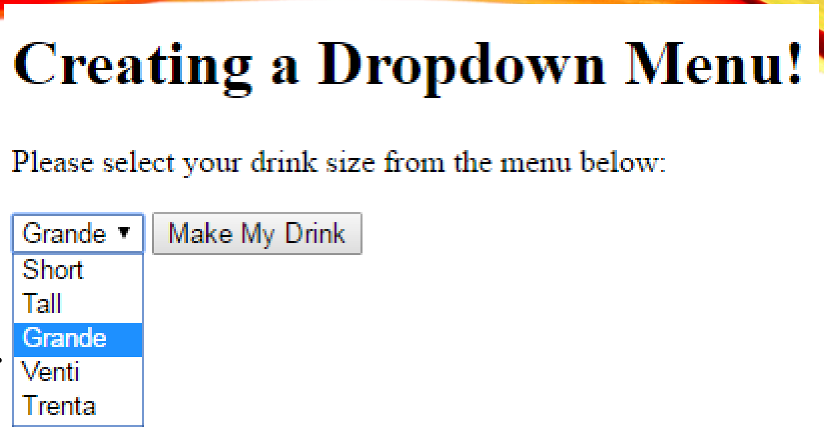
ARRAYS!

(and drop down boxes!)

ANOTHER HTML THING

- <select>
- Creates a dropdown box on the webpage
  - Allows the user to select from several choices
- The advantage of using select is that we are guaranteed that the input will be one of our choices (no invalid user input)
- To specify options in a dropdown box we use the <option> tag inside the <select> tag



Creating a Dropdown Menu!

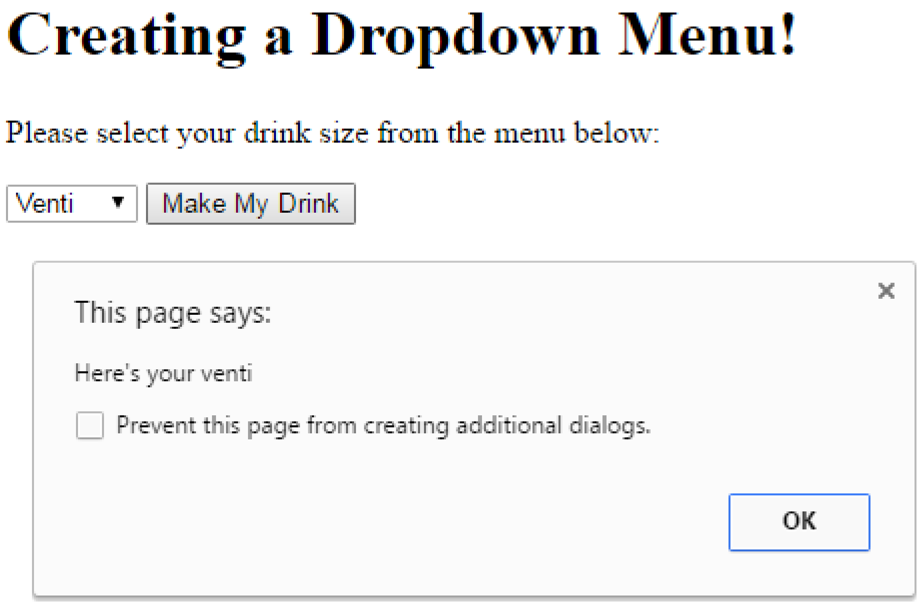
Please select your drink size from the menu below:

```
<h1>Creating a Dropdown Menu!</h
<p>Please select your drink size
</p>
<select id=“drinkSize”>
        <option value="short">Short</option>
        <option value="tall">Tall</option>
        <option value="grande">Grande</option>
        <option value="venti">Venti</option>
        <option value="trenta">Trenta</option>
</select>
<button type="button" onclick="makeDrink()">Make My
Drink</button>
```

- We must set the value attribute of the option, so we can access the value contained in it using JavaScript

<SELECT>

- How do you think we access the currently selected optior in the JavaScript?


Creating a Dropdown Menu!

Please select your drink size from the menu below:

Venti ▾ Make My Drink

This page says:

Here's your venti

☐ Prevent this page from creating additional dialogs.

