
Getaway to
THE ARRAY ARCHIPELAGO

LEVEL 5

THE ARRAY ARCHIPELAGO

WHAT IF WE WANTED A PASSENGER LIST?

How would we structure a list of passengers inside our train.js system?

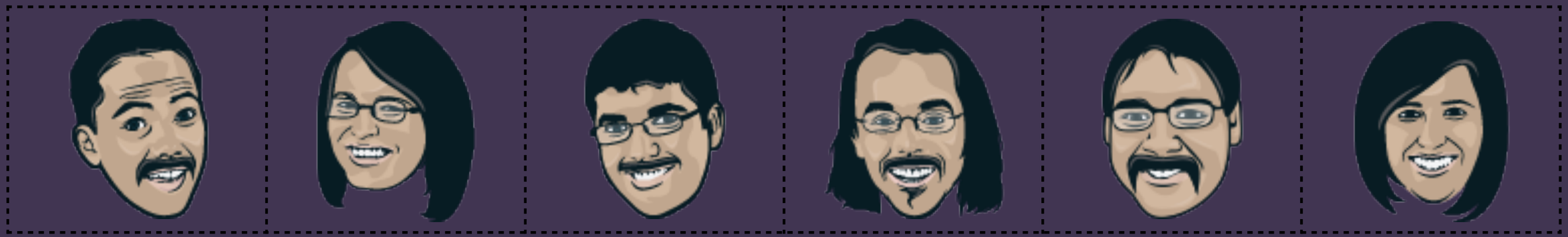
trains.js

```
...
function makeList ( ) {

    var passengerOne = "Gregg Pollack";
    var passengerTwo = "Aimee Simone";
    var passengerThree = "Thomas Meeks";
    var passengerFour = "Olivier Lacan";

    ...and on and on, typing through a list
    of sixty passengers, that might
    even change later?? No way.
}
...
```

THE ARRAY

An array is a data structure with automatically indexed positions

A 6-cell Array of Passengers



0 1 2 3 4 5

Just like Strings, Arrays have indices that are zero-based.

Despite his excellent disguise, it looks like Jon is in index 4. We mustache him a question.

ARRAY CELLS CAN HOLD ANY VALUE

Our picture array could also be an array of strings.

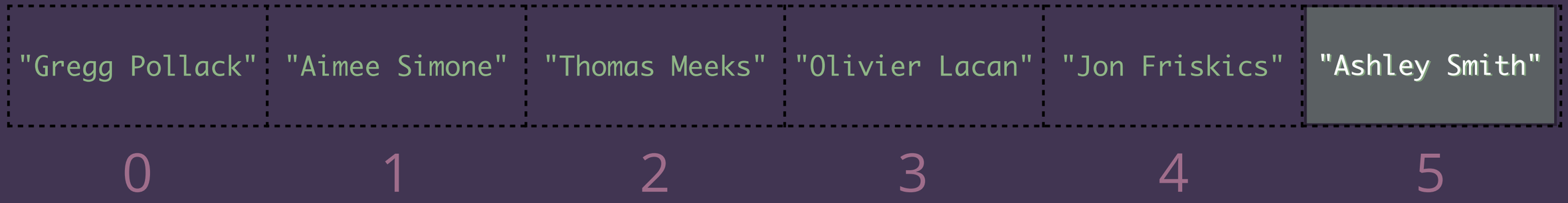
0 1 2 3 4 5

"Gregg Pollack"	"Aimee Simone"	"Thomas Meeks"	"Olivier Lacan"	"Jon Friskics"	"Ashley Smith"

0 1 2 3 4 5

BUILDING AND ACCESSING ARRAYS

Easy to build, easy to access with indices



To build this array in code, we write:

```
var passengers = [ "Gregg Pollack", "Aimee Simone", "Thomas Meeks",
                   "Olivier Lacan", "Jon Friskics", "Ashley Smith"];
```

The brackets indicate to the compiler to make an array and fill it with the comma-separated values between the brackets.

