

```
1
2
3  @Progressive (Web-Apps){
4
5     presentation: seminar;
6
7
8
9     /* Charlotte Brown */
10
11
12 }
13
14
```

```
1 Table Of 'Contents' {
2
3
4     01 Understanding
5       What are PWA's?
6
7         02 Presentation
8           Good thing I made one.
9
10           03 Key Features
11             Now I made one, let's talk
12             about it.
13 }
14
```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

01 {

[Understanding]

Let's get into it.

}

# Progressive Web Apps < 1/2 > {

Made with Web Technologies (HTML, CSS, JS), it provides users with a service as an application but runs like an app you'd get off the app store.

It can be used offline, installed onto different devices and be integrated into current apps on the same system.

```
/* MDN Web Docs – PWAs (2024) */
```

```
1 <QUOTE>
```

```
2  
3  
4     “[PWAs]... are built and enhanced with  
5     modern APIs to deliver enhanced  
6     capabilities, reliability, and  
7     installability while reaching anyone,  
8     anywhere, on any device with a single  
9     codebase.”
```

```
10  
11     /* Pete LePage – Wed.dev (2024)*/
```

```
12  
13 </QUOTE>
```

```
1 Continued.. < 2/2 > {  
2
```

```
3 |  
4 |     Progressive Web Apps can be installed by  
5 |     anyone, anywhere. They don't need to be hosted  
6 |     through a specific company app store, you can  
7 |     launch your own through your website.
```

```
8 |  
9 |     We can even discover PWA's in the search  
10 |    engine, if the app has a manifest. The browser  
11 |    is able to view them too!
```

```
12 | }  
13 |  
14 |
```

# Steps to make 'a Web App' {

Step 01 Create your app, using index.html,  
style.css and script.js

Step 02 Create your manifest within a JSON  
file.

Step 03 Set up your Web Services in another  
JS file.

Step 04 Host your website over a secure  
network (localhost or HTTPS).

}

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

02 {

[Presentation]

Let's get into it.

}



Let's 'Look' {  
Together!



<https://charlotteblogs.uk/pages/side-projects/pwa/index.html>

# Standard HTML, CSS & JS

```

1  <!--This is Charlotte's boilerplate, and if you are seeing this with the intent to use it -->
2
3  <meta charset="UTF-8">
4  <meta name="viewport" content="width=device-width, initial-scale=1">
5
6  <!--SEO Content Description tag-->
7  <meta name="description"
8  |   content="This is my Self Reminder Website Application, where
9  |   you can set reminders for things you need to do." />
10
11 <!--This doesn't have to be a PNG, change it to whatever you'd like-->
12 <link rel="icon" type="image/svg" href="heart.svg">
13 <!-- A title is a great help to SEO, and a good practise should be used -->
14 <title>Reminders | ReminderTime</title>
15 <!--Stylesheet File-->
16 <link rel="stylesheet" href="style.css">
17 <!-- Our Manifest for PWA's -->
18 <link rel="manifest" href="manifest.json">

```

```

// < Our function to fetch those reminders from JSON >
function getAllReminders(){
  // < Data is our keyword to fetch individual reminders from JSON with our storage key, which we use to store our data
  const data = window.localStorage.getItem(STORAGE_KEY);
  // < We parse the new data, which means essentially just resorting/checking it as data we can work with
  const reminders = data ? JSON.parse(data) : [];
  // < DIR displays a list of all the properties of the object, check the MDN for a better explanation
  console.dir(reminders);
  // < LOG sends a message to the console, and is apart of the Web Workers spec
  console.log(reminders);
  // < We return all of our reminders
  return reminders;
}