OK

```
function makeDrink(){
  let size= document.getElementById("drinkSize");
  alert("Here's your " + size.value);
}
```

REMEMBER

- Values
  - Have type
  - Represent a concrete number, string, or boolean
  - Ex. 1, “cow”, true
- Variables
  - Hold values in memory
  - Can change their values
  - Are used to carry information through our programs

ARRAYS MAKE LIFE EASIER

- Have you ever thought “Man, I have so much information to store, I don’t want to have to create a million variables to store it all!”
- If you have thought this, you’re going to love arrays
- Arrays are an ordered set of elements
- Arrays are used to store and number a list of things
  - We start numbering these items at 0
  - This number is called the index

CREATING ARRAYS

- Concept:
- We use a variable to refer to an array
- Each item in the array is a value and has an index number
- We can then refer to an item in the array using its array name and index number

METHOD 1 TO CREATE ARRAYS

Creates an empty array

- Using [ ]

```
let myArray = [];
```

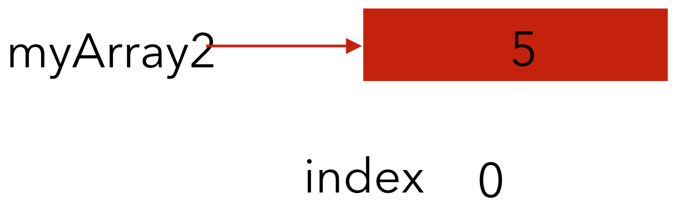
$$\text{myArray} \longrightarrow [\ ]$$

METHOD 1 TO CREATE ARRAYS

- Using [ ]

```
let myArray = [];
let myArray2 = [5];
```

Creates an array of size 1
That contains the number 5

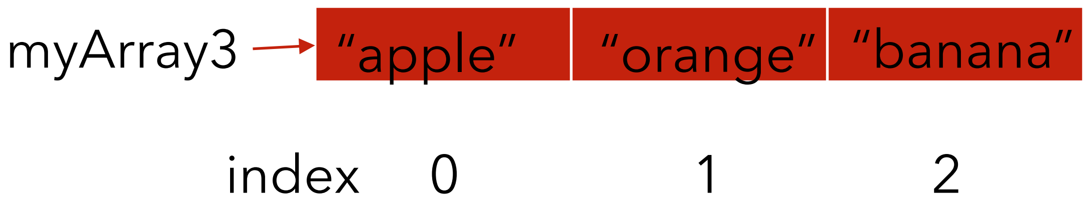


METHOD 1 TO CREATE ARRAYS

- Using [ ]

```
let myArray = [];
let myArray2 = [5];
let myArray3 = [“apple”, “orange”, “banana”];
```

