# Reactive Typesafe WebComponents

build with @skatejs

## typeof whoAmI

I skate

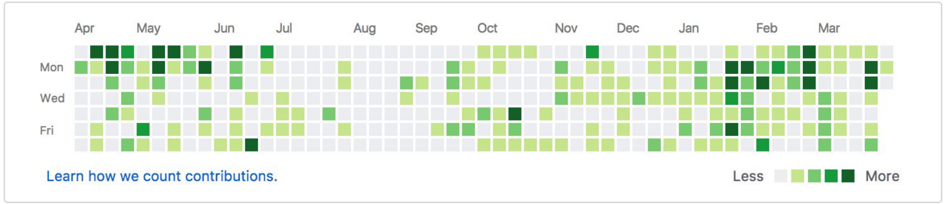
I wake

I surf

I snow

I DO & ❤️

Open Source



Martin Hochel
@martin_hotell
github.com/Hotell

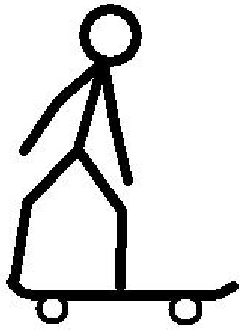
Technical Lead
EmbedIT
Prague, CZ

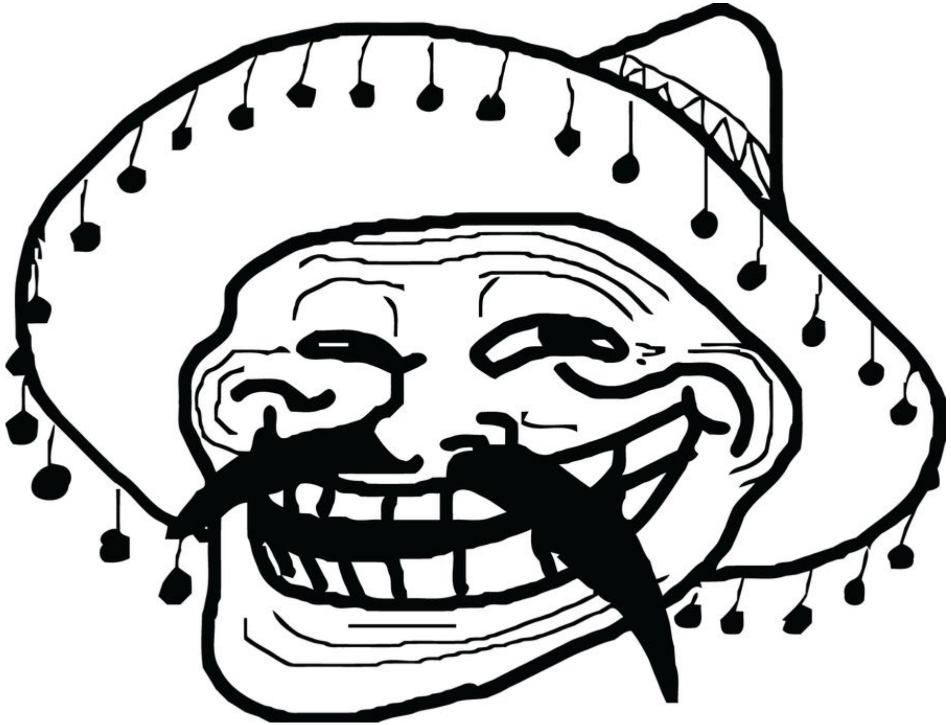
JS Community Leader
ngParty

Author of ng-metadata
Core member of @skateJS

Today’s talk
Will be all about
Skateboarding

## This is how you do an Ollie





## Why Web Components

THIS IS MADNESS!

