

SOLIDJS

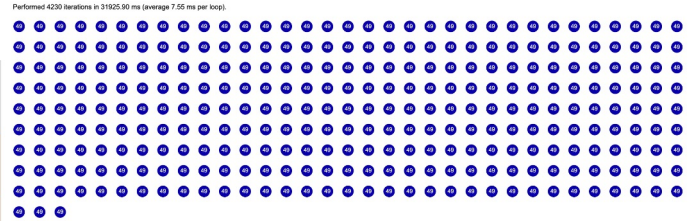
The World Beyond Components

Docs and information available at **www.solidjs.com**

SolidJS: Performance Champion

Duration in milliseconds ± standard deviation (Slowdown = Duration / Fastest)

Name	vanillajs-keyed	solid-v0.1.0-keyed	inferno-v5.3.0-keyed	elm-v0.19.0-bugfix2-keyed	hyperhtml-v2.13.0-keyed	preact-v8.2.6-keyed	vue-v2.5.16-keyed	svelte-v2.9.7-keyed	angular-v6.1.0-keyed	marko-v4.12.3-keyed	react-v16.4.1-keyed	mithril-v1.1.1-keyed	hyperapp-v1.2.9-keyed	ember-v3.3.0-keyed	knockout-v3.4.1-keyed
create rows Duration for creating 1000 rows after the page loaded.	126.7 ± 4.2 (1.0)	128.5 ± 4.1 (1.0)	141.0 ± 5.6 (1.1)	165.3 ± 9.3 (1.3)	177.6 ± 6.4 (1.4)	174.6 ± 9.3 (1.4)	182.1 ± 7.6 (1.4)	220.4 ± 4.9 (1.7)	185.2 ± 10.2 (1.5)	169.8 ± 8.2 (1.3)	180.5 ± 7.3 (1.4)	170.5 ± 7.6 (1.3)	145.3 ± 5.1 (1.1)	406.8 ± 15.8 (3.2)	336.7 ± 14.1 (2.7)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	134.6 ± 2.3 (1.0)	137.6 ± 2.3 (1.0)	136.7 ± 1.8 (1.0)	162.9 ± 11.9 (1.2)	172.5 ± 1.3 (1.3)	157.3 ± 4.5 (1.2)	158.8 ± 2.7 (1.2)	231.6 ± 3.3 (1.7)	161.2 ± 2.7 (1.2)	161.6 ± 3.0 (1.2)	157.3 ± 2.0 (1.2)	156.7 ± 3.3 (1.2)	162.0 ± 3.4 (1.2)	269.1 ± 23.6 (2.0)	335.9 ± 6.6 (2.5)
partial update Time to update the text of every 10th row (with 5 warmup iterations) for a table with 10k rows.	65.1 ± 2.6 (1.0)	64.5 ± 3.3 (1.0)	67.6 ± 2.7 (1.0)	80.1 ± 5.7 (1.2)	79.2 ± 4.2 (1.2)	96.2 ± 3.0 (1.5)	156.4 ± 9.8 (2.4)	75.1 ± 3.8 (1.2)	68.8 ± 3.7 (1.1)	89.9 ± 5.1 (1.4)	81.9 ± 2.7 (1.3)	134.9 ± 4.2 (2.1)	288.8 ± 18.9 (4.5)	134.9 ± 5.9 (2.1)	70.4 ± 3.4 (1.1)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	11.0 ± 2.3 (1.0)	9.5 ± 2.5 (1.0)	12.1 ± 3.2 (1.0)	10.6 ± 6.4 (1.0)	10.7 ± 3.5 (1.0)	10.5 ± 3.5 (1.0)	10.6 ± 2.0 (1.0)	10.7 ± 1.0 (1.0)	7.9 ± 4.3 (1.0)	8.4 ± 3.8 (1.0)	10.3 ± 2.1 (1.0)	8.7 ± 2.5 (1.0)	16.0 ± 2.2 (1.0)	8.7 ± 1.9 (1.0)	10.0 ± 2.8 