Creates an array of size 3, containing 3 strings

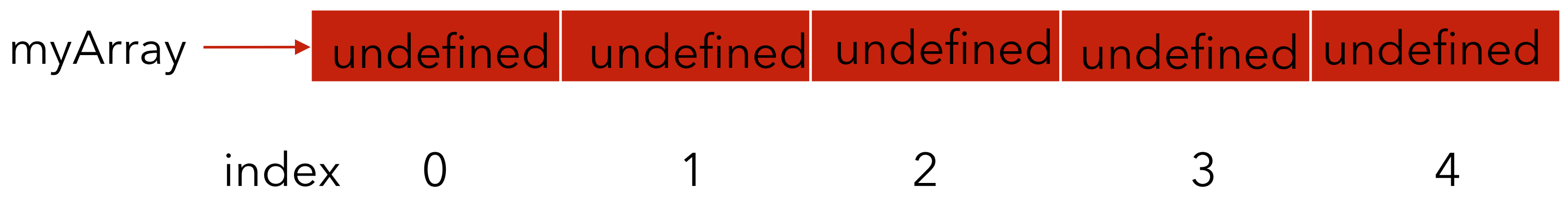


METHOD 2 TO CREATE ARRAYS

- Array() constructor

```
let myArray = new Array(5);
```

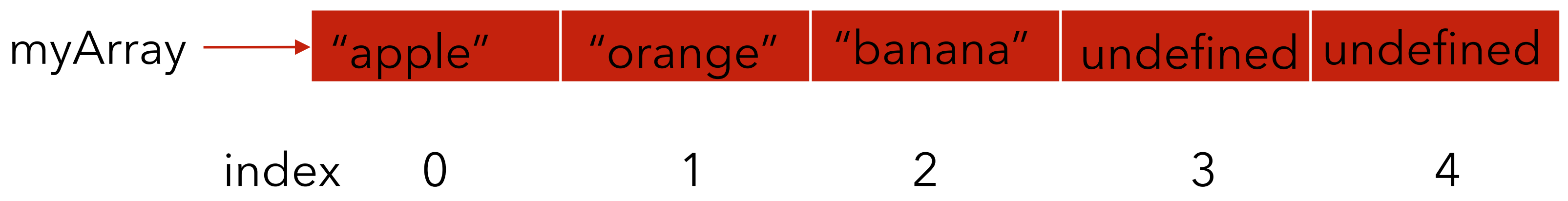
- Creates an array of length 5, with the value “undefined” stored in each space



METHOD 2 TO CREATE ARRAYS

- Then we can fill each space the array directly

```
myArray[0] = “apple”;
myArray[1] = “orange”;
myArray[2] = “banana”;
```



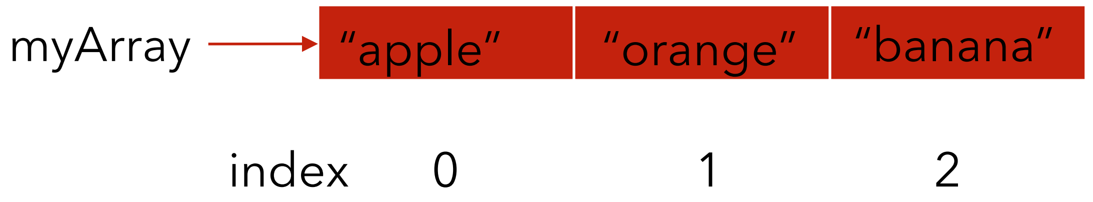
ACCESSING ARRAY ITEMS

- We can store items in the array and get items from the array
- We need two pieces of information to do this
  - Array name and index the item is stored at

`console.log(myArray[0])` prints "apple"

`console.log(myArray[1])` prints "orange"

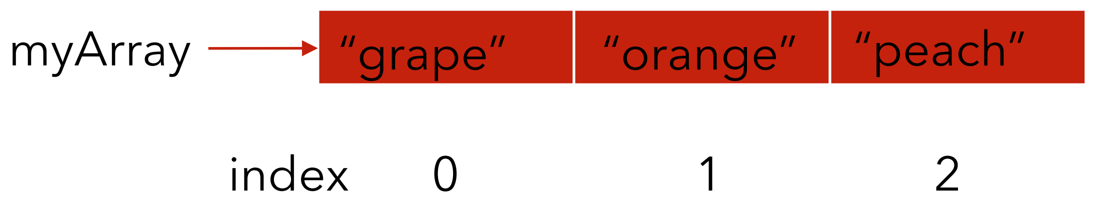
`console.log(myArray[2])` prints "banana"



MODIFYING ARRAY ITEMS

- We still need the same two pieces of information to do this

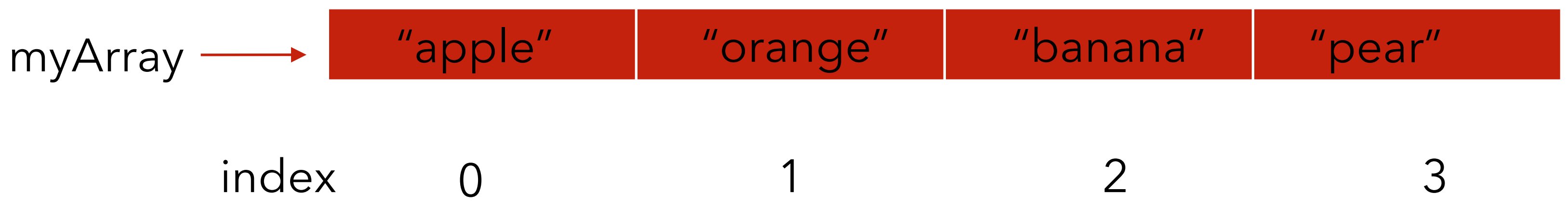
```
myArray[0] = “grape”;
myArray[2] = “peach”;
```



