Arrays

...

Lets talk about variables

Having a lot of variables can cause confusion and lead to many lines of code.

In the example below we have a list of tasks we want to complete today.

Example: var taskList1 = ‘Do Homework’;

var taskList2 = ‘Eat food’;

Var taskList3 = ‘Go to bed’;

This is a lot of code but don't worry arrays are here to help!

Arrays

Arrays are special variables that can hold multiple values

Example: var taskList1 = ‘Do Homework’;

var taskList2 = ‘Eat food’;

Var taskList3 = ‘Go to bed’;

Instead we can store our task list in an array as follows:

var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

How arrays work

Arrays are essentially a group of variables of the same type and javascript keeps track of where the values are stored in that list

The first position of an array is position 0

var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

Position: 0 1 2

Var Let and Const

You can use Var Let and Const to create your arrays

What you use depends on your use case

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

let taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

const taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

Lets try it

Create an array of whatever you want!

Example code:

var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

Sooo that's cool but how do I use it?

Simply type your arrays name to output it. If you want to reference a value in an array you need to reference its position

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

```
console.log(taskList[1]);
```

OUTPUT: Eat food

Lets try it

Output a value from your array either to the console or to an alert

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

Display full array:

```
console.log(taskList);
```

Display single element in array:

```
console.log(taskList[1]);
```

Lets try it! - You can display an array using innerHTML

```
<p id="arrayPlaceholder"></p>

var TaskList =  [‘Do Homework’, ‘Eat food’, ‘Go to bed’];
```

Display full array:

```
document.getElementById("arrayPlaceholder").innerHTML = TaskList;
```

Display single element in array

```
document.getElementById("arrayPlaceholder").innerHTML = TaskList[x];
```

Loops and Arrays

When you need to access each element in an array a loop is a great solution

You can use any type of loop and if statements to help filter and define specific types of information you want to pull from a loop

Always make sure you have a counter when using an array and that counter does not exceed the arrays length

Array length

To get the length of an array you can use ARRAY_NAME.length

```
var taskList = ["Do Homework", "Eat food", "Go to bed"];
console.log(taskList.length);
```

OUTPUT: 3

Looping through an Array Demo

Adding (push) a value to an array

To add a value to an array use the .push method. This will add a value to the end of the array

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList.push(‘Do the dishes’);

OUTPUT Array = [‘Do Homework’, ‘Eat food’, ‘Go to bed’, ‘Do the dishes’];

Lets try it

```
var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList.push(‘Do the dishes’);
```

Removing (pop) a value from the end of an array

To remove a value to an array use the .pop method. This will remove a value at the end of the array

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

```
taskList.pop();
```

OUTPUT Array = [‘Do Homework’, ‘Eat food’];

Lets try it

```
var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList.pop();

console.log(taskList);
```

Removing (shift) a value from the start of an array

To remove a value to an array use the .shift method. This will remove a value at the start of the array

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

```
taskList.shift();
```

OUTPUT Array = [‘Eat food’, ‘Go to bed’];

Lets try it

```
var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList.shift();

console.log(taskList);
```

These methods return the value of push, pop, shift, etc.

Note: These methods also return the value in the array location.

```
var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];
let popValue = taskList.pop();
alert(popValue);
```

OUTPUT:

Array ‘taskList’ will equal: [‘Do Homework’, ‘Eat food’]

Alert will be: “Go to bed”

Replace a value in an array

To replace a value in an array reference its position and assign it a new value.

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList[1] = ‘Eat breakfast’;

OUTPUT Array = [‘Do Homework’, ‘Eat breakfast, ‘Go to bed’];

Lets try it

var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

taskList[1] = ‘Eat breakfast’;

Removing (delete) a specific value from an array

You can remove a specific value from an array but BE CAREFUL! When you remove that value it leaves an empty spot in your array which can cause errors.

The recommended way to remove values is to replace them with another value.

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

delete tasklist[0];

OUTPUT Array = [, ‘Eat food, ‘Go to bed’];

Lets try it!

Remove (delete) a value in your array.

Example: var taskList = [‘Do Homework’, ‘Eat food’, ‘Go to bed’];

delete taskList[0];

Combine (Concat) arrays

You can combine two arrays into one by using concat

Note that you need to create a new array to store the two combine arrays

```
var taskList = ["Do homework", "Eat food"];

var taskList2 = ["Sleep", "Dont forget the dishes"];

var fullTaskList = taskList.concat(taskList2);
```

Split (slice) arrays

You can split or pull certain values in an array into a new array by using the slice function. To do this define what position you want to start your new array at.

```
var taskList = ["Do homework", "Eat food", "Sleep", "Dont forget the dishes"];

var smallerTaskList = taskList.slice(2);

OUTPUT Array: ["Sleep", "Dont forget the dishes"];
```

Lets try it

Split an array and combine two arrays

Split:

```
var taskList = ["Do homework", "Eat food", "Sleep", "Dont forget the dishes"];
var smallerTaskList = taskList.slice(2);
OUTPUT Array: ["Sleep", "Dont forget the dishes"];
```

Concat:

```
var taskList = ["Do homework", "Eat food"];
var taskList2 = ["Sleep", "Dont forget the dishes"];
var fullTaskList = taskList.concat(taskList2);
```

Need an empty array? No problem!

There will be instances in future assignments where we will need an empty array to store values in. There are two ways to create an empty array

```
var taskList = new Array();

var taskList = [];
```

Filter array using IF statement Demo

Wrap up

There are A LOT of things you can do with an array and ways to manipulate arrays

Typically you will only be adding values to arrays or looping through them to display information