If we wanted to access any particular index's value, we use:

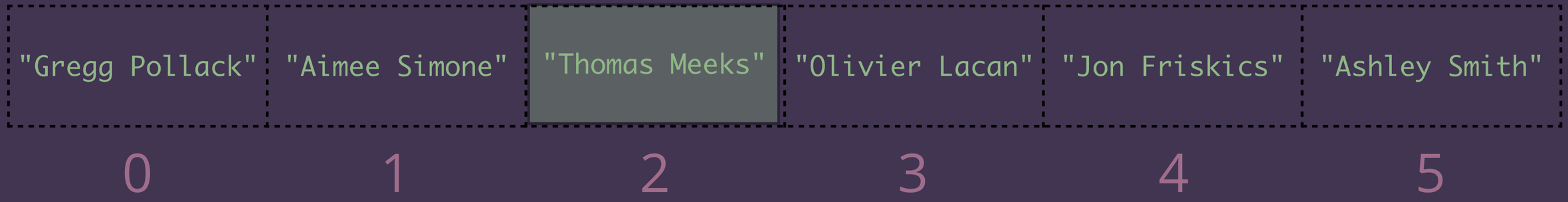
```
passengers[5];
```

→ `"Ashley Smith"`

Returns the value at index 5.

CHANGING ARRAY CONTENTS

We can also reference and change specific cells with indices



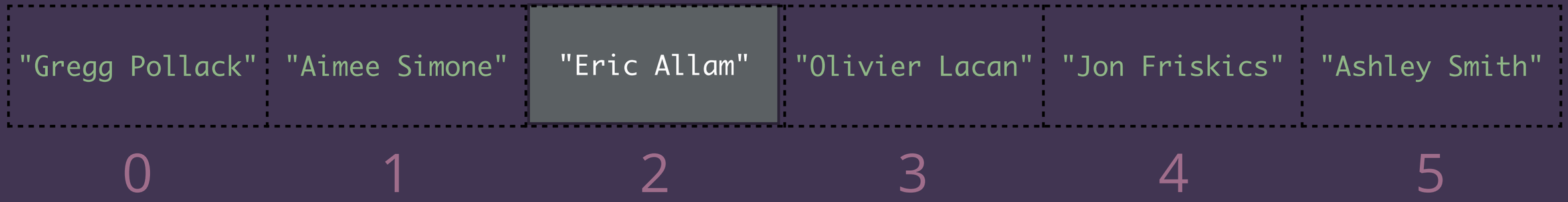
If we wanted to change the value contained at any index, we use:

```
passengers[2] = "Eric Allam";
```

This syntax says "Go over to index 2, and change its value to whatever comes after the = sign.

CHANGING ARRAY CONTENTS

We can also reference and change specific cells with indices



If we wanted to change the value contained at any index, we use:

```
passengers[2] = "Eric Allam";
```

This syntax says "Go over to index 2, and change its value to whatever comes after the = sign.

Like Strings, we can access the length of Arrays:

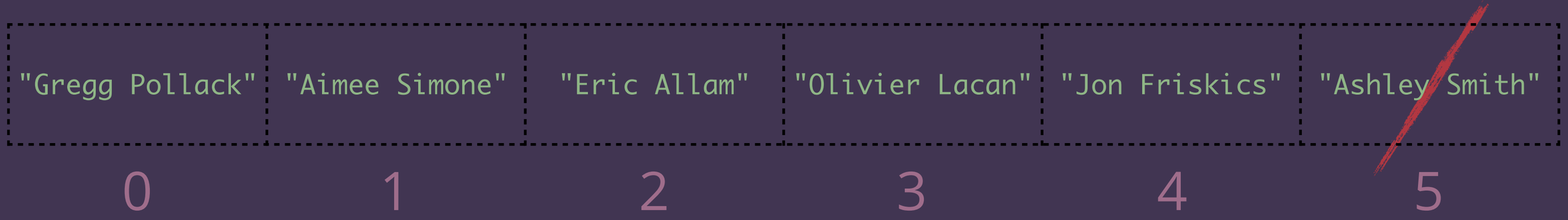
```
passengers.length;
```

→ 6

The length of an array is the actual number of cells, including any empty cells.