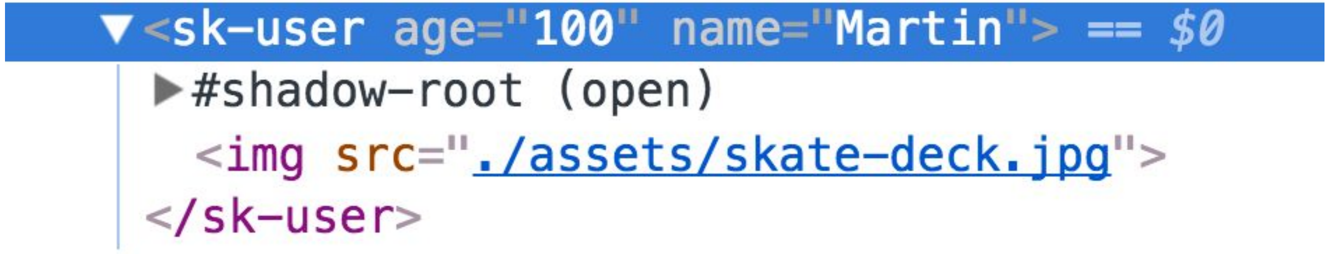
❤️

Write once
Use everywhere

## What is a Web Component

```html
<sk-user name="Martin" age="100">
  <img src="./assets/skate-deck.jpg">
</sk-user>
```

- Name: Martin
- Age: 100



```
▼<sk-user age="100" name="Martin"> == $0
   ▶#shadow-root (open)
    <img src="./assets/skate-deck.jpg">
  </sk-user>
```

