INTRODUCCIÓN A LAS

PROGRESSIVE WEB APPS

GABRIEL PERALES

FULL STACK JAVASCRIPT DEVELOPER

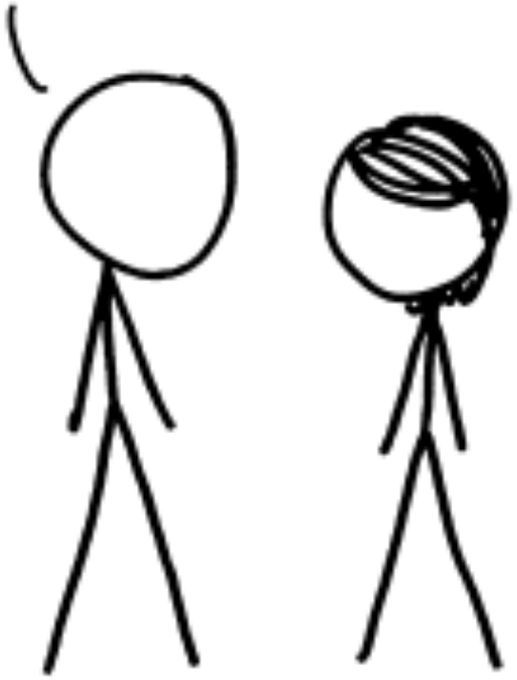
github.com/gabrielperales

twitter: @g_perales

INTRODUCCIÓN
¿QUE ES UNA PWA?

INSTALLING THINGS HAS GOTTEN SO FAST AND PAINLESS.

WHY NOT SKIP IT ENTIRELY, AND MAKE A PHONE THAT HAS EVERY APP "INSTALLED" ALREADY AND JUST DOWNLOADS AND RUNS THEM ON THE FLY?



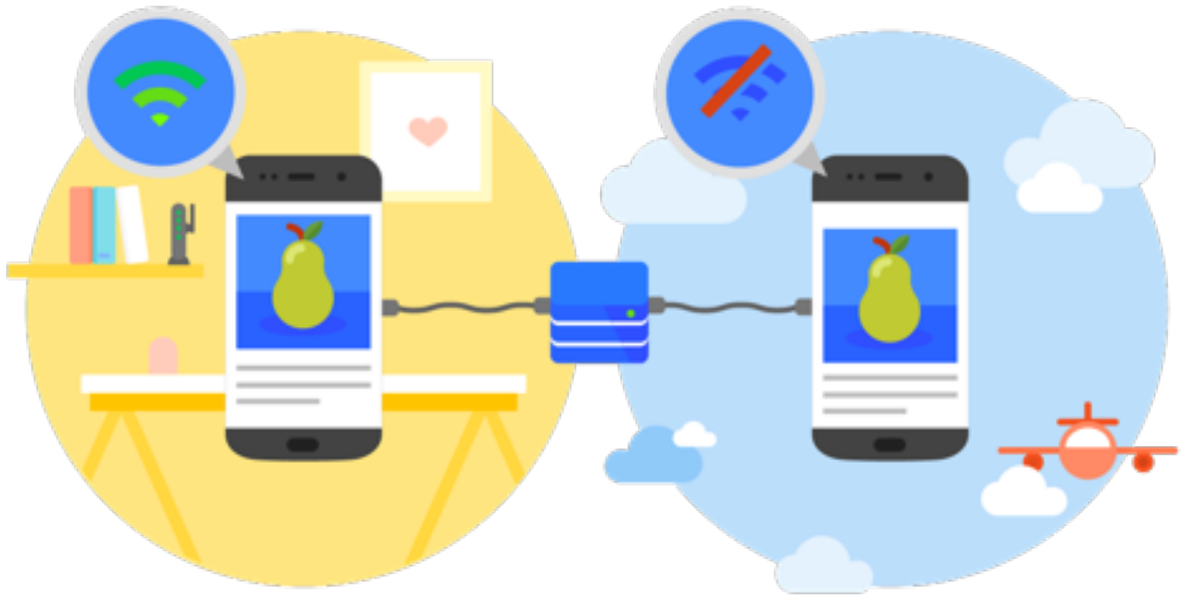
I FELT PRETTY CLEVER UNTIL I REALIZED I'D INVENTED WEBPAGES.

