# Common Patterns

The Web has many common UI patterns

- Those present in base HTML
    - bulleted lists, forms, etc
- Non-interactive, built from HTML
    - horizontal menus, "cards", etc
- Interactive, built from HTML
    - dropdown menus, accordions, modal

# Accordion Pattern

This UI Pattern is

- a collection of sections
- each with a title
- each can collapse/expand
- Sometimes only one/expanded at a time
- Sometimes only collapse by expanding another

# Accordion UI Pattern

**What to do about a catnip addiction?** +

**Will I ever catch that red dot?** −

Dolor dicta odio inventore ut qui dolorem. Omnis recusandae quo porro voluptatibus veniam? Sed earum rem nobis quasi odit, similique ducimus Voluptatibus fugit sint error?

**Why does the human have a problem with me claiming the center of the bed?** +

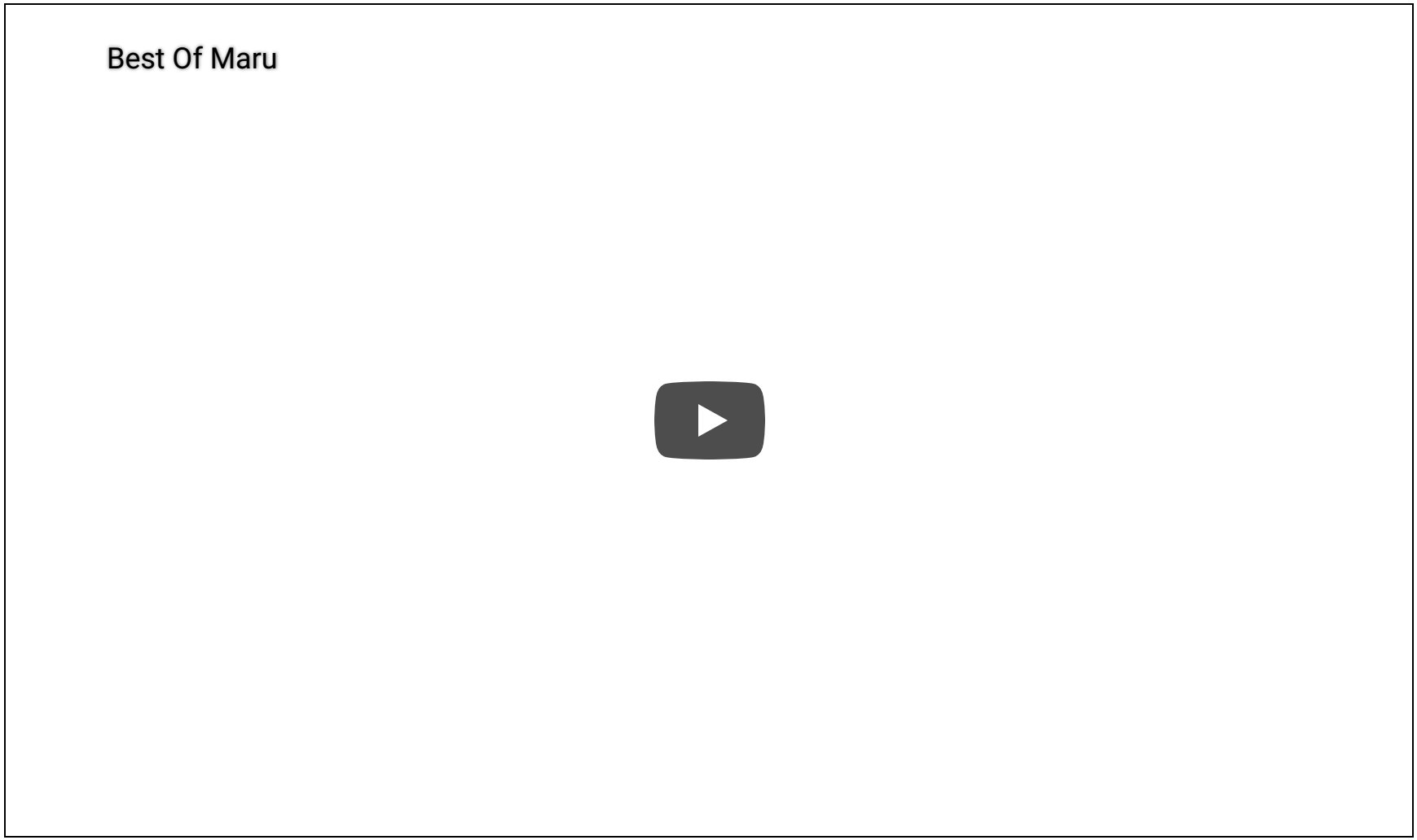
# Accordion considerations

- What (if any) limitations on expand/collapse?
- Both keyboard and mouse controls?
- Accessibility concerns?
    - Semantic HTML
    - Indicating open/close
    - More later, when we get to ARIA

# Implementing an Accordion in React

- We want a top level `<Accordion>` component
  - reusable
  - manages its own state
- Styling?
  - We will do as part of site CSS
  - Doing component specific CSS isn't wrong
- Structure?
  - Whole Container
  - Each entry container
  - Title (needs to be clickable)
  - open/close indicator ( +/-, arrow, etc)

# On-hold video while starting a new React project



# Initial Setup

- .js to .jsx, css className
- Put the data in state
    - In most cases this would come from a service
        - We will just keep it separate
        - Doesn't impact the implementation
- Create the stub of a component
    - accepts the list of entries

# Why a Stub?

- Confirm the pieces talk to one another
- Confirm output is visible
    - renders
    - not hidden by CSS
- Minimal meaningful change to test
    - Making a mistake in the last 5 mins
        - Easy to find/fix
    - Making a mistake in the last hour
        - Hard to find/fix
        - You likely break things while fixing

# Stub

```
function Accordion({ entries }) {
  console.log( { entries }); // confirm what we get
  return (
    <div className="accordion">
      Accordion
    </div>
  );
}
export default Accordion;
```

# Plan the HTML

- What is the "shape" of our data?
    - object with title/body pairs
