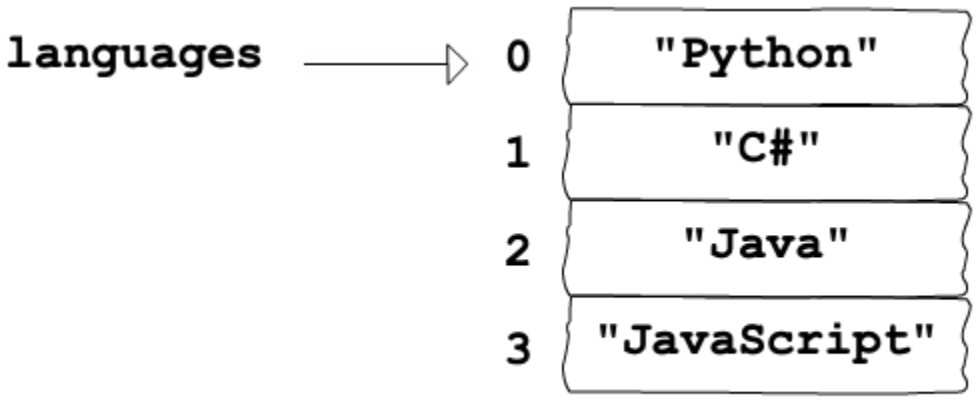
8.1. Arrays Are Like Strings

Arrays are similar to strings, but are a more general collection type. Like strings, **arrays** are a sequence of values that can be accessed via an ordered index. Unlike strings, arrays can store data of any type.

The figure below demonstrates an array named languages. The array contains four strings, each of those values has an index position.



8.1.1. Declaring an Array

Programmers use multiple ways to declare a new array. The simplest way is to use **array literal** notation `[]`. Anything enclosed in the square brackets will be *items* in the array. Each item should be followed by a comma `,`. If there are no items inside the brackets, then the array is considered empty.

```
let emptyArray = [];

let programmingLanguages = ["JavaScript", "Python", "Java", "C#"];
```

Array items can also be declared on multiple lines.

```
let javaScriptFrameworks = [
  "React",
  "Angular",
  "Ember",
  "Vue"
];
```

8.1.2. Array Length

To check the length of an array, use the `length` property, just like with strings. JavaScript array length is NOT fixed, meaning you can add or remove items dynamically.

Note

In other languages, such as Java and C#, arrays are of a static length requiring the length of the array to be declared upon creation.

Example

Print out the length of two arrays.

```
let emptyArray = [];
console.log(emptyArray.length);

let programmingLanguages = ["JavaScript", "Python", "Java", "C#"];
console.log(programmingLanguages.length);
```

Console Output

```
0
4
```

8.1.3. Varying Data Types

JavaScript arrays can hold a mixture of values of any type. For example, you can have an array that contains strings, numbers, and booleans.

```
let grabBag = ["A string value", true, 99, 105.5];
```

Note

It's rare that you would store data of multiple types in the same array, because grouped data is usually the same type. In other languages, such as Java and C#, all items in an array have to be of the same type.

8.1.4. Check Your Understanding

Question

What is the length of the two arrays?

Hint: look closely at the quotes in the classes array.

```
let classes = ["science, computer, art"];

let teachers = ["Jones", "Willoughby", "Rhodes"];
```

How can you change the `classes` array declaration to have the same number of items as the `teachers` array?

8.2. Working With Arrays

8.2.1. Bracket Notation and Index

As previously discussed, arrays are an ordered collection where each item can be accessed via index. Similar to strings, an **index** in an array is the number order given to items. Individual items can be accessed using bracket notation (`array[index]`). Indexes are zero-based, going from `0` to `array.length-1`.

Example

Use bracket notation and index to access items in an array.

```
let programmingLanguages = [
   "JavaScript", // index 0
   "Python",     // index 1
   "Java",       // index 2
   "C#"          // index 3
];
console.log(programmingLanguages[0]);
console.log(programmingLanguages[3]);

// What will happen when index 4 is requested?
console.log(programmingLanguages[4]);
```

Console Output

```
JavaScript
C#
undefined
```

Notice above that `undefined` was printed out when index 4 was referenced. `undefined` is returned when you request an index that the array does not contain.

Note

undefined is a special value in JavaScript that means no value has been assigned. We will discuss `undefined` more later in the class.

