The Behemoth Document: A Pack Rat Developer's Collection of Shit

-Jonathon Hibbard

About this Document

This is a compilation of information I've gathered over the years either for interview preparation, while researching a language/feature/etc., or referred to by a friend. Very little (maybe a few sentences here and there) are things I've written up (which you should be able to identify easily).

Who this Document is NOT Intended for

This document is **not** intended for those who wish to be considered a prodigy over a phone screen. If, however, you use this during your phone screens just to make it to the in-person interview – well, be prepared for the fact that you may never be invited back, may be exposed publically, and may even find yourself unable to land another interview at any other company. Just saying... Not for you. K.

Who this Document IS Intended for

This document ***is*** intended for anyone who understands the majority of the context within this document, and are simply looking for something to work as a refresher to some core concepts, definitions and terminology that you may need help remembering.

What is Contained Inside

There are 3 main parts to this document:

Software Engineer Interview Concepts to Review

This contains the material I go over to get a good refresher of common "definitions" that I've been asked almost every interview. This is ***not*** a complete or definitive list, but it's damn close. Feel free to add whatever I'm missing although there probably won't be much.

Helpful Books

This is currently a small list of books I currently wrote down but it will be updated as time permits. These books are intended to help you study for the actual interview, learn about a specific language, or be a good resource for brushing up on some core concepts listed in the **Software Engineer Interview Concepts to Review** section of this document.

Boatload of URLs

This is a collection of links that helped me build what is in this document. I have trimmed it **a lot** (believe it or not) – mainly to reduce repetitive domain references (and to shorten the list overall). However, it still remains quite large. Want to make it a larger collection? Send me some suggestions and I may add to this over sized monstrosity. Maybe...

Acknowledgements

Um.. yeah.. Well…shit. As I said this is a collection I've been compiling for almost 10 years now. Sorry but I have no idea who to thank, quote, or anything like that. If ***you*** know the source to something here – please – let me know and I'll update. Otherwise – your best guess is mine.

Either way – I hope this document is useful to someone. Enjoy!

Software Engineer Interview Concepts to Review

Data Structures

Data Structures are nothing more than different ways of representing data.

Arrays

Definition

It can store multiple values, and so is in the collection category. The stored values are called elements of the array, and are accessed by a sequence of indices.

In a program, indices may be a mixture of constant values or variables that allow the elements accessed to vary under program control. The indices of an array are totally ordered.

Arrays with more than one index are called *multidimensional arrays*. For example, a matrix is a two-dimensional array.

Common Operations

Indexing

Accessing an array element by its indices. (There is a one to one mapping between an index and its corresponding element).

While the cost of indexing is O(1) on typical current machines, the cost of indexing an element adjacent to a recently accessed element is much faster than indexing an arbitrary random element of a large array.

Slicing

Producing a subarray by putting some constraint on the indices.

For example, PL/1 provides extracting of a row or a column of an array. In Ada any range of the index can be used in order to extract a subarray from a single-dimensional array.

In Python slices can extract any contiguous subset of an array and extended slice notation can extract elements in reversed order and/or by traversing in a given "stride" --- for example a[100:0:-2] would return every odd element from 100 to the beginning of the list: a[99], a[97], ... a[1].

Element Iteration

Some languages make this a universal or implicit operation, some languages have a foreach loop construct for array iteration (in others this must be done with conventional looping and arithmetic).

Miscellaneous

- Sorting the contents of the array to satisfy some relationship.
- Searching an array for the location(s) of some value(s)

Iteration

Querying Bounds of Array Indices
Determining the maximum element index of offset

Querying Indices of an Associative Array
Determining if the collection contains a value for any given key

Common Method Definitions
- Next
- Previous
- Range
- Etc.

Associative Array

Definition
An associative array maps the association between index "keys" and their associated values, generally using more complex hash functions on the keys of the array to map them to their corresponding elements (by pointers, references or memory addresses of some sort).

Associative arrays are referred to variously as "**hashes**" (Perl), "**maps**" (or "mappings" (Lua)), or "**dictionaries**" (Python), as well as "*associative arrays*" (AWK, ksh, and others).

The keys into associative arrays are normally not constrained to be integers, unlike arrays, which generally required contiguous integer ranges.

Different languages may impose various constraints on these keys.

Examples

Creating an Associative Array

One Form of Creation
```
var assoc = {};
assoc['foo'] = 'bar';
assoc['another-key'] = 3;
// dot notation can also be used
assoc.thirdKey = 'we can also do this!';
assoc[2] = 'the index here is the string "2"';
assoc[null] = 'this also works';
assoc[(function(){return 'expr';})()] = 'Can use expressions too';
```

Another Form of Creation
```
var assoc = {
  foo: 'bar',
  'another-key': 3,
  '2': 'the index here is the string "2"',
 };

assoc.thirdKey = 'we can also still do this!';
assoc[null] = 'and this';
assoc[(function(){return 'expr';})()] = and this…';
```

Merging With "Concat"

```
var a = [1,2,3], b = [4,5,6];
var c = a.concat(b);  // [1,2,3,4,5,6]
```

Iterating

```
var key, output = '',
// the associative array
myhash = {
  hello: 3,
  world: 6,
  "!": 9
};

for(key in myhash) {
    if(myhash.hasOwnProperty(key)) {
        // The "key" variable is anonymous, and cannot be accessed via dot notation!
        output += "key is: " + key + " => value is: " + myhash[key] + "\n";
    }
}
```

AVL Tree

Lookup, insertion, and deletion all take O(log n) time…

Definition

In computer science, an AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

Lookup, insertion, and deletion all take O(log n) time in both the average and worst cases, where n is the number of nodes in the tree prior to the operation. Insertions and deletions may require the tree to be rebalanced by one or more tree rotations.

Collections

Definition

Collections are abstractions to represent sets of values. In statically-typed languages, the values are typically of a common data type.

Examples

Creating a Collection

```
var array = [];
array.push('abc');
array.push(123);
array.push(new MyClass);
alert( array[2] );

var map = {};
map['foo'] = 'xyz'; //equivalent to: map.foo = 'xyz';
map['bar'] = new MyClass; //equivalent to: map.bar = new MyClass;
map['1x; ~~:-b'] = 'text'; //no equivalent
alert( map['1x; ~~:-b'] );
```

Compound Data Type

Examples

Create a Two-Dimensional Array at Runtime

```
var w = parseInt( get_input("Enter a width:") );
var w = parseInt( get_input("Enter a height:") );

// create the 2-D array
var a = new Array(h);
for (var i = 0; i < h; i++)
  a[i] = new Array(w);

a[0][0] = 'foo';
WScript.Echo('a[0][0] = ' + a[0][0]);

a = null;

function get_input(prompt) {
   output(prompt);
   try {
      return WScript.StdIn.readLine();
   } catch(e) {
      return readline();
   }
}
function output(prompt) {
   try {
      return WScript.echo(prompt);
   } catch(e) {
      return print(prompt);
   }
}
```

Singly-Linked List

Examples

```
function LinkedList(value, next) {
   this._value = value;
   this._next = next;
}
LinkedList.prototype.value = function() {
   if (arguments.length == 1)
      this._value = arguments[0];
   else
      return this._value;
}
LinkedList.prototype.next = function() {
   if (arguments.length == 1)
      this._next = arguments[0];
   else
      return this._next;
}
// convenience function to assist the creation of linked lists.
function createLinkedListFromArray(ary) {
   var head = new LinkedList(ary[0], null);
   var prev = head;
   for (var i = 1; i < ary.length; i++) {
      var node = new LinkedList(ary[i], null);
      prev.next(node);
      prev = node;
   }
   return head;
}
var head = createLinkedListFromArray([10,20,30,40]);
```

Traversal

```
 LinkedList.prototype.traverse = function(func) {
   func(this);
   if (this.next() != null)
      this.next().traverse(func);
}
```

```
LinkedList.prototype.print = function() {
   this.traverse( function(node) {print(node.value())} );
}
var head = createLinkedListFromArray([10,20,30,40]);
head.print();
```

Doubly-Linked List

Definition

A linked list is a data structure which allows a great deal of flexibility in memory allocation and data sorting. Linked lists depend on references for their organization Information is stored in "nodes" which contain data (integers, strings, etc., sometimes called an "element") and one or more "links" to to other nodes.

The number of links determines what type of linked list it is:

- **One Link** - *singly-linked list*
- Two Links - *doubly-linked list*
- Three Links - *triply-linked list*
- And so on

One or two links are most common kinds of links.   Linked lists have a "**head**" (the first node in the list) and sometimes a "**tail**" (the last node).

