

How to create visual changes

- When event happens for a visual change
 - change a class on an element
 - CSS will now match different
- Just like `:hover` or `:focus`
 - but with having/not-having class instead

Do not make life harder

If flipping between two states

- Don't create an extra state you don't use
- Have a default state and an alternate state

Example: themes:

- NOT: `.page.light` AND `.page.dark`
 - what about `.page`?
 - made life harder for no gain
- DO: `.page` and `.page.dark`
 - Fewer states to manage

Why Propagation?

Imagine a TODO list

- Click on event to remove
- type in input and click button to add task

With many handlers

- Each ``
 - click handler to remove
 - need to remove handler before removing
 - memory leak
- When adding ``
 - add handler to new ``
- If HTML is regenerated
 - need to remove all handlers before
 - all handlers need to be added after

With one handler

- `` listens for clicks
 - checks `e.target` to find what was clicked
 - does what is needed with that node
- Rerender of ul content HTML
 - impacts no handlers

Much less work

- fewer places to make mistakes!

Dynamic elements

Some changes can't be done by just changing classes

- You may need to add/remove elements

Using State

state is a collection of all the values that can change in an application

- Like a vending machine tracking inventory, vending status, money entered, and change given

Rendering from state

One Pattern:

- Have some state variable
- generate an `.innerHTML` string based on that

Can always (re)generate the HTML from state

Rendering example

```
<ul class="todo-list"></li>
```

```
let todos = [
  {
    task: "Nap",
    done: true,
  },
  {
    task: "Eat",
    done: false,
  },
];

function render() {
  const list = todos.map( (todo, index) => `
    <li><label>
      <span>${todo.task}</span>
      <input type="checkbox" class="todo"
        data-index="${index}" ${todo.done ? "checked" : ""}>
    </span></li>
  `).join(' ');
  document.querySelector('.todo-list').innerHTML = list;
}
```

Propagation example

```
document.querySelector('.todo-list').addEventListener('click', (e) => {  
  if(!e.target.classList.contains('todo')) {  
    return; // discard clicks not on a checkbox  
  }  
  const index = e.target.dataset.index;  
  todos[index].done = !todos[index].done;  
  render();  
});
```