THE POP() FUNCTION

Removing a cell from the back of the array



The pop( ) function deletes the last position and retrieves its value:

```
passengers.pop();
```

→ `"Ashley Smith"`

pop() will automatically "pop" the last existing cell off the array while returning that cell's contents.

The array length automatically adjusts!

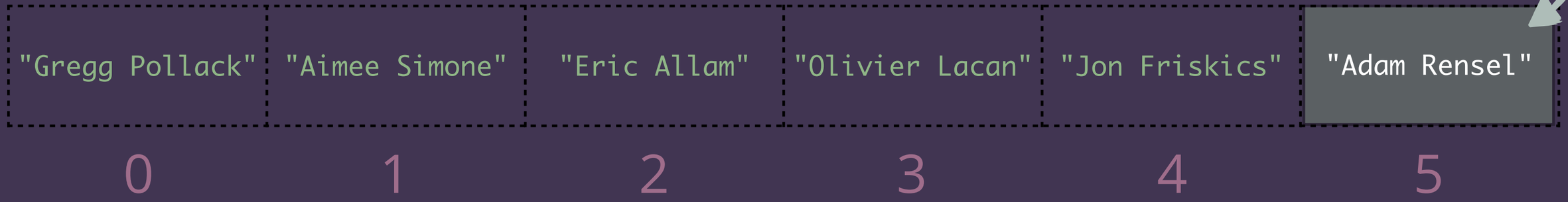
THE PUSH() FUNCTION

Adding a cell and its contents to the back of the array

The push( ) function adds a cell in the last position and enters a value:

```
passengers.push("Adam Rensel");
```

push() will "push" a cell onto the back of the array and automatically increase the array length.



ARRAYS CAN HOLD LOTS OF STUFF

Strings, values, variables, other arrays, and combinations of them all!

```
var comboArray1 = ["One", "fish", 2, "fish"];
```

"One"	"fish"	2	"fish"
0	1	2	3

```
var poisson = "fish";
```

The variable name disappears in the array and just the contents remain.

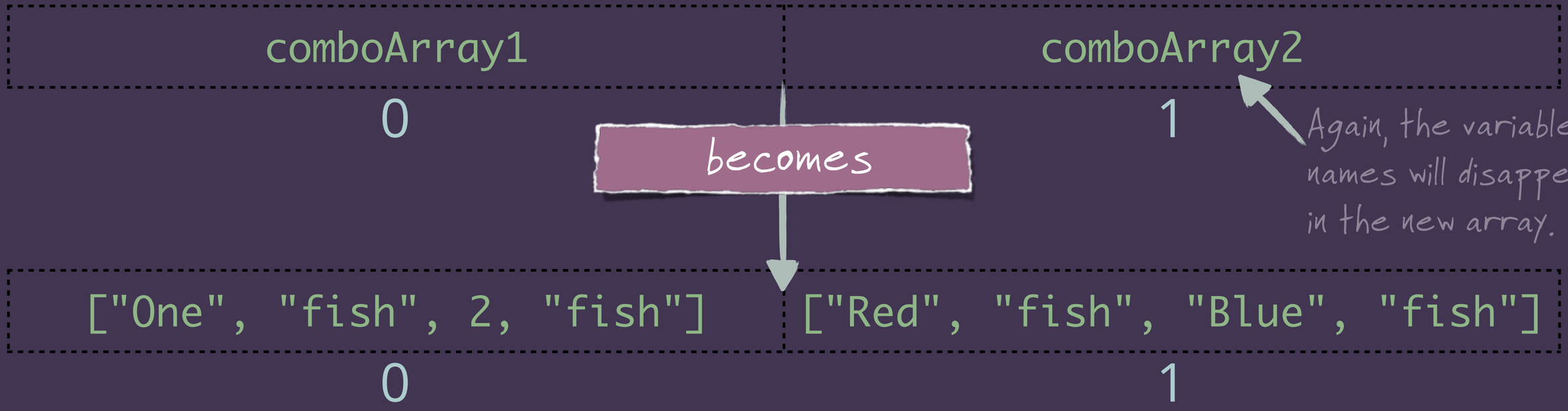
```
var comboArray2 = ["Red", poisson, "Blue", poisson];
```

