Name _____ Period _____ Role (Circle one) Programmer/Driver

Name _____ Period _____ Role (Circle one) Programmer/Driver

Arrays

Your Tasks (Mark these off as you go)

- ☐ Explain the purpose of an array
- ☐ Create an array
- ☐ Add and remove items from an array
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Access items in an array
- ☐ Update an item in an array
- ☐ Remove (or add) an item from (or to) a specific index
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Get the length of an array
- ☐ Get input from the user
- ☐ Have Ms. Pluska check off the above tasks
- ☐ Brainstorm a program
- ☐ Receive credit for the group portion of this lab

☐ Explain the purpose of an array

In our day-to-day lives we often encounter the need to make lists. For example, if we are packing for a trip we can make a packing list; if we are going to the grocery store, we may bring a grocery list. In this lesson, we will start looking at how we can use lists in our programs to organize data. The short video below explains,

An **array** is a specific data structure that stores data as a list

What are some examples in your day-to-day life that can be represented with a list or an array?

□ Create an array

Arrays have many features which make them different from variables, but most of what you've learned about variables also applies to arrays. For example, just like a variable:

- Arrays should be given a descriptive and meaningful name.
- Arrays are created using the *var* key word.
- Arrays can be initialized/set using the equals sign, =

The code below illustrates how to create an array in javascript

```
var myFirstArray = [100, 250, 500]
```

This array above contains 3 values: 100, 250, 500. Notice that the values are separated with commas (,) and that the entire array is enclosed in brackets, [ ]. We can use *console.log* to display the contents of an array just like we would a variable.

```
console.log(myFirstArray
```

For each of the lists you identified in your every day life, write code that could be used to store the items in an array.

□ Add and remove items from an array

In our last exercise we created our array and initialized it with some values. Another way to do this is to add items to your array on separate lines. The simplest way to do this is to add a new item to the end of your array using the *push* command.

```
var groceryList = [ ] ;
groceryList.push(“apples”);
groceryList.push(“carrots”);

var primeNumbers = [ ];
oddNumbers.push(3);
oddNumbers.push(5);
```

The *push* command, *pushes* an item to the end (or top) of the array. You think of building arrays in javascript, like building a stack of books – each new item gets placed on top of the previous,

`var groceryList = [ ] ;`
`groceryList.push(“apples”);`
`groceryList.push(“carrots”);`
`groceryList.push(“coffee”);`
	`coffee`
	`carrots`
	`apples`

Items can be removed from an array using the *pop* command. The pop command pops the item at the top of the array off the stack,

`groceryList.pop();`	`carrots` `apples`

(a) Create an empty array called oddNumbers (b) Use the *push* command to add the odd numbers 1-11 to your array.

(a) Use the pop command to remove the numbers in the oddNumbers array you created above

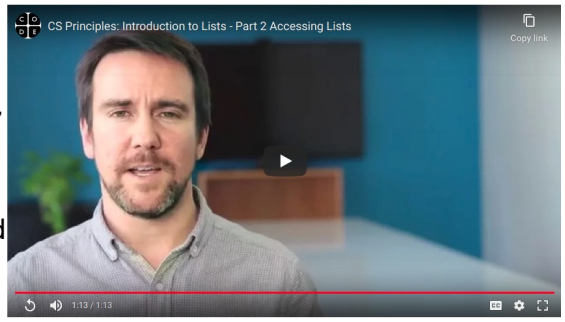
□ Have Ms. Pluska check off the above tasks

Before you continue have Ms. Pluska check off the above tasks

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature _____

□ Access items in an array

An array is comprised of many locations. You can individually set or reference the information at each location of your array just like a variable. To tell your locations apart each has a separate number, or index, that identifies it. This concept is explained in the video right,



As explained in the video, arrays in JavaScript are said to be *zero-indexed*, which means the first index is 0. For example an array of 10 items would have indexes 0-9. As a result the last index is always one less than the length of the array.

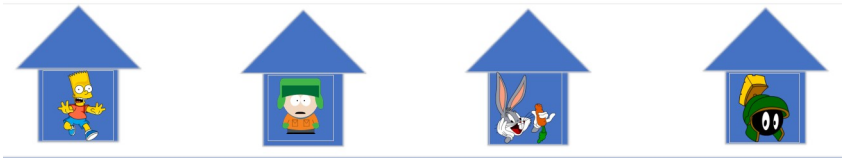
Index refers to the location of an element in an array. The first Index in an array is 0.

