

# React to Svelte

Story of one migration or how I spent last weekend



@hadoocken



@v\_hadoocken

## Infobip SPB



# **Infobip Frontend**

===  
**React**

React is:

- a. Need to follow best practices of this year
- b. Count of packages is limitless
- c. Codebase is not consistent without control

How about Svelte?

**Svelte? Svelte!**

03





# Prediction Award

Awarded to up-and-coming technology that  
might take over... or not?

State of JS 2019 / awards

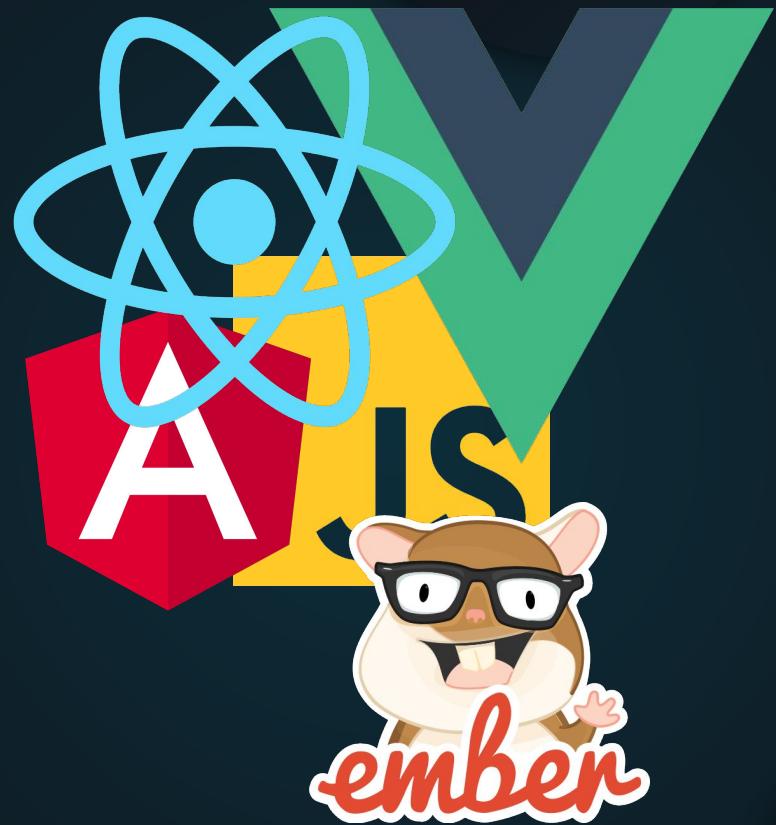
1		<b>Vue.js</b>	+31.4k ★
2		<b>React</b>	+22.9k ★
3		<b>Svelte</b>	+20.0k ★

In 2019, Svelte disrupted the category of the UI frameworks, taking the 3rd position behind Vue.js and React but before Angular.

Rising Stars 2019

**Back to 2007 2010s...**







# **Chapter I**

---

# **The Migration**

The slow or gradual movement of something from one place to another.

## **Migration in context of UI development**

- A. Continue develop react app and slowly rewrite old parts to svelte.
- B. Stop develop react app and start write svelte parts right into first one.



 Rich-Harris / react-svelte

Used by 2 Watch 6 Star 124 Fork 9

Code Issues 3 Pull requests 5 Actions Projects 0 Wiki Security Insights

Use Svelte components inside a React app

5 commits 3 branches 0 packages 3 releases 1 contributor View license

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Rich-Harris -> v1.0.2 Latest commit 10dcc68 on 21 Aug 2018

 demo	fix demo	17 months ago
 .gitignore	initial commit	17 months ago
 CHANGELOG.md	-> v1.0.2	17 months ago
 LICENSE	initial commit	17 months ago
 README.md	fix demo	17 months ago
 index.js	-> v1.0.2	17 months ago
 package-lock.json	initial commit	17 months ago
 package.json	-> v1.0.2	17 months ago

[pngwn / svelte-adapter](#)

Used by 4 Watch 3 Star 103 Fork 6

[Code](#) Issues 1 Pull requests 0 Actions Projects 0 Wiki Security Insights

### Use Svelte components with Vue and React

22 commits 1 branch 0 packages 0 releases 5 contributors

Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

[zxcabs](#) and [pngwn](#) Destroy svelte component when vue adepter destroyed ... Latest commit c60f7c8 on 1 Jul 2019

<a href="#">src</a>	Change vue watch syntax.	7 months ago
<a href="#">.gitignore</a>	Hmmmm...	7 months ago
<a href="#">README.md</a>	Update README.md	7 months ago
<a href="#">index.js</a>	Hmmmm...	7 months ago
<a href="#">package.json</a>	0.3.0	7 months ago
<a href="#">pwa.config.js</a>	Hmmmm...	7 months ago
<a href="#">react.js</a>	Add 'watch' option	7 months ago
<a href="#">rollup.config.js</a>	Hmmmm...	7 months ago
<a href="#">vue.js</a>	Destroy svelte component when vue adepter destroyed	7 months ago

