**BAB V**

**PWA #2**

* 1. **Tujuan**
  2. Praktikan mengetahui apa itu Progressive Web Application.
  3. Praktikan memahami kegunaan dari Progressive Web Application.
  4. Praktikan mengetahui cara mengimplementasikan Progressive Web Application.
  5. Praktikan dapat memahami cara kerja dari Progressive Web Application.
  6. **Alat dan Bahan**
  7. PC / Notebook
  8. Google Chrome (ver 52 atau diatasnya)
  9. Web Server for Chrome (ekstensi google chrome)
  10. Notepad++ / Sublime text 3/Visual Studio Code (Text editor)
  11. **Dasar Teori**
      1. **Progressive Web Application**

PWA adalah sebuah aplikasi web yang menggunakan beberapa teknologi modern yang membuat user experience seperti menggunakan Native App (Android/iOS).

Dengan konsep offline first dan Web APIs baru, aplikasi yang berjalan pada browser tidak akan memunculkan pesan network error atau white screen seperti yang selama ini seringkali kita alami ketika berada pada jaringan yang sangat buruk. Pada PWA, ketika kita meninggalkan browser, dan kembali lagi membuka aplikasi tersebut pada jaringan buruk atau tanpa jaringan sekalipun kita tidak akan kehilangan UI dan data-data terakhir.

* + 1. **Service Worker**

Service Workers adalah teknologi dibalik PWA yang memberikan fungsionalitas offline, notifikasi, update konten, pergantian konektifitas dan lainnya. Service workers adalah semacam proxy yang berada ditengah antara aplikasi web, browser, dan jaringan, atau lebih sederhana lagi bisa dianalogikan aplikasi kita akan diatur oleh service workers agar bisa berlajan offline terlebih dahulu, mengembalikan data dari cache jika request ke jaringan gagal.

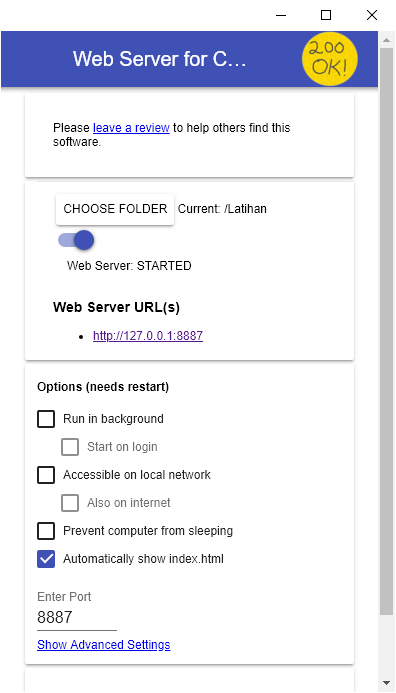
* + 1. **Web Server**

Web server adalah sebuah software yang memberikan layanan berbasis data dan berfungsi menerima permintaan dari HTTP atau HTTPS pada klien yang dikenal dan biasanya dikenal dengan nama web browser (Mozilla Firefox, Google Chrome) dan untuk mengirimkan kembali yang hasilnya dalam bentuk beberapa halaman web dan pada umumnya akan berbentuk dokumen HTML.

Fungsi utama Server atau Web server adalah untuk melakukan atau akan mentransfer berkas permintaan pengguna melalui protokol komunikasi yang telah ditentukan sedemikian rupa. halaman web yang diminta terdiri dari berkas teks, video, gambar, file dan banyak lagi. pemanfaatan web server berfungsi untuk mentransfer seluruh aspek pemberkasan dalam sebuah halaman web termasuk yang di dalam berupa teks, video, gambar dan banyak lagi.

