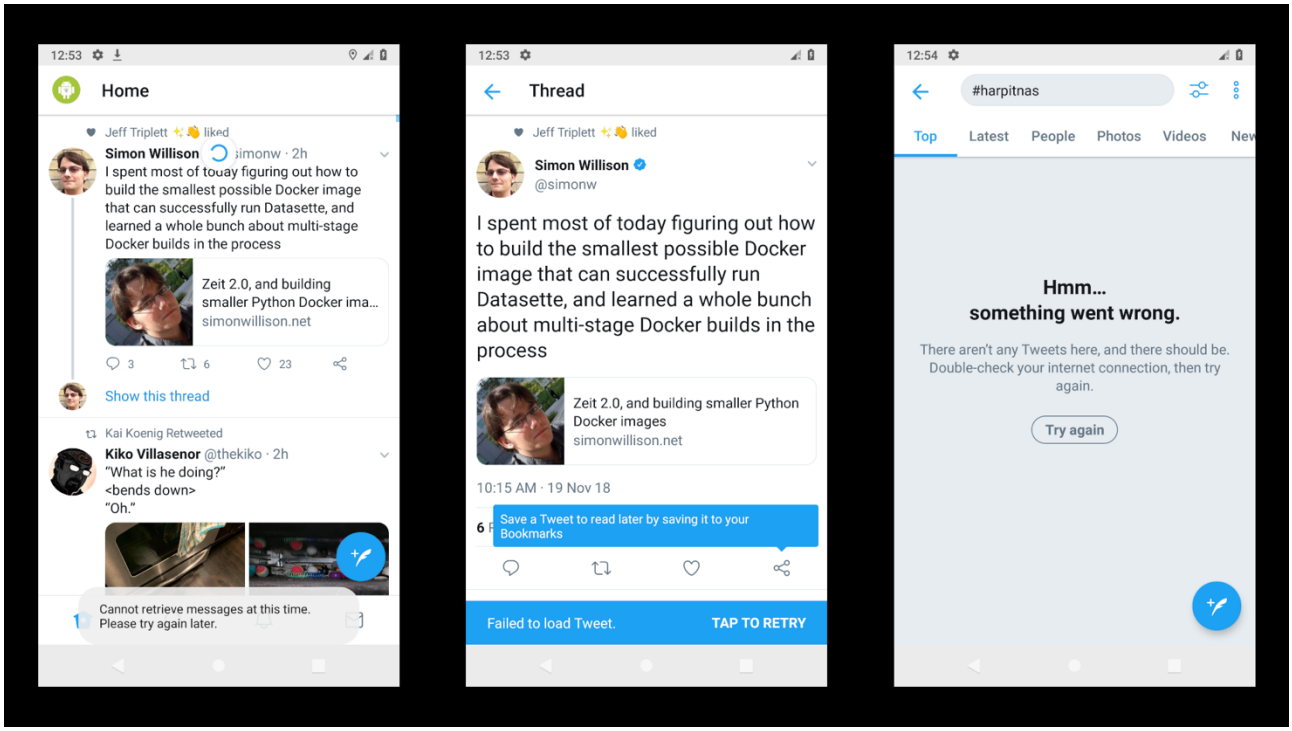
Modul 3 : Application Shell

Saat menggunakan antarmuka web tradisional, masing-masing halaman akan memiliki file HTML-nya sendiri. Dengan demikian, saat akan pindah dari satu halaman ke lainnya, ada jeda waktu di mana pengguna harus menunggu sampai halaman tersebut termuat dan menampilkan sesuatu. Hal ini berbeda dengan perilaku aplikasi *native* pada umumnya.

Contoh, saat menggunakan aplikasi Twitter di perangkat *mobile* tanpa koneksi, tidak bisa melihat konten linimasa karena tidak ada koneksi internet untuk mengunduh data-data tersebut, akan tetapi masih bisa melihat beberapa komponen-komponen UI-nya. Hal ini menunjukkan bahwa aplikasi *native* hanya mengunduh datanya saja dari server, sedangkan komponen UI lainnya sudah terpasang sejak awal dan akan langsung muncul saat aplikasi dibuka.