- What shape are we rendering?
    - each entry is a div(?) container
        - always question a div
        - but often a div
    - title will be button
        - for semantic meaning
    - body will be `<section>`
        - semantically valid
        - here vs entry container?

# Plan the HTML 2

What additional structure do we need?

- How are we showing open/close?
    - Icons, text would be additional element
        - Might require a wrapper if so
    - `::before/::after` pseudo-elements
        - No extra elements required
        - But cannot have extra a11y info
            - more later (ARIA)
- No extra validation info or feedback
    - For this pattern

# Plan the appearance

- Clues for CSS
- Reality-checks HTML plan
- title button full width
    - padding to make a nice block
- title will be on the left side of the button
- show a "+/-" on the right side of the button
    - based on open/close state
- body will show/hide content
    - show/hide via CSS vs include/not in HTML
        - judgment call, all the same "content"
    - no animation for now

# Planning state

Needs to track which item(s) are open

- allowing multiple open
- question I need to answer:
    - for any entry, is it open?
    - I will have the entry
        - no unique ids on this data
        - just title

# Planning State 2

Will use an object with title as key

- Hold boolean for "is open"
- Default to no open
- Can use empty object and add keys as opened

```
function Accordion({ entries }) {
  const [isEntryOpen, setIsEntryOpen] = useState({});

  function toggleEntry(title) {
    setIsEntryOpen({
      ...isEntryOpen,
      [title]: !isEntryOpen[title],
    });
  }
```

# Summary - Accordion

- A UI Pattern of multiple collapsable sections
    - differences in:
        - can many sections be open
        - which section(s) start open?
- We will implement as a React Component
    - pass object of entries
    - Component will manage its own state

# Summary - Process

- Implement in steps
- Verify before advancing
    - Stub
        - Confirm component will appear
        - Confirm passed data
    - HTML structure
        - Base on semantics
    - Appearance plan
        - define needs for HTML + CSS

# Summary - Implement

- CSS and JS
    - Build in cycles
        - Confirm each step
    - Manually force class states to confirm
- Implement state change handling last
    - You know the UI works already