"Red"	"fish"	"Blue"	"fish"
0	1	2	3

ARRAYS CAN HOLD LOTS OF STUFF

Strings, values, variables, other arrays, and combinations of them all!

```
var arrayOfArrays = [comboArray1, comboArray2];
```




```
console.log( arrayOfArrays );
```

→ `[ Array[4], Array[4] ]`

Here, the [4] and [4] are providing the lengths of each of the arrays, which here happen to be the same.

ARRAYS CAN HOLD LOTS OF STUFF

Strings, values, variables, other arrays, and combinations of them all!

```
var arrayOfArrays = [comboArray1, comboArray2];
```

```
["One", "fish", 2, "fish"]          ["Red", "fish", "Blue", "fish"]
```

0 1

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

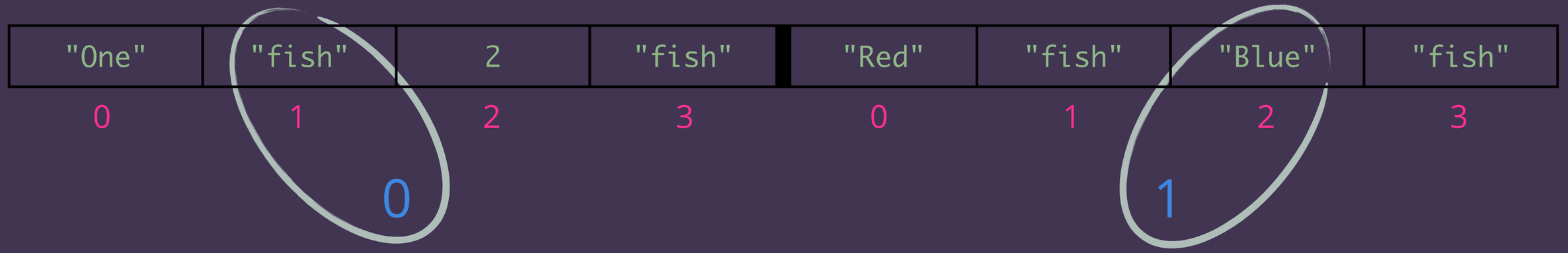
→ `["Red", "fish", "Blue", "fish"]`

When we reference the `[1]` index of `arrayOfArrays`, we get another entire array because that's what the cell contains. Specifically, our earlier `comboArray2`.

ARRAYS CAN HOLD LOTS OF STUFF

Strings, values, variables, other arrays, and combinations of them all!

```
var arrayOfArrays = [comboArray1, comboArray2];
```



The first bracket selects a cell in the master array.

The second bracket then selects a cell in the lower level array

```
console.log( arrayOfArrays[1][2] );
```

→ Blue

```
console.log( arrayOfArrays[0][1] );
```

→ fish

USING LOOPS WITH ARRAYS

Loops help us move through all indices of an array

```
var numberList = [ 2, 5, 8, 4, 7, 12, 6, 9, 3, 11 ];
```

```
for (var i = 0; i < numberList.length; i++){
}
```

You'll often see the variable `i` used as a loop counter by convention and for simplicity.

To look through our entire array, we continue only until we have reached the last index of the zero-based array. Since our array has a `length` of `10`, we want to stop checking at index `9`.

USING LOOPS WITH ARRAYS

Loops help us move through all indices of an array

```
var numberList = [ 2, 5, 8, 4, 7, 12, 6, 9, 3, 11 ];
```

```
for (var i = 0; i < numberList.length; i++){

  console.log("The value in cell " + i + " is " + numberList[i]);

}
```

Our loop counter can also serve as a current index position, helping us "iterate" over the entire contents of the array in order.