# Denis Mishunov

## I created Frankenstein: 3 stories of migration



# Web Components aka Custom Elements



# WEB COMPONENTS

```
1 class BarCode extends HTMLElement {...}  
2  
3 window.customElements.define('bar-code', BarCode);  
4  
5  
6  
7 ...  
8  
9 <bar-code></bar-code>  
10
```

# WEB COMPONENTS

```
1 class BarCode extends HTMLElement {  
2     get open() {  
3         return this.hasAttribute('open');  
4     }  
5  
6     set open(val) {  
7         if (val) {  
8             this.setAttribute('open', '');  
9         }  
10    }  
11  
12    constructor() {  
13        this.addEventListener('click', () => {...})  
14        ...  
15    }  
16  
17    connectedCallback() {  
18        ...  
19    }  
20  
21    disconnectedCallback() {  
22        ...  
23    }  
24  
25    attributeChangedCallback(attrName, oldVal, newVal) {  
26        ...  
27    }  
28 }
```

# WEB COMPONENTS

```
1 // Attach a shadow root to custom element
2
3 const shadowRoot = this.attachShadow({mode: 'open'});
4
5 shadowRoot.innerHTML =
6     `
```

# **Why I don't use web components.**

 **Rich Harris** on dev.to

- A. You need JS almost everytime.
- B. CSS in JS.
- C. Platform fatigue.
- D. Polyfills.
- E. Eager rendering.
- F. Confusion between props and attributes.
- G. Leaky design.
- H. All those problem are solved but not in WC.

## SVELTE AS WC

```
1 <svelte:options tag="my-element">
2
3 <script>
4
5 </script>
6
7 <div>
8   now it's a custom element
9 </div>
```

LIBRARY	SCORE	BASIC TESTS	ADVANCED TESTS
React 16.12.0	71%	16/16	0/14

A horizontal progress bar consisting of a yellow segment followed by a grey segment.

### Handling data

React passes all data to Custom Elements in the form of HTML attributes. For primitive data this is fine, but the system breaks down when passing rich data, like objects or arrays. In these instances you end up with stringified values like `some-attr="[object Object]"` which can't actually be used.

[custom-elements-everywhere.com](http://custom-elements-everywhere.com)

LIBRARY	SCORE	BASIC TESTS	ADVANCED TESTS
Preact 10.0.1	100%	16/16	14/14



### Handling data

Preact uses a runtime heuristic to determine if it should pass data to Custom Elements as either properties or attributes. If a property is already defined on the element instance, Preact will use properties, otherwise it will fallback to attributes. The exception to this rule is when it tries to pass rich data, like objects or arrays. In those instances it will always use a property.

Пора в светлое будущее!

# **Chapter 2**

---

## **Preparing the app**



Введи свой факт



ПРИВЕТ, ДОБАВЬ  
ПАРУ ФАКТОВ О  
СЕБЕ!

КЛИКНИ НА МЕНЯ!

ТЫ МОЖЕШЬ  
ДВИГАТЬ НАС КАК  
ЗАХОЧЕШЬ!

ПОТЯНИ МЫШКОЙ ЗА  
КРАЙ ЛИСТОЧКА!

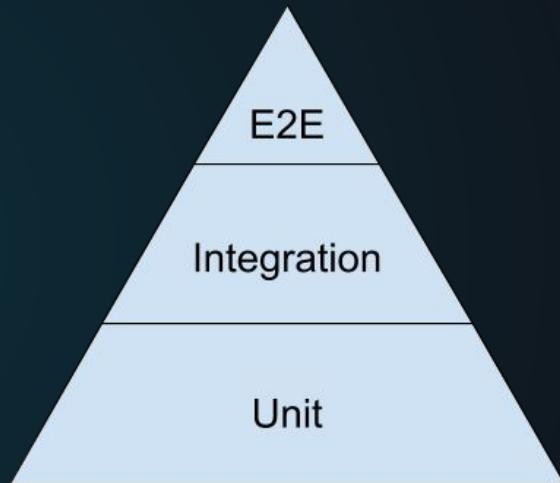
# Decomposition

1. First level - static html components. Changes only by outer state. Atoms.
2. Second level - positioning/aligning components aka styles containers.
3. Third level - logic containers/hoc/services/api
4. Four level - pages/containers for containers

**Sapper, Routing, APP level**



**Ughh.. tests?!**



# Basarat Ali Syed

## Documenting application behaviours with E2E tests



## CYPRESS .js

```
1 import po from './page-object';
2
3 describe('PovPage', () => {
4   beforeEach(() => {
5     cy.visit('http://localhost:8080');
6   });
7
8   it('should contains 4 default fuckts', () => {
9     cy.get(po.$fuckts).should('have.length', 4);
10  });
11
12  it('should add fuckt after filling input and press enter', () => {
13    po.addFuckt('new fuckt');
14
15    cy.get(po.$fuckts).should('have.length', 5);
16  });
17});
18
```

