Brain Teaser Answers

1. *You have 20 bottles of pills. 19 bottles have 1.0-gram pills, but one has pills of weight 1.1 grams. Given a scale that provides an exact measurement, how would you find the heavy bottle? You can only use the scale once.*

The constraint in this question is the clue. Because we can only use the scale once, we know something interesting: we must weigh pills from at least 19 bottles at the same time. Otherwise, if we skipped two or more bottles entirely, how could we distinguish between those missed bottles?

If we took one pill from Bottle #1 and two pills from Bottle #2, what the scale would show depends on which bottle is the heavy one. If the pills in Bottle #1 were heavy, we would get 3.1 grams. If Bottle #2 had the heavy pills, we would get 3.2 grams. By doing this we get the solution to the problem.

We know the expected weight of a bunch of pills. If we select a different number of pills from each bottle the difference between the expected weight and the actual weight will indicate which bottle contributed the heavier pills.

We can generalize this to the full solution: take one pill from Bottle #1, two pills from Bottle #2, three pills from Bottle #3, and so on. Weigh this mix of pills. If all pills were one gram each, the scale would read 210 grams (1 + 2+...+20 = 20* 21/2 = 210). Any excess of this amount must come from the extra 0.1-gram pills. This formula will tell you the bottle number: (weight - 210 grams) / 0.1 grams. So, if the set of pills weighed 211.3 grams, then Bottle #13 would have the heavy pills.

2. *There is an 8x8 chessboard in which two diagonally opposite corners have been cut off. You are given 31 dominos, and a single domino can cover exactly two squares. Can you use the 31 dominos to cover the entire board? Prove your answer by providing an example or showing why it's impossible.*

This board is 8x8, which has 64 squares, less the two that have been cut off (62). A set of 31 dominoes should be able to cover the board (31 x 2 = 62). If we lay the dominos down on a row that has one square missing, the last domino must extend to a different row. Because of this, each row we lay down will always have a domino that will extend to another row. This will always be the case, and we'll never be able to cover the entire board correctly. An easier way to solve this is by understanding that a chessboard has 32 black squares and 32 white squares and by removing opposite corners (which must be the same color), we're left with 30 of one color and 32 of the other. Each domino we lay on the board will take up one black square and one white square. That is to say, 31

dominos will take up 31 white squares and 31 black squares exactly. On this board, we have 30 of one color and 32 of another so doing this is impossible.

3. *You have a five-quart jug, a three-quart jug, and an unlimited supply of water (but no measuring cups). How would you come up with exactly four quarts of water? Note that the jugs are oddly shaped, such that filling up exactly 0.5 of a jug would be impossible.*

It's easiest to solve this by viewing the table. Keep in mind that this question has a foundation in math. As long as two jug sizes are relatively prime (have no common prime factors), you can find a pour sequence for any value between one and the sum of the jug sizes.

5-Quart	3-Quart	Note
5	0	Filled 5-quart jug.
2	3	Filled 3-quart with 5-quart's contents.
2	0	Dumped 3-quart.
0	2	Filled 3-quart with 5-quart's contents
5	2	Filled 5-quart.
4	3	Filled remainder of 3-quart with 5-quart. Done.

4. *There's a bunch of people living on an island, a visitor comes with a strange order: all blue-eyed people must leave the island as soon as possible. There will be a flight out at 8:00pm every evening. Each person can see everyone else's eye color, but they do not know his or her own and no one is allowed to ask. Additionally, they do not know how many people have blue eyes, although they do know that at least one person does. How many days will it take the blue-eyed people to leave?*

This is a classic "build from base case" question. Assume that there are n people on the island and c of them have blue eyes. We are explicitly told that $c > 0$.

Case c = 1: Exactly one person has blue eyes

Assuming all the people are intelligent, the blue-eyed person should look around and realize that no one else has blue eyes. Since he knows that at least one person has blue eyes, he must conclude that it is him. Therefore, he would take the flight that evening.

Case c = 2: Exactly two people have blue eyes.

The two blue-eyed people see each other, but be unsure whether c is 1 or 2.They know, from the previous case, that if c = 1, the blue-eyed person would leave on the first night. Therefore, if the other blue-eyed person is still there, he must deduce that c = 2, which means he himself has blue eyes. Both men would then leave on the second night.

Case c > 2: The General Case.

As we increase c, we can see that this logic continues to apply. If c = 3, then those three people will immediately know that there are either 2 or 3 people with blue eyes. If there were two people, then those two people would have left on the second night. So, when the others are still around after that night, each person would conclude that c = 3 and that they, therefore, have blue eyes too. They would leave that night. This same pattern extends up through any value of c. Therefore, if c men have blue eyes, it will take c nights for the blue-eyed men to leave. All will leave on the same night.

5. *There is a building of 100 floors. If an egg drops from the Nth floor or above, it will break. If it's dropped from any floor below, it will not break. You're given two eggs. Find N, while minimizing the number of drops for the worst case.*

This is a worst-case balancing question. Once we find the worst possible case, the answer is simple. Once we drop Egg 1, the Egg 2 must go from lowest to highest (or linear search) to find the next highest floor that doesn't cause the egg to break. As an example, if Egg 1 is dropped from floors 5 and 10 without breaking, but it breaks when it's dropped from floor 15, then Egg 2 must be dropped, in the worst case, from floors 11, 12, 13 and finally 14.

Suppose we drop an egg from the 10^{th} floor, then the 20^{th}

- If Egg 1 breaks on the first drop (floor 10), then we have at most 10 drops total.
- If Egg 1 breaks on the last drop (floor 100), then we have at most 19 drops total (floors 10, 20, ...,90,100, then 91 through 99).

This is the absolute worst case. By performing some load balancing we can make these two cases more even. Our goal is to create a system for dropping Egg 1 such that the number of drops is as consistent as possible, whether Egg 1 breaks on the first drop or the last drop.

1. A perfectly load-balanced system would be one in which Drops(Egg 1) + Drops(Egg 2) is always the same, regardless of where Egg 1 breaks.
2. For that to be the case, since each drop of Egg 1 takes one more step, Egg 2 is allowed one fewer step.

3. We must, therefore, reduce the number of steps potentially required by Egg 2 by one drop each time. For example, if Egg 1 is dropped on floor 20 and then floor 30, Egg 2 is potentially required to take 9 steps. When we drop Egg 1 again, we must reduce potential Egg 2 steps to only 8. That is, we must drop Egg 1 at floor 39.
4. We know Egg 1 must start at floor X, then go up by X - 1 floors, then X - 2 until it gets to 100.
5. Solve for X in: X + (X – 1) + (X – 2) + … + 1 = 100
   X (X + 1) / 2 = 100 -> X = 14.

We go to floor 14, then 27 and then 39. This takes 14 steps in the worst case.

6. *There are 100 closed lockers in a hallway. A man begins by opening all 100 lockers. Next, he closes every second locker. Then, on his third pass, he toggles every third locker (closes it if it is open or opens it if it is closed). This process continues for 100 passes, such that on each pass `i`, the man toggles every `i`th locker. After his 100th pass in the hallway, in which he toggles only locker #100, how many lockers are open?*

We can tackle this problem by thinking through what it means for a door to be toggled. This will help us deduce which doors at the very end will be left opened.

Question: For which rounds is a door toggled (open or closed)?
A door n is toggled once for each factor of n, including itself and 1. That is, door 15 is toggled on rounds 1,3,5, and 15.

Question: When would a door be left open?
A door is left open if the number of factors (which we will call x) is odd. You can think about this by pairing factors off as an open and a close. If there's one remaining, the door will be open.

Question: When would x be odd?
The value x is odd if n is a perfect square. Here's why: pair n's factors by their complements. For example, if n is 36, the factors are (1,36), (2,18), (3,12), (4,9), (6,6). Note that {6,6) only contributes one factor, thus giving n an odd number of factors.

Question: How many perfect squares are there?
There are 10 perfect squares. You could count them (1, 4, 9, 16, 25, 36, 49, 64, 81, 100), or you could simply realize that you can take the numbers 1 through 10 and square them: 1*1, 2*2, 3*3, ..., 10*10. Therefore, there are 10 lockers open at the end of this process.

Object Oriented Design Solutions

1. *Design the data structures for a generic deck of cards. Explain how you would subclass the data structures to implement blackjack.*

A possible solution for this problem can be found at Learn Ruby (http://learnruby.com/examples/ruby-quiz-151.shtml).

2. *Imagine you have a call center with three levels of employees: respondent, manager, and director. An incoming telephone call must be first allocated to a respondent who is free. If the respondent can't handle the call, he or she must escalate the call to a manager. If the manager is not free or not able to handle it, then the call should be escalated to a director. Design the classes and data structures for this problem. Implement a method* `dispatchCall()` *which assigns a call to the first available employee.*

The important thing to remember when answering this question is that all three ranks of employees have different roles and functions associated with them. There are several things that they have in common with one another – they all have a name, an address, a job title and an age. Good object-oriented design would keep these things in one class and extend them to the other classes.