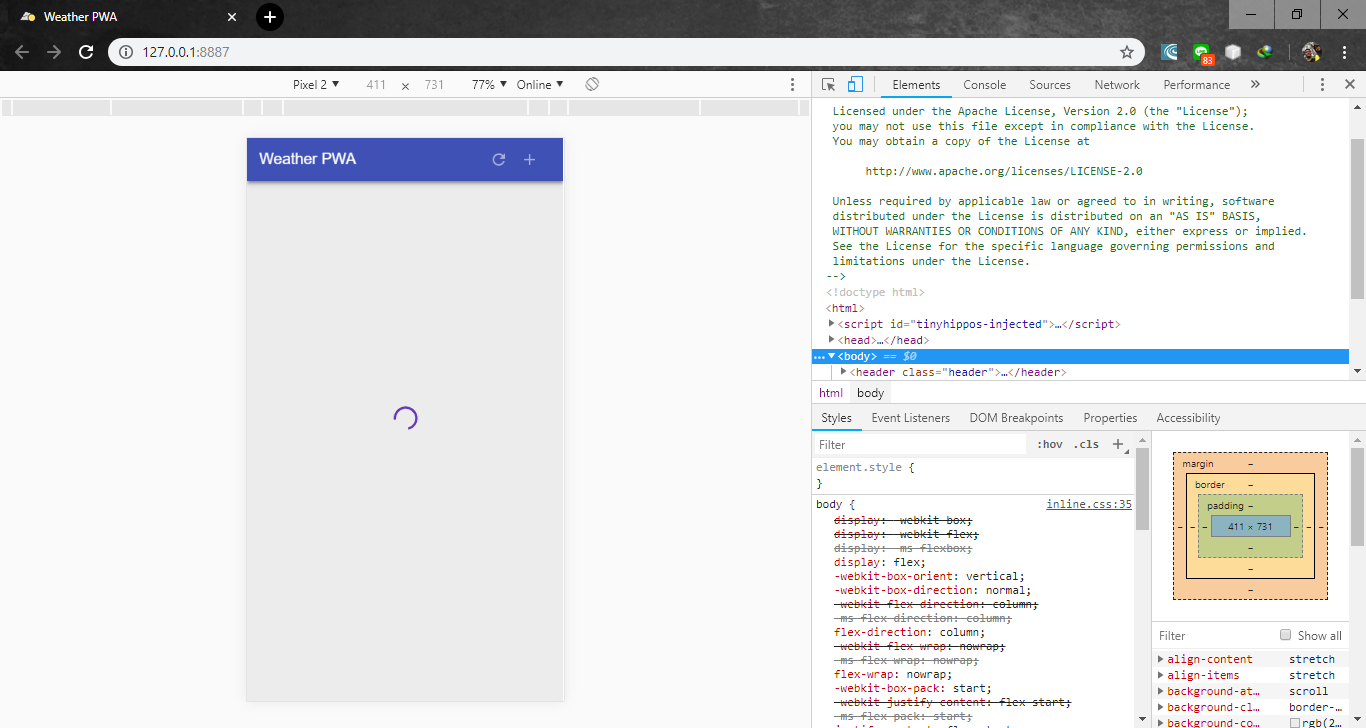
* 1. **Langkah Percobaan**

1. Menyalakan dan mengkonfigurasi Web Server for Chrome.



Choose folder lalu pilih folder Latihan yang telah didownload sebelumnya, kemudian centang Automatically show index.html. Lalu nyalakan web server.

1. Buka link dari web server dan lihat tampilannya.



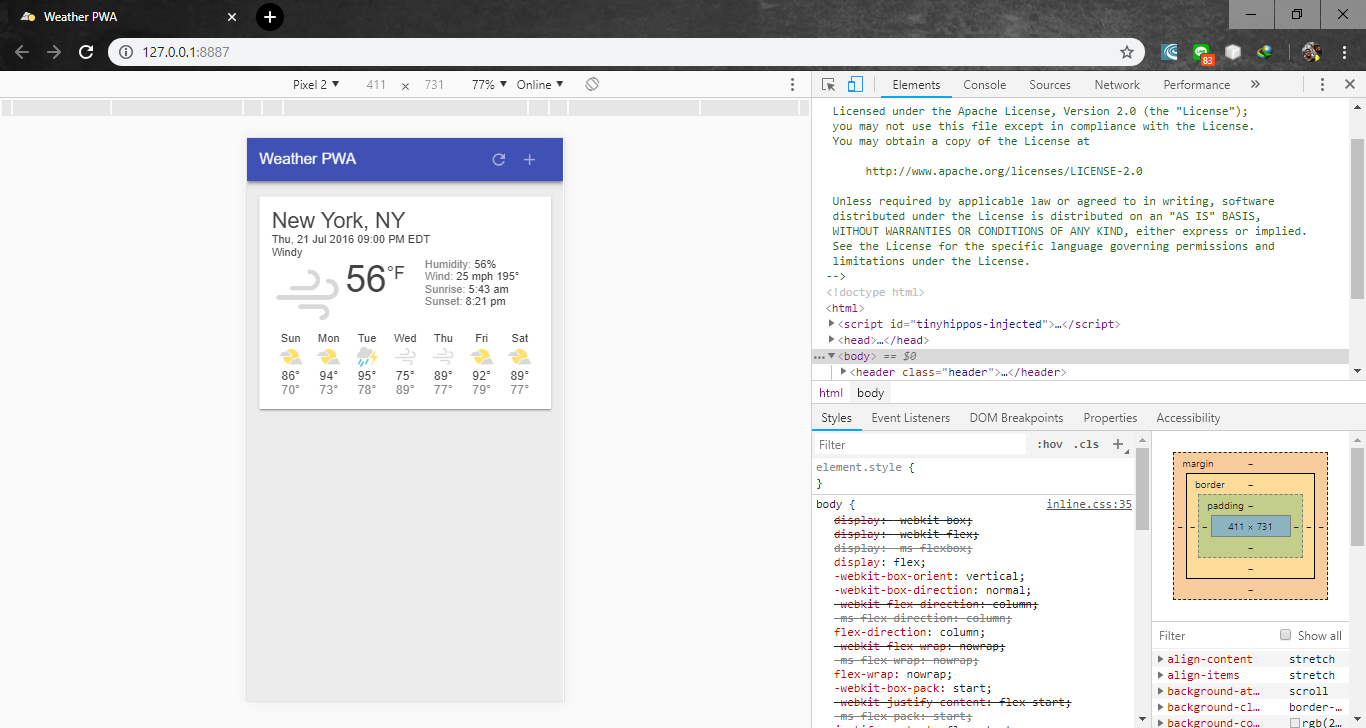
1. Untuk melihat bagaimana data cuaca palsu dirender, hilangkan tanda komentar pada baris berikut di bagian bawah file index.html

|  |
| --- |
| <!--<script src="scripts/app.js" async></script>--> |

Berikutnya, hilangkan tanda komentar pada baris berikut di bagian bawah file app.js

|  |
| --- |
| // app.updateForecastCard(initialWeatherForecast); |

Muat ulang aplikasi. Hasilnya akan menjadi kartu prakiraan cuaca yang terformat dengan baik (menggunakan data palsu yang telah dibuat) dengan spinner dinonaktifkan, seperti ini:



