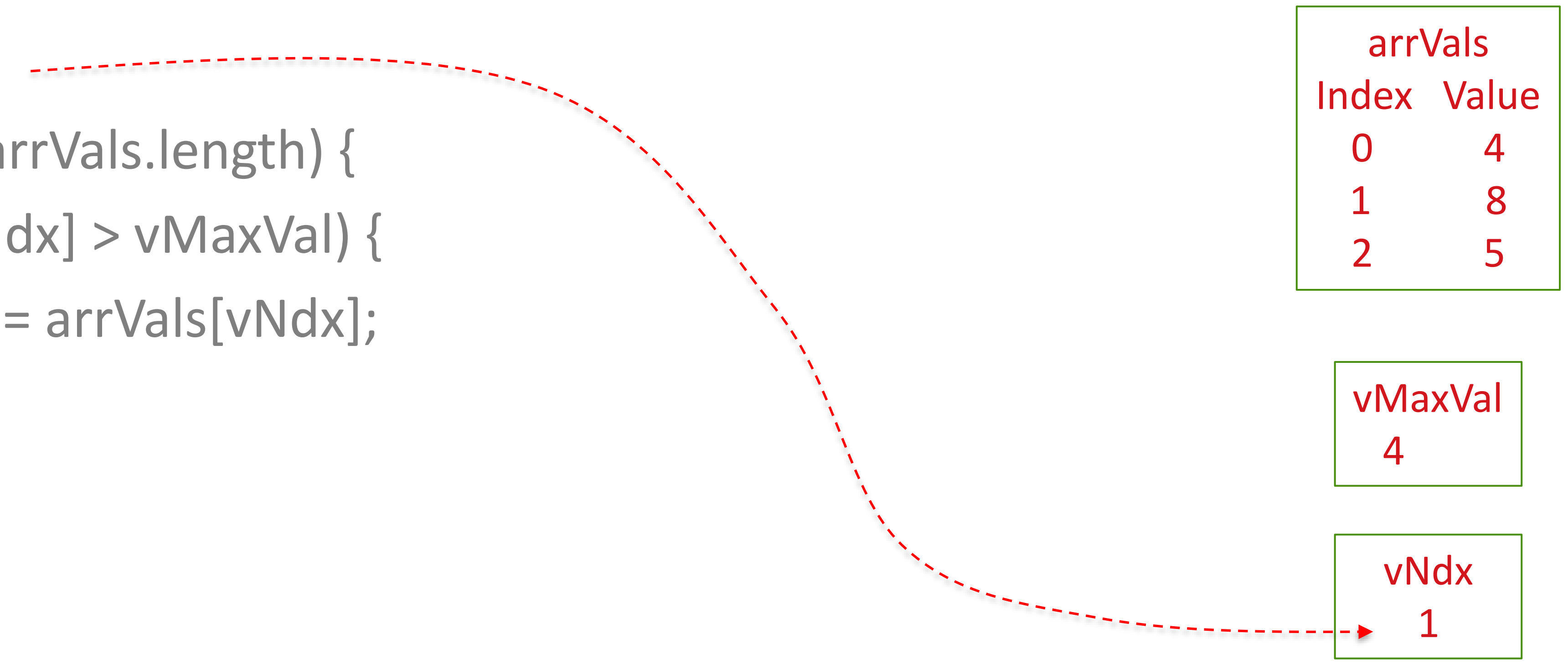


Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
   if (arrVals[vNdx] > vMaxVal) {
         vMaxVal = arrVals[vNdx];
   }
   vNdx++;
}
```

Pictorial view of variables