```

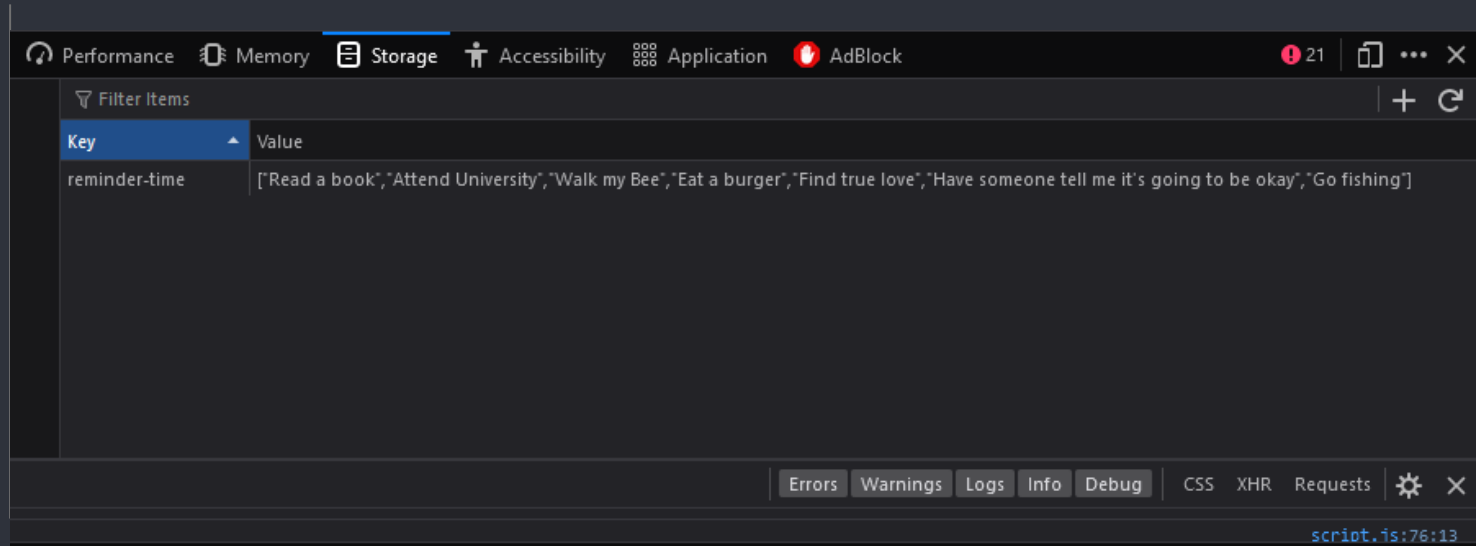
```

/*Fonts*/
font-family: Verdana, Geneva, Tahoma, sans-serif;
text-size-adjust: none; /* Chrome on Android devices likes to scale text, so this prevents that */

/*Spacing and Sizing*/
margin: 1rem;

```

# JSON Local Storage



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

03 {

[Key Features]

Let's get into it.

}

# Our Manifest; {

## What is a Manifest?



A file that outlines the metadata and behaviours of our app.

## Why do we need it?



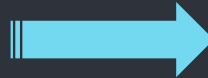
It helps outline data the clients' machine will need to help set up the app's installation properly.

## What does it contain?



Let's find out!

```
1 Our Manifest{
2
3     "name": "Reminder Time",
4     "short_name": "RT",
5     "description": "An app that helps you with
6     creating reminders.",
7     "start_url": "/",
8     "theme_color": "#ecb0b0",
9     "background_color": "#fff",
10    "display": "standalone",
11    "icons": [
12        {
13            "src": "heart.svg",
14            "sizes": "512x512"
15        }
16    ]
17 }
```



# Our Service Workers; {



## What are Service Workers?

A file that helps the app provide the minimal support for offline usage.



## What does it do?

They help the app run off cached assets (an 'offline first' approach) before resorting to the network to gain new information.

'A service worker functions like a proxy server, allowing you to modify requests and responses replacing them with items from its own cache.'

# Our Service Workers; {

```
1 // < When we want to access the app offline or avoid the network,  
2 // we add an event listener that catches the fetch request from the user  
3 self.addEventListener("fetch", (event) => {  
4   // < As a single-page app, direct the app to always go to the cached home page instead of fetching a new one  
5   if (event.request.mode === "navigate") {  
6     event.respondWith(caches.match("/"));  
7     return;  
8   }  
9  
10  event.respondWith(  
11    (async () => {  
12      try {  
13        const networkResponse = await fetch(event.request);  
14        const cache = await caches.open(CACHE_NAME);  
15        // < Update the cache with the latest response from the network  
16        cache.put(event.request, networkResponse.clone());  
17        // < Return the network response  
18        return networkResponse;  
19      } catch (error) {  
20        // < If the network request fails, try to serve from the cache  
21        const cacheResponse = await caches.match(event.request);  
22        return cacheResponse || new Response(null, { status: 404 });  
23      }  
24    })()  
25  );  
26 }
```



# Our Service Workers; {

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

```
const VERSION = "v1";

const CACHE_NAME = `reminder-time-${VERSION}`;

const APP_STATIC_RESOURCES = [
  "/",
  "/index.html",
  "/style.css",
  "/script.js",
  "/heart.svg",
]
```

}

# What Else Goes In 'SW.JS' ?{

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14



## Cache Upload

We cache our static resources to the local machine



## Cache Deletion

To avoid conflict, we delete our old cache



## Fetch Requests

If the user is offline, we need to focus on fetching cached files first



## 404 Handling

In case we can't access our cache or new data from the net, we 404 it!