ADDING ITEMS TO AN ARRAY

- Arrays in JavaScript can change size (this is not true in other languages)
- So we can add new elements to our arrays
- The function that allows us to do this is **`.push(item)`**

```
let myArray = [“apple”, “orange”, “banana”];
myArray.push(“pear”);
```

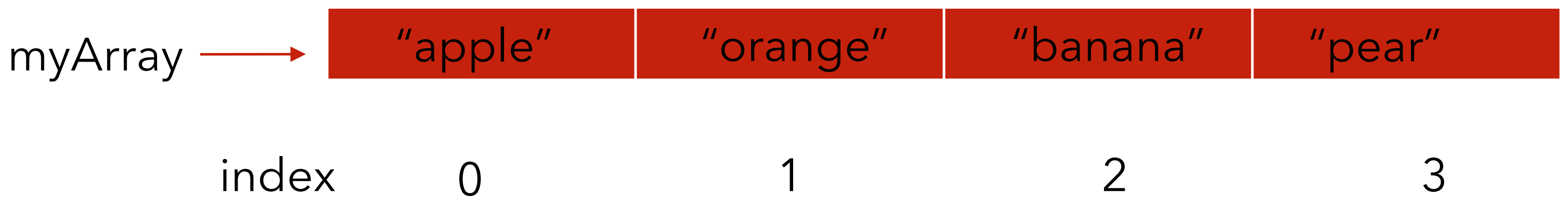


REMOVING ITEMS FROM ARRAY

- We can also remove items from **the end of** an array
- The function that allows us to do this is pop()
  - pop() will remove the LAST item from the array

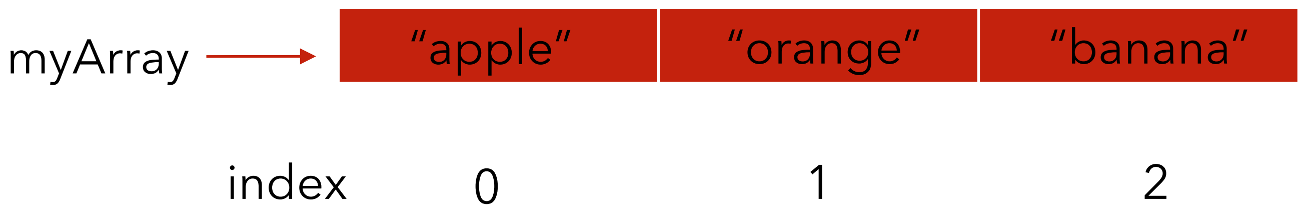
REMOVING ITEMS FROM ARRAY

```
let myArray = [“apple”, “orange”, “banana”, “pear”];
```



