THE CODE BEHIND COLLABORATION COMPONENTS

VÍTOR NORTON
DEV ADVOCATE @ SUPERVIZ

VITOR NORTON

Brazilian
10x Microsoft MVP
Developer Advocate @ SuperViz

@vt_norton

VITOR NORTON

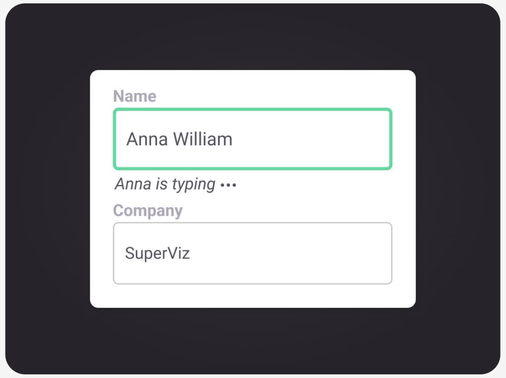
Passionate about connecting people
Highly dependent of my productivity tools
Love the idea of working anywhere in the world

Brazilian
10x Microsoft MVP
Developer Advocate @ SuperViz

@vt_norton

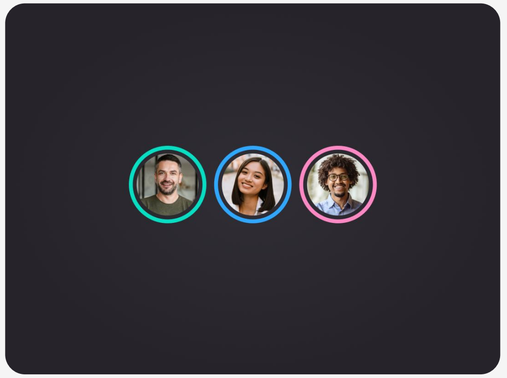
COLLABORATION COMPONENTS

COLLABORATION COMPONENTS



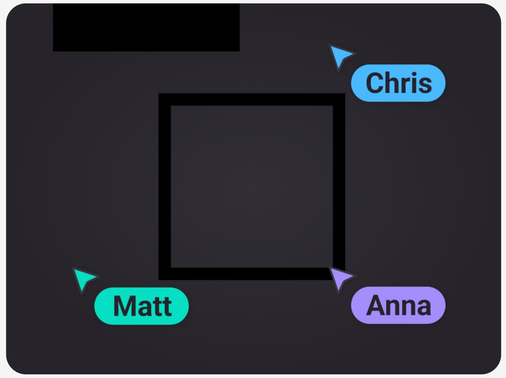
FORMS ELEMENTS

Enables real-time sync of form elements, such as input fields, checkboxes, among participants in the same room.



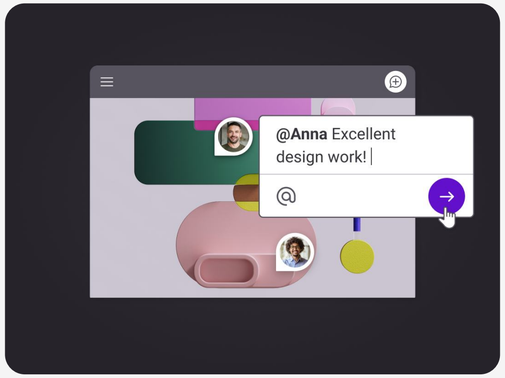
WHO-IS-ONLINE

The Who-is-On-line component allows you to see all the participants who are connected in a room.



MOUSE POINTERS

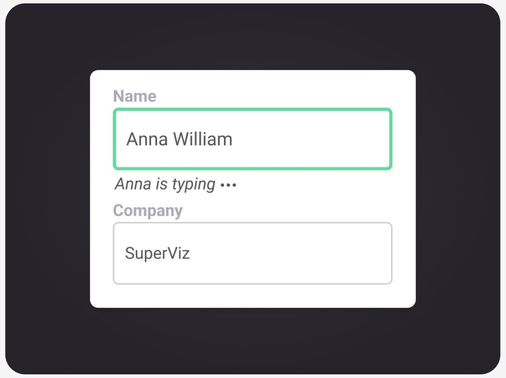
It enables real-time tracking of participants' cursor movements, allowing seamless collaboration within the same room.



CONTEXTUAL COMMENTS

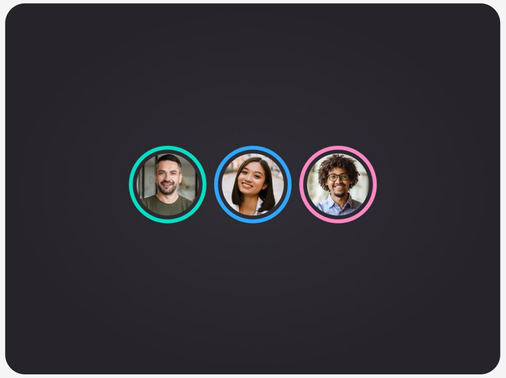
It embeds a customizable commenting experience into your page to enable people to collaborate.