Example

Basic

```
function DoublyLinkedList(value, next, prev) {
   this._value = value;
   this._next = next;
   this._prev = prev;
}

DoublyLinkedList.prototype = new LinkedList();
DoublyLinkedList.prototype.prev = function() {
   if (arguments.length == 1)
      this._prev = arguments[0];
   else
      return this._prev;
}

function createDoublyLinkedListFromArray(ary) {
   var node, prev, head = new DoublyLinkedList(ary[0], null, null);
   prev = head;
   for (var i = 1; i < ary.length; i++) {
      node = new DoublyLinkedList(ary[i], null, prev);
      prev.next(node);
      prev = node;
   }
   return head;
}

var head = createDoublyLinkedListFromArray([10,20,30,40]);
```

Insertion

```
DoublyLinkedList.prototype.insertAfter = function(searchValue, nodeToInsert) {
   if (this._value == searchValue) {
      var after = this.next();
      this.next(nodeToInsert);
      nodeToInsert.prev(this);
      nodeToInsert.next(after);
      after.prev(nodeToInsert);
   }
   else if (this.next() == null)
      throw new Error(0, "value '" + searchValue + "' not found in linked list.")
   else
      this.next().insertAfter(searchValue, nodeToInsert);
}

var list = createDoublyLinkedListFromArray(['A','B']);
list.insertAfter('A', new DoublyLinkedList('C', null, null));
```

Traversal

```
DoublyLinkedList.prototype.getTail = function() {
   var tail;
   this.traverse(function(node){tail = node;});
   return tail;
}
DoublyLinkedList.prototype.traverseBackward = function(func) {
   func(this);
   if (this.prev() != null)
      this.prev().traverseBackward(func);
}
DoublyLinkedList.prototype.printBackward = function() {
   this.traverseBackward( function(node) {print(node.value())} );
}

var head = createDoublyLinkedListFromArray([10,20,30,40]);
head.print();
head.getTail().printBackward();
```

Set

Definition:

A set is a collection of elements, without duplicates and without order.

One might implement a set using an associative array (with set elements as array keys and some dummy value as the values).

One might also implement a set with a binary search tree, or with a hash table, or with an ordered array of binary bits (operated on with bitwise binary operators).

The basic test, $m \in S$, is O(n) with a sequential list of elements, O(log n) with a balanced binary search tree, or (O(1) average-case, O(n) worst case) with a hash table.

Operations

- $A \cup B$ -- union; a set of all elements either in set A or in set B.
- $A \cap B$ -- intersection; a set of all elements in both set A and set B.
- $A \setminus B$ -- difference; a set of all elements in set A, except those in set B.
- $A \subseteq B$ -- subset; true if every element in set A is also in set B.
- $A = B$ -- equality; true if every element of set A is in set B and vice-versa.

Stack

Definition:

A stack is a container of elements with last in, first out access policy. Sometimes it also called LIFO. The stack is accessed through its top. The basic stack operations are:

- **Push** - *stores a new element onto the stack top*
- **Pop** - *returns the last pushed stack element, while removing it from the stack*
- **Empty** - *tests if the stack contains no elements*
- **Top** (aka **Peek**) - *returns the topmost element without modifying the stack.*

Stacks allow a very simple hardware implementation. They are common in almost all processors. In programming stacks are also very popular for their way (LIFO) of resource management, usually memory. Nested scopes of language objects are naturally implemented by a stack (sometimes by multiple stacks).

This is a classical way to implement local variables of a reentrant or recursive subprogram. Stacks are also used to describe a formal computational framework. See stack machine. Many algorithms in pattern matching, compiler construction (e.g. recursive descent parsers), and machine learning (e.g. based on tree traversal) have a natural representation in terms of stacks.

Examples

Basic

```
// The built-in Array class already has stack primitives.
var stack = [];
stack.push(1)
stack.push(2,3);
print(stack.pop());   // 3
print(stack.length);   // 2, stack empty if 0

// Here's a constructor that wraps the array:
function Stack() {
   this.data = new Array();
   this.push  = function(element) {this.data.push(element)}
   this.pop   = function() {return this.data.pop()}
   this.empty = function() {return this.data.length == 0}
   this.peek  = function() {return this.data[0]}
}
```

Tree traversal

```
function BinaryTree(value, left, right) {
   this.value = value;
   this.left = left;
   this.right = right;
}
BinaryTree.prototype.preorder  = function(f) {this.walk(f,['this','left','right'])}
BinaryTree.prototype.inorder   = function(f) {this.walk(f,['left','this','right'])}
BinaryTree.prototype.postorder = function(f) {this.walk(f,['left','right','this'])}
BinaryTree.prototype.walk = function(func, order) {
   for (var i in order)
      switch (order[i]) {
         case "this": func(this.value); break;
         case "left": if (this.left) this.left.walk(func, order); break;
         case "right": if (this.right) this.right.walk(func, order); break;
      }
}
BinaryTree.prototype.levelorder = function(func) {
   var queue = [this];
   while (queue.length != 0) {
      var node = queue.shift();
      func(node.value);
      if (node.left) queue.push(node.left);
      if (node.right) queue.push(node.right);
   }
}
```

```
// convenience function for creating a binary tree
function createBinaryTreeFromArray(ary) {
    var left = null, right = null;
    if (ary[1]) left = createBinaryTreeFromArray(ary[1]);
    if (ary[2]) right = createBinaryTreeFromArray(ary[2]);
    return new BinaryTree(ary[0], left, right);
}

var tree = createBinaryTreeFromArray([1, [2, [4, [7]], [5]], [3, [6, [8],[9]]]]);

print("*** preorder ***");   tree.preorder(print);
print("*** inorder ***");    tree.inorder(print);
print("*** postorder ***");  tree.postorder(print);
print("*** levelorder ***"); tree.levelorder(print);
```

Variable-length quantity

Examples

Note: this is in C, but should be simple to translate to Javascript…

```
#include <stdio.h>
#include <stdint.h>

void to_seq(uint64_t x, uint8_t *out) {
  int i, j;
  for (i = 9; i > 0; i--) {
    if (x & 127ULL << i * 7) break;
  }
  for (j = 0; j <= i; j++)
    out[j] = ((x >> ((i - j) * 7)) & 127) | 128;

  out[i] ^= 128;
}

uint64_t from_seq(uint8_t *in) {
  uint64_t r = 0;

  do {
    r = (r << 7) | (uint64_t)(*in & 127);
  } while (*in++ & 128);

  return r;
}

int main() {
  uint8_t s[10];
  uint64_t x[] = { 0x7f, 0x4000, 0, 0x3ffffe, 0x1fffff, 0x200000, 0x3311a1234df31413ULL};
  int i, j;

  for (j = 0; j < sizeof(x)/8; j++) {
    to_seq(x[j], s);
    printf("seq from %llx: [ ", x[j]);
    i = 0;
    do { printf("%02x ", s[i]); } while ((s[i++] & 128));
    printf("] back: %llx\n", from_seq(s));
  }
  return 0;
}
```

VList

Definition

In computer science, the VList is a persistent data structure that combines the fast indexing of arrays with the easy extension of cons-based (or singly-linked) linked lists.

Like arrays, VLists have constant-time lookup on average and are highly compact, requiring only O(log n) storage for pointers, allowing them to take advantage of locality of reference. Like singly-linked or cons-based lists, they are persistent, and elements can be added to or removed from the front in constant time. Length can also be found in O(log n) time.

Primary Operations

- Locate the kth element (O(1) average, O(log n) worst-case)
- Add an element to the front of the VList (O(1) average, with an occasional allocation)
- Obtain a new array beginning at the second element of an old array (O(1))
- Compute the length of the list (O(log n))