Don't confuse the index number (the *position*) with the contents of the cell (the *value*)!

USING LOOPS WITH ARRAYS

Loops help us move through all indices of an array

i	i < numberList.length ?	numberList[i]	printout
0	TRUE	2	The value in cell 0 is 2
1	TRUE	5	The value in cell 1 is 5
2	TRUE	8	The value in cell 2 is 8
3	TRUE	4	The value in cell 3 is 4
4	TRUE	7	The value in cell 4 is 7
5	TRUE	12	The value in cell 5 is 12
6	TRUE	6	The value in cell 6 is 6
7	TRUE	9	The value in cell 7 is 9
8	TRUE	3	The value in cell 8 is 3
9	TRUE	11	The value in cell 9 is 11
10	FALSE	NA	STOP!

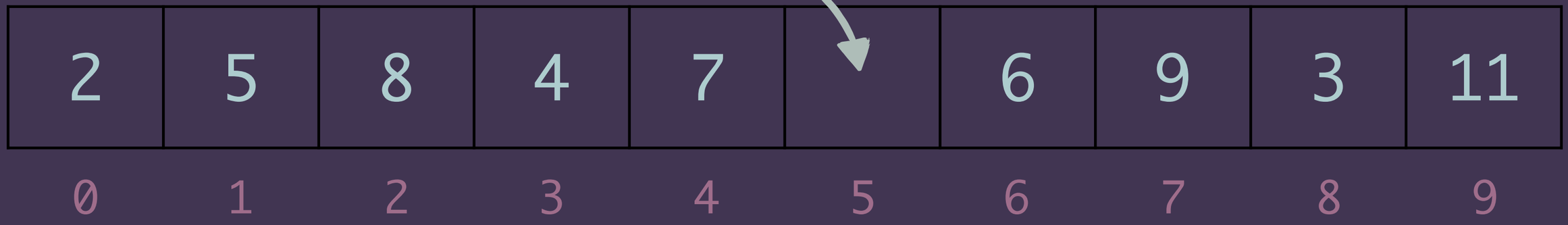
EMPTY CELLS IN ARRAYS?

Using the undefined value to create “empty” cells.

2	5	8	4	7	12	6	9	3	11
0	1	2	3	4	5	6	7	8	9

To make a cell empty, we’ll use the special `undefined` value, which means “no contents.”

```
passengers[5] = undefined;
```



A NEW FUNCTION WITH ARRAYS

Let's count even numbers AND erase odds.

```
var numberList = [ 2, 5, 8, 4, 7, 12, 6, 9, 3, 11 ];
```

```
var evenCount = 0;
for (var i = 0; i < numberList.length; i++) {
      if (numberList[i] % 2 == 0) {
            evenCount++;
      }


}
```

We'll set up a counter before the loop.

Even numbers will have a zero remainder when divided by 2!

A NEW FUNCTION WITH ARRAYS

Let's count even numbers AND erase odds.

```
var numberList = [ 2, 5, 8, 4, 7, 12, 6, 9, 3, 11 ];
```

```
var evenCount = 0;
for (var i = 0; i < numberList.length; i++) {
     if (numberList[i] % 2 == 0) {
          evenCount++;
     } else {
          numberList[i] = undefined;
     }
}
```

Otherwise, if not's even, we know it's odd! Here's where we will use undefined.

```
console.log(evenCount);
```

→ 5

USING LOOPS WITH ARRAYS

Loops help us move through all indices of an array

i	i < numberList.length ?	numberList[i]	numberList[i] % 2 == 0 ?	evenCount
0	TRUE	2	**TRUE**	**1**
1	TRUE	5	FALSE	1
2	TRUE	8	**TRUE**	**2**
3	TRUE	4	**TRUE**	**3**
4	TRUE	7	FALSE	3
5	TRUE	12	**TRUE**	**4**
6	TRUE	6	**TRUE**	**5**
7	TRUE	9	FALSE	5
8	TRUE	3	FALSE	5
9	TRUE	11	FALSE	5
10	FALSE	NA	STOP!	