1. Tambahkan kode yang diperlukan untuk menyimpan preferensi pengguna. Temukan komentar TODO berikut dalam app.js

|  |
| --- |
| // TODO add saveSelectedCities function here |

Dan tambahkan kode berikut di bawah komentar.

|  |
| --- |
| // Save list of cities to localStorage.  app.saveSelectedCities = function() {  var selectedCities = JSON.stringify(app.selectedCities);  localStorage.selectedCities = selectedCities;  }; |

1. Berikutnya, tambahkan kode startup untuk memeriksa apakah pengguna memiliki kota yang disimpan dan merender kota tersebut, atau menggunakan data yang dimasukkan. Temukan komentar berikut:

|  |
| --- |
| // TODO add startup code here |

Dan tambahkan kode berikut di bawah komentar.

|  |
| --- |
| app.selectedCities = localStorage.selectedCities;  if (app.selectedCities) {  app.selectedCities = JSON.parse(app.selectedCities);  app.selectedCities.forEach(function(city) {  app.getForecast(city.key, city.label);  });  } else {  /\* The user is using the app for the first time, or the user has not  \* saved any cities, so show the user some fake data. A real app in this  \* scenario could guess the user's location via IP lookup and then inject  \* that data into the page.  \*/  app.updateForecastCard(initialWeatherForecast);  app.selectedCities = [  {key: initialWeatherForecast.key, label: initialWeatherForecast.label}  ];  app.saveSelectedCities();  } |

Kode startup memeriksa bila ada kota yang disimpan dalam penyimpanan lokal. Jika ada, itu akan mem-parse data penyimpanan lokal dan kemudian menampilkan kartu prakiraan untuk masing-masing kota yang disimpan. Jika tidak, kode startup akan menggunakan data prakiraan palsu dan menyimpannya sebagai kota default.

1. Mengubah penangan tombol "add city" untuk menyimpan kota yang dipilih ke penyimpanan lokal. Cari butAddCity pada app.js lalu ubah menjadi seperti berikut.

|  |
| --- |
| document.getElementById('butAddCity').addEventListener('click', function() {  // Add the newly selected city  var select = document.getElementById('selectCityToAdd');  var selected = select.options[select.selectedIndex];  var key = selected.value;  var label = selected.textContent;  if (!app.selectedCities) {  app.selectedCities = [];  }  app.getForecast(key, label);  app.selectedCities.push({key: key, label: label});  app.saveSelectedCities();  app.toggleAddDialog(false);  }); |

1. Selanjutnya mendaftarkan service worker. Temukan komentar berikut pada app.js

|  |
| --- |
| // TODO add service worker code here |

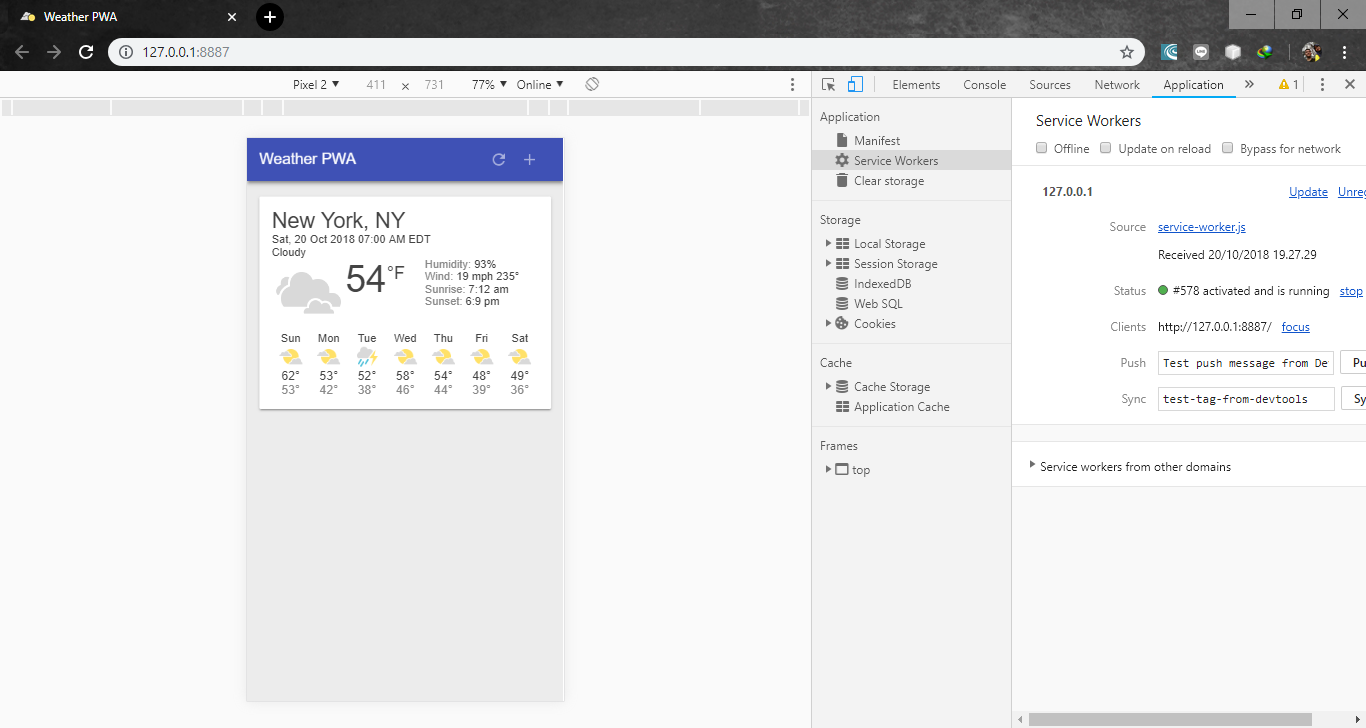
Tambahkan kode berikut setelah komentar tersebut

