React vs Vue

A Fundamental Comparison

This talk is NOT...

- How to decide which framework to use
- A discussion about state management solutions for either framework/library i.e. Redux, MobX, Vuex

This talk IS about … Patterns!

- Managing component state
- Binding to form inputs
- Event Handling
- Communication between components
- Template syntax

React CodeSandbox

Vue CodeSandbox

Managing State in React

Initialize state with useState hooks. useState returns two values → a stateful value, and a function to update it.

During the initial render, the returned state (billAmount) is the same as the value passed as the first argument (0).

Some tips for updating state with hooks:

- If the new state is computed using the previous state, pass a function to setState

From TipForm.js...

```
1  const [billAmount, setBillAmount] = useState(0);
2  const [tipPercentage, setTipPecentage] = useState(0);
3  const [split, setSplit] = useState(1);
```

```
1 <input
2   id="billAmount"
3   type="number"
4   value={billAmount}
5   onChange={e => setBillAmount(e.currentTarget.value)}
6 />
```

Managing State in Vue

Initialize state with data object

```
data() {
  return {
    fields: {
      billAmount: 0,
      tipPercentage: 0,
      split: 1
    }
  };
},
```

From TipForm.vue...

Update state data by setting new values

```
this.fields.billAmount = 100
```

Or by using v-model on a user input

```
<input
  id="billAmount"
  v-model.number="fields.billAmount"
  type="number"
  @input="$emit('submit', fields)"
>
```

Binding to Form Inputs in React

Controlled components:

- Use update state function from useState hook
- Value is read out from state

```
const [billAmount, setBillAmount] = useState(0);
const [tipPercentage, setTipPecentage] = useState(0);
const [split, setSplit] = useState(1);
```

```
<input
  id="billAmount"
  type="number"
  value={billAmount}
  onChange={e => setBillAmount(e.currentTarget.value)}
/>
```

Binding to Form Inputs in Vue

You can use the v-model directive to create two-way data bindings on form input, textarea, and select elements.

- It automatically picks the correct way to update the element based on the input type.
- Modifiers allow you to update user input automatically (i.e. .number type casts input to a number, .trim trims whitespace)

```
<input
  id="billAmount"
  v-model.number="fields.billAmount"
  type="number"
  @input="$emit('submit', fields)"
>
```

`v-model="fields.billAmount"` is just shorthand for:

```
:value="fields.billAmount"
@input="fields.billAmount = $event.target.value"
```

Event Handling in React

Event handling in React is similar to native event handling.

You can pass in a function name or pass in an anonymous function that will run onclick in this case

From IncrementField.js...

```
function incrementDown() {
  if (value - increment >= min) {
    onChange(value => value - increment);
  }
}
```

```
<button type="button" onClick={incrementDown}>
  -
</button>
```

Event Handling in Vue

Vue uses the v-on (@ for short) directive to listen to DOM events and run js. You can pass js to run or a method name as a string

You can also add modifiers to these event listeners like .prevent and .stop to do things like stopPropagation() or preventDefault()

From IncrementField.vue...

```
incrementDown() {
  if (this.value - this.increment >= this.min) {
    this.$emit("input", this.value - this.increment);
  }
}
```

```
<button type="button" @click="incrementDown">
  -
</button>
```

Component Communication in React

Child components in React will have access to parent functions via this.props.

```
export default function TipCalculator() {
  const [splitBillTotal, setSplitBillTotal] = useState(0);

  function calculateBillTotal(billAmount, tipPercentage, split) {
    const billNum = Number(billAmount);
    const tipNum = 1 + Number(tipPercentage) / 100;
    const billTotal = (billNum * tipNum) / Number(split);
    setSplitBillTotal(billTotal);
  }

  return (
    <div className="calculator">
      <h1>Tip Calculator</h1>
      <TipForm calculateBillTotal={calculateBillTotal} />
      <div className="total">
        <span>Split Bill Total:</span>
        <span className="split-total">${splitBillTotal.toFixed(2)}</span>
      </div>
    </div>
  );
}
```

```
export default function TipForm({ calculateBillTotal }) {
  const [billAmount, setBillAmount] = useState(0);
  const [tipPercentage, setTipPecentage] = useState(0);
  const [split, setSplit] = useState(1);

  useEffect(() => {
    calculateBillTotal(billAmount, tipPercentage, split);
  }, [billAmount, tipPercentage, split]);

  return (jsx stuff ... );
}
```

Component Communication in Vue

With Vue, you have to emit events from the child that will usually be collected inside the parent component.

