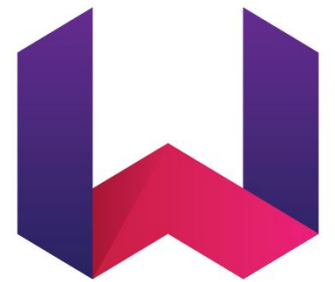


Push-notifications for

PWA



**WEBPURPLE**



Pavel Uvarov,  
Software Engineer at Epam Ryazan

pavel\_uvarov@epam.com



spiderpoul



poul\_uvarov

# Agenda

- What is PWA
- How Web Push works
- PWA example (Node.js, React)
- Restrictions and future of PWA
- QA



# What is PWA?

**Progressive Web Apps** are web apps that use emerging web browser APIs and features along with traditional progressive enhancement strategy to bring a native app-like user experience to cross-platform web applications.



# Was Steve Jobs the first to introduce the concept of PWAs?



Web 2.0 + AJAX

Integrate with iPhone services

Instant distribution

Easy to update

Secure - same as transactions  
with Amazon or a bank

Sandboxed on iPhone

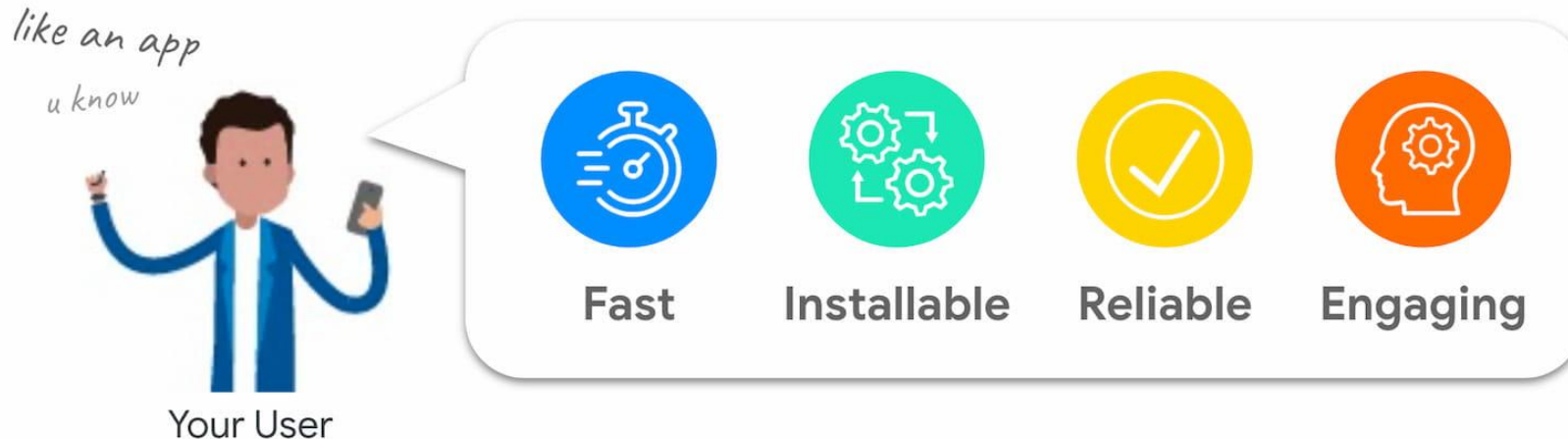
# PWAs solve customer needs

**Installable** - installed PWA run in a standalone window instead of a browser tab. They're launchable from on the user's home screen.

**Reliable** - PWA can work offline.

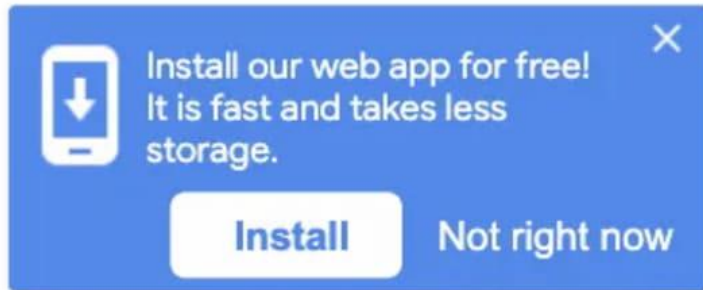
**Engaging** - Having an icon on the home screen makes it easy to get into the app and push notifications can help alert the user of important information that requires their attention.

