Future Improvements

There are only four rockets being returned in the query for "rockets" from SpaceX.
It would be interesting to see how this scales with 20+ results coming from the API.
It would make sense to incorporate some form of pagination or "infinite scroll"/"load more button" to account for this.

It would also be interesting to test this instead on the "launches" data, as there would be more fields to sort and filter.

One significant drawback in the code is how [not] extendable the sorting and filtering are.
It's relatively easy to add another button/click handler, an enum to represent the attribute, and call `updateFilterState` or `updateSortState`,
but the corresponding code in `sortRockets` and `filterRockets` is not very maintainable as filters & sort orders grow.

The available filters and sort orders are hardcoded and are creating duplicated code.
Some examples would be the the `switch` statement in `sortRockets` and the guard clauses inside `filterRockets`:

```
export function sortRockets(rockets: Rocket[], sortOrder: RocketSortState) {
  return [...rockets].sort((rocket1, rocket2) => {
    if (sortOrder.order === SortOrder.ASCENDING) {
      switch (sortOrder.attribute) {
        case SortAttribute.COST:
          return rocket1.cost_per_launch - rocket2.cost_per_launch;
        case SortAttribute.NUMBER_OF_ENGINES:
          return rocket1.engines.number - rocket2.engines.number;
        default:
          return rocket1.id.localeCompare(rocket2.id, 'en');
      }
    } else {
      switch (sortOrder.attribute) {
        case SortAttribute.COST:
          return rocket2.cost_per_launch - rocket1.cost_per_launch;
        case SortAttribute.NUMBER_OF_ENGINES:
          return rocket2.engines.number - rocket1.engines.number;
        default:
          return rocket2.id.localeCompare(rocket1.id, 'en');
      }
    }
  });
}
// and
export function filterRockets(rockets: Rocket[], filters: RocketFilterState) {
  if (isEmpty(filters)) {
    return rockets;
  }
  return rockets.filter((rocket) => {
    return Object.keys(filters).every((filter) => {
      if (filter === RocketFilter.ACTIVE) {
        return rocket.active;
      }
      if (filter === RocketFilter.MERLIN_ENGINES) {
        return rocket.engines.type === 'merlin';
      }
      return false;
    });
  });
}
```

For the filters, `filterRockets` is not great at selecting nested properties, like `engines.type`.
This is why I lazily added in guard clauses for each filter enum.
The ideal would be to make this more generic (`rocket[filter]`).

For the sorting, the `switch` statements will become unacceptable as the number of sort orders grows.
One way to make this easier is to conform all the types to a single type and do a comparison.
For example, I could convert all the attribute types to a `string` and use `localeCompare` (and setting the `numeric` option when comparing numbers).

UI/UX Improvements

The title and spaceship are shifted to the right between the "Loading rocket data…" state and when the rocket grid is displayed.
It would be interesting to investigate what is causing this, perhaps something related to padding/margins.

It would be nice to show the rocket `Card`s fade in and out when being filtered instead of abruptly disappearing.

I would like to implement a toggle between a grid (current) layout and a list layout.
This would become more useful the more items there are to be listed, instead of 4 large tiles taking up the viewport.

Domain Knowledge

I was confused why I had to create a copy of the array (`[...rockets]`) in order to call `.sort()` in `sortRockets`.
The TypeScript compiler was complaining that:

```
Cannot assign to read only property '0' of object '[object Array]'
```

I spread the contents of `rockets` into a new array in order to manipulate them.
My guess is `rockets` is flagged as `readonly` because React props are marked as `readonly` and `rockets` are part of `RocketLookupProps`.
This would make sense, because `.sort()` sorts the array in-place, meaning it is modifying the existing `rockets`.
`.filter()` does not do this, it iterates through `rockets` and returns a new array (therefore no TS error).