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	17.5 ± 6.7 (1.0)	17.9 ± 2.8 (1.0)	18.3 ± 4.6 (1.0)	18.4 ± 5.8 (1.1)	21.9 ± 5.3 (1.3)	23.1 ± 2.9 (1.3)	20.0 ± 2.9 (1.1)	20.4 ± 4.4 (1.2)	105.8 ± 1.8 (6.1)	104.4 ± 1.2 (6.0)	106.5 ± 1.9 (6.1)	107.4 ± 1.5 (6.1)	25.5 ± 2.7 (1.5)	122.0 ± 2.9 (7.0)	108.6 ± 1.9 (6.2)
remove row Duration to remove a row. (with 5 warmup iterations).	46.1 ± 0.9 (1.0)	48.0 ± 1.5 (1.0)	47.9 ± 2.2 (1.0)	61.2 ± 4.7 (1.3)	52.0 ± 1.7 (1.1)	49.3 ± 0.6 (1.1)	54.2 ± 2.2 (1.2)	48.2 ± 1.0 (1.0)	47.1 ± 3.0 (1.0)	47.6 ± 1.9 (1.0)	49.6 ± 0.8 (1.1)	50.2 ± 2.1 (1.1)	60.1 ± 4.6 (1.3)	55.7 ± 1.0 (1.2)	52.9 ± 1.0 (1.1)
create many rows Duration to create 10,000 rows	1,229.1 ± 39.7 (1.0)	1,313.1 ± 55.4 (1.1)	1,338.0 ± 42.5 (1.1)	1,663.2 ± 57.6 (1.4)	2,011.2 ± 81.3 (1.6)	1,852.0 ± 51.6 (1.5)	1,603.2 ± 34.8 (1.3)	2,376.0 ± 40.7 (1.9)	1,693.9 ± 70.1 (1.4)	1,562.1 ± 44.1 (1.3)	1,935.4 ± 33.6 (1.6)	1,519.1 ± 71.8 (1.2)	1,474.5 ± 35.9 (1.2)	2,931.9 ± 42.9 (2.4)	3,081.0 ± 130.9 (2.5)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	205.6 ± 4.0 (1.0)	209.5 ± 6.8 (1.0)	212.4 ± 4.1 (1.0)	244.8 ± 3.7 (1.2)	265.5 ± 7.7 (1.3)	271.7 ± 3.8 (1.3)	342.5 ± 6.0 (1.7)	354.4 ± 11.8 (1.7)	243.3 ± 6.3 (1.2)	270.9 ± 7.2 (1.3)	268.6 ± 6.9 (1.3)	301.1 ± 11.0 (1.5)	541.9 ± 23.7 (2.6)	403.7 ± 32.5 (2.0)	3,352.9 ± 71.8 (16.3)
clear rows Duration to clear the table filled with 10.000 rows.	131.4 ± 3.9 (1.0)	136.2 ± 2.7 (1.0)	149.1 ± 3.3 (1.1)	166.2 ± 2.1 (1.3)	152.7 ± 2.2 (1.2)	194.1 ± 2.1 (1.5)	191.9 ± 6.1 (1.5)	183.5 ± 4.1 (1.4)	263.9 ± 3.0 (2.0)	215.4 ± 1.8 (1.6)	175.4 ± 4.1 (1.3)	182.7 ± 1.6 (1.4)	264.1 ± 5.8 (2.0)	203.1 ± 3.6 (1.5)	466.7 ± 40.7 (3.6)
slowdown geometric mean	**1.00**	**1.03**	**1.06**	**1.21**	**1.25**	**1.29**	**1.38**	**1.39**	**1.50**	**1.50**	**1.50**	**1.56**	**1.62**	**2.11**	**2.69**





Who am I?


netlify
Ryan Carniato
Principal Engineer OSS
Creator of SolidJS

Where’s the Revolution?!

