

ARRAYS AND LOOPS

in JavaScript



ARRAYS

```
var fruits = ['banana', 'orange', 'apple'];
```

- Ordered collection of data combined into one variable.
- Each item in an array gets assigned an index value.

ARRAYS

```
var fruits = ['banana', 'orange', 'apple'];
```

```
fruits[0]; // will output 'banana'
```

```
fruits[1]; // will output 'orange'
```

```
fruits[2]; // will output 'apple'
```

-You call array variables using square brackets and putting their index value into the brackets. Notice anything weird?

[0]

The index value arrays always start with 0,
no exceptions.
It is never 1.

ARRAYS

Setting values at a given index

```
var fruits = ['banana', 'orange', 'apple'];  
  
fruits[0] = 'grapes';  
  
// fruits will now be:  
// ['grapes', 'orange', 'apple']
```

This will override any value that was
there before

CODEALONG

Let's make some arrays and
access their values



Array - **Index** - length - indexOf() - pop() - shift()
push() - unshift() - join() - Loop - for

.length

```
var fruits = ['banana', 'orange', 'apple'];  
fruits.length; // will output 3
```

- Use length to figure out how many items are in your array

.indexOf()

```
var fruits = ['banana', 'orange', 'apple'];  
  
fruits.indexOf('orange');  
// will output 1
```

-Use `indexOf()` to see what index value any item in the array has

.pop()

```
var fruits = ['banana', 'orange', 'apple'];  
  
fruits.pop();  
// fruits = ['banana', 'orange'];
```

-Pop method takes off the LAST item in the array.

.shift()

```
var fruits = ['banana', 'orange', 'apple'];  
  
fruits.shift();  
// fruits = ['orange', 'apple'];
```

-Shift method takes off the FIRST item in the array.

.push()

```
var fruits = ['orange'];  
  
fruits.push('kiwi');  
// fruits = ['orange', 'kiwi'];
```

-Push method takes whatever item you have within the method parenthesis and adds it to the END of the array, creating a new item.

.unshift()

```
var fruits = ['orange', 'apple'];  
  
fruits.unshift('kiwi');  
// fruits = ['kiwi', 'orange', 'apple'];
```

-Unshift method takes whatever item you have within the method parenthesis and adds it to the front of the array

.join()

```
fruits = ['kiwi', 'pear', 'cherry'];  
  
fruits.join(' and ');  
// fruits = 'kiwi and pear and cherry';
```

Join will take the value from within the parentheses and combine it with all values in your array to make one big string.

MULTI - DIMENSIONAL

```
var produce = [  
  ['kiwi', 'pear', 'cherry'],  
  ['broccoli', 'celery', 'carrots']  
];
```

```
produce[1] // => ['broccoli', 'celery', 'carrots'];  
produce[0][2] // => 'cherry';
```

You can put arrays inside of arrays. Access them with a second set of square brackets (first bracket is the array, second bracket is item).