Example

Note: this is in C, but should be simple to translate to Javascript…

```
#include <stdio.h>
#include <stdlib.h>

typedef struct sublist{
  struct sublist* next;
  int *buf;
} sublist_t;

sublist_t* sublist_new(size_t s)
{
  sublist_t* sub = malloc(sizeof(sublist_t) + sizeof(int) * s);
  sub->buf = (int*)(sub + 1);
  sub->next = 0;
  return sub;
}

typedef struct vlist_t {
  sublist_t* head;
  size_t last_size, ofs;
} vlist_t, *vlist;

vlist v_new()
{
  vlist v = malloc(sizeof(vlist_t));
  v->head = sublist_new(1);
  v->last_size = 1;
  v->ofs = 0;
  return v;
}

void v_del(vlist v)
{
  sublist_t *s;
  while (v->head) {
    s = v->head->next;
    free(v->head);
    v->head = s;
  }
  free(v);
}

inline size_t v_size(vlist v)
{
  return v->last_size * 2 - v->ofs - 2;
}

int* v_addr(vlist v, size_t idx)
{
  sublist_t *s = v->head;
  size_t top = v->last_size, i = idx + v->ofs;

  if (i + 2 >= (top << 1)) {
    fprintf(stderr, "!: idx %d out of range\n", (int)idx);
    abort();
  }
  while (s && i >= top) {
    s = s->next, i ^= top;
    top >>= 1;
  }
  return s->buf + i;
```

```
}

inline int v_elem(vlist v, size_t idx)
{
  return *v_addr(v, idx);
}

int* v_unshift(vlist v, int x)
{
  sublist_t* s;
  int *p;

  if (!v->ofs) {
    if (!(s = sublist_new(v->last_size << 1))) {
      fprintf(stderr, "?: alloc failure\n");
      return 0;
    }
    v->ofs = (v->last_size <<= 1);
    s->next = v->head;
    v->head = s;
  }
  *(p = v->head->buf + --v->ofs) = x;
  return p;
}

int v_shift(vlist v)
{
  sublist_t* s;
  int x;

  if (v->last_size == 1 && v->ofs == 1) {
    fprintf(stderr, "!: empty list\n");
    abort();
  }
  x = v->head->buf[v->ofs++];

  if (v->ofs == v->last_size) {
    v->ofs = 0;
    if (v->last_size > 1) {
      s = v->head, v->head = s->next;
      v->last_size >>= 1;
      free(s);
    }
  }
  return x;
}

int main()
```

```
{
  int i;

  vlist v = v_new();
  for (i = 0; i < 10; i++) v_unshift(v, i);

  printf("size: %d\n", v_size(v));
  for (i = 0; i < 10; i++) printf("v[%d] = %d\n", i, v_elem(v, i));
  for (i = 0; i < 10; i++) printf("shift: %d\n", v_shift(v));

  /* v_shift(v); */ /* <- boom */

  v_del(v);
  return 0;
}
```

On-Site Interviews

Rackspace

Interviewed by Two Engineers (with 5 to 8 years of experience)

- Focused on language specifics.
    - Rate yourself on language X
    - Asked Obscure questions on language usage
    - What's the most complicated task you have completed with language "X"
- Asked about formal definitions
    - Closures
    - class functions
    - polymorphism
- Asked to design a solution to integrate various independent businesses within the company.
- Asked what type of role you see yourself in
    - From what I could tell the company does not have systems engineers, data modeling specialists, etc.
    - They are all software engineers with various levels of responsibilities.
    - Some are entry level, domain owners (control all aspects of a subsystem), senior developers.)

Interviewed by Three Senior-level Developers

- Asked to do data modeling
    - **Advice**: *Study data modeling patterns and when to apply them.*
- Asked about design patterns
    - Implement singleton pattern.
- Asked critical thinking questions
    - See programming interviews exposed book.

Interviewed by Senior Management/Human Resources

- Where do you see yourself in 5 years?
- Why Rackspace?
- Rackspace is a big company, you have only worked at small companies, what kind of culture shock do you expect?

Common Development Definitions

Coupling

Coupling among classes or subsystems is a measure of how interconnected those classes or subsystems are.

Tight coupling

Means that related classes have to know internal details of each other, changes ripple through the system, and the system is potentially harder to understand

Loose Coupling

Patterns are loosely tied together with few connections. It makes it easier to separate the patterns and extend code. Promotes quicker, simpler future design changes.

Important Characteristics of Loose Coupling

- Makes the code easier to read.
- Makes our classes easier to consume by other developers by hiding the ugly inner workings of our classes behind well-designed APIs.
- Isolates potential changes to a small area of code.
- Reuse classes in completely new contexts

Cohesion

The academic definition of cohesion is that it is a measure of how closely related all the responsibilities, data, and methods of a class are to each other. I like to think of cohesion as a measure of whether a class has a well-defined role within the system.

To stretch my earlier conversation analogy, a system consisting of cohesive classes and subsystems is like a well-designed online discussion group. Each area in the online group is narrowly focused on one specific topic so the discussion is easy to follow, and if you're looking for a dialog on a certain subject, there's only one room you have to visit.

Polymorphism

Polymorphism is the ability to assign different meanings to something in different contexts.   That is, polymorphism allows an entity such as a variable, a function, or an object to have more than one form.

The ability to assign different meanings to something in different contexts and allows things like variables, functions, or an object to have more than one form.

Examples

Example 1

A variable with a given name may be allowed to have different forms and the program can determine which form of the variable to use at the time of execution.

For example, a variable named USERID may be capable of being either an integer (whole number) or a string of characters (perhaps because the programmer wants to allow a user to enter a user ID as either an employee number - an integer - or with a name - a string of characters).

By giving the program a way to distinguish which form is being handled in each case, either kind can be recognized and handled.

Example 2

A named function can also vary depending on the parameters it is given.

For example, if given a variable that is an integer, the function chosen would be to seek a match against a list of employee numbers;

if the variable were a string, it would seek a match against a list of names. In either case, both functions would be known in the program by the same name. This type of polymorphism is sometimes known as overloading.

Example 3

Polymorphism can mean, as in the ML language, a data type of “any,” such that when specified for a list, a list containing any data types can be processed by a function.

For example, if a function simply determines the length of a list, it doesn’t matter what data types are in the list.

Example 4

In PHP, polymorphism means that if B is a descendant of A and a function can accept A as a parameter, it can also accept B.

S.O.L.I.D

It represents principals for creating easier to maintain code.   It’s made up of:

(S)ingle responsibility principle

This is the notion that an object should have only a single responsibility.

(O)pen/closed principle

This is the notion that “software entities … should be open for extension, but closed for modification”.

(L)iskov substitution principle

This is the notion that “objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program”.

(I)nterface segregation principle
This is the notion that “many client specific interfaces are better than one general purpose interface.

(D)ependency inversion principle
This is the notion that one should “Depend upon Abstractions. Do not depend upon concretions.” (IE: Unit testing)

http://blog.objectmentor.com/articles/2009/02/12/getting-a-solid-start

Closures
It’s just an anonymous function. Values within have access to the parent’s scope.
jQuery example:
```
var $hello = ‘there’;
$bob = function() { alert($hello); };
```

Software Development Process

A **software development process**, also known as a **software development life-cycle (SDLC)**, is a structure imposed on the development of a software product. Similar terms include *software life cycle* and *software process*. It is often considered a subset of systems development life cycle. There are several models for such processes, each describing approaches to a variety of tasks or activities that take place during the process. Some people consider a life-cycle model a more general term and a software development process a more specific term. For example, there are many specific software development processes that 'fit' the spiral life-cycle model. ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

Software Development Activities

Planning

Planning is an objective of each and every activity, where we want to discover things that belong to the project. An important task in creating a software program is extracting the requirements or requirements analysis.[1] Customers typically have an abstract idea of what they want as an end result, but not what *software* should do. Skilled and experienced software engineers recognize incomplete, ambiguous, or even contradictory requirements at this point. Frequently demonstrating live code may help reduce the risk that the requirements are incorrect.

Implementation, Testing and Documentation

Implementation is the part of the process where software engineers actually program the code for the project.

Software testing is an integral and important phase of the software development process. This part of the process ensures that defects are recognized as soon as possible.

Documenting the internal design of software for the purpose of future maintenance and enhancement is done throughout development. This may also include the writing of an API, be it external or internal. The software engineering process chosen by the developing team will determine how much internal documentation (if any) is necessary. Plan-driven models (e.g., Waterfall) generally produce more documentation than Agile models.

Deployment and Maintenance

Deployment starts after the code is appropriately tested, approved for release, and sold or otherwise distributed into a production environment. This may involve installation, customization (such as by setting parameters to the customer's values), testing, and possibly an extended period of evaluation.[*citation needed*]

Software training and support is important, as software is only effective if it is used correctly.[*citation needed*]

Maintaining and enhancing software to cope with newly discovered faults or requirements can take substantial time and effort, as missed requirements may force redesign of the software.[

Software Development Models

Waterfall

The waterfall model shows a process, where developers are to follow these phases in order:

1. Requirements specification (Requirements analysis)
2. Software design
3. Implementation and Integration
4. Testing (or Validation)
5. Deployment (or Installation)
6. Maintenance

In a strict Waterfall model, after each phase is finished, it proceeds to the next one. Reviews may occur before moving to the next phase which allows for the possibility of changes (which may involve a formal change control process). Reviews may also be employed to ensure that the phase is indeed complete; the phase completion criteria are often referred to as a “gate” that the project must pass through to move to the next phase. Waterfall discourages revisiting and revising any prior phase once it’s complete. This “inflexibility” in a pure Waterfall model has been a source of criticism by supporters of other more “flexible” models.

