6. Aufgabe – Zippie AddToHomeScreen/ Notifications

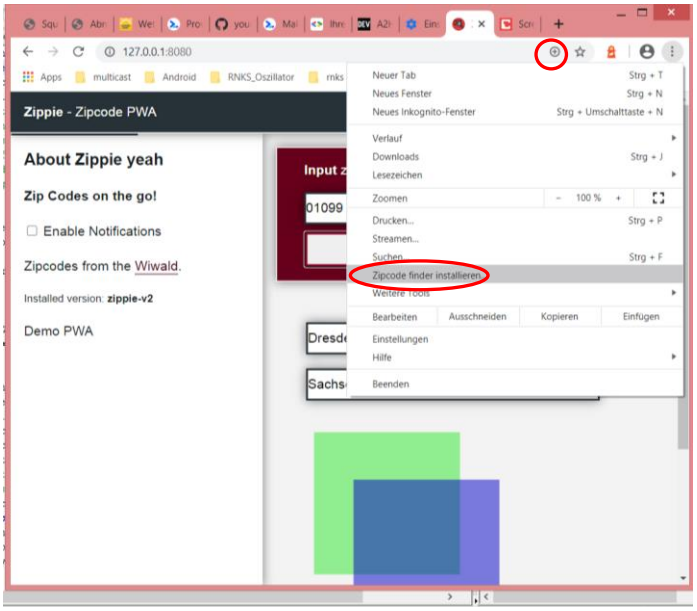
Beschreibung:

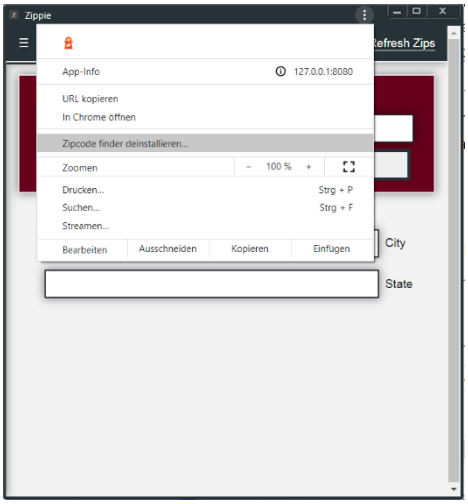
Fertigstellung der Zippie-PWA und Ergänzung um **AddToHomeScreen** Funktionalität, (**Push) Notifications**

Aufgaben:

I. Add to Home Screen (A2HS) (*beforeinstallprompt-Event).*

a. Wie Sie vielleicht schon bemerkt habe, lässt sich die PWA bereits installieren, da alle Voraussetzungen bereits geschaffen wurden (siehe auch Lighthouse Report). Zum einen ist die Installation über das plus in der Adresszeile möglich, aber auch über das Menü der drei Punkte (in Chrome) – siehe Abbildung unten links.





Probieren Sie es aus!

Hinweis: Für weiteres Testen des Installationsvorganges (falls nötig): die App deinstallieren. Dazu den Link mit App in Chrome öffnen, über die drei Punkte oben die App öffnen auswählen. Die App öffnet sich ohne Browserrahmen. Dort wiederum die drei (senkrechten) Punkte anwählen und deinstallieren wählen (siehe Abbildung oben rechts).

b. Um die Installation etwas nutzerfreundlicher zu gestalten, nehmen Sie nun in die Sidebar Ihrer html Datei folgenden Button auf:

```
<button class="a2hs-button">Add to home screen</button>
```

Da Ihr App nun alle Voraussetzungen für die Installation erfüllt, wird das Event *beforeinstallprompt* gefeuert. Erst mit diesem Event soll der Button erscheinen und die Installation aus der App heraus ermöglicht werden.