|  |
| --- |
| if ('serviceWorker' in navigator) {  navigator.serviceWorker  .register('./service-worker.js')  .then(function() { console.log('Service Worker Registered'); });  } |

1. Buatlah file bernama service-worker.js pada folder latihan lalu tambahkan kode berikut.

|  |
| --- |
| var cacheName = 'weatherPWA-step-6-1';  var filesToCache = [];  self.addEventListener('install', function(e) {  console.log('[ServiceWorker] Install');  e.waitUntil(  caches.open(cacheName).then(function(cache) {  console.log('[ServiceWorker] Caching app shell');  return cache.addAll(filesToCache);  })  );  }); |

1. Buka Dev Tools pada Google chrome (Ctrl + Shift + i) saat membuka page latihan. Jika berhasil maka tampilan pada Service Workers akan seperti ini.



1. Tambahkan event listener activate di atas event listener install dalam file service-worker.js

|  |
| --- |
| self.addEventListener('activate', function(e) {  console.log('[ServiceWorker] Activate');  e.waitUntil(  caches.keys().then(function(keyList) {  return Promise.all(keyList.map(function(key) {  if (key !== cacheName) {  console.log('[ServiceWorker] Removing old cache', key);  return caches.delete(key);  }  }));  })  );  return self.clients.claim();  }); |

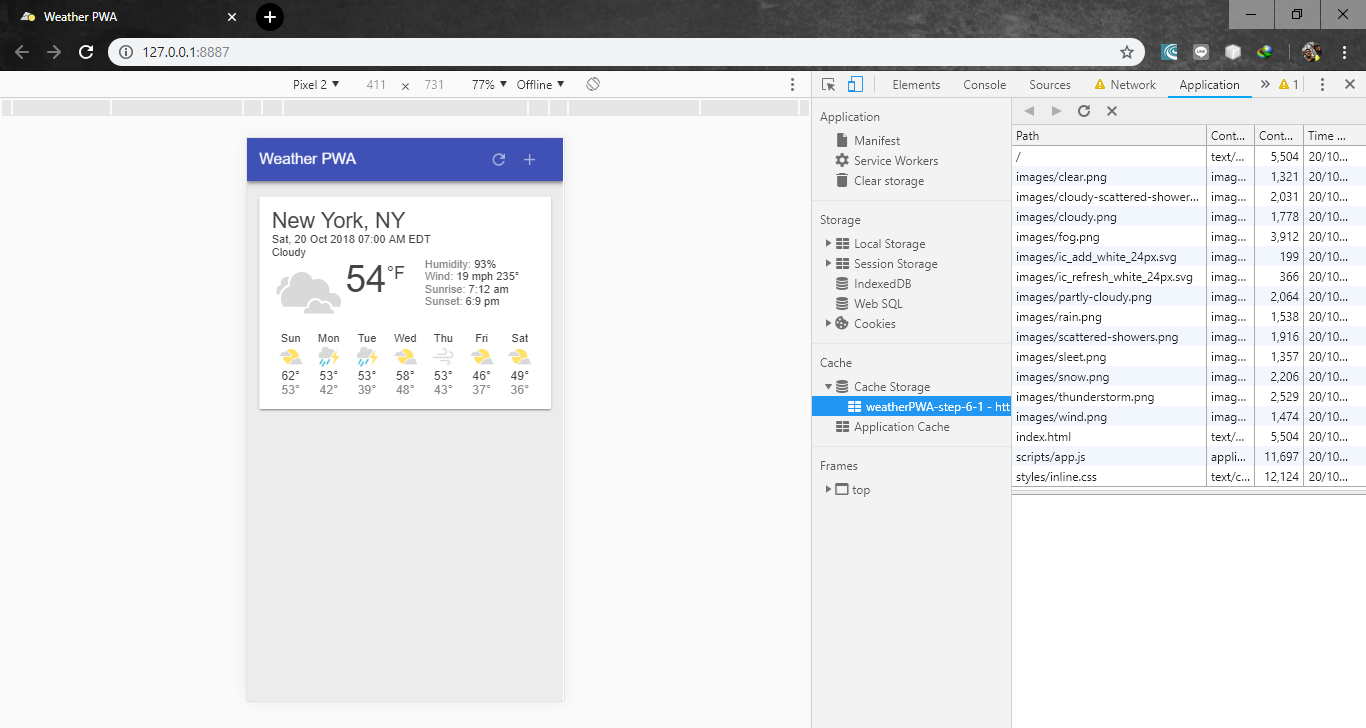
1. Pada bagian atas file service-worker.js, ganti var filesToCache = []; dengan kode di bawah ini:

|  |
| --- |
| var filesToCache = [  '/',  '/index.html',  '/scripts/app.js',  '/styles/inline.css',  '/images/clear.png',  '/images/cloudy-scattered-showers.png',  '/images/cloudy.png',  '/images/fog.png',  '/images/ic\_add\_white\_24px.svg',  '/images/ic\_refresh\_white\_24px.svg',  '/images/partly-cloudy.png',  '/images/rain.png',  '/images/scattered-showers.png',  '/images/sleet.png',  '/images/snow.png',  '/images/thunderstorm.png',  '/images/wind.png'  ]; |

1. Sediakan shell aplikasi dari cache. Tambahkan kode berikut ke bagian bawah file service-worker.js

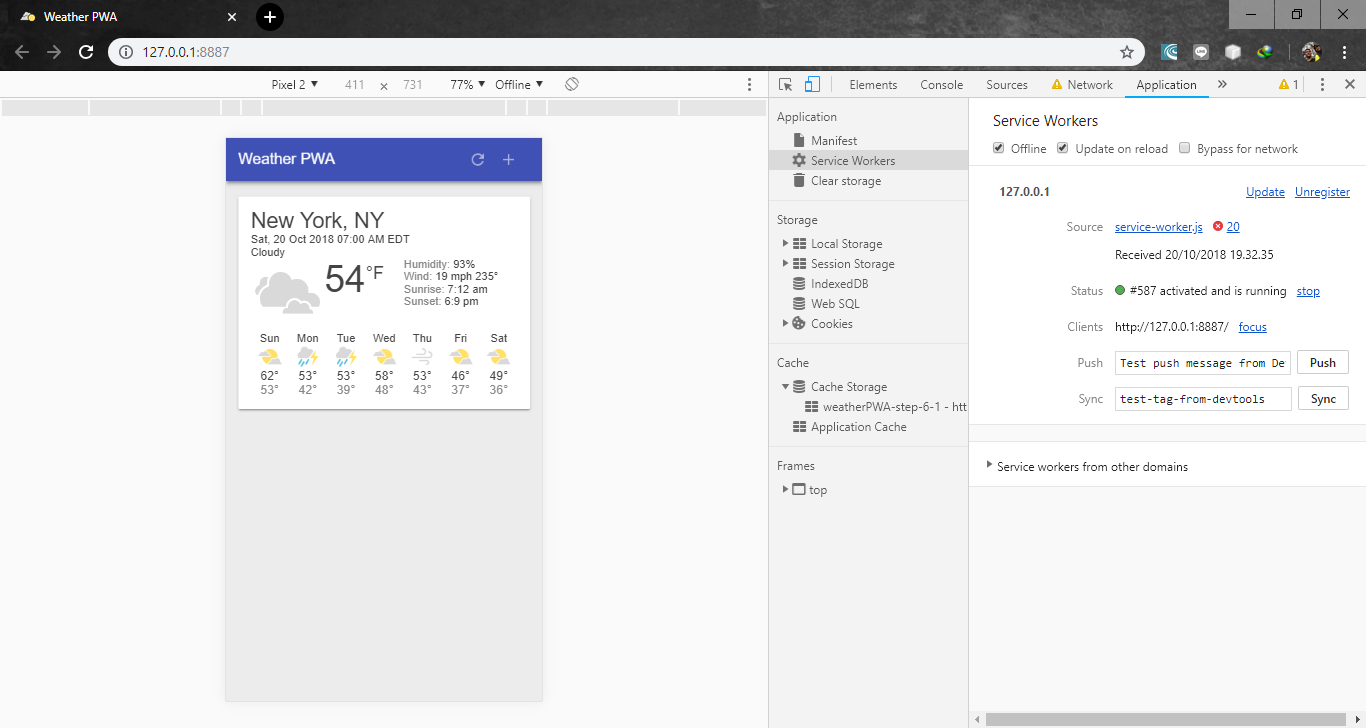
|  |
| --- |
| self.addEventListener('fetch', function(e) {  console.log('[ServiceWorker] Fetch', e.request.url);  e.respondWith(  caches.match(e.request).then(function(response) {  return response || fetch(e.request);  })  );  }); |

1. Muat ulang laman Anda lalu buka panel Cache Storage pada panel Application dari DevTools, lalu lihat apakah tampilan sudah seperti ini.