There should also be a `CallHandler` class, which routes the calls to the correct employee. Keep in mind that with any question related to OOD there are multiple ways to solve the problem, each with their own trade-offs. It's a good practice to design for long-term code flexibility and maintenance.

`CallHandler` should be implemented as a singleton class. It represents the body of the program and all the calls are funneled through it first.

`Call` represents a call from a user. A call has a minimum rank and is assigned to the first employee who can handle it.

`Employee` is a super class for the `Director`, `Manager`, and `Respondent` classes. It is implemented as an abstract class since there should be no reason to instantiate an `Employee` type directly.

The `Director`, `Manager` and `Respondent` classes are extensions of the `Employee` class.

3. *Design a musical jukebox using object-oriented principles.*

Normally we suggest that you ask your interviewer additional questions to narrow down the specific details of this question, but in the event that you're unable to you have to rely on assumptions. For this question, we assume that the Jukebox is a computer simulation of a physical jukebox, but without the need for coin insertion. A basic outline of the system components is as follows:

- Jukebox
- CD
- Song
- Artist
- Playlist
- Display Screen

The next step is to think about the possible actions of this application:

- Playlist creation (including add, delete and shuffle)
- CD selector
- Song selector
- Queuing up a song
- Get next song from playlist

By introducing a user we add the following functions:

- Adding
- Deleting
- Credit information

Each of these components translates roughly to an object, and each action translates to a method.

The `Jukebox` class represents the body of the solution. Most of the interactions between the components of the system, or between the system and the user, are channeled through here.

Similar to a real CD player, the `CDPlayer` class supports storing just one CD at a time. The CDs that are not in play are stored in the jukebox.

The `Playlist` manages the current and the next songs to play. It is essentially a queue.

There are also classes for a `CD`, `Song` and `User` are all basic, they consist mainly of member variables and getters and setters.

4. *Design a parking lot using object-oriented principles.*

This question is vague, so again we are required to make some assumptions, shown below:

- The parking lot has multiple levels. Each level has multiple rows of spots.
- The parking lot can park motorcycles, cars and buses.
- The parking lot has motorcycle spots, compact spots and large spots.
- A motorcycle can park in any spot.
- A car can park in either a single compact spot or a single large spot.
- A bus can park in five large spots that are consecutive and within the same row. It cannot park in small spots.

You should create an abstract class `Vehicle`, from which `Car`, `Bus`, and `Motorcycle` all inherit. To handle different parking spot sizes, create just one class `ParkingSpot` that has a member variable indicating size.

We then create a `ParkingLot` class, which is a wrapper class for an array of Levels. The reason for implementing it this way is to separate out logic that deals with finding free spots and parking cars out from the broader actions of the `ParkingLot`. Not doing this would require us to make a double array or hash table, which complicates things.

A `ParkingSpot` is implemented by having just a variable that represents the size of the spot.

5. *Design the data structures for an online book reader system.*

Start by assuming we want to design a basic online reading system that provides the following functionality:

- User membership creation and extension.
- Searching the database of books.
- Reading a book.
- Only one active user at a time
- Only one active book by this user.

To implement these operations we may require many other functions, like `get`, `set`, `update`, etc. The objects required would likely include `User`, `Book`, and `Library`.

The class `OnlineReaderSystem` represents the body of our program. We could implement the class such that it stores information about all the books, deals with user management, and refreshes the display, but that would make this class rather hefty. Instead, we've chosen to tear off these components into `Library`, `UserManager`, and

`Display` classes. The classes for `User` and `Book` simply hold data and provide minimal actual functionality.

The decision to tear off user management, library, and display into their own classes, when this functionality could have been in the general `OnlineReaderSystem` class, is for the purpose of keeping the code maintainable as it becomes more complex and additional functionality is added.

6. *Implement a jigsaw puzzle. Design the data structures and explain an algorithm to solve the puzzle. You can assume that you have a `fitsWith` method which, when passed two puzzle pieces, returns true if the two pieces belong together.*

We assume that this is a traditional, simple jigsaw puzzle. The puzzle is grid-like with rows and columns. Each piece is located in a single row and column and has four edges. Each edge comes in one of three types: inner, outer and flat. For example a corner piece will have two flat edges and two other edges (either outer or inner).

As we piece together the jigsaw puzzle we need to think about the position of each piece. It helps to think of the position as either absolute or relative:

- Absolute Position: "This piece is located at position (12,23)." Absolute position would belong to the Piece class itself and would include an orientation as well.
- *Relative Position:* "I don't know where this piece is actually located, but I know that it is next to this other piece.' The relative position would belong to the Edge class.

The algorithm for solving the puzzle should start with the corners and edge pieces. Search through all the pieces and find just the edges, then group these pieces by their edge type.

We now have a quicker way to zero in on potential matches for any given edge. We then go through the puzzle, line by line, to match pieces.

The solve method, implemented below, operates by picking an arbitrary corner to start with. It then finds an open edge on the corner and tries to match it to an open piece. When it finds a match, it does the following:

1. Attaches the edge
2. Removes the edge from the list of open edges.
3. Finds the next open edge.

The next open edge is defined to be the one directly opposite the current edge, if it is available. If it is not available, then the next edge can be any other edge. This will cause the puzzle to be solved from the outside to the inside.

The spiral comes from the fact that the algorithm always moves in a straight line, whenever possible. When we reach the end of the first edge, the algorithm moves to the only available edge on that corner piece—a 90-degree turn. It continues to take 90-degree turns at the end of each side until the entire outer edge of the puzzle is completed. When that last edge piece is in place, that piece only has one exposed edge remaining, which is again a 90-degree turn. The algorithm repeats itself for subsequent rings around the puzzle, until finally all the pieces are in place.

In an interview, even a computer science graduate wouldn't be expected to write out all of this code – just pseudocode to form a structure is fine.

7. *Explain how you would design a chat server. In particular, provide details about the various backend components, classes, and methods. What would be the hardest problems to solve?*

As a candidate your job is to focus on an aspect of the problem that is reasonably broad, but focused enough that you could accomplish it during an interview. It need not match real life exactly, but it should be a fair representation of an actual implementation.

In the case of our example we focus on core user management and conversation aspects: adding a user, creating a conversation, updating one's status, etc. Assume that "friending" is mutual and both parties need to add each other as a contact. The chat system will support both group chat and private chats.

Some of the actions that require implementation are as follows:

- Signing online and offline.
- Add requests (sending, accepting, and rejecting).
- Updating a status message.
- Creating private and group chats.
- Adding new messages to private and group chats.

Based on these requirements we must have a concept of users, add request status, online status, and messages.

The core components of the system would likely consist of a database, a set of clients, and a set of servers. These don't really factor into the object-oriented design nature of this question but it's still good to mention them during an interview. A SQL database will be required (or some other form of database), as will servers and significant network infrastructure.

Strings and Arrays Solutions

The solutions for these problems are available in the Strings & Arrays subfolder. These solutions are written in Ruby.

1. Create an algorithm to determine if a string has all unique characters. What if you cannot use additional data structures?
2. Given two strings, write a method to decide if one is a permutation of the other.
3. Write a method to replace all spaces in a string with '%20'. For example, an input of "`Bitmaker Labs Summer 2013`" would produce an output of "`Bitmaker%20Labs%20Summer%202013`".
4. Implement a method to perform basic string compression using the counts of consecutive repeated characters. For example, the string "`abbcccccaa`" would become "`a1b2c5a2`". If the compressed string isn't shorter than the original string then your method should return the original string.
5. Given an image represented by an NxN matrix, where each pixel in the image is 4 bytes, write a method to rotate the image by 90 degrees. Can you do this in place?
6. Write an algorithm such that if an element in an MxN matrix is 0, its entire row and column are set to 0.
7. Assume you have a method `isSubstring` which checks if one word is a substring of another. Given two strings, s1 and s2, write code to check if s2 is a rotation of s1 using only one call to `isSubstring` (e.g. "waterbottle" is a rotation of "erbottlewat".

Sorting and Searching Interview Questions

The response code for these questions is provided in DART (http://www.dartlang.org/), a programming language created by Google intended to be used instead of JavaScript. Code is available in the Sorting and Searching section of the answers file. Although you have not been formally taught DART, many of the elements of it are similar to JavaScript. A good programmer should be able to solve problems in multiple languages.

1. *Write a method to sort an array of strings so that all the anagrams are next to each other.*

One way to solve this problem is to apply any standard algorithm, like merge sort or quick sort, and adjust the comparator. This comparator will be used to indicate that two strings, which are anagrams of each other, are the same. There are multiple ways of doing this, one is to count the occurrences of each distinct character in a string and return true if they match. Another option is to sort the string; in the event that the two words are anagrams the sorted versions of them will be equivalent.