USING LOOPS WITH ARRAYS

Loops help us move through all indices of an array

```
console.log(numberList);
```

→ `[2, undefined, 8, 4, undefined, 12, 6, undefined, undefined, undefined]`


```
console.log(numberList.length);
```

→ `10`

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( *passenger's name*, *array of passengers*) {
        *if list is empty* {
            *add passenger to list*
        } *else* {
            *for all spots in the list*{
                *if the current spot is empty* {
                    *add passenger to that spot*
                    *return the list and exit the function*
                } *else, if the end of the list is reached* {
                    *add passenger to end of list*
                    *return the list and exit the function*
                }
            }
        }
}
```

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
        if (list.length == 0) {
            *add passenger to list*
        } *else* {
            *for all spots in the list*{
                    *if the current spot is empty* {
                            *add passenger to that spot*
                            *return the list and exit the function*
                    } *else, if the end of the list is reached* {
                            *add passenger to end of list*
                            *return the list and exit the function*
                    }
            }
        }
}
```

A length of 0 means the array is empty.

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
          *for all spots in the list*{
              *if the current spot is empty* {
                  *add passenger to that spot*
                  *return the list and exit the function*
              } *else, if the end of the list is reached* {
                  *add passenger to end of list*
                  *return the list and exit the function*
              }
          }
      }
}
```

We start the list by pushing a passenger into the empty array.

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
          for (var i = 0; i < list.length; i++) {
                *if the current spot is empty* {
                      *add passenger to that spot*
                      *return the list and exit the function*
                } *else, if the end of the list is reached* {
                      *add passenger to end of list*
                      *return the list and exit the function*
                }
          }
      }
}
```

We want to check all spots in the list, which will include all indices through `list.length - 1`

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name,  list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
           for (var i = 0; i < list.length; i++) {
                if(list[i] == undefined){
                     list[i] = name;
                      *return the list and exit the function*
                } *else, if the end of the list is reached* {
                      *add passenger to end of list*
                      *return the list and exit the function*
                }
           }
      }
}
```

If a passenger spot has been emptied, it will be `undefined`. We want to fill that empty spot before adding more spots to the list.

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
           for (var i = 0; i < list.length; i++) {
                if(list[i] == undefined){
                     list[i] = name;
                     return list;
                } *else, if the end of the list is reached* {
                      *add passenger to end of list*
                      *return the list and exit the function*
                }
           }
      }
}
```

If we've placed the passenger name, then we're done! No need to keep looping. We can now `return` the updated list and exit the function.

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
          for (var i = 0; i < list.length; i++) {
              if(list[i] == undefined){
                  list[i] = name;
                  return list;
              } else if (i == list.length - 1) {
                  list.push(name);
                   *return the list and exit the function*
              }
          }
      }
}
```

If we have reached the final index of `list` without finding an empty spot, then push the name onto the end of `list`.

BUILDING A PASSENGER LIST

Using an array and functions to keep track of train passengers

```
function addPassenger ( name, list ) {
      if (list.length == 0) {
          list.push(name);
      } else {
          for (var i = 0; i < list.length; i++) {
              if(list[i] == undefined){
                  list[i] = name;
                  return list;
              } else if (i == list.length - 1) {
                  list.push(name);
                  return list;
              }
          }
      }
}
```

If the list was initially empty, we can return the updated `list` and exit.

CREATING A NEW PASSENGER LIST

Let's make a new list and add a few passengers to it.

```
var passengerList = [ ];
```

An empty set of brackets will create an array with no cells.

