Intro to

SOLIDJS

for React Developers (or anyone else)

Travis Waith-Mair

@travisWaithMair

non-traditional.dev

plex

BEDROCK

LAYOUT PRIMITIVES

Solid Feels Familiar to React Devs

```
import { render } from "solid-js/web";
import { createSignal } from "solid-js";

function Counter() {
  const [count, setCount] = createSignal(0);
  const increment = () => setCount(count() + 1);

  return (
    <button type="button" onClick={increment}>
      {count()}
    </button>
  );
}

render(() => <Counter />, document.getElementById("app")!);
```

Solid uses JSX, Components and Props

```
function Welcome(props) {
  return <p>Welcome, {props.name}</p>;
}
```

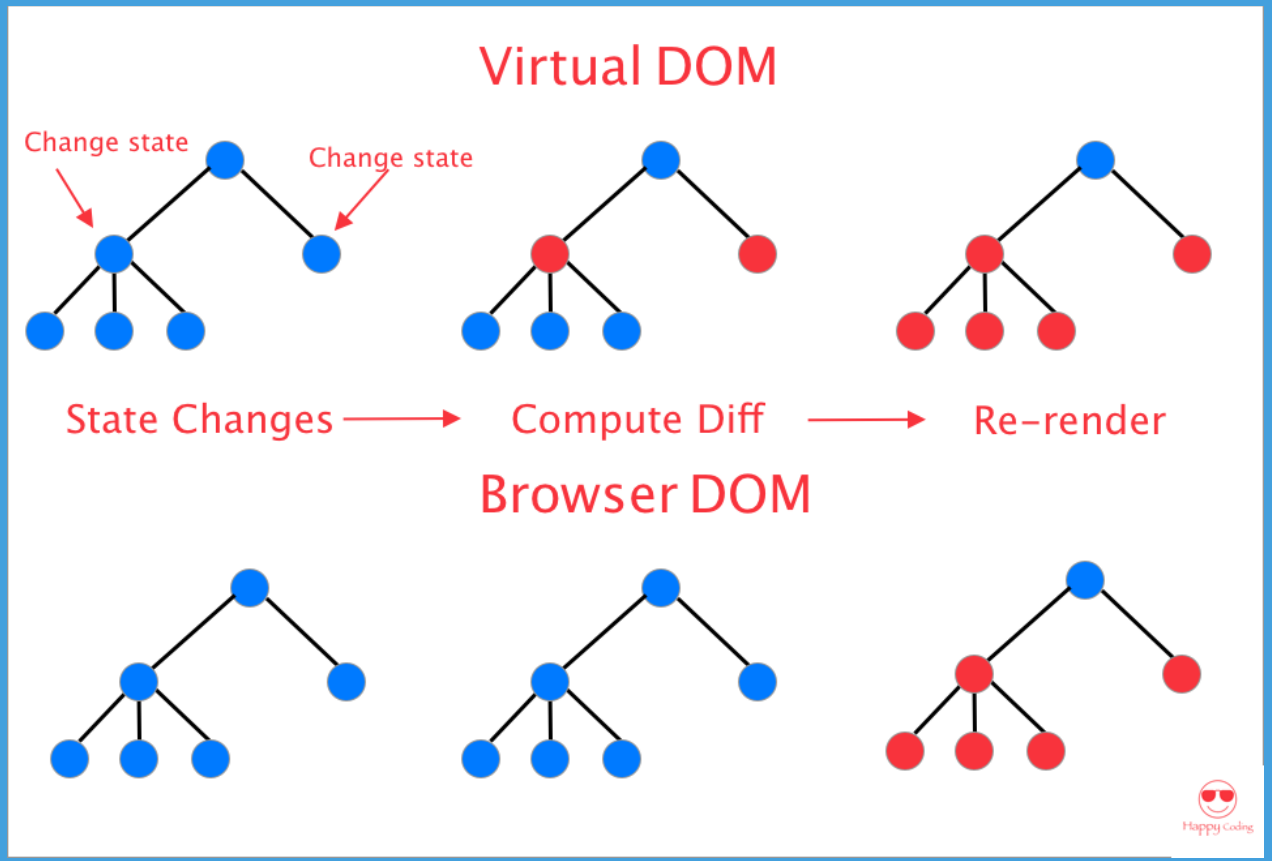
Solid primitives use a "Hook-like" API

```
const [user, setUser] = createSignal();

createEffect(() => {
      getUser(props.id)
      .then(user => {
            setUser(user)
      })

})
```

