Week 4 - Strings 1 (of 2)

Overview
This week, you will first briefly explore associative arrays, then continue onward into *strings* -- another special case of the basic indexed array.

Prerequisites
By now you should be very comfortable completing all 13 mandatory algorithm challenges from pre-boot-camp in less than two minutes each. Also, make sure to clearly understand the previous material.

Interview Best Practices
Remember best practices mentioned previously. Ask clarifying questions before rushing to write code. Think about special-case situations (corner cases) and note these. Verify understanding by restating problems and intended outputs for simple input. *Then* start coding.

Extra Goodies
At page bottom are examples of Javascript constructs we'll use this week. Remember these building blocks!

T-Diagrams
Being able to write a t-diagram to track your variables as you write an algorithm is extremely beneficial. Use a t-diagram for every algorithm challenge this week.

Concepts
Associative arrays (known as dictionaries, maps, or key-value pairs) are equivalent to objects in JavaScript. You can think of them as effectively arrays that can accept strings as indices. Empty associative arrays are initialized as { } instead of [ ], and they can be set or read by either object attribute syntax (array.attrib) or array index syntax (array['index']).

Strings are arrays of characters (or more accurately, arrays of one-character *strings*). Once a string is defined, individual characters can be referenced by [ ] but cannot be changed. Strings are considered *immutable*: they can be completely replaced in their entirety, but not changed piecewise. If you need to manipulate string characters, you must split the string to an *array*, make individual changes, then join it.

Associative arrays / objects / maps / dictionaries

```
var myAssocArr = {};
myAssocArr.IQ = 116;
myAssocArr['fun'] = "Martin honks on a tenor saxophone";
console.log(myAssocArr);              // { IQ: 116; fun: "Martin honks on a tenor saxophone" }
```

Strings

```
console.log(typeof myAssocArr.fun);        // 'string'
var myChar = myAssocArr.fun[26];           // 'x'
console.log(typeof myChar);                // 'string'
```

.length method

```
console.log(myAssocArr.fun.length);        // 33
console.log("".length);                    // 0
```

.split method

```
myArray = myAssocArr['fun'].split(" ");    // ["Martin","honks","on","a","tenor","saxophone"]
console.log(myArray[4].split("");          // ["s","a","x","o","p","h","o","n","e"];
```

.join method

```
console.log(myArray.join());               // "Martin, honks, on, a, tenor, saxophone"
console.log(myArray.join("-"));            // "Martin-honks-on-a-tenor-saxophone"
```

Challange: what is displayed by the following? Why?

```
console.log(1 + 2 + "3" + "4" + 5 + 6);
```

Tomorrow: TLAs

Week 4 - Monday - Strings 1

This week, you will first explore associative arrays, then continue into *strings*. Some or all of these methods will be used to solve this week's challenges.

.length *.split* *.join* *.concat* *for...in loops*

Arrs2Map

Given two arrays, create an associative array (map) containing keys of the first, and values of the second. For arr1 = ['abc', 3, 'yo'] and arr2 = [42, 'wassup', true], you should return {'abc': 42, 3: 'wassup', 'yo': true}.

Answer:

ReverseString

Implement a function reverseString(str) that, given a string, will return the string of the same length but with characters reversed. Example: given 'creature', return 'erutaerc'. Do not use the built-in reverse() function!

Answer:

InvertHash

Create invertHash(assocArr) that converts a hash's keys to values and values to corresponding keys. Example: given {'name': 'Zaphod'; 'numHeads': 2}, you should return {'Zaphod': 'name'; 2: 'numHeads'}. You will need to learn and use a JavaScript ***for … in*** here!

Answer:

Challenge solved: '`console.log(1 + 2 + "3" + "4" + 5 + 6)`' displays the string "**33456**". num+num=num, but num+string=string, and string+num=string. 1+2=3, but 3+"3" = "33", and later "334"+5 = "3345".