CS105 Problem Solving

Lists (aka Arrays)

Wholeness

- Sometimes you just wish you could easily work on lots of data without needing lots of variables, or perhaps that your program could create extra variables as it needed them while running.
- Lists let us do those things and more.

- The field of all possibilities is the source of all solutions.

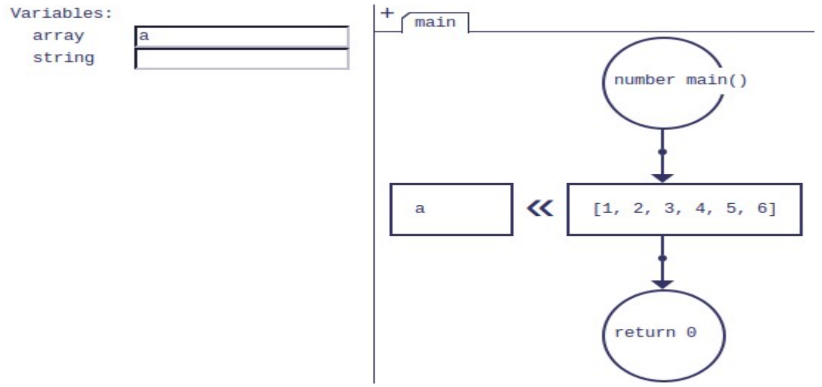
Lists

- A list is a collection of variables
  - We’ve already seen them with Strings (list of chars)
  - If a seat is a variable, a row of seats would be a list

- You can store things at specific positions
- You can retrieve things from positions
  - Assuming something was stored there

Creating a List

- You can create a list in two ways
  - [1, 2, 3, 4, 5, 6]
  - new Array(1, 2, 3, 4, 5, 6)



- You can give zero or more items to start
  - An empty array is no problem (add more later)
  - There is no limit to how many you can start with

Data Types

- The array itself has the array datatype
  - The elements inside it can have any datatype

- You can put a mixture of numbers, strings, booleans, arrays and objects into an array
  - But you shouldn't
  - Considered bad practice / hard to reason about

List vs Array

- In most languages they have slightly different meanings
  - Not so in JavaScript

- Other languages:
  - Arrays have fixed size (fixed amount of space)
  - Lists can dynamically grow and shrink

Example

- Lets create a list that contains the first 5 letters of the alphabet

Exercise

- Create an array that holds all the single digit numbers
- We will expand on it coming up

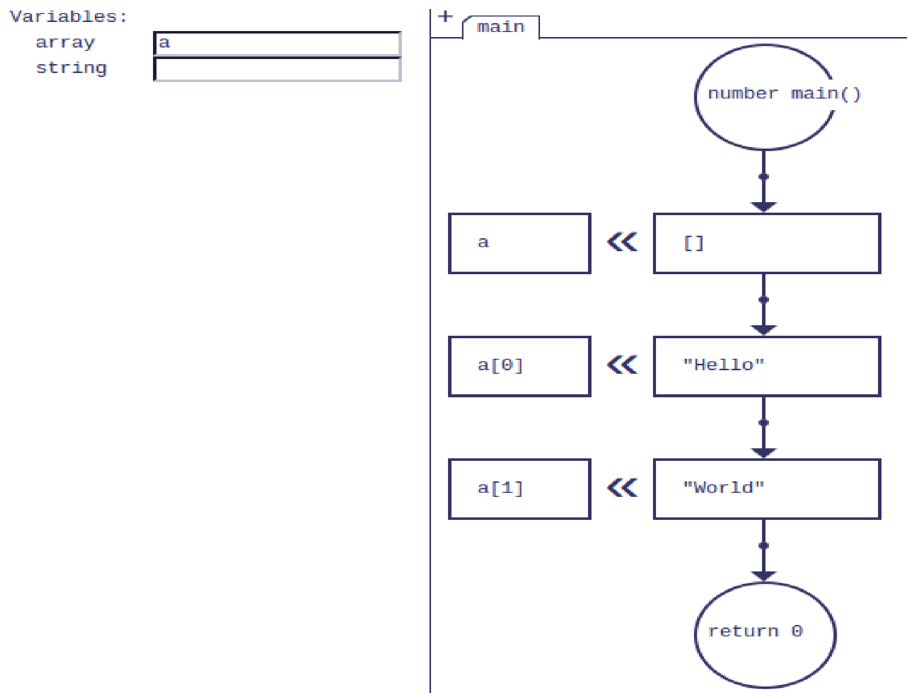
Main Point 1

- A list is simple data structure that can hold multiple values inside of it.
- You can create an array with either the “new Array()” constructor, or the [ ] array literal
  - Both of these can also specify values at creation
- The whole is greater than the sum of the parts