The Waterfall model is also commonly taught with the mnemonic A Dance in the Dark Every Monday, representing Analysis, Design, Implementation, Testing, Documentation and Deployment, and Maintenance.[

Iterative and Incremental Development

Iterative development[2] prescribes the construction of initially small but ever-larger portions of a software project to help all those involved to uncover important issues early before problems or faulty assumptions can lead to disaster.

Agile Development

Agile software development uses iterative development as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes use feedback, rather than planning, as their primary control mechanism. The feedback is driven by regular tests and releases of the evolving software.

There are many variations of agile processes:

- In Extreme Programming (XP), the phases are carried out in extremely small (or “continuous”) steps compared to the older, “batch” processes. The (intentionally incomplete) first pass through the steps might take a day or a week, rather than the months or years of each complete step in the Waterfall model. First, one writes automated tests, to provide concrete goals for development. Next is coding (by a pair of programmers), which is complete when all the tests pass, and the programmers can’t think of any more tests that are needed. Design and architecture emerge out of refactoring, and come after coding. The same people who do the coding do design. (Only the last feature — merging design and code — is common to *all* the other agile processes.) The incomplete but functional system is deployed or demonstrated for (some subset of) the users (at least one of which is on the development team). At this point, the practitioners start again on writing tests for the next most important part of the system.[3]

- Scrum
- Dynamic systems development method

Rapid Application Development

Rapid application development R.A.D is a software development methodology that uses minimal planning in favor of rapid prototyping. The "planning" of software developed using RAD is interleaved with writing the software itself. The lack of extensive pre-planning generally allows software to be written much faster, and makes it easier to change requirements.RAD involves methods like iterative development and software prototyping. According to Whitten (2004), it is a merger of various structured techniques, especially data-driven Information Engineering, with prototyping techniques to accelerate software systems development.[4]

In rapid application development, structured techniques and prototyping are especially used to define users' requirements and to design the final system. The development process starts with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements are verified using prototyping, eventually to refine the data and process models. These stages are repeated iteratively; further development results in "a combined business requirements and technical design statement to be used for constructing new systems".[4]

Design Patterns

Creational Patterns

Factory Method

Along with singleton pattern the factories are the most used patterns. The Factory Pattern defines an interface for creating an object, but let subclasses decide which class to instantiate.

When to Use:

- The creation of objects derived from a common superclass to the factory
- The base class does not know what concrete classes will be required to create - delegates to its subclasses the creation of concrete objects
- Subclasses are aware of the concrete classes that must be instantiated

How to Use:

Create an abstract class and a set of concrete factories subclasses. The subclasses are responsible for creating concrete product objects; for factory method is possible adding new product classes without changing the abstract factory. The same result can be achieved for simplified factory pattern if reflection is used.

Abstract Factory

Provides an interface for creating families of related or dependent objects without specifying their concrete classes.

When to use:

- A system should be configured with one of multiple families of products
- A system should be independent of how its products are created, composed and represented
- Products from the same family should be used all together, products from different families ahould not be used togheter and this constraint must be ensured.
- Only the product interfaces are revealed, the implementation remains hidden to the clients.

Examples:

- java.awt.Toolkit - the abstract superclass of all actual implementations of the Abstract Window Toolkit. Subclasses of Toolkit are used to bind the various components to particular native toolkit implementations(Java AWT).
- javax.swing.LookAndFeel - an abstract swing factory to swithct between several look and feel for the components displayed(Java Swing).
- java.sql.Connection - an abstract factory which create Statements, PreparedStatements, CallableStatements,... for each database flavor.

Builder

Separate the construction of a complex object from its representation so that the same construction process can create different representations.

Prototype

Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.

Singleton

Ensure a class only has one instance, and provide a global point of access to it.

Other design patterns implemented as Singletons:

- Factories and Abstract Factories
- Builder
- Prototype

When to use:

Singleton pattern should be used when we must ensure that only one instance of a class is created and when the instance must be available through all the code. A special care should be taken in multithreading environments when multible threads must access the same resources throught the same singleton object.

Example/Common Usage:

- Logger Classes
- Configuration Classes
  Accesing resources in shared mode

Structural Patterns

Adapter

Convert the interface of a class into another interface clients expect.

Bridge

Decouple an abstraction from its implementation so that the two can vary independently.

Composite

Compose objects into tree structures to represent part-whole hierarchies.

Decorator

Attach additional responsibilities to an object dynamically.

Façade

Provide a unified interface to a set of interfaces in a subsystem.

Flyweight

Use sharing to support large numbers of fine-grained objects efficiently.

Proxy

Provide a surrogate or placeholder for another object to control access to it.

Behavioral Patterns

Chain of Responsibility

Avoid coupling the sender of a request to its receiver ... Chain the receiving objects and pass the request along the chain until an object handles it.

Command

Encapsulate a request as an object, thereby letting you parameterize clients with different requests...

Interpreter

Given a language, define a representation for its grammar along with an interpreter…

Iterator

Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.

Mediator

Define an object that encapsulates how a set of objects interact...

Memento

Capture and externalize an object’s internal state so that the object can be restored to this state later.

Observer

Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated.

State

Allow an object to alter its behavior when its internal state changes.

Strategy

Define a family of algorithms, encapsulate each one, and make them interchangeable.

Template

Define the skeleton of an algorithm in an operation, deferring some steps to subclasses…

Visitor

Represent an operation to be performed on the elements of an object structure.

Examples

Factory

Instantiates objects at runtime based on criteria (type) needed.

Example:

```
class fooBar {}
class Factory {
  public static function build($class) {
    if(!class_exists($class)) {
      throw new Exception("Invalid class ($class) requested!");
    }
    return new $class;
  }
}
```

Adapter

An adapter pattern is used when we want to combine the methods of 2 separate classes together without affecting an object's structure/implementation.

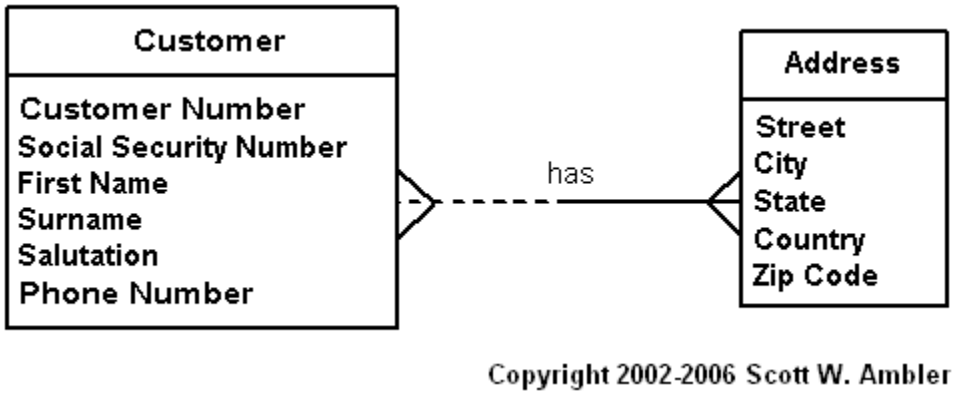
Example:

```
class foo {
  public function say() {
    return "Say Hello!";
  }
}

class bar {
  public function beforeYouWave($say_something_string) {
    echo "I'm doing something cook, but I want to say " . $say_something_string;
  }
}

class fooBarAdapter extends bar {
    public function __construct(foo $fooObj){
      $this-> beforeYouWave ($fooObj->say());
    }
  }
```

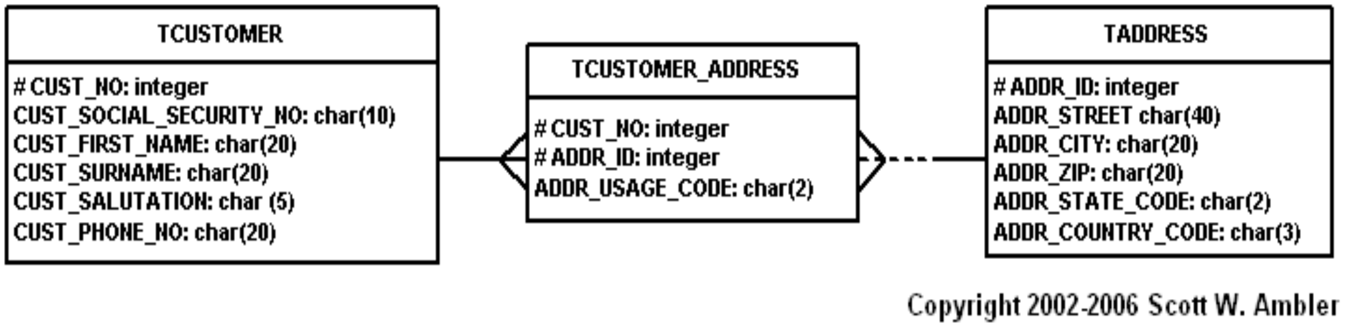
Data Modeling

Data Modeling is just breaking down the requirements into a "flow" style that makes sense both from a developer's perspective, but also a client's. It clearly explains the requirements of the application, scope of each step, and what the overall objective/goal is.

