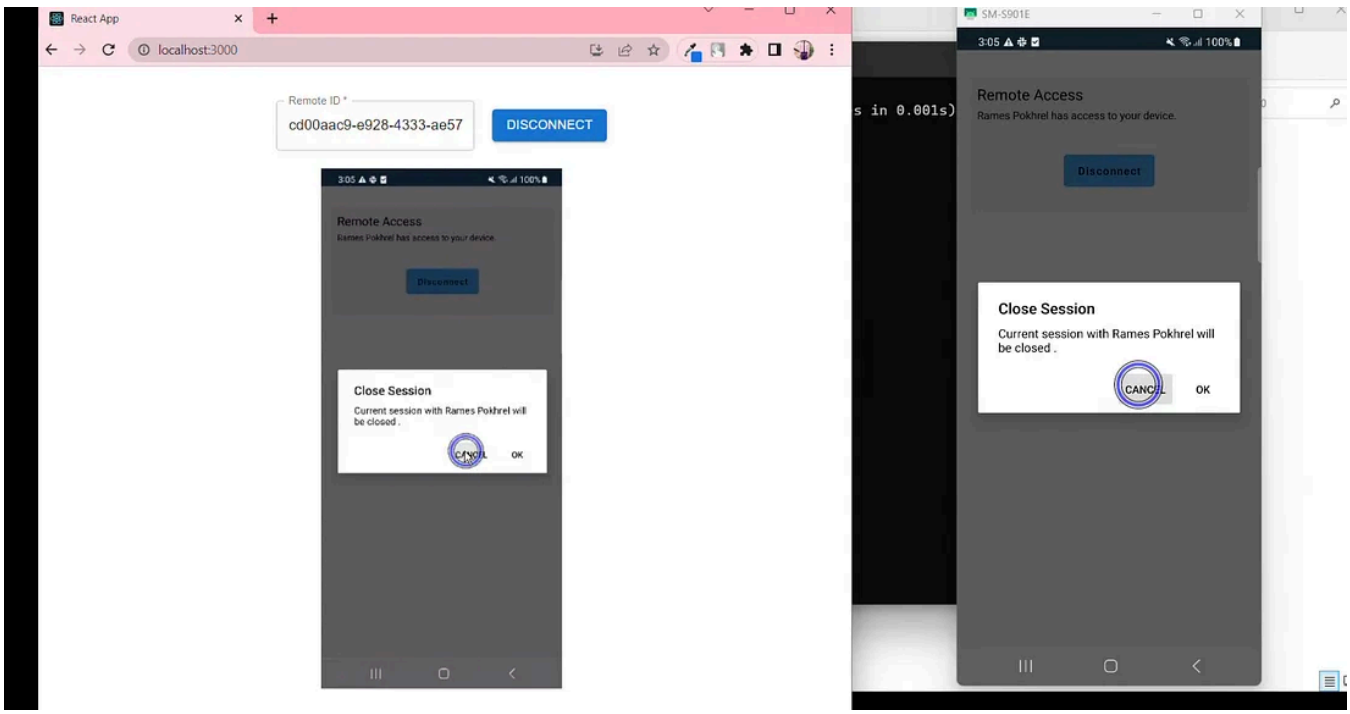
Remote Screen Sharing and Controlling Application: Implementation

Ramesh Pokhrel · Follow
6 min read · Jun 29, 2023

Listen Share

After a long break, I am back with the implementation of remote screen sharing and control on Android. This section will focus on implementation details. Please make sure to review the theoretical terms discussed in Part 1.



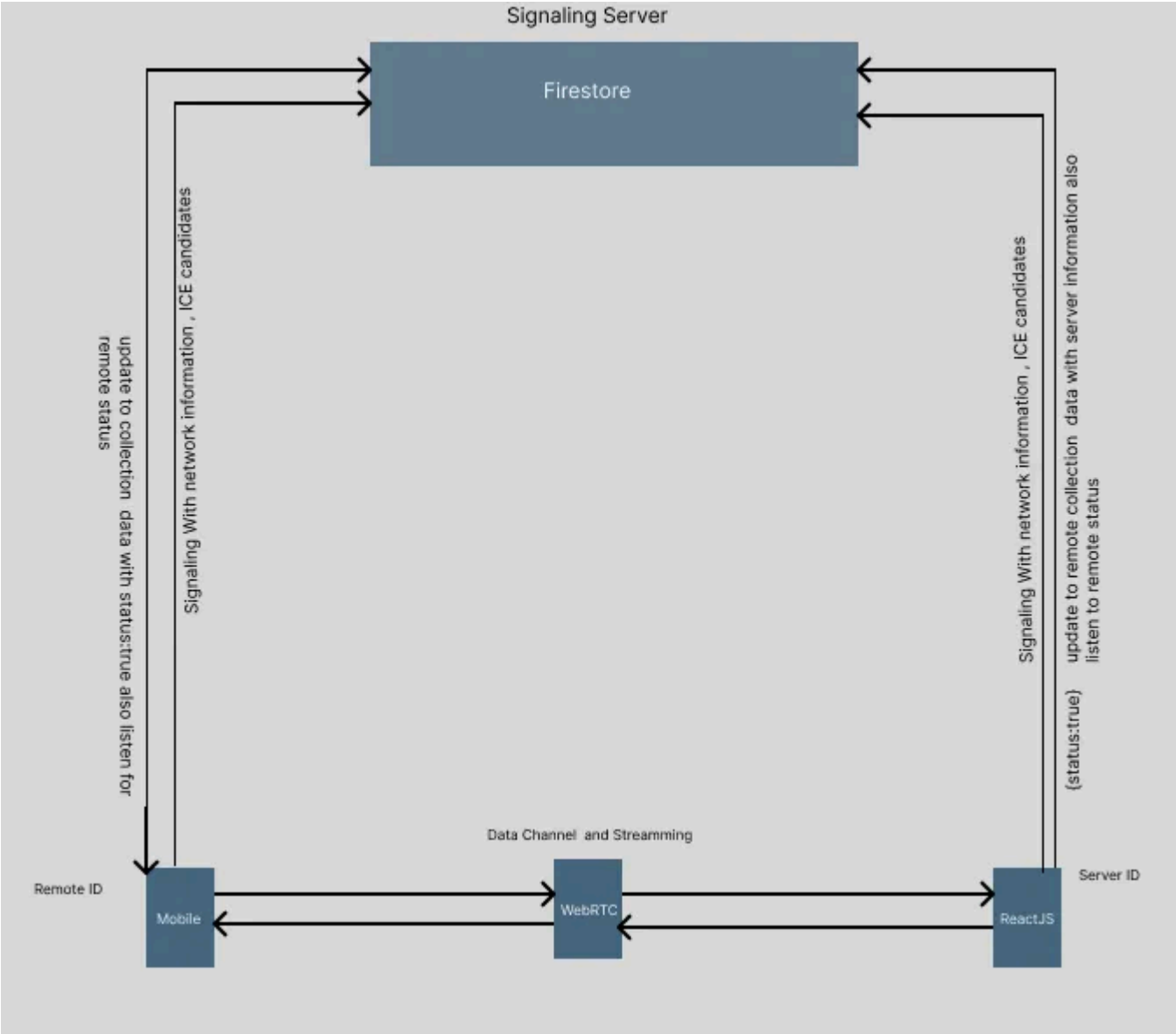
You can watch this video to know what this blog is about. 😁