DEFINICIÓN

- Progressive Web Apps son aplicaciones web que son:
  - Fehacientes - Cargan instantáneamente, incluso con malas condiciones de conexión.
  - Rápidas - Responden rápido a la interacción con el usuario con animaciones suaves.
  - Atractivas - Parecen aplicaciones nativas, con una inmersiva experiencia de usuario.

FEHACIENTES: SERVICE WORKERS I

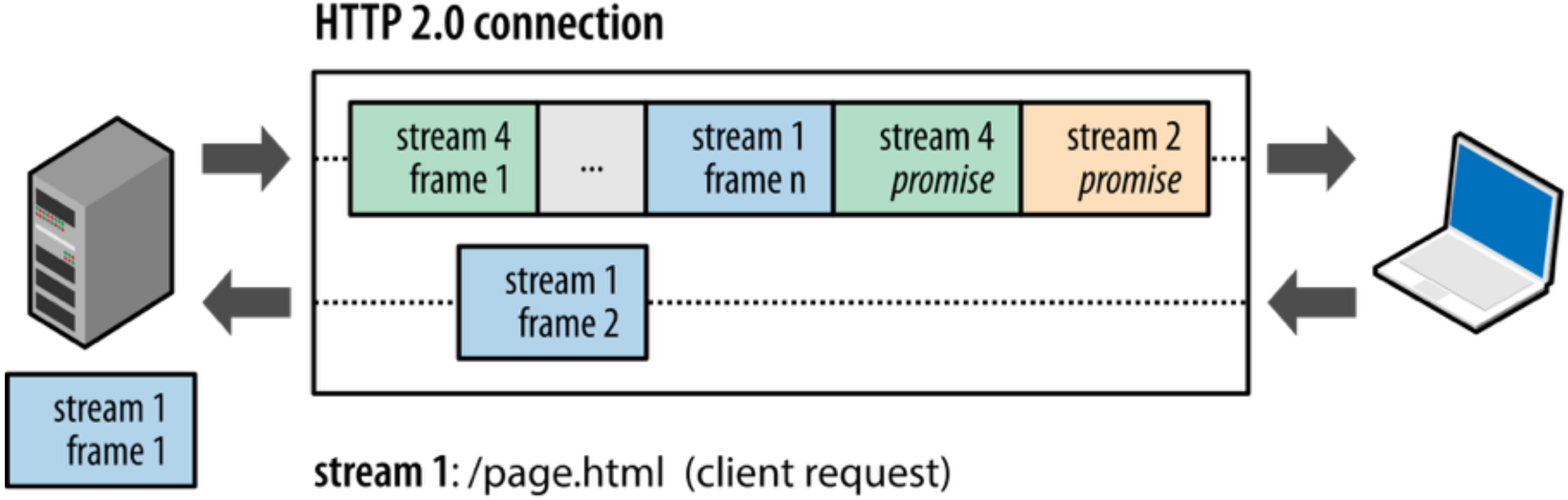
- Cuando lanzamos la aplicación, el *service worker* permite a la PWA cargar inmediatamente, sin importar el estado de la conexión.
- Un *service worker* es como un proxy escrito en JavaScript que te permite controlar la caché y cómo responder a peticiones de recursos.
- Pre-cacheando recursos podemos eliminar la dependencia de conexión.