If you know the index of the item you wish to access you can reference it using square brackets, list[index]. This is illustrated below,

```
var groceryList = [“apples”, “carrots”, “coffee”];

console.log(groceryList[0]);//logs apples
console.log(groceryList[1]);//logs carrots
console.log(groveryList[2]);//logs coffee
```

The image below represents an array of String type variables called houses. The value associated with each house corresponds to the name of the person who lives there.

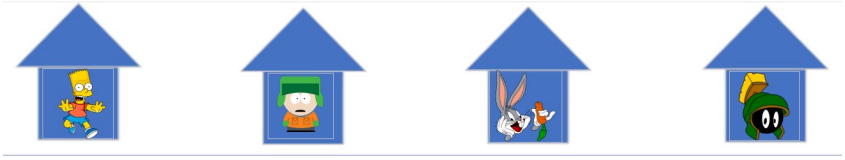


Write the address of each house on the roof.	
Who lives at index = 0?	
Who lives at index = 2?	
If the houses on the street represent an array, how long is the array?	
Who lives at index = 4?	
What is Marvin’s address?	
What is Kyle’s address?	
Write code to display the array in alphabetical order in the console: Bart, Bugs, Kyle, Marvin	

□ Update an item in an array

Each location in an array can be treated like its own variable. We've already seen how we can use bracket notation to reference values stored at specific locations in an array. Just like with variables, we can assign the value of a specific location in an array using "=" (the assignment operator). In the example below, the value of the element at index 0 is set to 10,

```
MyFirstArray[0] = 10;
```

The image below represents an array of String type variables called houses. The value associated with each house corresponds to the name of the person who lives there. 
Write code that could be used to assign the value of house 3 to "Wilma", and the value of house 2 to "Barney", and the value of house 1 to "Homer"
Homer and Barney have decided to trade houses. Write code to assign Homer and Barney to their new homes.

□ Remove (or add) an item from (or to) a specific index

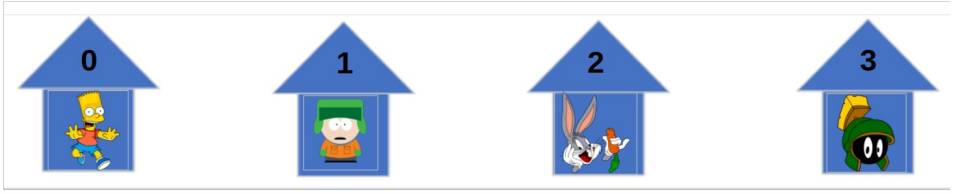
Previously we learned how to remove an item from the end of an array using the *pop* method. We also learned how to add an item to the end of an array using the *push* method.

The *splice* method can be used to add or remove elements from an array at a specific index. The first argument specifies the location at which to begin adding or removing elements. The second argument specifies the number of elements to remove. The third and subsequent arguments are optional; they specify elements to be added to the array.

Let's consider the example below.

The code below could be used to create the array shown,

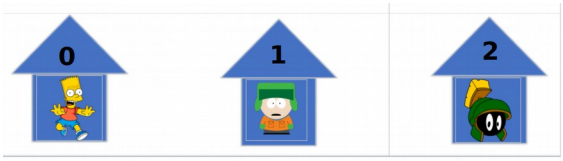
```
houses = [ ] ;
houses[0] = “Bart”, houses[1] = “Kyle”, houses[2] = “Bugs”, houses[3] =
“Marvin”;
```



The splice command can be applied to remove Bugs.

```
houses.splice(2, 1);
```

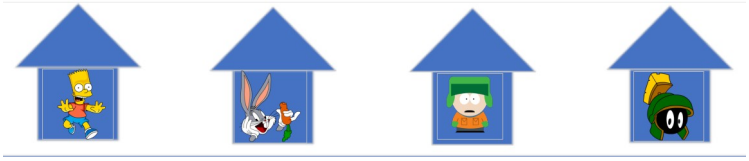
After the code above is executed, the houses array looks as follows,



The splice command can also be applied to add elements to an array,

```
houses.splice(1, 0, “Bugs”);
```

After the code above is executed, “Bugs” is added to position 1. Kyle and Marvin are shifted to positions 2 and 3 respectively.