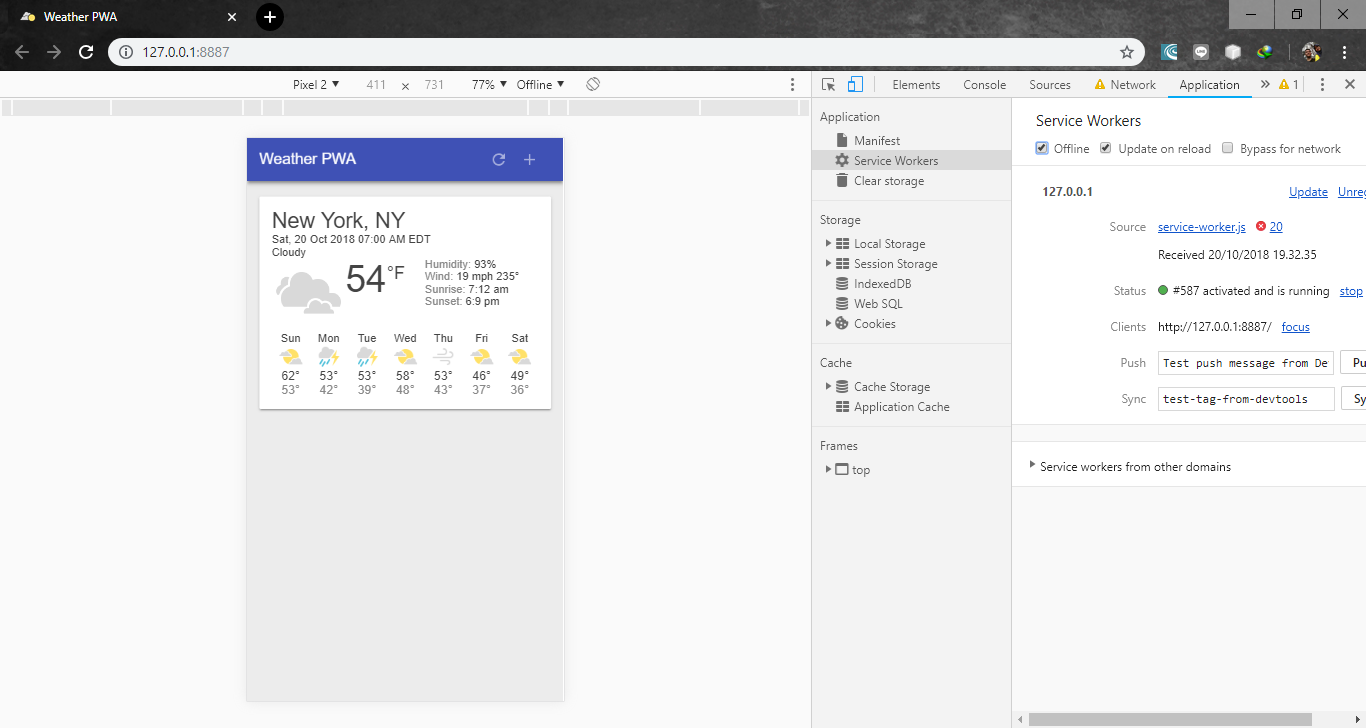


Jika sudah seperti itu maka resource yang digunakan telah berhasil di cache.

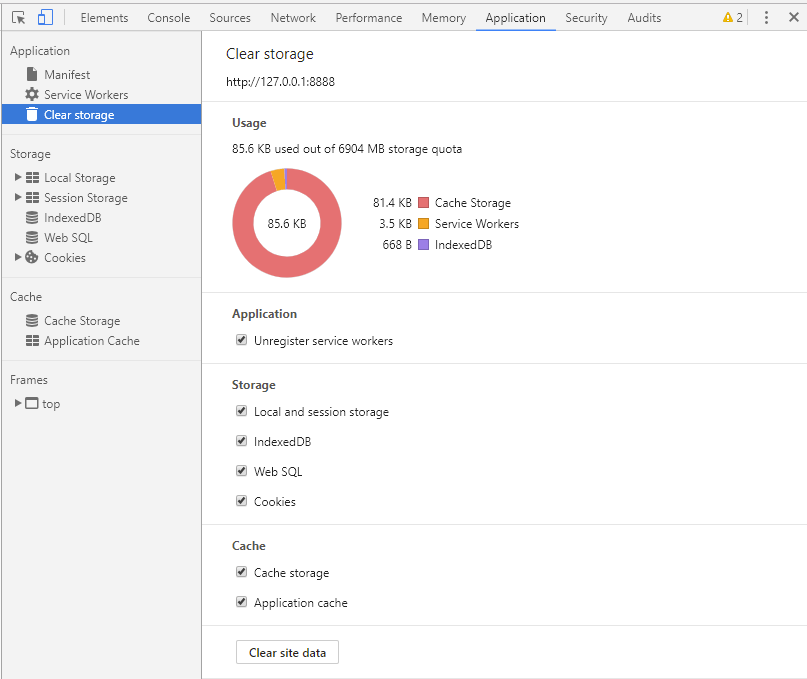
1. Sekarang menguji mode offline. Kembali ke panel Service Worker dari DevTools dan aktifkan kotak centang Offline. Setelah mengaktifkannya, Anda akan melihat ikon peringatan kecil berwarna kuning di sebelah tab panel Network. Ini menunjukkan bahwa kondisi telah offline.