}

1 'SW.JS' & 'index.html' {  
2  
3

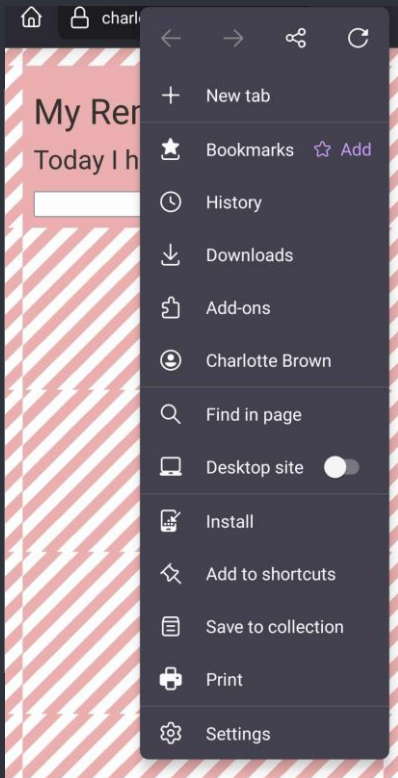
```
4 <script src="script.js" defer></script>  
5 <!-- Register the app's service worker. -->  
6 <script>  
7   if ("serviceWorker" in navigator) {  
8     navigator.serviceWorker.register("sw.js")  
9       .then((registration) => {  
10        console.log("Service Worker registered with scope:", registration.scope);  
11        })  
12        .catch((error) => {  
13          console.error("Service Worker registration failed:", error);  
14        });  
15   }  
16 }
```

13  
14 }

index.html

style.css

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14



Mobile Ver.




# A Quick Overview {

## Languages

JS &  
JSON  70%

The main magic happens  
in our sw.js file

HTML,  
CSS  30%

We can turn any website  
into a PWA if we work  
hard enough.

## Big Steps



Create a  
Manifest



Create your  
Service  
Worker



Watch the  
magic happen

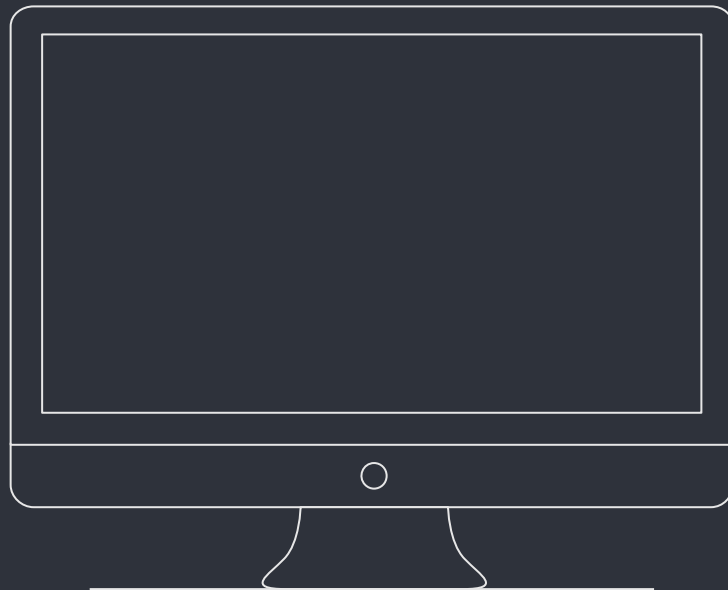
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

## Give it a Go

```
{
```

```
    Find full commented  
    code on my GitHub, or  
    follow the helpful  
    tutorials on MDN Web  
    Docs
```

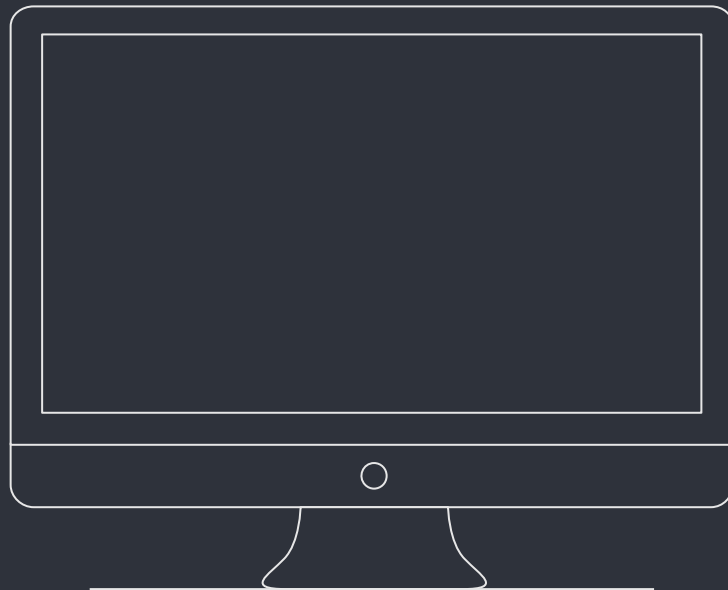
```
}
```



1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

# Thank You!

Thank you to MDN Web  
Docs, Chrome Developer  
Tools, Firefox  
Developer Handbook,  
Microsoft Learning Hub  
for my information!



```
1 Thank You! ; {
2
3     'Do you have any questions?'
4
5     Thank you to MDN Web Docs,
6         Chrome Developer Tools,
7         Firefox Developer Handbook,
8         Microsoft Learning Hub for
9         my information!
10
11     Credit to Freepix for the icons
12         used in this presentation.
13
14     You can find the example page
15         on Charlotteblogs.uk or
16         Github/cb1140
17 }
```