# CYPRESS JS

```
1 const PageObject = {
2   $fuckts: '[data-cy-fuckt]',
3   $createFucktInput: '[data-cy-create-fuckt]',
4   $saveMapBtn: '[data-cy-save-map]',
5   $saveMapNameInput: '[data-cy-save-map-name-input]',
6   $sidebarMenuItem: '[data-cy-sidebar-menu-item]',
7   $sidebarMenuItemNewMap: '[data-cy-sidebar-menu-item-new-map]',
8   $sidebarMenuItemRemoveMap: '[data-cy-sidebar-menu-item-remove-map]',
9   $fucktTextarea: '[data-cy-fuckt-textarea]',
10  addFuckt(text) {
11    cy.get('create-fuckt')
12      .find('input')
13      .type(text);
14
15    cy.get('create-fuckt')
16      .find('button')
17      .click();
18  },
19
20  ...
```

**So, why test?**



# Shmapter 3

---

## React to Preact

# PREACT

```
1 npm i -S preact preact-compat
2
3 ...
4
5 alias: {
6   react: 'preact/compat',
7   'react-dom/test-utils': 'preact/test-utils',
8   'react-dom': 'preact/compat',
9   svelte: path.resolve('node_modules', 'svelte'),
10 },
11
12 ...
13
14
15 import { h } from 'preact';
```

# TESTs

## ▼ PovPage

- ✓ should redirect to /fuckts/new after open
- ✓ should contains 4 default fuckts
- ✓ should add fuckt after filling input and press enter
- ✓ should save map after filling name
- ✓ should show correct count of fuckts
- ✓ should remove map

# **Chapter 4**

---

# **End Game**

# Migration

1. Rewrite 1,2,3,4 level's components to svelte and align them like in react
2. Use svelte stores if we need to manage state.
3. Get frankenstein - svelte components and pages working in react app.
4. Remove WC directives from components and get svelte free.
5. Use sapper if needed

# SVELTE IN REACT

```
<Content style={style.content.wrapper}>
  /* <CreateFucktInput handleCreateFuckt={handleCreateFuckt} /> */
  <svelte-create-fuckt handleCreateFuckt={handleCreateFuckt} />

  <DraggableBoard
    style={style}
    fuckts={fuckts}
    zoomOut={zoomOut}
    onUpdate={setFuckts}
```

# SVELTE IN REACT

```
▼<main class="ant-layout-content" style="padding: 5px 5px 10px;">
  ▼<svelte-create-fuckt>
    ▼#shadow-root (open)
      ▼<style>
        .create-fuckt{display:flex;align-items:center;width:100%;height:80px}.create-fuckt
          input{width:calc(100% - 80px);border-
            radius:8px}:root{background:#f5f6fa;color:#9c9c9c;font:1rem 'PT Sans', sans-
            serif}.btn{display:inline-
            block;background:transparent;color:inherit;font:inherit;border:0;outline:0;padding:0
            ;transition:all 200ms ease-in;cursor:pointer}.btn--primary{background:#7f8fff4;color:#fff;box-shadow:0 0 10px 2px rgba(0, 0,
            0.1);border-radius:2px;padding:12px 36px}.btn--primary:hover{background:#6c7fff2}.btn--primary:active{background:#7f8fff4;box-
            shadow:inset 0 0 10px 2px rgba(0, 0, 0, 0.2)}.btn--inside{margin-
            left:-96px}.form__field{width:360px;background:#fff;color:#a3a3a3;font:inherit;box-
            shadow:0 6px 10px 0 rgba(0, 0, 0, 0.1);border:0;outline:0;padding:22px 18px}
        </style>
      ▼<form data-cy-create-fuckt> == $0
        ▼<div class="create-fuckt">
          <input class="form__field" type="text">
          <button type="submit" class="btn btn--inside btn--primary">Добавить</button>
        </div>
      </form>
    </svelte-create-fuckt>
```

## TESTs

▼ PovPage

- ✓ should redirect to /fuckts/new after open
- ✓ should contains 4 default fuckts
- ✗ should add fuckt after filling input and press enter ⚠

▼ BEFORE EACH

(XHR)	● GET 200 /sockjs-node/info?t=15797...
1 VISIT	http://localhost:8080

▼ TEST

1 GET	svelte-create-fuckt	
2 - FIND	input	0
(XHR)	● GET 200 /sockjs-node/info?t=15797...	

CypressError: Timed out retrying: Expected to find element: 'input', but never found it. Queried from element: <svelte-create-fuckt>

## CYPRESS JS

```
1 import 'cypress-shadow-dom';
2
3 addFuckt(text) {
4   cy.shadowGet('svelte-create-fuckt')
5     .shadowFind('input')
6     .shadowType(text);
7
8   cy.shadowGet('svelte-create-fuckt')
9     .shadowFind('button')
10    .shadowClick();
11 },
12
13 ...
```

**That's easy one?**



## Limitations

1. Your styles will break
2. If you use styles library - your styles will break  
and you need rewrite it in svelte
3. If you forgot to add WC directive - your styles  
will break

**But...**

1. Confident structure but native JS.
2. No overhead.
3. Tru reactive.
4. Want to migrate from other framework - just skip step with preact.



Do experiments  
and do those a lot!

# THANKS!