

The background is a dark purple gradient. It features several abstract geometric elements: a large dark blue circle in the center containing the text; a pink circle with diagonal stripes on the left; a blue circle with a dotted pattern below it; a yellow zigzag line on the far left; a yellow triangle with a dashed outline above the central circle; a solid yellow triangle below it; a light blue dashed circle to the left of the central circle; a light blue dashed circle above the central circle; a light blue dashed circle below the central circle; a light blue dashed circle to the right of the central circle; a pink dashed triangle to the right of the central circle; a solid pink triangle above it; a solid pink pentagon below it; a yellow triangle with vertical stripes to the right of the central circle; a solid yellow circle on the far right; and a solid pink circle on the far right.

# Multi-window Web Apps with React



# Hello!

**I'm Blake Zimmerman**

I am a software engineer building cutting-edge tools for  
Intelligent Retail Lab – Walmart's in-store AI lab

# The Web: A Growing Application Platform



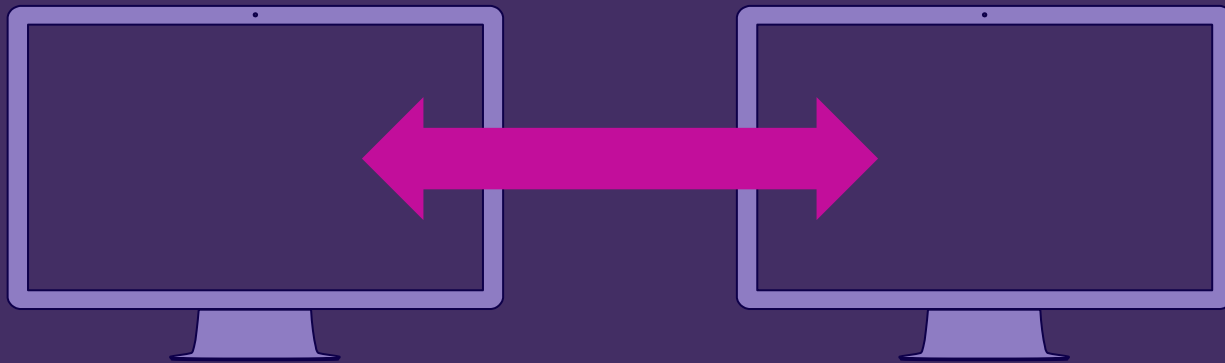
- Web apps can do more than ever
- Replacing desktop counterparts in areas



# Web APIs: Unlocking Potential



- WebUSB API (Connecting USB devices)
- WebGL (Render interactive 3D graphics)
- Web Assembly (Near-native performance)
- Bluetooth API (Connect to wireless devices)
- Notifications API (Display system notifications)
- Service Workers (Offline usage)



# Multi-Window

Enabling seamless communication between multiple  
windows running the same application

# Agenda

Use cases for multi-window

Web APIs we might use

Write a React hook

Demo the result 🎉

# Use Cases



Utilizing multi-window capabilities allows you to create applications that are not confined to a single browser window.

Possible use cases:

- Detect user actions in other tabs
- Know when a user logs into their account in another window/tab
- Instruct a worker to perform some background work
- Share state between multiple windows



{ api }

Web APIs







1.

`window.postMessage()`



# window.postMessage()



## Basic Usage

```
const newWindow = window.open("your-url.com");

// Send message
newWindow.postMessage({ data: "demo" });

// Receive message
function receiveMessage(event) {
  if (event.origin !== "your-url.com") return;

  // handle event.data
}

window.addEventListener("message", receiveMessage);
```

# window.postMessage()



## Pros

- 98.33% Browser Compatibility
- Can be used cross-origin
- Simple usage

## Cons

- Requires a handle to the other window
- Refreshing either window puts them out of sync



2.

SharedWorker



# SharedWorker

## Basic Usage

```
// Initialize
const worker = new SharedWorker("worker.js");
worker.port.start();

// Send message to work
worker.port.postMessage("message to worker");

// Receive message
worker.port.onmessage = (event) => {
  // Handle event.data
};
```

# SharedWorker



## Set up shared worker

```
// worker.js
onconnect = (event) => {
  const port = event.ports[0];

  port.addEventListener("message", (e) => {
    // handle e.data

    // Send message to clients
    port.postMessage("message to clients");
  });

  port.start();
};
```

# SharedWorker



## Pros

- Powerful and robust
- Windows do not have to know about each other

## Cons

- Cannot be polyfilled
- 40.50% Browser Compatibility
- Same-origin only
- More complex usage



3.