1. Reload halaman dan lihat apakah yang akan ditampilkan dalam keadaan offline.



1. Selanjutnya adalah membuat service worker untuk meng-cache data prakiraan cuaca. Pertama hapus semua data yang tersimpan (localStorage, data indexedDB, file cache) dan membuang service worker, gunakan panel Clear storage di tab Application lalu klik Clear site data.



1. Buka file service-worker.js lalu tambahkan kode berikut ke bagian atas dari file tersebut.

|  |
| --- |
| var dataCacheName = 'weatherData-v1'; |

1. Perbarui if (key !== cacheName) { dari event listener activate sehingga tidak menghapus cache data ketika membersihkan cache shell aplikasi.

|  |
| --- |
| if (key !== cacheName && key !== dataCacheName) { |

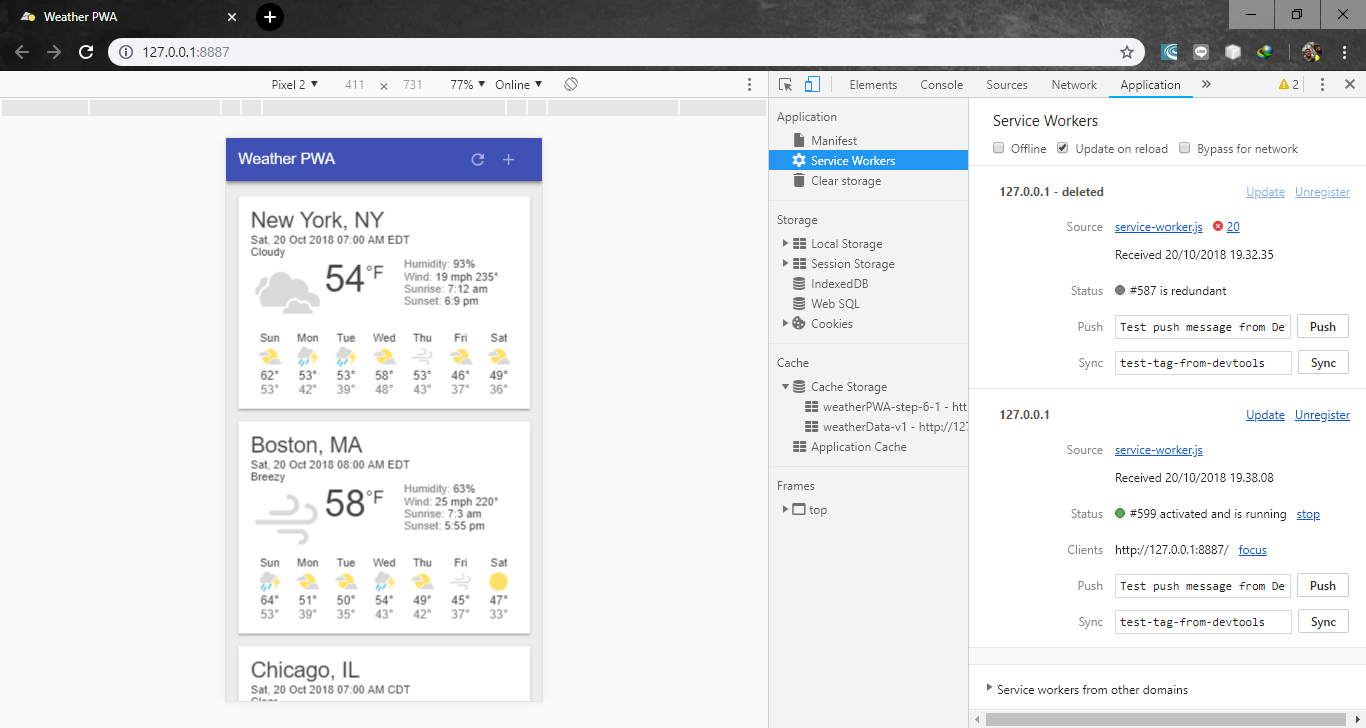
1. Perbarui event listener fetch untuk menangani permintaan ke data API secara terpisah dari permintaan lainnya.

