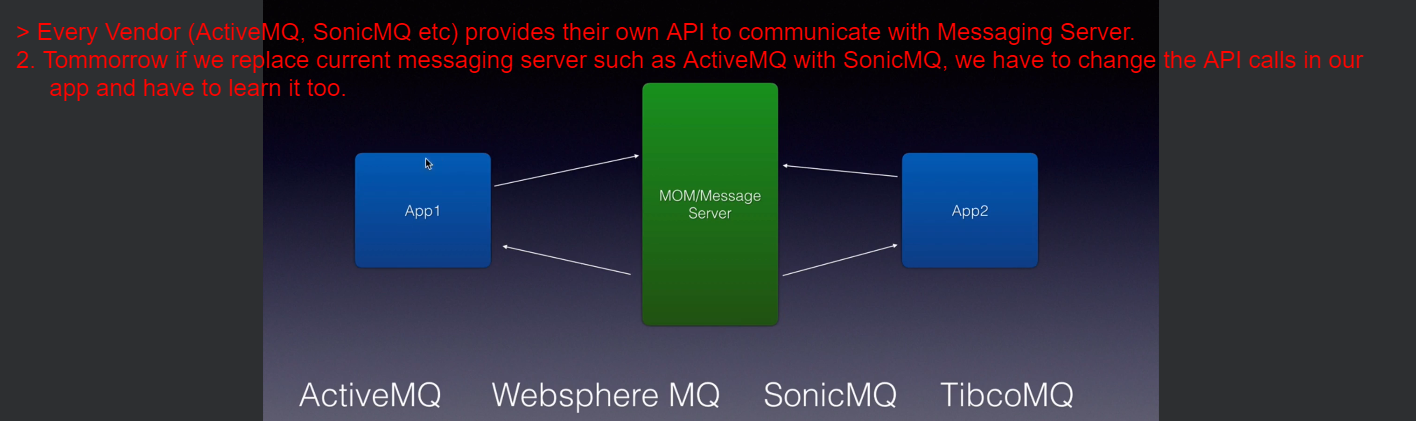
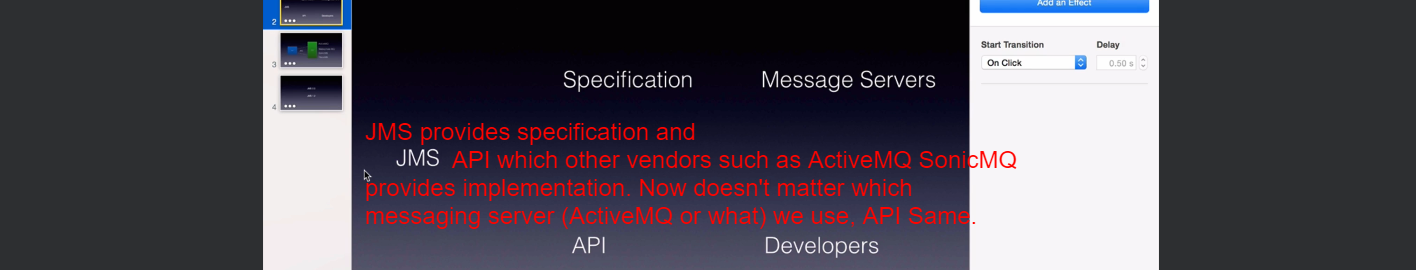
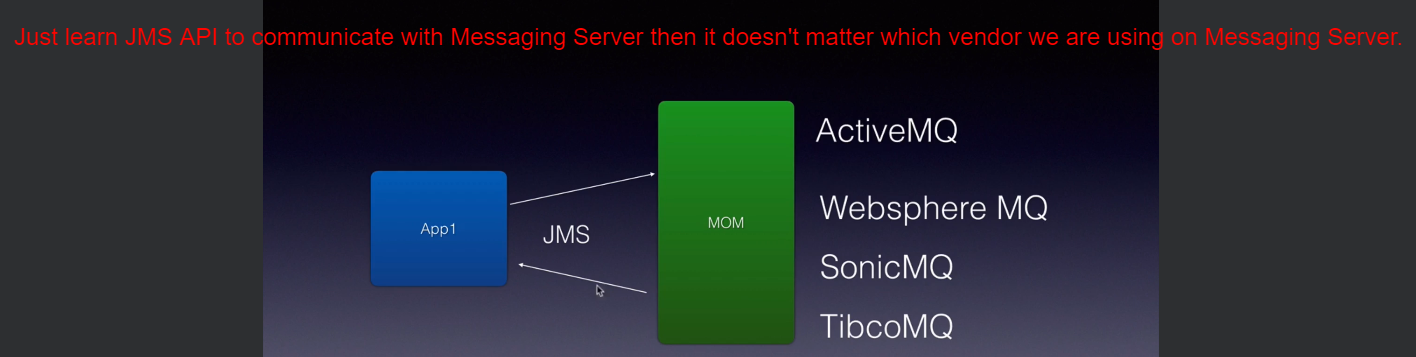
1. When two or more applications communicate with each other using messaging, we as a developer needs to use some API to communicate with the **messaging server** to send the msgs and also to receive them.
2. These Messaging Servers could be (these are actually Vendors)
   1. ActiveMQ.
   2. SonicMQ.
   3. TibcoMQ.
   4. Websphere MQ.
3. Before introducing JMS, developers have to use vendor specific API such as ActiveMQ.  
   So, if currently our message server is ActiveMQ, then we have to learn ActiveMQ API to Communicate with Messaging Server.  
   Tomorrow, our architect or organization decides to use SonicMQ, we have to change our app to use SonicMQ API.
4. That is where JMS Standard came in introduced by Sun Microsystems now maintained by oracle.
5. Like any other Java EE standard, it has specification and an API.
6. **Specification & API**:
   1. Set of rules written in plain English for Messaging Server such as ActiveMQ, SonicMQ etc.
   2. They follow the specification and implement the JMS standards to comply to the standards and the implementation is the API which is used by the Developers.
7. Now we as Developers need to learn API from all the Messaging Server vendors such as ActiveMQ, SonicMQ.
8. We will learn just one single API specified by JMS and once we do that we can switch to any Messaging Server that implements the JSM Standard.
9.   
   So, once we learn JMS, our app will be able to send and receive messages from any f these messaging servers (ActiveMQ, SonicMQ) irrespective of the vendors.
10. 
11. 
12. If we are aware of any JEE Standard such as JDBC, then JMS is for messaging where JDBC is for DB.