2. *Imagine you have a 20 GB file with one string per line. Explain how you would sort that file.*

When an interviewer gives a size limit of 20 gigabytes, it implies that they don't want the data to be stored in memory. That being said, you can still keep part of the data in memory. The key is to divide the data into chunks which are x megabytes each, where x is the amount of memory we have available. Each chunk is sorted separately and then saved back to the file system. Once all the chunks are sorted, merge the chunks one by one. By the end we have a fully sorted file, this method is known as an external sort.

3. *Given an M x N matrix in which each row and each column is sorted in ascending order, write a method to find an element.*

One approach is to attempt to do a binary search on every row to find the element. This algorithm will be O(M log(N)), since there are M rows and it takes O(log(n)) time to search each one. This approach isn't ideal, but it's still good to mention before you proceed to a better algorithm. The below example illustrates this point.

15	20	40	85
20	35	80	95
30	55	95	105

40	80	100	120

If we're tasked with finding the element 55, we can start looking at the first row of a column. If the start of a column is greater than 55, we know that 55 can't be in that column, since the start of the column is always the minimum element. Additionally, we know that 55 can't be in any columns on the right, since the first element of each column must increase in size from left to right. Because of this, we know that if the start of the column is greater than the element x that we are searching for, we know that we need to move further to the left. We can follow the same logic for rows. If the end of a column or row is less than x then we know that we move down for rows and to the right for columns. This is because the end is always the maximum element. By doing this, we'll eventually find x. The rules for this are outlined below:

- If the start of a column is greater than x, then x is to the left of the column.
- If the end of a column is less than x, then x is to the right of the column.
- If the start of a row is greater than x, then x is above that row.
- If the end of a row is less than x, then x is below that row.

You can begin anywhere, but a good place to start is the first element of each column. Start at the greatest column and work left. This means that the first element for comparison is array[0][c-1], where c is the number of columns. By comparing the start of columns to x (which is 55), we'll find that x must be in columns 0, 1, or 2. We will have stopped at array[0][2]. This element may not be the end of a row in the full matrix, but it is an end of a row of a submatrix. The same conditions apply, the value at array[0][2], which is 40, is less than 55, so we know we have to move downwards. We can repeatedly apply these conditions to search for 55.

Testing Interview Questions

1. *Find the mistake(s) in the following Ruby code:*

```
function(first_name, last_name) {
  Name = first_name + last_name;
  return Name;
}
```

The major problem with the above code is that this is not the proper way to declare a function in JavaScript. The next problem is that the variables are named in a way that violates JavaScript conventions. These variables should be lowercase, and use camel-case instead of snake-case. Also, this code should be accompanied by a portion where the actual code is called – although this is not an error, it is an omission that it would be good to mention. An improved version of this code is visible below.

```
function myFunction(firstName, lastName) {
      name = firstName + " " + lastName;
      return name;
}

myFunction("John", "Doe");
```

2. *You are given the source to an application that crashes when it is run. After running it ten times in a debugger, you find it never crashes in the same place. The application is written in Ruby. What programming errors could be causing this crash? How would you test each one?*

This question is largely focused on determining your brainstorming ability and whether or not you approach the problem in a logical manner. We can give some general causes of random crashes.

- Random Variable: The application may use some random number or variable component that may not be fixed for every execution of the program. Examples include user input, a random number created in the program, or the time of day.
- Uninitialized Variable: The application could have an uninitialized variable that, in some languages, may cause it to take on an arbitrary value. The values of this variable could result in the code taking a slightly different path each time.
- Memory Leak: The program may have run out of memory. Other culprits are totally random for each run since it depends on the number of processes running at that particular time.
- External Dependencies: The program may depend on another application, machine, or resource. If there are multiple dependencies, the program could crash at any point.

To track down the issue, we should start with learning as much as possible about the

application. Who is running it? What are they doing with it? What kind of application is it? Also, although the application doesn't crash in exactly the same place, it's possible that it is linked to specific components or scenarios. It may be useful to approach this by elimination. Close down all other applications on the system and track resource use. If there are parts of the program we can disable, do so. Run it on a different machine and see if we experience the same issue. The more we can eliminate (or change), the easier we can track down the issue.

3. *How would you load a test webpage without using any test tools?*

Load testing helps to identify a web application's maximum operating capacity, as well as any bottlenecks that may interfere with its performance. To effectively perform load testing, we must first identify the scenarios and the metrics that fulfill our performance objectives. The typical performance metrics may include response time, throughput, resource utilization and maximum system load.

We then design tests to simulate the load while measuring each of these criteria. In the absence of formal testing tools, we can simulate users and scenarios relatively easily. We would then analyze the results based on the data gathered during the tests and compare it with the accepted values.

4. *How would you test a pen?*

This problem is largely about understanding the applicable constraints and approaching the problem in a structured manner. To understand the constraints, you should ask a lot of questions to understand the "who, what, where, when, how and why" of a problem.

As an example of this, we've provided a mock conversation with an interviewer below.

Interviewer: How would you test a pen?

Candidate: Let me find out a bit about the pen. Who is going to use the pen?
Interviewer: Children.

Candidate: What will they be doing with it? Will they be writing, drawing, or doing something else?
Interviewer: Drawing.

Candidate: What will they be drawing on? Paper? Clothing? Walls?
Interviewer: Clothing.

Candidate: Great. What kind of tip does the pen have? Felt? Ballpoint? Is it intended to wash off, or is it intended to be permanent?
Interviewer: It's intended to wash off.

Eventually, you'll be able to reiterate all the required information to the interviewer to confirm your understanding of the situation. Now that you understand what you're testing the best approach to take is to develop a structured plan. You should break apart the object or problem and analyze each. In this case the components might be:

- Fact check: Verify that the pen is felt tip and that the ink is one of the allowed colors.
- Intended use: Drawing. Does the pen write properly on clothing?
- Intended use: Washing. Does it wash off of clothing (even if it's been there for an extended period of time)? Does it wash off in hot, warm and cold water?
- Safety: Is the pen safe (non-toxic) for children?
- Unintended uses: How else might children use the pen? They might write on other surfaces, so you need to check whether the behavior there is correct. They might also stomp on the pen, throw it, and so on. You'll need to make sure that the pen holds up under these conditions.

Remember that in any testing question, you need to test both the intended and unintended scenarios.

5. *How would you test an ATM at a bank?*

Again, the first thing to do on this question is to clarify assumptions by asking the following questions:

- Who is going to use-the ATM? Answers might be "anyone," or it might be any number of other answers.
- What are they going to use it for? Answers might be "withdrawing money," "transferring money," "checking their balance," etc.
- What tools do we have to test? Do we have access to the code, or just to the ATM?

Once we understand the problem, we can break it down into testable components that include logging in, withdrawing money, depositing money, checking balance and transferring money. It's a best practice to include a mix of manual and automated testing.

Manual testing would involve going through the steps above, making sure to check for all the error cases (low balance, new account, nonexistent account, and so on). Automated testing involves testing specific scenarios that you could also test manually, but we'd also want to test more complex scenarios by creating mock users and accounts and performing mock transactions.

Knowledge Based Interview Questions

One of the best ways to increase your preparedness for technical interview questions is by practicing answering them by using a question bank. The above Smarterer tests are also a good source of questions to prepare with. These questions are in no particular order and vary significantly in their difficulty. Interviewers will not expect you to know the answers to all of these questions, as some of them focus on advanced concepts that are typically reserved for intermediate-level developers, but knowing how to answer them will impress your interviewers. These questions are denoted with an asterisk.

Ruby and Ruby on Rails

1. *What are Ruby gems?*

A gem is nothing more than a piece of ruby code packaged as a library so that it can be imported and used by others in their programs. A Ruby gem is therefore simply a library that is written in the ruby programming language.

You can add that you often look for ruby gems on rubygems.org. If you have downloaded any recent gems it might be a good idea to mention those. Some of the popular Ruby gems that your interviewer will likely be familiar with are:

- rails
- activerecord
- rake
- activeadmin

Finally Rubygems is the name of the project that wrote the "gem" ruby library. [You need not mention this during the interview]

2. *What's the difference between a Symbol and a String?*

Symbols and string are used interchangeably by various developers and their usage within gems can be confusing at times. You can think of Symbols as faster & immutable strings. Once a string is used up it is marked for cleaning by the garbage collector but it is not cleaned up immediately and it cannot be reused.

Symbols live for the duration of the session. You might say that this leads to increased memory usage, but by keeping the symbol alive a bit longer it can be reused again. Another big difference is that strings can be modified. They are mutable.

3. *What is the purpose of yield?*

The interpreter essentially invokes a separate piece of code and places it in the location. You might say it is similar to a method calling another method. Let's understand a little bit of background about where yield might be useful first.