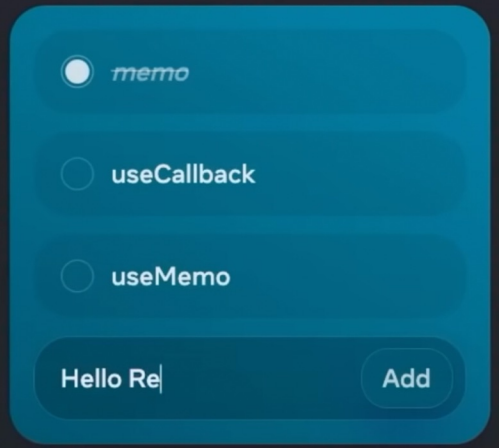



Components Are Pure Overhead
97

When to Memo?

```
function TodoList() {
  const [todos, setTodos] = useState(initialTodos);
  const handleChange = todo => setTodos(todos => getUpdated(todos, todo));

  return (
    <div>
      <ul>
        {todos.map(todo => (
          <Todo key={todo.id} todo={todo} onChange={handleChange} />
        ))}
      </ul>
      <AddTodo setTodos={setTodos} />
    </div>
  );
}
```



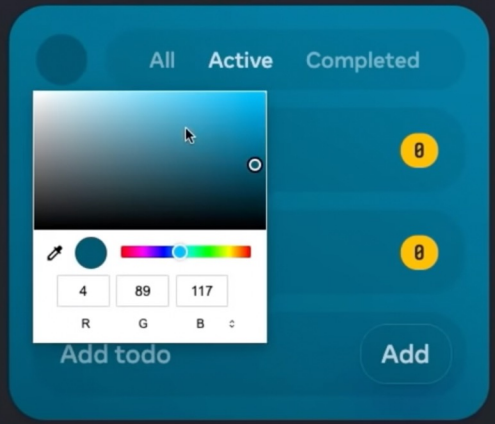
```jsx
const Todo = React.memo(UnmemoizedTodo);

function TodoList({ visibility, themeColor }) {
  const [todos, setTodos] = useState(initialTodos);
  const handleChange = useCallback(
    todo => setTodos(todos => getUpdated(todos, todo)),
    []
  );
  const filtered = useMemo(
    () => getFiltered(todos, visibility),
    [todos, visibility]
  );
  return (
    <div>
      <ul>
        {filtered.map(todo => (
          <Todo key={todo.id} todo={todo} onChange={handleChange} />
        ))}
      </ul>
      <AddTodo setTodos={setTodos} themeColor={themeColor} />
    </div>
  );
}

function BlazingTodoList() {
  const [visibility, ...] = useState("all");
  const [themeColor, ...] = useState("#045975");
  return ...<TodoList visibility={visibility} themeColor={themeColor} />...;
}
```

(6X CPU SLOWDOWN)

getFiltered() was called 3 times



```
import React, { useState } from 'react';

function Example() {
    const [count, setCount] = useState(0);
}
```



```
let count = 0;
```



```
function TodoList({ visibility, themeColor }) {
  const [todos, setTodos] = useState(initialTodos);

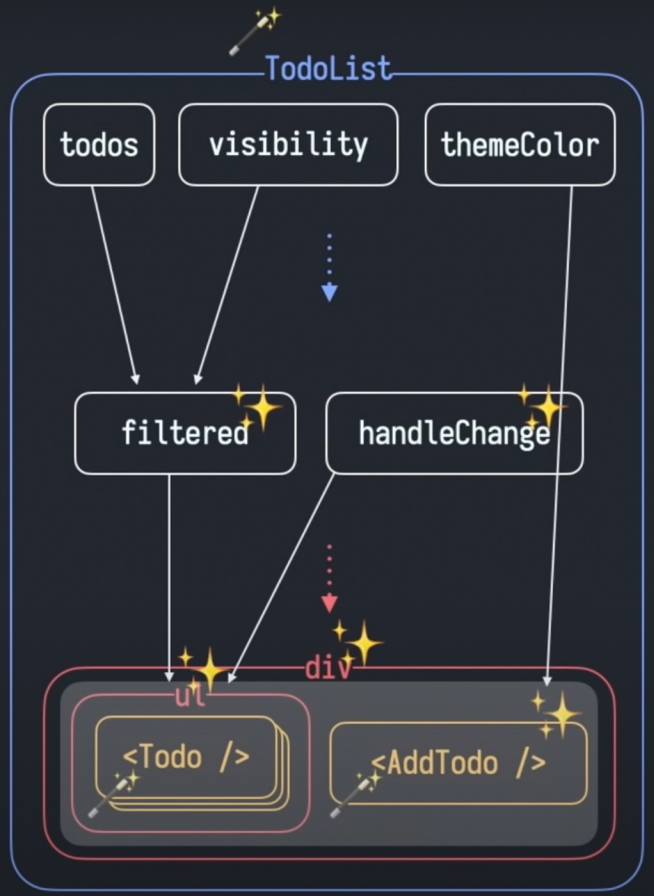
  let hasVisibilityChanged, hasThemeColorChanged, hasTodosChanged, memoCache;

  if (hasVisibilityChanged || hasThemeColorChanged || hasTodosChanged) {
    const handleChange =
      memoCache[0] ||
      (memoCache[0] = todo => setTodos(todos => getUpdated(todos, todo)));

    let filtered, jsx_todos;
    if (hasVisibilityChanged || hasTodosChanged) {
      filtered = memoCache[1] = getFiltered(todos, visibility);
      jsx_todos = memoCache[2] = (<ul>{filtered.map(...)}</ul>);
    } else {
      filtered = memoCache[1];
      jsx_todos = memoCache[2];
    }

    const jsx_addTodo = hasThemeColorChanged
      ? (memoCache[3] = <AddTodo setTodos={setTodos} themeColor={themeColor} />)
      : memoCache[3];

    return (memoCache[4] = <div>{jsx_todos}{jsx_addTodo}</div>);
  } else {
    return memoCache[4];
  }
}
```