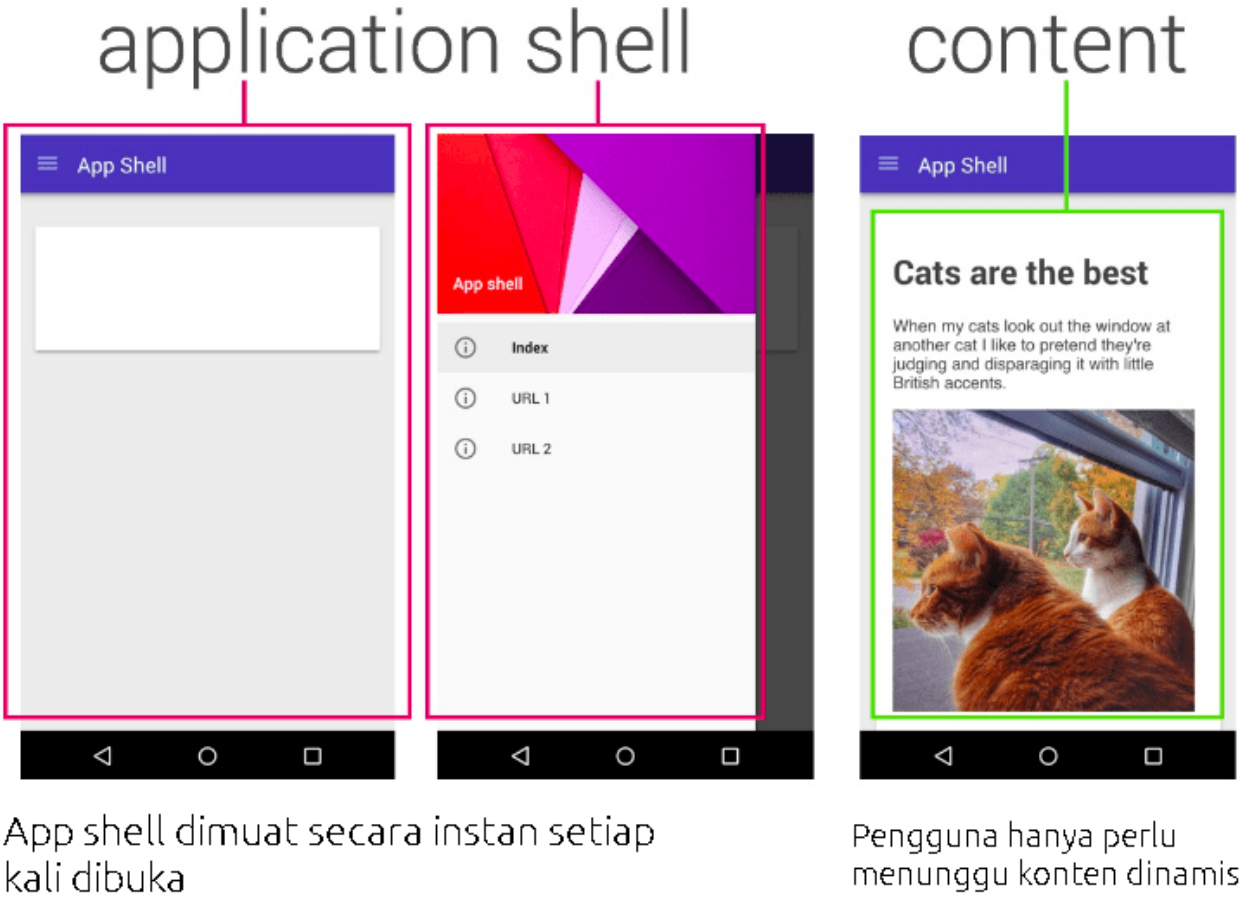


Salah satu tujuan PWA adalah memberikan pengalaman yang semirip mungkin dengan menggunakan aplikasi native. Apabila PWA masih harus memuat masing-masing halaman sebelum ditampilkan kepada user, artinya belum memberikan sebuah pengalaman yang mirip dengan aplikasi native. Dengan menggunakan teknik PWA, dapat menyimpan komponen UI aplikasi pada perangkat, sehingga aplikasi dapat langsung menampilkan antarmuka sebelum menampilkan data dari server. Teknik ini bernama Application Shell.