**Responsive** - should be able to adapt to different screen sizes and orientations

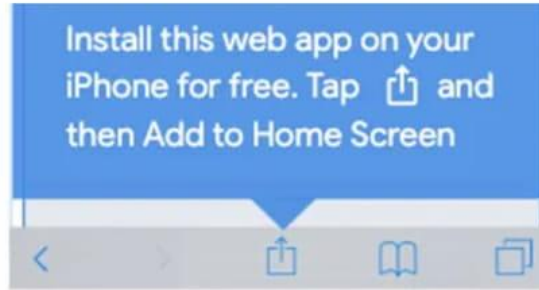


# One app, every platform

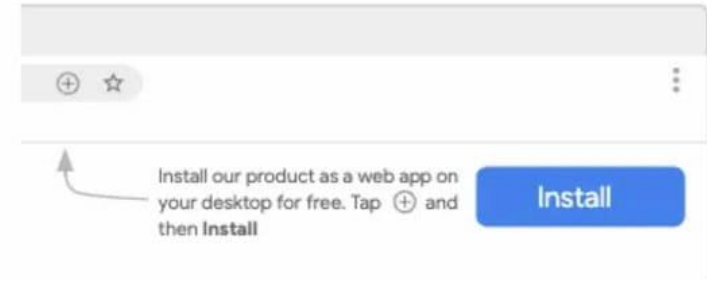
Install your PWA everywhere



Mobile



Mobile (iOS)



Desktop



# One app, every platform

- PWA are more affordable to develop and integrate than native mobile apps
- PWAs do not need to be updated



# Who is already using PWA?

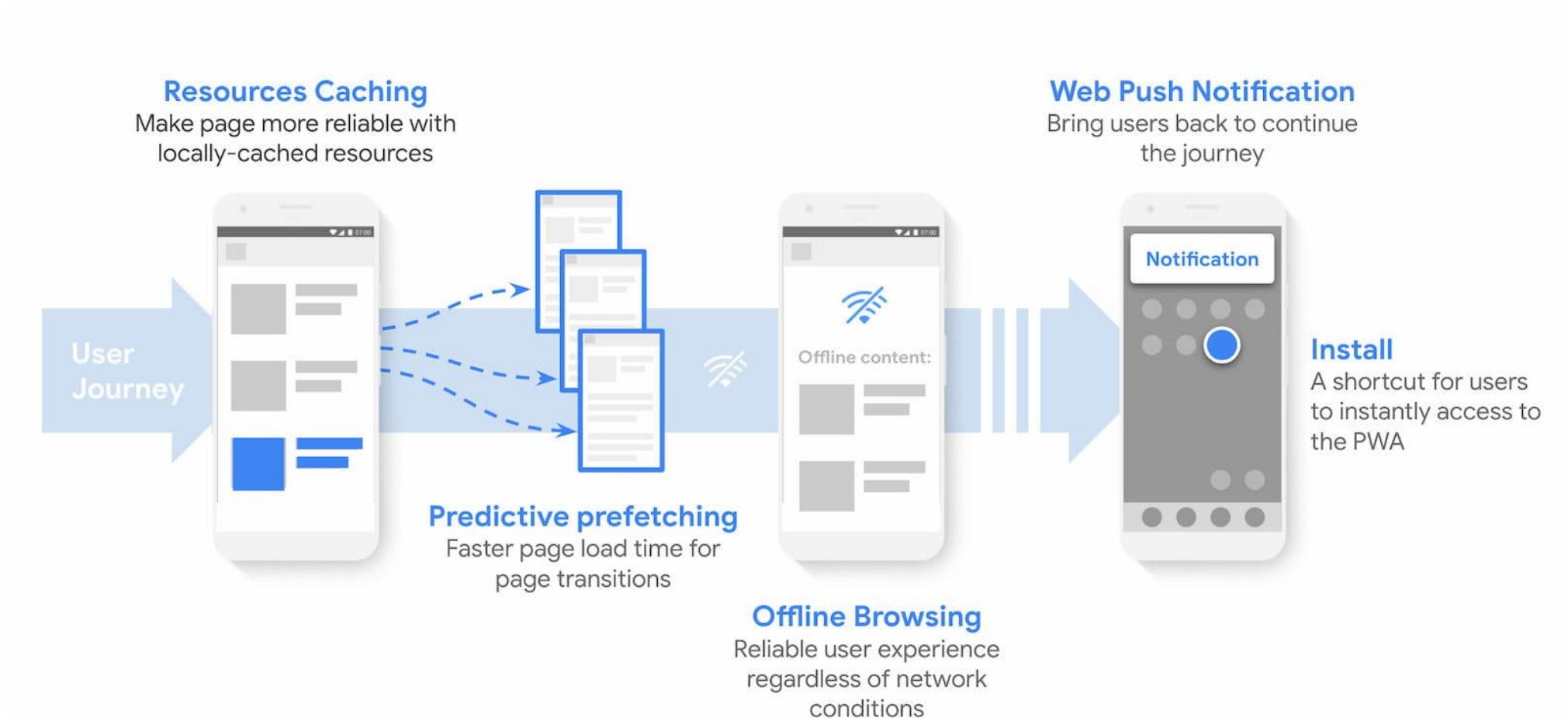
Twitter saw a 65% increase in pages per session, 75% more Tweets, and a 20% decrease in bounce rate, all while reducing the size of their app by over 97%.

