# Complex Interactivity

Session 04

## Forms and Input Fields

## Controlled vs. Uncontrolled

**Controlled Inputs**

- State is handled by React
- `value` and `onChange` Prop on `input`

Use when:

Handle each field individually

**Uncontrolled Inputs**

- State is handled by DOM itself
- `onSubmit` on `<form>`

Use when:

Handle each fields all together

## Controlled vs. Uncontrolled

### Controlled Inputs

```
const [name, setName] = useState("");

function onSubmit(event) {
  event.preventDefault();
  console.log("Name value: " + name);
}

return (
  <form onSubmit={onSubmit}>
    <input
      name="name"
      value={name}
      onChange={(event) => setName(event.target.value)}
    />
    <button type="submit">Submit</button>
  </form>
);
```

### Uncontrolled Inputs

```
function onSubmit(event) {
  event.preventDefault();
  const formData = new FormData(event.target);
  const fields = Object.fromEntries(formData);
  console.log("Name value: " + fields.name);
}

return (
  <form onSubmit={onSubmit}>
    <input
      name="name"
    />
    <button type="submit">Submit</button>
  </form>
);
```

# Controlled vs. Uncontrolled

| | **Controlled** | **Uncontrolled** |
|---|---|---|
| Pro | respond to each key stroke | no states required |
| Con | a lot of states to handle<br>(one state per input field) | respond only to form submit |

## Arrays of Objects in State

## Setup

Change names with

*highlights*

```
const initialTodos = [
  { id: "c92054d1dd6", title: "Make Dinner", completed: false },
  { id: "ac84bbb3728", title: "Clean up", completed: true },
];

function App() {
  const [todos, setTodos] = useState(initialTodos);

  // … more code
}
```

## Adding new Item

Change names with

*highlights*

Create a new Array.

Copy/Spread value of current state.

Add new object to array.

Generate random unique ID for new item.

Copy/Spread data of new item.

```
function addTodo( newItem ) {
  setTodos(
    [
      ...todos,
      { id: uid(), ...newItem }
    ]
  )
}

addTodo({
  title:"Go Shopping", completed: false,
});
```

# Editing/Updating an Item

Change names with

*highlights*

Map all existing items in current state.

Compare each item with ID of item to be updated.

If match:
Create new object and copy/spread data of existing item. Copy/spread data of update.

If not match:
Keep item without a change.

```
function updateTodo(id, updatedItem){
  setTodos(
    todos.map(
      (item) =>
        item.id === id
        ? {...item, ...updatedItem}
        : item
    )
  )
}

updateTodo(
  "c92054d1dd6"
  { completed: true }
);
```

# Delete an Item

Change names with

*highlights*

Filter all existing items in current state.

Compare each item with ID of item to be removed.

All items where ID does not match are kept.

```
function deleteTodo(id){
  setTodos(
    todos.filter(
      (item) => item.id !== id
    )
  )
}

removeTodo("c92054d1dd6");
```