Apa itu Application Shell

Application shell atau disingkat app shell adalah salah satu teknik dalam Progressive Web App untuk membuat bagian antarmuka yang akan termuat secara instan tanpa harus menunggu respon dari server. App shell hanya terdiri dari file HTML, CSS, dan javaScript yang minimalis dan disimpan di dalam cache sehingga aplikasi dapat bekerja secara offline. Karena disimpan di

dalam cache, maka pengguna tidak perlu meminta app shell ke server berulang kali setiap membuka aplikasi.



App shell hanyalah sebuah teknik, bukan library ataupun framework. Jika sudah pernah membangun sebuah *single-page app* (SPA), sesungguhnya sudah menerapkan sesuatu yang sangat mirip dengan app shell. Kita dapat membuat app shell dengan atau tanpa framework (*framework agnostic*).

Manfaat Menggunakan App Shell

Ada banyak keuntungan menggunakan app shell dalam membangun sebuah website. Beberapa di antaranya sebagai berikut:

- **Memiliki performa yang bisa diandalkan dan sangat cepat.** Karena aset statis (HTML, CSS, JavaScript, dan gambar) untuk antarmuka di simpan di dalam cache saat kunjungan pertama, maka pengguna yang kembali di sesi berikutnya dapat memuatnya secara instan.
- **Pengalaman seperti aplikasi native.** Dengan menerapkan teknik app shell, kita bisa memberikan aplikasi web dengan antarmuka, sistem navigasi hingga dukungan offline yang sangat mirip dengan aplikasi native.
- **Lebih hemat.** Meski secara umum harga paket internet sudah cukup terjangkau, namun masih ada daerah-daerah dimana paket internet masih berharga mahal dengan kecepatan yang masih lambat. Dengan menggunakan app shell kita bisa membantu

pengguna dalam menghemat penggunaan data karena tidak semua data harus diunduh berulang kali.

Kapan Menggunakan App Shell

App shell cocok diterapkan pada hampir semua skenario pembuatan website atau aplikasi berbasis web. Umumnya sebuah website pasti memiliki komponen yang sama di setiap halamannya, seperti header, footer, atau sidebar. Komponen-komponen yang berulang ini dapat disimpan di dalam cache browser agar pada saat pengguna mengakses halaman lain, browser hanya perlu menerima respon konten yang terkait halaman itu saja.

Hanya saja untuk menerapkan App Shell, perlu mengubah sedikit paradigma pengembangan aplikasi web. Dari yang biasanya dibuat dalam bentuk halaman per halaman atau *multi page app* (MPA), menjadi basis satu halaman atau single page app (SPA). Adapun permintaan data konten halaman, dilakukan secara asinkron atau yang dikenal dengan istilah AJAX (Asynchronous JavaScript and XML-HTTP). Dengan menggunakan teknik AJAX, pergantian konten halaman tak lagi perlu pemuatan ulang seluruh halaman.

Syarat Menggunakan App Shell

Beberapa hal yang harus dipenuhi untuk menggunakan teknik App Shell diantaranya:

- Menggunakan Service Worker untuk mem-*bypass* jalur request. Pada aplikasi web standar, setiap request asset akan langsung diarahkan ke jaringan internet oleh browser. Dengan menggunakan Service Worker, kita dapat memilah request mana yang perlu diarahkan ke jaringan dan request mana yang cukup diakomodir menggunakan aset yang sudah disimpan di browser cache. Selengkapnya tentang Service Worker dapat kamu temukan di modul 3.
- Menyimpan aset utama untuk app shell di cache. Agar aplikasi tidak perlu lagi mengirim request untuk aset yang sama, terlebih untuk digunakan pada app shell, kita perlu menyimpannya di cache browser menggunakan Cache API. Selengkapnya tentang Cache API dapat kamu temukan di modul 4.
- Menggunakan AJAX request untuk mengambil data. Seperti yang sudah dijelaskan sebelumnya, kita perlu mengirim request data menggunakan AJAX dan hasil responnya kita pasang menggunakan JavaScript pada elemen yang telah kita siapkan untuk menyimpan konten dinamis. Kamu dapat menggunakan Fetch API, method XMLHttpRequest atau menggunakan method ajax bawaan library JavaScript seperti `$.ajax()` pada jQuery.

akan dibuat sebuah aplikasi web progresif berisi halaman-halaman yang dapat diakses secara statis. dsiapkan terlebih dahulu aplikasi web standar yang menampilkan konten halaman berbasis AJAX, dikenal juga dengan istilah *single page application* atau SPA.