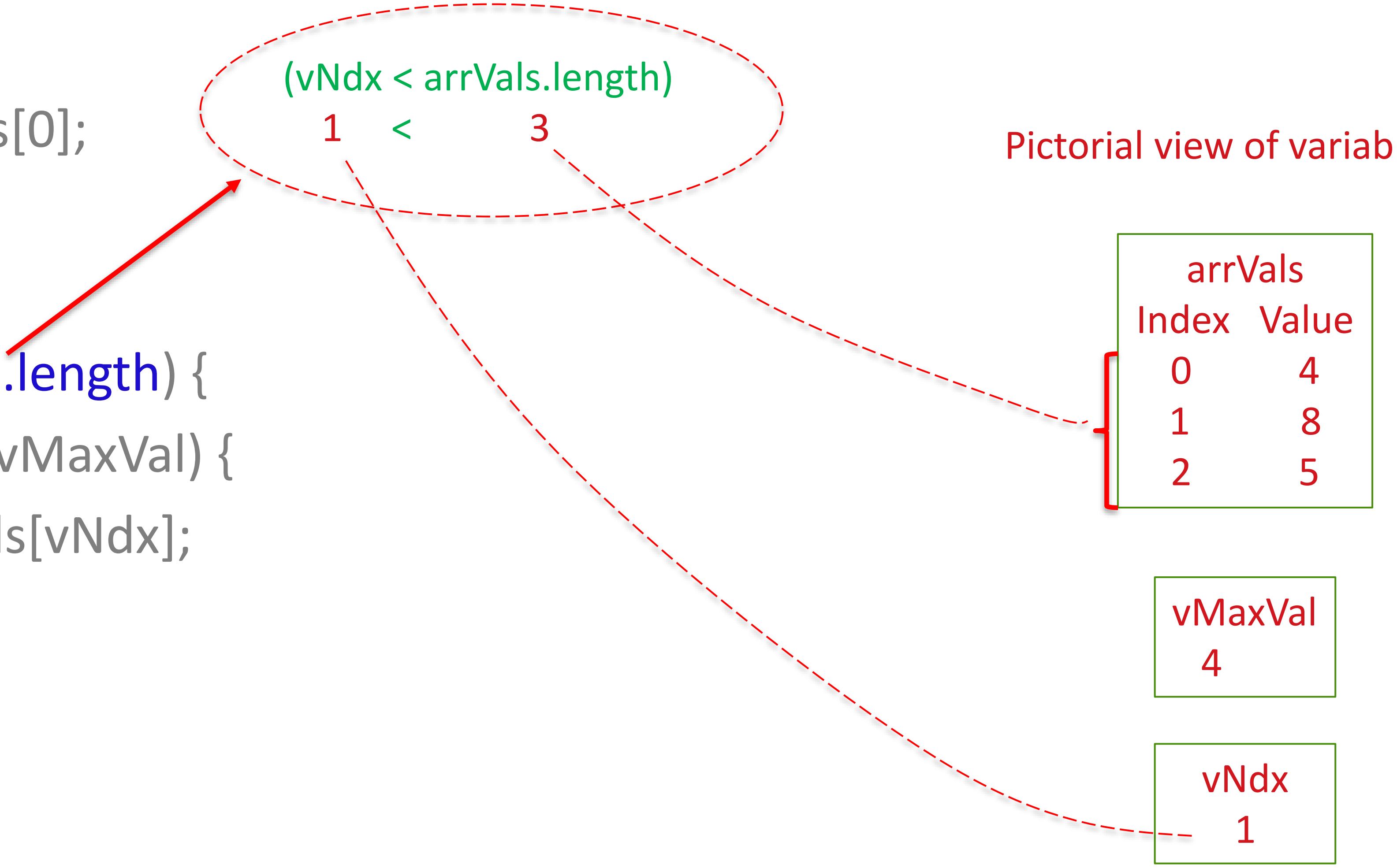


Comments: Set vNDX to 1

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```

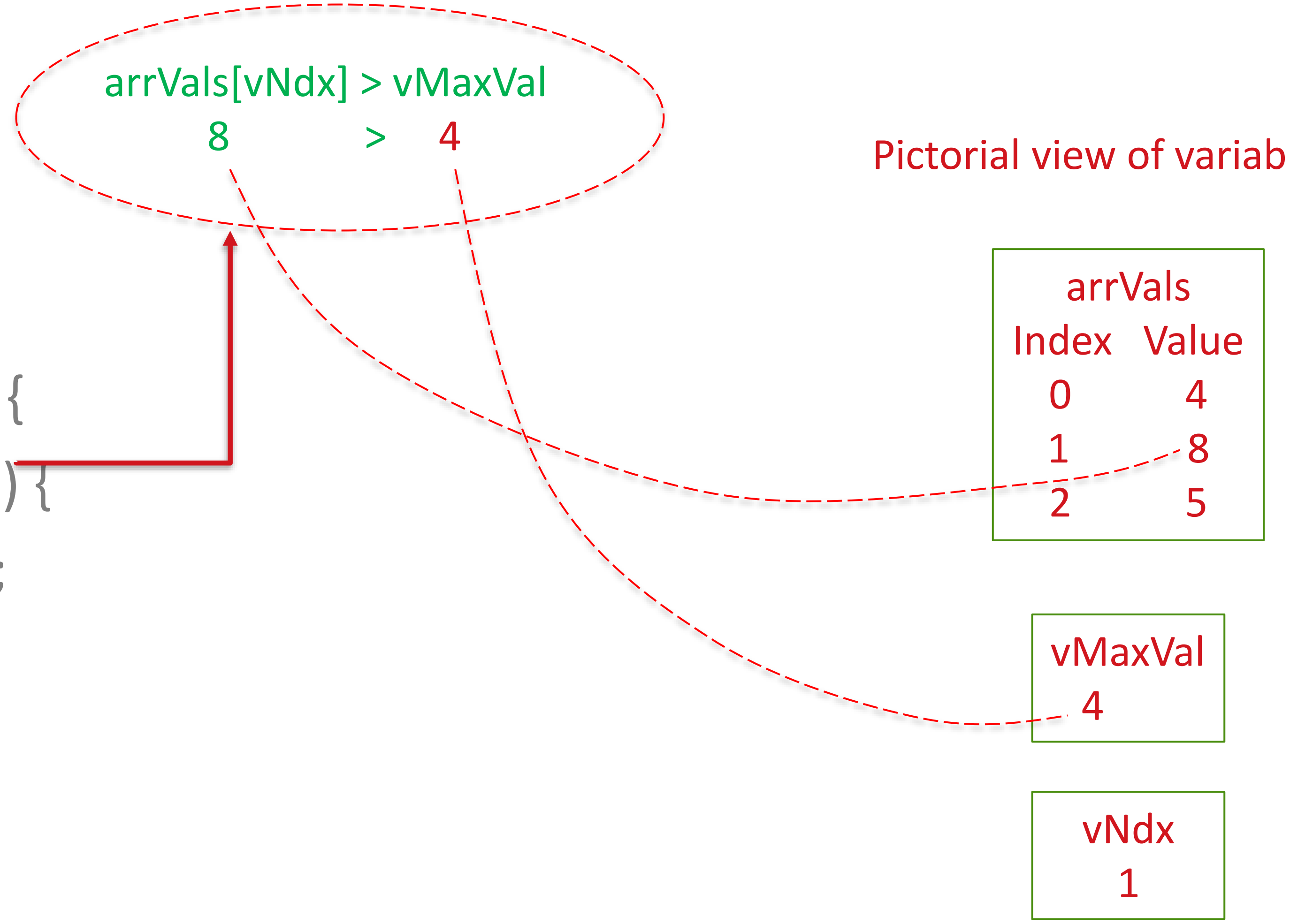


Comments: Compare vNDX and length of the array

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```



Comments: Compare value of element 1 with vMaxVal

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
   if (arrVals[vNdx] > vMaxVal) {
      vMaxVal = arrVals[vNdx];
   }
   vNdx++;
}
```