Erstellen Sie dazu eine neue install.js Datei und binden Sie diese in Ihre HTMl Datei ein (in Analogie zur script.js).
Übernehmen Sie dann folgenden Code in die install.js:

install.js

```
let deferredPrompt;
const addBtn = document.querySelector('.a2hs-button');
addBtn.style.display = 'none';

//Add event listener for beforeinstallprompt event
window.addEventListener('beforeinstallprompt', (e) => {
  console.log('beforeinstallprompt', e);
  e.preventDefault();

  deferredPrompt = e;
  // Update UI to notify the user they can add to home screen
  addBtn.style.display = 'block';

  addBtn.addEventListener('click', (e) => {
    // hide our user interface that shows our A2HS button
    addBtn.style.display = 'none';
    // Show the prompt
    deferredPrompt.prompt();
    // Wait for the user to respond to the prompt
    deferredPrompt.userChoice.then((choiceResult) => {
        if (choiceResult.outcome === 'accepted') {
          console.log('User accepted the A2HS prompt');
        } else {
          console.log('User dismissed the A2HS prompt');
        }
        deferredPrompt = null;
      });
```

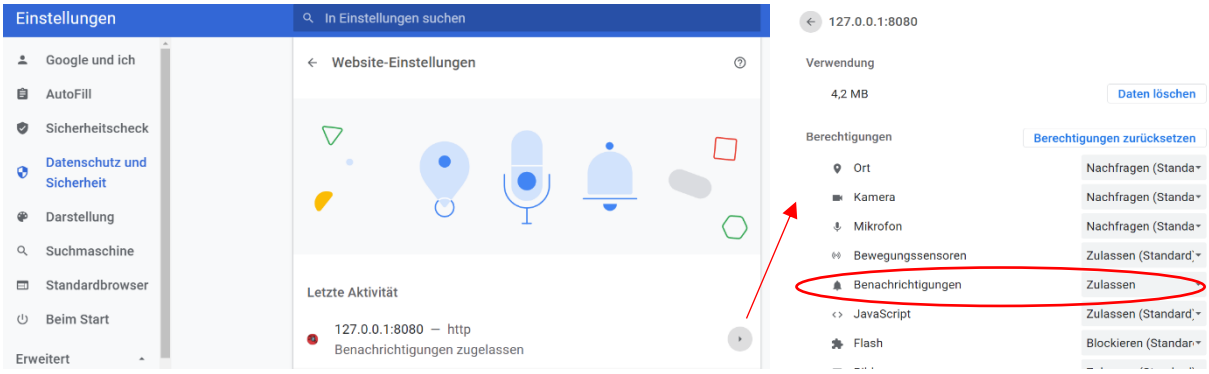
Testen Sie das Ganze! Machen Sie ein Screenshot mit der Anzeige des Buttons und der Anzeige des Installationsprompts! Speichern Sie das Ganze in der Datei mit folgendem Namen: prakt6_A1b

Hinweis: Bei Fehlerausschriften: Denken Sie daran, **alle** statischen Assets im Worker-Cache zu halten!

II. Push-Notifications

a. Tatsächlich existieren 2 APIs: die Notification API und die Push API. Erstere ist leichter und bezieht sich auf Nachrichten, die vom Service Worker an die geöffnete WebApp zugestellt werden. Letztere bedingt einen zusätzlichen Push – Server und ist für die Kommunikation zwischen diesem und dem Service Worker nutzbar. Beide können isoliert aber auch in Kombination genutzt werden. Fangen wir mit der Leichteren an.
Bevor Benachrichtigungen in der Webanwendung angezeigt werden können, muss zunächst der Anwender seine Erlaubnis geben. Das kann zum Beispiel über die Methode `requestPermission` des `Notification` Objektes geschehen. Allerdings sollte dies nicht sofort mit dem Laden der Seite geschehen, um den Nutzer nicht reihenweise mit Popups zu belästigen (Best Practices), sondern erst nach Betätigen eines Buttons (z. B: die Checkbox mit Label „Enable Notifications“). Dazu mehr in II.b.