For this project, I am utilizing ReactJS and Android as the two clients that will communicate using WebRTC. We will use Google Firestore as our signaling server. Since we need to handle connections through various proxy networks, we require

ICE servers. In this case, I am utilizing the ICE servers provided by Google for this purpose.

```
stun:stun1.l.google.com:19302
stun:stun2.l.google.com:19302
stun:stun.l.google.com:19302
```

How WEBRTC works in this project?



Each client (Android) has a unique ID assigned to it, and it continuously listens to the same document in Firestore to check if any server wants to establish a connection with it.

ReactJS

We have *RTCPeerConnection* and need to initialize with ice servers.

```
const servers = {
    iceServers: [
        {
            urls: [
                "stun:stun1.l.google.com:19302",
                "stun:stun2.l.google.com:19302",
            ],
        },
    ],
    iceCandidatePoolSize: 10,
};
//initialize RTC with ice servers
const pc = new RTCPeerConnection(servers);
```

We do not send any stream from ReactJS to Android. As we do not have a local stream, the onicecandidate event has not been invoked. so, we utilize a data channel to transmit injected coordinates, which also triggers the onicecandidate method.

```
const dataChannel = pc.createDataChannel("channel");
//we are receiving remote video only
pc.addTransceiver('video');
```

Status update

The server will set its status as “ready” to inform the client using {status: true}.

```
 myDoc.set({ "status": true })
```

Request to the callee...

Please update the remote collection(Firebase ref for remote Android) to inform the server that it wishes to connect with the remote Android device.

```
 remoteControl.doc(remoteId).set({
            caller: {
```

```
            callerId: MY_REMOTE_ID, callerName: "Rames Pokhrel"
        }
    })
```

Awaiting the callee...

Ready to listen to the status of the Android device (remote). The remote device also provides information about its screen resolution. We will calculate the screen resolution scale factor.

```
remoteControl.doc(remoteId).onSnapshot((snapshot) => {
    const data = snapshot.data();
    if (data?.status) {
        //update video width height
        var dWidth = data?.dWidth;//1080
        var dHeight = data?.dHeight;//2260

        var scaleWidth = dWidth / 270
        var scaleHeight = dHeight / 584

        setScaleXFactor(scaleWidth)
        setScaleYFactor(scaleHeight)

        setVideoWidth(270)
        setVideoHeight(584)

        setIceAndOfferCandidates()

    }
});
```

After Remote Device is ready, Send an Offer

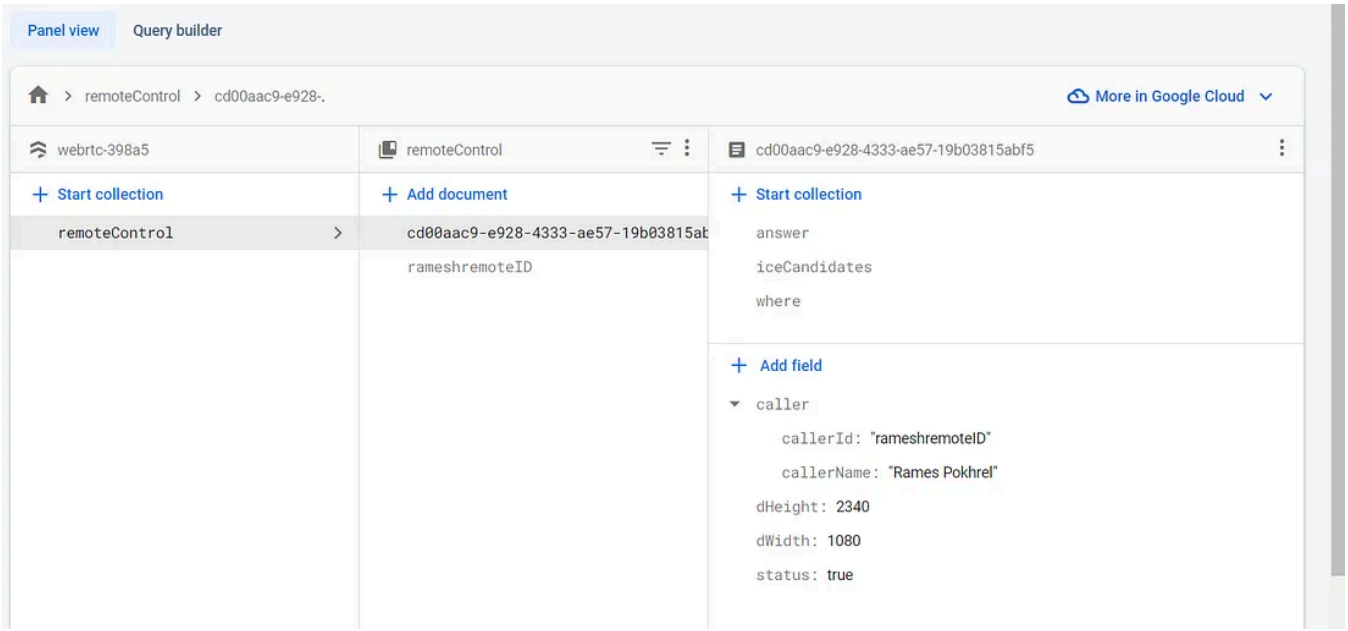
```
const offerDescription = await pc.createOffer();
await pc.setLocalDescription(offerDescription);
const offer = {
    sdp: offerDescription.sdp,
    type: offerDescription.type,
};

await myOffer.add(offer);
```

