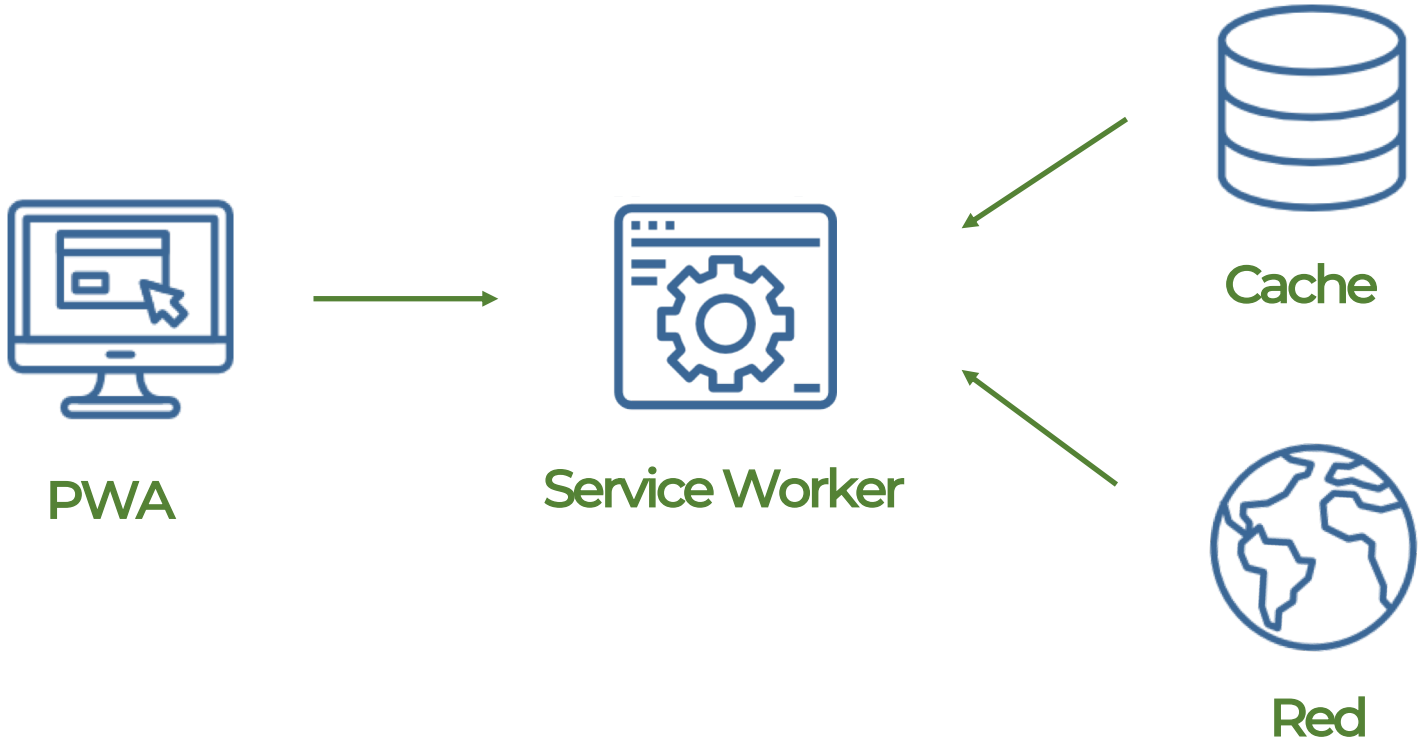
ESCRIBIENDO SERVICE WORKERS CON WORKBOX

侍 Pablo Magaz

https://pablomagaz.com

侍 ¿Qué es un Service Worker?



侍 ¿Qué puede hacer un Service Worker?