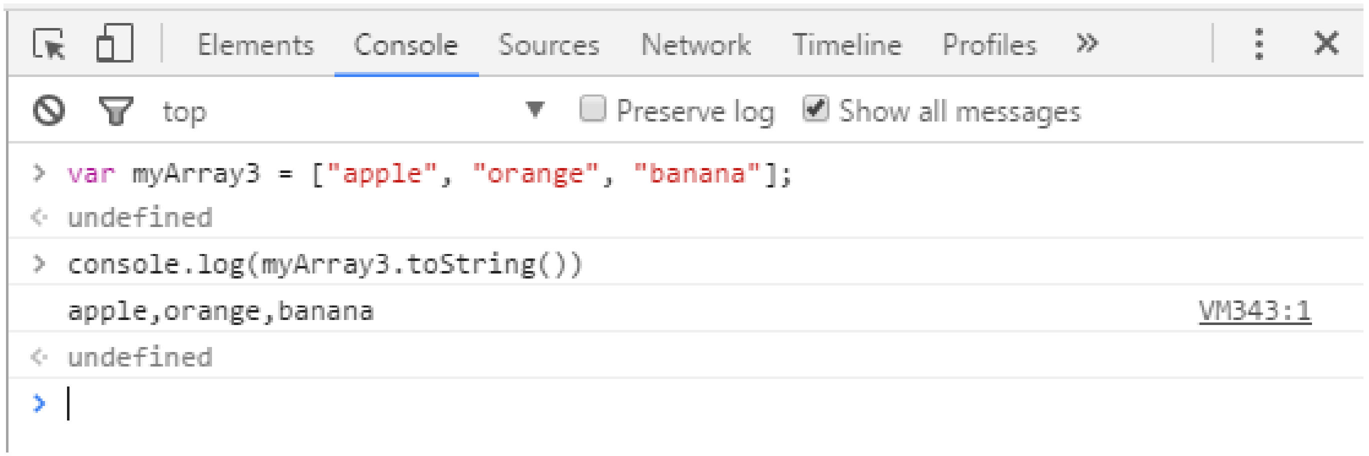
```
let item = myArray.pop();
```

Then item contains “pear”



PRINTING ARRAYS

- You can print the contents of an entire array to the console using
- **`console.log(myArray.toString());`**



```
> var myArray3 = ["apple", "orange", "banana"];
< undefined
> console.log(myArray3.toString())
  apple,orange,banana                                    VM343:1
< undefined
> |
```

EXERCISE

- Write the JavaScript to create an array that stores the 12 months of the year
- Then modify the printMonthName function to use your array but produce the same output

```
function printMonthName(m){
       if      (m ==  1) console.log("Jan");
       else if (m ==  2) console.log("Feb");
       else if (m ==  3) console.log("Mar");
       else if (m ==  4) console.log("Apr");
       else if (m ==  5) console.log("May");
       else if (m ==  6) console.log("Jun");
       else if (m ==  7) console.log("Jul");
       else if (m ==  8) console.log("Aug");
       else if (m ==  9) console.log("Sep");
       else if (m == 10) console.log("Oct");
       else if (m == 11) console.log("Nov");
       else if (m == 12) console.log("Dec");
}
```

DID YOU NOTICE?

- Did you notice that the array index starts at 0 and counts the number of items stored in the array?
- Did that make you think “Hey, I could probably use a for loop with arrays!”
- If you thought this, you’re right, and you’re thinking like a computer scientist.

EXAMPLE

- Let's create an array of size 5 and fill it with zeroes

```
let zeroArray = new Array(5);
for(let i = 0; i < zeroArray.length; i++){
     zeroArray[i] = 0;
}
```

- What do you think zeroArray.length returns?
- Why did we use < and not <= ?

FOR LOOPS AND ARRAYS

- Because we can access each element of an array via an index, it makes sense that we can then process arrays with loops
- Ex. Let's try to double the value of each element stored in an array

```
let someNums = [5,10,20,30];
for(let j = 0; j < someNums.length; j++){
      someNums[j] = someNums[j] * 2;
}
```

- Can you think of another way we could have written this?

COPYING ARRAYS

- We can use the slice() method to create copies of arrays
- slice() method returns the selected elements in an array, as a new array object
- slice() method selects the elements starting at the given *start* argument, and ends at, *but does not include*, the given *end* argument
- If we don't specify any parameters the whole array is copied

```
let fruits =
["Banana", "Orange", "Lemon", "Apple", "Mango"];
let citrus = fruits.slice(1, 3);
let variety = fruits.slice();
```

FINDING VALUES IN ARRAYS

- The indexOf(someValue) method searches through the array and looks to see if someValue is stored in the array
- If someValue is in the array, the method returns the first index the value is located at
- If someValue is not in the array, the method will return -1

```
let fruits =
["Banana","Orange", "Lemon", "Apple", “Lemon"];
console.log(fruits.indexOf(“Lemon”));
console.log(fruits.indexOf(“Potato”));
```

EXERCISE

- What is the index of Big White in the following array?