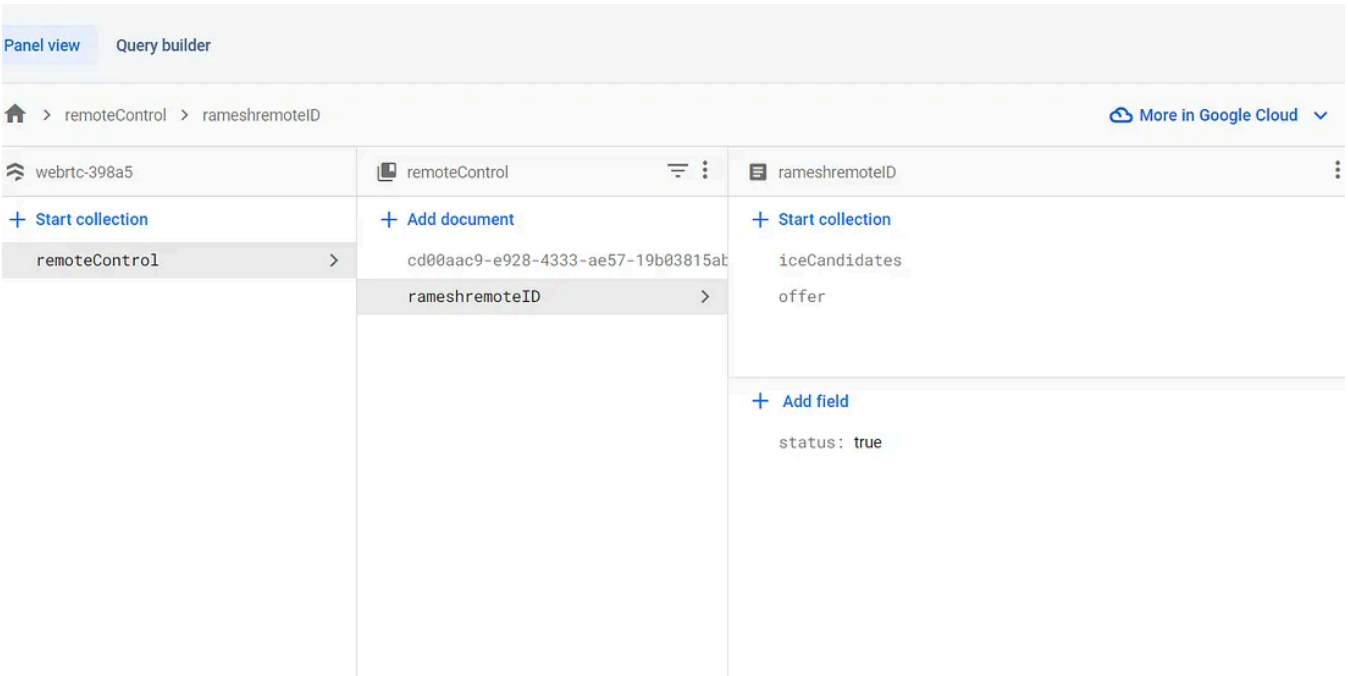
Listen for remote answers and Remote ICE candidates

```
    calleeAnswer.onSnapshot((snapshot) => {
              snapshot.docChanges().forEach((change) => {
                  if (change.type === "added") {
                      try {
                          var a = change.doc.data();
                          const candidate = new RTCSessionDescription(a);
                          pc.setRemoteDescription(candidate)
                              .then(() => {
                                  calleeIceCandidates.onSnapshot((snapshot) => {
                                      snapshot.docChanges().forEach((change) => {
                                          if (change.type === "added") {
                                              var a = change.doc.data()
                                              const candidate = new RTCIceCandida
                                                  a
                                              );
                                              pc.addIceCandidate(candidate);
                                          }
                                      });
                                  });
                              })
                              .catch((error) => {
                                  console.error("Error setting remote description
                              });
                      } catch (e) {
                          console.log("e", e.error)
                      }
                  }
              });
          });
```

Firestore Handshaking

Firestore is used as a signal server.





