Storybook Framework Support

One of Storybook's most powerful aspects is that it's architected to support any web framework. It supports React, Vue, Angular, Web Components, Svelte and over a dozen others. This *work in progress* document helps you get started on adding new framework support for Storybook.

🏛️ Framework architecture

Storybook's framework support consists of two main aspects:

1) Babel/webpack config
2) Story rendering

❏ *TODO: better overview*

🏗️ Scaffolding a new framework

The first thing to do is scaffold your framework support in its own repo.

We've done this recently in the AEM project, and we recommend adopting the same project structure. That structure is a monorepo that contains the framework package ("app/<framework>") and an example app ("examples/<framework>-kitchen-sink") as well as other associated documentation and configuration as needed.

This structure mirrors the way Storybook's own monorepo is structured, and will make it easier to move the project into the monorepo at a later point if that is desirable.

We recommend using storybook/html as a starter framework since it's the simplest one and doesn't contain any framework-specific oddities. There is a boilerplate to get you started here:

https://github.com/CodeByAlex/storybook-framework-boilerplate

👷 Building out your framework

To build out your framework, there are two main tasks. (1) Building the “framework preset”, which is the basic babel/webpack for that framework that all users will get automatically when they use your framework. (2) Building the story rendering function that takes a “renderable object” and renders it to the screen.

⚙️ Framework preset

Storybook has the concept of presets, which are typically babel/webpack configurations for file loading. If your framework has its own file format, e.g. “.vue,” you might need to transform these files into JS files at load time. If you expect every user of your framework to need this, you should add it to the framework. So far every framework added to Storybook has done this, because Storybook’s core configuration is very minimal.

For more information about presets, see the presets documentation, Storybook’s presets repo, or any of the existing frameworks supported by Storybook.

- ❏ *TODO: example of a framework preset*

🖼️ Render function

Storybook stories are ES6 functions that return a “renderable object.” Consider the following React story:

```
import { Button } from './Button';

export default { title: 'Button', component: Button }
export const Sample = () => (
  <Button label='hello button' />
);
```

In this case, the renderable object is the React element, `<Button .../>`.

In most other frameworks, the renderable object is actually a plain old javascript object. Consider the following hypothetical example:

```
import { Button } from './Button';

export default { title: 'Button', component: Button }
```

```
export const Sample = () => ({
  template: '<button label=:label />',
  data: { label: 'hello button' }
});
```

The design of this “renderable object” is framework-specific, and should ideally match the idioms of that framework.

The frameworks *render* function is the thing responsible for converting this renderable object into DOM nodes. This is typically of the form:

```
const rootElement = document.getElementById('root');

export default async function renderMain({
  storyFn,
}: RenderMainArgs) {
  const storyObj = storyFn();
  const html = fn(storyObj);
  rootElement.innerHTML = html;
}
```

- ❏ *TODO: detailed description of render function*

📣 Promoting your framework

Storybook has an active user community, and we are happy to help you promote your framework. Historically, we’ve done this by incorporating new frameworks into the Storybook monorepo. However, over time the monorepo has gotten prohibitively large, and we’ve started to ask contributors to build and release frameworks in their own repos. In the future we’ll provide a set of guidelines for how to contribute frameworks to the monorepo.

- ❏ *TODO: contribution & promotion guidelines*