**Multi-tier architecture**

Monolithic:

* Presentation layer: to present to end customer
  + Web Application:
    - HTML5: new tags ?semantic tags?
    - CSS3: 🡺responsive design
      * Media queries
      * Bootstrap is CSS3 frame work(12 grid model)
      * Preprocessing 🡺SaaS, less 🡺 to dynamic preprocess the CSS.
    - JavaScript: can use it in front end and in middle ware.
      * Frameworks: jQuery, Angular, React
      * Libraries: Grunt and Gulp( Node modules)(similar to Maven)
  + Mobile Application:
    - Hybrid and Native
    - App containers 🡺SDK 🡺application code
    - Hybrid application returns HTML,CSS JavaScript
    - Native applications returns swift, Objective C for Ios and Kottlina for Android.
    - React native: uses JavaScript to develop native mobile apps.
    - Angular Ionic: builds hybrid apps.
* Back-end:
  + Servlet and JSP used to send HTML response to client.
  + Application server: primary responsibility is to create thread for each request and maintains servlet life cycle.
  + Two frames in backend:
    - First frame has frameworks like Spring and Struts: provides libraries and architectural style.
    - Second Frame has frameworks like Hibernate and Ibatis.
  + Web Server: It loads all static web pages to client. All other requests gets JSON and XML from backend.
  + After 2nd request proxy is used.
  + AJAX?
  + Webservices are used to respond with XML or JSON.

Where all service side code(classes, JSP code and everything) is bundled in one EAR/WAR file and deployed on app server.

One small thing can bring down the application.

One feature release depends on the other features code and delay release times.

Scalability cannot be done in this architecture. Even though one feature is not used much, you have to scale it with along the more used features because whole application is tightly coupled.

Team coordination will become the important aspect.

Microservice:

* They are independent components and self-contained(it has its own app server) and bounded context(should not keep all features, should be bounded to single feature).
* There will be a governance model and architecture team who knows about all services.
* Horizontal scaling: increasing instances according to the requirement.
* Vertical scaling: increase resources according to requirement.