Get hyper-excited for web standards

Jiayi Hu

1. Web Components
2. Custom Elements
3. hyperHTML & HyperHTMLElement

Foreword

Not trying to convince you to abandon React/Angular/Vue 🙃

Web Components

```
<video controls src="video.mp4"></video>

<select>
  <option>Option 1</option>
  <option>Option 2</option>
  <option>Option 3</option>
</select>

<input type="date">
<input type="range">
```

```
<div class="dropdown">
  <button class="btn btn-secondary dropdown-toggle">
    Dropdown button
  </button>
  <div class="dropdown-menu">
    <a class="dropdown-item" href="#">Action</a>
    <a class="dropdown-item" href="#">Another action</a>
    <a class="dropdown-item" href="#">Something else here</a>
  </div>
</div>
```

ANGULARJS

by Google

```
<tabs>
  <pane title="Hello">Hello</pane>
  <pane title="World">World</pane>
</tabs>
```

What if want

— Truly reusable components

— Interoperability

Web Components

A suit of web standards to define reusable custom elements

1. Custom Elements
2. Shadow DOM

Custom Elements

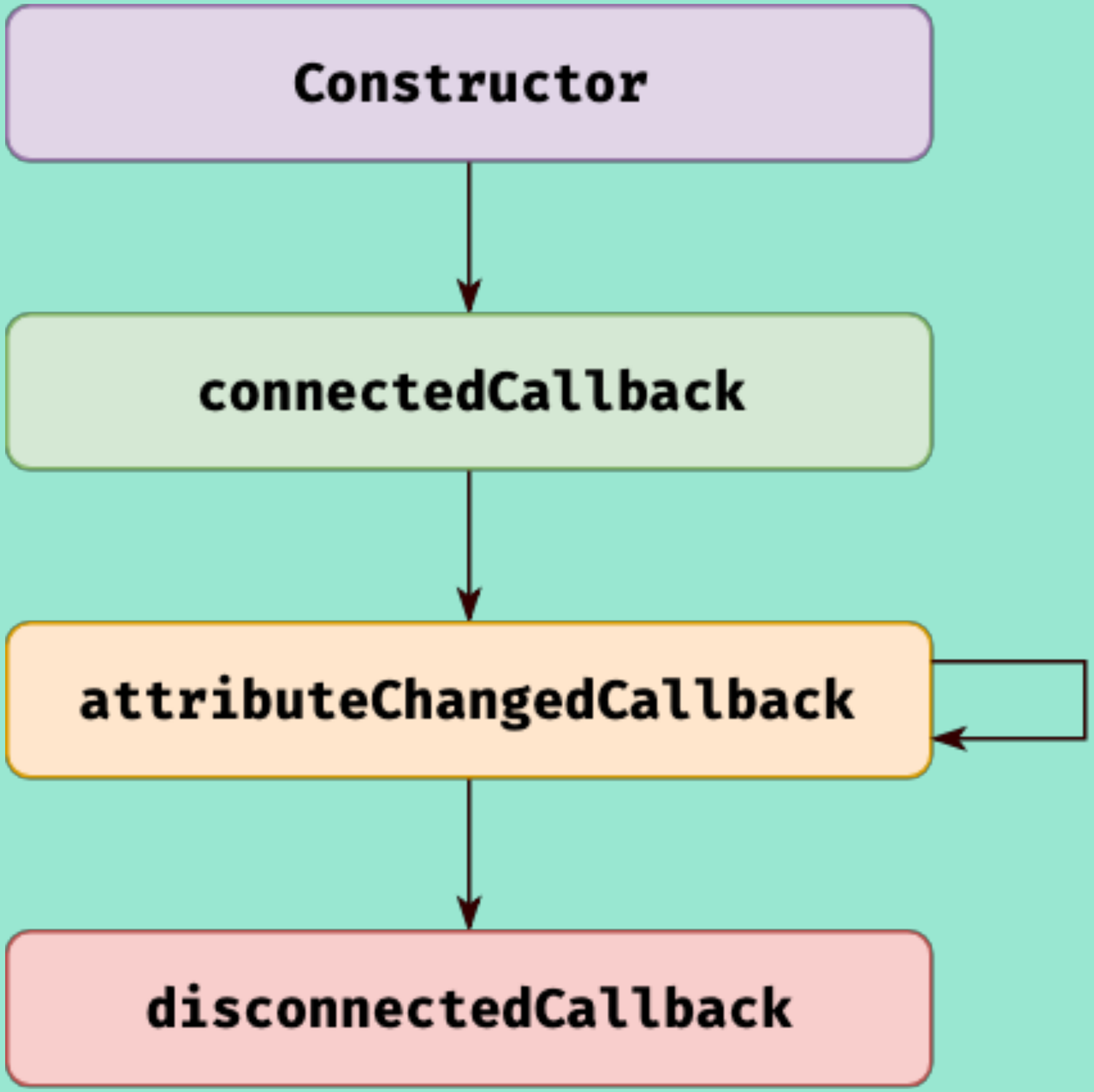
Create new HTML tags

```
<c-clock></c-clock>
```