The splice method can be used to add or remove items from an array, each application is summarized below,

Removing items

```
var houses = ["Bart", "Bugs", "Kyle", "Marvin"];

houses.splice(1,1);
```

The name of the array

The index which removing begins

The number of items to remove

The houses array after the code to the left is executed becomes,

```
var houses = [“Bart”, “Kyle”, “Marvin”];
```

Adding items

```
var houses = ["Bart", "Kyle", "Marvin"];

houses.splice(1,0,"Homer");
```

The items to add separated by commas

The name of the array

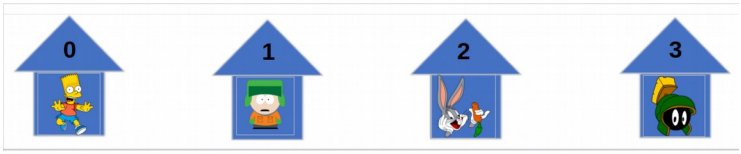
The index which adding begins

The number of items to remove. If you only want to add items, indicate 0

The houses array after the code to the left is executed becomes,

```
var houses = [“Bart”, “Homer”, “Kyle”, “Marvin”];
```

The image below represents an array of String type variables called houses. The value associated with each house corresponds to the name of the person who lives there.



(a) Bugs got a new job and moved out. Write code to remove Bugs.

(b) Two new houses are being built at the end of the block. “Kenny” and “Stan” are moving in. Write code to add the two new neighbors to the end of the houses array.

□ Have Ms. Pluska check off the above tasks

Before you continue have Ms. Pluska check off the above tasks

Do not continue until you have Ms. Pluska’s (or her designated TA’s) signature _____

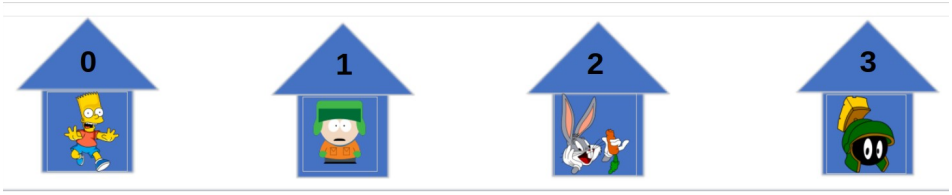
□ Get the length of an array

Because arrays can grow and shrink as items are added and removed, it is often useful to know the length of an array at a given time. The short video below explains,

As the video explains the following syntax can be used to find the length of an array,

```
myArray.length;
```

For the houses array below, the length is 4. However, do not confuse length with the location of the last element! Notice, that because array indexing begins at 0, the last index is always 1 less than than the length, or 3, in the example below.



The code below could be used to access the last element in the houses array shown above,

```
console.log(houses[houses.length – 1]); //prints Marvin
```

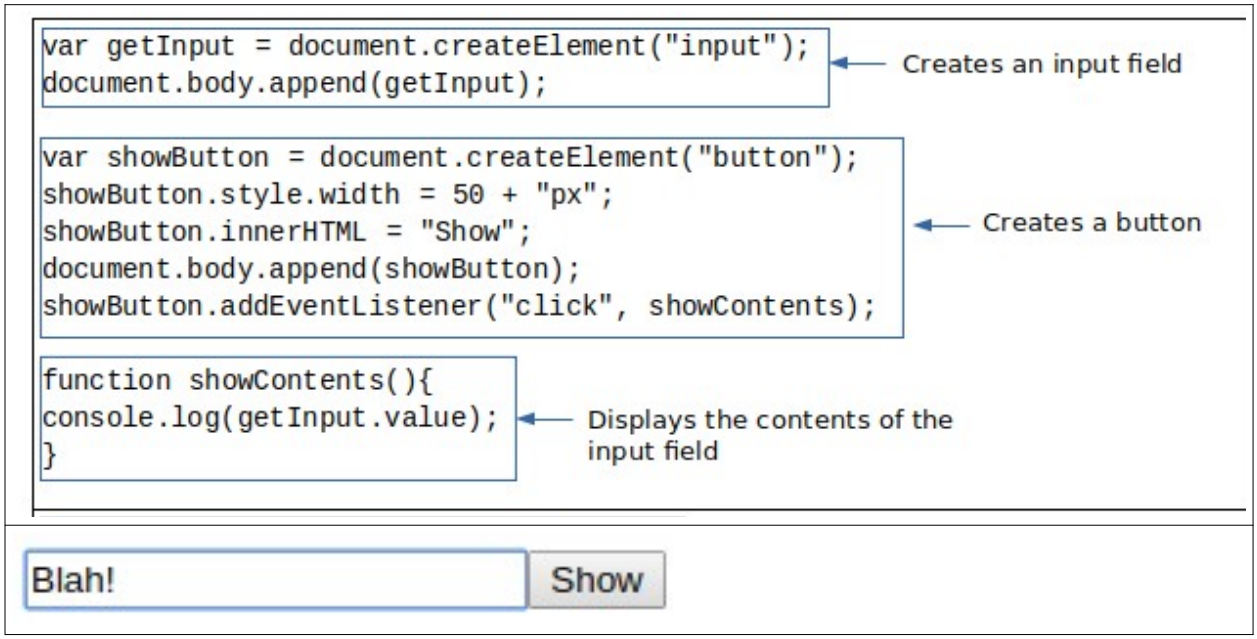
Consider the cards below, which can be represented as an array named cards.

