**Angular Vs Veu.js Vs React.js**

To understand angular, veu and react we have to first understand what each individual javascript library do. Angular is handy where there is two-way binding, component and controller structure of the app. Veu.js is handy to develop frontend of the apps it barely focuses on the two way bind then angular. React.js is mainly about the frontend, it focuses on component structure of the app rather than how data is coming from the backend, that's why it has one-way binding, though two-way binding can also be achieved using mixins.

**Now what actually is One-Way Binding and Two-Way Binding ?**

**One-Way Binding**

One-Way Binding refers to the variables which can be mapped only on the template i.e can't be used for validation purpose.

**Two-Way Binding**

Two-Way Binding refers to the variables which can be mapped on the template and in the script and can be used for validation purpose or send data to the server.

**Component and Non Component Structure**

Usually there is a confusion between the component and the non component, but actually they are quite easy to understand.

Component Structure can be understood as the module which contains submodule which performs a certain task, like:

<root>

<task\_1>

<task\_1\_1>

</task\_1\_1>

</task\_1>

<task\_2>

<task\_2>

</root>

Usually an app is broken down into many components so that they can be easily designed and recompiled with less headache.

Non component structure can be understood as a single module which performs all the tasks,like:

<my-app>

</my-app>

here my-app will perform all the tasks

Note: Component structure can have both one-way binding and two-way binding.

**Task Given:**

Make a simple form which asks user to input firstname, lastname and email, store it in the table and perform edit and delete functions using different JS framwork(angular, veu.js and react.js)

To solve the given task, our first approach is to think whether to finish the task using component or non component structure but since the task given is very small so it can be completed using non-component structure but it can also be completed using component structure(easy to maintain, resuable code).

Our next approach is clear we have to use two-way binding because we need to interact with the form and store its data in the form of table.

Task Completed using Angular 1.X

The two-way binding in angular is very rich, any input field can be mapped to the variable using ng-model directive and can be accessed by using $scope.

In angular controllers are used to perform logical operations. Angular is also rich in directives like ng-show, ng-hide which can be used to show and hide elements dynamically. Also if we want to create variable during runtime, they can be created and we need not to register these in controller.

Overall angular is very handy framework for the given task, but with the availabilty of the so many directives, there might be a little bit of confusion.

Task Completed in Veu.js

Veu.js also supports for the two-way binding but it is not as rich as angular. Every element need to be registered before use. The syntax is quite similiar to that angular, so if one has a good grap on angular he/she can easily work on veu.js. The interesting thing about veu.js is the limited number of directive which are so designed that can implement most of the things in the two-way binding like v-\*for. These things are also available in the angular also but the thing is sometimes less is good for making such small apps.

Task Completed in React.js

To complete the task in react.js is the most difficult task because it has one-way binding but also supports for two-way binding using mixins. Every element need to the registered in the app before using just like veu.js, but as far as the task is concerned its difficult to implement two-way binding which mainly supports for one-way binding.

**Conclusion:**

Overall to complete the task veu.js is most easy because of lesser directives(less search in documentation for method lookup), clear examples are given how to use a certain directive.

But when it comes to build large scalable apps, angular will be choice.