Event Dispatch

We will use the data channel to send a dispatcher event. I am using a div to show the remote Android screen view. The div has an event listener to send a dispatch to the remote.

```
    <div
        onMouseDown={handleMouseDown}
        onMouseMove={handleMouseMove}
        onMouseOut={handleMouseOut}
        style={{ width: videoWidth, height: videoHeight }}
        onMouseUp={handleMouseUp}>
          <video
            ref={localRef}
```

```
            autoPlay
            playsInline
        />
    </div>
```

dispatch key events to callee in the following format

x:y:ACTION_DOWN, *x:y:ACTION_UP* and *x:y:ACTION_MOVE*

```
dataChannel.send(`${event.nativeEvent.offsetX * scaleX}:${event.nativeEvent.of
```

Android Client

I am developing a new remote control application for Android, designed to function on any Android device with API level 26 or higher.

To handle WebRTC and event dispatch, I am creating a new Android module library. For remote event dispatch functionality, our app requires accessibility permission to be granted.

In order to dispatch key events on Samsung devices, we are utilizing the Samsung Knox SDK. It's important to note that Event dispatch in Samsung Knox support is available only at Knox API level greater than 100.

Implement Remote Control Library

I made an android library to handle Screen share and remote control. You can find an Android sample project here.

Add maven repo for Webrtc. Webrtc is currently taking from jCenter.

I am uploading my Android Remote library to bitbucket. You can access it from following credentials.

```
allprojects {
    ...
    repositories {
```

```
        ...
        jcenter()
        maven {
            credentials {
                username "kanxoramesh"
                password "ATBB578FL2Hxm5TFdgexbfvu3zwJ9CF06277"
            }
            authentication {
                basic(BasicAuthentication)
            }
            url "https://api.bitbucket.org/2.0/repositories/kanxoramesh/control
        }
      ...
    }
    ...
}
```

add gradle dependencies script in app level *build.gradle* file.

```
implementation 'com.remote.remote:remotedispatch:1.0.3'
```

Now you are ready to run your application. 🤟🤟

If you are still interested in how the library works, The rest of the following task is high-level implementation inside the Remote library.

Find the Knox SDK from Knox's official site. Add gradle dependency for Webrtc and Knox.

```
compileOnly files('libs/knoxsdk.jar')
implementation 'org.webrtc:google-webrtc:1.0.32006'
```

Unique remote ID

Generate random UUID for remote ID. You can use different techniques

```
UUID.randomUUID().toString()
OR
```

```
android.os.Build.getSerial()
OR
android.os.Build.SERIAL
```

Implement WEBRTC

PeerConnectionFactory will be used to provide configurations for webrtc. Listen to different WEBRTC events. Create a Data channel to listen to dispatch key events.

Event parse

For non-Samsung devices and Samsung with Knox Api below 100, create a Dispatch key from (x,y) coordinates and the key events

```
dispatch = DispatchKey(xCoordinate, yCoordinate, action)
```

actions are from MotionEven android, they might be **MotionEvent.***ACTION_DOWN,* **MotionEvent.***ACTION_UP,* **MotionEvent.***ACTION_MOVE etc.*

Motion Event

Motion events describe movements in terms of an action code and a set of axis values.

```
var motionEvent = MotionEvent.obtain(
    SystemClock.uptimeMillis(), SystemClock.uptimeMillis(),
    dispatch.action, dispatch.x, dispatch.y, 0
)
int actionMasked = motionEvent.getActionMasked();
int actionIndex = motionEvent.getActionIndex();
int pointerId = motionEvent.getPointerId(actionIndex);
float x = motionEvent.getX(actionIndex);
float y = motionEvent.getY(actionIndex);
```

Inject Service

For Samsung devices, we will be injecting events from **EnterpriseDeviceManager,** Please check Knox API level first.

```
fun isSupportedKnox():Boolean{
    int i;
    try {
        i = EnterpriseDeviceManager.getAPILevel();
    } catch (Throwable unused) {
        i = 0;
    }
  return knoxApilevel > 100
}
```

If Knox API is supported by Samsung,

```
val enterpriseDeviceManager =
    EnterpriseDeviceManager.getInstance(getApplicationContext())
motionEvent = MotionEvent.obtain(
     SystemClock.uptimeMillis(), SystemClock.uptimeMillis(),
     action, x, y, 0
     )
val remoteInjection = enterpriseDeviceManager.remoteInjection
remoteInjection.injectPointerEvent(motionEvent, true);
```

Add Accessibility Service

```
<service
    android:name=".RemoteAccessibilityService"
    android:permission="android.permission.BIND_ACCESSIBILITY_SERVICE"
    android:enabled="true" android:exported="true">
    <intent-filter>
        <action android:name="android.accessibilityservice.AccessibilityService
    </intent-filter>
  <meta-data
          android:name="android.accessibilityservice"
          android:resource="@xml/service" />
 </service>
```

To utilize the Accessibility service for devices that are non-Samsung or have a Knox level below 100, you can pass the extracted values to the Accessibility service and use the `**dispatchGesture**` method to dispatch events.