Example

`undefined` will be returned for any index that is outside of the array's index range.

```
let programmingLanguages = ["JavaScript", "Python", "Java", "C#"];
console.log(programmingLanguages[-1]);
console.log(programmingLanguages[100]);
```

Console Output

```
undefined
undefined
```

8.2.2. Arrays are Mutable

In programming, mutability refers to what happens when you attempt to change a value. Remember that strings are immutable, meaning that any change to a string results in a new string being created. In contrast, arrays are **mutable**, meaning that individual items in an array can be edited without a new array being created.

Example

Update an item in an array using bracket notation and index.

```
let javaScriptFrameworks = ["React", "Angular", "Ember"];
console.log(javaScriptFrameworks);

// Set the value of index 2 to be "Vue"
javaScriptFrameworks[2] = "Vue";

// Notice the value at index 2 is now "Vue"
console.log(javaScriptFrameworks);
```

Console Output

```
[ 'React', 'Angular', 'Ember' ]
[ 'React', 'Angular', 'Vue' ]
```

8.3. Array Methods

As with strings, JavaScript provides us with useful **methods** for arrays. These methods will either *alter* an existing array, *return* information about the array, or *create and return* a new array.

8.3.1. Common Array Methods

Here is a sample of the most frequently used array methods. More complete lists can be found here:

1. W3 Schools Array Methods
2. MDN Web Docs

To see detailed examples for a particular method, control-click (or right-click) on its name.

Methods That Return Information About The Array

Method	Syntax	Description
includes	`arrayName.includes(item)`	Checks if an array contains the specified item.
indexOf	`arrayName.indexOf(item)`	Returns the index of the FIRST occurence of an item in the array. If the item is not in the array, -1 is returned.

Methods That Rearrange The Entries In The Array

Method	Syntax	Description
reverse	`arrayName.reverse()`	Reverses the order of the elements in an array.
sort	`arrayName.sort()`	Arranges the elements of an array into increasing order (kinda).

Methods That Add Or Remove Entries From An Array

Method	Syntax	Description
pop	`arrayName.pop()`	Removes and returns the LAST element in an array.

Methods That Add Or Remove Entries From An Array

Method	Syntax	Description
push	`arrayName.push(item1, item2, ...)`	Adds one or more items to the END of an array and returns the new length.
shift	`arrayName.shift()`	Removes and returns the FIRST element in an array.
splice	`arrayName.splice(index, number, item1, item2, ...)`	Adds, removes or replaces one or more elements anywhere in the array.
unshift	`arrayName.unshift(item1, item2, ...)`	Adds one or more items to the START of an array and returns the new length.

Methods That Create New Arrays

Method	Syntax	Description
concat	`arr.concat(otherArray1, otherArray2, ...)`	Combines two or more arrays and returns the result as a new array.
join	`arr.join('connecter')`	Combines all the elements of an array into a string.
slice	`arr.slice(start index, end index)`	Copies selected entries of an array into a new array.
split	`stringName.split('delimiter')`	Divides a string into smaller pieces, which are stored in a new array.

8.3.2. Check Your Understanding

Follow the links in the table above for the `sort`, `slice`, `split` and `join` methods. Review the content and then answer the following questions.

Question

What is printed by the following code?

```
let charles = ['coder', 'Tech', 47, 23, 350];
charles.sort();
console.log(charles);

```

a. `[350, 23, 47, 'Tech', 'coder']`
b. `['coder', 'Tech', 23, 47, 350]`
c. `[23, 47, 350, 'coder', 'Tech']`
d. `[23, 350, 47, 'Tech', 'coder']`

Question

Which statement converts the string `str = 'LaunchCode students rock!'` into the array `['LaunchCode', 'students', 'rock!']`?

a. `str.join(" ");`
b. `str.split(" ");`
c. `str.join("");`
d. `str.split("");`

Question

What is printed by the following program?

```
let groceryBag = ['bananas', 'apples', 'edamame', 'chips', 'cucumbers', 'milk', 'cheese'
];
let selectedItems = [];

selectedItems = groceryBag.slice(2, 5).sort();
console.log(selectedItems);