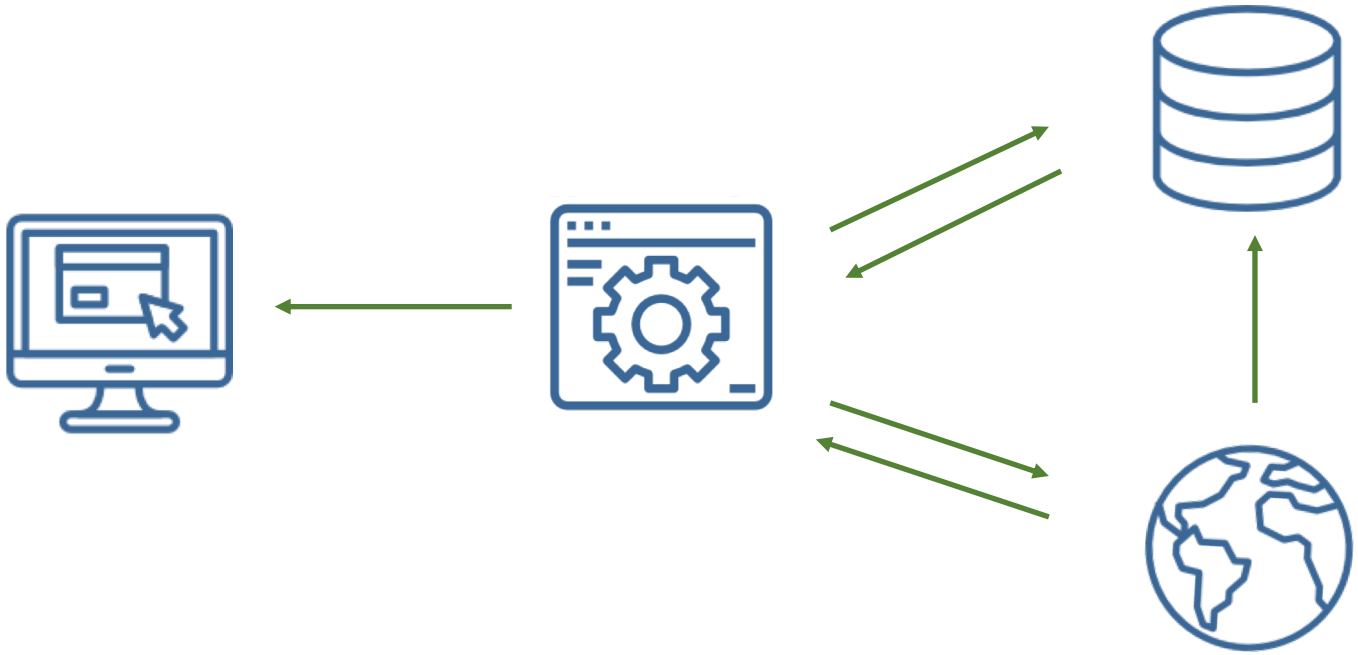
- ✔ Cache
- ✔ Contenido Offline
- ✔ Notificaciones Web Push
- ✔ Sincronización en Segundo plano

侍 Cache Storage

caches.open

```
caches.open('myCache').then(cache => (
  cache.match(myUrl).then(cachedResponse, => ...
)) };
```

cache.match	Devuelve el archivo cacheado si existe
cache.put	Inserta un archivo en la cache
cache.delete	Borra un archivo de la cache



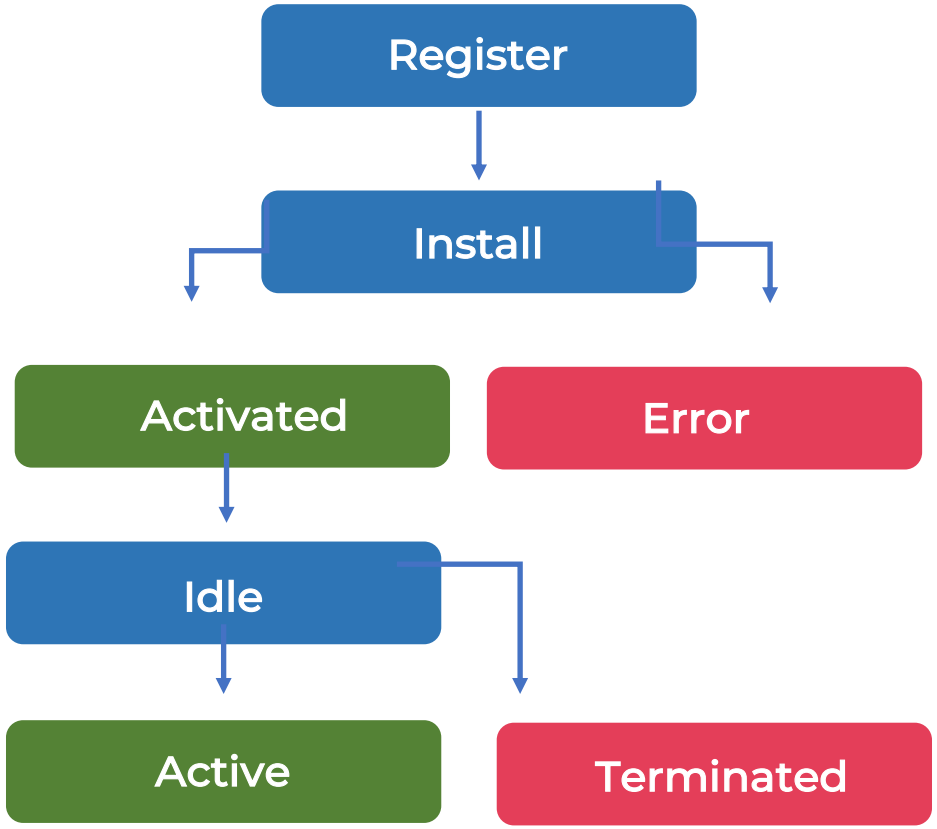
```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.open('myCache').then(cache => {
      return cache.match(event.request).then(cachedResponse => {
        const fetchPromise = fetch(event.request).then(response => {
          cache.put(event.request, response.clone());
            return response;
        });
        return cachedResponse || fetchPromise;
      })
    }));
});
```

