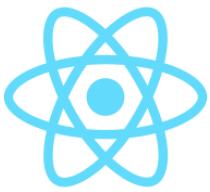
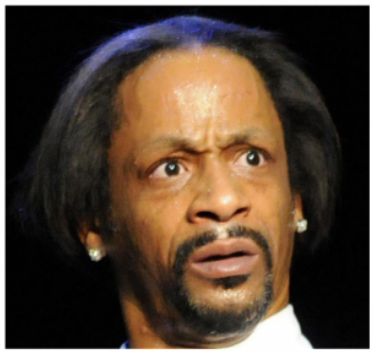


REACT PRESENTATION

Kash Karimi
Senior Platform Engineer



WAIT WHAT?

MY INTEREST & EXPERIENCE

- Started using React in 2014
- Few apps in production
- Teach React at Code Your Future
- Share passion

Consider alternatives

- SE Tech Constitution

TALK OVERVIEW

- React Concepts & Patterns
- Why React is Awesome
- Why & How-to use it at Secret Escapes
- QAs

REACT: CONCEPTS

LIBRARY NOT FRAMEWORK

DECLARATIVE STYLE

STATEFUL COMPONENTS

```
export default class Stars extends Component {
    async componentDidMount() {
        let stars = await githubStars(this.props.repo);
        this.setState({ stars });
    }
    render({ repo }, { stars=0 }) {
        let url = `//github.com/${repo}`;
        return (
            <a href={url} class="stars">
                * {stars} Stars
            </a>
        );
    }
}
```

REMEMBER THIS?

```
function init() {
    canvas = document.getElementById("gacanvas");
    if ( ! canvas || ! canvas.getContext ) {
        // This browser apparently does not support canvasses since the canvas
        // element has no getContext function.  Give up!
        document.getElementById("message").innerHTML = "Sorry, your browser doesn't support the canvas element!<br>"
                + "This page should work with most modern web browsers";
        return;
    }
    graphics = canvas.getContext("2d");
    graphics.strokeStyle = "#FF0000";
    statslog = document.getElementById("statslog");
    stepButton = document.getElementById("stepButton");
    runButton = document.getElementById("runButton");
    runToSeasonButton = document.getElementById("runToSeasonButton");
    pauseButton = document.getElementById("pauseButton");
    restartButton = document.getElementById("restartButton");
    stepButton.onclick = step;
    restartButton.onclick = startFromScratch;
    runButton.onclick = run;
    runToSeasonButton.onclick = runToEndOfYear;
    pauseButton.onclick = pause;
    speedSelect = document.getElementById("speedSelect");
    speedSelect.value = "1";
    delayTime = 10;
    eaterBirthSelect = document.getElementById("eaterBirthSelect");
    eaterBirthSelect.value = "2";
    populationSelect = document.getElementById("populationSelect");
    populationSelect.value = "50";
    plantCountSelect = document.getElementById("plantCountSelect");
    plantCountSelect.value = "250";
    plantBirthSelect = document.getElementById("plantBirthSelect");
    plantBirthSelect.value = "1";
    plantReplacementSelect = document.getElementById("plantReplacementSelect");
    plantReplacementSelect.value = "0";
    mutationProbabilitySelect = document.getElementById("mutationProbabilitySelect");
    mutationProbabilitySelect.value = "0.001";
    crossoverProbabilitySelect = document.getElementById("crossoverProbabilitySelect");
    crossoverProbabilitySelect.value = "0.8";
    fitnessBiasSelect = document.getElementById("fitnessBiasSelect");
    fitnessBiasSelect.value = "1";
    speedSelect.onchange = changeSpeed;
    eaterBirthSelect.onchange = function() { changeSetting("eaterBirth", parseInt(eaterBirthSelect.value)) };
    populationSelect.onchange = function() { changeSetting("populationSize", parseInt(populationSelect.value)) };
    plantCountSelect.onchange = function() { changeSetting("plantCount", parseInt(plantCountSelect.value)) };
    plantBirthSelect.onchange = function() { changeSetting("plantBirth", parseInt(plantBirthSelect.value)) };
    plantReplacementSelect.onchange = function() { changeSetting("plantRebirth", parseInt(plantReplacementSelect.value)) };
    mutationProbabilitySelect.onchange = function() { changeSetting("mutationProbability", parseFloat(mutationProbabilitySelec
    crossoverProbabilitySelect.onchange = function() { changeSetting("crossoverProbability", parseFloat(crossoverProbabilitySe
    startFromScratch();
};
```

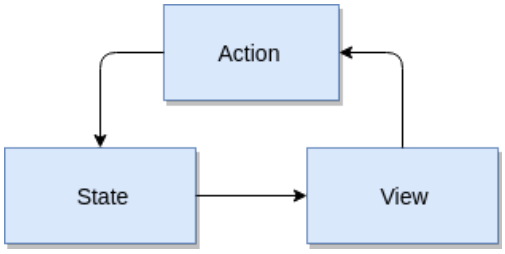
REUSABLE COMPONENTS

EXERCISE

How many components do you see?

VIRTUAL DOM

UNIDIRECTIONAL FLOW

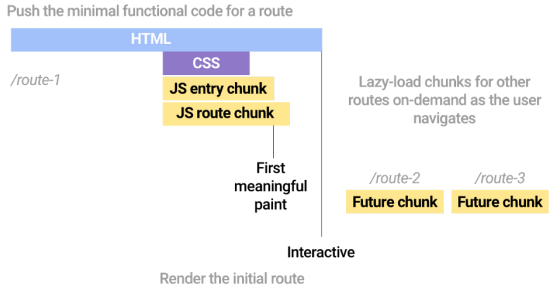


COMPETITION

HOW REACT BECAME THE QUEEN OF UI

- DOM and two-way binding
- Components are the future
- Developer experience
- Modern tools: hot reloading, code splitting (bundles), ServiceWorkers
- Progressive Web Apps (..or React Native apps)
- Flexible, rendered on server

ROUTE-BASED CHUNKING



- Faster initial load, and less heavy on browsers
- Lazy load future paths
- Webpack Tree-shaking ensures no unused code gets bundled

FUTURE

FUTURE

- REST shortcomings (too many requests, e.g. n+1 queries)

```
// Fetch the list of story IDs but not their details:
rest.get('/stories').then(stories =>
  // This resolves to a list of items with linked resources:
  // `[ { href: "http://.../story/1" }, ... ]`
  Promise.all(stories.map(story =>
    rest.get(story.href) // Follow the links
  ))
).then(stories => {
  // This resolves to a list of story items:
  // `[ { id: "...", text: "..." } ]`
  console.log(stories);
});


graphql.get(`query { stories { id, text } }`).then(
  stories => {
    // A list of story items:
    // `[ { id: "...", text: "..." } ]`
    console.log(stories);
  }
);
```

FUTURE IS HERE: GRAPHQL

- GraphQL offers significantly more flexibility
- Define precisely the data you want—and only the data you want
- Replace multiple REST requests with a single call
- Github API (v4) is no longer REST but GraphQL
- Relay: React Components request what they need

WHY SHOULD WE USE REACT AT SE?

- Already have REST API - decoupling easier
- More state on client side (cache)
- Less calls to the back-end
- User perceives better performance
- Progressive Web-App (PWA) ready
- Developer experience
- Future proofing

HOW CAN WE USE REACT AT SE?

- Select a test page
- Mount on to the page
- Convert existing pages to components
- New features/pages as components

QUESTIONS?

THANK YOU