(a) Write code that could be print the length of the cards array

(b) Write code that could be used to print the last card in the array

□ Get input from the user

Up until now the only way we could get input from the user was from a browser prompt. The HTML provides an input element that makes this process easier and more seamless. Consider the example below,



The code below creates an input field called *getInput*

```
var getInput = document.createElement(“input”);
```

The code below accesses the value the user input and stores it in a variable called *userInput*.

```
var userInput = getInput.value;
```

Write code that could be used to create a shopping list. (a) Declare an array called shoppingList (b) Create an input field (c) Create a button, that when clicked calls a function that adds the value of the item to the shoppingList (d) Once the user has five items, alert the user that their shopping list is full

□ Have Ms. Pluska check off the above tasks

Before you continue have Ms. Pluska check off the above tasks

Do not continue until you have Ms. Pluska's (or her designated TA's) signature _____

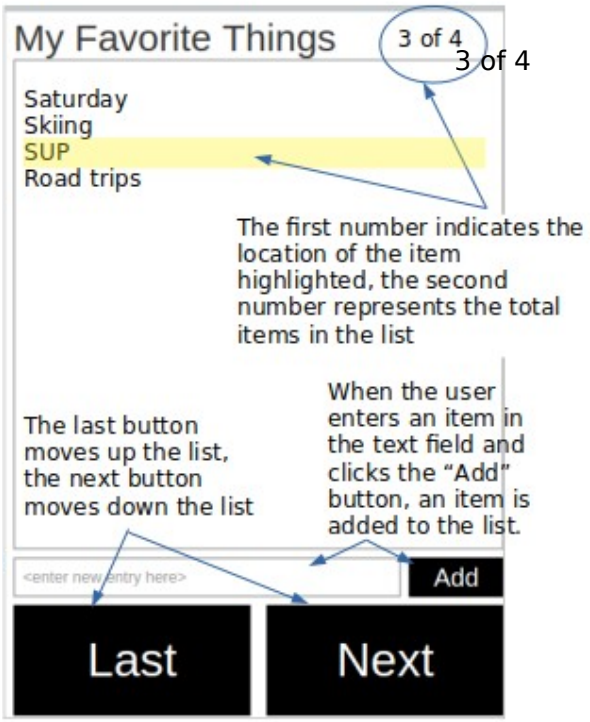
□ Brainstorm a program

Challenge 1

The image shown to the right represents a Graphical User Interface (GUI) for which you must make work as intended. A description of how each feature should function is provided.

The code for each feature should be organized into its own function. For example, the text which indicates "3 of 4" should be updated in its own function. There should also be function that controls the highlighting of the list items when last and next are clicked.

Write your pseudocode for both challenges on a separate sheet of paper. Your code should be legible and "make sense" to receive credit.



□ Receive Credit for the group portion of this lab

- Indicate the names of all group members.
- Make sure both you and your partner have completed the above tasks
- Have Ms. Pluska check off the group tasks
- Submit your lab to the needs to be graded folder to receive credit for the group portion of this lab.
- Do not submit your lab until you have Ms. Pluska's (or her designated TA's) signature

_____