Eine andere Möglichkeit besteht in der manuellen Vergabe der Erlaubnis für die Benachrichtigungen über die Chrome Einstellungen
Einstellungen -> Datenschutz und Sicherheit -> Website Einstellungen unter letzte Aktivität. Dort die Zippie-App aussuchen und die Berechtigung für Benachrichtigungen ggf. auf „Zulassen“ ändern.



Notifications anzeigen

Ergänzen Sie nun die folgende Notification in der Callback Methode für das *load* Event. Hier wird einmalig bei der Installation der App eine lokale Benachrichtigung ausgegeben:

script.js

```
//-------worker registrieren----//
window.addEventListener("load", async () => {
  if ("serviceWorker" in window.navigator) {

     const registration = await
        window.navigator.serviceWorker.register("worker.js");

     // Prakt6 (local) Notification:
     // If there's no active service worker this is the first installation
     if (!registration.active && window.Notification.permission === "granted") {
        new Notification("Ready for offline use", {
            icon: "img/icon192.png", badge: "img/badge.png",
            body: "You can use this web app at any time,
                   even when you're offline.",
            tag: "installed",
        });
      }
      // prakt6 : Ende notification
    …
}
```

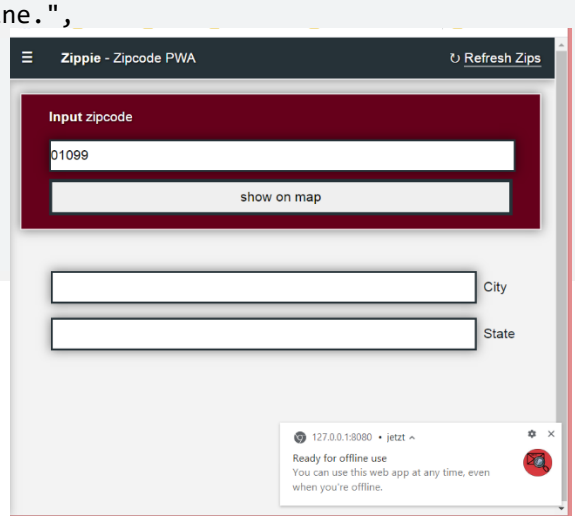


Abb. rechts: so sollte es dann beispielsweise (nur) nach der Neuinstallation der App aussehen, falls Benachrichtigungen erlaubt wurden.

Keine Notification unter z.B. Windows 10?? Checken Sie: Systemeinstellungen-> Benachrichtigungen und Aktionen für Chrome!

b. Request Permission:

Mit dem Anwählen der Checkbox in der Sidebar verbinden wir nun zweierlei: zum einen die Abfrage nach den Permissions, zum anderen soll der Service Worker