Nikkei saw 2.3 times more organic traffic, 58% more subscriptions, and 49% more daily active users.

Hulu replaced their platform-specific desktop experience with a Progressive Web App and saw a 27% increase in return visits.



# PWAs leverage modern web capabilities



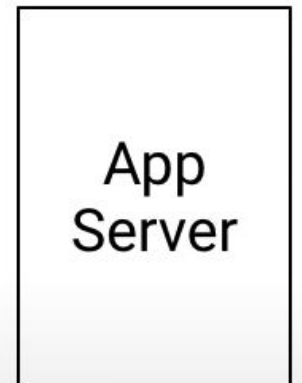
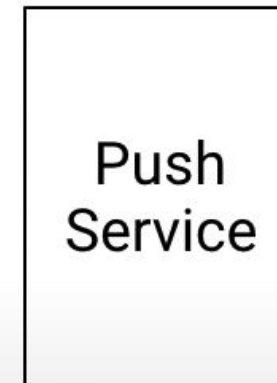
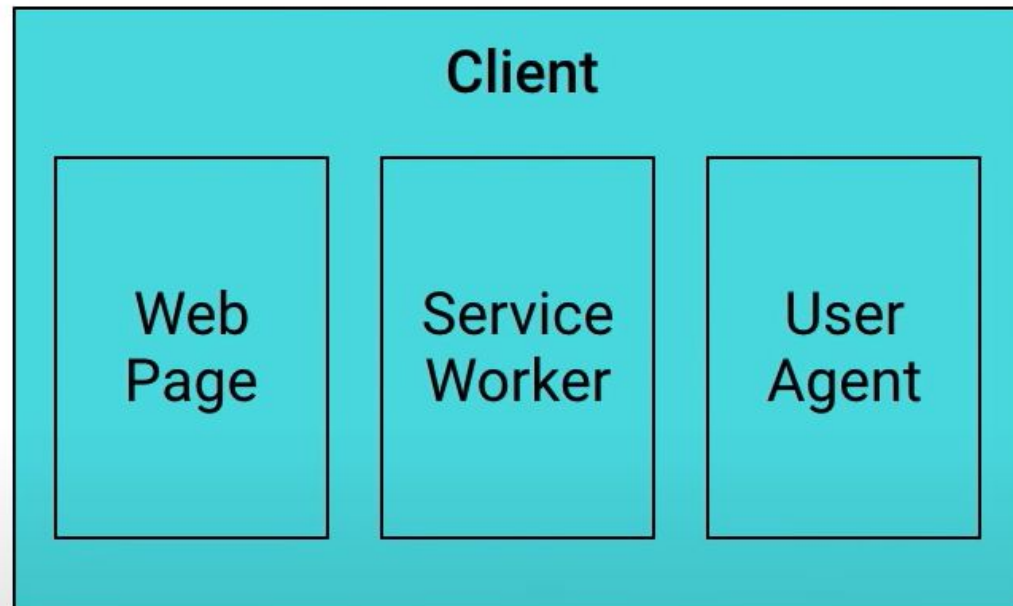
# Progressive Web App checklist

- ✓ Use a secure connection (HTTPS)
- ✓ Registers a service worker
- ✓ Manifest file
- ✓ Cross browser support
- ✓ Responsive to any screen size
- ✓ Is installable
- ✓ Provides an offline experience
- ✓ Provides context for permission requests



# Push Notifications

- System level notifications, like native apps
- Fresh content that engages users
- Works even when page is closed
- Push Notifications are assembled using two APIs: the [Notifications API](#) and the [Push API](#).