COLLABORATION COMPONENTS



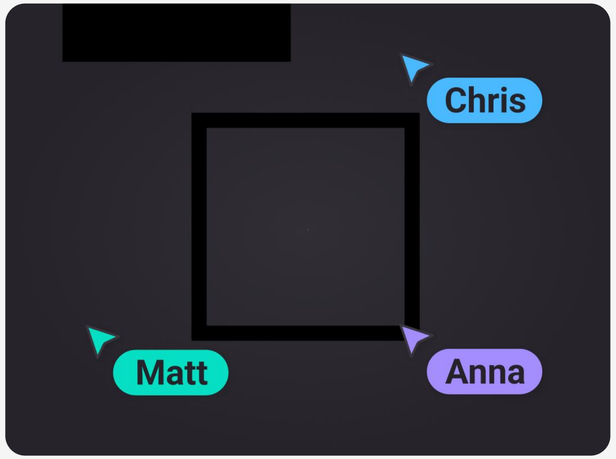
FORMS ELEMENTS

Enables real-time sync of form elements, such as input fields, checkboxes, among participants in the same room.



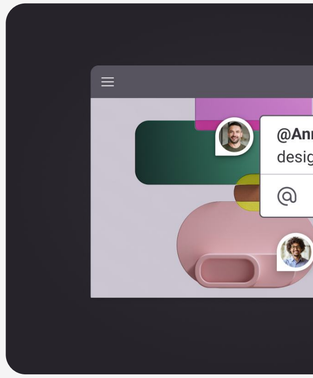
WHO-IS-ONLINE

The Who-is-On-line component allows you to see all the participants who are connected in a room.



MOUSE POINTERS

It enables real-time tracking of participants' cursor movements, allowing seamless collaboration within the same room.

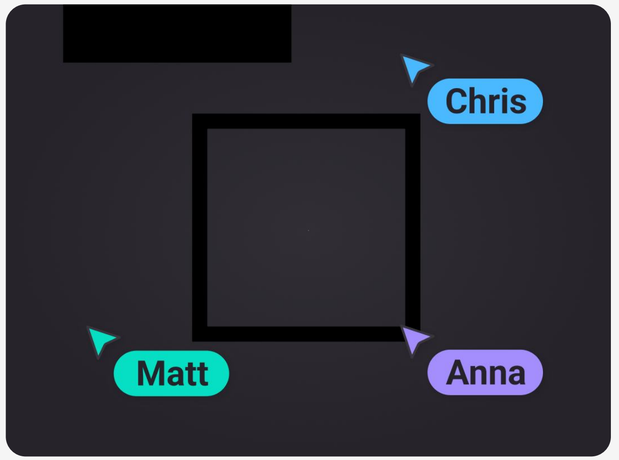


CONTEXT COMME

It embeds a cust commenting exper your page to enable collaborat

LET'S BREAK IT DOWN

Go to definition F12

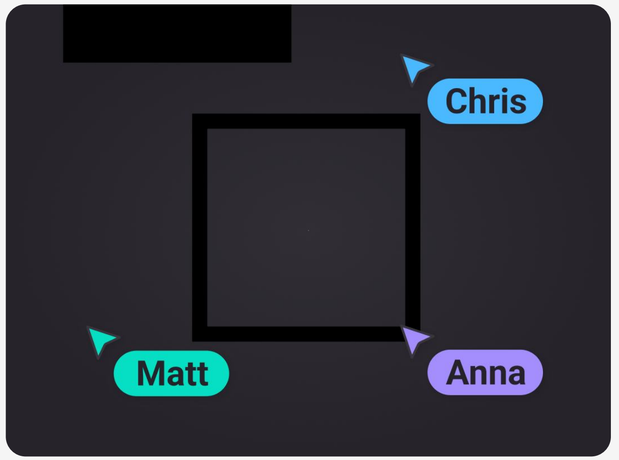


MOUSE POINTERS

It enables real-time tracking of participants' cursor movements, allowing seamless collaboration within the same room.

LET'S BREAK IT DOWN

Go to definition F12



MOUSE POINTERS

It enables real-time tracking of participants' cursor movements, allowing seamless collaboration within the same room.

LET'S BREAK IT DOWN

Go to definition F12

```
1 <MousePointers elementId="element-id" />
2 <div id="element-id" />
```

LET'S BREAK IT DOWN

Go to definition F12

```
1 <MousePointers elementId="element-id" />
2 <div id="element-id" />
```

LET'S BREAK IT DOWN

Go to definition F12

```
const room = await SuperVizRoom(DEVELOPER_KEY,
  {
    roomId: groupId,
    group: {
      id: groupId,
      name: groupName,
    },
    participant: {
      id: userId,
      name: userName,
    },
  }
);

const mousePointers = new MousePointers("element-id");

room.addComponent(mousePointers);

return room;
```

LET'S BREAK IT DOWN

```
const room = await SuperVizRoom(DEVELOPER_KEY,
  {
    roomId: groupId,
    group: {
      id: groupId,
      name: groupName,
    },
    participant: {
      id: userId,
      name: userName,
    },
  }
);

const mousePointers = new MousePointers("element-id");

room.addComponent(mousePointers);

return room;
```

NEW MOUSEPOINTER()

See when someone moves their mouse

Get information about the position of other’s participants mouse pointers.

NEW MOUSEPOINTER()

See when someone moves their mouse

Get information about the position of other's participants mouse pointers.

Update your position to everyone

Let the other participants know the position of your pointer.

NEW MOUSEPOINTER()

See when someone moves their mouse

Get information about the position of other's participants mouse pointers.

Update your position to everyone

Let the other participants know the position of your pointer.

Make it really fast and smooth

Update it in a 30fps (frames per second) so it doesn't seem laggy.

