Quiz #3 Solutions

Question 1

```
function ConvertEmitterToCalllback(emitter, callback) {

      emitter.on('EventA', function(arg1, arg2) {
            callback('EventA', arg1, arg2);
      });
      emitter.on('EventB', function() {
            callback('EventB');
      });
}
```

Question 2

2.1

The code would call the listeners on event 'X' with the parameter 'Hi' and the listeners on event 'Y' with the parameter 'Bye'. This is not an obvious error and could do something useful so "code could be correct" is the answer.

2.2

This code emits two events. Event 'X' is emitted with its parameter being a function that logs something and Event 'Y' is emitted with its parameter being a similar function that logs something else. This is not an error and could do something useful so code could be correct" is the answer.

2.3

This code registers an event handler on events 'X' and 'Y' but rather than passing a function it passes a string. This doesn't make any sense and "code could be an error".

2.4

This code registers event handlers for events 'X' and 'Y'. It's not obviously an error and does something so "code could be correct".

Question 3

The `callback` function is defined in the `async.each` code and used by that code to get informed when the iteratee function is finished processing the item. The async.each function will wait until all the items' callback functions are made before continuing; alternatively, `callback` can be called with an argument to indicate an error occurring in one of the iteratee functions, which will end the waiting of async.each earlier.

Question 4

4.1

The ExpressJS next function parameter is used by a handler to continue processing the request by other handlers. This is a sign that the handler is acting as "middleware". The modification of httpRequest object is frequently used by middleware to pass information to other handlers. For example, the JSON body parser will parse the JSON string in the body and attach the resulting object to the httpRequest. Similarly, the Express session middleware will add the "session" property the httpRequest.

4.2

Doing a httpRespone.send() does an httpResponse.end() causing the response to the HTTP request to be sent out to the client. A subsequent call to next would pass the request on to other handlers with the request that has already been responded to. Since these other handlers wouldn't be able to alter the response already sent there is less reasons for middleware to pass the request to them.

Question 5

MongoDB's document data model is effectively a JSON object. As an JavaScript object it can have an arbitrary set of properties. The relational model has tables with a fixed set of columns. It is not possible to map arbitrary sets of properties into a row in a relational database table. An ODL forces the objects to follow a particular schema making it possible to map to rows in a table.

Question 6

Session manager layers like Express Session use Message Authentication Codes (MACs) to sign session cookies effectively preventing a forger from generating a valid looking session cookie.

Question 7

If you lost all the session state in your backend all the currently logged in users of the web application would have a frontend that thought they were logged in and a backend having no knowledge of them. The requests sent by the frontend to the backend would be treated as if they can from an unauthorized user. The users would see a web application that could no longer access backend functionality.

Question 8

8.1

The web application's backend must deal with a threat model of an attacker taking the current session cookie and using it to launch requests to the backend. Doing careful validation of input in the frontend code does nothing to prevent this attack.

8.2

Validation done in the frontend can inform the user of a problem near-instantly rather than having to wait until a request to the backend has detected the problem. Letting the user know about a problem quickly and giving them a chance to fix it leads to a better user experience.

Question 9

A cross site scripting attack involves injecting HTML into a browser that contains HTML `script` tags loading the attacker's JavaScript code. To prevent this a sanitization process on HTML must find and remove the problematic `script` tags.

Question 10

10.1

An eavesdropper attack involves an attacker passively spying on network traffic between the frontend and backend of a web application. Encryption/decryption prevents this kind of spying on network traffic.

10.2

Denial of service attack involves using of resources of a web application so legitimate users can get service. Encryption, MACs, and HTTPS certs do not help with this kind of attack.

10.3

Phishing attacks involve trying to fool the user with a fake copy of the web application. HTTPS certificates allow a user to validate the backend they are talking to making a fake copy harder.

10.4

SQL injection attacks involve sending input to a web application backend that gets interpreted as SQL code. Encryption, MACs, and HTTPS certs do not help with this kind of attack.

Question 11

Scale-out architectures scale by adding or subtracting instances of the service. Failures of instances naturally map to subtraction of instances and can be handled in this way. Scale-up architectures, on the other hand, put all resources into one big machine making failure containment a difficult problem.

Question 12

Content distribution network (CDN) works by replicating content widely in geographically diverse areas. Updating content requires updating all the replicas in geographically diverse areas, a significant task that would be time and resource intensive to do frequently.