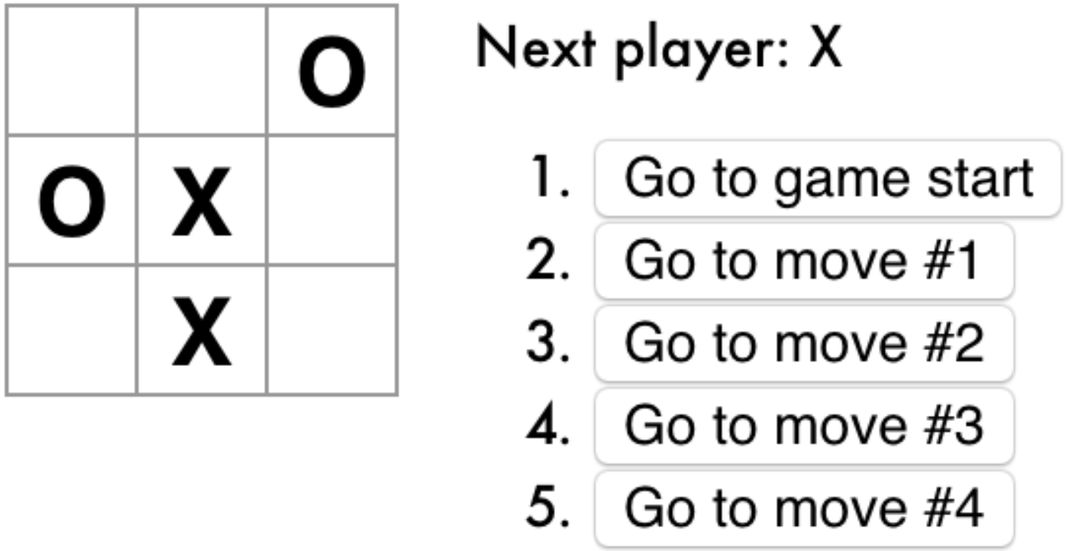
Codepen

```
const template = document.createElement('template')
template.innerHTML = `
  <div>
      <h1>Hello, world!</h1>
      <h2>It is <span class="time"></span>.</h2>
  </div>
`

class Clock extends HTMLElement {
  constructor() {
    super();

    this.appendChild(template.content.cloneNode(true))
    this.timeEl = this.querySelector('.time')
  }

  connectedCallback() {
    this.token = window.setInterval(() => {
      this.timeEl.textContent = new Date().toLocaleTimeString();
    }, 1000)
  }

  disconnectedCallback() {
    if (this.token) window.clearInterval(this.token);
  }
}

customElements.define('c-clock', Clock)
```