The Rails framework encourages you to write code that is DRY (Don't Repeat Yourself). Developers often write common code in a central file and then they write the custom code in the specific files. Let's say you are building a web application and you want all pages to have a common header, a common footer, the same "Welcome user-name!" message. You can put all this common code in your application.html.erb file.

```
<html> .... common page title
      .. standard header...
      <body>
      ..common page title,
      <%= yield %>
      ..footer code can go here... </body>
</html>
```

The rest of the custom code can go in your specific file. Say the page you are creating is the list of articles. Then in your implementation file you would just write the code for pulling in the articles and the final page displayed to the user would be your custom code which will be placed instead of the `<%= yield %>` code in the application.html.erb file.

4. What are class variables? How do you define them?

Class variables are created using the @@ prefix to denote the variable as class level. It works just like any other variable, however in the case of inheritance it works more like a static variable that is accessed across all variable instances.

5. How do you define instance variables?

Instance variables are defined using single @ symbol.

```
@foo = "Hello"
```

Within a class they can be declared as below:

```
class Animal
 attr_accessor :name, :age
end
```

Next you can query an object instance to find which instance variables it has.

```
anim = Animal.new
anim.instance_variables
=> [ ]
anim.name="John"
anim.age = 3
=> [:@age, :@name]
```

In the above case we did not put the @ symbol before the instance variables but it is implied.

6. How do you define global variables?

Global variables are defined using single $ symbol.

```
$foo = 5
```

It can be declared anywhere and used anywhere. Generally you shouldn't declare too many global variables but sometimes it makes sense to do so. One nice feature of a global variable is that it can be used to trigger a procedure if it's value changes.

```
trace_var :$foo, proc{puts "$foo is now #{$foo}"}
```

This set the tracking and the procedure is called whenever the value of $foo changes.

```
$foo=7
$foo is now 7
 ⇨  7
```

7. Does Ruby support constructors? How are they declared?

Constructors are supported in Ruby. They are declared as the method initialize, shown below. The initialize method gets called automatically when Album.new is called.

```
class Album
  def initialize(name, artist, duration)
    @name     = name
    @artist   = artist
    @duration = duration
  end
end
```

8. *How can you dynamically define a method body?**

An instance method can be defined dynamically with

```
Module#define_method(name, body)
```

In this instance, name is the method's name given as a Symbol, and body is its body given as a Proc, Method, UnboundMethod, or block literal. This allows methods to be defined at runtime, in contrast to "def" which requires the method name and body to appear literally in the source code.

```
class Conjure
  def self.conjure(name, lamb)
```

```
      define_method(name, lamb)
    end
  end
```

Define a new instance method with a lambda as its body

```
Conjure.conjure(:glark, ->{ (3..5).to_a * 2 })
Conjure.new.glark #=> [3, 4, 5, 3, 4, 5]
```

Module#define_method is a private method so must be called from within the class the method is being defined on. Alternatively, it can be invoked inside class_eval like so:

```
Array.class_eval do
  define_method(:second, ->{ self.[](1) })
end
[3, 4, 5].second #=> 4
```

Kernel#define_singleton_method is called with the same arguments as Module#define_method to define a singleton method on the receiver.

```
File.define_singleton_method(:match) do |file, pattern|
  File.read(file).match(pattern)
end
File.match('/etc/passwd',/root/) #=> #<MatchData "root">
```

9. *What is a Range?*

Range is a great way to declare continuous variables. You should use it to declare arrays and other types of collections.

```
range1 = (1..4).to_a
 => [1, 2, 3, 4]
puts range1
1
2
3
4
```

You can also create strings in this format and it fills in the interim values automatically.

```
range2 = ('bar'..'bat').to_a
puts range2
bar
bas
bat
```

Since the end result of using range is an array you can also iterate over it just like any other array.

```
range2.each do |str|
   puts "In Loop #{str}"
end
```

This produces the result as shown below:

```
In Loop bar
In Loop bas
In Loop bat
```

*10. How can you implement method overloading?**

This one's a tricky question. If you have a background in Java then you must know that method overloading is simply multiple methods with same name but different signatures/parameters. In the case of Ruby method overloading is not supported. However, it does support the overall goal of passing variable number of parameters to the same method. You would implement it like this:

```
class MyClass
  def initialize(*args)
    if args.size < 2  || args.size > 3
      puts 'This method takes either 2 or 3 arguments'
    else
      if args.size == 2
        puts 'Found two arguments'
      else
        puts 'Found three arguments'
      end
    end
  end
end
```

The output can be seen here:

```
MyClass.new([10, 23], 4, 10)
Found three arguments

MyClass.new([10, 23], [14, 13])
Found two arguments
```

By doing this you can get the same effect as method overloading but you just have to manage the number of variables inside your method itself.

11. What is the difference between '&&', 'and' and '&' operators?

The '&&' and 'and' are both logical and statements. They '&&' operator has higher precedence though. Here's an example of illustrate this in more detail:

```
foo = 3
```

```
bar = nil
a = foo and bar
# => nil
a
# => 3
a = foo && bar
# => nil
a
# => nil
```

Notice how the statement `'a = foo and bar'` actually behaves like `'(a = foo) and bar'`

12. How can you create setter and getter methods in Ruby?

The setter and getter methods can be created manually by the developer or it can be auto-generated by Ruby using the attr_accessor method specifier.

```
class Animal
   attr_accessor :name, :age
end
anim = Animal.new
 => #<Animal:0x007f8ea20841d8>
anim.age = 3
 => 3
anim.name = "Steve"
 => "Steve"
puts anim.name, anim.age
Steve
3
```

Of course you can achieve the same result by implementing all the setter and getter methods like this as well:

```
class Animal
  def name # this is the getter method
    @name
  end
  def name=(name)  # this is the setter method
    @name = name
  end
#...same for age...
end
```

13. What is the convention for using "!" at the end of a method name?

The ! indicates that the method is about to change the object itself.
Here's an example:

```
foo = "A TEST STRING"  # a string called foo

foo.downcase!     # modifies foo permanently
a test string

puts foo          # prints modified foo
a test string
```

Similarly if you did not want the object to be changed you could have something simple like:

```
foo2 = "A 2nd Test String"  # a string called foo

foo2.downcase     # modifies foo temporarily
a 2nd test string

puts foo2 nbsp;    # prints original foo
A 2nd Test String
```

14. What is a module?

A module is like a class. Except that it can't be instantiated or subclassed.
In OOP paradigm you would store methods & variables that represent variables in a single class. Say you want to create an Employee representation then the employee's name, age, salary, etc. would all go inside a Employee class, in a file called Employee.rb

Any methods that act on those variables would also go inside that class.
You can achieve the same effect by putting all the variables and methods inside a Employee module:

```
module Employee
  ..variables.
  ...methods
end
```

The main difference between the class & module is that a module cannot be instantiated or sub-classed.
Modules are better suited for library type classes such as Math library, etc.

15. Does Ruby support multiple inheritance?

No, Ruby does not support multiple inheritance.

16. How can you achieve the same effect as multiple inheritance in Ruby? What is a mixin?

Ruby offers a very neat alternative concept called mixin. Modules can be imported inside other class using mixin. They are then mixed-in with the class in which they are imported. Here's an example:

```
module Debug
  def whoAmI?
    "I am #{self.to_s}"
  end
end

class Photo
 include Debug
end

ph = Photo.new

"I am : #<Photo:0x007f8ea218b270>"
```

As you can see above the class Debug and it's method "whoaml?" were mixed-in (added) with the class Photo. That's why you can now create an instance of the Photo class and call the whoAmI? method.

```
ph.whoAmI?
 ⇨  "I am : #<Phonograph:0x007f8ea218b270>"
```

17. How can you implement a singleton pattern?

Singleton means single instance. So, the goal of a singleton pattern is to write a class definition but only allow the creation of the single instance of that object. This can be achieved nicely with the singleton gem as shown below:

```
require 'singleton'
 class Logger
  include Singleton
  def initialize
    @log = File.open("logfile.txt", "a")
  end
  def log(msg)
    @log.puts(msg)
  end
end
```

Adding the singleton as a mixin to the Logger.instance.log('This is just a test message'). The code above will create a single instance of Logger and simply put the message in the logger file.

Singleton patterns are mostly used for DB instance, Logger instance, etc. —- cases where there should be ONE and only ONE instance of the object that is used.

Sometimes you might like to actually hold on to the logger object and use it everywhere you can do so by the following command:

```
logObj = Logger.instance
```

Notice you cannot use the Logger.new to create an object instance because this is a singleton object and therefore calling 'new' would fail.

