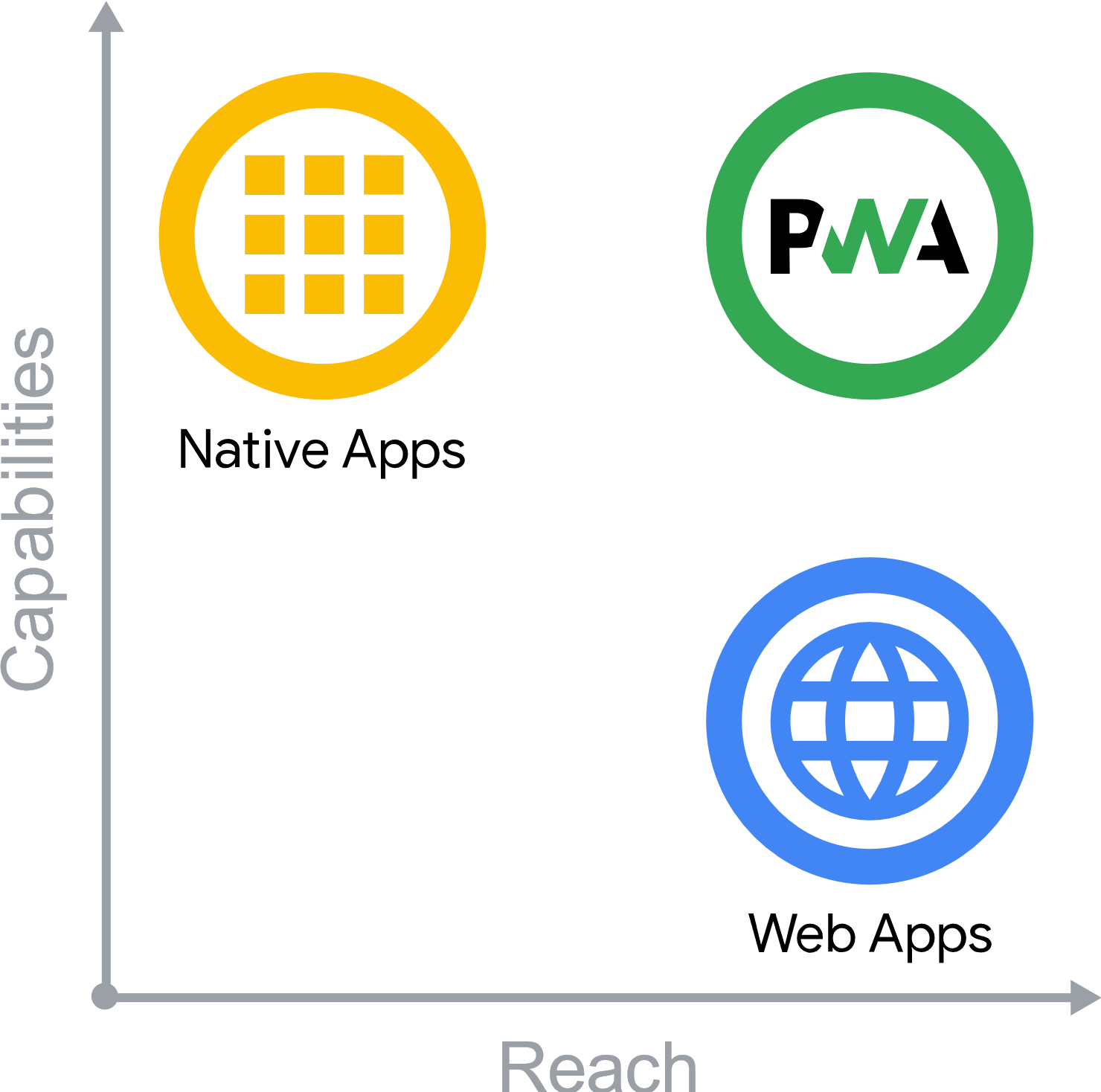
**Rangkuman PWA**

PWA merupakan aplikasi hybrid yang terinspirasi dari kemudahan penggunaan browser di berbagai platform dan kenyamanan menggunakan aplikasi mobile di ponsel atau tablet.



Dari gambar diatas, kita dapat mengetahui bahwa aplikasi *native* memiliki kemampuan yang lebih baik daripada *web application*. Akan tetapi, *web application* memiliki jaungkauan yang lebih luas dibanding *web application*.