NEW MOUSEPOINTER()

See when someone moves their mouse

Get information about the position of other's participants mouse pointers.

```
1 const [mousePosition, setMousePosition] = useState({
2     x: 0,
3     y: 0
4   });
5
6 channel.subscribe("mouse.moved", onMouseMove);
7
8 function onMouseMove(message: RealtimeMessage) {
9   setMousePosition({
10    x: message.data.x,
11    y: message.data.y,
12  });
13 }
```

NEW MOUSEPOINTER()

Update your position to everyone

Let the other participants know the position of your pointer.

```
1 document.addEventListener("mousemove", onMouseMove);
2
3 function onMouseMove(event) {
4   const position = {
5     x: event.clientX,
6     y: event.clientY,
7   };
8
9   channel.publish("mouse.moved", position);
10 }
```

NEW MOUSEPOINTER()

Make it really fast and smooth

Update it in a 30fps (frames per second) so it doesn't seem laggy.

```
let isAnimationFrameRequested = false;
let lastMouseEvent: MouseEvent | null = null;

document.addEventListener("mousemove", (event) => {
  lastMouseEvent = event;
  if (!isAnimationFrameRequested) {
    isAnimationFrameRequested = true;
    requestAnimationFrame(onMouseMove);
  }
});

function onMouseMove() {
  if (lastMouseEvent) {
    const position = {
      x: lastMouseEvent.clientX,
      y: lastMouseEvent.clientY,
    };

    channel.publish("mouse.moved", position);
  }
  isAnimationFrameRequested = false;
}
```

CONST CHANNEL

What is this channel you've been using?

I know, it's ridiculous of me not to show what is this `channel` const, but it is actually implementing an event broker, which plays a crucial role in our system architecture.

```
room = await SuperVizRoom(DEVELOPER_KEY, {
  roomId: groupId,
  participant: {
    id: participant,
    name: "John " + participant,
  },
});

const realtime = new Realtime();

channel = await realtime.connect('mouse.handler');

room.addComponent(realtime);
```

CONST CHANNEL

What is this channel you've been using?

I know, it's ridiculous of me not to show what is this `channel` const, but it is actually implementing an event broker, which plays a crucial role in our system architecture.

Go to definition F12

```
export declare class Channel {
  static events = {};
  subscribe: (
    event: string,
    callback: (data: any) => void)
    => void;
  publish: (
    event: string,
    data?: unknown)
    => void;
}
```

NEW REALTIME()

How dows Pub Sub works?

Let's delve into the functionality of the Publisher and Subscriber methods to gain a comprehensive understanding of how they operate.

CONST CHANNEL

The subscribe method

Go to definition F12

```
1 function subscribe(eventName, callback) {
2   // If the event doesn't exist yet,
3   // initialize it as an empty array
4   if (!this.events[eventName]) {
5     this.events[eventName] = [];
6   }
7
8   // Push the callback function into the
9   // array of callbacks for the given event
10  this.events[eventName].push(callback);
11 }
```

CONST CHANNEL

The subscribe method

Go to definition F12

```
1 function subscribe('mouse.moved', callback) {
2   // If the event doesn't exist yet,
3   // initialize it as an empty array
4   if (!this.events['mouse.moved']) {
5     this.events['mouse.moved'] = [];
6   }
7
8   // Push the callback function into the
9   // array of callbacks for the given event
10  this.events['mouse.moved'].push(callback);
11 }
```

The publish method

Go to definition F12

```
// The publish method takes an event name and data
function publish(event, data) {

  // For each subscriber of this event,
  // call the callback function with the provided data
  this.events[event].forEach((callback) => {
      callback(data);
  });
}
```

CONST CHANNEL

The publish method

Go to definition F12

```
1 // The publish method takes an event name and data
2 function publish('mouse.moved', data) {
3
4   // For each subscriber of this event,
5   // call the callback function with the provided data
6   this.events['mouse.moved'].forEach((callback) => {
7       callback(data);
8   });
9 }
```

Well, it is this... but

Well, it is this... but

one more thing

one more thing

web sockets

web sockets

PubSub works locally,
until you add a
websocket to it

web sockets

WebSockets provide a protocol that allows for a continuous, two-way communication channel between a client and server, enabling real-time data exchange over a single, persistent connection without the need for repeated HTTP requests.

web sockets

WebSockets provide a protocol that allows for a continuous, two-way communication channel between a client and server, enabling real-time data exchange over a single, persistent connection without the need for repeated HTTP requests.

PERSISTENT CONNECTIONS

Stays open for continuous data exchange

LOW LATENCY

Ideal for real-time applications like chat, gaming, and live updates

BIDIRECTIONAL

Both client and server can send messages independently

web sockets

WebSockets provide a protocol that allows for a continuous, two-way communication channel between a client and server, enabling real-time data exchange over a single, persistent connection without the need for repeated HTTP requests.