MAGIC

# Web Component

```
<sk-user name="Martin" age="100">
  <img src="./assets/skate-deck.jpg">
</sk-user>
```

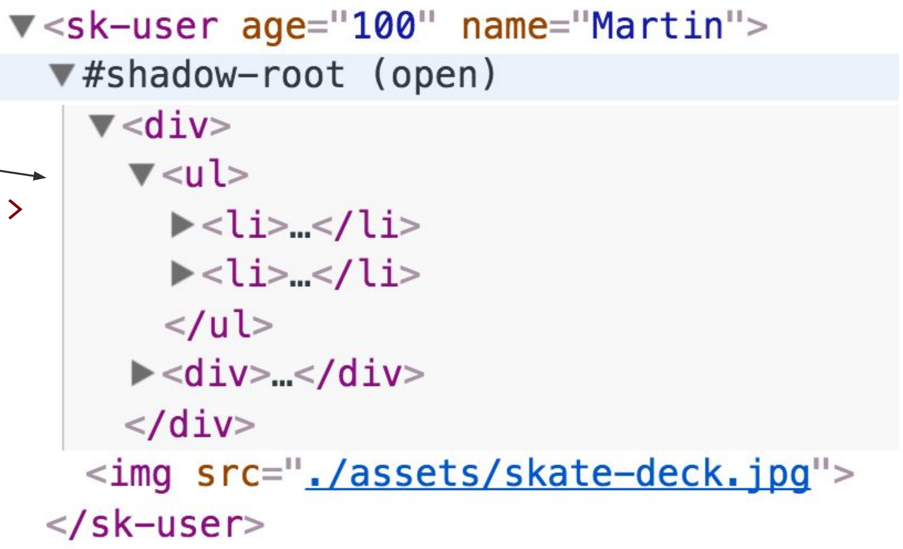
Custom Elements

Shadow DOM

HTML <template>

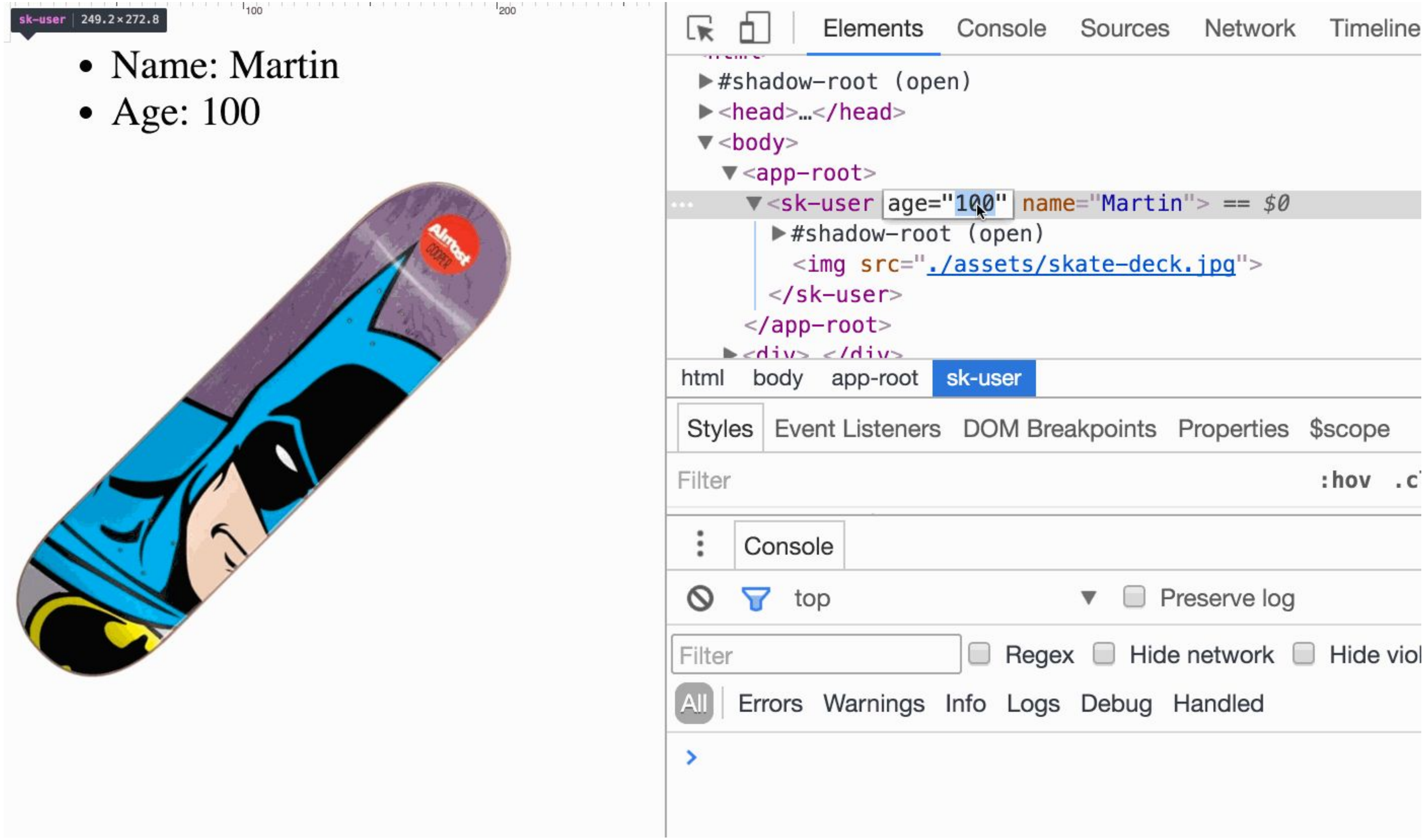
Inputs `<sk-user name="Martin" age="100">`

Outputs ( Custom events )



```
▼<sk-user age="100" name="Martin">
  ▼#shadow-root (open)
    ▼<div>
      ▼<ul>
        ▶<li>…</li>
        ▶<li>…</li>
       </ul>
      ▶<div>…</div>
     </div>
   <img src="./assets/skate-deck.jpg">
</sk-user>
```

## It’s alive



sk-user 249.2×272.8

- Name: Martin
- Age: 100

```
▶#shadow-root (open)
▶<head>…</head>
▼<body>
  ▼<app-root>
    ▼<sk-user age="100" name="Martin"> == $0
      ▶#shadow-root (open)
       <img src="./assets/skate-deck.jpg">
     </sk-user>
   </app-root>
```

## Implementation

```
<script>
  class User {
    static get observedAttributes() {
      return ['name', 'age'];
    }

    _name = ''
    get name() {
      return this.getAttribute('name') || this._name;
    }
    set name(val) {
      if (val) {
        this._name = val;
      }
    }

    _age = 0;
    get age() {
      return Number(this.getAttribute('age')) || this._age;
    }
    set age(val) {
      if (val) {
        this._age = val;
      }
    }
    constructor(){
      super();
      this.attachShadowRoot({open:true});
      const templateRef = document.getElementById('sk-user');
      const instance = templateRef.content.cloneNode(true);
      shadowRoot.appendChild(instance);
    }
  }

  customElements.define('sk-user',User);
</script>
```

```
<template id="sk-user">
 <div>
  <ul>
    <li>Name: <b id="name"></b></li>
    <li>Age: <b id="age"></b></li>
  </ul>
  <div>
    <slot></slot>
  </div>
</div>
</template>
```