https://developers.google.com/web/fundamentals/instant-and-offline/offline-cookbook/

```
workbox.routing.registerRoute(
 /\.(?:js|css)$/,
 workbox.strategies.staleWhileRevalidate()
);
```


Workbox

侍 Ciclo de Vida de un Service Worker



侍 Registrando un Service Worker

```
<script>
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/serviceWorker.js');
  });
}
</script>
```

侍 Importando Workbox

```
importScripts('https://storage.googleapis.com/workbox/3.2.0/workbox-sw.js');

if (workbox) { // workbox solo existe en el scope del serviceWorker
  console.log('Workbox loaded!');
} else {
  console.log('Can't load Workbox!')
}
```

侍 Tipos de Cache

PreCache

Cache estática para archivos que conocemos de antemano

Runtime

Cache dinámica para archivos dinámicos como imágenes de contenido, etc.

侍 Configurando Workbox

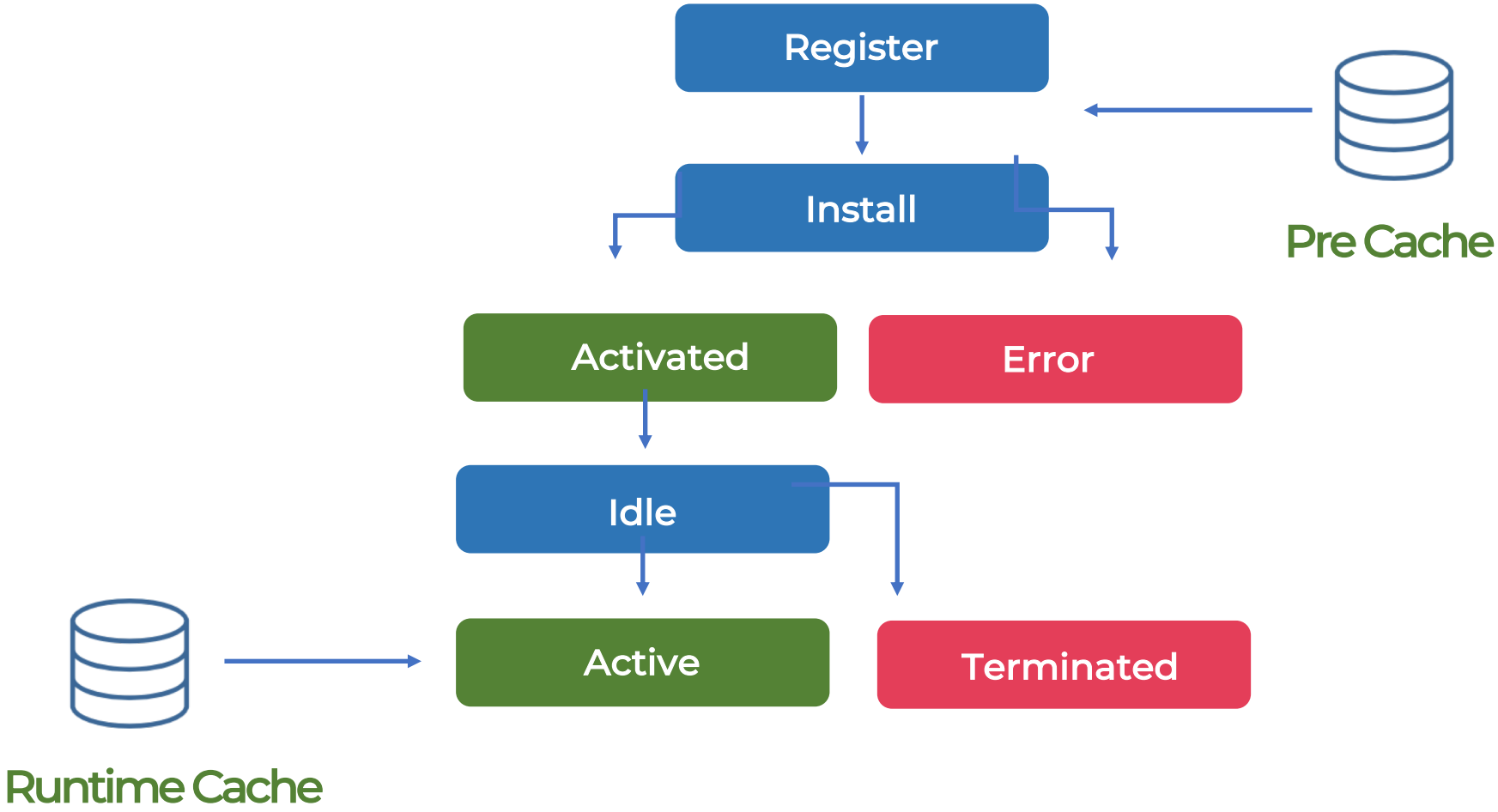
```
workbox.core.setCacheNameDetails({
  prefix: 'my-app',
  suffix: 'v1',
  precache: ‘my-precache',
  runtime: ‘my-runcache'
});

// my-app-my-precache-v1

// my-app-my-runcache-v1
```

侍 Configurando Workbox

```
const staticCache = 'my-cache-static-v1'