FEHACIENTES: SERVICE WORKERS II

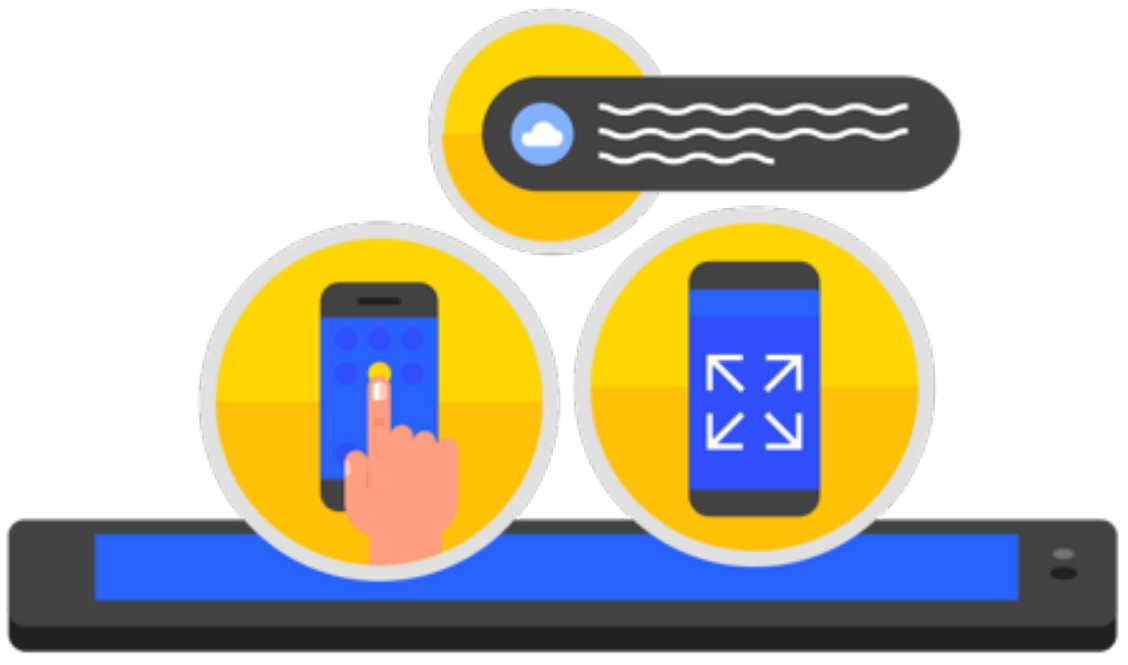
RÁPIDAS: RENDERIZACIÓN OPTIMIZADA – HTML IMPORTS – HTTP/2

- Transiciones CSS3 (alcanzando 60fps con propiedades como *will-change*)
- Html imports
- HTTP/2 permite enviar varias respuestas para una sola petición. El servidor puede enviar recursos adicionales al cliente, sin que el cliente los haya pedido explícitamente.