PERSISTENT CONNECTIONS

Stays open for continuous data exchange

LOW LATENCY

Ideal for real-time applications like chat, gaming, and live updates

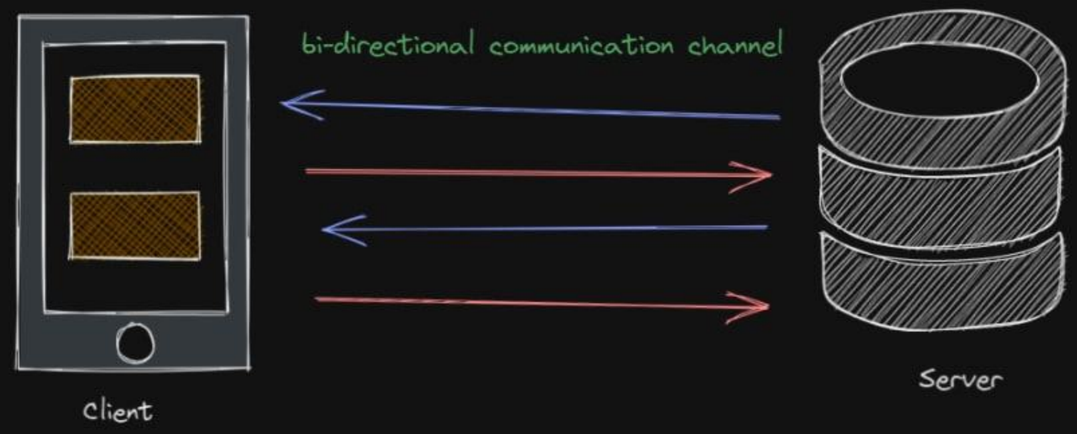
BIDIRECTIONAL

Both client and server can send messages independently

HOW IT WORKS

Client initiates a WebSocket handshake with the server.

Once established, messages can be sent in both directions without re-establishing the connection.



WHO IS USING THIS?

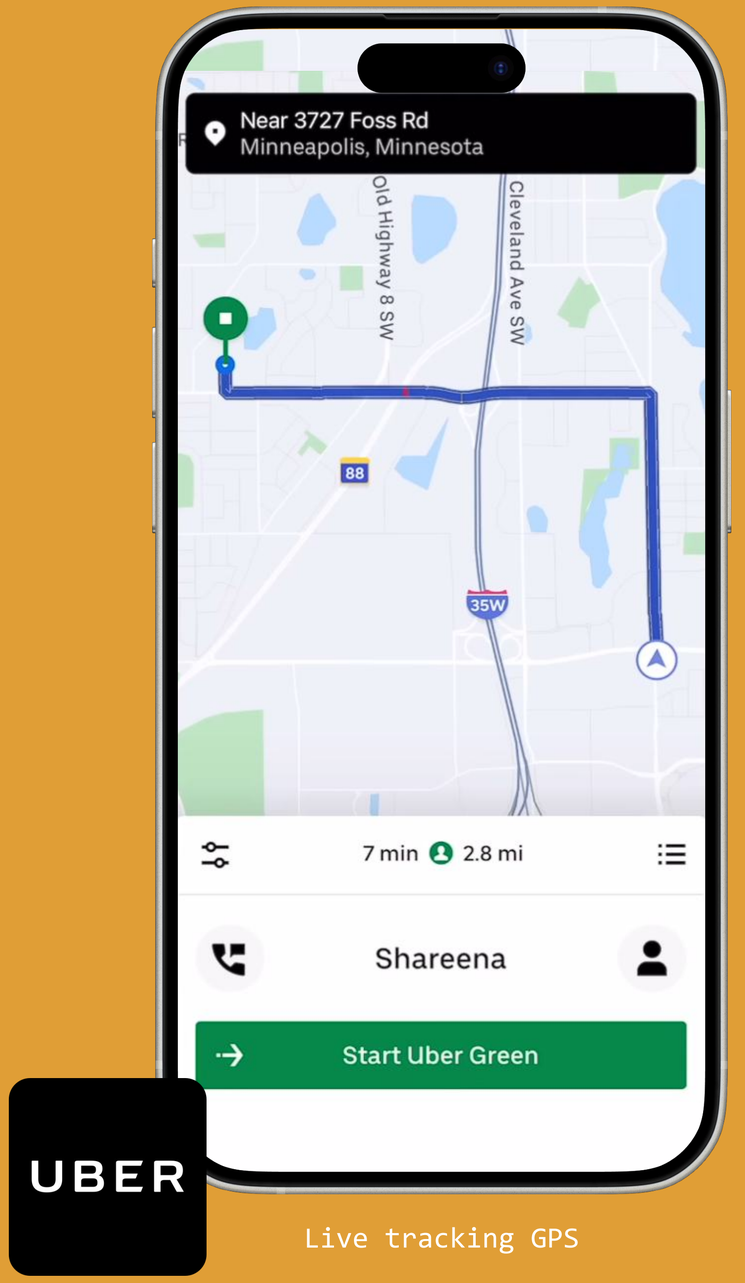
Ask ChatGPT

WHO IS USING THIS?

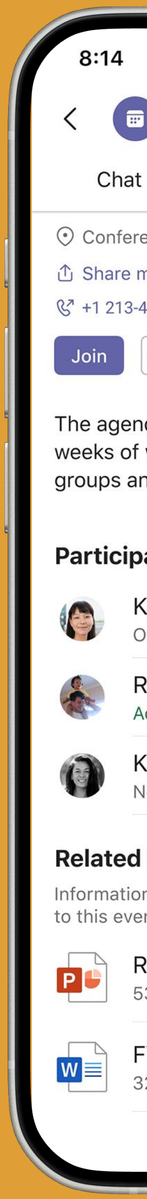
Ask ChatGPT

WHO IS USING THIS?

