Lendflow Frontend Assessment

Please spend no more than a couple hours working through the problems below. Return code back to us in whatever way is most convenient for you.

1. Watching an Object

Using TypeScript or vanilla Javascript *(no 3rd party libraries/packages allowed)*, create a `watch()` function with the following signature:

```
watch(
  obj: Object
  getCallback: Function(key, value) | undefined
  setCallback: Function(key, value) | undefined
)
```

The returned value should allow inspection & mutation of `obj` , and will:

- call (if available) any time an **existing** property on `obj` is accessed.
- call (if available) any time a property is set or updated on the underlying object.

An example of its usage/behavior might look something like this:

```
const obj = {};

const watchedObj = watch(
  obj,
  (key, value) => console.log('property accessed: ', key),
  (key, value) => console.log('property modified: ', key, value)
)

watchedObj.foo; => // no watch called, property doesn't exist
watchedObj.foo = true; => property set: foo
watchedObj.foo; => property accessed: foo
```

2. TypeScript Type Definition

Given a TypeScript interface that defines an existing Post from an API:

```
interface post {
  id: number;
  title: string;
  const: string| null;
}
```

Define a `type NewPost` based off of this Post interface which makes id optional. We're looking for a type that utilizes the Post above to effectively create NewPost as shown below:

```
interface NewPost {
  id?: number;
  title: string;
  content: string| null;
}
```

In short, complete the definition:

```
type NewPost = ....
```

3. JSON Manipulation

Using the attached `products.json` file as source input, please provide code to answer each of these questions as succinctly as possible.

1. Which products are out of stock, **not** on sale, and under $20?
2. What is the most commonly used category?
3. What is the average price of sale items?
4. How many women's products are out of stock, broken down by color?

4. Card Visibility

1. Unzip the attached `scroll-cards.zip`.
2. Run `npm ci && npm run dev`
3. Open `src/App.vue`

The `src/Card.vue` component has an `isInFocus` prop.

Please add code to `App.vue` to set this isInFocus on a single card, respecting the following conditions:

- a card is considered eligible for focus only when it's text is within the viewport
- by default (on load), the topmost card eligible for focus should be focused
- when scrolling **down** the page (as cards move into the viewport from the bottom), the **topmost** card eligible for focus should be focused
- when scrolling **up** the page (as cards move into the viewport from the top), the **bottommost** card eligible for focus should be focused