## Well defined API
## WebComponents v1

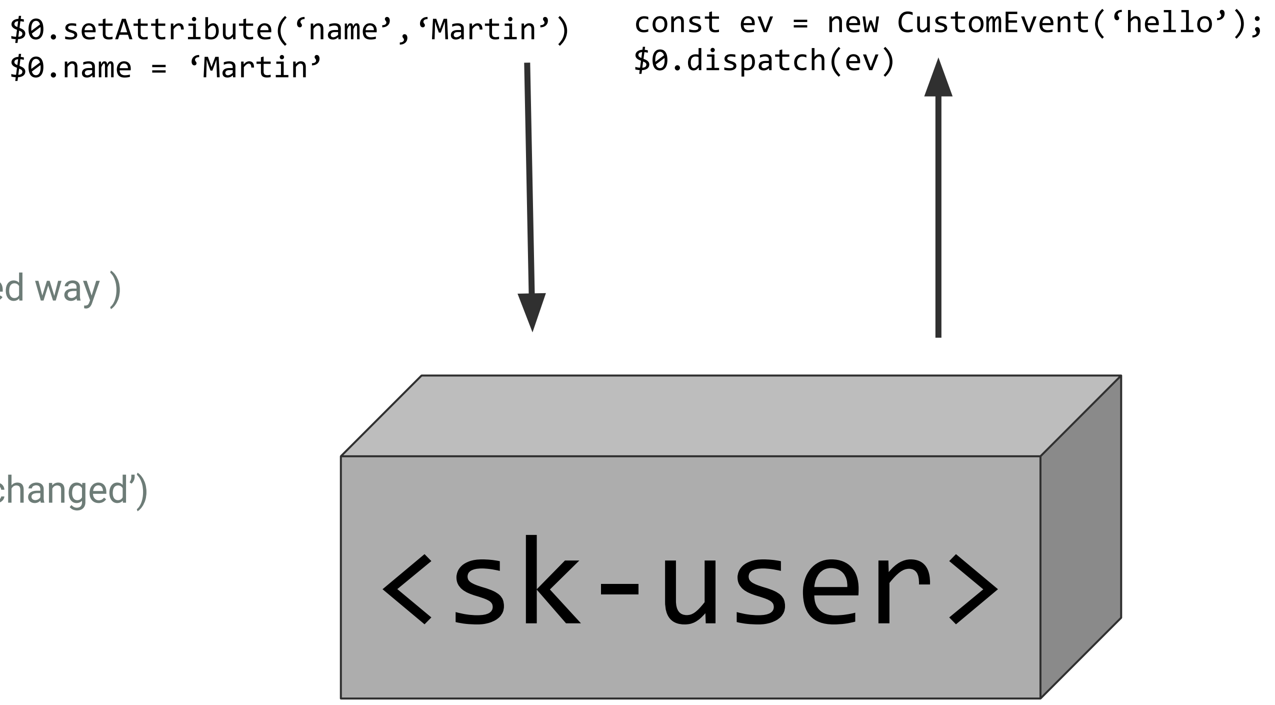
## API - Data Binding / Flow

Inputs

- DOM -> Attribute reflection
- Properties ( faster, preferred way )

Outputs

- CustomEvent(‘something-changed’)



```
$0.setAttribute(‘name’,‘Martin’)
$0.name = ‘Martin’
```

```
const ev = new CustomEvent(‘hello’);
$0.dispatch(ev)
```

## API - Reactions and LC Hooks

constructor

connectedCallback

disconnectedCallback

---

attributeChangedCallback

getters/setters

## So what’s the problem with vanilla WebComponents ?

Magic strings - HTML
Data Binding ?
Hard to maintain
Huge boilerplate
Unpredictable re-render
Mutation of your state