Alternative Reality



Hello World

```
function greet(name) {
  console.log(`Hello ${name}`); // Hello John
}riend = "Mary";
console.log(`Hello ${friend}`); // Hello Mary
let friend = "John";
greet(friend); // Hello John
friend = "Mary";
greet(friend); // Hello Mary
```

What’s Reactive Programming?

$$a = b + c$$

* where the value of **a** updates whenever the value of **b** or **c** changes.

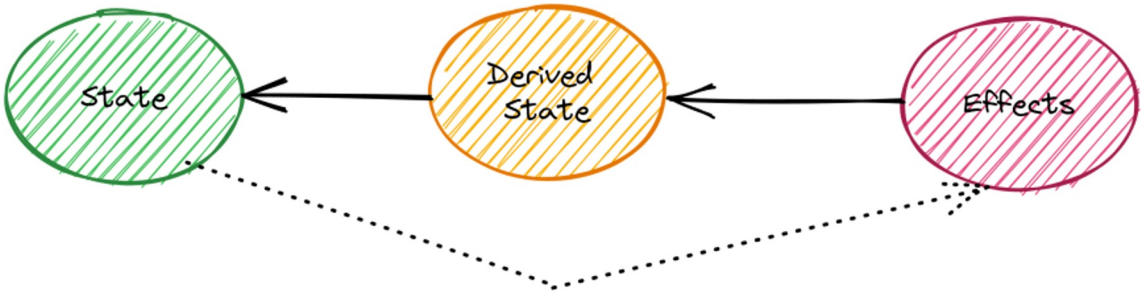
Fine-Grained Reactive Programming

Declarative

Composable

Simple model consists of only 3 concepts:

- Signals
- Derivations
- Effects




So like React Hooks?
LIGHTYEAR

Enter Reactivity

```
const [friend, setFriend] = createSignal("John");
const upper = () => friend().toUpperCase();
  () => console.log(`Hello ${friend()}`)
createEffect(ohn
  () => console.log(`Hello ${upper()}`)l("John");
); // Hello JOHN); // Hello Mary
setFriend("Mary");
setFriend("Mary"); // Hello MARY
```

```
function createSignal(value) {
  const subscribers = new Set();
  const read = () => {
    const current = getCurrentObserver();
    if (current) subscribers.add(current);
    return value;
  };
  const write = (nextValue) => {
    value = nextValue;
    for (const sub of subscribers) sub();
  };
  return [read, write];
}
```

```
const context = [];

function getCurrentObserver() {
  return context[context.length - 1];
}

function createEffect(fn) {
  const execute = () => {
    context.push(execute);
    try {
      fn();
    } finally {
      context.pop();
    }
  };

  execute();
}
```