const dynamicCache = 'my-cache-dynamic-v1'

workbox.core.setCacheNameDetails({
  precache: staticCache,
  runtime:  dynamicCache
});

workbox.core.cacheNames.precache
// my-app-my-precache-v1
```

侍 Precache

```
workbox.precaching.precacheAndRoute([
  '/assets/app.27afb7e661ed43f51738.js',
  '/assets/vendor.27afb7e661ed43f51738.js',
   '/assets/styles.d62ba9e609ca8215905e.css'
]);
```


Register
Install
Pre Cache
Activated
Error
Idle
Active
Terminated
Runtime Cache

侍 Routing

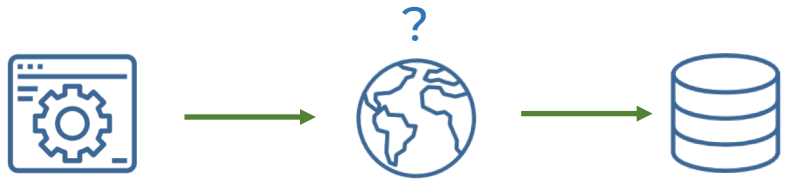
```
workbox.routing.registerRoute(
  /\.(?:js|css)$/,
    ...
);

workbox.routing.registerRoute(
  new RegExp('https://fonts.(?:googleapis|gstatic).com/(.*) '),
  ...
);
```

“Una estrategia de ’cacheo’ es el comportamiento que vamos a aplicar a una determinada ruta”

侍 Cache First


```
workbox.routing.registerRoute(
 /\.(?:png|gif|jpg|jpeg)$/,
  workbox.strategies.cacheFirst({
    cacheName:  workbox.core.cacheNames.runtime
  })
);
```

侍 Network First




```
workbox.routing.registerRoute(
  new RegExp('/users/myAccount/(.*)'),
  workbox.strategies.netWorkFirst({
    cacheName:  'runtimeCache',
    networkTimeoutSeconds: 2
  })
);
```