```

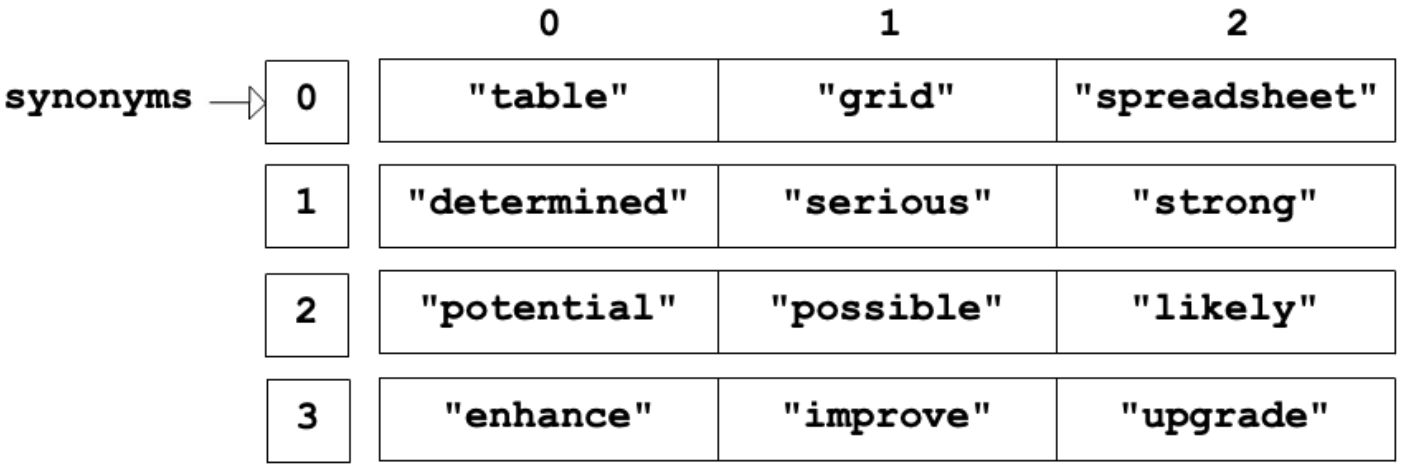
a. `['chips', 'cucumbers', 'edamame']`
b. `['chips', 'cucumbers', 'edamame', 'milk']`
c. `['cheese', 'chips', 'cucumbers']`
d. `['cheese', 'chips', 'cucumbers', 'edamame']`

8.4. Multi-Dimensional Arrays

Earlier we learned that arrays can store any type of value. If that is true, can we store arrays inside of arrays? Well yes we can….

A **multi-dimensional array** is an array of arrays, meaning that the values inside the array are also arrays. The *inner* arrays can store other values such as strings, numbers, or even more arrays.

The figure below demonstrates a synonyms array that has arrays as values. The *inner* arrays contain words that are synonyms of each other. Notice each inner array has an index position.



8.4.1. Two Dimensional Arrays

The simplest form of a multi-dimensional array is a two dimensional array. A two dimensional array is like a spreadsheet with rows and columns. To access items in a two dimensional array, use square bracket notation and two indexes `array[0][0]`. The first index is for the outer array, or the "row", and second index is for the inner array, or the "column".

Note

The row and column analogy is used to help visualize a two dimensional array, however it's not a perfect analogy. There are no specific JavaScript language rules forcing the inner arrays to all have the same length. The inner arrays are separate arrays that can be of different length.

Example

Use a two dimensional array to contain three different lists of space shuttle crews.

```
let shuttleCrews = [
  ['Robert Gibson', 'Mark Lee', 'Mae Jemison'],
  ['Kent Rominger', 'Ellen Ochoa', 'Bernard Harris'],
  ['Eilen Collins', 'Winston Scott',  'Catherin Coleman']
];

console.log(shuttleCrews[0][2]);
console.log(shuttleCrews[1][1]);
console.log(shuttleCrews[2][1]);
```

Console Output

```
Mae Jemison
Ellen Ochoa
Winston Scott
```

8.4.2. Multi-Dimensions and Array Methods

Both the inner and outer arrays in a multi-dimensional array are still altered with array methods.

Example

Use array methods to add an additional crew array and alter existing arrays.

```
let shuttleCrews = [
  ['Robert Gibson', 'Mark Lee', 'Mae Jemison'],
  ['Kent Rominger', 'Ellen Ochoa', 'Bernard Harris'],
  ['Eilen Collins', 'Winston Scott',  'Catherin Coleman']
];

let newCrew = ['Mark Polansky', 'Robert Curbeam', 'Joan Higginbotham'];

// Add a new crew array to the end of shuttleCrews
shuttleCrews.push(newCrew);
console.log(shuttleCrews[3][2]);

// Reverse the order of the crew at index 1
shuttleCrews[1].reverse();
console.log(shuttleCrews[1]);

```

Console Output

```
Joan Higginbotham
[ 'Bernard Harris', 'Ellen Ochoa', 'Kent Rominger' ]
```

8.4.3. Beyond Two Dimensional Arrays

Generally there is no limit to how many dimensions you can have when creating arrays. However it is rare that you will use more than two dimensions. Later on in the class we will learn about more collection types that can handle complex problems beyond the scope of two dimensional arrays.