*18. How can you implement an observer pattern?**

The observer pattern (sometimes known as publish/subscribe) is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems. You might have used them in other programming languages as listener objects. You use them whenever a button is clicked on the screen and a method gets called automatically. As in the case of the singleton pattern, the observer pattern is also implemented by mixing in a module.
In the Ruby implementation, the notifying class mixes in the Observable module, which provides the methods for managing the associated observer objects. And, the observers must implement the update method to receive notifications. Here's an example. Say you want to send an SMS alert to users if a company stock drops then you can do something like this:

```
require "observer"
  class Ticker # Periodically fetch a stock price
    include Observable
     attr_accessor :price
    def initialize symbol, price
      @symbol = symbol
     @price = price
     end

     def run
      lastPrice = nil
      loop do
        @price = @price+Random.rand(11)
        print "Current price: #{price}\n"
        if @price != lastPrice
          changed                # notify observers
          lastPrice = @price
           notify_observers(Time.now, @price)
         end
       end
    end
  end
```

```
class Warner
     def initialize ticker
     ticker.add_observer(self)   # all warners are observers
  end
end

class SMSAlert < Warner
   def update time, price       # callback for observer
      print "--- #{time.to_s}: SMS Alert for price: #{price}\n"
   end
end

class EmailAlert < Warner
   def update time, price       # callback for observer
      print "+++ #{time.to_s}: Email Alert Price changed to
#{price}\n"
   end
 end

ticker = Ticker.new("MSFT", 307)
SMSAlert.new(ticker)
EmailAlert.new(ticker)
 ticker.run

 Current price: 312
 --- 2012-02-22 01:26:04 -0800: SMS Alert for price: 312
 +++ 2012-02-22 01:26:04 -0800: Email Alert Price changed to 312
 Current price: 321
 --- 2012-02-22 01:26:04 -0800: SMS Alert for price: 321
 +++ 2012-02-22 01:26:04 -0800: Email Alert Price changed to 321
 Current price: 323
 --- 2012-02-22 01:26:04 -0800: SMS Alert for price: 323
 +++ 2012-02-22 01:26:04 -0800: Email Alert Price changed to 323
 Current price: 329
 --- 2012-02-22 01:26:04 -0800: SMS Alert for price: 329
 +++ 2012-02-22 01:26:04 -0800: Email Alert Price changed to 329
```

19. What is the purpose of the environment.rb and application.rb files?

There are two files where variables and configuration settings are stored.

- config/environment.rb : Environment settings go here
- config/application.rb : Application level global settings go here

```
config.time_zone = 'Central Time (US & Canada)'
config.i18n.default_locale = :de
config.filter_parameters += [:password] # ensures that
passwords are not logged
```

The same file is also used for configuring various environment settings such as:

```
config.action_mailer.smtp_settings # various email settings
go here
```

What is the purpose of config/environments/development.rb file? You would specify various config settings the development environment in this file.

```
config.action_controller.perform_caching = false # to enable
caching
```

This is because you typically do not want to enable caching in the development environment. The same config setting in the production environment would be equal to true.

20. How can you define a constant?

Create a new file as shown below under: config/initializers/my_constants.rb

```
COLORS = ['white', 'red', 'green']
```

21. How can you trigger a method when a module is included inside a class?

Fire a method inside a class is very simple.
Say you have a module file trig.rb:

```
module Trig
  PI = 3.141592654
  def Trig.sin(x)
   # ..
  end
  def Trig.cos(x)
   # ..
  end
end
```

Now you simply import this module inside your class and invoke the method using the "module.method_name" syntax as shown below

```
require "trig"

class myclass
y = Trig.sin(Trig::PI/4)
```

This type of invocation ensures that the right module method gets called.

22. What is the default access modifier (public/protected/private) for a method?

By default all methods are public, except the initialize(constructor) method.

You can make methods private using this declaration within your class:

```
class MyClass
    def method_public_here
    end
    private# all methods that follow will be made private:
not accessible for outside objects
    def method_private_here
    end
  end
```

23. How can you call the base class method from inside of its overridden method?

If you are inside the overridden method in the derived class then a simple call to super will call the right method in the base class

```
class Parent
   def try_this()
      puts "parent"
   end
end

class Child < Parent
   def try_this()
      super()
      puts "child"
   end
end

ch = Child.new
ch.try_this()
```

This generates the output

```
parent
child
```

Now if you just want to call the base class without calling the derived class then the best way to do that is to simply assign an alias to the parent method like this:

```
class Parent
  def knox
    puts 'parent'
  end
end

class Child < Parent
   alias_method :parent_knox, :knox
   def knox
     puts 'child'
   end
```

```
end

ch = Child.new
ch.parent_knox
ch.knox
```

This allows you to call the base class method with the alias parent_knox and the derived class method knox can be called directly.

```
parent
child
```

24. Define the Rails MVC implementation using an example.

25. What is a migration?

Migrations are a feature of Active Record that allows you to evolve your database schema over time. Rather than write schema modifications in pure SQL, migrations allow you to use an easy Ruby DSL to describe changes to your tables.

26. What are some of the methods available to migrations?

- Announce
- Check_pending!
- Connection
- Copy
- Disable_ddl_transaction!
- Down
- Exec_migration
- Method_missing
- Migrate
- New
- Next_migration_number
- Reversible
- Revert
- Reverting?
- Run
- Say
- Say_with_time
- Suppress_messages
- Up
- Write

27. What can I find in the schema.rb file?

Schema.rb is the authoritative source for your database schema. These files are not designed to be edited, they just represent the current state of the database.

28. Tell me some of the types of validations and their intended use.

- :foreign_key -> verifies that the foreign key is correct
- uniqueness: true -> verifies the uniqueness of a column

29. Tell me some of the types of callbacks and their intended use.

- Save
- Valid
- Before_validation
- After_validation
- Before_save
- Before_create

30. What are all of the types of associations? When would I use each one?

- Belongs_to
- Has_one
- Has_many
- Has_many :through
- Has_one :through
- Has_and_belongs_to_many
- Belongs_to
- Has_one
- Has_many :through
- Has_and_belongs_to_many
- Polymorphic associations

31. What's the purpose of the id column in a database?

To provide an entry with a universal id that's accessible from anywhere.

32. Why would I use a polymorphic association?

A polymorphic association is best used when a model can belong to more than one other model, on a single association.

33. What would the following queries return:

```
a. client = Client.find(10)
```

```
b. Client.order("created_at DESC")
c. Client.limit(5).offset(30)
```

34. *What are the HTTP verbs, paths, action and usage created by including the following in your routes.rb file?* `resources :photos`
35. *What is the params hash and how do you retrieve data from it?*

See the following discussion: http://stackoverflow.com/questions/6885990/rails-params-explained

36. *How can you accept parameters in JSON?*

See the following explanation:
http://guides.rubyonrails.org/action_controller_overview.html

37. *What is a session and how does it work?*

Sessions are a way to keep track of a certain state of a particular user. Sessions allow a user to avoid identifying themselves and authenticating on every request. Rails will create a new session automatically if a new user accesses the application.

38. *How does a before_filter work?*

This is code that is run before another section of code.

39. *How is calling render different from calling redirect_to?*

Render will render a particular view using the instance variables available in the action. E.g. if a render was used for the new action, when a user goes to /new, the new action in the controller is called, instance variables are created and then passed to the new view.

Redirect_to will send a redirect to the user's browser telling it to re-request a new URL. Then the browser will send a new request to that URL and it will go through the action for that URL, oblivious to the fact that it was redirected to. None of the variables created in the action that caused the redirect will be available to the redirected view.

40. *How do you render JSON?*

In a view, you can render JSON by using format.json

41. *What are asset tag helpers and what do they do?*

This is a module that provides methods for generating HTML that links views to assets such as images, javascripts, stylesheets, and feeds.

42. *What does yield do and how do I use it?*

Duplicate question.

43. What is a partial used for? How do I name the files?

See the following explanation:
http://guides.rubyonrails.org/layouts_and_rendering.html

44. How is a form helper used?

See the following explanation: http://guides.rubyonrails.org/form_helpers.html

45. What are some inputs for forms and what is the Rails method for creating them?

See the following explanation: http://guides.rubyonrails.org/form_helpers.html

46. What are some of the ways that Active Support extends the functionality of Ruby?

See the following for an explanation:
http://guides.rubyonrails.org/active_support_core_extensions.html

47. How does the Rails Asset Pipeline work? When was it introduced?

See the explanation here: https://makandracards.com/makandra/8951-rails-asset-pipeline-how-to-organize-stylesheets-in-sub-folders

48. What is the correct way to reference a file in the asset pipeline?

See the explanation here: https://makandracards.com/makandra/8951-rails-asset-pipeline-how-to-organize-stylesheets-in-sub-folders

49. What is a preprocessor and how does it work?

A preprocessor is a program that processes its input data to product output that is used as input to another program.

50. What are some of the ways that you can configure your Rails generators? Why would someone do this?

People customize and create their own Rails generators to expedite specific components of their workflow. You can configure your ORM, your template engine and your testing framework.

