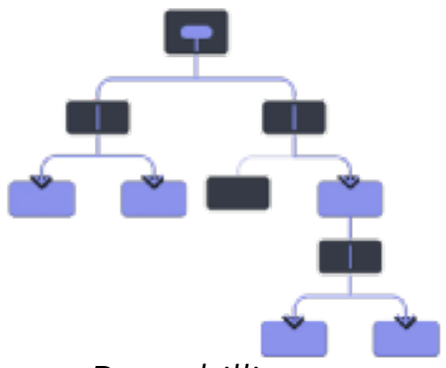
GESTION D'ÉTATS

ELM / CONTEXT

Pour éviter la complexité liée à Redux dans un premier temps, on peut implémenter
la logique de la Elm Architecture à l'aide de l'API **`Context`** et d'un hook **`useReducer`**

Un contexte permet d'éviter le « prop drilling » et ainsi de pouvoir utiliser des données facilement, partout dans notre application.

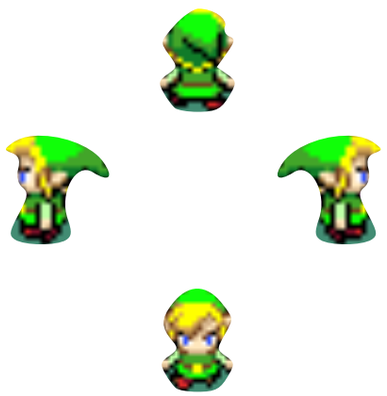


Prop drilling

```
const initialModel = {};
const ModelContext = createContext(null);
const SendMessageContext = createContext(null);

const update = (model, message) => {
  switch (
    message.type
    // Describe state machine here
  ) {
  }
  return model;
};

export const ModelProvider = ({ children }) => {
  const [model, sendMessage] = useReducer(update, initialModel);

  return (
    <ModelContext.Provider value={model}>
      <SendMessageContext.Provider value={sendMessage}>
        {children}
      </SendMessageContext.Provider>
    </ModelContext.Provider>
  );
};
```

STATE MACHINE STRIKE BACK

SI ON ADAPTAIT NOTRE MACHINE



```ts
interface North {
    type: "north";
}
interface East {
    type: "east";
}
interface South {
    type: "south";
}
interface West {
    type: "west";
}

type Direction = North | East | South | West

const label = (d: Direction) => {
    switch (d.type) {
        case "north": return "North"
        case "east": return "East"
        case "south": return "South"
        case "west": return "West"
    }
}
```