# Broadcast Channel API





# Broadcast Channel API

## Basic Usage

```
// Initialize
const channel = new BroadcastChannel("myChannel");

// Send message
channel.postMessage("this is a message");

// Receive message
channel.onmessage = (event) => {
  // Handle event.data
};
```

# Broadcast Channel API



## Pros

- Windows do not have to know about each other
- Can be “polyfilled”
- Simple usage

## Cons

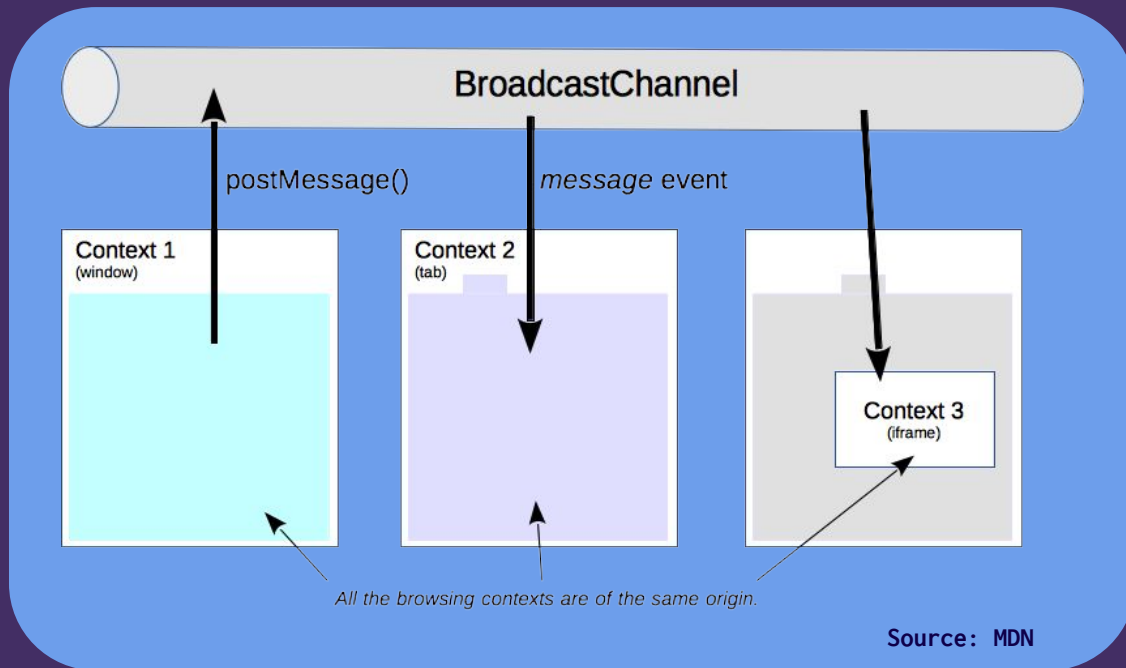
- Same-origin only
- 74.71% Browser Compatibility

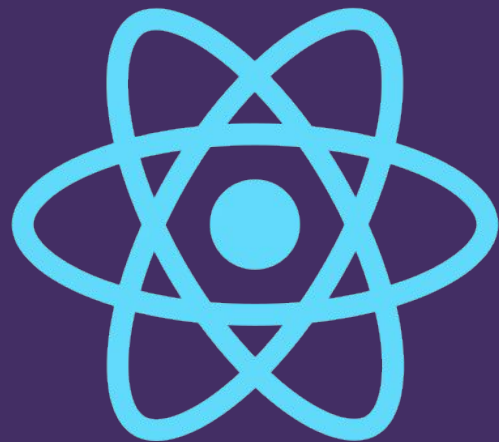
# Comparison



	<code>window.postMessage</code>	<code>SharedWorker</code>	<code>BroadcastChannel</code>
Ease of use	★ ★	★	★ ★ ★
Robustness	★	★ ★ ★	★ ★
Compatibility	★ ★ ★	★	★ ★

# More on Broadcast Channel API





Demo Time

# Takeaways



**Broadcast Channel API** is a simple and effective way to enable seamless communication between browser windows.

Possible use cases:

- Detect user actions in other tabs
- Know when a user logs into their account in another window/tab
- Instruct a worker to perform some background work
- Share state between multiple windows



# Thanks!

**Any questions?**

You can find the slides and code examples at  
[github.com/blakezimmerman/multi-window-react](https://github.com/blakezimmerman/multi-window-react)