Solution Review

Review with explanations. (3 min. read)

assoc

We learned that a pure function must follow 2 rules

1. Same inputs => same outputs
2. No side-effects

`assoc` currently mutates the input object, which is a side-effect.

```
const assoc = (key, value, object) => {
  object[key] = value;
};
```

Cure it by cloning the input and merging the desired properties.

```
const assoc = (key, value, object) => ({
  ...object,
  [key]: value
});
```

getNames

```
const getNames = (users) => {
  console.log('getting names!');

  return users.map((user) => user.name);
};
```

This was sort of a trick question. `getNames` is pure if you remove `console.log`.

```
const getNames = (users) => {
  return users.map((user) => user.name);
};
```

You can now refactor it to a single-line statement, if you prefer.

```
const getNames = (users) => users.map((user) => user.name);
```

append

```
const append = (item, list) => {
  list.push(item);

  return list;
};
```

This is impure because it mutates the input list. Like `assoc`, you can return a clone with the desired output.

```
const append = (item, list) => [...list, item];
```

sortAscending

```
const sortAscending = (numbers) => numbers
  .sort((a, b) => a > b);
```

Once again we're using an impure Array method, `sort`. Cloning the input purifies us right up.

```

```

```
const sortAscending = (numbers) => [...numbers]
  .sort((a, b) => a > b);
```