# Request permission

```
Notification.requestPermission(  
  function(status) {  
    console.log('Notification permission  
      status:', status);  
  });  
  function displayNotification() {  
    if (Notification.permission === 'granted') {  
      navigator.serviceWorker.getRegistration()  
        .then(function(reg){  
          reg.showNotification('Hello world!');  
        });  
    }  
  }  
}
```



```
var options = {  
  body: 'Here is a notification body!',  
  icon: 'images/example.png',  
  vibrate: [100, 50, 100],  
  // allows us to identify notification  
  data: { primaryKey: 1 }  
};
```

```
reg.showNotification('Hello world!', options);  
  
var options = {  
  body: 'First notification!',  
  actions: [  
    {action: 'explore', title: 'Go to the  
      site', icon: 'img/check.png'},  
    {action: 'close', title: 'No thank you',  
      icon: 'img/x.png'},  
  ]  
};  
reg.showNotification('Hello world!', options);
```

# How Web Push works

- Each browser manages push notifications through their own system, called a **"push service"**.
- This creates a special subscription object that contains the "endpoint URL" of the push service.
- You send your push messages to this URL, encrypted with the public key, and the push service sends it to the right client
- The endpoint URL contains a unique identifier

```
{
  "ref": Ref(Collection("user-subscriptions"), "264847305464611337"),
  "ts": 1588836923005000,
  "data": {
    "subscription": {
      "endpoint": "https://fcm.googleapis.com/fcm/send/dCnElwutgGU:APA91bH6_qaSJ1K4jI",
      "keys": {
        "p256dh": "BOza3v3KFfYa_iUy2ngBZdYfoCLJrmCMx8LEitqXyF3gueIuhGoTf8Tap_24iCZ1k2",
        "auth": "QIwxJ_o_uwcIeFH7bBiuxQ"
      }
    },
    "userId": "0"
  }
}
```



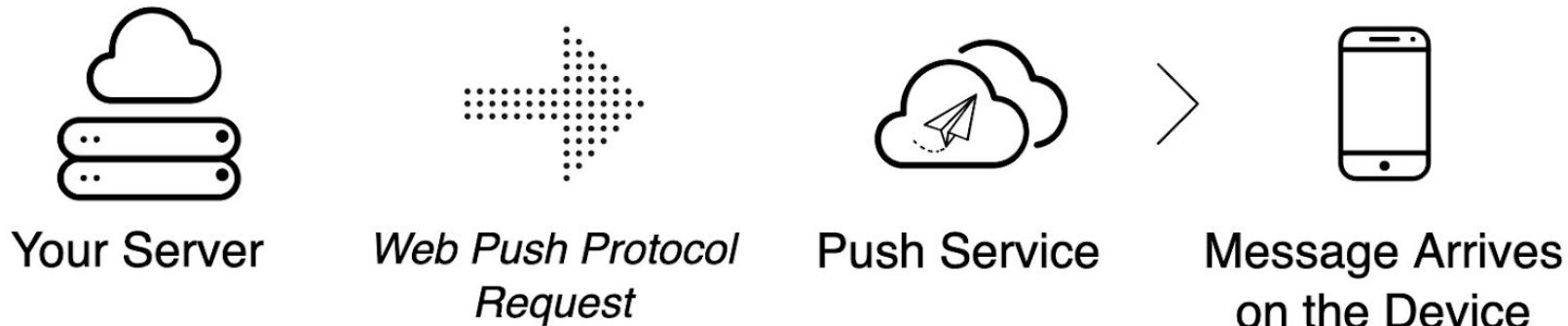
# How Web Push works

- **On the client:**

- Subscribe to the push service
- Send the subscription object to the server

- **On the server:**

- Generate the data that we want to send to the user
- Encrypt the data with the user public key
- Send the data to the endpoint URL with a payload of encrypted data.



