Set 33: Arrays

Skill 33.01: Describe the purpose of an array
Skill 33.02: Create an array with JavaScript
Skill 33.03: Add or remove items from an array
Skill 33.04: Access items in an array
Skill 33.05: Update an item in an array
Skill 33.06: Remove (or add) an item from (or to) a specific index
Skill 33.07: Get the length of an array
Skill 33.08: Get input from the user

Skill 33.01: Describe the purpose of an array

Skill 33.01 Concepts

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

https://www.youtube.com/watch?v=KFy7u3Rhozs

Skill 33.01 Exercise 1

Skill 33.02: Create an array with JavaScript

Skill 33.02 Concepts

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];
```

The 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.

Code	Output
```var myFirstArray = [100, 250, 500];``` ```console.log(myFirstArray);```	▼ *(3) [100, 250, 500]* 0: 100 1: 250 2: 500 length: 3

Skill 33.02 Exercise 1

Skill 33.03: Add or remove items from an array

Skill 33.03 Concepts

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.

Code	Output

```var groceryList = [] ;```
 	
```groceryList.push("apples");```
```groceryList.push("carrots");```
 	
```console.log(groceryList);```
	▼ *(2) ["apples", "carrots"]*
	    0: "apples"
	    1: "carrots"
	    length: 2

```var primeNumbers = [];```
 	
```primeNumbers.push(3);```
```primeNumbers.push(5);```
 	
```console.log(primeNumbers);```
	▼ *(2) [3, 5]*
	    0: 3
	    1: 5
	    length: 2

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();
```
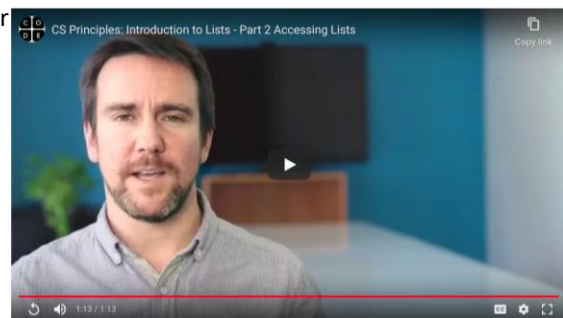
carrots
apples

Skill 33.03 Exercise 1

Skill 33.04: Access items in an array

Skill 33.04 Concepts

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**.

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,

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

Code	Output

``var groceryList = [] ;
groceryList.push("apples");
groceryList.push("carrots");
groceryList.push("coffee");
 	
console.log(groceryList[0]);
console.log(groceryList[1]);
console.log(groveryList[2]);```
	apples
	
	carrots
	
	coffee

Skill 33.04 Exercise 1

Skill 33.05 Concepts

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 other variables, we can assign or reassign the value of a specific location in an array using the assignment (=) operator. In the example below, we have written code to swap the values assigned to locations 0 and 1 in our *groceryList* array.

Code
| ```javascript
var groceryList = [] ;
groceryList.push("apples");
groceryList.push("carrots");
groceryList.push("coffee");

var temp = groceryList[0];//assigns apples to temp
groceryList[0] = groceryList[1];//assigns carrots to position 0
groceryList[1] = temp;//assigns apples to position 1

console.log(groceryList);
``` |
| **Output** |
| ```
▼(3) ["carrots", "apples", "coffee"]
    0: "carrots"
    1: "apples"
    2: "coffee"
    length: 3
``` |

Skill 33.05 Exercise 1

Skill 33.06 Concepts

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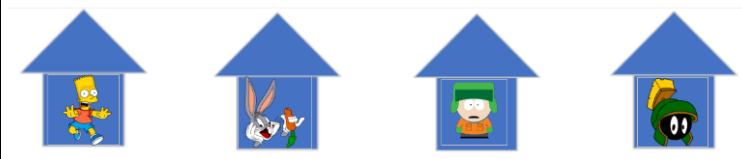
After the code above is executed, the houses array looks as follows,

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 number of items to remove
The name of the array
The index which removing begins
	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"];```

Skill 33.06 Exercise 1

Skill 33.07: Get the length of an array
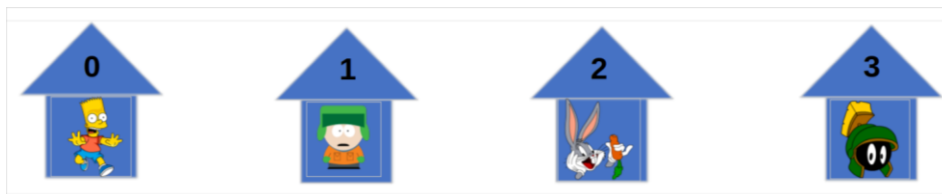
Skill 33.07 Concepts

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,

```
console.log(houses.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 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
```

Skill 33.07 Exercise 1

Skill 33.08: Get input from a user

Skill 33.08 Concepts

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* and appends to the body of the page

```
var getInput = document.createElement("input");
document.body.append(getInput);
```

The code below creates a button, assigns an action listener to the button, and appends it to the page,

```
var showButton = document.createElement("button");
showButton.style.width = "100px";
```

```
showButton.style.height = "25px";
showButton.innerHTML = "SHOW";
document.body.append(showButton);

showButton.addEventListener("click", getValue);
```

Finally, when the button is clicked the code below accesses the value the user input and stores it in a variable called *userInput*.

```
function getValue(){
    var userInput = getInput.value;
}
```

Skill 33.08 Exercise 1