51. What are some new features in Rails 4?

See the following explanation:

http://edgeguides.rubyonrails.org/4_0_release_notes.html

52. What is becoming deprecated in Rails 4?

See the following explanation:
http://edgeguides.rubyonrails.org/4_0_release_notes.html

53. Why are you excited about Rails 4?

Subjective.

54. What is scope?

See this discussion: http://stackoverflow.com/questions/4869994/what-is-scope-named-scope-in-rails

55. Can you give an example of a class that should be inside the lib folder?

Modules are often placed in the lib folder.

56. Where should you put code that is supposed to run when your application launches?

To run some code before Rails itself is loaded, put it above the call to require 'rails/all' in config/application.rb or initializers.rb

57. Can you give me an example of a tool that's used specifically for deployment?

Heroku and Capistrano are popular deployment tools.

58. How can you migrate your database schema one level down?

The rake tool does most of the migrations. It has this nifty syntax to go back one step:

```
rake db:rollback
```

If you want to rollback all the way to the beginning you would use:

```
rake db:reset
```

This would drop the database, recreate the Database and load the current schema into it. If you want to rollback multiple steps at the same time you would use:

```
rake db:rollback STEP=3
```

To rollback all the way and if you are not worried about losing the data then you can drop the database completely with purge like this:

```
rake db:purge
```

*59. What is a sweeper?**

Sometimes you want to have control over how often and when the cache expires.

Sometimes it is a good idea to have the system determine that on a logical basis. Say you have a list of product on your site and you want to reload the cache each time a new product is added/updated/deleted, then you can achieve this by using the sweeper.

```
class ProductSweeper < ActionController::Caching::Sweeper
  observe Product# This sweeper is going to keep an eye on the
Product model
  # If our sweeper detects that a Product was created call this
  def after_create(product)
    expire_cache_for(product)
  end
  # If our sweeper detects that a Product was updated call this
  def after_update(product)
    expire_cache_for(product)
  end
  # If our sweeper detects that a Product was deleted call this
  def after_destroy(product)
    expire_cache_for(product)
  end
  private
  def expire_cache_for(product)
    # Expire the index page now that we added a new product
    expire_page(:controller => 'products', :action => 'index')
    # Expire a fragment
    expire_fragment('all_available_products')
  end
end
```

60. How can you implement caching in Rails?

Rails offers multiple ways to cache content. Fragment caching is my favorite because it gives you the choice to fragment to pull a portion from the cache and the remaining from a real-time DB call. Say you wanted to show all the orders placed on your website in real time and didn't want to cache that part of the page, but did want to cache the part of the page which lists all products available, you could use this piece of code:

```
<% Order.find_recent.each do |o| %>
  <%= o.buyer.name %> bought <%= o.product.name %>
<% end %>
<% cache do %>  All available products:
  <% Product.all.each do |p| %>
    <%= link_to p.name, product_url(p) %>
  <% end %>
<% end %>
```

Another technique that works well for static pages is page caching. This technique is often used for home pages and is super fast.

```
class ProductsController < ActionController
   caches_page:index
  def index
    @products = Products.all
  end
end
```

You can cache: views, and page fragments

61. What is a filter? When is it called?

Filters are methods that are called either before/after a controller action is called. Say a user requests a controller action such as userdashboard/index. In such a case a filter can be setup so that the UserDashboard/index page is only accessible to loggedin users by adding the following lines towards the beginning of the page:

```
class UserDashboardController < ApplicationController
  before_filter :confirm_logged_in,  :except => [:login,
:attempt_login, :logout]
  def index
 ......
  end
```

After filters are not used too much but they have the effect of exe

62. What do controllers do in Rails?

Once a request comes into the Rails stack, it goes to the routes table to determine which controller and action should be called.

Once a controller action is determined the request is routed to the controller and it does the needed processing by connecting with the DB if needed and then it sends control to the View to render the output.

So, really the flow for Rails goes somewhat like this:

Customer-> Routes-> Controller -> Model(DB) -> Controller -> View -> Customer

63. What is RESTful routing?

Routing is fun. If you have ever dealt with IIS you will fall in love with RESTful routing. Here's how it works.

Say you want your users to have access to certain pages such as:

/photos/new

/photos/1/edit

/photos/1

And, you want the right controller to get called.

And, you want the right view to get rendered.

All this is made possible with a single entry in the routes.rb file as shown below:

resources :photos

In Rails, a resourceful route provides a mapping between HTTP verbs and URLs to controller actions. By convention, each action also maps to particular CRUD operations in a database. The single entry in the routing file creates seven different routes in your application, all mapping to the Photos controller:

HTTP Verb	Path	action	used for
GET	/photos	index	display a list of all photos
GET	/photos/new	new	return an HTML form for creating a new photo
POST	/photos	create	create a new photo
GET	/photos/:id	show	display a specific photo
GET	/photos/:id/edit	edit	return an HTML form for editing a photo
PUT	/photos/:id	update	update a specific photo
DELETE	/photos/:id	destroy	delete a specific photo

64. How can you list all routes for an application?

rake routes -- will display all routes for an application.

65. How can you send a multi-part email?

Action Mailer will automatically send multipart emails if you have different templates for the same action. So, for our UserMailer example, if you have welcome_email.text.erb, Action Mailer will automatically send a multipart email with the HTML and text versions setup as different parts.

66. What is the purpose of layouts?

Layouts are partial ruby/html files that are used to render the content pages.
There are placed in the folder: app/views/layouts. Items that you would typically put in this folder are things like headers/footers, navigation elements, etc. Here's a sample layout file: /app/views/layout/application.html.erb

```
<html lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
    <title>Learning System | <%= @page_title || 'Admin Area'
%></title>
    <meta name="author" content="Anil Punjabi">
    <%= stylesheet_link_tag('public', 'admin', :media => 'all')
%>
    <%= javascript_include_tag('application') %>
  </head>
  <body>
    <div id="header">
      <h1>Learning System</h1>
    </div>
    <div id="main">
      <% if !flash[:notice].blank? %>
      <div class="notice">
        <%= flash[:notice] %>
      </div>
      <% end %>
      <div id="content">
        <%= yield %>
      </div>
    </div>
    <div id="footer">
      <p id="copyright">&copy; / Anil Punjabi</p>
    </div>
  </body>
</html>
```

67. Is it possible to embed partial views inside layouts? How?

That is the purpose of layouts. You embed partial views inside the file /app/views/layout/application.html.erb and then whenever you render any page this layout is merged with it.

68. What is Rake?

Rake is a popular ruby gem that makes the job of running tasks simpler.

Rake is most often used for DB tasks, but it can be used for m

The common DB commands are:

```
rake db:migrate
```

```
rake db:reset
```

You can use cron to schedule rake tasks.

Sometimes you would create a dataloader.rake file and put it in the lib/tasks folder so that it can be used to populate the database on startup.

69. What is Capistrano?

Capistrano is a popular deployment tool — it allows developers to push code from their desktop to the servers.

70. What is a "has and belongs to many" (HABTM) association?

The has_and_belongs_to_many association creates a many-to-many relationship with another model.

71. What is the difference between has_one and belongs_to?

A has_one relationship is used to define a 1:1 relationship between two objects. Examples are:

A project has_one projectManager
A sandwich has_one buyer
A sandwich has_one seller
A belongs_to relationship on the other hand is used to define the reverse association for the same 1:1 relationship that is defined using the has_one keyword.

```
class Employee < ActiveRecord::Base
  has_one :office
end
class Office < ActiveRecord::Base
  belongs_to :employee    # foreign key - employee_id
end
```

The important thing to keep in mind is that you need to declare both associations in order for the relationships to work correctly.

72. How can you implement single table inheritance?

See http://www.therailworld.com/posts/18-Single-Table-Inheritance-with-Rails for an in-depth explanation.

73. What is a polymorphic association?

A polymorphic association is what one would call "open-ended" association of one class with multiple objects. Let's say you have a generic class called "Picture". Now this

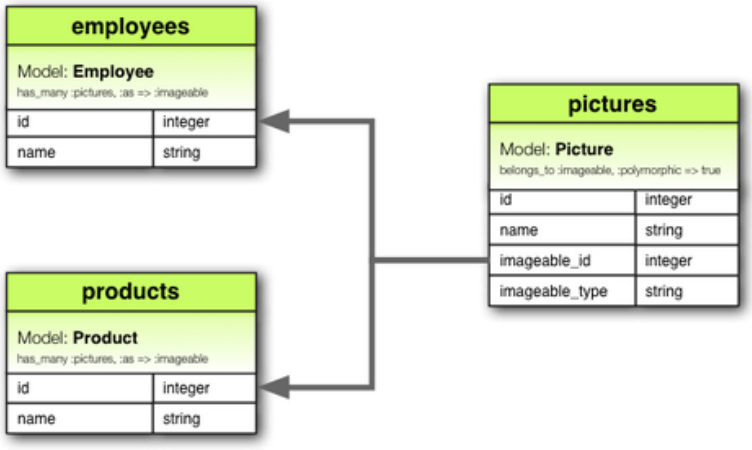
“Picture" class might be used to store Pictures for Employees, Products and Dogs. They all have pictures and they all are associated to the Picture class. Hence, this type of open-ended association is called “Polymorphic" association. Let’s see how we can declare them.

```
class Picture < ActiveRecord::Base
  belongs_to :imageable, :polymorphic => true
end

class Employee < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end

class Product < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end

class Dog < ActiveRecord::Base
  has_one :picture, :as => :imageable
end
```

Notice that the Dog object has a 1:1 relationship with the Picture class, whereas the Product class has a 1:many relationship with the Picture class.