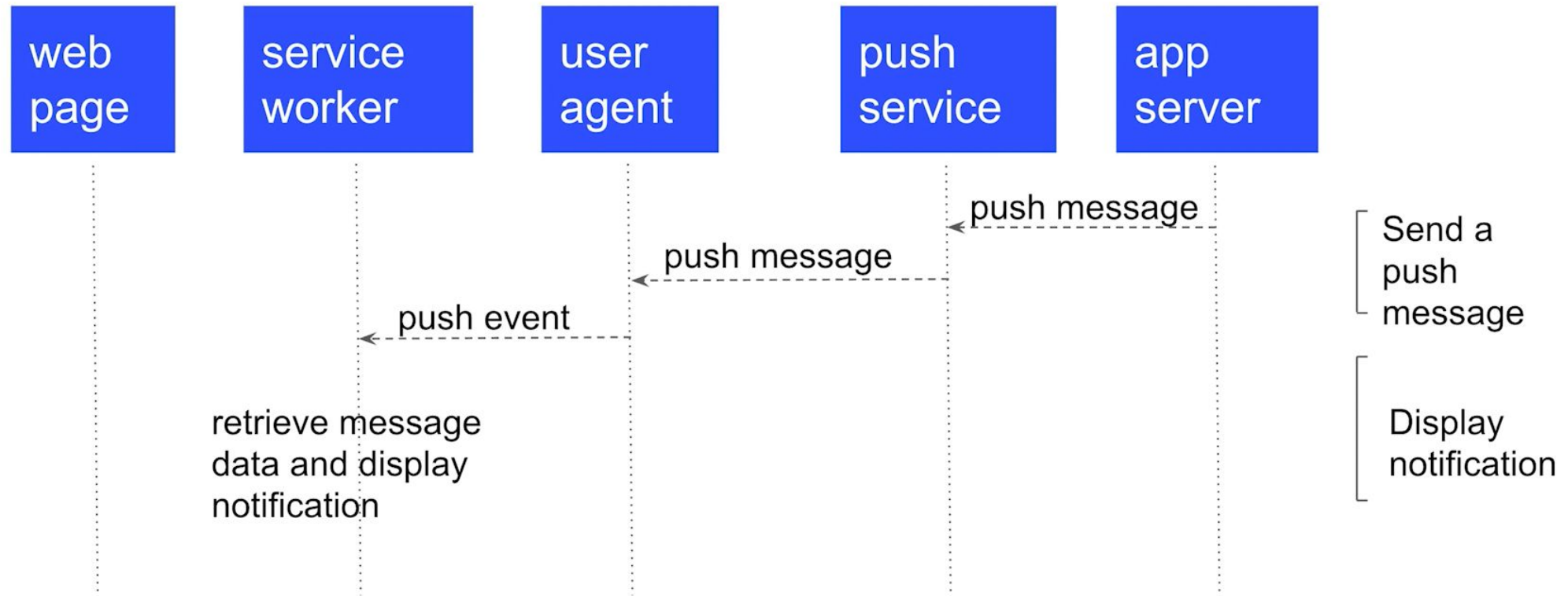
```
navigator.serviceWorker.ready
.then(function(reg) {
  reg.pushManager.getSubscription()
  .then(function(sub) {
    if (sub == undefined) {
      // ask user to register for Push
    } else {
      // You have subscription, update the
      database on your server
    }
  });
});

navigator.serviceWorker.getRegistration()
.then(function(reg) {
  reg.pushManager.subscribe({
    userVisibleOnly: true
  }).then(function(sub) {
    // send sub.toJSON() to server
  });
});
```

# The subscription object

```
{  
  "endpoint":  
    "https://android.googleapis.com/gcm/send/  
f1LsxxKp...",  
  "keys": {  
    "p256dh": "BLc4xRzK1K0RKW1b0QRv-1n...",  
    "auth": "5I2Bu2oKdyy9CwL8QVF0NQ=="  
  }  
}
```

# Send a push notification



# Send a push from server

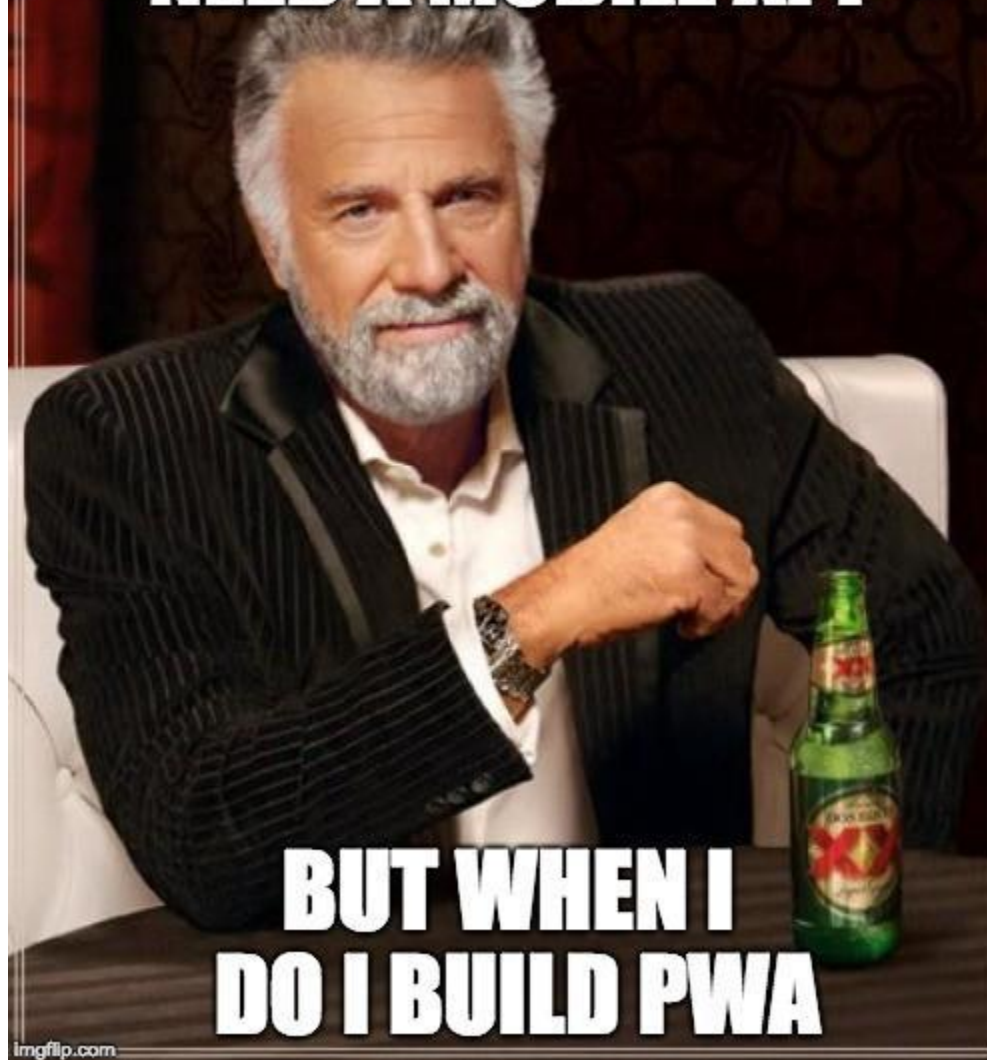
```
var webPush = require('web-push');  
var payload = 'Here is a payload!';  
var options = {  
  gcmAPIKey:  
    'AIzaSyBVImB3hJJ...8J5D4xnFo2fFI',  
  TTL: 60  
};  
webPush.sendNotification(pushSubscription,  
  payload, options);
```