8.4.4. Check Your Understanding

Question

What are the two dimensional indexes for `"Jones"`?

```
let school = [
  ["science", "computer", "art"],
  ["Jones", "Willoughby", "Rhodes"]
];

```

How would you add `"dance"` to the array at `school[0]`?

How would you add `"Holmes"` to the array at `school[1]`?

8.5. Exercises: Arrays

OK, rookie. It's time to train you on how to modify the shuttle's cargo manifest. The following actions will teach you how to add, remove, modify and rearrange our records for the items stored in our hold.

1. Create an array called `practiceFile` with the following entry: 273.15. Use the `push` method to add the following elements to the array. Add items a & b one at a time, then use a single `push` to add the items in part c. Print the array after each step to confirm the changes.

    a. 42
    b. "hello"
    c. `false`, -4.6, "87"

    Code it at repl.it

    Congratulations, rookie. You can now add items to an array.

2. `push`, `pop`, `shift` and `unshift` are used to add/remove elements from the beginning/end of an array. **Bracket notation** can be used to modify any element within an array. Starting with the `cargoHold` array `['oxygen tanks', 'space suits', 'parrot', 'instruction manual', 'meal packs', 'slinky', 'security blanket']`, write statements to do the following:

    a. Use bracket notation to replace `'slinky'` in the array with `'space tether'`. Print the array to confirm the change.
    b. Remove the last item from the array with `pop`. Print the element removed and the updated array.
    c. Remove the first item from the array with `shift`. Print the element removed and the updated array.
    d. Unlike `pop` and `shift`, `push` and `unshift` require arguments inside the `()`. Add the items 1138 and '20 meters' to the the array - the number at the start and the string at the end. Print the updated array to confirm the changes.
    e. Use a template literal to print the final array and its length.

    Code it at repl.it

    Status check, rookie. Which array methods ADD items, and where are the new entries placed? Which methods REMOVE items, and where do the entries come from? Which methods require entries inside the ``()``?

3. The `splice` method can be used to either add or remove items from an array. It can also accomplish both tasks at the same time. Review the splice appendix if you need a syntax reminder. Use `splice` to make the following changes to the final `cargoHold` array from exercise 2. Be sure to print the array after each step to confirm your updates.

    a. Insert the string `'keys'` at index 3 without replacing any other entries.
    b. Remove 'instruction manual' from the array. (Hint: `indexOf` is helpful to avoid manually counting an index).
    c. Replace the elements at indexes 2 - 4 with the items `'cat'`, `'fob'`, and `'string cheese'`.

    Code it at repl.it

    Well done, cadet. Now let's look at some finer details about array methods. We've got to keep our paperwork straight, so you need to know when your actions change the original records.

4. Some methods—like `splice` and `push`—alter the original array, while others do not. Use the arrays

```
5. holdCabinet1 ['duct tape', 'gum', 3.14, false, 6.022e23]
```

and

```
holdCabinet2 ['orange drink', 'nerf toys', 'camera', 42, 'parsnip']
```

to explore the following methods: `concat`, `slice`, `reverse`, `sort`. Refer back to the chapter if you need to review the proper syntax for any of these methods.

   a. Print the result of using `concat` on the two arrays. Does `concat` alter the original arrays? Verify this by printing `holdCabinet1` after using the method.
   b. Print a `slice` of two elements from each array. Does `slice` alter the original arrays?
   c. `reverse` the first array, and `sort` the second. What is the difference between these two methods? Do the methods alter the original arrays?

Code it at repl.it

Good progress, cadet. Here are two more methods for you to examine.

6. The `split` method converts a string into an array, while the `join` method does the opposite.

   a. Try it! Given the string `str = 'In space, no one can hear you code.'`, see what happens when you print `str.split()` vs. `str.split('e')` vs. `str.split(' ')` vs. `str.split('')`. What is the purpose of the parameter inside the `()`?
   b. Given the array `arr = ['B', 'n', 'n', 5]`, see what happens when you print `arr.join()` vs. `arr.join('a')` vs. `arr.join(' ')` vs. `arr.join('')`. What is the purpose of the parameter inside the `()`?
   c. Do `split` or `join` change the original string/array?
   d. The benefit, cadet, is that we can take a string with **delimiters** (like commas) and convert it into a modifiable array. *Try it!* Alphabetize these hold contents: "water,space suits,food,plasma sword,batteries", and then combine them into a new string.