Example 1 (Logical Data Model: high-level (client) representation)



Example 2 (Physical Data Model: low-level (table schema almost) representation)



REST API Development

Representational State Transfer

Definition

If you don't know what REST, RESTful, or ROI is – a definition will not help you. Instead, here is a great book that you should read (carefully!): http://www.amazon.com/RESTful-Web-Services-ebook/dp/B0043D2ED6/ref=sr_1_1?ie=UTF8&qid=1382677039&sr=8-1&keywords=Restful+Web+Services

Response Codes for HTTP

100 = "Continue"

101 = "Switching Protocols"

200 = "OK"

201 = "Created"

202 = "Accepted"

203 = "Non-Authoritative Information"

204 = "No Content"

205 = "Reset Content"

206 = "Partial Content"

300 = "Multiple Choices"

301 = "Moved Permanently"

302 = "Found"

303 = "See Other"

304 = "Not Modified"

305 = "Use Proxy"

306 = "(Unused)"

307 = "Temporary Redirect"

400 = "Bad Request"

401 = "Unauthorized"

402 = "Payment Required"

403 = "Forbidden"

404 = "Not Found"

405 = "Method Not Allowed"

406 = "Not Acceptable"

407 = "Proxy Authentication Required"

408 = "Request Timeout"

409 = "Conflict"

410 = "Gone"

411 = "Length Required"

412 = "Precondition Failed"

413 = "Request Entity Too Large"

414 = “Request-URI Too Long”
415 = “Unsupported Media Type”
416 = “Requested Range Not Satisfiable”
417 = “Expectation Failed”
500 = “Internal Server Error”
501 = “Not Implemented”
502 = “Bad Gateway”
503 = “Service Unavailable”
504 = “Gateway Timeout”
505 = “HTTP Version Not Supported”

REST vs SOAP (Simple Object Access Protocol)

SOAP

- A protocol for exchanging data
- Biggest problem is complexity, but with that complexity does come added security
    - Can send/receive over protocols other than HTTP/HTTPS
- Uses many different protocols (HTTP, HTTPS, SMTP, JMS (Java Message Service), etc.)
- Sends requests/responses via soap envelopes in xml
- Very tough to use with Javascript as it requires parsing XML which is no fun - even with libraries.

REST

REST is an ***architecture***. It typically is used to send JSON and/or XML using protocols supported by HTTP (HTTP/HTTPS, SSL/TSL). Operations are performed using **CRUD** (Create, Read, Update, Delete).

Create (POST or PUT)

Think of the differences between these two in terms of a MySQL Insert Statement..

POST

Equivalent to “INSERT INTO ‘table‘ …” my values, when the record is going to be brand new, and the ID is generated automatically.

PUT

Equivalent to “INSERT INTO TABLE ON DUPLICATE KEY UPDATE”

READ (GET)

A **GET** operation is typically used for the following operations:

- SELECT – Retrieve/request information to be *returned* as the result
- UPDATE (PUT) – Used when the *PUT* protocol (see above) may not be available or disabled by the server
- DELETE – Used when the *DELETE* protocol (see below) is not available or disabled by the server

DELETE (DELETE)

Issued to *delete* something.

ROA (Resource Oriented Architecture) Guidelines/Definitions

Common Terminology/information

Resources

A **Resource** is an object that represents *something*. For example, *books* would be the **resource** of this request: http://www.example.com/**books**/

Promotes Descriptive URI's

Example: *http://www.example.com/software/releases/1.0.3.tar.gz*

Is Statelessness

The Server never relies on previous requests because if that information was important the client would send that in with the follow-up request.

Put vs POST

The client uses PUT when its in charge of deciding which URI the new resource should have. The client uses POST when the *server* is in charge of deciding which URI the new resource should have.

URL Design Best Practices

Domains and subdomains should be used to group or separate responses for localization, distribution, monitoring/security policies, etc.

Examples of Localization and Partitioning

Localization

- en.example.com
- fr.example.com
- etc.

Partitioning (Separating Resources)

Allows for different routing, monitoring, or security policies for HTML and non-HTML representations

- http://www.example.com/books/1234
  - Used with browsers
- http://api.exampe.com/books/1234
  - Custom Clients (PHP CURL, etc.)

Forward-slashes separate Paths in a URI

Hierarchical Relationships Example

http://www.example.com/book/computer/

Properly use Singular/Plural Names to Represent Resource Return Result(s)!

Plural (/books/) – Multiple Books can/will be returned

Singular (/book/) – One Book can/will be returned

Underscores (_) and Hyphens (-)
Separate words, makes a more "user friendly" reading url than no spaces.

Ampersand (&)
Separate params as we have all done with GET (right…?)

Comma (,) and Semicolon (;)
Indicates non-hierarchical portions like arrays/matrix/etc. Matrix patterns like coordinates for pixel location on an image, or coordinates on a map (coordinantes;x=123&y=123.0003)

FullStop (.)

Used for filenames

Opaque URL Identifiers
Requires to makes sure that each resource has a distinct URL

Keeping URLs Cool
- Using rewrite rules for implementation changes.
- Redirect old URIs and issue a 301 redirect (Moved Permanently) or 410 (Gone)

Examples:

REQUEST

GET /users/1 HTTP/1.1
HOST: www.example.com
Accept: application/json

RESPONSE

HTTP/1.1 301 Found
Location: http://www.example.com/administration/users/1

Conditional Requests (LOCK RECORD)

Pessimistic Concurrency Control (POC)
Locks a resource, makes changes, unlocks it

Optimistic Concurrency Control (OCO)
Client receives a token, attempts a "write" (PUT/POST/DELETE) request and succeeds only if the token is still valid.

Last-Modified and If-None-Match
Note: *A client's intent is to ask for a fresh representation only if the representation was modified since the date-time specified and/or if none of the supplied entry tags matches.*

Last-Modified

Validates Cached Representations:

If-Modified-Since and If-None-Match

- Validates cached representations

If –Unmodified-Since and If-Match

Represents a preconditions for concurrency control

Note *Client's intent is to perform an operation only if the representation was not modified since the date-time specified and/or if the supplied entity tags match.*

Generate Unique/Identifiable ETag Headers

Used for supporting conditional Requests

JSONP

This is a method for allowing cross-site JSON requests. It achieves it by the philosophy of dynamically creating a <script> tag in the document body with a source to the referring site.

Options and Usage

Dynamically create the source tag, and parse out the <script></script> tags from the body and use the RESPONSE data returned (JSON)

A typical response will look something like:

```
<script>
{ key: value…….}
</script>
```

You can issue a callback for services (such as **REST API's**) that support them.

To supply callback, append callback=? where "?" is the method you want the REST API to wrap the JSON response up with.

```
$.get('example.com/books/?callback=myMethod');
//….
// in some js file/section of the page later…
// do something.
myMethod = function($jsonData) { … };
```

The response from the API will look something like this:

```
<script>
myCallbackMethod({key:value…..});
</script>
```

Agile Engineering Practices

Test Driven Development

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards (phpUnit). – **ya may want to look into this young fella!**

Continuous Integration

To integrate, update your sandbox with the latest code from the repository, make sure everything builds, then commit your code back to the repository. You can integrate any time you have a successful build. With test-driven development, that should happen every few minutes. I integrate whenever I make a significant change to the code (or create something you think the rest of the team will want right away).

At the end of each Iteration, the team's code should be as ready to a real release as possible. The goal is to make preparing for a release such an ordinary occurrence that, when you actually do ship, it's a nonevent1. Some teams that use continuous integration automatically burn an installation CD every time they integrate. Others create a disk image or, for network-deployed products, automatically deploy to staging servers.

http://jamesshore.com/Agile-Book/continuous_integration.html

MySQL Database

Relational Database Design (MySQL)

Just think about how you build tables using MyISAM and you'll be fine (indexes, primary keys, relations through foreign keys (used by the indexes), etc.).

Clustered and non-Clustered Index

I have only worked with the MyISAM engine in my professional career. This was mainly due to the guidance I was given from my DBA in which I stuck to what was correct for our company. I've read up, though, on innoDB and how it uses "clustered indexes" but I am by no means an expert on the subject as I've already stated. It is something I'm interested in learning and using though for sure.

Normalized vs Denormalized Table Examples

Normalization attempts to remove redundancy in a table by following the rules of normal forms.