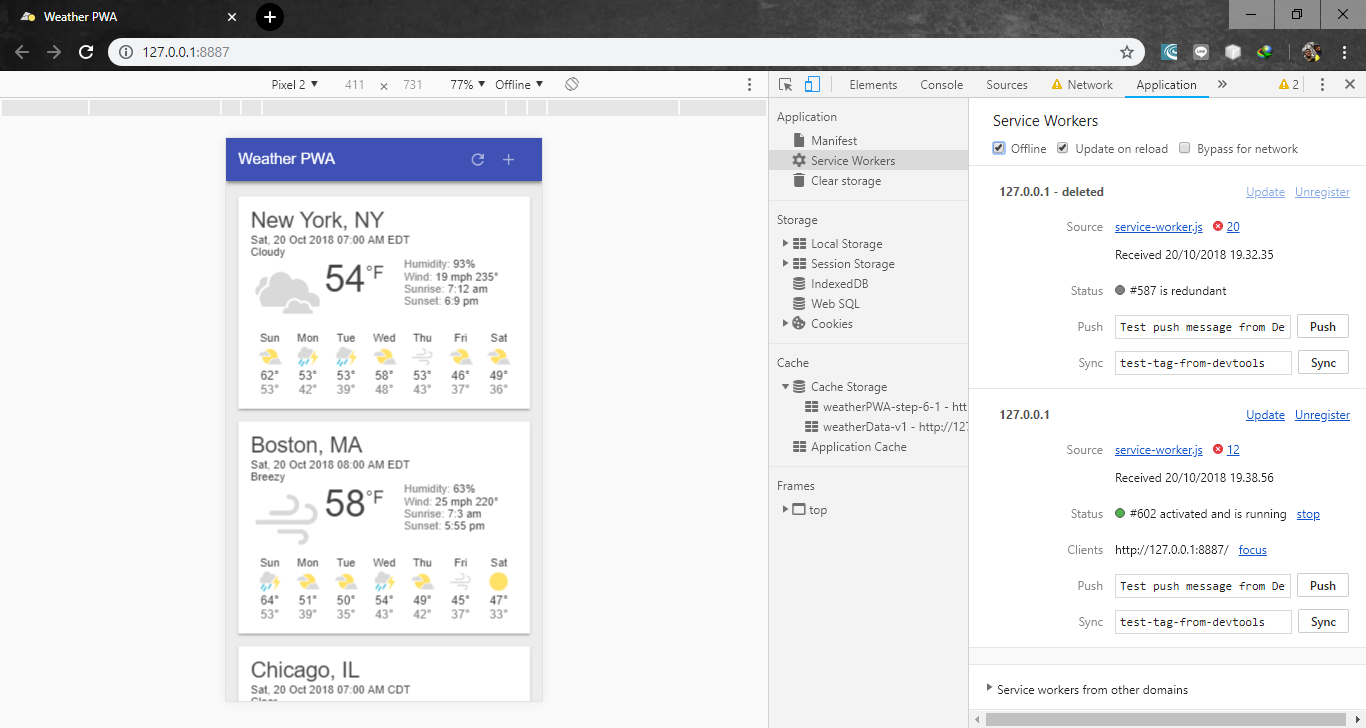
|  |
| --- |
| self.addEventListener('fetch', function(e) {   console.log('[Service Worker] Fetch', e.request.url);   var dataUrl = 'https://query.yahooapis.com/v1/public/yql';   if (e.request.url.indexOf(dataUrl) > -1) {     /\*      \* When the request URL contains dataUrl, the app is asking for fresh      \* weather data. In this case, the service worker always goes to the      \* network and then caches the response. This is called the "Cache then      \* network" strategy:      \*/     e.respondWith(       caches.open(dataCacheName).then(function(cache) {         return fetch(e.request).then(function(response){           cache.put(e.request.url, response.clone());           return response;         });       })     );   } else {     /\*      \* The app is asking for app shell files. In this scenario the app uses the      \* "Cache, falling back to the network" offline strategy:      \*/     e.respondWith(       caches.match(e.request).then(function(response) {         return response || fetch(e.request);       })     );   } }); |

1. Berikutnya buka app.js, kita harus memeriksa apakah objek caches ada dan meminta data terbaru dari situ. Temukan komentar TODO add cache logic here di app.getForecast(), dan kemudian tambahkan kode berikut di bawah komentar tersebut.

|  |
| --- |
| if ('caches' in window) {  /\*  \* Check if the service worker has already cached this city's weather  \* data. If the service worker has the data, then display the cached  \* data while the app fetches the latest data.  \*/  caches.match(url).then(function(response) {  if (response) {  response.json().then(function updateFromCache(json) {  var results = json.query.results;  results.key = key;  results.label = label;  results.created = json.query.created;  app.updateForecastCard(results);  });  }  });  } |

1. Lakukan Pengujian dengan menambahkan kota baru pada prakiraan cuaca, lalu centang Offline pada Service workers di Dev Tools Google Chrome, lalu coba reload halaman tersebut.





1. Buat file bernama manifest.json di folder Latihan dan masukkan kode berikut.

|  |
| --- |
| {   "name": "Weather",   "short\_name": "Weather",   "icons": [{     "src": "images/icons/icon-128x128.png",       "sizes": "128x128",       "type": "image/png"     }, {       "src": "images/icons/icon-144x144.png",       "sizes": "144x144",       "type": "image/png"     }, {       "src": "images/icons/icon-152x152.png",       "sizes": "152x152",       "type": "image/png"     }, {       "src": "images/icons/icon-192x192.png",       "sizes": "192x192",       "type": "image/png"     }, {       "src": "images/icons/icon-256x256.png",       "sizes": "256x256",       "type": "image/png"     }],   "start\_url": "/index.html",   "display": "standalone",   "background\_color": "#3E4EB8",   "theme\_color": "#2F3BA2" } |

1. Sekarang tambahkan baris berikut ke bagian bawah elemen <head> dalam file index.html

|  |
| --- |
| <link rel="manifest" href="/manifest.json"> |

1. Run aplikasi pada google chrome lalu buka Developer tools (Ctrl + Shift + i) lalu klik tab application dan klik manifest, pada bagian identity klik Add to homescreen lalu reload page, maka akan muncul pop up pada bagian atas google chrome, klik Add pada pop up tersebut dan lihat hasilnya.