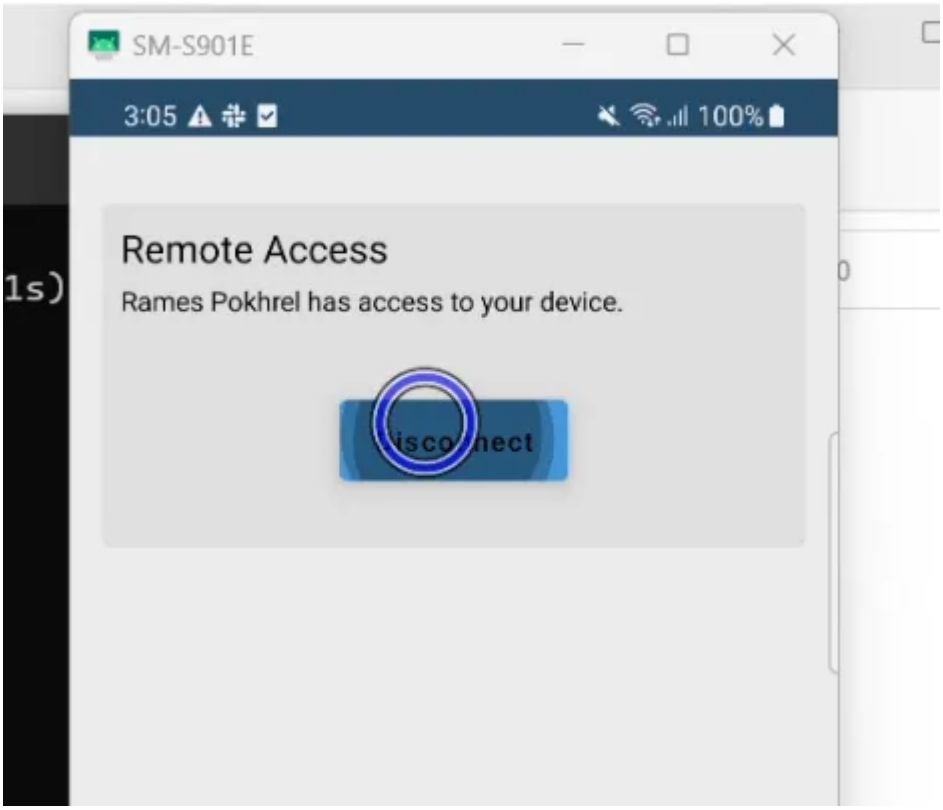
```
RemoteAccessibilityService a = RemoteAccessibilityService.Companion.getSharedIr
GestureDescription.Builder builder = new GestureDescription.Builder();
var path = Path(pointerId, x, y)
var strokeDescription = new GestureDescription.StrokeDescription(path, 0, 10, t
builder.addStroke(strokeDescription);
a.dispatchGesture(builder.build(), (AccessibilityService.GestureResultCallback)
```

Android Animator

To display an animation while clicking on the Android screen, you can utilize the `TimeAnimator` class provided by Android. Additionally, can create a custom `FrameLayout` to overlay the screen and show the pointer.

When you click on the screen, the custom overlay will display a growing circle animation at the clicked position. Adjust the animation properties (e.g., animation speed, maximum radius) according to your requirements.

```
WindowManager.LayoutParams layoutParams2 = new WindowManager.LayoutParams(-1, -
layoutParams2.gravity = 51;
layoutParams2.setTitle("Pointer");
```



TURN Servers

I am using Google ICE servers. They work if both clients are in the same network. However, for clients from different networks, we need to set up our own TURN servers. I was able to set up a TURN server using coTURN.

Please use this tutorial to make your own turn servers.

The final video of this project is linked on youtube.

WEB client sample hosted in GitHub:

github.com

GitHub - kanxoramesh/webrtc-android-client

Contribute to kanxoramesh/webrtc-android-client development by creating an account on GitHub.

github.com

Thank You 😊

Android | Remote Control | Touch Screen Control | WebRTC | Accessibility

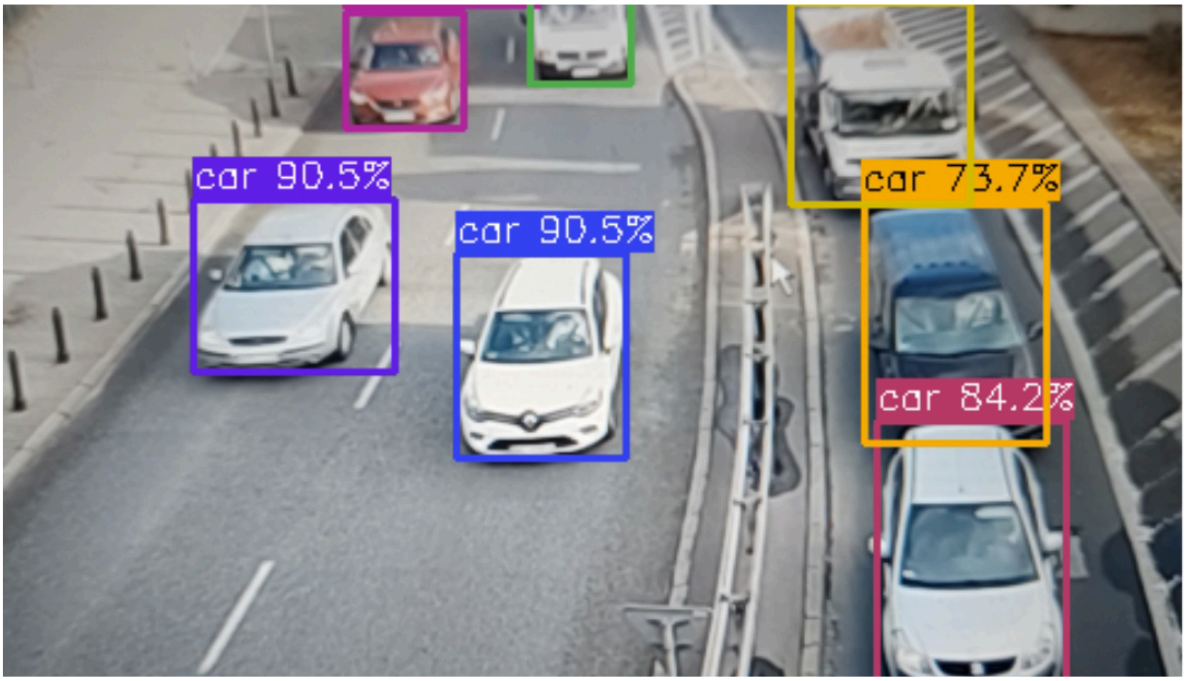
Follow

Written by Ramesh Pokhrel

29 Followers

Software Engineer (https://rameshpokhrel.web.app)

More from Ramesh Pokhrel



Ramesh Pokhrel

Object Detection and Tracking in Android (Native C++)- Part 1

This will be a series of discussions on detection and tracking in Android. I will cover various topics and points that I have encountered...