This capability prevents the headache of having of declare new classes for each new type of Picture.

The polymorphic relationship would look somewhat like this.




```
class Picture < ActiveRecord::Base
  belongs_to :imageable, :polymorphic => true
end

class Employee < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end

class Product < ActiveRecord::Base
  has_many :pictures, :as => :imageable
end
```

74. What is eager loading?

Eager loading is a great optimization strategy to reduce the number of queries that are made against the DB.

Say you are finding 10 employees and then you are looking for their post codes. Then your query would appear something like this:

```
clients = Client.limit(10)
clients.each do |client|
  puts client.address.postcode
end
```

This may seem fine at first look but really this implementation leaves much to be desired. It makes 11 DB calls just to get the results.

Now you can optimize this query by making a slight change in the request like this:

```
clients = Client.includes(:address).limit(10)
clients.each do |client|
```

```
  puts client.address.postcode
end
```

This new request makes two SQL calls like this:

```
SELECT * FROM clients LIMIT 10
SELECT addresses.* FROM addresses
    WHERE (addresses.client_id IN (1,2,3,4,5,6,7,8,9,10))
```

So, as you can see it really loads a lot more upfront and therefore it is called eager loading.

75. How can you eager load associated objects?*

See the following links:

http://guides.rubyonrails.org/active_record_querying.html#eager-loading-associations

And

http://stackoverflow.com/questions/9642900/eager-loading-the-right-way-to-do-things

76. How does validation work?

Validation means checking to see if data is good before it is stored in the database.

During signups and other such user input cases you want to check and be sure that the data is validated. In the past developers would often put this type of validation logic as triggers in the database.

In a MVC architecture one can do validations at each level.

You can do validations in the controllers but it is usually a good idea to keep your controllers skinny.

Views suffer from the javascript limitation because javascript can be disabled on the client side so they are not completely reliable.

The best way to manage validation is to put it in the model code. This model code is really the closest as you can be to the database and works very well for Rails applications.

Here are a few validation examples:

```
class Person < ActiveRecord::Base
  validates :name, :length => { :minimum => 2 }
```

```
  validates :points, :numericality => { :only_integer =>
true }  # only integer
  validates :age,  :numericality => { :greater_than => 18 }
# greater than 18
  validates :email, :uniqueness => true
  validates :email, :confirmation => true  # this is to
validate that the two email fields are identical
  validates :email_confirmation, :presence => true # this is
to validate that the email confirmation field is not nil
```

In your view template you may use something like this:

```
<%= text_field :person, :email %> <%= text_field :person,
:email_confirmation %>
```

77. How can you add a custom validation on your model?

78. What is "Flash"?

Now custom validations takes it to the next step. Say you want to confirm that the data meets certain criteria. Flash is simply a way to pass some value to the next action. Anything you place in the flash will be exposed to the very next action and then cleared out. Here's an example:

```
def destroy

 section = Section.find(params[:id])
 section.destroy
 flash[:notice] = "Section destroyed."
 redirect_to(:action => 'list', :page_id => @page.id)

end
```

Then wherever you want to use the flash you can write this code. I often put this snippet in the application.html.erb file, somewhere towards the top:

```
<% if !flash[:notice].blank? %>
      <div class="notice">
      <%= flash[:notice] %>
```

79. How can you install the missing gems that your application requires in the simplest way?

```
bundle install
```

This command will install all dependencies and missing gems. It looks at your Gemfile to figure out what gems are needed by your application.

80. How can you implement internationalization?

Ruby ships with i18n which is an internationalization gem.

You need to create locale files and save them under the config/locales directory as:

en.yml
es.yml
fr.yml

In your code you would need to specify that the text would be locale specific. So change it to something like this:

```
.content %h1 = t("main_page.hello")
%p           = t("main_page.welcome")
```

Then you have to select the actual locale.

81. What plugin would you recommend for authentication and authorization?

Devise or Sorcery are both great choices.

82. What plugin do you use for full-text search?

Sunspot is a good choice for full-text search. It uses solr.

83. What is the difference between a plugin and a gem?

A gem is just ruby code. It is installed on a machine and it's available for all ruby applications running on that machine.

Rails, rake, json, rspec — are all examples of gems.

Plugin is also ruby code but it is installed in the application folder and only available for that specific application.

Sitemap-generator, etc.

In general, since Rails works well with gems you will find that you would be mostly integrating with gem files and not plugins in general. Most developers release their libraries as gems.

*84. How can you implement a search feature that searches for multiple models?**

If you are using acts_as_solr for your search you will be able to use multi_solr_search to

enable search across multiple models. Also, you can configure Sunspot/Solr to support search across multiple models. Sphinx, another powerful search server can be used to search across multiple models and it works great.

85. How can you upload a file to a server?

Paperclip is a good solution to manage file uploads to a server. It can also help you with multiple file uploads and associate it with ActiveRecord. There are also good examples online that show how you can make rotating sliders with the paperclip images. Another nice solution is using carrier_wave gem. The nice thing about carrier_wave is that it has good documentation on how to integrate with S3, Google & Rackspace for file storage. You can achieve the same file storage capability with Paperclip as well though.

*86. Is Rails scalable?**

Yes Rails gives you complete freedom to use all traditional means of scaling an application. Things like memcached, caching full pages, caching fragments are all supported. You can use any standard CDN to serve your media and static content as well. Database scaling using sharding is supported.

Finally heroku makes your life easier by giving you the flexibility to scale up/down based on your need. Mostly websites have a peak time during which you need more servers and then there is a sleep time. Heroku makes that on-demand scaling process simpler. Companies such as HireFireApp.com makes the autoscale process easier.

87. How can you secure a Rails application?

Rails has a lot of in-built capabilities to deal with common web-security issues.

- SQL Injection
- Cross-Site
- Session fixation and Session hijacking
- Captcha

88. What are some good community resources for Rails?

Stackoverflow, Github, various Meetup groups

89. Where would you ask the community to get answers for your technical questions?

Stackoverflow and meetup groups.

90. Explain the difference between Controller & Model logic in Rails

JavaScript

1. *What are the three layers of progressive enhancement?*

Structure, presentation, and behaviour

2. *Why do you generally place JavaScript (JS) at the bottom of an HTML document?*

JavaScript at the bottom allows the entire document to load before JavaScript blocks the download of subsequent objects on the page

3. *What is the most widely supported method of persistent data storage in JS?*

Cookies

4. *What is an anonymous function?*

A function that is declared as it runs and has no assigned name

5. *When is a function considered to be a callback function?*

When it is called by another function

6. *How do you obtain feedback from a user?*

Through events

7. *What are four types of DOM nodes?*

Document, Element, Text, Attribute

8. *Explain what the DOM is and how it relates to HTML and JS?*

The DOM is a standardized outline of an HTML document that creates access points or "hooks" in the form of nodes by which JavaScript can enter.

9. *What happens if you try to set a node attribute for an attribute that does not exist on the node already?*

The attribute node will be created automatically and then set as normal.

10. *What are the three data types you can use with a variable?*

Number, String, Boolean.

11. What is an associative array?

An associative array is a normal array that uses strings as index values instead of numbers.

12. What is a multidimensional array?

An array that contains other arrays.

13. Why is it best to position all variables at the top of your JS file?

Variables cache in JavaScript, so it's best to cache them all at once for better referencing.

14. Why are some of the words reserved in JS?

Reserved words are references to terms that already exist in the JavaScript language. Therefore, using them in your code would create unwanted collisions and errors.

15. How are anonymous functions different from basic functions?

Anonymous functions execute immediately and have no label or name assigned to them.

16. What is the difference between an event handler and an event listener?

With event handlers, you can attach only a single function to a specific event of a DOM node. This limitation does not exist with event listeners.

17. What method is used for fallback support of event listeners in IE8 and earlier?

`attachEvent()`

18. What is the purpose of the `preventDefault()` method?

The `preventDefault()` method is used to stop the default browser behavior of a given event, such as stopping the browser from executing an href or a form submit.