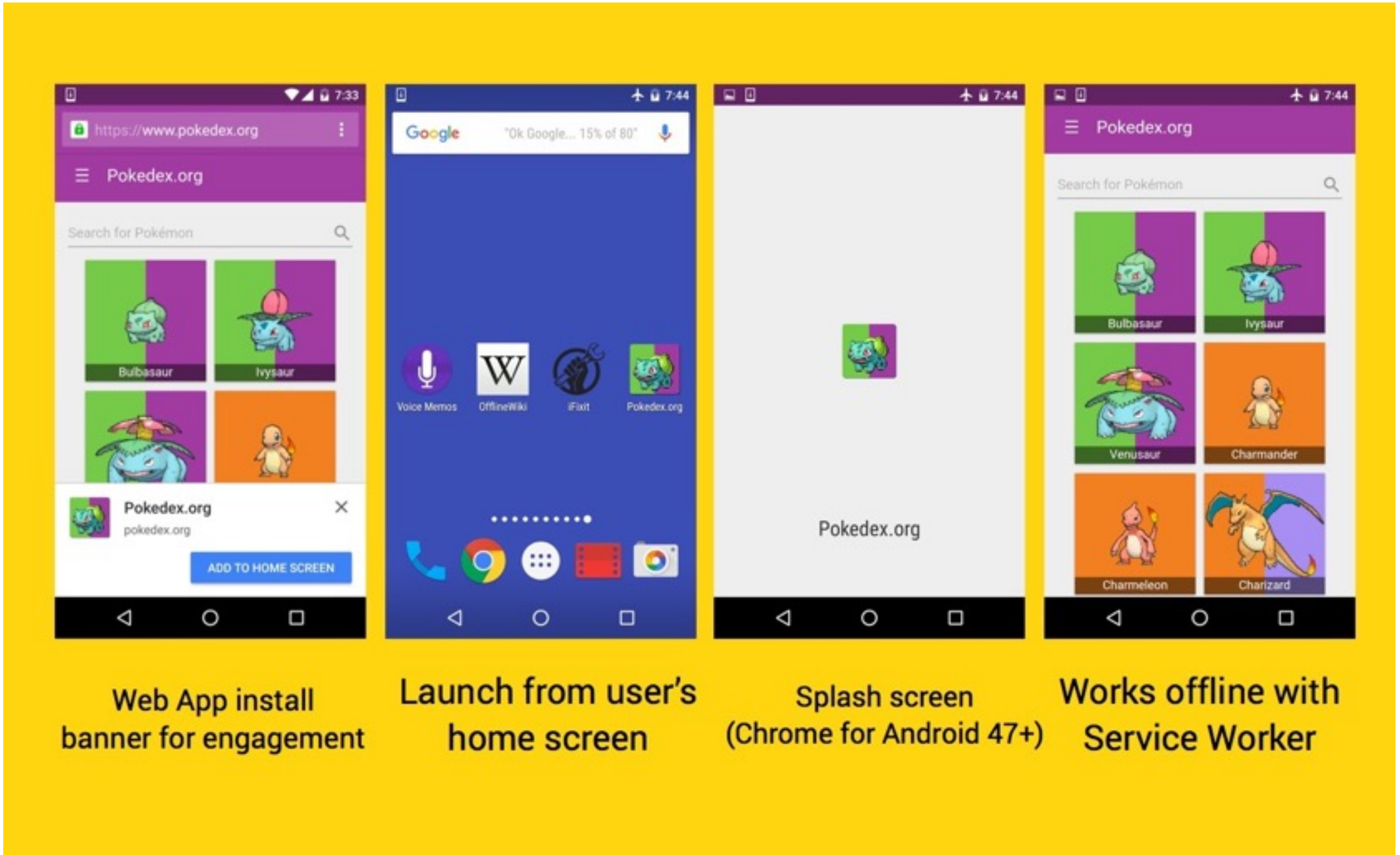


ATRACTIVAS I

- Conseguiremos que nuestras aplicaciones sean más atractivas y capten mejor la atención de los usuarios.
- Podremos tener aplicaciones web que funcionan a pantalla completa, instalables y que reciben notificaciones *push* aún cuando no esté en primer plano.