From TipForm.vue...

```
<input
  id="billAmount"
  v-model.number="fields.billAmount"
  type="number"
  @input="$emit('submit', fields)"
>
```

```
<template>
  <div class="calculator">
    <h1>Tip Calculator</h1>
    <tip-form @submit="calculateBillTotal"/>
    <div class="total">
      <span>Split Bill Total:</span>
      <span class="split-total">${{ splitBillTotal.toFixed(2) }}</span>
    </div>
  </div>
</template>

<script>
import TipForm from "./TipForm";
export default {
  name: "TipCalculator",
  data() {
    return {
      splitBillTotal: 0
    };
  },
  components: {
    TipForm
  },
  methods: {
    calculateBillTotal({ billAmount, tipPercentage, split }) {
      this.splitBillTotal = (billAmount * (1 + tipPercentage / 100)) / split;
    }
  }
};
</script>
```

Template syntax

React

```jsx
export default {
  // ...
  render() {
    return (
      <ul>
        {this.posts.map(post => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    );
  }
};
```

Vue

```vue
<template>
  <ul>
    <li v-for="post in posts" :key="post.id">{{ post.title }}</li>
  </ul>
</template>
```

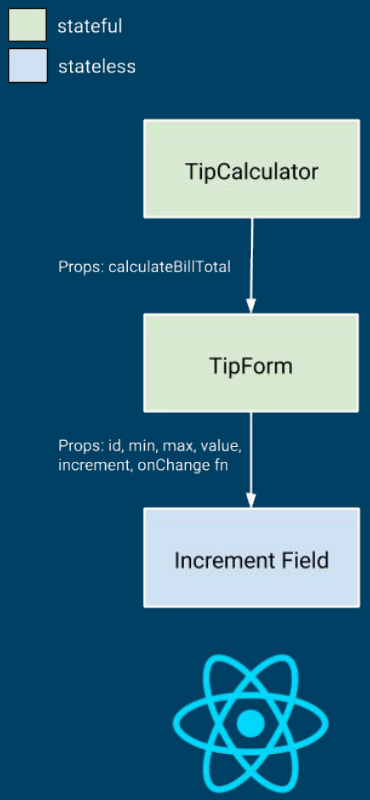
Takeaways

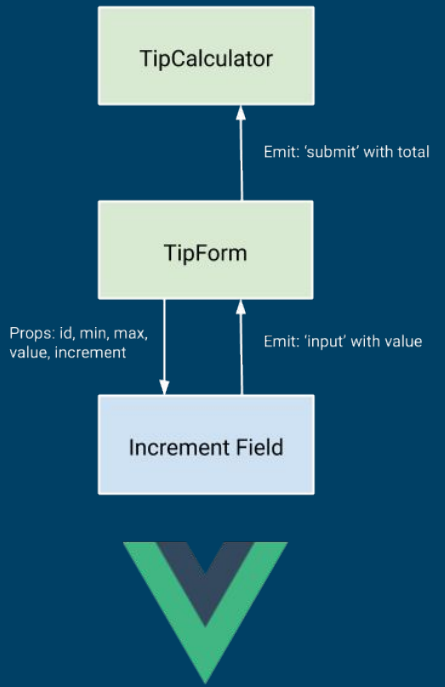
React

One way data binding: Components receive information through arguments (props) and pass information via their return values (the return value of the render function). This is called unidirectional data flow.

Vue

Two way data binding: The v-model directive updates the template whenever the model changes and updates data model whenever the template changes.





Takeaways cont.

Differences

React

- React uses plain JavaScript (function calls, usually with JSX) to construct views. This gives you the full freedom JavaScript expressions and logic.
- Encourages immutability

Vue

- Vue opts for its own templating language using directives in the HTML template to decide how the view should look, making it feel more traditional and accessible to beginners.
- Encourages mutation

Similarities

- Both utilize a virtual DOM
- Both provide reactive and composable view components
- Both support render functions and using JSX although JSX in Vue might detract from one of the biggest selling points of Vue (single file components with separate HTML, CSS, JS)