codecademy

Arrays

You can access characters of Strings like array indexing!
And I don't think JS has characters. str[1] will return a string containing one character

Property `.length`

The `.length` property of a JavaScript array indicates the number of elements the array contains.

Notice that accessing a property you don't put () after the keyword, unlike functions!
This is the same with String.length

```
const numbers = [1, 2, 3, 4];

numbers.length // 4
```

Index

Array elements are arranged by *index* values, starting at `0` as the first element index. Elements can be accessed by their index using the array name, and the index surrounded by square brackets.

```
// Accessing an array element
const myArray = [100, 200, 300];

console.log(myArray[0]); // 100
console.log(myArray[1]); // 200
console.log(myArray[2]); // 300
```

Method `.push()`

The `.push()` method of JavaScript arrays can be used to add one or more elements to the end of an array. `.push()` mutates the original array returns the new length of the array.

```
// Adding a single element:
const cart = ['apple', 'orange'];
cart.push('pear');

// Adding multiple elements:
const numbers = [1, 2];
numbers.push(3, 4, 5);
```

Method `.pop()`

The `.pop()` method removes the last element from an array and returns that element.

```
const ingredients = ['eggs', 'flour',
'chocolate'];

const poppedIngredient =
ingredients.pop(); // 'chocolate'
console.log(ingredients); // ['eggs',
'flour']
```

Mutable

JavaScript arrays are *mutable*, meaning that the values they contain can be changed.

Even if they are declared using `const`, the contents can be manipulated by reassigning internal values or using methods like `.push()` and `.pop()`.

```
const names = ['Alice', 'Bob'];

names.push('Carl');
// ['Alice', 'Bob', 'Carl']
```

Arrays

codecademy

Arrays are lists of ordered, stored data. They can hold items that are of any data type. Arrays are created by using square brackets, with individual elements separated by commas.

```
// An array containing numbers
const numberArray = [0, 1, 2, 3];

// An array containing different data types
const mixedArray = [1, 'chicken', false];
```

same with val in Scala!

Function that mutates arrays can use the array as the argument to manipulate it.
These functions uses "reference copying" when doing this, same in Scala

Create nested arrays with
> const nested = [[1,2],[3]]
Then access them using nested notation
> console.log(nested[0][1]) //2

You can access arrays at non-contiguous locations!!

There's nothing special about accessing arrays with []
JS simply forbids accessing properties with name starting with a number using .
So if you have another object with a property name "3d", you need to access it via object["3d"] not object.3d
Yes, properties are in String type. JS implicitly converts all your indices into String before accessing

splice(i,0,elem) inserts elem at position i
push(elems), pop(), shift(), unshift(elems), length, splice(pos, n, elem), slice()/(i)/(i, j), filter(predicate), findIndex(predicate), map(), forEach(item,index,array), join(optional: separation string (default ", ")), reduce(optional:first val)
length

Length in JS arrays are very flexible.
The length property of an object which is an instance of type Array sets or returns the number of elements in that array. So by adding elements to the end (contiguous or not), you increase the length of the array.
You can arbitrarily increase or decrease length using arr.length = ...
Increasing length gives undefined empty spaces at the end. Decreasing length shortens the array.

You can create Arrays with specified initial size in 2 ways.
1:
> const arr = new Array(10) //using the array constructor
2:
> const arr = []
> arr.length = 10

Initializing a nest array:
> const a = new Array(10)
> for (let i = a.length-1; i >=0; i--) a[i] = new Array(10)

It's difficult to do this with .forEach as forEach skips all undefined entries

reduce() with lack of elements
If the array only has one element (regardless of position) and no initialValue is provided, or if initialValue is provided but the array is empty, the solo value will be returned without calling callback.