# HogWild: The React app for fans of prize-winning pigs

[(https://github.com/learn-co-curriculum/react-hooks-hogwild)](https://github.com/learn-co-curriculum/react-hooks-hogwild) [(https://github.com/learn-co-curriculum/react-hooks-hogwild/issues/new)](https://github.com/learn-co-curriculum/react-hooks-hogwild/issues/new)

## React Week 1 Project

## Deliverables

- *When the app first loads*, display a tile for each hog in the `porker_data.js` file. In the tile, display the **name** and **image** for each hog.
- *When the user clicks on the hog tile*, display the other details about the hog (its **specialty**, **weight**, **greased**, and **highest medal achieved**)
- Allow the user to *filter* the hogs that are **greased**
- Allow the user to *sort* the hogs based on **name** or **weight**
- BONUS: Allow users to *hide* hogs (not delete them, just hide them from view!)
- BONUS: Add a form to allow users to *add* new hogs to the page
- BONUS: Implement **Semantic Cards** (https://semantic-ui.com/views/card.html) for each hog

## Project Guidelines

- Follow **React best practices** (https://reactjs.org/docs/thinking-in-react.html) to create components and decide where state needs to live
- Pass props down from parent components to children
- Use inverse data flow and callback functions to pass data up from child components to parents
- Re-render components by setting state

## What we have so far

- A file containing all our hog data ( `./src/porkers_data.js` ) imported into `App.js`
- A `<Nav>` component rendered in our `App.js`

## Trying to figure out where to start?

There are lots of ways to build this project, and while some ways are better than others, there is no 'right' way! Start by wireframing what you want the app to look like and breaking it up into components.

One good model to follow for this is **Thinking in React (https://reactwithhooks.netlify.app/docs/thinking-in-react.html)** .

Once you've decided on your components, use the MVP (minimum viable product) approach. What's the simplest thing we can render to the page? Perhaps a paragraph tag displaying each hog's name? Which components would this involve?

When building your filter and sort functionalities, consider what you want to store in state and where that state should be stored. How can a child component pass information up to its parent component? Think about what needs to happen upon each index rerender. What if a user filters out un-greased pigs, and then wants the remaining pigs sorted by weight?

Be sure to use good programming practices, such as clear variable names and single responsibility functions. React apps can quickly become tangled and hard to debug if built without best practices!

## Styling

We've imported the Semantic CSS library to keep your piggies looking pretty. To keep your hogs in columns, make sure their parent container has the class `"ui grid container"` . The children in the columns should have class `"ui eight wide column"` . (Semantic uses a grid with a default of 16 units wide, so 8-wide will make two columns and 4-wide will make 4 columns.)

Semantic will take care of assigning the columns for you. You can also try implementing **Semantic Cards (https://semantic-ui.com/views/card.html)** for each hog.