**3 Pilar PWA**

1. **Mampu**

Saat ini web cukup mampu untuk berdiri sendiri. Contohya, kita dapat membuat aplikasi obrolan video *hiper-lokal* menggunakan WebRTC, geolokasi, dan pemberitahuan *push*. Kita dapat membuat aplikasi itu dapat diinstal dan menjadikan percakapan itu virtual dengan WebGL dan WebVR. Dengan diperkenalkannya Perakitan Web, pengembang dapat memanfaatkan ekosistem lain, seperti C, C ++, dan Rust, serta menghadirkan kerja dan kemampuan puluhan tahun ke web juga.

Hingga saat ini, hanya aplikasi khusus *platform* yang benar-benar dapat mengklaim kemampuan ini. Meskipun beberapa kemampuan masih di luar jangkauan web, akan ada API baru dan yang akan datang akan mengubahnya, memperluas apa yang dapat dilakukan web dengan fitur seperti akses sistem *file*, kontrol media, lencana aplikasi, dan dukungan papan klip penuh.

*Progressive web application* (PWA) menghadirkan fasilitas & pengalaman *user* yang sama dengan ketika *user* menggunakan web browser / app di *mobile*.

1. **Andal**

PWA yang andal terasa cepat dan dapat diandalkan pada kondisi jaringan apapun.

Kecepatan adalah hal yang penting untuk pegalaman pengguna. Performa memengaruhi seluruh pengalaman pengguna, mulai dari cara mereka memandang aplikasi Anda hingga performa sebenarnya.

Terakhir, aplikasi yang andal harus dapat digunakan terlepas dari koneksi jaringan. Pengguna mengharapkan aplikasi memulai pada koneksi jaringan yang lambat atau tidak stabil atau bahkan saat *offline*. Mereka mengharapkan konten terbaru yang berinteraksi dengan mereka, seperti trek media atau tiket dan rencana perjalanan, tersedia dan dapat digunakan meskipun sulit mendapatkan permintaan ke server Anda.

1. **Dapat dipasang**

PWA dapat dipasang di layar beranda, bukan hanya di *tab* browser.

Saat PWA berpindah dari *tab* dan masuk ke jendela aplikasi mandiri, cara ini mengubah cara pengguna berinteraksi dengannya.

Aplikasi Web Progresif memungkinkan apa yang Anda buat untuk dipasang oleh *siapa pun, di mana pun, di perangkat apa pun* dengan satu basis kode.

**Kelebihan PWA**

Google menyebut kelebihan PWA dengan istilah **FIRE** (*Fast, Integrated, Reliable, and Engaging*). Cepat, Terintegrasi, Bisa Diandalkan dan Menarik.

1. Kemudahan Digunakan di Berbagai Perangkat,

Membangun PWA memungkinkan Anda untuk merancang tampilan aplikasi yang seragam di berbagai perangkat dengan berbagai ukuran layar baik ponsel maupun tablet.

1. Tidak Bergantung App Store,

Tidak lagi bergantung pada AppStore seperti Google Play Store, Apple Store, dan Windows Store ketika akan mempublikasikan aplikasi Anda.

1. Instalasi Cepat,

Cukup menambahkan website Anda ke **HomeScreen.** Maka, aplikasi akan terpasang otomatis.

1. Berfungsil layaknya Aplikasi Mobile,

PWA menghadirkan pengalaman seperti aplikasi mobile.

1. Kinerja Apik

Sebuah PWA memiliki *service workers* yang bertugas mengelola *cache* aset website dengan efisien. Hasilnya, aplikasi berjalan dengan cepat sehingga meningkatkan kenyamaman pengguna.

1. Update Otomatis

PWA melakukan update secara otomatis tiap kali pengguna mengakses aplikasi.

1. Bisa Berjalan Tanpa Jaringan Internet

Dengan *service workers,* semua data yang sudah di-*load* akan disimpan di *cache*.

1. Ramah *Storage*

*Progressive web app* berukuran sangat kecil jika dibandingkan dengan aplikasi *mobile*.

1. Ramah Bandwidth

Interaksi ketika menggunakan PWA pun ramah *bandwidth* karena hadirnya optimasi gambar yang mampu memangkas kebutuhan data hingga 70%.

1. Keamanan yang Terjamin

Tingkat keamanan pada PWA lebih tinggi berkat protokol jaringan[HTTPS](https://www.niagahoster.co.id/blog/https-adalah-protokol-versi-aman/) yang digunakan.

**Service Workers**

*Service Worker* adalah script yang dijalankan di balik layar. Hadirnya Service Worker memungkinkan adanya fitur yang berjalan tanpa interaksi pengguna seperti *push notifications*, *auto synchronize*, dan lainnya.

