# Learning to like frameworks

I've been vocal about frameworks ever since I started working with some of them a few years ago. Perhaps I'm an old curmudgeon who wants the kids to get off his lawn but I've sseen something like this happen with many of the frameworks I've tried and worked with over the years:

1. New framework launches
2. Everyone thinks the new framework is awesome and wants to use it
3. Training for the new framework covers only basic and few, if any, new users move beyond the basics.
4. Basic training and tutorials for most frameworks I've seen does not include accessibility training or considerations at the beginning level
5. Sites are not fit for accessibility because there are no users who needs accommodations using the site. ***There is never time to do it right but there's always time to do it over***
6. When we find out we need to make sites accessible either because of goveernment regulatios or as a result of a lawsuit, it becomes a much harder endeavor that is less appealing and potentially more expensive

Don't get me wrong. Learning a framework is awesome as it abstracts a lot of the basic boilerplate of building a web application. My problems are two-fold.

## Being less naive about how we use frameworks

Every so often I see questions in Quora that go something like this: ***should I learn a framework or plain Javascript first?*** or something to that effect. This shows, to me, the assumptions that beginners make and that we, as a community, don't bother to correct. That a framework use of Javascript is different than learning the basic language first.

In an ideal world we wouldn't have to answer this question because basic Javascript or how to do things without a framework or library before starting to work with the library. This would stabd you in a better place to learn any library or ddecide that a framework not needed.

But this is not the case, most people will not even consider stopping you from learning the framework if you don't have basic knowledge of Javascript. React has

the following in their tutorial page:

> We'll assume some familiarity with HTML and JavaScript but you should be able to follow along even if you haven't used them before.
>
> If you need a refresher on JavaScript, we recommend reading this guide. Note that we're also using some features from ES6, a recent version of JavaScript. In this tutorial, we're using arrow functions, classes, let, and const statements. You can use Babel REPL to check what ES6 code compiles to.

So the tutorial tells you that it will use Javascript and what parts of ES6 it will use but sends you to external resources to optionally follow up on them. If not, it's ok too, you should be able to tell what the code does and you should be able to move through the tutorial.

At the most basic level, the level covered in most introductory tutorial, this is true. We can move through the tutorial by guessing what each tag does. For example, in the code below, we create nested elements using `React.createElement` and adding attributes and classes where appropriate.

```
React.createElement(
  "div",
  { className: "shopping-list" },
  React.createElement("h1", null, "Shopping List for ", props.name),
  React.createElement("ul", null,
    React.createElement("li", null, "Instagram"),
    React.createElement("li", null, "WhatsApp"),
    React.createElement("li", null, "Oculus")
  )
);
```

Far less intuitive is how we'd use the component in a page. It looks something like this. Pay attention to the name property. We'll use to populate who the list is for.

```
<shopping-list name="Carlos">
```

And the result looks like this. React will take the name from the attribute and use it to replace `props.name`.

```
<div className="shopping-list">
  <h1>Shopping List for carlos</h1>
  <ul>
    <li>Instagram</li>
    <li>WhatsApp</li>
    <li>Oculus</li>
  </ul>
</div>
```

But as we get deeper into React I've discovered that you need more and more Javascript and the way in which we combine all three doesn't really lend itself to getting through the examples and demos without actually knowing Javascript.

In the same tutorial we start seeing examples of classes extending `React.Component` and creating elements are not stricly HTML:

```
class Square extends React.Component {
  render() {
    return (
      <button className="square" onClick={() => alert('click')}>
        {this.props.value}
      </button>
    );
  }
}
```

I've written about Javascript classes and know enough to figure out that class and extend are part os ES6/2015 but if you look at it and essentially cut and paste the example like we used to do with jQuery how can you tell what this means and how it works?

And there, in my opinion, lies the problem. Like jQuery back in the day, people assumed that they only had to learn jQuery and not Javascript. Then when jQuery fell short they had to discover the hard way what Javascript was and how to leverage the raw language to accomplish their goals.

Perhaps this is my problem and not frameworks… true. Perhaps it has to do with the fact that no matter how many frameworks we see in the market there is always something else and I'm always afraid that we'll forget the basics just for the sake of the "cool factor".

## Accessibility

Accessibility is the step child of the web. It is one of the aspects that seldom, if ever, gets considered during initial development. While most frameworks will produce HTML it will not always produce HTML that will works well with assistive technology such as screen readers or when working with only the keyboard. This is particularly true when working creating link-like elements or changing the intended use of an existing element.

# developer learning curve

# framework and library interactions

- https://www.smashingmagazine.com/2015/05/client-rendered-accessibility/
- https://www.smashingmagazine.com/2015/03/web-accessibility-with-accessibility-api/
- https://www.smashingmagazine.com/2017/02/designing-html-apis/
- https://alistapart.com/article/let-links-be-links
- https://www.w3.org/TR/using-aria/
- http://unobfuscated.blogspot.com/2015/03/polymer-and-web-component-accessibility.html
- https://docs-05-dot-polymer-project.appspot.com/0.5/articles/accessible-web-components.html
- https://www.paciellogroup.com/blog/2014/09/web-components-punch-list/