Custom Element Lifecycle



Tic-tac-toe XO



in React

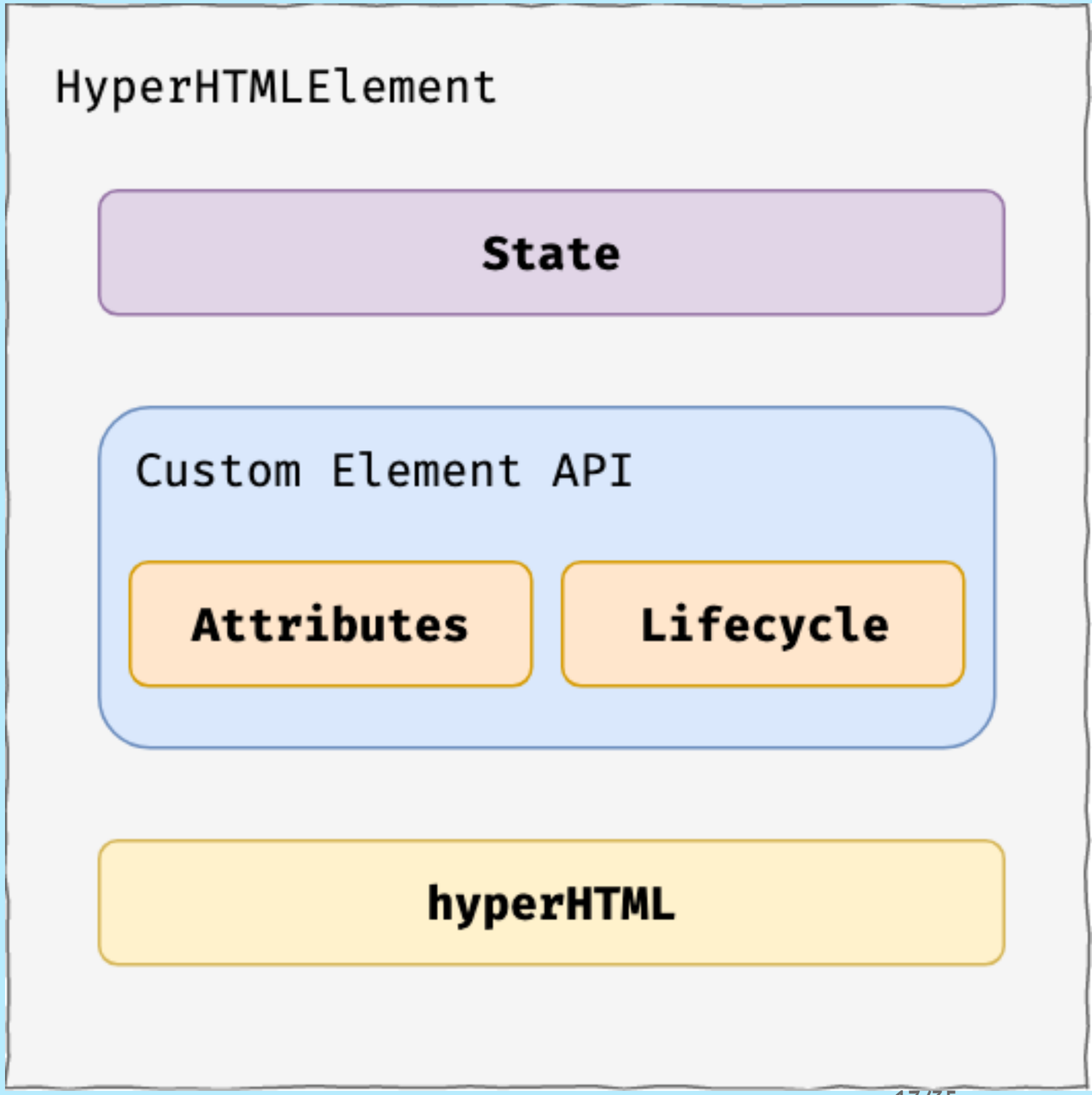
```
<c-square value="X"></c-square>
```

```
const template = document.createElement('template')
template.innerHTML = `<button class="square"></button>`

class Square extends HTMLElement {
  static get observedAttributes() {
    return ['value']
  }

  get value() { return this.getAttribute('value') }
  set value(val) { this.setAttribute('value', val) }

  constructor() {
    super();

    this.appendChild(template.content.cloneNode(true))
    this.btnEl = this.querySelector('.square')
  }

  attributeChangedCallback(attr, old, curr) {
    if (attr === 'value') this.btnEl.textContent = curr;
  }
}

customElements.define('c-square', Square)
```

