When writing a front-end Javascript application you need to handle:

<table>
<thead>
<tr>
<th>State</th>
<th>Organisation</th>
<th>HTML generation</th>
<th>Testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keep track of dynamic data and user preferences</td>
<td>Organise code clearly</td>
<td>Add and remove HTML elements dynamically</td>
<td>Unit test logic and test user interactions work as expected</td>
</tr>
</tbody>
</table>

✓ state management and data-binding
✓ component based architecture
✓ HTML templating
✓ testing utilities

A framework can help by providing:

Key concepts:

2-way data-binding
A ViewModel (VM) reflects the state of the UI
- user actions update VM
- VM changes update UI

1-way data-binding
A pattern pioneered in React and known as Flux
- store updates component
- component updates UI
- state updates store
- user action updates store

Components
A component is like a custom HTML element with special behaviour attached:
<error-list errors="errArray">
  <li v-for="e in errors">
      {{e}}
  </li>
</error-list>

Timelines

Components

<error-list errors="errArray">
  <li v-for="e in errors">
      {{e}}
  </li>
</error-list>

Directives
A directive is like a custom data attribute on a regular HTML element, with special behaviour attached:

<error-list errors="errArray">
  <li v-for="e in errors">
      {{e}}
  </li>
</error-list>

The Big 3: Angular, React & Vue

<table>
<thead>
<tr>
<th>Directives</th>
<th>Components</th>
<th>Template Language(s)</th>
<th>2-way data binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>HTML</td>
<td>✓</td>
</tr>
<tr>
<td>X</td>
<td>✓</td>
<td>JSX</td>
<td>X</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>HTML, JSX</td>
<td>✓</td>
</tr>
</tbody>
</table>

React has the most weekly downloads 1.

Vue has the most GitHub stars 2.

Vue is the easiest to learn (And Angular the hardest)

Developers don’t like Angular as much 3.

React & Vue are more lightweight 4.
- min app bundle size
  - Angular: 32kb
  - React: 30kb
  - Vue: 65kb

React & Vue are slightly faster than Angular 4.
(but they're all pretty fast!)

Do I really need a framework?
If your app is not very interactive the costs may outweigh the benefits:
- steep learning curve
- slow to develop
- lots of ‘boilerplate’ code

1. https://www.npmjs.com/
2. https://github.com/