## Skate JS

## WAT is Skate?

Reactive WebComponent Micro-library

**4kb**, yes sir you read that damn right!

Functional approach

Virtual DOM for rendering

It’s just javascript

No magic

Refactorable code

## But most importantly

It's just

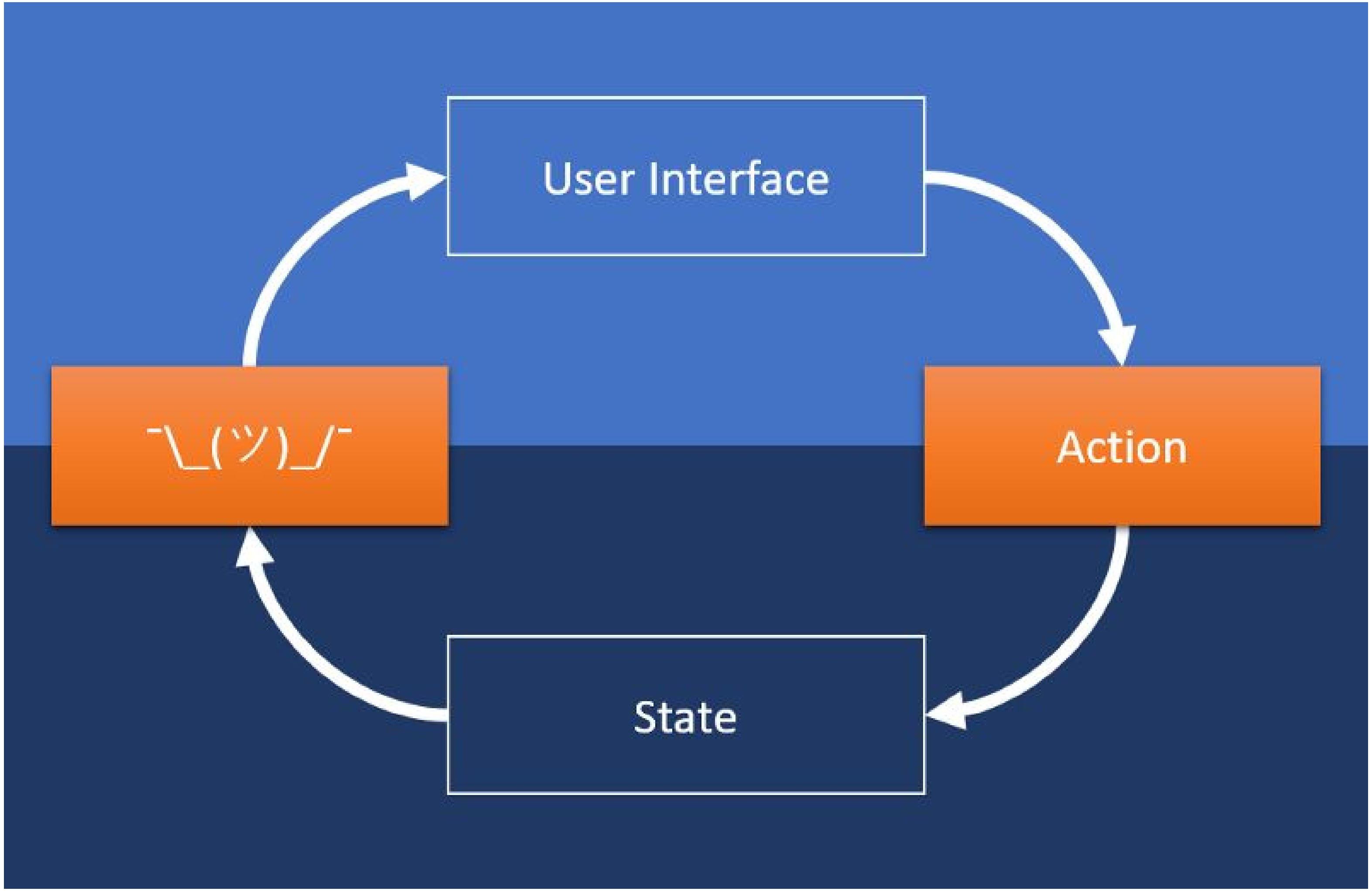
vanilla web components

with

Functional rendering pipeline

On steroids

## Reactivity


User Interface
¯_(ツ)_/¯
Action
State

~~State~~ is the only source of truth

Props

Props changes batched

--->

VDOM diff

--->

Re Render

--->

Perf!