Rules of Normalization

1st Normal Form Rule:

Eliminate repeating groups of data by creating separate tables of related data.

2nd Normal Form

Eliminate redundant data by making sure the fields in the table are dependent on the WHOLE key, not just part of it.

The main idea here is applied to things when relating to candidate keys. Candidate keys are just keys that, when put together, allow us to find the row we're looking for (or even the primary key).

This is true because there shouldn't be, at this stage, repeating groups of data by rule of the 1st normal form rule.

3rd Normal Form

Eliminate all attributes from a table that are not directly dependent upon the primary key.

If we have fields in this table that are not dependent on the primary key, they probably don't need to be in the table.

Example of Normalization

customer

customer_id

frirst_name

last_name

item

item_id

item_name

<u>customer_item</u>

customer_id

item_id

Example of DeNormalization

This one breaks 1st Normal Rule

<u>customer</u>

customer_id
first_name
last_name
item_name

This one breaks 2nd Normal Rule

<u>customer</u>

customer_id
first_name
last_name
item_id

<u>item</u>

item_id

item_name

Hands-on Interview Questions

Implement a data model for a toy train company.

The company produces parts that are included for purchase in various kits. Design a data model that keeps track of parts, kits (with different -version of the same kit), orders, and inventory.

What is the one big mistake you've done in the last project and how did you solve it?

Write a function that takes in the following array, and returns one list that is even, and another that is odd.

```
// Given
$ints=array(1,21,53,84,50,66,7,38,9);

Function getEvenOddFromArray(array $ints){
  $evenOddArray = array('even' => array(), 'odd' => array());
  if(empty($ints)) return $evenOddArray;
  foreach($ints as $int) {
    if(!is_int($int)) continue;

    if($int%2 == 0) {
      $evenOddArray ['even'][] = $int;
    } else {
      $evenOddArray ['odd'][] = $int;
    }
  }
  return $evenOddArray;
}
```

Reorder the string in reverse order without using str_rev()

HelloWorld

```
foreach(rsort(str_split("HelloWorld")) as $char) echo $char;
```

Define classes for describing a soda vending machine

```
class vendingMachine {
  private $required_inventory_criteria = array('name', 'price', 'quantity', ' size');
  private $inventory = array();

  private $current_balance = 0;
  private $currencies = array();

  public function __construct(array $inventory, array $currencies) {
    try {
      $this->validateInventoryCriteria($inventory);
    } catch(Exception $e) {
      throw new Exception('Required Inventory Check Failed!  Report Was: ' . $e->getMessage());
    }
    try {
      $this->validateCurrencies($currencies);
    } catch(Exception $e) {
      throw new Exception('Required Currency Check Failed!  Report Was: ' . $e->getMessage());
  }
  // blah blah blah blah….
}
```

Miscellaneous Questions (and scribblings…)

What is a “.frm” file for MySQL?

It describes the table’s format (i.e. the table definition).

What makes you different?

If you had a problem with another employee, how would you deal with it?

How would you cope with a difficult customer?

Useful Books

Useful Books

This list will continue to grow…

For Interviews

Cracking the Coding Interview: 150 Programming Questions and Solutions

Programming Interviews Exposed: Secrets to Landing Your Next Job

For Development (Algorithms, Services, etc.)

Programming Pearls

Introduction to Algorithms o Authors: Cormen/Leiserson/Rivest/ Stein: o Alternative Reading: CLR textbook

Restful Web Services

Boatload of URLs

BOATLOAD OF URLS

Google Interview Resources

Interview Process and Tips

http://www.payscale.com - Payscale.com
http://www.careercup.com/page?pid=google-interview-questions - Google Interview Questions
http://www.quora.com/Google-Interview-Questions/recent - Google Interview Questions Recent Feed
http://www.google.com/about/jobs/lifeatgoogle/hiringprocess/ - How we hire - Google Jobs
http://www.youtube.com/watch?v=w887NIa_V9w - Interviewing at Google - YouTube
http://www.youtube.com/lifeatgoogle - Life at Google - YouTube
https://gist.github.com/KWMalik/3734578 - My answers to over 100 Google interview questions
http://nexxus21.blogspot.com/2013/08/getting-job-google-edition.html - Getting the job
https://groups.google.com/forum/#!topic/stljs/zy0ekJBCKTs - Tough Javascript Interview Questions
http://wesbos.com/interviewing-with-google/ - What I learned interviewing with Google

Algorithms

GREAT Resources

http://www.topcoder.com - TopCoder
http://www.geeksforgeeks.org/ - GeeksforGeeks
http://www.anylogic.com/anylogic/help/index.jsp?topic=/com.xj.anylogic.help/html/code/Arrays_Collections.html - AnyLogic Advanced
http://leetcode.com/ - LeetCode
http://rosettacode.org - Rosetta Code
http://c2.com/ - c2.com
http://bigocheatsheet.com/ - Big-O Algorithm Complexity Cheat Sheet
http://www.2ality.com/2012/11/var-statement-rules.html - Variable declarations: three rules you can break
http://coding.smashingmagazine.com/2012/11/05/writing-fast-memory-efficient-javascript/ - Writing Fast, Memory-Efficient JavaScript

Graphs

http://www.radford.edu/~nokie/classes/360/graphs-terms.html - Graphs - Terminology and Representation
http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index - Algorithm Tutorials
http://en.wikipedia.org/wiki/Depth-first_search - Depth-first search - Wikipedia, the free encyclopedia

Algorithms and Data Structures to Study

http://steve-yegge.blogspot.ca/2008/03/get-that-job-at-google.html - Stevey&#39;s Blog Rants: Get that job at Google

http://www.careercup.com/forumpost?id=14998753 - Use of advanced data structures on a Google interview | CareerCup
http://java.dzone.com/articles/interviewing-google-read - Interviewing with Google? Read this! | Javalobby
http://katemats.com/interview-questions/ - Epic List of Interview Questions | kate{mats}
http://stackoverflow.com/questions/13736362/javascript-floor-number - JavaScript .floor number
http://www.refactoring.com/catalog/replaceRecursionWithIteration.html - replaceRecursionWithIteration
http://en.wikipedia.org/wiki/Tail_call - Tail call
http://www.devthought.com/2012/01/18/an-object-is-not-a-hash/ - www.devthought.com/2012/01/18/an-object-is-not-a-hash/
http://www.radford.edu/~nokie/classes/360/graphs-mst.html - Graphs - Minimum Spanning Trees
http://www.radford.edu/~nokie/classes/360/ - ITEC 360 - Data Structures and Analysis of Algorithms
http://www2.hawaii.edu/~janst/211/lecture/Heaps.htm - Heaps
http://en.literateprograms.org/Merge_sort_%28JavaScript%29 - Merge sort (JavaScript) - LiteratePrograms
http://jibbering.com/faq/notes/type-conversion/ - Javascript Type-Conversion
http://www.programmerinterview.com/index.php/data-structures/big-o-notation/ - Big O Notation Explained with an Example

Data Modeling

http://www.agiledata.org/essays/agileDataModeling.html - Agile/Evolutionary Data Modeling: From Domain Modeling to Physical Modeling
http://eng.pulse.me/2012/08/ - August « 2012 « Pulse News Engineering Blog
http://www.agiledata.org/essays/dataModeling101.html - Rackspace
http://searchdatamanagement.techtarget.com/definition/data-modeling - What is data modeling?
https://en.wikipedia.org/wiki/Heap_(data_structure) - Heap (data structure) - Wikipedia, the free encyclopedia

Terminology

http://theothersideofcode.com/renaming-backend-frontend-to-application-ui-developers - A Proposal: Renaming Backend/Frontend to Application/UI Developers - The Other Side of Code
http://en.wikipedia.org/wiki/Declarative_programming - Declarative programming - Wikipedia, the free encyclopedia
http://blog.mediarain.com/2012/08/dynamic-data-with-mongodb-2/ - Dynamic Data with MongoDB | Rain Blog
http://en.wikipedia.org/wiki/Reactive_programming - Reactive programming - Wikipedia, the free encyclopedia

Javascript Resources

Great Resources

http://addyosmani.com - Learning JavaScript Design Patterns
http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf - ECMAScript Online PDF
http://javascript.info - The Javascript Tutorial
https://www.owasp.org - OWASP
https://developer.mozilla.org - MDN