Service Workers dan Javascript adalah hal yang sama namun juga berbeda.

Ada beberapa perilaku dari *service workers* yang berbeda dari *script* javascript biasa.

1. **Layer baru**

Biasanya saat membangun situs, Anda hanya memiliki dua lapisan untuk dipikirkan: klien dan server. *Service workers* adalah lapisan baru yang berada di tengah.



*Service workers* *memperluas* browser untuk situs Anda dengan lapisan tengah yang kuat. Lapisan ini ( *service workers* ) dapat mencegat dan menangani semua permintaan yang dibuat situs Anda.

Lapisan pekerja layanan memiliki siklus hidupnya sendiri yang tidak bergantung pada tab browser.

Bayangkan pekerja layanan Anda sebagai lapisan tengah baru dengan siklus proses dan metode pembaruannya sendiri.

1. **Kuat, tapi terbatas**

Memiliki *service workers* di situs Anda memberi Anda manfaat luar biasa. Situs Anda dapat:

* bekerja dengan sempurna bahkan saat pengguna sedang *offline*
* dapatkan peningkatan kinerja besar-besaran melalui caching
* gunakan pemberitahuan push
* diinstal sebagai PWA

Sebanyak yang bisa dilakukan *service workers*, mereka dibatasi oleh desain. *Servie workers* tidak dapat melakukan apa pun yang sinkron atau di bagian yang sama dengan situs Anda. Jadi itu berarti tidak ada akses ke:

* penyimpanan lokal
* DOM
* jendela

Pikirkan *service workers* Anda sebagai sesuatu yang berada di luar laman Anda. Anda dapat berbicara dengannya, tetapi tidak dapat mengakses halaman Anda secara langsung.

1. **Umur panjang, tapi berumur pendek**

*Service workers* terus hidup bahkan setelah pengguna meninggalkan situs Anda atau menutup *tab*. Browser mempertahankan *service workers* ini sehingga akan siap saat pengguna kembali ke situs Anda lagi. Sebelum permintaan pertama dibuat, *service workers* mendapat kesempatan untuk mencegatnya dan mengontrol halaman. Inilah yang memungkinkan situs bekerja secara offline — *service workers* dapat menyajikan versi *cache* dari laman itu sendiri, meskipun pengguna tidak memiliki koneksi ke internet.

1. **Berhenti**

Meskipun *service workers* tampak abadi, mereka dapat dihentikan hampir setiap saat. Browser tidak ingin membuang *resource* pada *service worker* yang saat ini tidak melakukan apa pun. Dihentikan tidak sama dengan *dieksekusi* — *service workers* tetap terpasang dan aktif. Ini hanya untuk ditidurkan. Saat berikutnya diperlukan (misalnya, untuk menangani permintaan), browser membangunkannya kembali.

1. **waitUntill**

Karena kemungkinan terus-menerus tertidur, *service worker* Anda memerlukan cara untuk memberi tahu browser saat sedang melakukan sesuatu yang penting. Di sinilah **event.waitUntil()**berperan. Metode ini memperpanjang siklus proses yang digunakan, menjaganya agar tidak dihentikan dan dilanjutkan ke fase siklus proses berikutnya hingga kita siap. Ini memberi kita waktu untuk mengatur *cache*, mengambil sumber daya dari jaringan, dll.

Pikirkan pekerja layanan Anda sebagai anjing [whippet](https://www.akc.org/dog-breeds/whippet/) . Cepat, setia, dan luar biasa. Itu akan tetap ada di sisi Anda apa pun yang terjadi. Tapi kebanyakan hanya ingin tidur. Sepanjang waktu. Anda harus memberi tahu saat Anda menginginkannya tetap terjaga.

1. **Bersama – sama, tapi terpisah**

Halaman Anda hanya dapat *dikontrol* oleh satu *service worker* dalam satu waktu. Tapi bisa *dipasang* dua *service worker* sekaligus. Saat Anda membuat perubahan pada kode pekerja layanan dan menyegarkan laman, Anda sebenarnya tidak mengedit pekerja layanan Anda sama sekali. Pekerja layanan tidak *berubah* . Anda malah membuat yang baru. Pekerja layanan baru ini (sebut saja SW2) akan dipasang tetapi belum dapat diaktifkan . Itu harus menunggu pekerja layanan saat ini (SW1) berhenti (ketika pengguna Anda meninggalkan situs Anda).

Setelah Anda *service workers*, Anda akan dapat menciptakan pengalaman yang luar biasa bagi pengguna Anda.