Scope of Work

Charlie McVicker and MathJax December 2019

Objective

React.js (React) is a popular MVC library for creating front-end experiences. It is focused on the ideolgy of "reactive" programming and makes heavy use of a "virtual DOM." A React webapp consists of many "components" with have, amoung other things, "states" and "properties." A component's state represents its interal owned data. A component's properties are passed down to it, like HTML attributes. Unlike HTML attributes, they can be any JavaScript object. Any time the state or properties of a component change, a function called "render" is invoked on the component by the React library. The render function will return new React virtual DOM nodes, representing how the component should now render, given the change. In the React virtual DOM, elements can be either plain HTML elements or React components. It is common for components closer to the root level to be more stateful, and to pass that state to children components through properties. Stateful components are often called "smart" compared to "dumb" children components, which are defined only by their properties. All components have a property called "children" representing the components/elements contained within them. Because interaction with the virtual DOM is so important, React is often written in a JavaScript dialect called JSX, which allows for inline HTML-like syntax for creating virtual DOM nodes, with the ability to insert JavaScript easily.

I (Charlie McVicker) will design, build, and deploy a React component library (the Project), which exposes typesetting functionality from MathJax.js (MathJax) to the React ecosystem. The Project will be kept in source control under my personal GitHub account (`https://github.com/CharlieMcVicker`).

Deliverables

1. A bundled NPM package that can be served from the NPM registry or another server, containing a single component (MathComponent?). This component:

   - Can be imported easily into the standard NPM/React/Webpack workflow
   - Renders (inline or display) TeX, MathML, or other input to SVG, PNG, CHTML, etc. (provided I/O Jax exist)
   - Accepts a property containing TeX, MathML, etc. and renders via MathJax an HTML representation of the typeset content. This HTML is stored in the state, and injected into the React virtual DOM.

   Other possible features (time willing) include:

   - Supporting Typescript style imports with type declarations (`.d.ts`)
   - Efficient usage of Webpack aliasing/externals to minimize package size
   - Releases for non-managed (ie. no bundler and plain JavaScript) workflows

2. Another component (MathSection?), bundled with the first or served separately, which can integrate with a Jax's `findMath` logic to keep track of inline math and manage reactive updates in larger bodies of text, containg only some typeset sections. The MathSection could be used on a large prose section of text containing some MathJax.

- Built on-top the other component and supporting the same bundling and development features.
- Make use of the "children" propety to wrap prose text and allow for using MathJax wrappers ($$, \(\), etc.).
- Hopefully allows developers to interact with MathJax in the ways in which they are accustomed.

Schedule

Work can begin upon request and will run until the Project is deployed, stopping early if progress or timing is unsatisfactory, and continuing if responding to Issues, Feature, and Pull Requests is desired.

Price

Work will be done at a rate of $40 per hour. I will track hours in weekly statements, containing the approximate allocation of worked hours to deliverables and objectives. Whenever an appropriate amount of work is complete, I will file an invoice through NumFOCUS's system. A soft or hard limit can be placed on hours worked per week or month. Should it then be exceeded, I would stop work and notify a MathJax representative.

Core Assumptions and Risks

1. React, TypeScript, Webpack, and NPM are platforms of variable support and reliability. Properly and efficiently bundling packages can be slow, and sometimes is outside of my scope. Some functionality, such as TypeScript-React interoperability is currently an unknown for me and could take any amount of time. For that reason, I've left it as a possible extension.

2. I have only one professional React project under my belt. I will focus on designing the Project with best practices but may not always capture the current culture and expectations of the community.

3. On a similar note, I am also unaware of the feature set desired by the Project's eventual user-base. It is very possible iteration will be required to meet the needs of the community.

4. The typesetting process used in the proof-of-concepts is currently synchronous. This can lead to shuttering. Options to remedy this include:

    - Finding async/promise bindings for the MathJax core (I think I may have just missed them)
    - Using a service worker to prevent thread blocking (I think the former is better)