Pictorial view of variables

arrVals

Index	Value
0	4
1	8
2	5

vMaxVal
8

vNdx
1

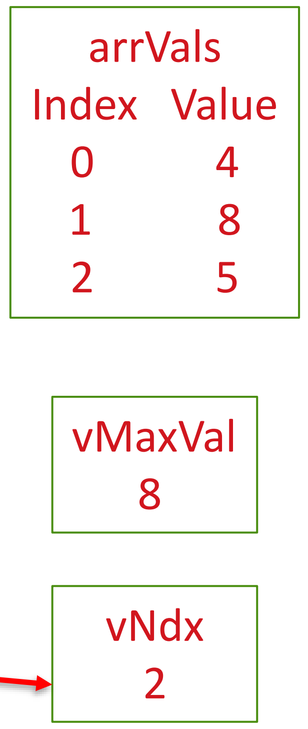
Comments: Set vMaxVal to value of element 1 of the array

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```

Pictorial view of variables

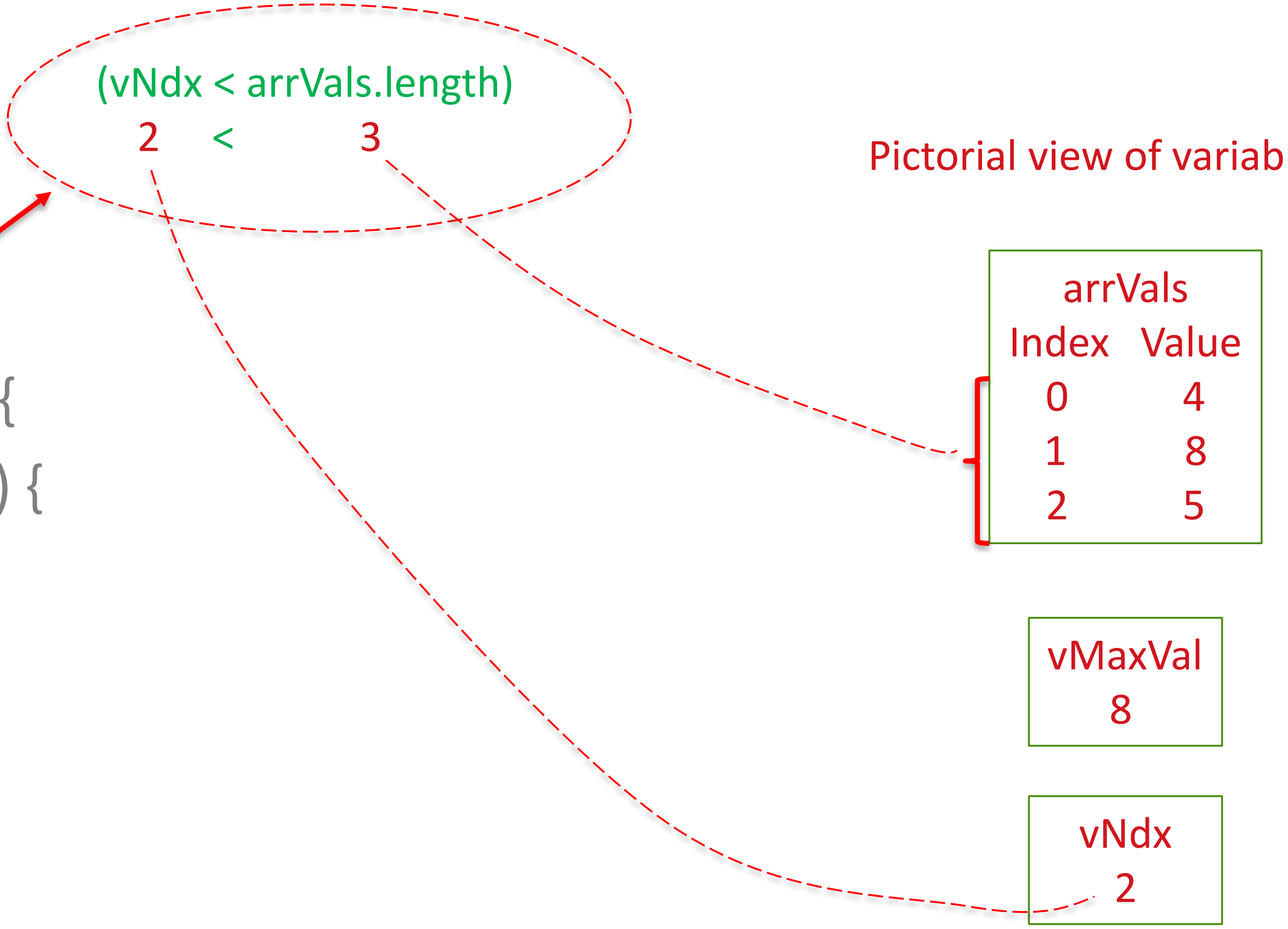


Comments: Increment value of vNdx
And jump back to the while condition…

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```

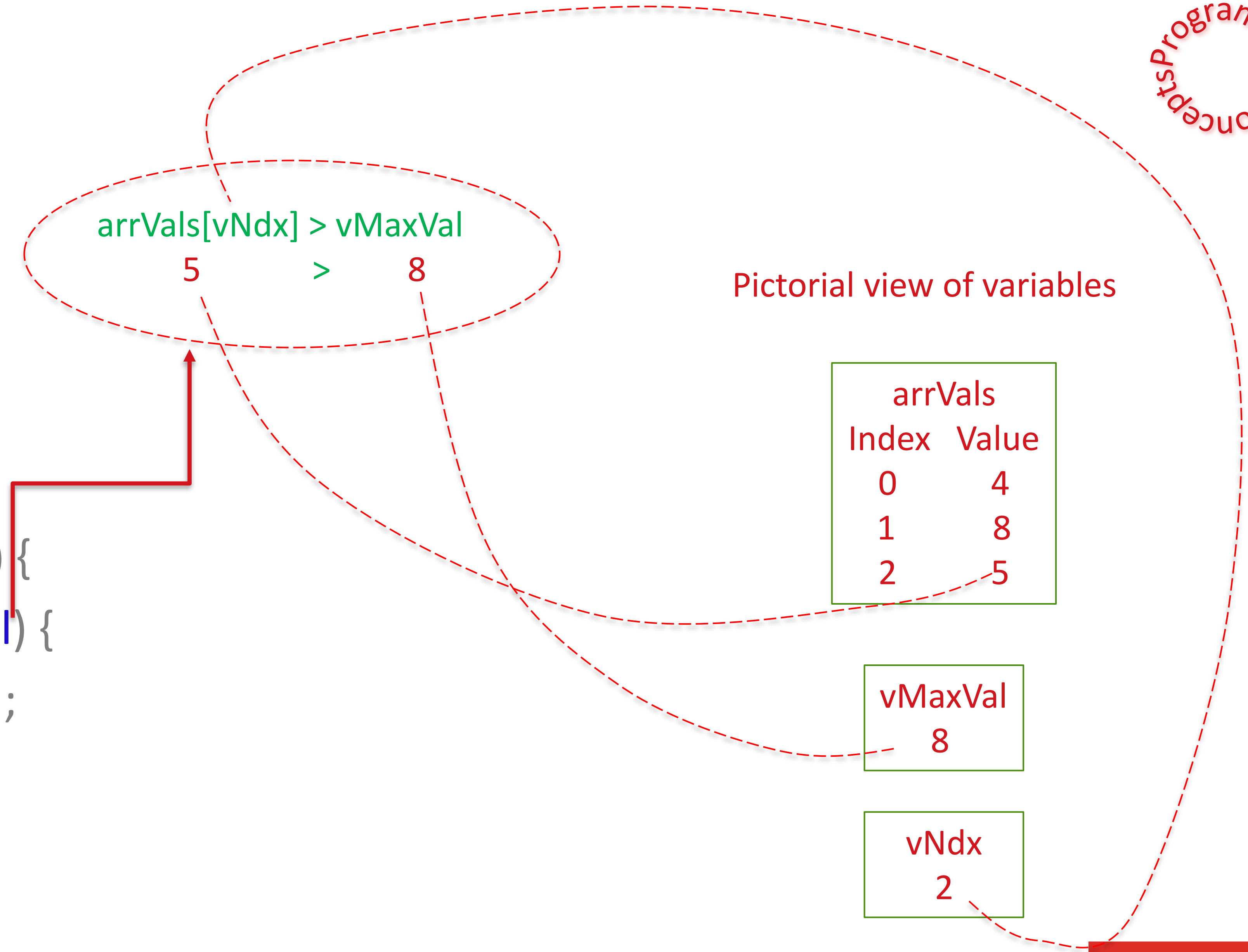


Comments: Compare vNDX and length of the array

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];


var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```



arrVals	
Index	Value
0	4
1	8
2	5

vMaxVal
8

vNdx
2

Comments: Compare value of element 2 with vMaxVal

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
   if (arrVals[vNdx] > vMaxVal) {
      vMaxVal = arrVals[vNdx];
   }
   vNdx++;
}
```

Pictorial view of variables

arrVals

Index	Value
0	4
1	8
2	5

vMaxVal
8

vNdx
3

Comments: Element at position 2 is less than the value of vMaxVal
So no need to update vMaxVal
Simply increment vNdx

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
```

Pictorial view of variables

arrVals

Index	Value
0	4
1	8
2	5

vMaxVal
8

vNdx
3

Comments: Back to the condition of the loop.
3<3 is false => the loop stops

Step by Step

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

var vNdx = 1;
while (vNdx < arrVals.length) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    }
    vNdx++;
}
alert("Maximum value found: " + vMaxVal);
```

Comments:

The process continues until the value of vNdx is equal the length of the array.

At that point the value of vMaxVal will be 8

Pictorial view of variables

arrVals

Index	Value
0	4
1	8
2	5

vMaxVal
8

vNdx
3

Finding max value using **for** loop