## Type safety

It’s just Javascript + JSX

So

Yes

Please Welcome

## Typescript

SHOW ME WHAT YOU GOT

## Web Components with Skate

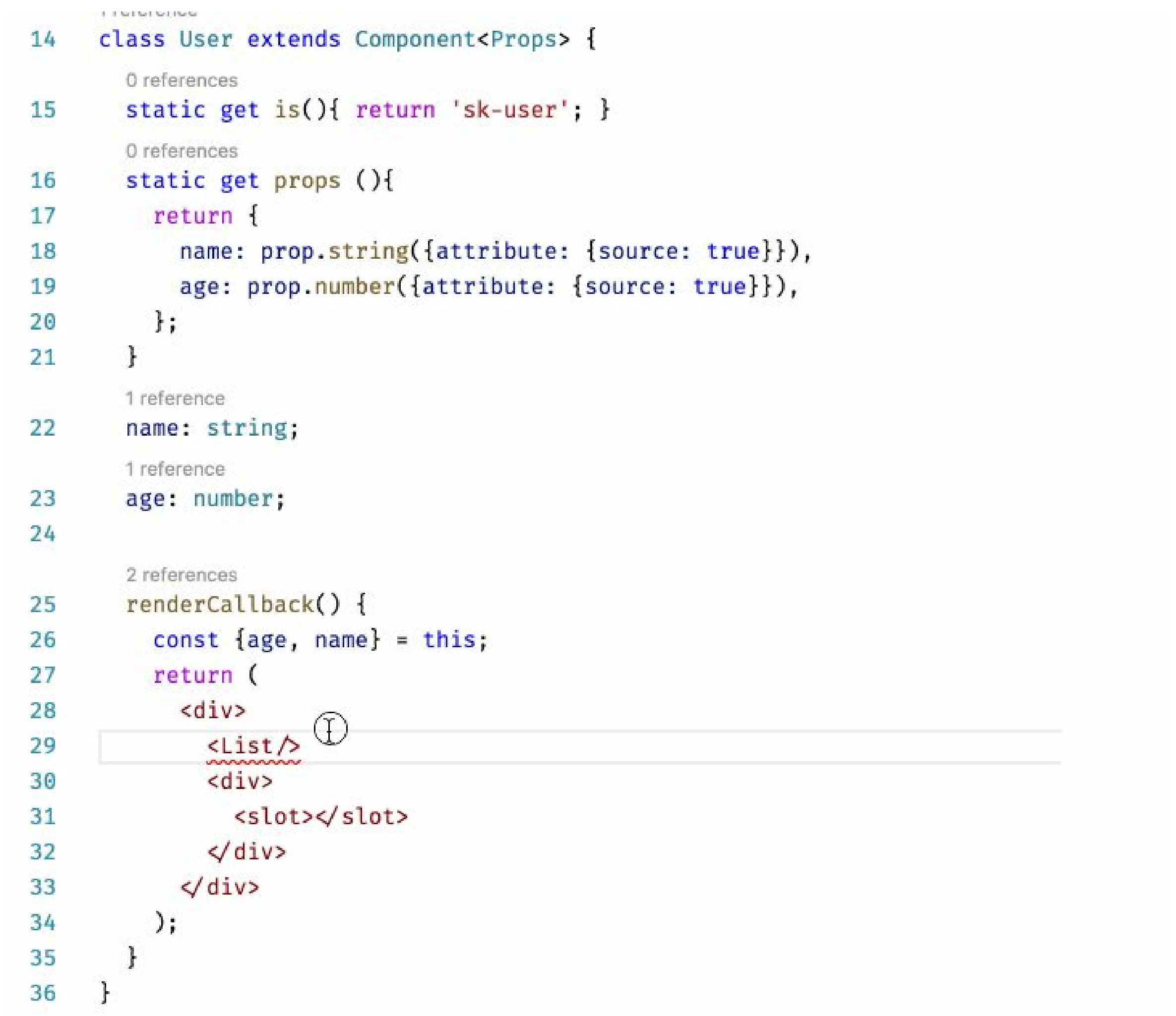
```
import { Component, h, prop } from 'skatejs';

class User extends Component<Props> {
  static get is(){ return 'sk-user'; }
  static get props (){
    return {
      name: prop.string(),
      age: prop.number(),
    };
  }
  name: string;
  age: number;

  renderCallback() {
    const {age, name} = this;
    return (
      <div>
        <ul>
          <li>Name: {name}</li>
          <li>Age: {age}</li>
        </ul>
        <div>
          <slot/>
        </div>
      </div>
    );
  }
}
```

```
import { define } from 'skatejs';
import { User } from './User';

define(User);
```