Jun 9, 2023 48

Ramesh Pokhrel

Bitbucket as Maven Repository

There are lots of maven repositories that gave us to upload and retrieves our private artifacts. Some of them are free, and some with...

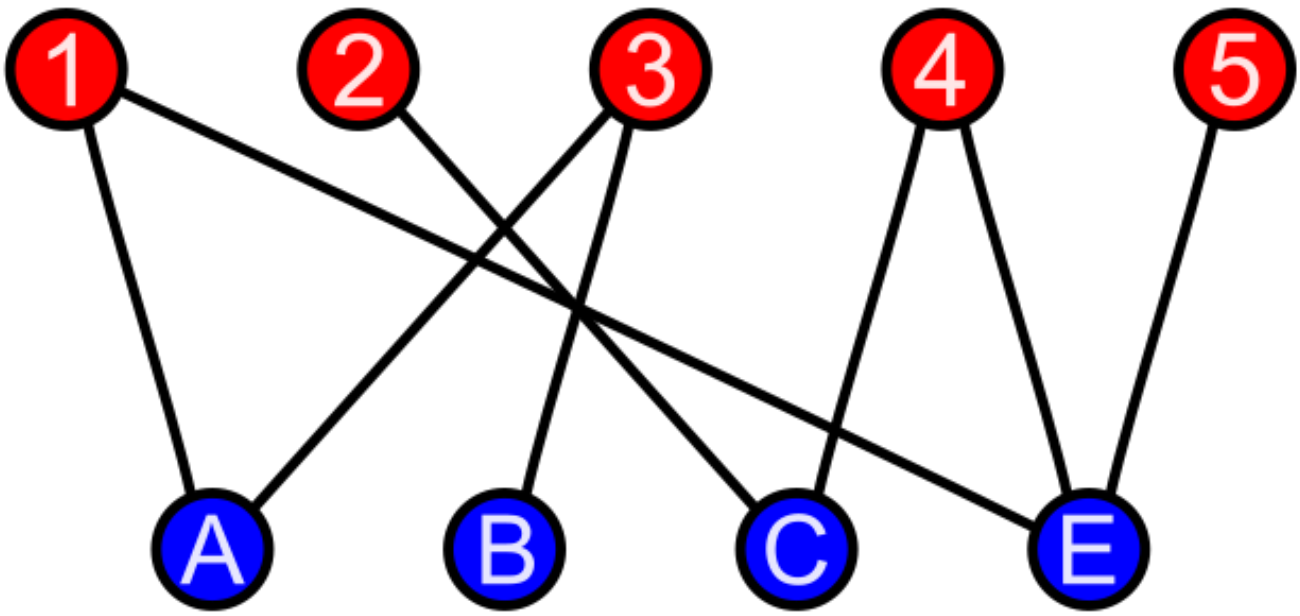
Feb 22, 2023 13 2

Ramesh Pokhrel

Remote Screen Sharing Application : Technical Issues and How I Overcame Them. Part-1

As software engineers, we face numerous difficulties, some of which can be easily solved, while others may require considerable effort. We...

Feb 23, 2023 1 1



Ramesh Pokhrel

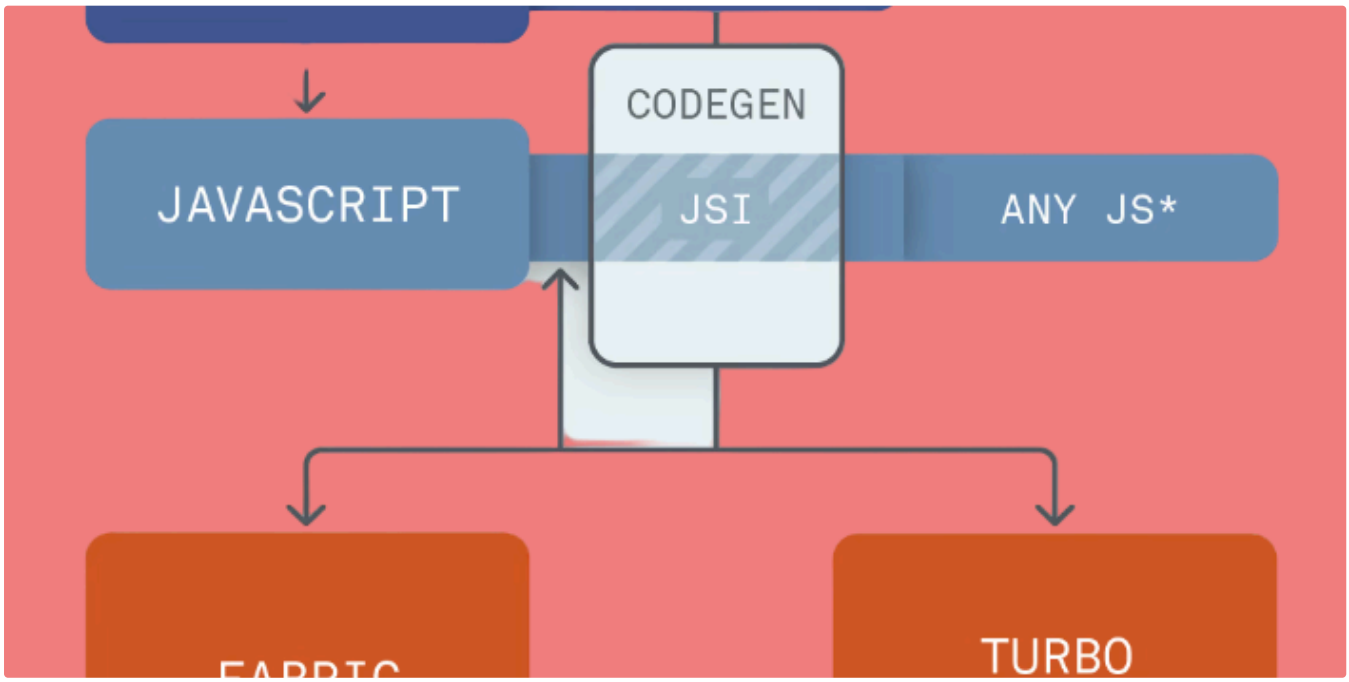
Tracking with Deepsort

Let’s cover some common terms I used in Deep Learning.

