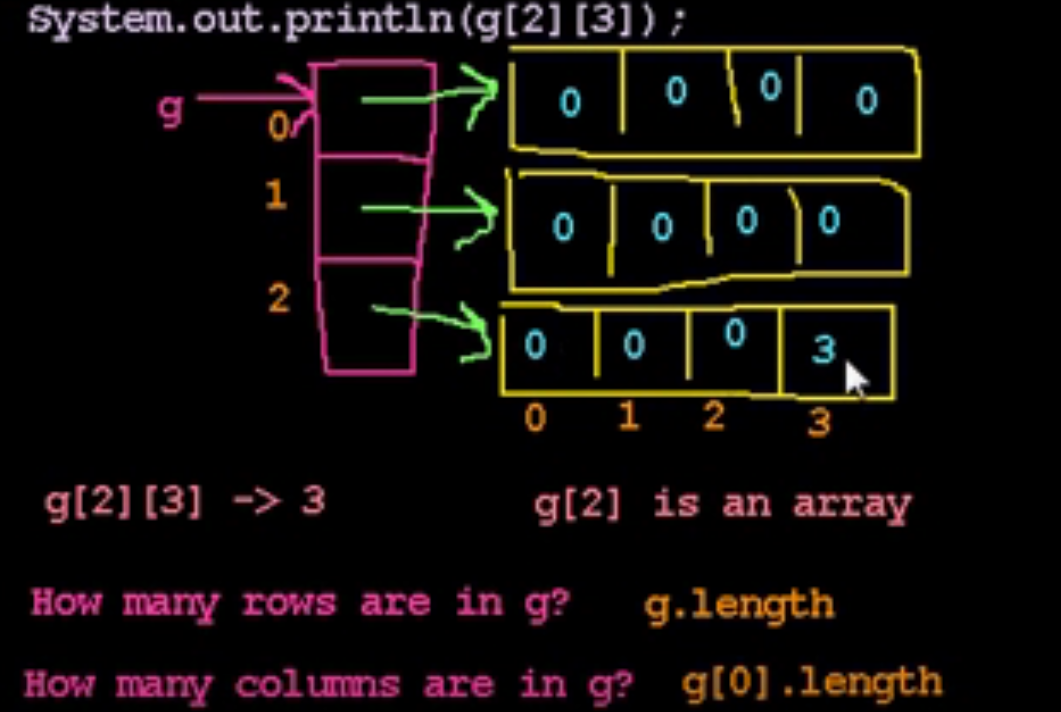
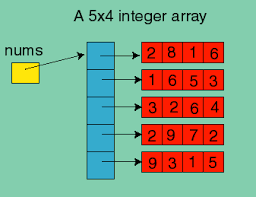
2-D Arrays



2-Dimensional array is really an array of arrays.



****

****

* Array is a linear collection of elements, where the elements can be accessed via indices

Creating Arrays

var numbers = [];

* Determine the length of array as like numbers.length
* To create an array with array constructor and also identify the length of the array

var numbers = new Array(1,2,3,4,5);

print(numbers.length); // displays 5

* We can create an array by calling the Array constructor with a single argument specifying the length of the array:

var numbers = new Array(10);

print(numbers.length); // displays 10

* JavaScript array can contains different types of data elements.

var objects = [1, "Joe", true, null];

* Identify an object is array or not

var numbers = 3;

var arr = [7,4,1776];

print(Array.isArray(number)); // displays false

print(Array.isArray(arr)); // displays true

Creating Arrays from Strings

* Arrays can be created as the result of calling the split() function on a string. This function breaks up a string at a common delimiter, such as a space for each word, and creates an array consisting of the individual parts of the string.

Aggregate Array Operations

* Assign one array to another array
* When you assign one array to another array, you are assigning a reference to the assigned array. When you make a change to the original array, that change is reflected in the other array as well.
* This is called a shallow copy.
* A better alternative is to make a deep copy, so that each of the original array’s elements is actually copied to the new array’s elements. An effective way to do this is to create a function to perform the task.

Accessor Functions

* The indexOf() method returns the position of the first occurrence of a specified value in a string.

Searching for a Value

* One of the most commonly used accessor functions is indexOf(), which looks to see if the argument passed to the function is found in the array. If the argument is contained in the array, the function returns the index position of the argument. If the argument is not found in the array, the function returns -1.
* lastIndexOf(), will return the position of the last occurrence of the argument in the array, or -1 if the argument isn’t found.

Creating New Arrays from Existing Arrays

* There are two accessor functions that allow you create new arrays from existing arrays: concat() and splice().
* The concat() function allows you to put together two or more arrays to create a new array, and the splice() function allows you to create a new array from a subset of an existing array.
* There are other uses for splice() as well, such as modifying an array by adding and/or removing elements.

Mutator Functions

* JavaScript has a set of mutator functions that allow you to modify the contents of an array without referencing the individual elements.

Adding Elements to an Array

* There are two mutator functions for adding elements to an array: push() and unshift().
* The push() function adds an element to the end of an array.
* The mutator function for adding array elements to the beginning of an array is unshift().

Removing Elements from an Array

* Removing an element from the end of an array is easy using the pop() mutator function

Adding and Removing Elements from the Middle of an Array

* Trying to add or remove elements at the end of an array leads to the same problems we find when trying to add or remove elements from the beginning of an array—both operations require shifting array elements either toward the beginning or toward the end of the array. However, there is one mutator function we can use to perform both operations—splice().

To add elements to an array using splice(), you have to provide the following arguments:

* The starting index (where you want to begin adding elements)
* The number of elements to remove (0 when you are adding elements)
* The elements you want to add to the array

Putting Array Elements in Order

* reverse(), reverses the order of the elements of an array.
* By default, the sort() method sorts the values as strings in alphabetical and ascending order.
* compareFunction Optional. A function that defines an alternative sort order. The function should return a negative, zero, or positive value, depending on the arguments, like:
* function(a, b){return a-b}
* When the sort() method compares two values, it sends the values to the compare function, and sorts the values according to the returned (negative, zero, positive) value.
* Example:
* When comparing 40 and 100, the sort() method calls the compare function(40,100).
* The function calculates 40-100, and returns -60 (a negative value).
* The sort function will sort 40 as a value lower than 100.
* For numbers, the ordering function can simply subtract one number from another number. If the number returned is negative, the left operand is less than the right operand; if the number returned is zero, the left operand is equal to the right operand; and if the number returned is positive, the left operand is greater than the right operand.

Iterator Functions

* The final set of array functions we examine are iterator functions. These functions apply a function to each element of an array, either returning a value, a set of values, or a new array after applying the function to each element of an array.
* forEach() function takes a function as an argument and applies the called function to each element of an array.
* every(), applies a Boolean function to an array and returns true if the function can return true for every element in the array.
* The some() function will take a Boolean function and return true if at least one of the elements in the array meets the criterion of the Boolean function.
* The reduce() function applies a function to an accumulator and the successive elements of an array until the end of the array is reached, yielding a single value.