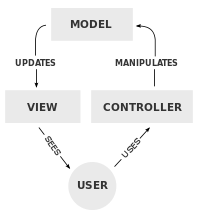
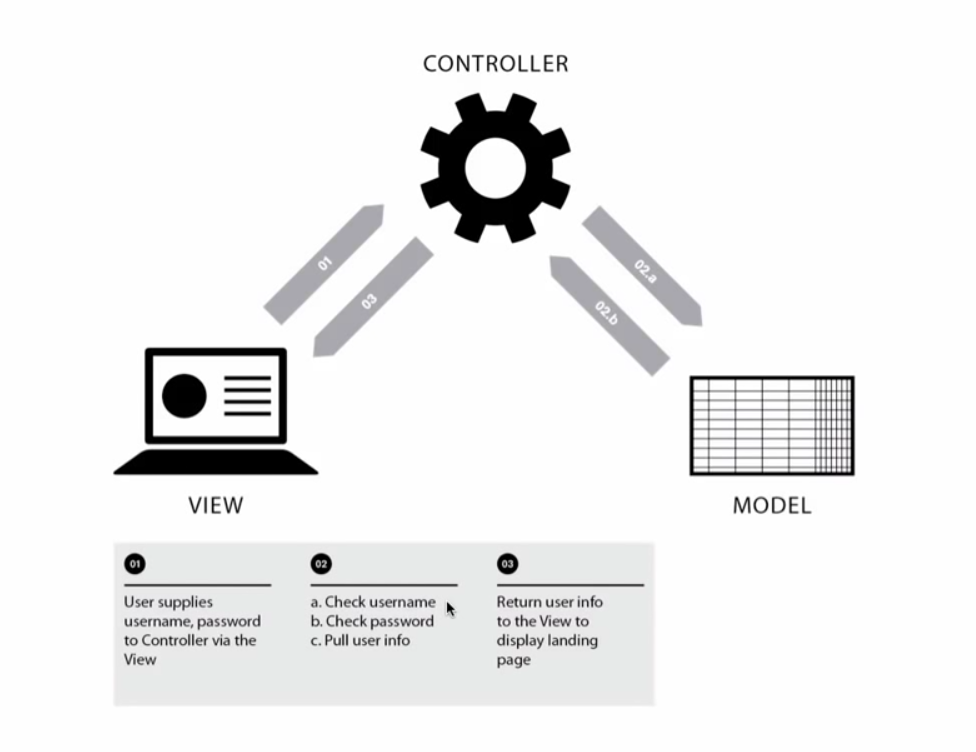
**Model View Controller** : Model on which is based Angular: “software architectural pattern for implementing user interfaces” (Wikipedia). It is a software pattern for implementing user interfaces. It divides a given software application into three interconnected parts to separate internal representations of information from the ways that information is presented to or accepted from the user. The *model*, consists of application data, business rules, logic and functions. A *view* can be any output representation of information (multiple views are possible). The *controller*, accepts input and converts it to commands for the model or view.



The model is where the data lives. It encompasses all the classes and objects as well as the database. A controller is what is going to react to an input from the client (eg: username & password) and adapt the software behavior accordingly.



Controller: CEO, he adapts the company output to the market

Model: The dev and R&D team, conceptualize and store the goods.

View: Design team and marketing team, design how the product will be displayed.

**Data Binding**: Process through which, when the UI is changed, the data will reflect this change (ex: live chat). It often refers to the immediate update of a UI based on input.

**Dependency injection**: in object-oriented computer programming is a technique that indicates to a part of a program which other parts it can use, i.e. to supply an external dependency (i.e. a reference) to a software component. In technical terms, it is a design pattern that separates behavior from dependency resolution, thus decoupling highly dependent components. Software procedure through which you separate parts of code, so that instead of A saying “I have the power to create an instance of B (eg: the design of the page) and only I have it”, it will refer to a third party script and say “yeaaah, so please create me an instance of B, you know, just put it on my desk, I don’t need to know how you did it.” A doesn’t need to know how B is created; everybody sticks to their scope of action and this makes modifying parts of code safer and easier. A class that creates a service should not be the one that also creates the design, how this service will be displayed to the client. It’s a fancy way of saying: “pass on the dirty work to a dedicated function” For example, In Angular, you inject the data obtained somewhere else (service) into the controller when he needs it. It’s not the controller’s job to go look for the information.

Yeoman is a great tool for developing apps in Javascript.

**Controller**

Here is, as an example, how a webpage can look with different controllers assigned to different chunks of your view



And here is the javascript counterpart