Solid is not a drop-in replacement of React

Reactivity instead of VDOM

React VDOM



img src: https://tonynguyenit.medium.com/

Reactivity

Solid JS will surgically update only those parts of the App that are dependent on reactive values

How does it do it?

Solid is a compiled framework

Suit of "fine-grained" reactive primitives

Solid does all the difficult work wiring up the reactivity during compilation

It's all about the Primitives

Components in Solid "disappear" after the initial render.

Solid is built on "fine-grained" reactive primitives.

Everything in Solid can be broken down into a Signal, a Memo, or an Effect.

createSignal

Signals are simply functions that return a value

```
const [count, setCount] = createSignal(0);

<p>The current count is: {count()}</p>
```

createEffect

Effects let you do actions based on when signals update

```
createEffect(() => {
  document.title = `The current count is: ${count()}`;
});
```

*No Dependency Array needed

createMemo

It is both and effect and a signal

```
const fullName = createMemo(() => `${firstName()} ${lastName()}`);

<p>{fullName()}</p>
```

Works best for expensive calculations

Simple Counter Example

```
function Counter() {
const [count, setCount] = createSignal(0);

createEffect(() => {
  document.title = `The current count is: ${count()}`;
});

return (
    <div>
      <p>The current count is: {count()}</p>
      <button onClick={() => setCount(count() + 1)}>Plus</button>
    </div>
  );
}
```

Primitives don't follow React's Hook rules

```
export const [count, setCount] = createSignal(0);

function Counter(props) {

if(props.logCount){
  createEffect(() => {
    console.log(count());
  });
}

return (
    <div>
      <p>The current count is: {count()}</p>
      <button onClick={() => setCount(count() + 1)}>Plus</button>
    </div>
  );
}
```

Gotchas coming from React

"False Friends" in Spanish

English		Spanish
University	→	Universidad
President	→	Presidente
Embarrassed	→	Embarazada?

Solid has it's "false friends" for React Devs

Don't return early

```
function DataDisplay(props){

  if(props.data === undefined){
      return <div>Loading...</div>
  }

  return <div>{/* display props.data */}</div>
}
```

Instead, use Control Flow components

```
import { Show } from "solid-js";

funciton DataDisplay(props){
  return (
    <Show when={props.data} fallback={<div>Loading...</div>}>
        <div>{/* display props.data */}</div>
    </Show>
    );
}
```

Other Control Flow components like Switch, For, ErrorBoundries, and Suspense.

Don't do logic outside of JSX or reactive primitives

```
function DoubleCounter() {
const [count, setCount] = createSignal(0);

const doubleCount = count() * 2 //NO NO

return (
    <div>
      <p>The double count is: {doubleCount}</p>
      <button onClick={() => setCount(count() + 1)}>Plus</button>
    </div>
  );
}
```

Derive the value in a function

```
function DoubleCounter() {
const [count, setCount] = createSignal(0);

const doubleCount = () => count() * 2

return (
    <div>
      <p>The double count is: {doubleCount()}</p>
      <button onClick={() => setCount(count() + 1)}>Plus</button>
    </div>
  );
}
```

OR.....

