Splice- var itDiv = ["Mike","Clayton","Terrill","Raymond","Cynthia","Danny","Jennifer"];

var dmpDept = itDiv.splice(3,3);

var cisDept = itDiv;

print(dmpDept); // Raymond,Cynthia,Danny

print(cisDept);

var itDiv = ["Mike","Clayton","Terrill","Raymond","Cynthia","Danny","Jennifer"];

var dmpDept = itDiv.splice(3,3);

var cisDept = itDiv;

undefined

console.log (dmpDept);

(3) ['Raymond', 'Cynthia', 'Danny']

undefined

console.log (cisDept);

 ['Mike', 'Clayton', 'Terrill', 'Jennifer']

undefined

it starts on 3 and delets the next 3 and puts it in a new array

POP- Removes the last entry of an array

Shift- removes the first element of an array

Sort sorts everything out -We often need to sort the elements of an array into order. The mutator function for this

task, sort(), works very well with strings:

var names = ["David","Mike","Cynthia","Clayton","Bryan","Raymond"];

nums.sort();

print(nums); // Bryan,Clayton,Cynthia,David,Mike,Raymond

But sort() does not work so well with numbers:

var nums = [3,1,2,100,4,200];

nums.sort();

if you want to use numbers you must use compare- function compare(num1, num2) {

return num1 - num2;

}

var nums = [3,1,2,100,4,200];

nums.sort(compare);

print(nums); // 1,2,3,4,100,200

print(nums); // 1,100,2,200,3,4

in order to put in random numbers on a array you need to greate a new function wuth works with the previose one

Array.matrix = function(numrows, numcols, initial) {

var arr = [];

for (var i = 0; i < numrows; ++i) {

var columns = [];

for (var j = 0; j < numcols; ++j) {

columns[j] = initial;

}

arr[i] = columns;

}

return arr;

}

So you need to create a new function that works with the top one that will generate a random number

Creating Two-Dimensional Arrays

A two-dimensional array is structured like a spreadsheet with rows and columns. To

create a two-dimensional array in JavaScript, we have to create an array and then make

each element of the array an array as well. At the very least, we need to know the number

of rows we want the array to contain. With that information, we can create a two-

dimensional array with n rows and one column:

var twod = [];

var rows = 5;

for (var i = 0; i < rows; ++i) {

twod[i] = [];

}

The problem with this approach is that each element of the array is set to undefined.

A better way to create a two-dimensional array is to follow the example from JavaScript:

Two-Dimensional and Multidimensional Arrays | 27

www.it-ebooks.info

The Good Parts (O’Reilly, p. 64). Crockford extends the JavaScript array object with a

function that sets the number of rows and columns and sets each value to a value passed

to the function. Here is his definition:

Array.matrix = function(numrows, numcols, initial) {

var arr = [];

for (var i = 0; i < numrows; ++i) {

var columns = [];

for (var j = 0; j < numcols; ++j) {

columns[j] = initial;

}

arr[i] = columns;

}

return arr;

}

Here is some code to test the definition:

var nums = Array.matrix(5,5,0);

print(nums[1][1]); // displays 0

var names = Array.matrix(3,3,"");

names[1][2] = "Joe";

print(names[1][2]); // display "Joe"

We can also create a two-dimensional array and initialize it to a set of values in one line:

var grades = [[89, 77, 78],[76, 82, 81],[91, 94, 89]];

print(grades[2][2]); // displays 89

For small data sets, this is the easiest way to create a two-dimensional array.

var grades = [[89, 77, 78],[76, 82, 81],[91, 94, 89]];

var total = 0;

var average = 0.0;

for (var row = 0; row < grades.length; ++row) {

for (var col = 0; col < grades[row].length; ++col) {

total += grades[row][col];

}

average = total / grades[row].length;

print("Student " + parseInt(row+1) + " average: " +

average.toFixed(2));

total = 0;

average = 0.0;

}

List is a popular class

It is also called a list collection

List ADT (Abstarct data typy)

When you create a class you also create it as a function

listSize (property) Number of elements in list

pos (property) Current position in list

length (property) Returns the number of elements in list

clear (function) Clears all elements from list

toString (function) Returns string representation of list

getElement (function) Returns element at current position

insert (function) Inserts new element after existing element

append (function) Adds new element to end of list

remove (function) Removes element from list

front (function) Sets current position to first element of list

end (function) Sets current position to last element of list

prev (function) Moves current position back one element

next (function) Moves current position forward one element

currPos (function) Returns the current position in list

moveTo (function) Moves the current position to specified position

find returns the index

contains returns the Boolean(true or false)

getElement returns the element

a stack follows a rule called lifo(last in first out)

you create it the same way

Sorting Data with Queues This sorting technique is called a radix sort

You will find in data structures that there are array methods

Array is a index and not an object

Hasing is a way

First project is presenting croking algorithms

You can combined hashes, that is the basics of mercle tree