Other Primitives

```
const expensive = createMemo(
  () => computeExpensive(signal())
);

const [todos, setTodos] = createStore([
  { id: 1, title: "Hello", done: false }
]);

const [data, { refetch }] = createResource(
  () => fetchUser(`/api/users`)
);
```

Component Props

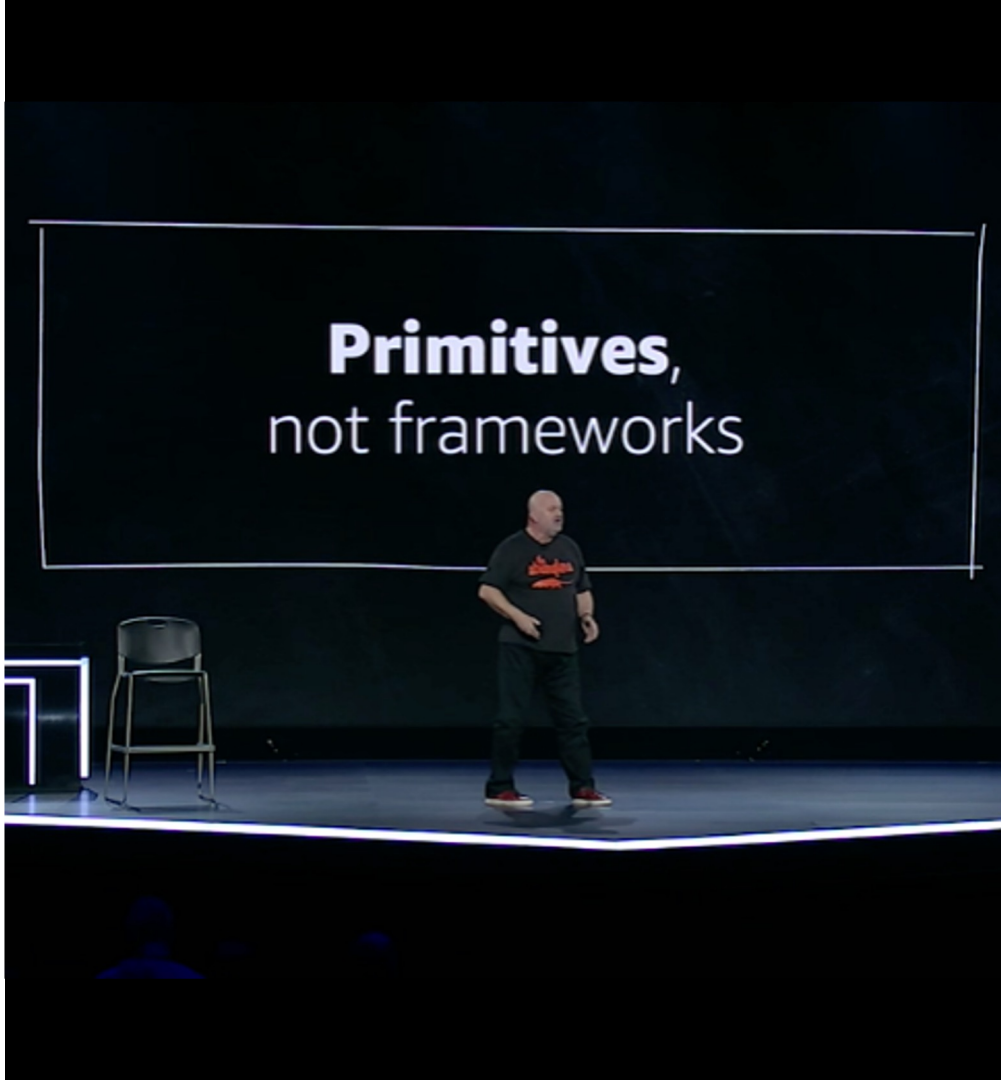
```
Greet({
  get name() { return name(); },
  onChange: () => setName(name() + "!")
});
```