Enter hyperHTML & HyperHTMLElement

```
class Square extends HyperHTMLElement {
  static get observedAttributes() {
    return ['value']
  }

  render() {
    return this.html`
      <button class="square">
        ${this.value}
      </button>
    `
  }
}

Square.define('c-square')
```

HyperHTML Element



hyperHTML (4kB)

```
const render = hyperHTML.bind(document.body);

function tick() {
  render`
    <div>
      <h1>Hello, world!</h1>
      <h2>It is ${new Date().toLocaleTimeString()}.</h2>
    </div>
  `;
}

setInterval(tick, 1000);
```

Codepen

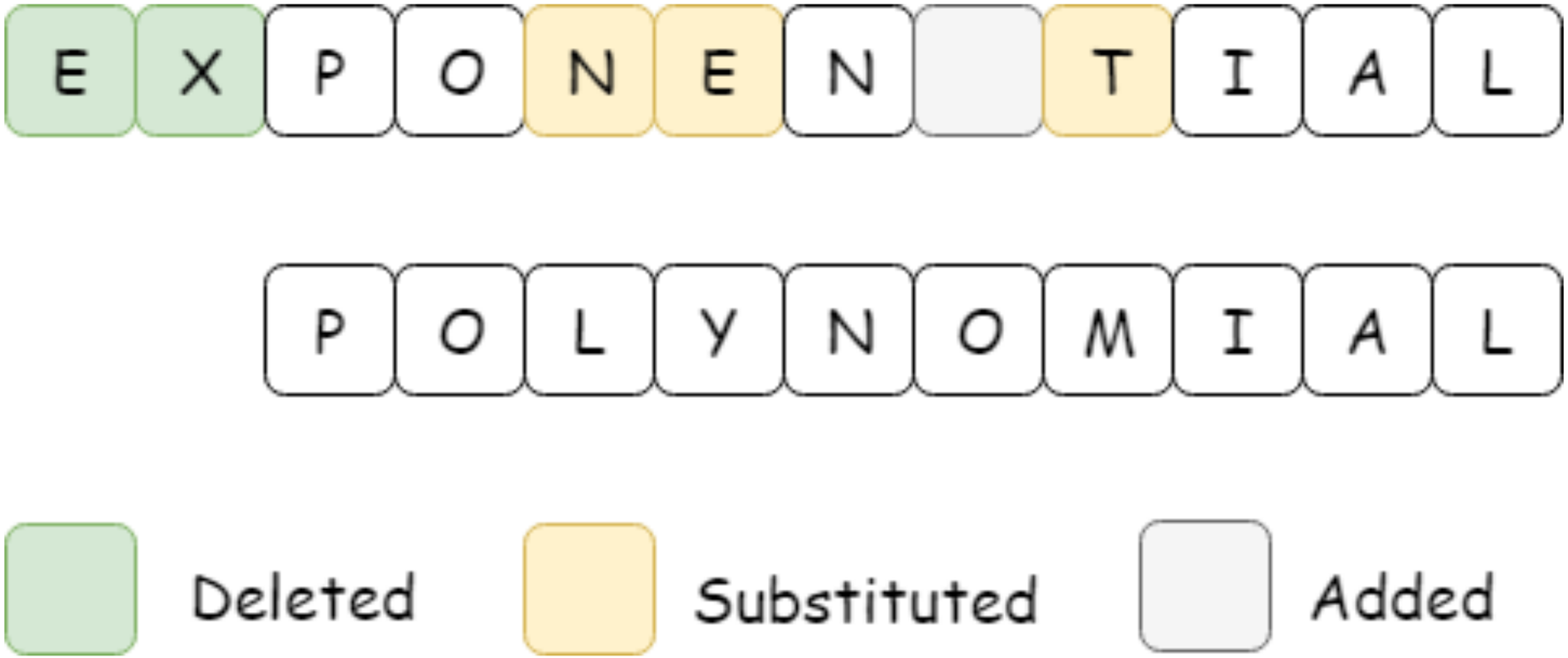
Tagged template literal

```
function html(chunks, ...interpolations) {
  console.log(chunks);              // ['1 ', ' 3']
  console.log(interpolations);  // [2] or [4]
}

html`1 ${2} 3`;
html`1 ${4} 3`;
```