Menyiapkan Aset

Pertama-tama unduh terlebih dahulu framework Materialize CSS yang akan kita pakai untuk mempermudah pembuatan app shell di sini. Ekstrak berkas yang telah diunduh ke dalam folder first-pwa/. Struktur berkas dari aplikasi kita harus seperti ini:

```
first-pwa/
├── css/
│   ├── materialize.css
│   └── materialize.min.css
├── js/
│   ├── materialize.js
│   └── materialize.min.js
├── LICENSE
├── README.md
```

Selanjutnya, buat sebuah berkas pada folder project dengan nama `index.html` untuk menyimpan template app shell:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"/>
  <title>My First PWA</title>
  <meta name="description" content="My first PWA"/>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="css/materialize.min.css">
</head>
 <body>

   <!-- Navigasi -->
   <nav class="teal lighten-1" role="navigation">
     <div class="nav-wrapper container">
       <a href="#" class="brand-logo" id="logo-container">My App</a>
       <a href="#" class="sidenav-trigger" data-target="nav-mobile">☰</a
  >

       <ul class="topnav right hide-on-med-and-down"></ul>
       <ul class="sidenav" id="nav-mobile"></ul>
     </div>
   </nav>
   <!-- Akhir Navigasi -->

   <div class="container" id="body-content"></div>

   <script src="js/materialize.min.js"></script>
   <script src="js/nav.js"></script>
 </body>
 </html>
```

Pada berkas index.html di atas, buat elemen navigation dengan menggunakan tag `<nav>` dan elemen konten dengan id `#body-content`. Di dalam navigation terdapat `.brand-logo` dan juga dua buah elemen menu yaitu `.topnav` untuk menampilkan menu atas dan `.sidenav` untuk menampilkan menu samping. Menu samping hanya akan muncul pada mode layar mobile dan menu atas hanya akan tampil pada mode layar besar. Biarkan kedua elemen menu tersebut dan juga elemen `#body-content` kosong pada template. Kemudian buat berkas baru di dalam folder js/ dengan nama nav.js. Script ini akan berisi semua kode JavaScript kita. Tuliskan kode berikut di dalam berkas tersebut:

```
document.addEventListener("DOMContentLoaded", function() {
  // Activate sidebar nav
  var elems = document.querySelectorAll(".sidenav");
  M.Sidenav.init(elems);
```