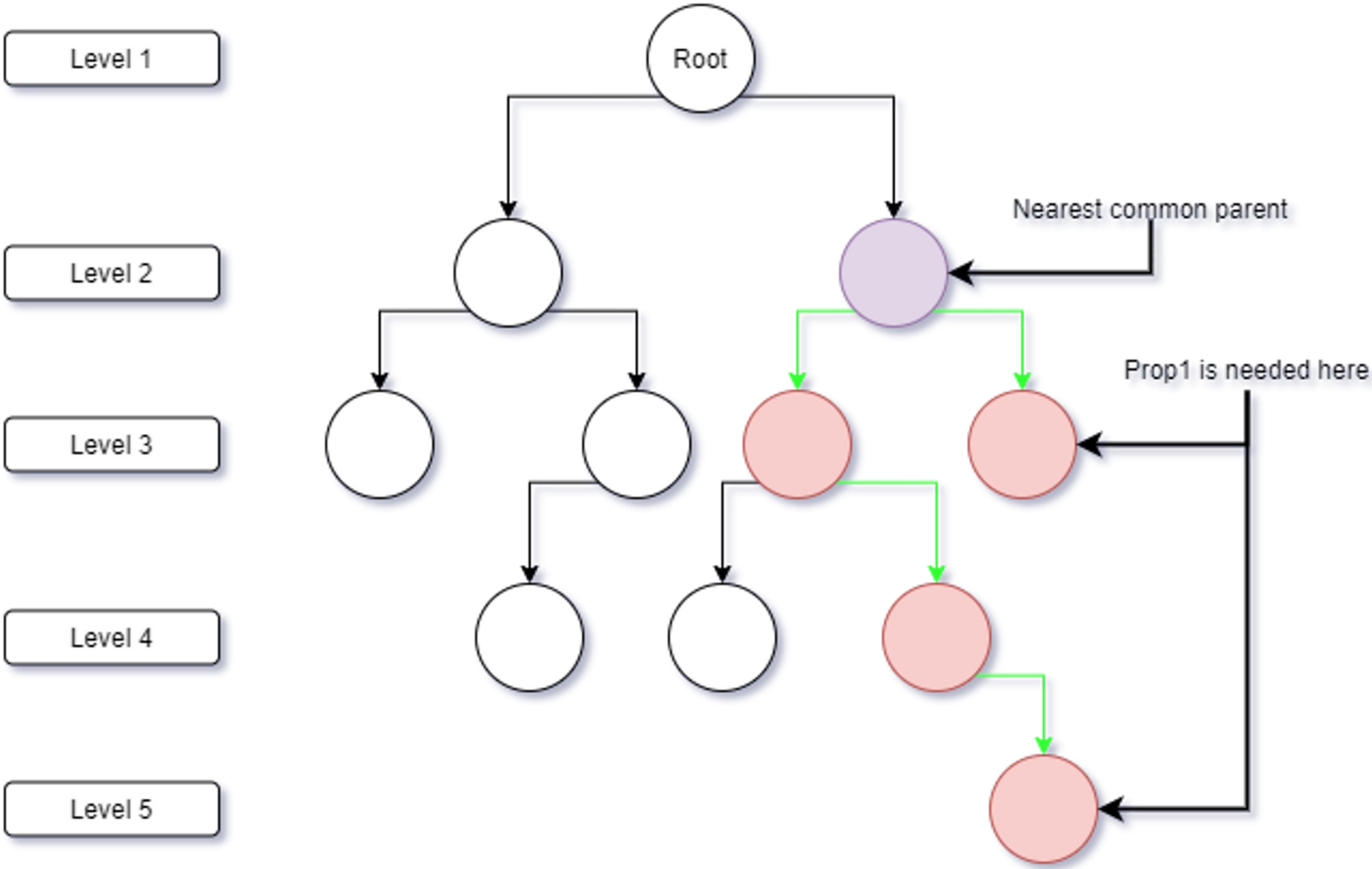
Controlling the Flow

```
<ul>{
  <Paginated each={list()}>{
    (item) => <li>{item}</li>}
  </Paginated>
</ul>>
```

Reactive Advantage

- Components Run Once
- Templates compile to Real DOM Nodes
- State Independent of Component




Level 1
Level 2
Level 3
Level 4
Level 5
Root
Nearest common parent
Prop1 is needed here