Efficient DOM

```
const bodyRender = hyperHTML.bind(document.body);
const names = [
  { name: 'First item' },
  { name: 'Second item' },
  { name: 'Third item' }
]

hyperHTML.bind(document.body)`
  <h1>${document.title}</h1>
  <ul>
    ${names.map(item => `<li>${item.name}</li>`)}
  </ul>
`;
```

Levenshtein algorithm



domdiff based on petit-dom

Not primitive types

```
customElements.define('h-welcome', class HyperWelcome extends HTMLElement {
    constructor() {
      super();
      this.html = hyperHTML.bind(this);
    }

    get user() { return this._user; }
    set user(value) { this._user = value; this.render(); }

    render() { return this.html`<h1>Hello, ${this._user.name}</h1>`; }
  }
);

hyperHTML.bind(document.getElementById('root'))`
  <h-welcome user=${{ name: 'Sara' }} />
  <h-welcome user=${{ name: 'Cahal' }} />
`;
```

HyperHTMLElement

```
class Game extends HyperHTMLElement {
  get defaultState() {
    return { history: [], stepNumber: 0 };
  }

  handleClick(event) { this.setState({ ... }); }

  render() {
    const current = this.state.history[this.state.stepNumber];

    return this.html`
      <div class="game-board">
        <c-board
          squares=${current.squares}
          onsquareclick=${e => this.handleClick(e.detail)}
        />
      </div>
    `;
  }
}
```

Server side rendering with viperHTML

hyperHTML vs lit-html vs omi

vs lit-html 1.0

Tencent/omi

Custom Elements in React

— custom-elements-everywhere

— Web Components in React

Use cases for Custom Elements and hyperHTML

- **cross-framework** UI components and libraries
  - Primer - Github
  - Vaadin
- Long-lasting web projects
- Lightweight **framework-less compiler-less** development

> Standards are the best way, if not the only one, to move the Web forward.
>
> — Andrea Giammarchi

Last notes - Redux

```
export class Homepage extends ConnectedHyperElement {
  connectedCallback() {
    super.connectedCallback();

    getFeeds().then(feeds => this.dispatch(addFeeds(feeds)));
    this.render();
  }

  stateChanged(state) {
    this.feeds = state.feeds;
    this.render();
  }

  render() {
    console.log(this.feeds)
  }
}
```

Last notes - Hooks

```
import stardust, {html, useState} from 'neverland';

const Counter = stardust(() => {
  const [count, setCount] = useState(0);

  return html`
    <p>You clicked ${count} times</p>
    <button onclick=${() => setCount(count + 1)}>
      Click me
    </button>
  `;
});
```

Last notes - Testing

```
import { Button } from './Button';

describe('<mr-button>', () => {
  it('renders correct className', () => {
    const element = new Button();
    element.setAttribute('kind', 'primary');
    element.render();
    const button = element.querySelector('button');

    expect(button.classList.contains('button--primary')).toBe(true);
  });
});
```

One last slide…

Jiayi Hu

[dʒʌɪ]

Front-end developer

— Twitter: @jiayi_ghu

— GitHub: github.com/jiayihu/talks

— italiajs.slack.com

Get hyper-excited