JQuery Mobile

http://www.webdesignerdepot.com/2011/05/10-handy-jquery-mobile-tips-and-snippets-to-get-you-started - 10 handy jQuery mobile tips and snippets to get you started | Webdesigner Depot
http://jquerymobile.com/test/docs/buttons/buttons-icons.html - jQuery Mobile Docs - Button icons
http://jquerymobile.com/demos/1.1.0/docs/forms/forms-all-mini.html - jQuery Mobile Docs - Gallery of Form Controls
http://jquerymobile.com/test/docs/forms/forms-all-native.html - jQuery Mobile Docs - Native Form Controls
http://jquerymobile.com/test/docs/pages/page-scripting.html - jQuery Mobile Docs - Scripting pages
http://jquerymobile.com/branches/popup-widget/docs/toolbars/footer-persist-a.html - jQuery Mobile Framework - Persistent footer A
http://jqueryui.com/demos/autocomplete/#combobox - jQuery UI - Autocomplete Demos &amp; Documentation
http://stackoverflow.com/questions/728260/html-minification - minify - Html minification?
http://css-tricks.com/examples/ResponsiveTables/responsive.php - Responsive Table
http://www.htmlgoodies.com/beyond/reference/article.php/3472881/So-You-Dont-Want-To-Cache-Huh.htm - So, You Don't Want To Cache, Huh?
http://webdevrefinery.com/forums/topic/10245-nodejs-or-ajax-for-contact/ - Node.js or ajax for contact - webdevRefinery Forum

Patterns and Concepts Help

https://gist.github.com/miku/741526 - Linear Time Example
https://gist.github.com/p0rsche/2763377 - jQuery.extend implementation in pure JS
http://www.jspatterns.com/ - JSPatterns.com
http://davidwalsh.name/javascript-clone - Clone JavaScript Arrays and Objects
http://www.phpied.com/3-ways-to-define-a-javascript-class/">3 ways to define a JavaScript class / Stoyan&#39;s phpied.com
http://www.dreamincode.net/forums/topic/280783-how-can-i-get-inheritance-with-my-object-literal-notation/ - How Can I Get Inheritance With My Object Literal Notation - JavaScript | Dream.In.Code
http://stackoverflow.com/questions/728360/most-elegant-way-to-clone-a-javascript-object - Most elegant way to clone a JavaScript object
http://stackoverflow.com/questions/3774008/cloning-a-javascript-object - Cloning a JavaScript object?
http://stackoverflow.com/questions/10711064/javascript-object-literal-reference-in-own-keys-function-instead-of-this - Javascript: Object Literal reference in own key&#39;s function instead of &#39;this&#39;
http://www.htmlgoodies.com/beyond/javascript/object.create-the-new-way-to-create-objects-in-javascript.html - Object.create(): the New Way to Create Objects in JavaScript
http://dojotoolkit.org/reference-guide/1.7/dojo/clone.html#id3 - dojo.clone — The Dojo Toolkit - Reference Guide
http://stackoverflow.com/questions/11193421/stack-vs-heap-in-javascript - memory management - Stack vs. Heap in Javascript?
http://sporto.github.io/blog/2012/12/09/callbacks-listeners-promises/ - Asynchronous JS: Callbacks, Listeners, Control Flow Libs and Promises - Sebastian&#39;s Blog
http://stackoverflow.com/questions/18096715/implement-deferred-object-without-using-jquery - javascript - Implement Deferred object without using jquery
http://www.html5rocks.com/en/tutorials/async/deferred/ - Asynchronous JavaScript Programming. The Power Of $.Deferred for HTML5 Application - HTML5 Rocks

http://eleventyone.done.hu/OReilly.JavaScript.The.Good.Parts.May.2008.pdf - eleventyone.done.hu/OReilly.JavaScript.The.Good.Parts.May.2008.pdf
http://googlecode.blogspot.com/2009/03/doug-crockford-javascript-good-parts.html - Doug Crockford: JavaScript: The Good Parts - The official Google Code blog
http://www.quora.com/JavaScript/How-do-you-make-an-async-function-synchronous - JavaScript: How do you make an async function synchronous? - Quora
http://stackoverflow.com/questions/4458712/custom-events-observer-pattern - javascript - Custom events (observer pattern)
http://docs.oracle.com/cd/B10501_01/workflow.920/a95265/wfapi11.htm - Synchronous, Asynchronous, and Forced Synchronous Processes (Oracle Workflow Help)

iPhone Development

Great Community Resources

http://www.raywenderlich.com - Ray Wenderlich
http://www.bignerdranch.com - Big Nerd Ranch
http://www.cocoacontrols.com - CocoaControls
http://www.cimgf.com/ - Cocoa Is My Girlfriend | Taglines are for Windows programmers
http://www.cocoawithlove.com/ - Cocoa with Love
http://www.iphonedevsdk.com - iPhone Dev SDK
http://maniacdev.com - ManiacDev
http://mobile.tutsplus.com - MobileTutsPlus
http://www.iosdevnotes.com - iOS Dev Notes
http://www.freelancemadscience.com/ - Freelance Mad Science Labs - Blog
http://www.appcoda.com - appcoda
http://iosboilerplate.com/ - iOS Boilerplate - A base template for iOS apps
http://www.techotopia.com - Techotopia
http://sensiblecocoa.com - Sensible Cocoa
http://www.stanford.edu - Stanford University
http://useyourloaf.com - Use Your Loaf
http://www.wannabegeek.com - wannabegeek
http://ashfurrow.com/ - Ash Furrow

Core Data

http://adevelopingstory.com/ - A Developing Story
http://www.slideshare.net/Inferis/adventures-in-multithreaded-core-data - Adventures in Multithreaded Core Data
http://inferis.org/">@Inferis
http://lukeredpath.co.uk/blog/a-note-on-objective-c-singletons.html - A note on Objective-C singletons - Luke Redpath
http://blog.ablfx.com/2012/03/using-coredata-magicalrecord/ - Ablfx | no limits to creativity
http://krisallenfields.tumblr.com/post/30508162711/core-data - Adventures into iOS: the App Academy Experience • Day 37 - Core Data
http://oleb.net/blog/ - Blog Archive – Ole Begemann
http://cutecoder.org/featured/asynchronous-core-data-document/ - Bringing Asynchronous Core Data documents to OS X | Making the Switch
http://cocoadevcentral.com/articles/000085.php - Cocoa Dev Central: Build a Core Data Application
http://richardwarrender.com/category/programming/core-data/ - Core Data - Richard Warrender

http://www.scoop.it/t/core-data - Core data | Scoop.it
http://www.tumblr.com/tagged/core-data - core data | Tumblr
http://maybelost.com/2010/12/core-data-helper-revised/ - Core Data helper – REVISED | MaybeLost.com
http://blog.zssz.me/2010/01/asynchronous-fetch-in-core-data.html - blog.zssz.me: Asynchronous fetch in Core Data
http://9elements.com/io/index.php/customizing-core-data-migrations/ - Customizing Core Data Migrations « IO 9elements

Navigation, Segue, Unwind, etc.

http://sketchytech.blogspot.com/2012/09/container-view-controllers-parent-view.html - Container View Controllers, Parent View Controllers and Child View Controllers in Xcode (iOS 5 onwards) | sketchyTech
http://stackoverflow.com/questions/12509422/how-to-perform-unwind-segue-programmatically - ios - How to perform Unwind segue programmatically?
http://stackoverflow.com/questions/12561735/what-are-unwind-segues-for-and-how-to-use-them - ios - What are Unwind segues for and how to use them?
http://stackoverflow.com/questions/12416050/xcode-4-5-storyboard-exit - iphone - Xcode 4.5 Storyboard &#39;Exit&#39;
http://www.techotopia.com/index.php/Using_Xcode_Storyboarding_(iOS_6) - Using Xcode Storyboarding (iOS 6) - Techotopia

NSNumber and Literals

http://useyourloaf.com/blog/2012/07/09/formatters-and-locale-changes.html - Formatters and Locale Changes - Use Your Loaf
http://mobile.tutsplus.com/tutorials/iphone/objective-c-literals/ - Objective-C Literals
http://useyourloaf.com/blog/2012/06/14/using-number-formatters.html - Using Number Formatters - Use Your Loaf

Scroll View Scrolling

Themes - Customizing UIView and UIKit
http://mobile.tutsplus.com/tutorials/iphone/ios-sdk-uikit-theme-customization/ - iOS SDK: UIKit Theme Customization

UICollectionView (and CoreData)

http://stackoverflow.com/questions/13433634/uicollection-view-not-redrawing-gradient-buttons - ios - UICollection view not redrawing gradient buttons
http://www.appcoda.com/ios-programming-uicollectionview-tutorial/ - iOS Programming Tutorial: Create Grid Layout Using UICollectionView
http://www.techotopia.com/index.php/Subclassing_and_Extending_the_iOS_6_Collection_View_Flow_Layout - Subclassing and Extending the iOS 6 Collection View Flow Layout - Techotopia