Local State

```
function Counter() {
  const [count, setCount] = createSignal(0);
  return <div>{count()}<div>
}
```



Global State

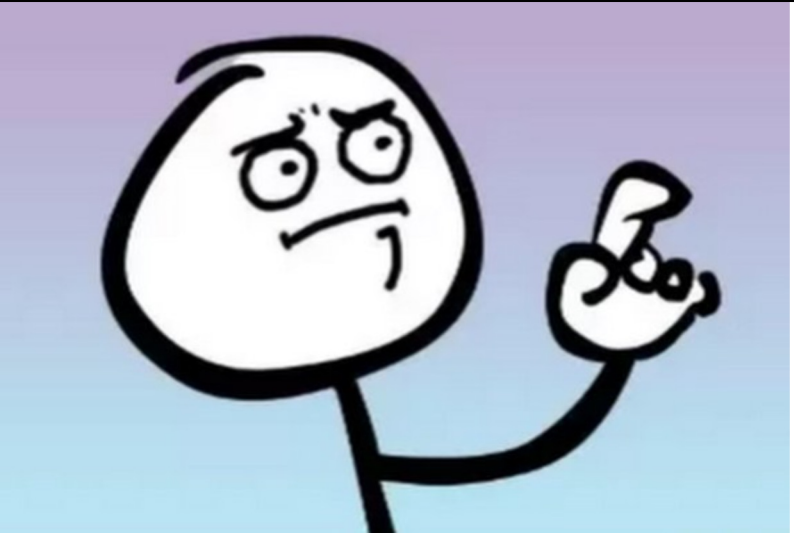
```
const [count, setCount] = createSignal(0);
function Counter() {
  return <div>{count()}<div>
}
```





Local State

```
let count = 0;
```



Global State

. . .

Reactivity: The Framework Rosetta Stone



```svelte
<script>
  let count = 0;
  $: doubleCount = count * 2;
  $: console.log("Double count is", doubleCount);
</script>

<button on:click={() => count++}>
  Count: {count}
</button>
```

```react
function Counter() {
  const [count, setCount] = useState(0);
  const doubleCount = useMemo(() => count * 2, [count]);
  useEffect(() => {
    console.log("Double count is", doubleCount);
  }, [doubleCount]);

  return <button onClick={() => setCount(count + 1)}>
    Count: {count}
  </button>
}
```



Build me a performant eCommerce web application

Sure. No problem. To accomplish this task please answer the following questions:

1. What are the specific requirements and features you want to include in your eComme web application?
2. What is the expected traffic volume and number of concurrent users for your web application?
3. What is your budget and timeline for the project?
4. Do you have any specific technology or platform preferences for your web applicatio
5. Do you have an existing design or branding that needs to be incorporated into the application?



Joe Savona
@en_JS

Starting late last year, @_gsathya, @potetotes, @zmofei, and I rewrote the compiler from scratch to align w the auto-reactivity model. The post talks about how we incrementally built up from a small subset of the language in order to quickly validate the overall design.

2:02 PM · Mar 22, 2023 · **12.7K** Views

---

1 Retweet **16** Likes

What Executes?

```
function ShowTotal({ items, federalTaxRate, stateTaxRate }) {
  const subTotal = calculateSubTotal(items);
  const federalTax = calculateTax(subTotal, federalTaxRate);
  const stateTax = calculateTax(subTotal, stateTaxRate);

  return <div>{ subTotal + federalTax + stateTax }</div>
}
```