# What is VAPID?

- Web Push Protocol not requiring strong authentication between your app and the push service
- Publisher optionally identify themselves using the Voluntary Application Server Identification for Web Push (VAPID) protocol.
- This provides a stable identity for the application server

```
var webPush = require('web-push');
var payload = 'Here is a payload!';
var options = {
  vapidDetails: {
    subject: 'mailto:
      example-email@example.com',
    publicKey: vapidPublicKey,
    privateKey: vapidPrivateKey
  }
};
webPush.sendNotification(pushSubscription,
  payload, options);
```

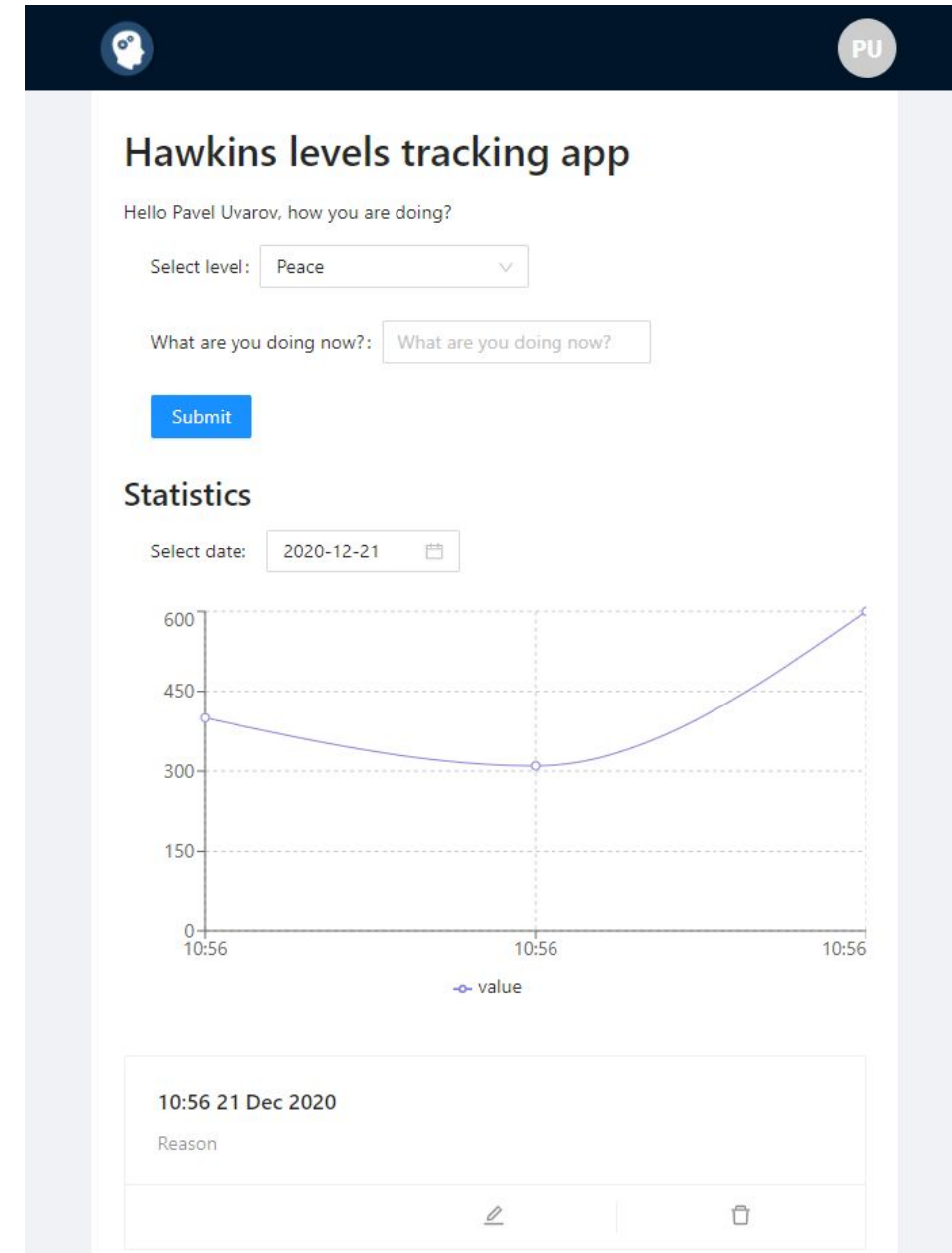
**I DON'T ALWAYS  
NEED A MOBILE APP**





# What app should do

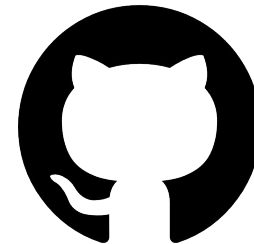
- Login/Sign in for new users
- Show notification for concrete user
- Should show notification on desktop and mobile simultaneously
- 1 hour reminder
- <https://hawkins-levels-tracker.now.sh/>





# Tech Stack

- Serverless functions
- Node.js
- FaunaDB
- CRA
- TS
- Auth verified by JWT
- SWR
- Web push
- Deployed by Vercel



<https://github.com/spiderpoul/hawkins-levels-tracker>

# CRA PWA

- <https://create-react-app.dev/docs/making-a-progressive-web-app/>

```
npx create-react-app my-app --template cra-template-pwa
```

The TypeScript equivalent is:

```
npx create-react-app my-app --template cra-template-pwa-typescript
```

# manifest.json

```
{  
  "short_name": "Tracking Hawkinks levels",  
  "name": "Tracking Hawkinks levels",  
  "icons": [ ...  
],  
  "splash_pages": null,  
  "start_url": "/",  
  "background_color": "#F7F0ED",  
  "theme_color": "#F7F0ED",  
  "display": "standalone"  
}
```

# Demo

- <https://hawkins-levels-tracker.now.sh/>
- Show on mobile
- Notifications on both platforms
- Show project, what makes it PWA, CRA service workers, manifest.json

```
export const requestPermissionsNotifications = async () => {  
  let { state } = await navigator.permissions.query({  
    name: 'notifications',  
  });  
  if (state === 'prompt') {  
    await Notification.requestPermission();  
  }  
