React Keys CheatSheet

by Ndeye Fatou Diop

What Are Keys?

- Keys are unique identifiers used by React to uniquely identify **component instances** (i.e. a specific, rendered versions of a React component in memory, holding its current state, props, and lifecycle methods during its existence in the app.)
- They help React efficiently update and reconcile the DOM by identifying which items have changed, added, or removed.

When Are Keys Required?

- Keys are required when rendering a collection of elements with **.map()** or other iteration methods.

```
const items = ["A", "B", "C"];

return items.map((item) => {
   return <div key={item}>{item}</
div>;
});
```

How Keys Work

- React uses keys to:
  - Match elements between renders.
  - Reuse or unmount DOM nodes as efficiently as possible.
- Without keys, React falls back to inefficient re-renders of all child components.

What to Avoid

- **Random Keys Changing on Every Render:**
  - Don't use random keys like Math.random() in render:

```
items.map((item) => {
  return <div
key={Math.random()}>{item}
</div>;
});
```

- **Using Index as Key in Mutable Lists:**
  - If the list order or content changes, keys based on indices can lead to bugs.

```
items.map((item, index) =>
{
  return <Item key={index}
item={item} />;
});
```

- **Abusing of Keys:**
  - Don't carelessly use keys to remount items since this can cause perf issues.

When to Use Keys

- **Dynamic Lists:** Use keys to help React track individual items.

```
return todos.map((todo) => {
   return <Todo key={todo.id}) todo={todo} />;
});
```

- **Force ErrorBoundary Reset:** Changing a key forces React to unmount and remount the component.

```
<ErrorBoundary key={userId}>
  <UserProfile userId={userId} />
</ErrorBoundary>
```

- **Force Component Re-render:** Change the key to reset a component's state and lifecycle.

```
const [selectedType, setSelectedType] = useState("posts");
return (
  <>
    <Navbar type={selectedType}
onTypeChange={setSelectedType} />
    <Resource type={selectedType} key={selectedType} />
  </>
);
```

How to Generate Keys

- **Use Stable Identifiers:** Prefer stable, unique IDs from your data.

```
items.map((item) => {
  return <div key={item.id}>{item.name}</div>;
});
```

- **Generate unique keys:** You can generate keys using crypto.randomUUID(), Math.random or the uuid library.

```
// Fetch quotes and add ID
useEffect(() => {
  const loadQuotes = () =>
    fetchQuotes().then((result) => {
      // We add the `ids` as soon as we get the results
      setQuotes(
        result.map((quote) => ({
          value: quote,
          id: crypto.randomUUID(),
        }))
      );
    });
  loadQuotes();
}, []);
```