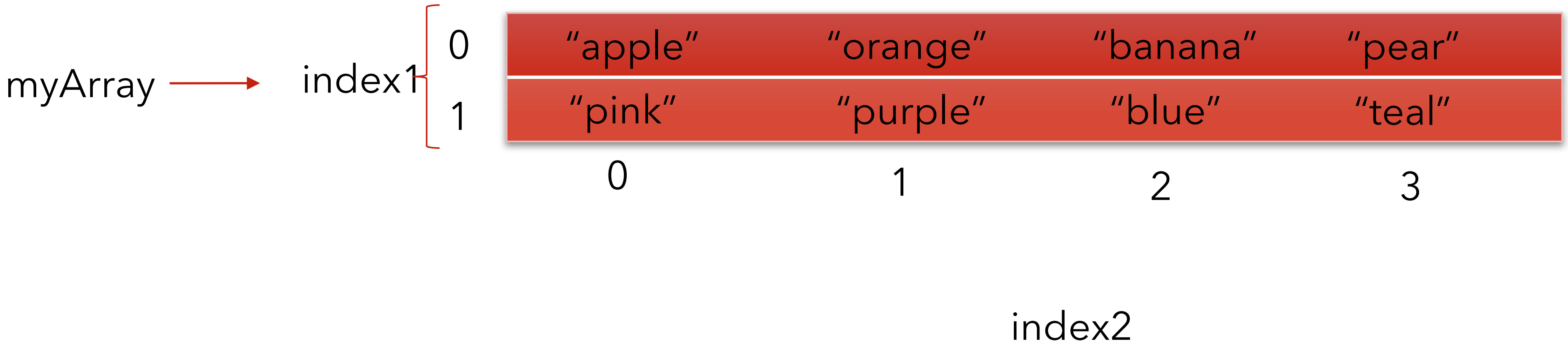
```
let resorts = ["Whistler", “Silverstar",
                "Big White", “Revelstoke",
                "Sun Peaks", "Red Mntn"];
```

- Write an expression that refers to the string Revelstoke within the array.
- What is the value of the expression resorts.length?
- What is the index of the last item in the array?
- What is the value of the expression resorts[5]?
- Write an expression to find the first index of "Silverstar" within the array

2D ARRAYS

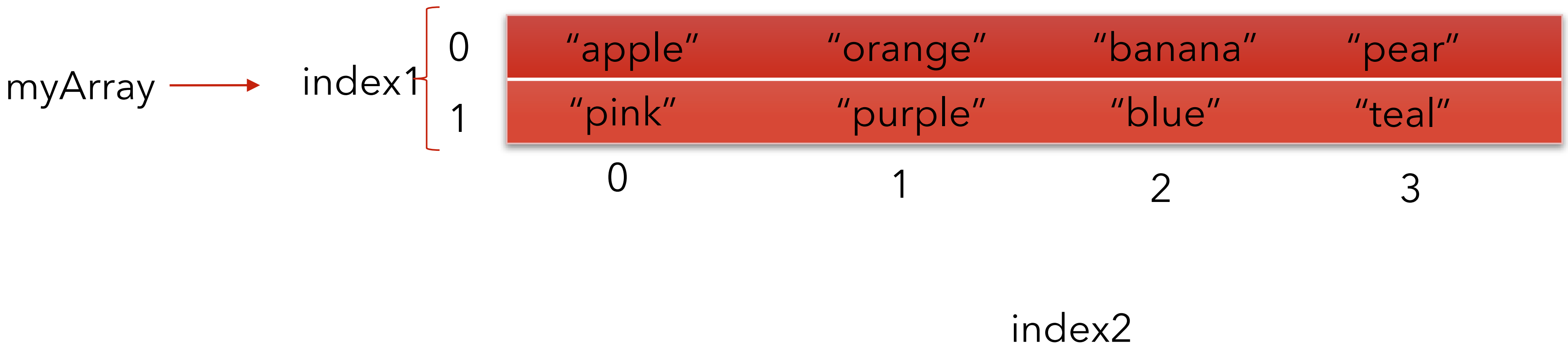
- 2D arrays are just arrays that store arrays
- They are handing for representing grids or tables of information in our programs

```
let myArray = new Array(2);
myArray[0] = [“apple”, “orange”, “banana”, “pear”];
myArray[1] = [“pink”, “purple”, “blue”, “teal”];
```