  state = (await navigator.permissions.query({ name: 'notifications' })).  
    state;  
  if (state !== 'granted') {  
    return alert(  
      'You need to grant notifications permission for this app to work.'  
    );  
  }  
};
```

```
export const subscribePush = async (token) => {  
  // Get the `registration` from service worker and create a new  
  // subscription using `registration.pushManager.subscribe`. Then  
  // register received new subscription by sending a POST request with  
  // the subscription to the server.  
  await requestPermissionsNotifications();  
  const registration = await navigator.serviceWorker.getRegistration();  
  // Get the server's public key  
  const response = await Axios.post('/api/vapidPublicKey', { token });  
  const vapidPublicKey = await response.data?.key;  
  const convertedVapidKey = urlBase64ToUint8Array(vapidPublicKey);  
  // Subscribe the user  
  const subscription = await registration?.pushManager.subscribe({  
    userVisibleOnly: true,  
    applicationServerKey: convertedVapidKey,  
  });  
  await Axios.post('/api/subscribePush', {  
    subscription,  
    token,  
  });  
};
```

```
export const unsubscribePush = async (token) => {  
  const registration = await navigator.serviceWorker.getRegistration();  
  const subscription = await registration?.pushManager.getSubscription();  
  
  if (subscription) {  
    subscription.unsubscribe();  
    await Axios.post('/api/unsubscribePush', {  
      subscription,  
      token,  
    });  
  }  
};
```



```
const res = await Promise.all(  
  subscriptions?.map((subscription) =>  
    webPush.sendNotification(subscription)  
  )  
);
```



```
serviceWorkerScope.addEventListener(  
  'push',  
  function showPushNotification(event) {  
    try {  
      const options = { ...  
    };  
  
    const payload = event.data ? event.data.text() : 'no payload';  
  
    event.waitUntil(  
      serviceWorkerScope.registration.showNotification(payload, {  
        tag: Math.random().toString().substr(2),  
        body: `push message`,  
        icon: options.icon,  
      })  
    );  
  }  
);
```