```
5.    loadNav();
6.
7.    function loadNav() {
8.      var xhttp = new XMLHttpRequest();
9.      xhttp.onreadystatechange = function() {
10.         if (this.readyState == 4) {
11.           if (this.status != 200) return;
12.
13.           // Muat daftar tautan menu
14.           document.querySelectorAll(".topnav, .sidenav").forEach(function(
    elm) {
15.             elm.innerHTML = xhttp.responseText;
16.           });
17.         }
18.       };
19.       xhttp.open("GET", "nav.html", true);
20.       xhttp.send();
21.     }
22.   });
```

Pada kode di atas mengaktifkan elemen sidebar bawaan framework Materialize agar dapat muncul saat burger menu diklik (baris 3-4). juga memanggil fungsi `loadNav()` yang berisi kode AJAX menggunakan method XMLHttpRequest untuk mengambil isi dari berkas nav.html dan menyimpannya ke dalam elemen menu `.topnav` dan `.sidenav`.

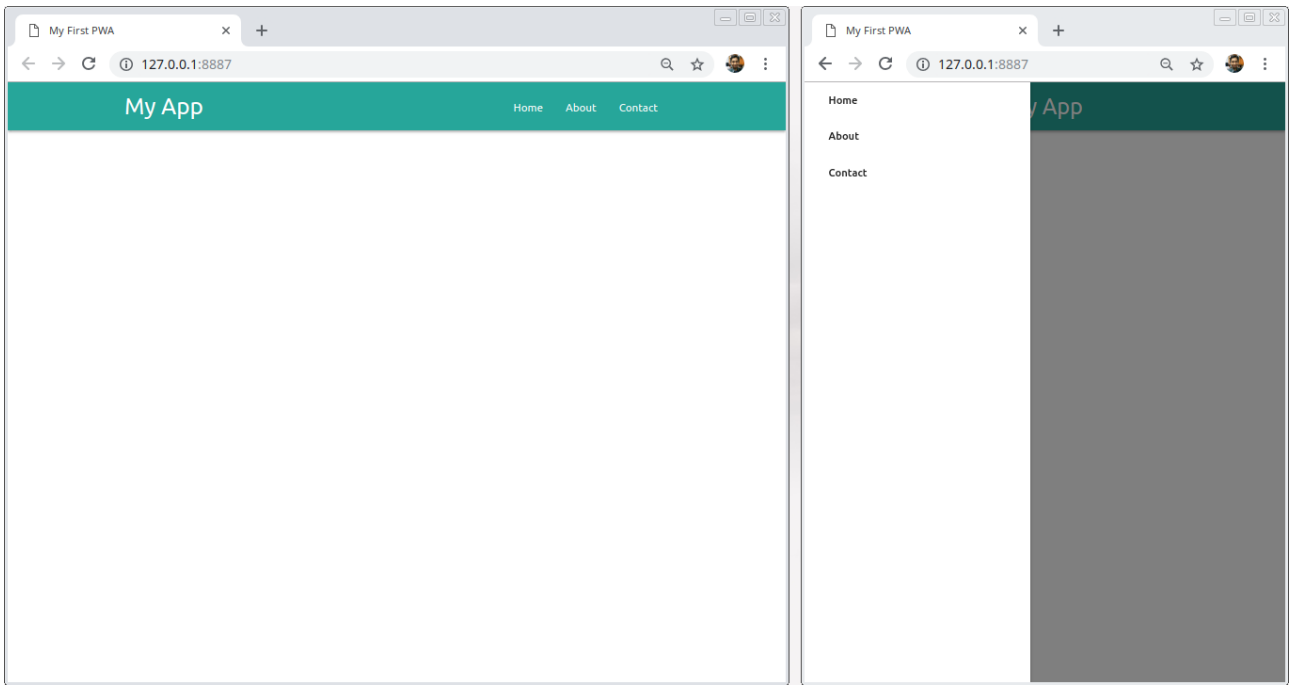
Perhatian: Semua kode JavaScript kita simpan di dalam blok kode dari fungsi callback untuk event `DOMContentLoaded`untuk menjamin kode dijalankan setelah semua elemen pada dokumen html selesai dimuat.

Sebelum mencoba di browser, siapkan satu berkas lagi yakni nav.html yang berisi daftar menu untuk sidenav dan topnav. Buat berkas nav.html di dalam folder project berdampingan dengan file index.html:

```
1. <li><a class="waves-effect" href="#home">Home</a></li>
2. <li><a class="waves-effect" href="#about">About</a></li>
3. <li><a class="waves-effect" href="#contact">Contact</a></li>
```

Sekarang buka aplikasi di browser. Tapi ingat. Aplikasi menggunakan method XMLHttpRequest untuk mengambil data menggunakan AJAX, yang mana tidak akan berjalan menggunakan protokol file://. Jadi bila Anda hanya membuka langsung berkas index.html ke browser (dengan klik dua kali file index.html atau men-drag file tersebut ke browser), aplikasi akan gagal menampilkan daftar menu di sidenav dan di topnav. Anda harus menjalankan aplikasi menggunakan protokol http seperti pada panduan di awal.

Bila berhasil membuka aplikasi di browser dengan benar, akan nampak sebuah elemen bar berisi judul aplikasi. juga dapat mencoba mengecilkan ukuran browser untuk mensimulasikan ukuran layar mobile sehingga muncul tombol *burger menu* di kiri atas yang bila diklik akan memunculkan sidebar.



Menampilkan Konten Halaman

setelah sudah berhasil menampilkan app shell berikut navigasi menu. lengkapi terlebih dahulu aplikasi dengan menampilkan konten halaman. Konten akan dipanggil secara asinkron atau menggunakan teknik AJAX dan disimpan ke dalam elemen `.body-content`. Tambahkan kode berikut di dalam file nav.js (tuliskan di dalam callback event `DOMContentLoaded`):