UITableView

http://www.therefinedgeek.com.au/index.php/tag/abtableviewcell/ - Abtableviewcell | The Refined Geek
http://www.fieryrobot.com/blog/2008/10/01/glassy-scrolling-with-uitableview/ - Glassy Scrolling with UITableView
http://www.therefinedgeek.com.au/index.php/tag/ios/ - Ios | The Refined Geek

http://mobile.tutsplus.com/tutorials/iphone/uitableview-2/ - iPhone SDK: Working with the UITableView Class – Part 2
http://media.pragprog.com/titles/cdirec/table.pdf - media.pragprog.com/titles/cdirec/table.pdf
http://stackoverflow.com/questions/8839817/iphone-keyboard-toolbar-class - objective c - iPhone keyboard toolbar class
http://forums.macrumors.com/showthread.php?t=1333756 - TableView scroll is choppy - MacRumors Forums

Multitasking (Background Threads, GCD, etc)

http://mobile.tutsplus.com/tutorials/iphone/ios-multitasking-background-tasks/ - iOS Multitasking: Background Tasks
http://krisallenfields.tumblr.com/post/30441865741/grand-central-dispatch - Adventures into iOS: the App Academy Experience • Day 36 - Grand Central Dispatch
http://www.fizzymouse.com/forge/2012/5/10/grand-central-dispatch.html - R&amp;D - Forge - Grand Central Dispatch
http://stackoverflow.com/questions/6641540/xcode-4-how-to-profile-memory-usage-performance-with-instruments - Xcode 4: How to profile memory usage &amp; performance with Instruments?
http://stackoverflow.com/questions/8795000/live-bytes-vs-real-memory-in-activity-monitor-on-ios/8797272#8797272 - objective c - Live Bytes vs Real Memory in Activity Monitor on iOS
http://maniacdev.com/2012/01/open-source-library-for-easy-icloud-image-and-document-transfers-with-a-blocks-based-syntax/ - Open Source: Library For Easy iCloud Image And Document Transfers With A Blocks Based Syntax
http://blog.slaunchaman.com/2011/02/28/cocoa-touch-tutorial-using-grand-central-dispatch-for-asynchronous-table-view-cells/ - Cocoa Touch Tutorial: Using Grand Central Dispatch for Asynchronous Table View Cells | Jeff Kelley's Blog
http://blog.sallarp.com/iphone-ipad-wait-for-asynchronous-nsrunloop-task-to-complete/ - iPhone/iPad – Wait for asynchronous tasks to complete | blog.sallarp.com

Optimization Tips

http://nathanbarry.com/designing-buttons-ios5/ - Designing Buttons in iOS 5 | Nathan Barry
http://robsprogramknowledge.blogspot.pt/2012/01/uistoryboard-best-practices.html - Programming Knowledge Base: UIStoryboard Best Practices
http://robsprogramknowledge.blogspot.com/2012/06/linking-storyboard.html - Programming Knowledge Base: Linking Storyboards
http://www.blu-ray.com/movies/top.php?country=us&page=23 - Blu-ray.com - Top Blu-ray Sellers
http://www.icodeblog.com/2011/12/06/using-xcode-4-snippets/ - Using XCode 4 Snippets | iPhone Programming Tutorials
http://useyourloaf.com/blog/2012/06/21/storyboard-segues.html - Storyboard Segues - Use Your Loaf
http://maniacdev.com/ - iOS App Dev Libraries, Controls, Tutorials, Examples and Tools
http://www.fizzymouse.com/forge/2012/10/4/comparing-objects.html - R&amp;D - Forge - Comparing Objects
http://google-styleguide.googlecode.com/svn/trunk/objcguide.xml - Google Objective-C Style Guide
http://clang.llvm.org/docs/AutomaticReferenceCounting.html - Objective-C Automatic Reference Counting (ARC)

Java

EclipseWTP - Web Development

http://www.vogella.com/articles/EclipseWTP/article.html - Servlet and JSP development with Eclipse WTP
http://community.java.net/community/jcp?force=358 - The Java Community Process(SM) Program | Java.net
http://wolfpaulus.com/journal/mac/tomcat7 - Installing Tomcat 7.0.x on OS X | Wolf Paulus
http://www.vogella.com/articles/Eclipse/article.html#updatemanager - Eclipse IDE Tutorial
http://www.vogella.com/articles/JavaWebTerminology/article.html - Introduction into Java Web development

Maven

http://www.sonatype.com/books/mvnref-book/reference/installation-sect-maven-install.html - Maven: The Complete Reference: 2.3. Installing Maven / Documentation Sonatype.com
http://www.javacodegeeks.com/2011/04/java-7-mac-os-x-status.html - Official Java 7 for Mac OS X – Status | Java Code Geeks
http://www.javacodegeeks.com/2011/03/agile-software-development.html - Agile software development recommendations for users and new adopters | Java Code Geeks
http://www.javacodegeeks.com/2011/03/save-money-from-agile-development.html - Save money from Agile Development | Java Code Geeks
http://www.javacodegeeks.com/2011/02/9-tips-on-surviving-wild-west.html">9 Tips on Surviving the Wild West Development Process | Java Code Geeks
http://www.javacodegeeks.com/2010/12/things-every-programmer-should-know.html - Things Every Programmer Should Know | Java Code Geeks

REST with Jersey

http://www.vogella.com/articles/JavaXML/article.html - Java and XML - Tutorial
http://www.vogella.com/articles/JAXB/article.html - JAXB Tutorial
http://www.vogella.com/articles/REST/article.html - REST with Java (JAX-RS) using Jersey - Tutorial
http://jersey.java.net/nonav/documentation/latest/chapter_deps.html - Chapter 11. Dependencies
http://www.vogella.com/articles/ApacheHttpClient/article.html - Apache HttpClient - Tutorial
https://groups.google.com/forum/?fromgroups=#!topic/vogella/4RSupINCeAc - Cant access filecounter (HTTP 404) - Google Groups
http://stackoverflow.com/questions/7240039/eclipse-tomcat-404-error - java - Eclipse, tomcat, 404 error

Tomcat

http://www.vogella.com/articles/ApacheTomcat/article.html - Apache Tomcat - Tutorial
http://stackoverflow.com/questions/1577014/how-install-a-specific-jdk-on-mac-os-x - java - How install a specific jdk on Mac OS X?
http://maven.apache.org/download.cgi - Maven - Download Apache Maven
http://nfarina.com/post/8239634061/ios-to-android - Nick Farina - An iOS Developer Takes on Android
http://niklasschlimm.blogspot.com/2011/06/setting-up-java-ee-6-development.html - Niklas&#39; Blog: Setting up a Java EE 6 development environment

REST and SOAP

http://predic8.com/rest-webservices.htm - Introduction into REST Web Services

http://stackoverflow.com/questions/1237649/json-or-soap-xml - iphone - JSON or SOAP (XML)?
http://stackoverflow.com/questions/4573305/rest-api-why-use-put-delete-post-get - php - REST API - why use PUT DELETE POST GET?
http://en.wikipedia.org/wiki/REST - Representational state transfer - Wikipedia, the free encyclopedia
http://stackoverflow.com/questions/209905/rest-and-soap - REST and SOAP
http://weblogs.asp.net/mikeiskiw/archive/2009/03/26/rest-vs-soap-feeling-restful.aspx - REST vs. SOAP - Feeling RESTful - Mike Iskiw
http://geeknizer.com/rest-vs-soap-using-http-choosing-the-right-webservice-protocol/ - REST vs. SOAP – The Right WebService
https://devcentral.f5.com/weblogs/macvittie/archive/2008/12/05/soap-vs-rest-the-war-between-simplicity-and-standards.aspx - SOAP vs REST: The war between simplicity and standards
http://msdn.microsoft.com/en-us/magazine/dd942839.aspx - SOAP, REST, and More
http://stackoverflow.com/questions/1649793/why-do-people-want-to-deliver-both-json-and-xml-as-output-to-their-rest-interfac - web services - Why do people want to deliver both Json and XML as output to their REST interfaces?
http://tldp.org/HOWTO/XML-RPC-HOWTO/xmlrpc-howto-intro.html - What is XML-RPC?
http://kbeezie.com/view/apache-with-nginx/ - Apache and Nginx Together » KBeezie
http://interfacelab.com/nginx-php-fpm-apc-awesome/ - NGINX + PHP-FPM + APC = Awesome
http://till.klampaeckel.de/blog/archives/148-nginx-configuration-gotchas.html - nginx configuration gotchas - till&#39;s blog
http://desmondbrand.com/blog/2011/12/05/using-nginx-as-a-reverse-proxy-for-speedy-app-engine-development/ - Using nginx as a reverse proxy for speedy App Engine development - Desmond Brand