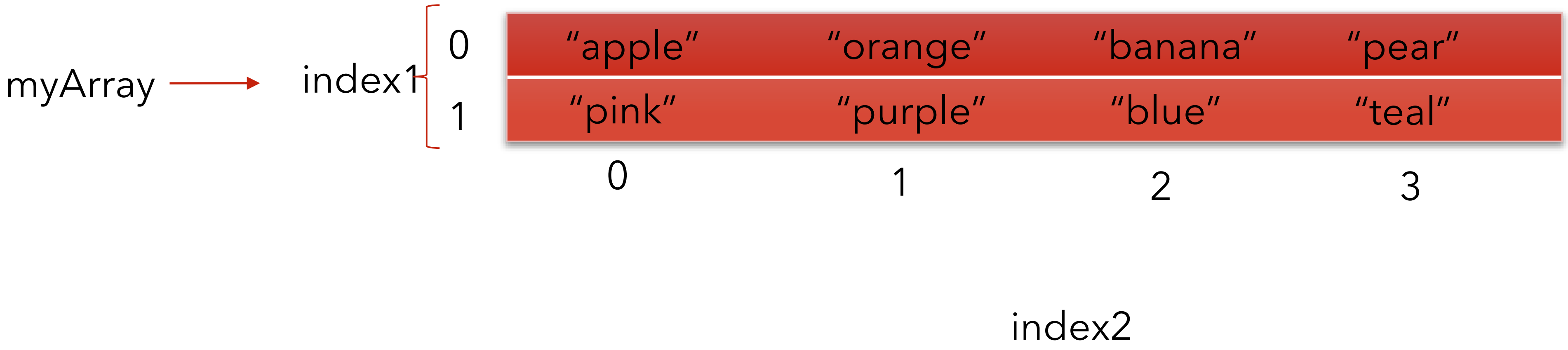
2D ARRAYS

- **`myArray[1] is the array [“pink”, “purple”, “blue”, “teal”]`**
- **`myArray[0][0] is apple`**
- **`myArray[1][1] is purple`**



QUESTION

- **`What is stored at myArray[0][3]?`**
- **`What is stored at myArray[1][2]?`**



EXERCISE

- Write the JavaScript to create a 2D array. This array should store the following names and midterm exam grades for each of the following students:
    - Aman100
    - Brad 75
    - Manpreet 75
    - Soren 50
    - Teika 25
- Now, use a for loop to calculate the average of the midterm exam grades stored in the array

ADDING ITEMS TO ARRAYS

- **`myArray.splice(position, remove, add);`**
- Parameters:
  - position: position to splice at
  - remove: number of elements to delete
  - add: the elements to add

```
let array = ["one", "two", "four"];
array.splice(2, 0, "three");
```

- **array would contain ["one", "two", "three", "four"]**

ADDING ITEMS TO ARRAYS

- The splice method returns an empty array when no elements are removed
- otherwise it returns an array containing the removed element

```
let ar = [1, 2, 3, 4, 5, 6];

let item = ar.splice(3, 1, “a”, “b”, “c”);

console.log(item);

//prints 4

console.log( ar.toString());
//prints 1,2,3,"a","b","c",5,6
```

PUTTING ARRAYS TOGETHER

- the concat() method will join two or more arrays together
- This method doesn't change the existing array, it returns a new array containing the values of all joined arrays

```
let heroes = ["Batman", "Robin"];
let villains =
["Joker","Penguin","Riddler"];


let characters = heroes.concat(villains);
```

REFERENCE VARIABLES

- What do you think the following code does?

```
let myArray = [1,2,3];
let myArray2 = myArray;

myArray2[0] = 5;
console.log(myArray2.toString());
console.log(myArray.toString());
```

REFERENCE VARIABLES

```
let myArray = [1,2,3];
let myArray2 = myArray;
```

- myArray and myArray 2 point to the same array
- when we modify one we are modifying both
- if we want myArray and myArray2 to be two different, distinct arrays, what can we do?

REFERENCE VARIABLES

- if we want myArray and myArray2 to be two different, distinct arrays, what can we do?
- Create a **copy** of myArray using slice

```
let myArray = [1,2,3];
let myArray2 = myArray.slice();
```

EXERCISE

- Given the following array

```
let arr = ["dog", "cat", "bird"];
What is the value of result:
let result = arr[0] = arr[2];
```

- Write a function called oddArray(N) that accepts the size of an array as input. This function should then return an array filled with the first N odd numbers.