Ask ChatGPT

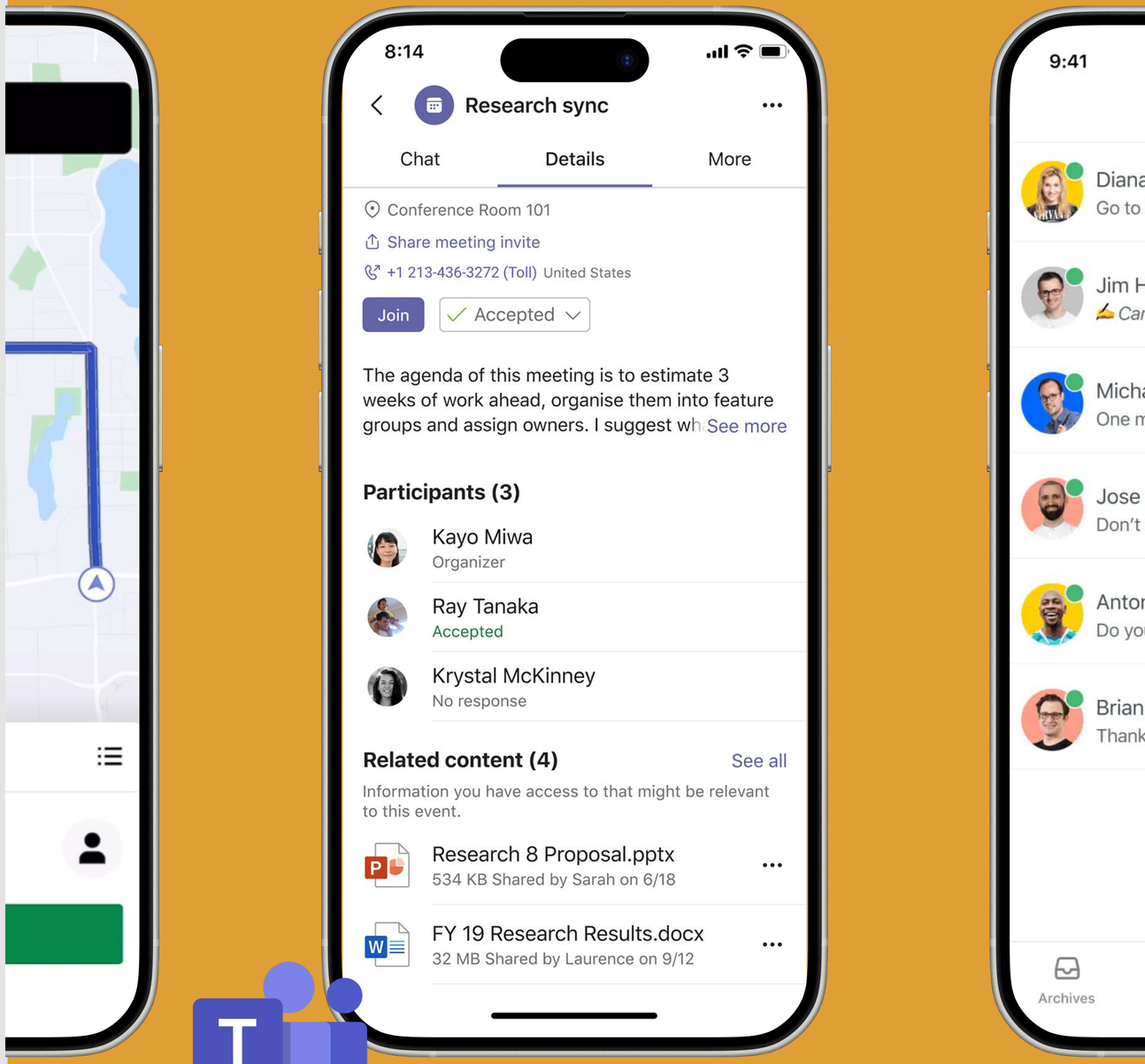


Live tracking GPS



WHO IS USING THIS?