In the JSX directly

```
function DoubleCounter() {
const [count, setCount] = createSignal(0);

return (
    <div>
      <p>The double count is: {count()*2}</p>
      <button onClick={() => setCount(count() + 1)}>Plus</button>
    </div>
  );
}
```

Don't destructure Props

```
function Greeting({salutation, name}){
        return <p>{salutation}, {name}</p>
}

// or

function Greeting(props){
        const {salutation, name} = props

        return <p>{salutation}, {name}</p>
}
```

Prop values could be reactive

```
<Greeting salution="Hello" name={userName()}/>
```

```
props = {
  salutation:'Hello',
  get name(){
        return userName()
  }
}
```

Instead use props in JSX

```
function Greeting(props}){
        return <p>{props.salutation}, {props.name}</p>
}
```

Solid also provide the **splitProps** and **mergeProps** utility functions. These utilities allow you to safely split or merge props without breaking reactivity

There is no need for useRef or useCallback

```
import {  onMount } from "solid-js";

function MyForm() {
  const handleSubmit = (e) => {
    /* handle submit */
  };

  let myInput;

  onMount(() => myInput.focus());

  return (
    <form onSubmit={handleSubmit}>
      <input ref={myInput} />
    </form>
  );
}
```

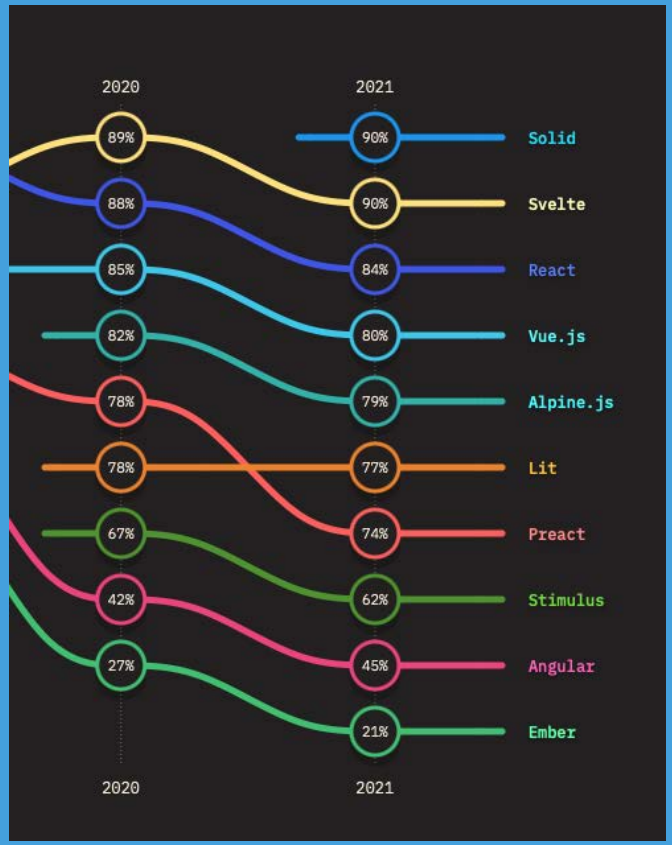
Before you give up

There is an eslint plugin

```
npm install --save-dev eslint eslint-plugin-solid
# or
pnpm add --save-dev eslint eslint-plugin-solid
yarn add --dev eslint eslint-plugin-solid

# optional, to create an ESLint config file
npx eslint --init
# or
pnpm eslint --init
yarn eslint --init
```

Why Bother Learning Solid

Solid has the highest Satisfaction rating



StateofJS 2021 results

Solid.JS is Performant by default

Solid.JS has a robust ecosystem

https://www.solidjs.com/ecosystem

- Routers
- UI Components
- Tooling
- Starters

Shameless plug for Solid Bedrock Layout

Solid Start Meta Framework

https://start.solidjs.com/getting-started/what-is-solidstart

Where do I go from here?

solidjs.com

Tutorial:
https://www.solidjs.com/tutorial/introduction_basics

Playground:
https://playground.solidjs.com/

Solidjs Discord

https://discord.gg/solidjs

Thank You

Twitter:
@travisWaithMair

Blog:
non-traditional.dev