19. Why shouldn't you start a function or variable name with an underscore?

In other languages an underscore signifies privacy, which doesn't exist in JavaScript.

20. When is it best to use an anonymous function?

When you want to use a block of code only once.

21. What is the data type of 3.14?

typeof 3.14

"number"

22. What is the data type of 3?

typeof 3

"number"

Many programming languages have different data types for floating and fixed point numbers, but JavaScript only has one data type for all numbers.

23. The function `Math.sqrt(-5);` returns `NaN`. What does `NaN` mean?

NaN stands for Not-A-Number
This mean that the expression does not evaluate to a number.

24. Write a function that calculates `Math.sqrt(-5);` and evaluates to `NaN` and false otherwise.

```
isNaN(Math.sqrt(-5));
```

25. What does the following statement evaluate to? `NaN === NaN`

```
false
```

26. Write a function to round 3.78 down to the nearest integer

```
Math.floor(3.78);
```

27. Write a function to calculate the length of the string "bob".

```
"bob".length
```
Notice that length is not a function.

28. Are JavaScript strings mutable?

No.

29. Write a function to convert all the letters of the string "Happy New Year" to uppercase.

```
"Happy New Year".toUpperCase();
```

30. *What are falsy values? What are falsy values in JavaScript?*

Falsy values are considered false in a boolean context (i.e. a falsy value is considered the same as false in the conditional of an if statment). The falsy values in JavaScript are false, null, undefined, "" (the empty string), 0, and NaN.

31. *Write a ternary operator that prints "Oh Yea" to the console if 5 is greater than 3 and prints "Huh" otherwise.*

```
5 > 3 ? "Oh Yea" : "Huh";
```

32. *Show to ways to access the height of the following "Shaq" object.*

```
var shaq = {
  body: {
    height: "7'1",
    weight: 325
  },
  stats: {
    points_per_game: 24,
    minutes_per_game: 11
  }
}


shaq.body.height
OR
shaq["body"]["height"]
```

jQuery

1. *Create an HTML page. Using JavaScript, set variable `a` to 3 and variable `b` to 5 and display the sum of `a` and `b` in the browser.*

```
var a = 3;
var b = 5;
document.write(a+b);
```

2. *Create a box that gives the user the message "Hello World!" and makes the user click "OK" before the rest of the page loads.*

```
alert("Hello World");
```

The browser runs the JavaScript as soon as it encounters the code - in this case the browser runs the JavaScript before it encounters the <body> code.

3. *Write the text "Hello World" in JavaScript in a way that preserves the CSS formatting related to the `<p>` tag.*

```
document.write("<p>Hello World</p>");
```

4. *Fade the <p> tag below on to the screen over the course of three seconds*

```
<!DOCTYPE html>
<html>
  <head>
    <title>Fade In</title>
  </head>
  <body>
    <p>This text will fade in over three seconds</p>
  </body>
</html>

<script src =
"https://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.m
in.js"></script>
<script type="text/javascript">
  $ (document).ready(function() {
    $('p').hide().fadeIn(3000);
  });
</script>
```

The 3000 argument to the fadeIn function means that the image should fade in over 3000 miliseconds, or 3 seconds

5. *Create a page that asks the user "What is your name?" and prints "Welcome NAME" where NAME is the name supplied by the user.*

```
var name = prompt("What is your name?", "Enter name here");
document.write("<p>Hello " + name + ".</p>")
```

6. *Create an array of four economists: "Taylor", "Mankiw", "Roubini", "Krugman". Iterate over the array and print the names in the browser.*

```
var economists = [ "Taylor", "Mankiw", "Roubini", "Krugman"];
$.each(economists, function(index, economist){
  document.write("<p>" + economist + "</p>");
}); //end each
```

Notice that the each method takes two arguments: the first argument is an array and the second is a block. The block is an anonymous function.

Can also solve this with a Javascript while loop:

```
var counter = 0;
while (counter < economists.length)
{
  document.write("<p>" + economists[counter] + "</p>");
  counter += 1;
}
```

Make note of the Javascript while loop syntax - this is similar to the if/else Javascript syntax.

7. *Write a function that prints out the current date in mm/dd/yyyy format.*

```
var x = new Date();
var day = x.getDate();
var month = x.getMonth() + 1;
var year = x.getFullYear();
document.write(month + "/" + day + "/" + year)
```

8. *Create a function print, so that* `print("Hello World")` *will render the text "Hello World" in the browser.*

```
function print (message) {
  document.write(message);
};
print("Hello World")


<div class="intro">
  <h1>Mental Status?</h1>
</div>

$('.intro').append("<p>JavaScript drives me crazy
sometimes</p>");
```

9. *Add the text "Sup" to the beginning of* ***<h1>My homies</h1>****.*

```
$('h1').prepend("Sup");
```

10. *Delete* ***<h3>Delete me</h3>****.*

```
$('h3').remove();
```

11. *Replace* ***<h4>This is the old h4</h4>*** *with* ***<h4>This is the BRAND NEW h4 </h4>****.*

```
$('h4').replaceWith('<h4>This is the BRAND NEW h4</h4>');
```

12. *Remove the "title" class from* *** Breaking Bad ***

```
$('.title').removeClass();
```

13. *Add a solid red border around* *** jQuery will make this have a solid red border ****.*

```
$('#block').css({'border': "2px solid red"})
```

There is another syntax you can use when you only pass one argument to the css function, but don't use it because it does not work when multiple arguments are passed.

14. Make the following formatting changes to the div below: background color FF7373, font color white, border black

```
<div id = "meow">
  <p>CSS is FUN</p>
</div>

$('#meow').css({'background-color' : '#FF7373', 'color' :
'white', 'border' : 'black solid 2px'})
This could also be written as follows:
$('#meow').css({
  'background-color' : '#FF7373',
  'color' : 'white',
  'border' : 'black solid 2px'
})
```

15. *Store the value of the src attribute of* ****** *in a variable to the screen to demonstrate it has been done correctly.*

```
var imageFile = $('#chapel').attr('src');

document.write('<p>'+imageFile+'</p>');
```

16. *Change the picture source of* `<img id = “picture” src=”indian_chapel.jpg” height = ‘300px’ width = ‘400px’ />` *to be “oktoberfest.jpg”*

```
$('#picture').attr('src', "oktoberfest.jpg");
```

17. *Make the following image gradually disappear over a period of 3 seconds.* `<img id = "picture" src="oktoberfest.jpg" height = '300px' width = '400px'/>`

```
$('img').fadeOut(3000);
```

18. *Create a page with two images. For each image in the page, give the user the alert “I found an image”.*

```
<img src="oktoberfest.jpg" height = "300px" width =
"400px"/>
  <br />
<img src="indian_chapel.jpg" height = "300px" width =
"400px"/>

$('img').each(function(){
  alert('I found an image');
});
```

19. *Make a paragraph that says "This will be changed" move 650 pixels to the right, grow to 50px font size, and change opacity to 0.5 over the course of 1.5 seconds.*

```
<body>
  <div id ="message">
    <p>This will be changed</p>
  </div>
</body>

<script type="text/javascript">
  $(document).ready(function(){
    $('#message').animate(
    {
      'margin-left': '650px',
      opacity: .5, //this makes it turn grey towards the end
      'font-size': '50px'
    },
      1500
    ); //end animate
  }); //end ready
</script>
```

Git

1. *How do you start Git and what is the process called?*

```
$ git init
```
The process is called initializing a git repository

2. *How do you determine the current state of the project?*

$ git status

3. *How do you move all your changes since your last commit to the staging area?*

$ git add .

4. *Store the saved changes in the repository and add a message "first commit".*

$ git commit -m "first commit"

5. *Show a list of all the commits that have been made.*

$ git log

6. *How do you put your local repository (repo) on the GitHub server?*
    - create a new repository in github
    - get the SSH key (SSH stands for secure shell)
    - `$ git remote add origin [SSH key - something like git@github.com…]`
    - `$ git push –u origin master`

7. *Take a git project from GitHub to your local machine to work on it.*

```
$ git pull origin master
```
Or, if you are taking a project that is not already on your local machine, use $ git clone.

8. *What is the purpose of working off multiple branches.*

You can create a separate branch when you are developing a feature, etc. and can commit to this branch separately. When the feature branch is working properly, you can merge it with the master branch.

9. *Create a branch called working.*

```
$ git checkout –b working
```

10. *List all of the branches.*

```
$ git branch
```

11. *Switch to the working branch.*

```
$ git checkout working
```

12. *Merge the changes that were made in the working branch with the master branch.*

```
$ git checkout master
$ git merge working
```

13. *Delete the working branch.*

```
$ git branch -d working
```

14. *Put everything up on the remote repository.*

```
$ git push
```

15. *Pull a branch from a remote repository and create the branch locally.*

```
$ git checkout -b fix_stuff origin/fix_stuff
```