Ask ChatGPT

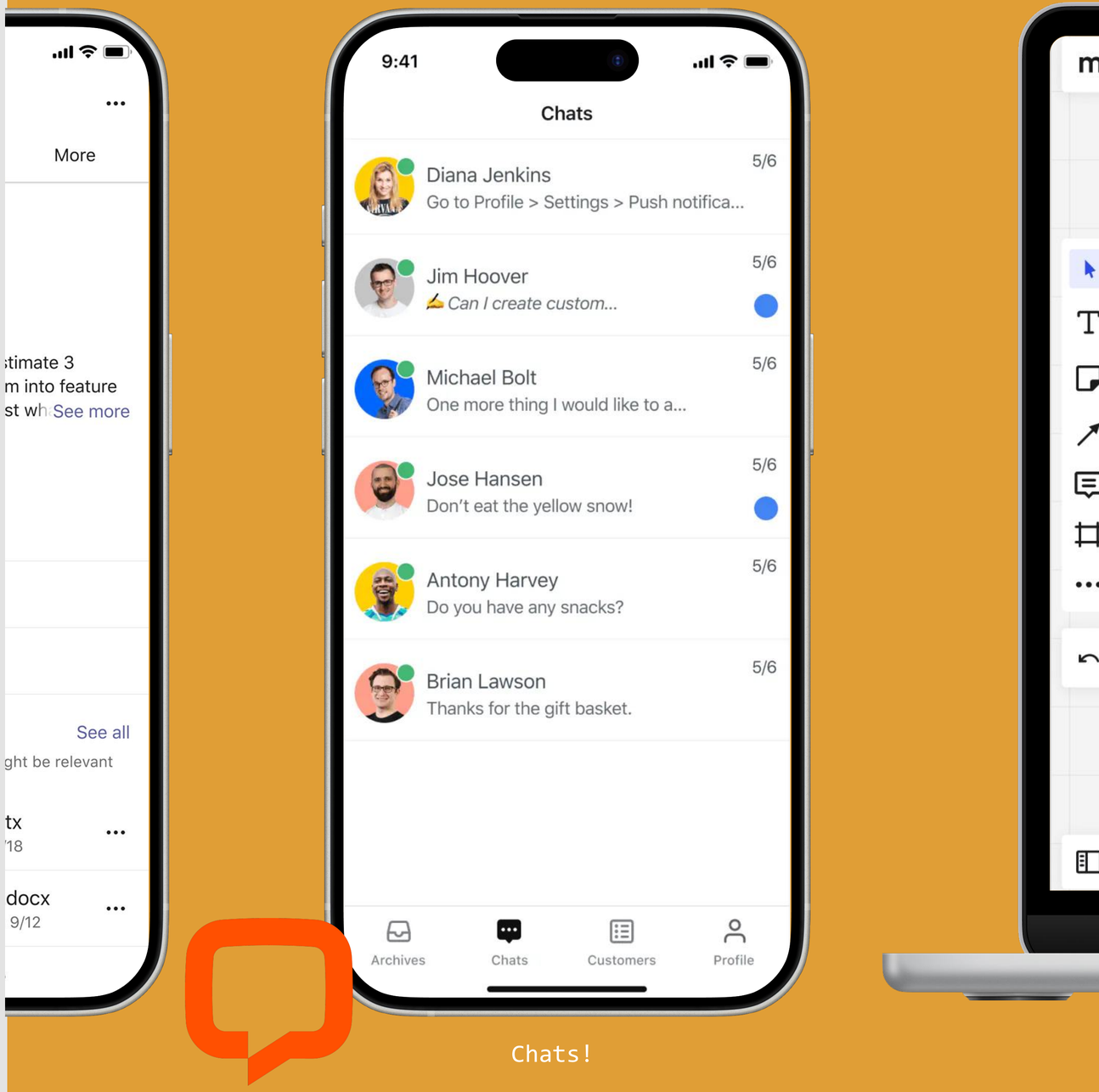


Updated info where it needs real-time

@vt_norton

WHO IS USING THIS?