ATRACTIVAS II

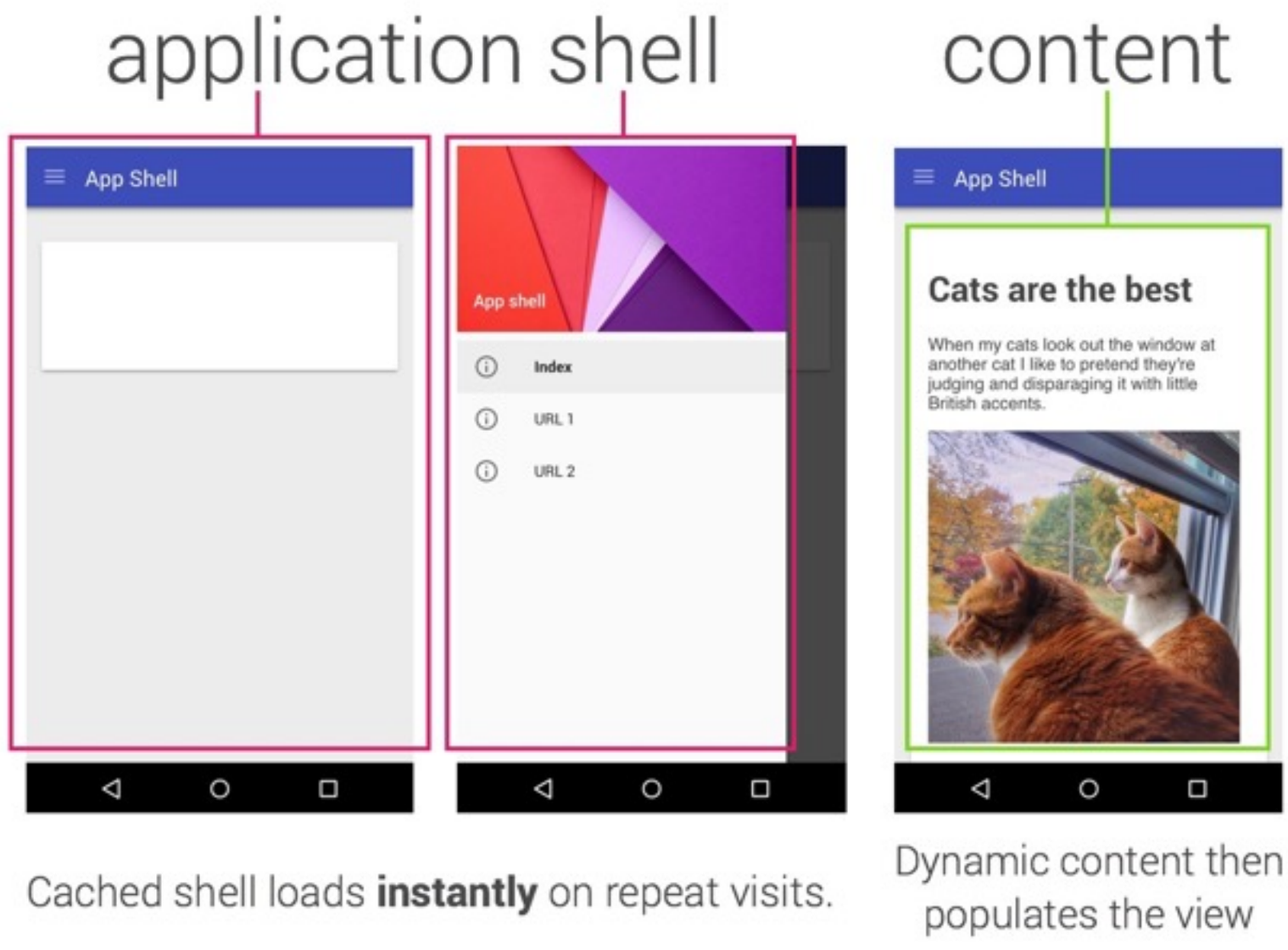


SERVICE WORKERS,
OFFLINE FIRST Y APP SHELLS

APPLICATION SHELL

QUE ES UN APPLICATION SHELL?

- Un **application shell** es el HTML, CSS y JavaScript mínimo necesario para crear una interfaz de usuario.
- Debe
  - Cargar rápido
  - Ser cacheado
  - Mostrar contenido dinámico



WEB APP MANIFEST FILE

- En las pwa podemos definir un *manifest.json* en el cual podemos incluir el nombre de nuestra aplicación, iconos, si queremos que la aplicación se vea en pantalla completa cuando la instalemos.

```
{
  name: "MurciaFrontend WPA",
  gcm_sender_id: "156559432692",
  permissions: [
    "gcm"
  ],
  icons: [
    {
    src: "assets/icons/96.png",
    sizes: "96x96",
    type: "image/png"
    },
    {
    src: "assets/icons/144.png",
    sizes: "144x144",
    type: "image/png"
    },
    {
    src: "assets/icons/192.png",
    sizes: "192x192",
    type: "image/png"
    }
  ],
  start_url: "/",
  display: "standalone",
  theme_color: "#9A1319"
}
```

SERVICE WORKERS

REGISTRANDO UN SERVICE WORKER

```
if('serviceWorker' in navigator) {
  navigator.serviceWorker
           .register('./service-worker.js', {scope: ‘/app'})
           .then(function(register){
              console.log('Service Worker Registered');
            });
}
```

- Los *service workers* se definen en un archivo diferente al de nuestra aplicación.
- Importante!: El *sw* debe ser servido desde **https**
- Al registrar el *sw* podemos indicar un *scope* para especificar un subconjunto de los contenidos sobre los que queramos que el *sw* tenga control. Fuera de ese ámbito, el *sw* no tendrá efecto.

SERVICE WORKERS: EVENTOS

- Los service workers responden a eventos
  - Eventos de ciclo de vida
    - install
    - *activate*
    - *message*: Envío de mensajes al service worker
      - *navigator.serviceWorker.controller.postMessage(message);*
- *Eventos funcionales*
  - *fetch*: Petición http
  - *sync*: Sincronización en segundo plano
  - *push*: Notificación push

CICLO DE UN SERVICE WORKER

- Instalación: La primera vez que se ejecuta un *sw* cacheamos la *app shell*.
- Activación: Si la fase de instalación se completa con éxito, activamos el *sw* cacheando nuevos recursos y borrando la caché obsoleta.
- En espera: Esperamos que ocurran eventos (fetch, push, sync...)

```
var log = console.log.bind(console);
var err = console.error.bind(console);

self.addEventListener('install', (e)=>{ log('Service Worker: Installed'); });
self.addEventListener('activate',(e)=>{ log('Service Worker: Active'); });
self.addEventListener('fetch',   (e)=>{ log('Service Worker: Fetch'); });
```

CACHEANDO RECURSOS

CACHEANDO EL APPLICATION SHEL

```
const CACHE_VERSION = 'v1';

self.addEventListener('install', (event) => {
  console.log('Service worker installing...');

  event.waitUntil(
    caches.open(CACHE_VERSION)
    .then(cache => cache.addAll([
      '/',
      'index.html',
      'main.js',
      'css/style.css',
    ]))
  );
});
```

En el momento en el que instalamos un service worker podemos cachear el *application shell*, puesto que será la base de nuestra aplicación y siempre necesitaremos que esté disponible.