Experimental new compilation strategy

```
import { ref, effect } from 'vue'
import { template, on, setClass, setAttr, setText } from 'vue/vapor'

const t0 = template(`<div><button>`)

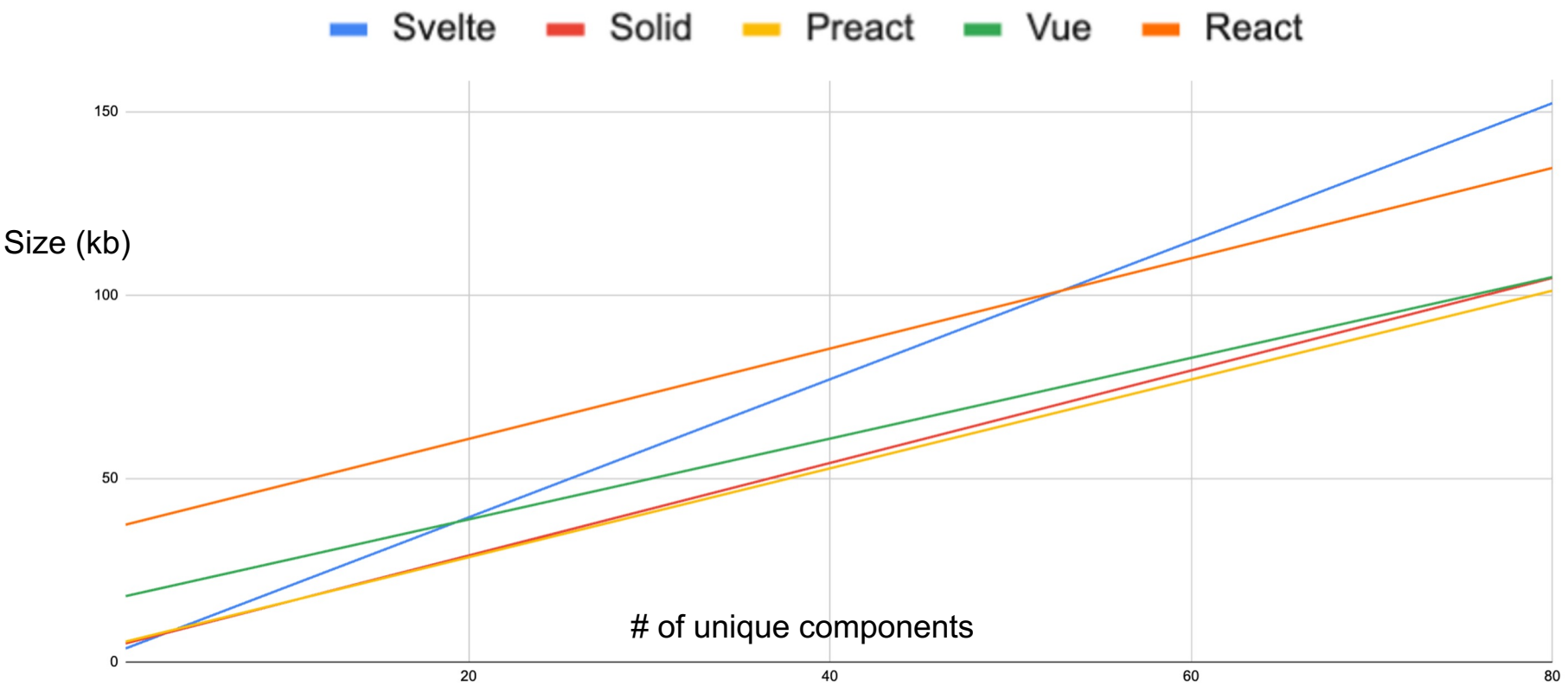
export default () => {
  const count = ref(0)

  let div = t0()
  let button = div.firstChild
  let _button_class, _button_id, _button_text
  effect(() => {
    setClass(button, _button_class, (_button_class = { red: count.value % 2 }))
    setAttr(button, 'id', _button_id, (_button_id = `foo-${count.value}`))
    setText(button, _button_text, count.value)
  })
  on(button, 'click', () => count.value++)
  return div
}
```

Conquering Hydration



SPAs Scale Linearly



Taste Movies App - https://tastejs.com/movies/

Framework	Demo	Size
Next	https://next-movies-zeta.vercel.app/	190kb
Nuxt	https://movies.nuxt.space/	90.8kb
Angular	https://angular-movies-a12d3.web.app/	121kb
SvelteKit	https://sveltekit-movies.netlify.app/	34.8kb
Lit	https://lit-movies.netlify.app/	108kb

```
<button onClick={server$(() => {
  console.log('I run on the server!')
})}>
  Click me
</button>
```

Never delivered to the browser

Traceability


Reactivity: Gateway to the future


www.solidjs.com
@RyanCarniato
youtube.com/@ryansolid