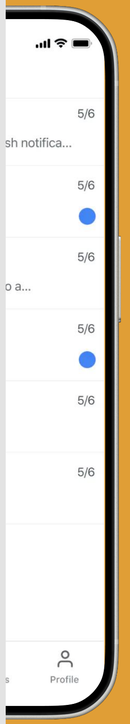
Ask ChatGPT

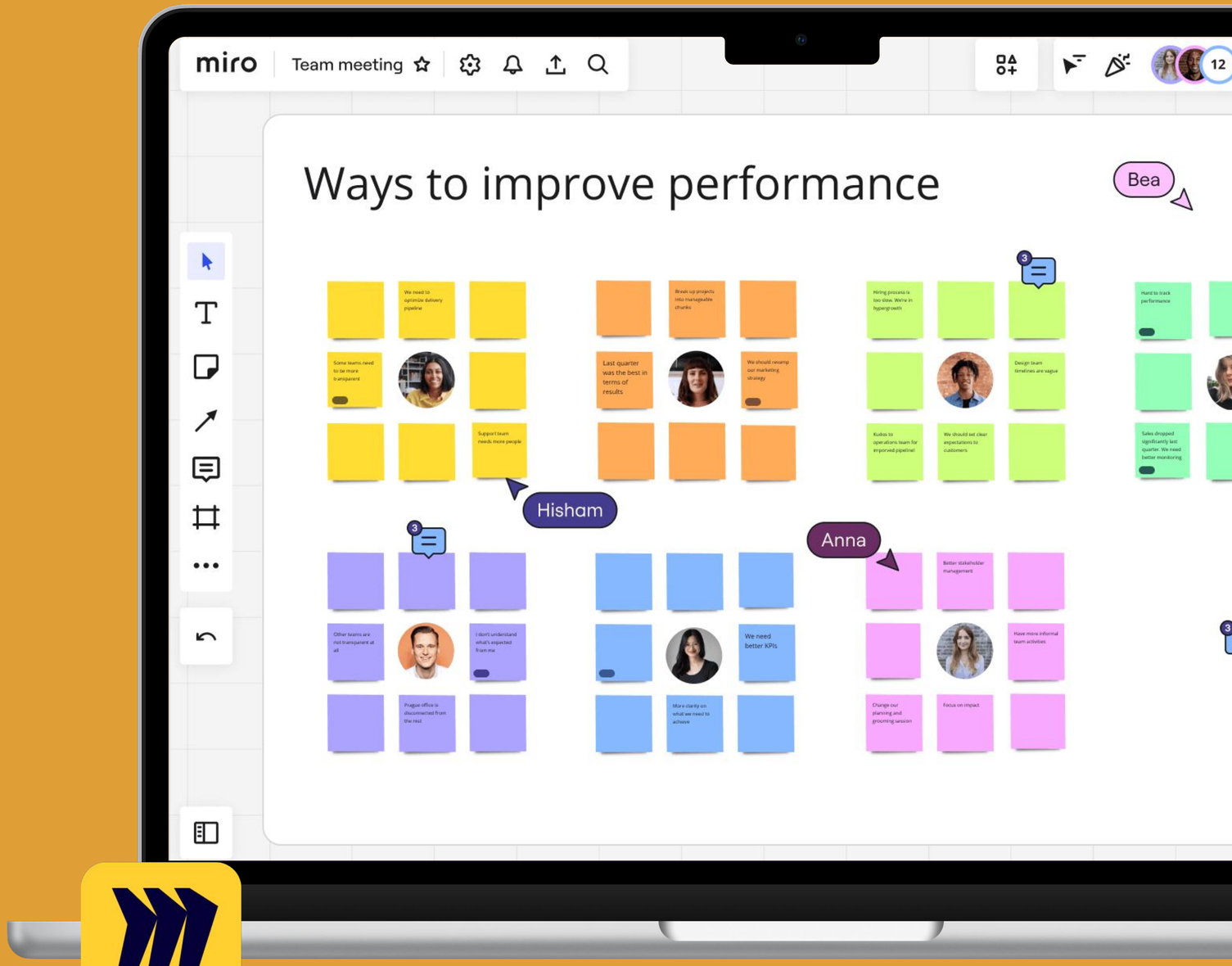


Chats!

WHO IS USING THIS?

Ask ChatGPT



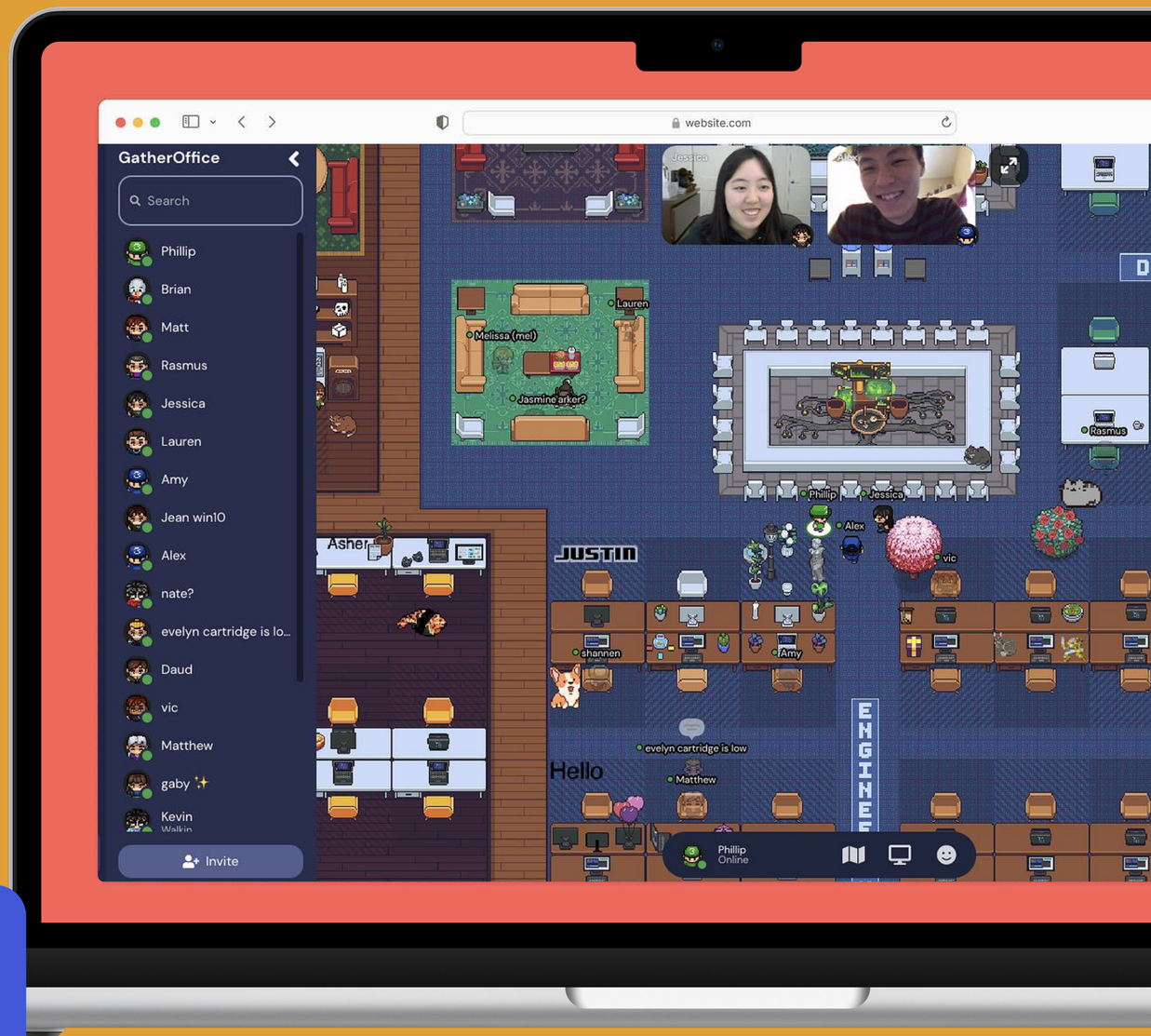


Collaborative Whiteboard

@vt_norton

WHO IS USING THIS?