script.js

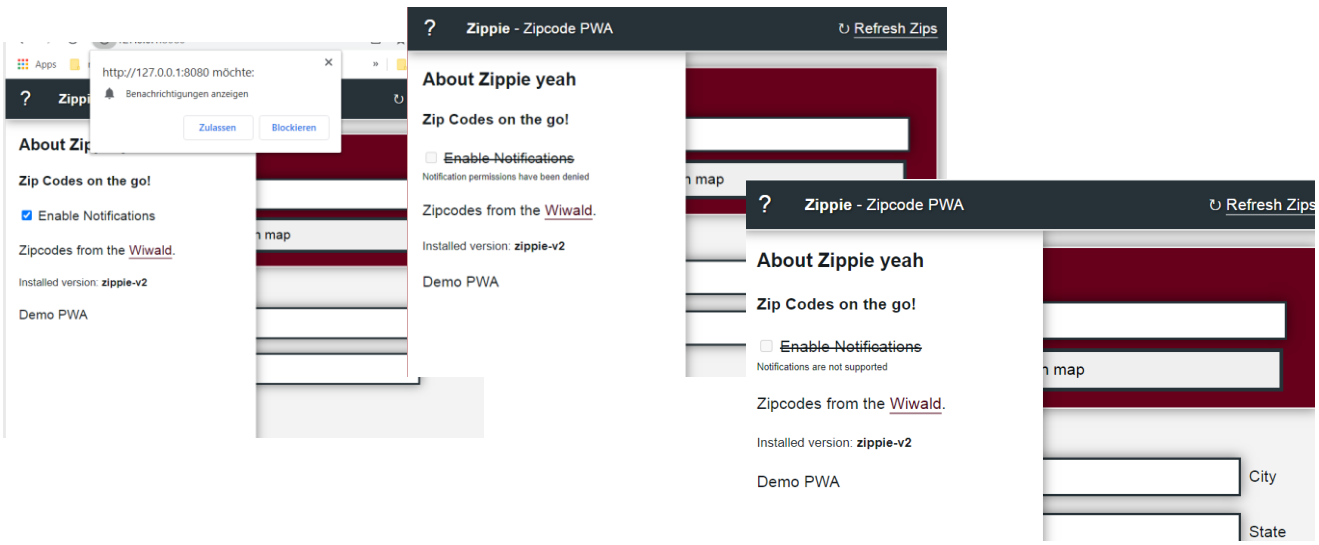
```
const notificationCheckbox = $(".notifications");
const init = (zipcodes) => {
     console.log("--> in init");
     // Setup event handlers for inputs, links and notification
     // Execute a function when the user presses down a key on the keyboard
     ...
     // Prakt6: Notification checkbox
     on( notificationCheckbox, "change", handleNotificationCheckboxChange);
}
```

darüber informiert werden, ob Benachrichtigungen erlaubt sind oder nicht. Ergänzen Sie daher in script.js in der init-Funktion den Eventhandler für die Notification Checkbox, als auch den Zugriff auf das Element.

Allerdings ist das Anwählen der Checkbox nur dann sinnvoll, wenn Benachrichtigungen vom Browser überhaupt unterstützt werden bzw. diese nicht schon vorab blockiert wurden. Daher ergänzen Sie oberhalb der Anbindung des Eventhandlers noch folgendes in der init-Funktion:

```
const notificationCheckbox = $(".notifications");
const init = (zipcodes) => {
   console.log("--> in init");
   // If notifications are not supported, disable and uncheck the checkbox
   if (!("Notification" in window)) {
        notificationCheckbox.setAttribute("disabled", true);
        notificationCheckbox.checked = false;
        notificationCheckbox.parentElement.classList.add("not-supported");
   }
   // If notification permissions have been denied, disable and uncheck
   // the checkbox
   if ("Notification" in window && window.Notification.permission === "denied")
   {
        notificationCheckbox.setAttribute("disabled", true);
        notificationCheckbox.checked = false;
        notificationCheckbox.parentElement.classList.add("disabled");
    }
    ...
     // Prakt6: Notification checkbox
     on( notificationCheckbox, "change", handleNotificationCheckboxChange);
}
```

Ergänzen Sie Ihre style.css mit den erforderlichen Klassen für die Elemente.



Im Folgenden sind die Inhalte des Eventhandlers und der ausgelagerten Funktion für die Kommunikation per PostMessage mit dem ServiceWorker gegeben:

```
const setNotificationsState = async (value, postToWorker = true) => {
  if (!("Notification" in window)) {
    return;
  }
  const permission = window.Notification.permission;
  if (postToWorker && "serviceWorker" in window.navigator) {
    const registration = await window.navigator.serviceWorker.getRegistration();
    if (registration.active) {
      registration.active.postMessage({
        type: "SET_NOTIFICATION_PERMISSIONS",
        payload: value,
      });
    }
  }
  notificationCheckbox.checked = (value && permission === "granted");
}

const handleNotificationCheckboxChange = async (event) => {
  if (!("Notification" in window)) {  //for mobile safari
    return;
  }
  if (event.target.checked) {
    const permission = await window.Notification.requestPermission();
    if (permission === "granted") {
      setNotificationsState(true)
      return;
    } else {
      setNotificationsState(false);
    }
  } else {
    setNotificationsState(false);
  }
};
```

c. Notifications vom Service Worker:
   Bisher haben wir nur lokale Benachrichtigungen ermöglicht. Folgende Schritte müssen unternommen werden, um (Push) Benachrichtigungen an die App weiterleiten zu können:
   1. Ergänzen Sie den `EventListener` für das Ereignis „`message`“ im Service Worker für den Nachrichtentyp `SET_NOTIFICATION_PERMISSIONS`! Speichern Sie sich den Wert des Payloads in der Variablen `userDefinedPermission`!