Si nuestra aplicación tiene muchos archivos, podemos usar plugins para grunt, gulp o wepback para que nos cachee los ficheros estáticos eel *application shell*.

ELIMINANDO CACHES ANTIGUAS

```
self.addEventListener('activate', (event) => {
  console.log('Service worker activating...');

  console.log('Clearing old caches');
  caches.keys()
    .then(cacheNames => cacheNames
    .filter(cacheName => cacheName !== CACHE_VERSION)
    .forEach(cacheName => caches.delete(cacheName)));
});
```

El evento *activate* ocurre cuando se instala satisfactoriamente un *service worker*.

Si los recursos de nuestra aplicación han cambiado, y por eso estamos cargando un nuevo service worker, éste es el momento para borrar los viejos recursos.

CACHEANDO RECURSOS I

```
self.addEventListener('fetch', (event) => {
  console.log('Fetching: ', event.request.url);

  if (event.request.method !== 'GET')
    return fetch(event.request);

  event.respondWith(
    caches.match(event.request)
    .then(
      response =>
        response
      ||
        fetch(event.request)
          .then(response =>
            caches.open(CACHE_VERSION)
              .then(cache => {
                console.log('caching: ', event.request);
                cache.put(event.request, response.clone());
                return response;
              })
          )
          .catch(err => cache.match('/fallback.html'))
    )
  );
});
```

CACHEANDO RECURSOS II

```
self.addEventListener('fetch', (event) => {
  console.log('Fetching: ', event.request.url);

  if (event.request.method !== 'GET')
    return fetch(event.request);

  event.respondWith(
      fetch(event.request)
      .then(response =>
        caches.open(CACHE_VERSION)
          .then(cache => {
            console.log('caching: ', event.request);
            cache.put(event.request, response.clone());
            return response;
          })
      )
      .catch(err =>
        cache.match(event.request)
          .catch(() => cache.match('/fallback.html'))
      )
    )
  );
});
```

CACHEANDO RECURSOS III

- Condición de carrera?
  - Promise.race(fetch(…), caches.match(…)) ?
- Gestión más compleja
  - Google SW-Toolbox

```
// Match URLs that end in index.html
toolbox.router.get(/index.html$/, function(request) {
  return new Response('Handled a request for ' + request.url);
});

// Match URLs that begin with https://api.example.com
toolbox.router.post(/^https:\/\/api.example.com\//, apiHandler);
```

RECIBIENDO NOTIFICACIONES

NOTIFICACIONES PUSH I

```
navigator.serviceWorker
  .register('/sw.js')
  .then(function(registration){
    registration.pushManager
      .subscribe({
        userVisibleOnly: true,
        applicationServerKey: publicKey,
      })
      .then(function(subscription){
        // send subscription object to server
        fetch('/subscription', {
          headers: {
            'Accept': 'application/json',
            'Content-Type': 'application/json',
          },
          method: 'POST',
          body: JSON.stringify({ subscription: subscription }),
        });
      })
})
```

NOTIFICACIONES PUSH I

```
self.addEventListener('push', (event) => {
  const json = JSON.parse(event.data.text());

  event.waitUntil(
    self.registration.showNotification(json.title, {
      body: json.body,
    })
  );
});
```

DEMO

DEMO

- código: github.com/gabrielperales/MurciaFrontendPWA
- website: https://murciafrontendpwa.now.sh

EJEMPLOS DE PWA

EJEMPLOS DE PWA

- mobile.twitter.com
- flipkart.com
- pokedex.org
- m.aliexpress.com
- washingtonpost.com/pwa
- wiki-offline.jakearchibald.com
- voice-memos.appspot.com
- pwa.rocks/

RECURSOS

RECURSOS

- Chrome Lighthouse extension
- SW Toolbox
- PouchDB
- whatwebcando.today
- PRPL Pattern
- sw-precache-webpack-plugin
- github.com/GoogleChrome/sw-precache

¿PREGUNTAS?


MUCHAS GRACIAS
THE LATE LATE SHOW
JAMES CORDEN

ARTÍCULOS

ARTÍCULOS

- https://www.fastly.com/blog/service-workers-pwas-and-whats-next-for-mobile
- https://www.polymer-project.org/1.0/toolbox/server
- https://medium.com/@addyosmani/progressive-web-apps-with-react-js-part-i-introduction-50679aef2b12