Aug 1, 2023 7

See all from Ramesh Pokhrel

Recommended from Medium



Aman Kumar in Stackademic

Enabling React Native’s New Architecture: A Practical Guide to Using Fabric and TurboModules

The React Native New Architecture introduces Fabric and TurboModules, which are designed to improve the performance, flexibility, and...

6d ago 1

Amazon.com Seattle, WA
Software Development Engineer Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by $25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay

Alexander Nguyen in Level Up Coding

The resume that got a software engineer a $300,000 job at Google.

1-page. Well-formatted.

Jun 1 23K 462

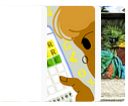
Lists

Interesting Design Topics

257 stories · 819 saves

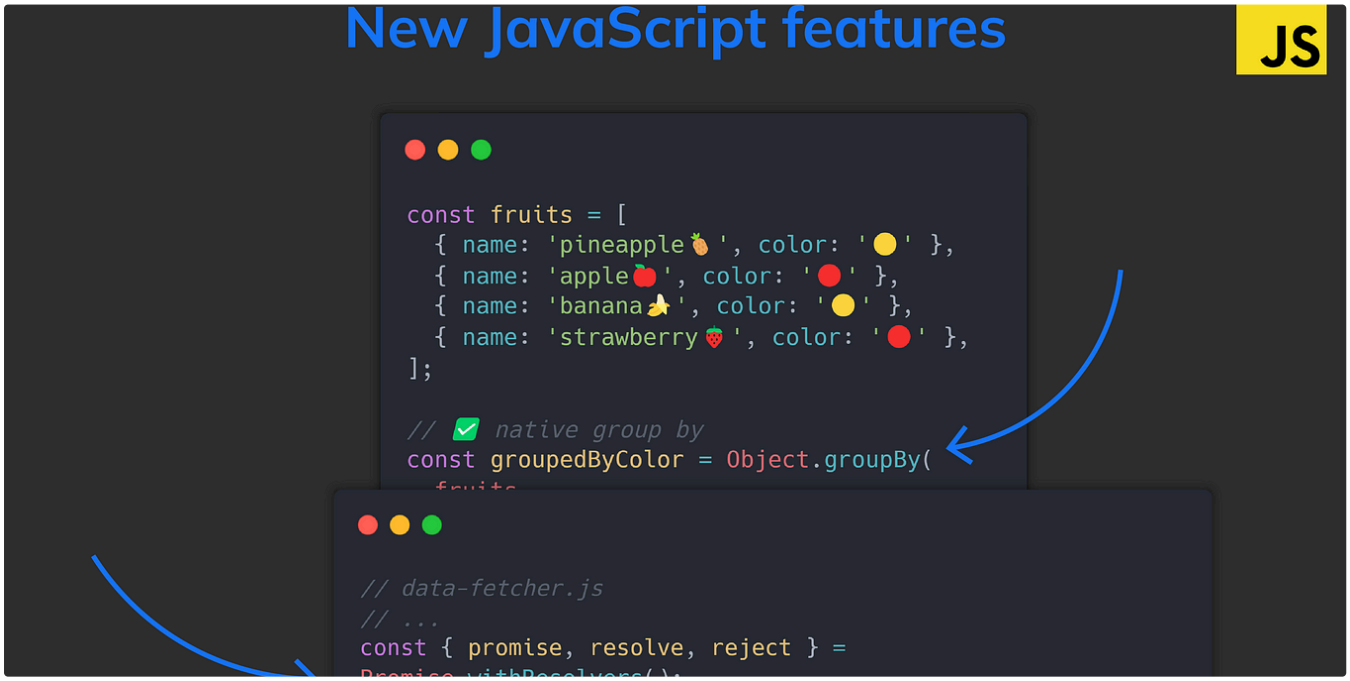
Icon Design

36 stories · 430 saves



Staff Picks