```
passengerList = addPassenger("Gregg Pollack", passengerList );
```

```
["Gregg Pollack"]
```

```
passengerList = addPassenger("Ashley Smith", passengerList );
```

```
["Gregg Pollack", "Ashley Smith"]
```

```
passengerList = addPassenger("Jon Friskics", passengerList );
```

```
["Gregg Pollack", "Ashley Smith", "Jon Friskics"]
```


REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        }









}
```

If the list is empty, log it to the user.

REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        } else  {
            for (var i = 0; i < list.length; i++) {









            }
        }
}
```

REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

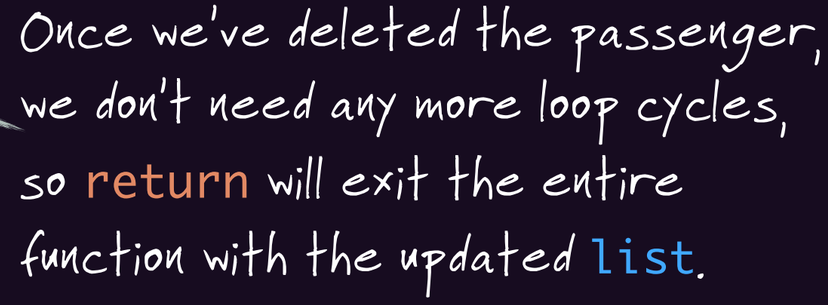
```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        } else  {
            for (var i = 0; i < list.length; i++) {
                if(list[i] == name){
                    list[i] = undefined;
                }
        }
}
```

If the contents of the index match the name exactly, delete it by setting the index to undefined.

REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        } else  {
            for (var i = 0; i < list.length; i++) {
                  if(list[i] == name){
                        list[i] = undefined;
                        return list;
            }
        }
}
```



REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        } else  {
            for (var i = 0; i < list.length; i++) {
                if(list[i] == name){
                    list[i] = undefined;
                    return list;
                } else if (i == list.length - 1) {
                    console.log("Passenger not found!");
                }
            }
        }
}
```

If we get to the end, and we haven't deleted a name, then we know the passenger wasn't present!

REMOVING PASSENGERS

Using an array and functions to keep track of train passengers

```
function deletePassenger ( name, list ) {
        if (list.length == 0){
            console.log("List is empty!");
        } else  {
            for (var i = 0; i < list.length; i++) {
                if(list[i] == name){
                    list[i] = undefined;
                    return list;
                } else if (i == list.length - 1) {
                    console.log("Passenger not found!");
                }
            }
        }
        return list;
}
```

If the list was empty, or if we never found the passenger, we just return the same list.

MODIFYING OUR PASSENGER LIST

Let's take some passengers out, and put some back in.

```
passengerList = ["Gregg Pollack", "Ashley Smith", "Jon Friskics"];
```

```
passengerList = deletePassenger( "Ashley Smith", passengerList );
```

```
["Gregg Pollack", undefined, "Jon Friskics" ]
```

```
passengerList = addPassenger( "Adam Rensel", passengerList );
```

```
["Gregg Pollack", "Adam Rensel", "Jon Friskics" ]
```

```
passengerList = deletePassenger( "Ashley Smith", passengerList );
```

```
→ Passenger not found!
```

MODIFYING OUR PASSENGER LIST

Let’s take some passengers out, and put some back in.

```
passengerList = ["Gregg Pollack", "Adam Rensel", "Jon Friskics"];
```

```
passengerList = deletePassenger( "Ashley Smith", passengerList );
```

```
→ Passenger not found!
```

MODIFYING OUR PASSENGER LIST

Let's take some passengers out, and put some back in.

```
passengerList = ["Gregg Pollack", "Adam Rensel", "Jon Friskics"];
```

```
passengerList = deletePassenger( "Ashley Smith", passengerList );
```

→ Passenger not found!

```
passengerList = deletePassenger("Gregg Pollack", passengerList );
```

[undefined, "Adam Rensel", "Jon Friskics"]

```
passengerList = addPassenger("Jennifer Borders", passengerList );
```

["Jennifer Borders", "Adam Rensel", "Jon Friskics"]

FIN?