```
var arrVals = [4,8,5];
var vMaxVal = arrVals[0];

for (var vNdx = 1; vNdx < arrVals.length; vNdx++) {
    if (arrVals[vNdx] > vMaxVal) {
        vMaxVal = arrVals[vNdx];
    } //if
 } //for
alert("Maximum value found: " + vMaxVal);
```

Parallel Arrays

- Arrays where related values are stored in the same position are referred to as parallel arrays.
- Example:

```
var arrStudentIDs = ["1012233", "1012323", "1100123", "1122345"];
var arrSubjectMarks = [97, 85, 94, 57];
```

- arrStudentIDs[0] corresponds to arrSubjectMarks[0]
- arrStudentIDs[1] corresponds to arrSubjectMarks[1]

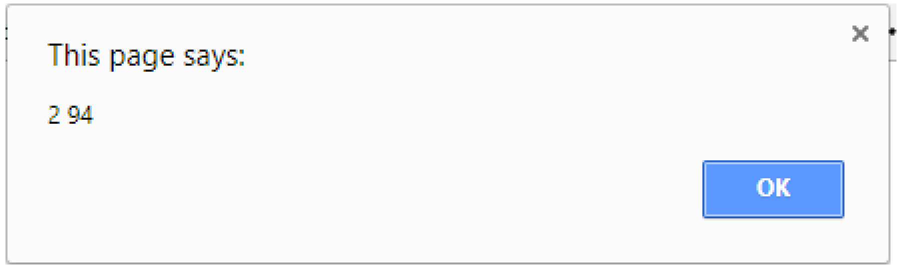
Finding data using parallel arrays

- We need to find student's subject mark knowing student ID, e.g. 1100123.
- Steps:
  1. Check arrStudentIDs array: at which position it contains this ID
     - The search function takes 2 parameters: array to search and search value
     - The search function returns the index (position) after the search is complete. If the value is found, index holds its position, otherwise it holds the length of the array
  2. Check if index equals the length of the array. If yes, the ID was not found. If not, arrSubjectMarks array contains the corresponding mark at the same position.

Finding data using parallel arrays (JavaScript)

```
// find item in the array
function search(array, searchItem) {
  var index = 0;
  var found = false;
  while (index<array.length && !found) {
   if (array[index]==searchItem)
        found = true;
   else
        index++; // if item found, index will NOT increment
  } // while
  return index; // contains the value of the found item or array.length
}

function init() {
  var arrStudentIDs = ["1012233", "1012323", "1100123", "1122345"];
  var arrSubjectMarks = [97, 85, 94, 57];
  var pos = search(arrStudentIDs, "1100123");
  if (pos==arrStudentIDs.length)
   alert("not found");
  else
   alert(pos + " " + arrSubjectMarks[pos]);
}
```


This page says:

2 94

Writing small chunks of code

Novice programmers often type too many lines of code at once.

When they test their code, they don't know which piece of code is causing problems.

Solution: Start small. Start simple.

```
function start() {
   alert ("function start is running");
   var vResult = func1(25);
   alert("The value of result is:" + vResult);
   alert ("function start is finished");
}
```

```
function func1(pNum) {
   alert("func1 is running");
   alert("The value of pNum is: " + pNum);
   alert("func1 has finished");
   return 100;
}
```

When this code executes, you simply want to check that each function executes successfully.
Perhaps func1 is supposed to perform a **complex** calculation.
Don't code the calculation yet. Simply test that the function can successfully return a value.
Until this code works, don't bother adding additional lines of code.

Writing small chunks of code

Add comments // or /* and */ to stop blocks of code from executing.

```
function start() {
    alert ("function start is running");
    //var vResult1 = func1(25);
    //alert("The value of result1 is:" + vResult1);
    var vResult2 = func2(25);
    alert("The value of result2 is:" + vResult2);
    alert ("function start is finished");
}
```

```
function func1(pNum) {
    alert("func1 is running");
    alert("The value of pNum is: " + pNum);
    alert("func1 has finished");
    return 100;
}

function func2(pNum) {
    alert("func2 is running");
    alert("The value of pNum is: " + pNum);
    alert("func2 has finished");
    return 500;
}
```

The function funct1() is **not** executed when the code runs.
You can now concentrate on getting func2() to work.