MagnumJS Micro UI – Quick Start Tutorial

This tutorial walks you through building a simple parent-child component setup with local state, slots, and shared state. By the end, you'll understand the core concepts of MagnumJS Micro UI.

Step 1: Create Your First Component

Let's start with a simple Counter component that uses local state and an event.

```
const Counter = createComponent({
  state: { count: 0 },
  render() {
    return `<button>Clicked ${this.state.count} times</button>`;
  },
  on: {
    "click button": () => this.setState({ count: this.state.count + 1 })
  }
});

Counter.mount(document.body);
```

Step 2: Parent–Child with Slots

Now let's create a Parent component with a slot. Child content is declarative and auto-mounted.

```
const Parent = createComponent({
  render() {
    return `
      <div class="parent">
        <h1>Parent Component</h1>
        <slot></slot>
      </div>
    `;
  }
});

const Child = createComponent({
  render() {
    return `<p>I am a child component!</p>`;
  }
});

Parent.mount(document.body, {
  slots: {
    default: () => Child
  }
});
```

Step 3: Shared State Across Components

Finally, let’s use a shared store so multiple components stay in sync.

```
import { shared } from "@magnumjs/micro-ui/compose";

// Shared store
const counterStore = shared({ count: 0 });

// Button component increments shared state
const Incrementer = createComponent({
  render() {
    return `<button>Add One</button>`;
  },
  on: {
    "click button": () => counterStore.set({ count: counterStore.get().count + 1 })
  }
});

// Viewer component subscribes to shared state
const Viewer = createComponent({
  render() {
    return `<p>Count: ${this.state.count}</p>`;
  },
  onMount() {
    counterStore.subscribe((s) => this.setState(s));
  }
});

// Mount both together
Incrementer.mount(document.body);
Viewer.mount(document.body);
```

■ You’re Done!

You now know how to: • Build a component with state and events • Compose parent-child relationships with slots • Share state across components with reactive stores This covers the essentials of MagnumJS Micro UI — from here, you can scale to full apps with minimal code.