## Functional Components

```
const List = ({name, age}: Props) => (

    <ul>

      <li>Name: {name}</li>

      <li>Age: {age}</li>

    </ul>

);
```

```
renderCallback() {
  const {age, name} = this;
  return (
    <div>
      <ul>
        <li>Name: {name}</li>
        <li>Age: {age}</li>
      </ul>
      <div>
        <slot/>
      </div>
    </div>
  );
}
```



```
class User extends Component<Props> {
  0 references
  static get is(){ return 'sk-user'; }
  0 references
  static get props (){
    return {
      name: prop.string({attribute: {source: true}}),
      age: prop.number({attribute: {source: true}}),
    };
  }
  1 reference
  name: string;
  1 reference
  age: number;

  2 references
  renderCallback() {
    const {age, name} = this;
    return (
      <div>
        <List/>
        <div>
          <slot></slot>
        </div>
      </div>
    );
  }
}
```

## Pluggable API

→ Mixins/Decorators

→ functional composition

```
const Component = ( Base = HTMLElement ) => {
  return class extends withUnique(
    withRender(
      withProps(
        Base
      )
    )
  ) { }
}
```

## CSS and encapsulation

## CSS

$$\begin{tikzcd}
\text{Native Shadow Dom} \arrow[dr] & & \text{ShadyCSS} \arrow[dl] \\
& \text{CSS variables and mixins for theming} &
\end{tikzcd}$$

**CSS Modules → Webpack**

**CSS in JS → JSS**

## Summary

**Web Components** ➕ **SkateJS**

- Reusability
- Cross framework compatibility
- CSS Encapsulation via ShadowDOM
- Life Cycle Hooks
- Content Projection

- Functional fast renderer
- Simple API and data binding
- Typed JS from the box
- Reuse of modern JS tooling
- Pluggability

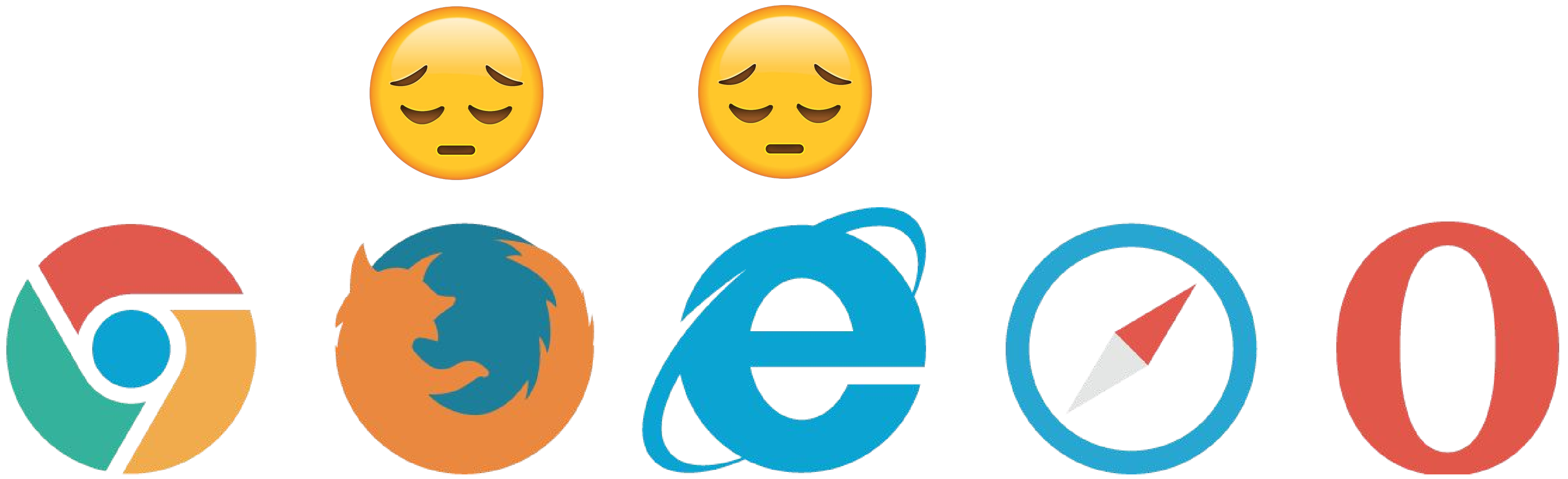
## Skate --> next

- 5.0 soon
- Preact as default renderer
- Mixins for everything

github.com/skatejs/skatejs
@skate_js
@treshugart

## What’s the catch?

Not every browser **supports** web components natively



Polyfills

huge/slow?/buggy

ShadyDOM

## Server side rendering

## Angular + Skate

## Demo

Thank you !