745 stories · 1352 saves

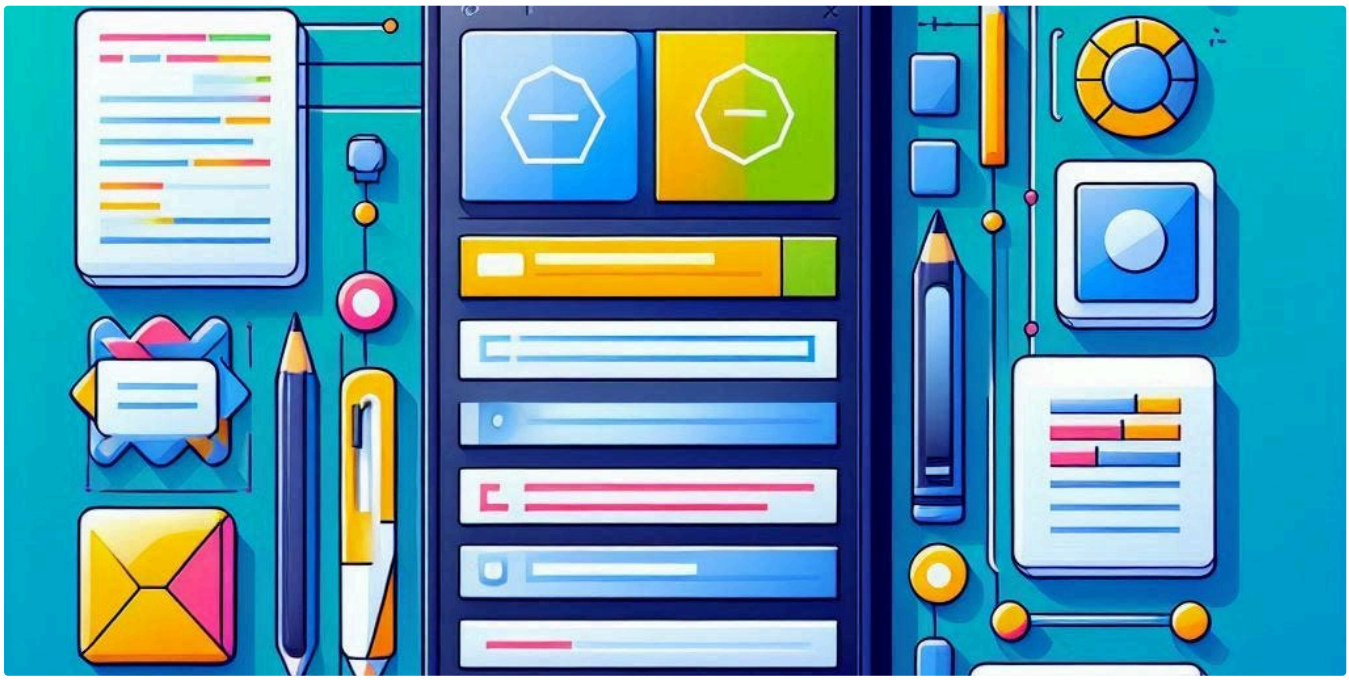


Tari Ibaba in Coding Beauty

5 amazing new JavaScript features in ES15 (2024)

5 juicy ES15 features with new functionality for cleaner and shorter JavaScript code in 2024.

Jun 2 2.9K 20

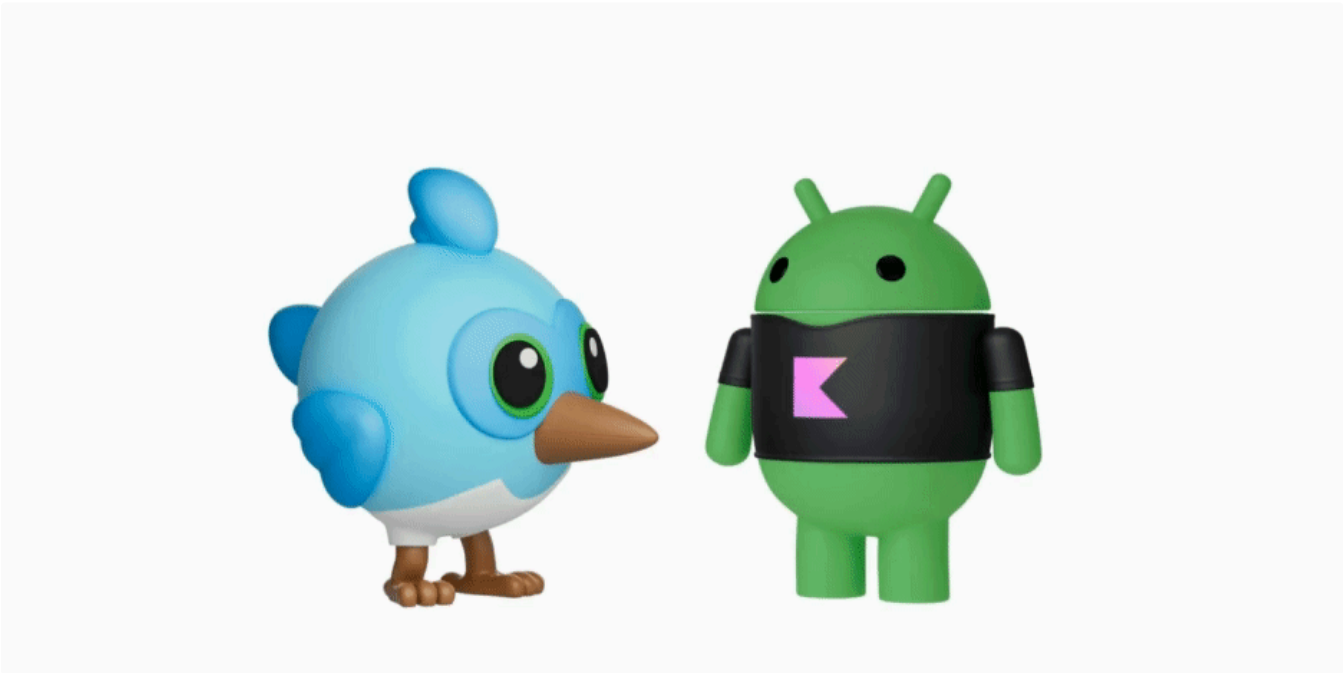


Ali Mansour

Getting Started with Jetpack Compose: A Beginner's Guide

Jetpack Compose has transformed Android UI development by providing a declarative framework that makes it easier to create modern...

Sep 22



Thomas Middel

Is Kotlin Multiplatform killing Flutter?

The current options of cross-platform solutions are endless, and honestly: there's no right or wrong. KMP and Compose look like a serious...

May 24 820 15

Prakhar

WebRTC Basics: A Beginner’s Friendly Guide to Real-Time Communication.

What is WebRTC?

Sep 3 18 1

See more recommendations