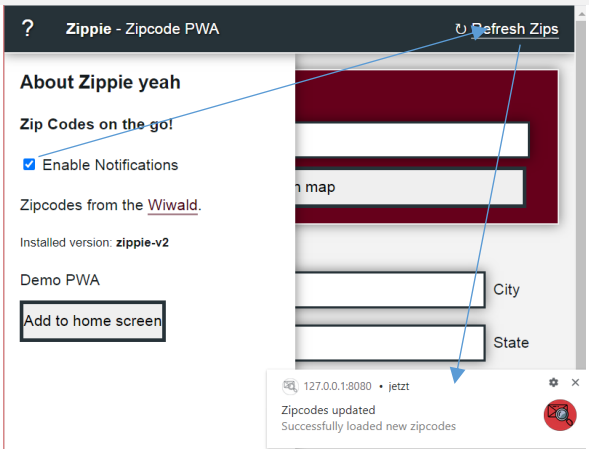
2. Im Falle eines Refreshs soll eine Benachrichtigung vom SW an die App geschickt werden. Übernehmen Sie folgenden Quellcode oberhalb der handleRefresh Funktion.

```
const asCacheUrl = (url) => {
  return new Promise( async (resolve) => {
    const response = await caches.match(url);
    // If there's no cache entry to be found, resolve with the real url as a
    // fallback for the data url from the cache
    if (!response) {
      return resolve(url);
    }
    const blob = await response.blob();
    const reader = new FileReader();
    reader.onloadend = () => resolve(reader.result);
    reader.readAsDataURL(blob);
  });
}

const asCacheUrls = (urls) => Promise.all(urls.map( (url) => asCacheUrl(url) ));

const notify = async (title, data = {}) => {
  if (self.registration
    && self.Notification.permission === "granted"
    && userDefinedPermission === true
  ) {
    const [ icon, badge ] = await asCacheUrls([
      "img/icon192.png", "img/badge.png"
    ]);
    const options = Object.assign({ icon, badge, }, data);
    const notification = self.registration.showNotification(title, options);
    return notification;
  }
}
```

3. Ersetzen Sie nun die Funktion `handleRefresh` mit dem nachfolgenden Code, der Benachrichtigungen an die App sendet im Fall erfolgreich bezogener Daten bzw. im Fehlerfall.

```
// Handle requests for new Zipcode. Attempt to get Zipcode from the network, fall
// back to cache if something goes wrong
const handleRefresh = async () => {
  try {
    const response = await fetch("api/latest.json");
    if (response.ok) {
        const clone = response.clone();
        const cache = await caches.open(CACHE_ID);
        await cache.put("api/latest.json", clone);
        console.log("--> in handleRefresh: cache updated!")
        //prakt6 - Notifications
        notify("Zipcodes updated", {
                body: `Successfully loaded new zipcodes`,
                tag: "rates",
        });
        //prakt6 - Ende
        return response;
      } else
        throw new Error(`Zip code request failed with code ${ response.status }`)
    }
  } catch (err) {
    console.log("Failed to update zipcodes",err.message);
    //prakt6 - Notifications
    notify("Failed to update rates", {
      body: err.message,
      tag: "rates",
    });
    //prakt6 - Ende
    return self.caches.match("api/latest.json");
  }
};
```

Wer sich noch mit Push Nachrichten vom Server zum SW beschäftigen möchte, schaue beispielsweise mal hier.



Bitte auf Ihr Repo in den Ordner prakt6 pushen!