```
1. // Load page content
2. var page = window.location.hash.substr(1);
3. if (page == "") page = "home";
4. loadPage(page);
5.
6. function loadPage(page) {
7.   var xhttp = new XMLHttpRequest();
8.   xhttp.onreadystatechange = function() {
9.     if (this.readyState == 4) {
10.        var content = document.querySelector("#body-content");
11.        if (this.status == 200) {
12.          content.innerHTML = xhttp.responseText;
13.        } else if (this.status == 404) {
14.          content.innerHTML = "<p>Halaman tidak ditemukan.</p>";
15.        } else {
16.          content.innerHTML = "<p>Ups.. halaman tidak dapat diakses.</p>";
17.        }
18.      }
19.    };
20.    xhttp.open("GET", "pages/" + page + ".html", true);
21.    xhttp.send();
22.  }
```

Pada kode di atas, ambil terlebih dahulu hash dari url halaman sebagai nama halaman yang akan dipanggil. Misalnya kita buka url http://127.0.0.1:8887/#contact berarti kita mengakses halaman contact. Bila hash tidak ditemukan berarti halaman default yakni home yang akan

ditampilkan. Kita memanggil fungsi `loadPage()` untuk memanggil konten dari berkas halaman menggunakan AJAX. Konten halaman kemudian akan disimpan ke dalam elemen `.body-content`. Bila halaman tidak ditemukan (status code 404) maka kita tampilkan elemen paragraf yang memberitahukan bahwa halaman yang dimaksud tidak ada pada daftar halaman yang tersedia. Berkas-berkas konten halaman disimpan di dalam folder pages/.

Pada file nav.html sebelumnya kita telah menambahkan tiga buah tautan untuk tiga buah halaman, diantaranya home, about dan contact. Berarti kita harus menyiapkan tiga berkas halaman tersebut. Buatlah tiga buah berkas berikut di dalam folder pages/.

pages/home.html

```
1. <h2>Selamat datang!</h2>
2. <p>Selamat datang di laman aplikasi web progresif pertama saya.</p>
```

pages/about.html

```
1. <h2>Tentang Saya</h2>
2. <p>Saya adalah programmer masa depan.</p>
```

pages/contact.html

```
1. <h2>Hubungi Saya</h2>
2. <p>Hubungi saya melalui email saya@mail.com</p>
```

Sekarang, bila coba muat ulang halaman http://127.0.0.1:8887/ akan muncul konten halaman home. Begitu pula bila buka:

http://127.0.0.1:8887/#about

http://127.0.0.1:8887/#contact

akan muncul konten halaman sesuai dengan halaman yang dipilih. Hanya saja bila klik menu, konten halaman belum langsung berubah. harus membuat event listener click terlebih dahulu untuk setiap link menu.

Menambahkan Event Listener untuk Klik Menu

Modifikasi fungsi `loadNav()` dengan menambahkan kode event listener untuk setiap tautan pada menu menjadi seperti ini:

```
1. function loadNav() {
2.   var xhttp = new XMLHttpRequest();
3.   xhttp.onreadystatechange = function() {
4.     if (this.readyState == 4) {
5.       if (this.status != 200) return;
6.
7.       // Muat daftar tautan menu
8.       document.querySelectorAll(".topnav, .sidenav").forEach(function(elm)
   {
9.         elm.innerHTML = xhttp.responseText;
10.       });
11.
12.       // Daftarkan event listener untuk setiap tautan menu
```

```
13.         document.querySelectorAll(".sidenav a, .topnav a").forEach(functio
    n(elm) {
14.           elm.addEventListener("click", function(event) {
15.             // Tutup sidenav
16.             var sidenav = document.querySelector(".sidenav");
17.             M.Sidenav.getInstance(sidenav).close();
18.
19.             // Muat konten halaman yang dipanggil
20.             page = event.target.getAttribute("href").substr(1);
21.             loadPage(page);
22.           });
23.         });
24.       }
25.     };
26.     xhttp.open("GET", "nav.html", true);
27.     xhttp.send();
28.   }
29.
```

Muat ulang halaman dan sekarang mestinya konten halaman akan langsung berubah ketika menu diklik.