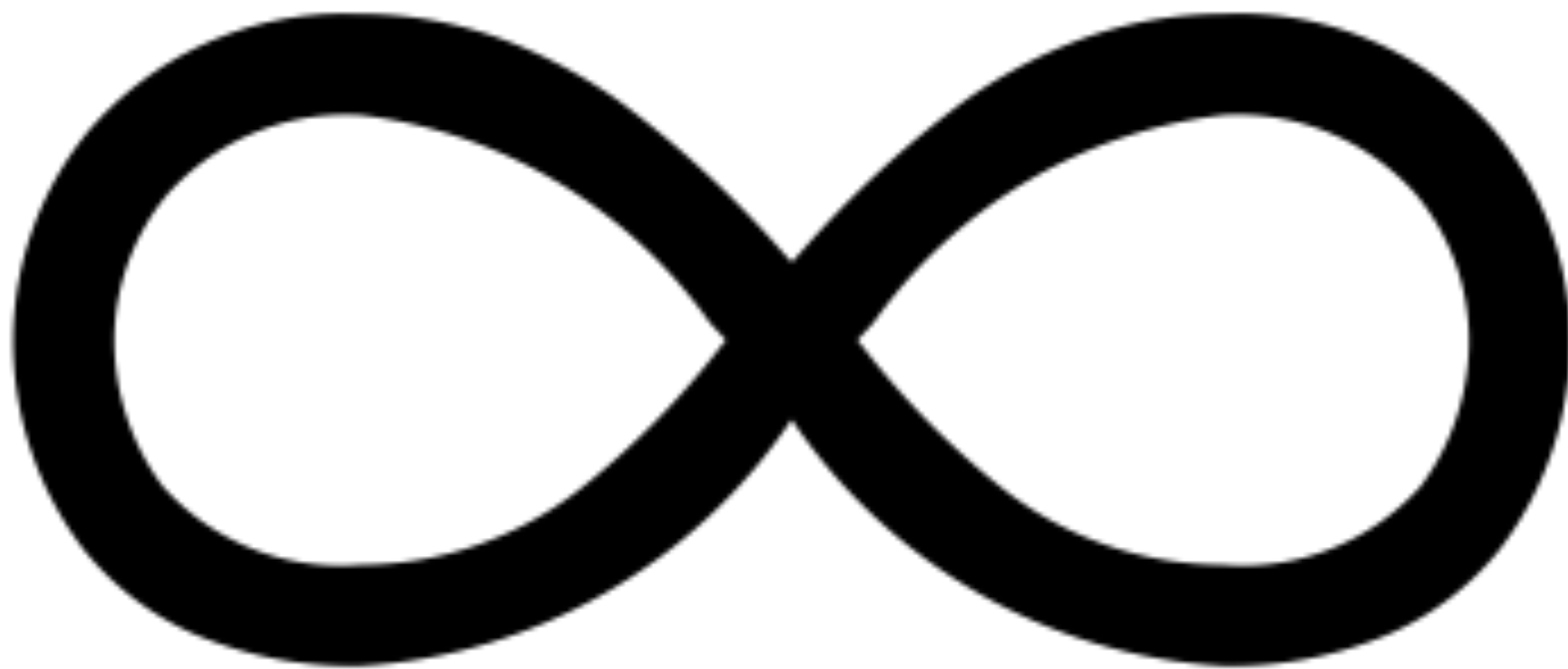


Array - Index - length - indexOf() - pop() - shift()
push() - unshift() - join() - Loop - for

YOUR TURN

Complete the tasks in the repl

LOOPS



LOOP

Repetitious program that runs over
a defined range.

WHY LOOPS?

- Loops take advantage of what computers do best: evaluate instructions across organized sets of data very quickly.
- Computers think best in isolated patterns, which is exactly how a loop works.
- Efficient in terms of memory and processor usage

FOR LOOP

Works similar to an if statement but with more conditions. Three declarations:

- 1- Define variable
- 2- Establish condition for loop to run
- 3- Increment variable

```
for (var i = 0; i < 10; i++) {  
    console.log(i);  
    // outputs 0,1,2,3,4,5,6,7,8,9  
}
```

FOR LOOP & ARRAYS

Blending the two concepts, we can create powerful programs that move through large amounts of data very quickly.

FOR LOOP & ARRAYS

Same number of declarations but notice the condition in the parenthesis. The array's length limits the amount of loops.

```
var myArray = ["John", "Benjamin", "Victor"];

for (var i = 0; i < myArray.length; i++) {
    console.log(myArray[i]);
    // outputs "John,Benjamin,Victor"
}
```

BIG IDEA

The major use case for arrays is to create itemized groups of data computers can manipulate (often with loops).

YOUR TURN

Complete the tasks in the repl



**Array - Index - length - indexOf() - pop() - shift()
push() - unshift() - join() - Loop - for**