Setting Values

- You can set a position in an array to a certain value by editing the assignment block and specifying the location with square brackets

```
var a; // array
a = [];
a[0] = “Hello”
a[1] = “World”
```

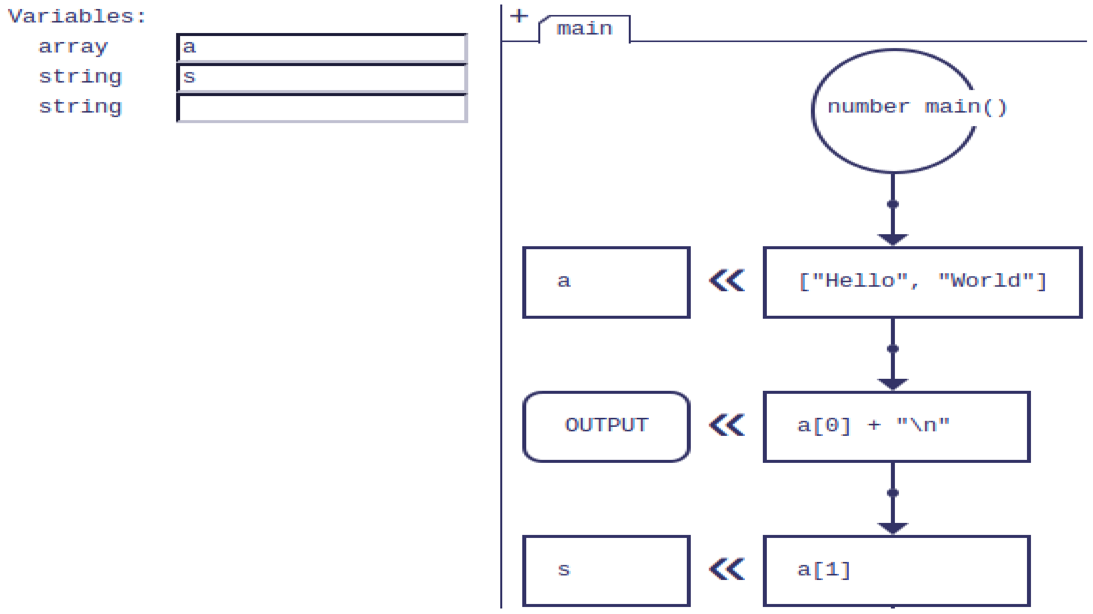


```
Variables:
  array   a
  string
```

Getting

- You can get something out of a position in an array by using the square brackets in an expression

```
var a; // array
var s; // string
a = [“Hello”, “World”]
OUTPUT << a[0] + “\n”
s = a[1]
```



Example

- Lets make an array
  - Put in the numbers 10, 11, 12, 13, 14, 15
  - Output the numbers at index 1, 2, 3

Exercise

- Write a program that creates a list
    - Adds 5 random numbers between 0 and 100
    - Outputs the numbers at index 0, 2, 4

Main Point 2

- You can access a position in an array with the square bracket notation.
  - Either to set (overwrite) something there
  - Or get (retrieve) what is there
- Arrays give us the flexibility to programmatically create extra variables
- One array, many variables inside it – unity in diversity

Loops and Arrays

- Loops are often used in combination with Arrays
  - To look at each element in an array
  - Regardless of the length of the array

- These are always for loops
  - Loops with a counter
  - The counter can be used as the index into the array!

Example

- Lets make an array with 50 random numbers
- Lets sum all the odd numbers in the array

Exercise

- Use a loop to add all the numbers from 1 through 100 to an array
- Then use a loop to go through the array
  - If a number is a multiple of 3 print “Fizz”

Main Point 3

- An array is one thing that can hold many things, while a loop holds one set of instructions that can be applied to many times (to many things)
- Both provide a unity within diversity, showing us that life is found in layers

Array API

- .concat()
- .indexOf()
- .join()
- .lastIndexOf()
- .length ← not method
- .pop()
- .push()
- .reverse()
- .shift()
- .slice()
- .sort()
- .splice()
- .toString()
- .unshift()
- Other more advanced methods also exist

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array

Adding to the end?

- A function could receive an array as input, and then may want to add new values to the end of the list – but what location (index) is the end?

```
a.push(5)
a[a.length] = 5
```

Going through in reverse?

- What if you want to output the numbers in an array in reverse?

- Either:
  - Use a loop with an index that counts down
  - Or user a.reverse() and then count upwards

Main Point 4

- The array object has a variety of method (functions on the array object) that help us to common tasks.
- Do less and accomplish more.

Summary

- A list holds multiple sequential values inside it
- You can access values (get / set) with square brackets
- Loops are usually used with lists (to go through them)
- The Array object API provides useful methods