Code it at repl.it

Nicely done, astronaut. Now it's time to bring you fully up to speed.

7. Arrays can hold different data types, even other arrays! A **multi-dimensional array** is one with entries that are themselves arrays.

   a. Define and initialize the following arrays, which hold the name, chemical symbol and mass for different elements:
      i. `element1 = ['hydrogen', 'H', 1.008]`
      ii. `element2 = ['helium', 'He', 4.003]`
      iii. `element26 = ['iron', 'Fe', 55.85]`
   b. Define the array `table`, and use `push(arrayName)` to add each of the element arrays to it. Print `table` to see its structure.
   c. Use bracket notation to examine the difference between printing `table[1]` and `table[1][1]`. Don't just nod your head! I want to HEAR you describe this difference. Go ahead, talk to your screen.
   d. Using bracket notation and the `table` array, print the mass of element1, the name for element 2 and the symbol for element26.

e. `table` is an example of a *2-dimensional array*. The first "level" contains the element arrays, and the second level holds the name/symbol/mass values. **Experiment!** Create a 3-dimensional array and print out one entry from each level in the array.

Code it at repl.it

Excellent work, records keeper. Welcome aboard.

8.6. Studio: Strings and Arrays

Strings are **ordered collections** of *characters*, which are strings of length 1. The characters in a string can be accessed using **bracket notation**.

Arrays are ordered collections of items, which can be strings, numbers, other arrays, etc. The items/elements/entries stored in an array can be accessed using bracket notation.

Strings are **immutable**, whereas arrays can be changed.

Strings and arrays have **properties** and **methods** that allow us to easily perform some useful actions.

8.6.1. Before You Start

If you are enrolled in a LaunchCode program, access this studio by following the repl.it classroom links posted in your class at learn.launchcode.org.

If you are working through this material on your own, use the repl.it links contained on this page.

8.6.2. String Modification

Use string methods to convert a word into pseudo-pig latin.

a. Remove the first three characters from a string and add them to the end.
   Ex: `'LaunchCode'` becomes `'nchCodeLau'`. Use a template literal to print the original and modified string in a descriptive phrase.
b. Modify your code to accept user input. Query the user to enter the number of letters that will be relocated.
c. Add validation to your code to deal with user inputs that are longer than the word. In such cases, default to moving 3 characters. Also, the template literal should note the error.

Code it at repl.it

8.6.3. Array and String Conversion

The `split` and `join` methods convert back and forth between strings and arrays. Use **delimiters** as reference points to split a string into an array, then modify the array and convert it back to a printable string.

a. For a given string, use the `includes` method to check to see if the words are separated by commas (`,`), semicolons (`;`) or just spaces.
b. If the string uses commas to separate the words, `split` it into an array, reverse the entries, and then `join` the array into a new comma separated string.
c. If the string uses semicolons to separate the words, `split` it into an array, alphabetize the entries, and then `join` the array into a new comma separated string.
d. If the string uses spaces to separate the words, `split` it into an array, reverse alphabetize the entries, and then `join` the array into a new space separated string.
e. *Consider*: What if the string uses 'comma spaces' (,) to separate the list? Modify your code to produce the same result as part "b", making sure that the extra spaces are NOT part of the final string.

Code it at repl.it

8.6.4. Bonus Mission: Multi-dimensional Arrays

Arrays can store other arrays!

a. The cargo hold in our shuttle contains several smaller storage spaces. Use `split` to convert the following strings into four cabinet arrays. Alphabetize the contents of each cabinet.
    i. “water bottles, meal packs, snacks, chocolate”
    ii. “space suits, jet packs, tool belts, thermal detonators”
    iii. “parrots, cats, moose, alien eggs”
    iv. “blankets, pillows, eyepatches, alarm clocks”
b. Initialize a `cargoHold` array and add the cabinet arrays to it. Print `cargoHold` to verify its structure.
c. Query the user to select a cabinet (0-3) in the `cargoHold`.
d. Use bracket notation and a template literal to display the contents of the selected cabinet. If the user entered an invalid number, print an error message instead.
e. *Bonus to the Bonus*: Modify the code to query the user for BOTH a cabinet in `cargoHold` AND a particular item. Use the `includes` method to check if the cabinet contains the selected item, then print “Cabinet ____ DOES/DOES NOT contain ____.”

Code it at repl.it