Ask ChatGPT

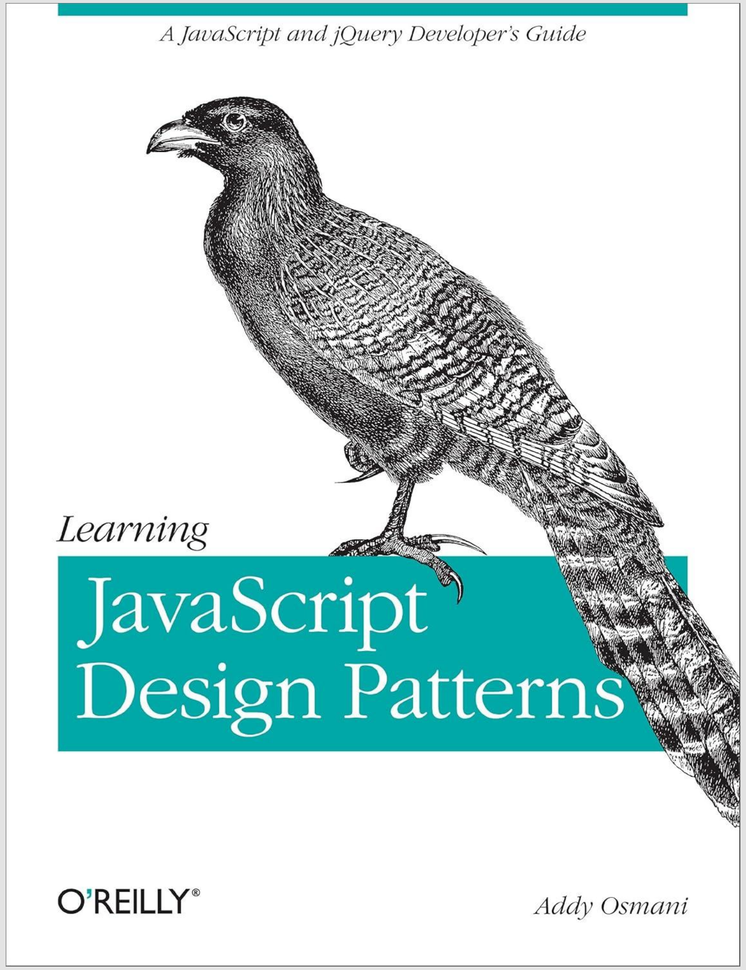


Interactive work place

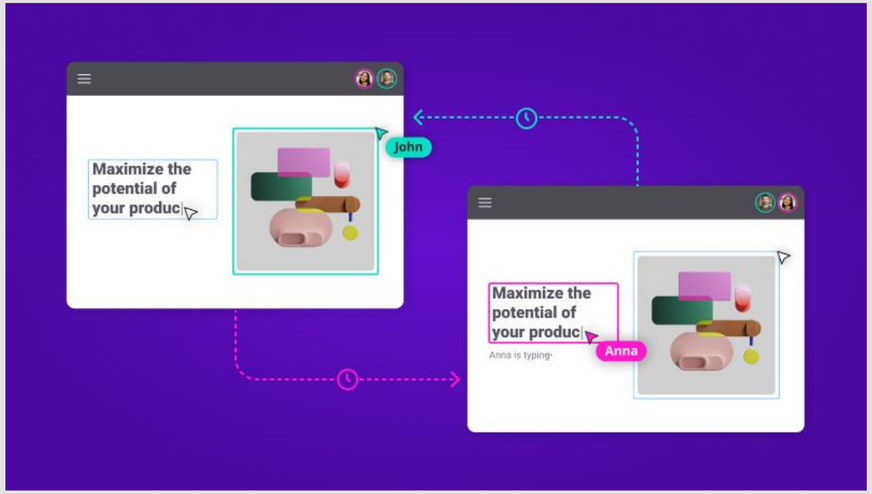
@vt_norton

RESOURCES

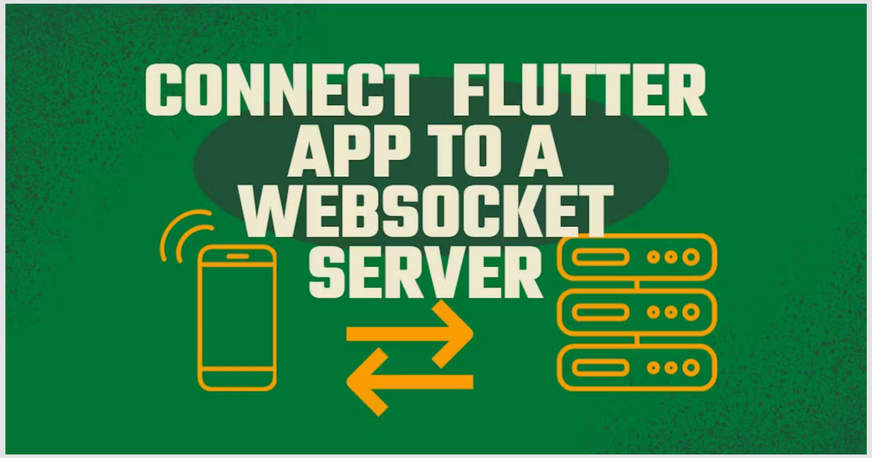
Learning JavaScript Design Patterns
Addy Osmani



How you can use WebSockets with Flutter
dev.to: Vibali Joshi
https://dev.to/vibalijoshi/how-you-can-use-websockets-with-flutter-ipn



Understanding and implementing Event-Driven Communication in Front-End Development
dev.to: Vitor Norton
https://dev.to/superviz/understanding-and-implementing-event-driven-communication-in-front-end-development-e75



THANKS ;)

LET'S BUILD SOMETHING GREAT

Vítor Norton
@vt_norton