

Handling Selection Logic

We're almost done with this set of changes. The last thing to add for now is the ability for the user to click on either of the lists and select the item that was clicked on. Right now, we're just defaulting to using the first item in an array as the "current" item for display in the Details sections.

We'll start with the Pilots list. We don't have a reducer for anything pilot-related yet, so we'll create one. Going along with the idea of "normalization", all we need to store is the ID of the currently selected pilot. We'll actually get a bit fancy with the reducer logic, and handle de-selecting the current item entirely if the user clicks on it again:

Commit 593e570: Add logic for tracking the currently selected pilot

[features/pilots/pilotsReducer.js](#)

```
import {createReducer} from "common/utils/reducerUtils";

import {PILOT_SELECT} from "../pilotsConstants";

const initialState = {
  currentPilot : null
};

export function selectPilot(state, payload) {
  const prevSelectedPilot = state.currentPilot;
  const newSelectedPilot = payload.currentPilot;

  const isSamePilot = prevSelectedPilot === newSelectedPilot;

  return {
    // Deselect entirely if it's a second click on the same pilot,
    // otherwise go ahead and select the one that was clicked
    currentPilot : isSamePilot ? null : newSelectedPilot
  };
}
```

```

    currentPilot : isSamePilot ? null : newSelectedPilot,
  };
}

export default createReducer(initialState, {
  [PILOT_SELECT] : selectPilot,
});

```

That gives us some data handling, but we need to hook that up to the UI. We need to pull the `currentPilot` ID value into `<Pilots>`, and use that in a couple places. We should pass the actual entry for the current pilot into `<PilotDetails>`, and it would also be nice to highlight the row for that pilot in the list. We also need to call the `selectPilot` action creator with the ID of the pilot whose row was just clicked on. Let's look at the relevant changes:

Commit 9062899: Implement selection handling for pilots

features/pilots/Pilots.jsx

```

// Omit initial imports

+import {selectPilot} from "../pilotsActions";
+import {selectCurrentPilot} from "../pilotsSelectors";

const mapState = (state) => {}
  // Omit pilot objects lookup

+  const currentPilot = selectCurrentPilot(state);

  // Now that we have an array of all pilot objects, return it as a prop
-  return {pilots};
+  return {pilots, currentPilot};
}

+// Make an object full of action creators that can be passed to connect
+// and bound up, instead of writing a separate mapDispatch function
+const actions = {
+  selectPilot,
+};

export class Pilots extends Component {
  render() {

```

```

-     const {pilots = []} = this.props;
+     const {pilots = [], selectPilot, currentPilot} = this.props;

-     const currentPilot = pilots[0] || {};
+     const currentPilotEntry = pilots.find(pilot => pilot.id === currentPilot) || {}

    // Omit irrelevant layout component rendering for space
    return (
      <Segment>
-        <PilotsList pilots={pilots} />
+        <PilotsList
+          pilots={pilots}
+          onPilotClicked={selectPilot}
+          currentPilot={currentPilot}
+        />
-        <PilotDetails pilot={currentPilot} />
+        <PilotDetails pilot={currentPilotEntry} />
      </Segment>
    );
  }
}

-export default connect(mapStateToProps)(Pilots);
+export default connect(mapStateToProps, actions)(Pilots);

```

features/pilots/PilotsList/PilotsList.jsx

```

export default class PilotsList extends Component {
  render() {
-    const {pilots} = this.props;
+    const {pilots, onPilotClicked, currentPilot} = this.props;

    const pilotRows = pilots.map(pilot => (
-      <PilotsListRow pilot={pilot} key={pilot.name}/>
+      <PilotsListRow
+        pilot={pilot}
+        key={pilot.name}
+        onPilotClicked={onPilotClicked}
+        selected={pilot.id === currentPilot}
+      />
    ));

    // Omit layout rendering for space
  }
}

```

```

+import _ from "lodash";

-const PilotsListRow = ({pilot={}}) => {
+const PilotsListRow = ({pilot={}, onPilotClicked=_.noop, selected}) => {
  const {
+    id = null,
    // Omit other fields
  } = pilot;

  return (
-    <Table.Row>
+    <Table.Row onClick={() => onPilotClicked(id)} active={selected}>

```

And voila! If we click on an entry in the pilots list, we should now see that row highlighted, and the entry also shown in the `<PilotDetails>` form:

Project Mini-Mek

Unit Info

Pilots

Mechs

Unit Organization

Tools

Pilot List

Name	Rank	Age	Skills	Mech
Natasha Kerensky	Captain	52	2/2	WHM-6R
Colin Maclaren	Sergeant	43	3/4	MAD-3R
Lynn Sheridan	Corporal	27	4/5	CRD-3R
John Hayes	Sergeant	34	3/4	GRF-1N
Takiro Ikeda	Lieutenant	41	3/4	ARC-2R
Miklos Delius	Corporal	31	4/4	ARC-2R
Nikolai Koniev	Private	39	3/4	WSP-1A
Alex Ward	Corporal	36	4/5	STG-3R
John Clavell	Lieutenant	40	3/4	RFL-3N
Piet Nichols	Corporal	37	4/5	PXH-1K
Simon Fraser	Sergeant	32	3/4	STG-3R
Mohammar Jahan	Corporal	29	3/5	STG-3R

Pilot Details

Name

Lynn Sheridan

Rank

Corporal

Age

27

Gunnery

4

Piloting

5

Mech

And finally, the last thing we'll do for this section is implement the same behavior for the mechs list as well.

Commit 8471119: Add logic for tracking the currently selected mech

Commit 0fce867: Implement selection handling for mechs

And we see the same nicely highlighted selection for the mechs list:

Project Mini-Mek

Unit Info

Pilots

Mechs

Unit Organization

Tools

Mechs List

IDs	Name	Model	Weight (tons)	Class
1	Warhammer	WHM-6R	70	Heavy
2	Marauder	MAD-3R	75	Heavy
3	Crusader	CRD-3R	65	Heavy
4	Griffin	GRF-1N	55	Medium
5	Archer	ARC-2R	70	Heavy
6	Archer	ARC-2R	70	Heavy
7	Wasp	WSP-1A	20	Light
8	Stinger	STG-3R	20	Light
9	Rifleman	RFL-3N	60	Heavy
10	Phoenix Hawk	PXH-1K	45	Medium
11	Stinger	STG-3R	20	Light
12	Stinger	STG-3R	20	Light

Mech Details

ID

4

Name

Griffin

Model

GRF-1N

Weight

55

Class

Medium

Let's see how the complete app behaves in action so far:

```
.App-header {
  background-color: #222;
  height: 70px;
  padding: 20px;
  color: white;
}
```