

Arrays

tl;dr

- How to declare arrays, read elements, and change elements
- Spread operator (...)
- Loops
 - for-in
 - for-of
 - Array.forEach
 - Traditional for
- Array.prototype.* functions like map, some, every, filter, reduce, map

Creating Arrays

```
days = new Array(); // Not recommended, but works
days = [];
days = [ 'Mon', 'Tues', 'Wed', 'Thu', 'Fri' ];
```

- Note that you don't specify a size

Reading and writing arrays

```
x = days[0]; // Mon
y = 1;
x = days[y+2]; // Thu
x = days.length; // 5
// Arrays are sparse, not dense
days[54] = ""; // Now days.length=55
x = days[1000]; // Not an error!! Merely undefined
```

Adding to & removing from arrays

```
let numElements = a.push(newVal);  
let thingRemoved = a.pop();  
let numElements = a.unshift(newVal);  
let thingRemoved = a.shift();
```



Looping through arrays

Top four ways to loop
through arrays

1. Old-school for
2. for-in
3. forEach()
4. for-of



The Old-school for loop

```
for (var i=0 ; i < array.length ; i++) {  
    console.log(array[i]);  
}
```

IE 1+



for-in

```
for (var i in array) {  
  console.log(array[i]);  
}
```

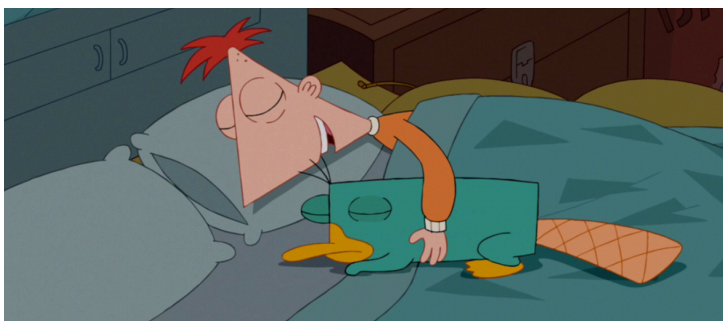
IE 6+



for-in was written to loop through objects

- Arrays just happen to work also!
- We can loop through the object keys with *for-in*

```
for (var prop in destructure) {  
  console.log(destructure[prop]);  
}
```



But be careful! *for-in* can be wrong.

```
var theGang= ["Buford", "Baljeet",  
             "Isabella", "Ferb", "Phineas" ];  
Array.prototype.today'sProject = "Backyard Beach";  
for (var kid in theGang) {  
    console.log(theGang[kid]);  
}  
//includes "Backyard Beach"
```

forEach

```
array.forEach(function (a) {  
    console.log(a);  
})
```

IE 9+



for-of

```
for (let a of array) {  
  console.log(a);  
}
```

Edge 13+



```
for (let kid of theGang) {  
  console.log(kid);  
});  
// Does not include "Backyard Beach"
```

Best practice: Use *for of* to loop
through arrays

Spread operator

... and rest parameters

Spread syntax

Spreads an
array into
values



`...someArray`

- In any JavaScript line
- Called "spread operator"
- Because it *spreads* out the array into its parts

Traditional way to include one array into another

```
var fourTo9 = [4, 5, 6, 7, 8, 9];  
var foo = [1, 2, 3, 10];  
fourTo9.reverse().forEach(x => {  
  foo.splice(3, 0, x);  
});  
// foo now has 1 - 10
```

To do the same thing with spread

```
const fourTo9 = [4, 5, 6, 7, 8, 9];  
const foo = [1, 2, 3, ...fourTo9, 10];  
// foo now has 1 - 10
```

Array functions

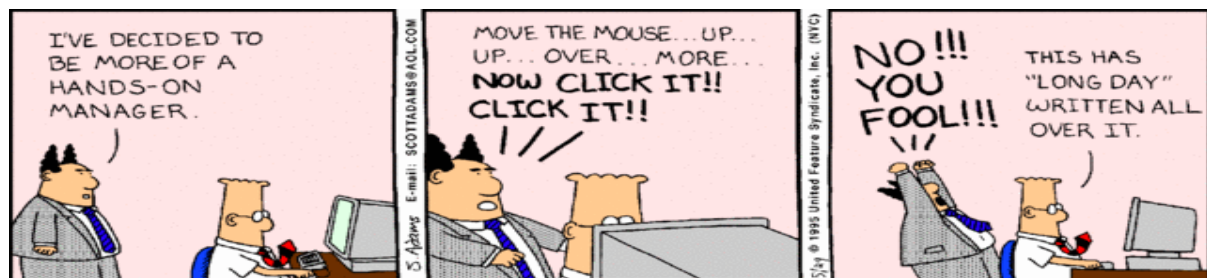
Looping arrays is usually too imperative!

```
let allAdults = true;
for (let p of people)
  if (p.age < 18)
    allAdults=false;
```

(meh).

Better!

```
let allAdults = people.every(p => p.age>18);
```



Array.find()

- Returns the **first thing** in the array that matches some criteria

```
const searchString = "Isabella";
const thePerson = people.find(p =>
    p.firstName === searchString);
console.log(thePerson);

// { id:2483,
//   firstName:"Isabella",
//   lastName:"Garcia-Shapiro"
// }
```

Array.findIndex()

- Returns the location of the first thing in the array that matches some criteria

```
const searchString = "Isabella";
const location= people.findIndex(p =>
    p.firstName === searchString);
console.log(location);

// 4303
```

Array.filter()

- When you want only the members that match some criteria.
- Returns a new, smaller array based on an old one.

```
var a = [1, 2, 3, 4, 5, 6, 9];  
var newArray =  
  a.filter(x => x % 2 === 0);  
console.log(newArray); // [2, 4, 6]
```

Array.map()

- When you want a new array the same size as an existing one.
- Processes each member of the old array.

```
var a = [1, 3, 5, 7, 9];  
newArray = a.map( x => x ** 2 );  
console.log(newArray); // [1, 9, 25, 49, 81]
```

Array.reduce()

- Returns a single value by running a function over each member of the array
- Great for currying!

```
let a = [21, 52, 83, 94, 15, 76, 39];  
let s = a.reduce((x,y) => Math.min(x,y));  
console.log(s); // 15
```

Array.every() & Array.some()

- When you want to see if all or some of the members match some criteria.
- myArray.every() - Do ALL members match some criteria?
- myArray.some() - Do ANY members match some criteria?

```
var a = [1, 2, 3, 4, 5, 6, 7, 9];  
allEven = a.every( x => x % 2 === 0 );  
console.log(allEven); // false  
someEven = a.some( x => x % 2 === 0 );  
console.log(someEven); // true
```

tl;dr

- How to declare arrays, read elements, and change elements
- Spread operator (...)
- Loops
 - for-in
 - for-of
 - Array.forEach
 - Traditional for
- Array.prototype.* functions like map, some, every, filter, reduce, map