▼ { "subscription": { "endpoint": "https://fcm.googleapis.com/fcm/send/dCnEWwutgGU:APA91b...

264847305464611337



```
{
  "ref": Ref(Collection("user-subscriptions"), "264847305464611337"),
  "ts": 1588836923005000,
  "data": {
    "subscription": {
      "endpoint": "https://fcm.googleapis.com/fcm/send/dCnEWwutgGU:APA91bH6_qaSj1K4jIQjju8A9uAt2Rn67GWHBi5UjaRxWzuY1lqaKnNGFHRjc4qcFivrB9voGQ1DqawowiDDr1vAts_3eNQykD9TXLT41e19H-X2OFR1ehu2tvERTXLRXsN-Q",
      "keys": {
        "p256dh": "BOza3v3KFFYa_iUy2ngBZdYfoCLJrmCMxBLEitqXyF3gueIuhGoTf8Tap_24iCZ1k2u2ZTexibyPP0gQ8GRXa9s",
        "auth": "QIxxWj_o_uwcIeFH7bBiuxQ"
      }
    },
    "userId": ""
  }
}
```

▼ { "subscription": { "endpoint": "https://fcm.googleapis.com/fcm/send/fv3oqsfDqzE:APA91bGJA...

264848289078182401



```
{
  "ref": Ref(Collection("user-subscriptions"), "264848289078182401"),
  "ts": 1588837861050000,
  "data": {
    "subscription": {
      "endpoint": "https://fcm.googleapis.com/fcm/send/fv3oqsfDqzE:APA91bGJA03IG1SzDhPx9FExW5NskjhBnrsYGtRpu8hf1uFiRyLUc6IOPSBvqpGFm6j35RD7oRYKA0avQji4F_GH2t1UDcLGLkV11Zf30RjmyOkkQOETLe6e1TDblnEC31iB-",
      "keys": {
        "p256dh": "BKou9cHW0n05Tf6LUJ1Zg8jOEiw6i1p1_fyOdT12bSzJRDe823965bVQx-NoHjZxtNv8sDU0ZwBjULo5JVE73T8",
        "auth": "P_fw79ZN10SmjUgP3Nm5Vw"
      }
    },
    "userId": ""
  }
}
```

▼ { "subscription": { "endpoint": "https://fcm.googleapis.com/fcm/send/evt19ynMIEE:APA91bGklI...

265161869610713601



```
{
  "ref": Ref(Collection("user-subscriptions"), "265161869610713601"),
  "ts": 1589136914750000,
  "data": {
    "subscription": {
      "endpoint": "https://fcm.googleapis.com/fcm/send/evt19ynMIEE:APA91bGklIIPB8RdkDUGjWqH41UKMA1b848ICDwbLDOzaCRU_s9kFfj-hWosclm_IYN5DsSW8n26MepetZDas1TdtY0DUZZnQzBcN81ghtV_uZdZjkgBGrZbb_BuyVgKK1xVr8",
      "keys": {
        "p256dh": "BLrC8RT0dMeiN7Jc2MJumPLXgQJUjDihukLTuoYuo1jfK0hzKewojd-0E0tv2i1aoY5b3Exv1-Lc18CpL3Vx3PE",
        "auth": "3bJhqdfQqY1tjQEn3Ad2Xw"
      }
    },
    "userId": ""
  }
}
```



**ANDROID/IOS  
APP**



**HYBRID APP**



**PROGRESSIVE  
WEB APP**

# PWA limitations

- Still a new technology – all browsers don't support it fully yet
- More limited hardware access
- Limited performance for computation heavy operations – although WebAssembly is improving this
- High battery usage

# Links

- <https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures>
- <https://web-push-book.gauntface.com/>
- <https://medium.com/progressivewebapps/history-of-progressive-web-apps-4c912533a531>
- <https://serviceworker.rs/push-simple.html>
- <https://web.dev/progressive-web-apps/>



Спасибо за внимание!



**WEBPURPLE**