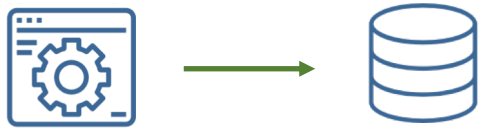
侍 Stale while revalidate


```
workbox.routing.registerRoute(
  /\.(?:js|css)$/,
  workbox.strategies.staleWhileRevalidate({
    cacheName: dynamicCache,
  })
);
```

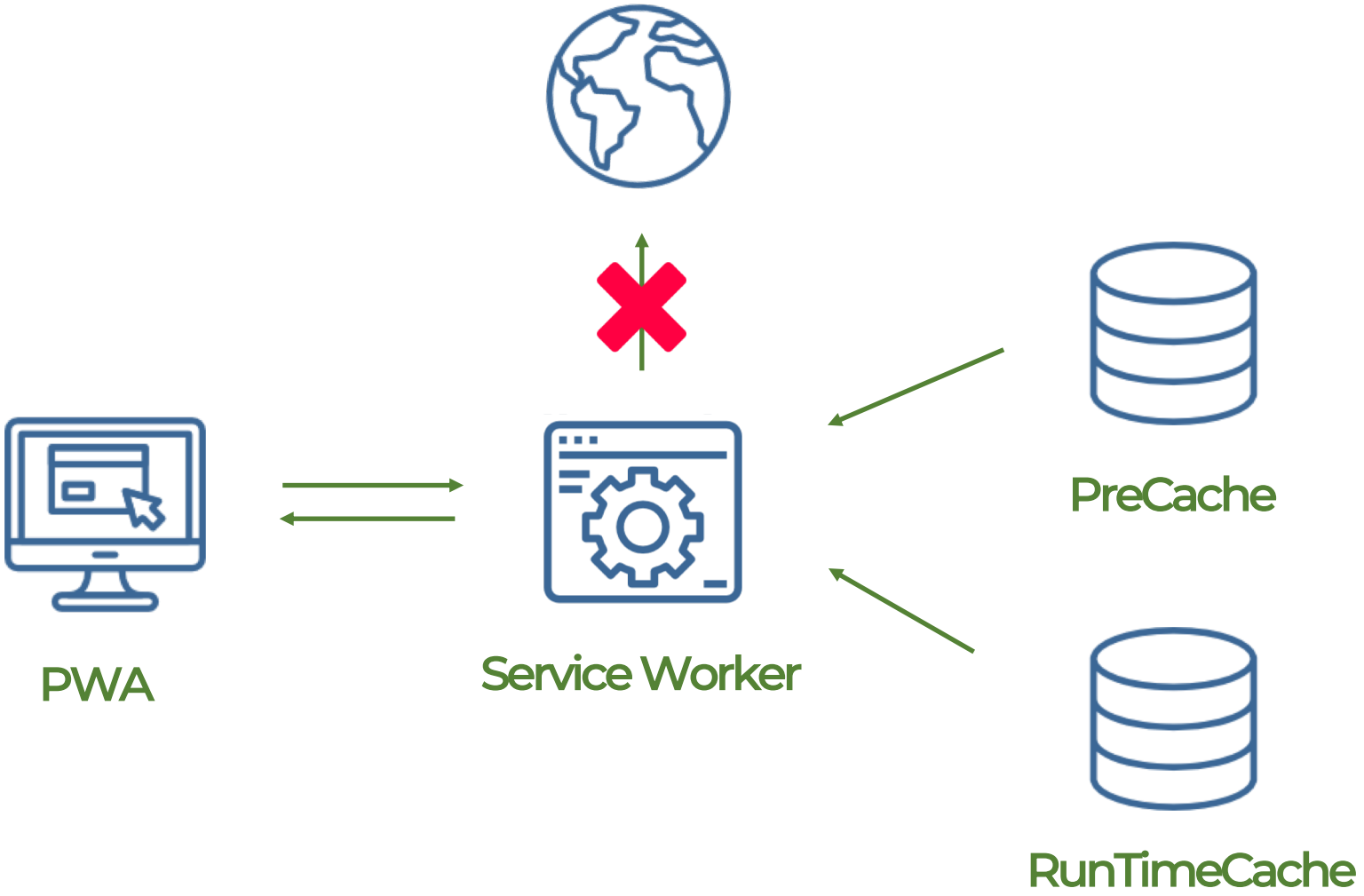
侍 Network Only

```
workbox.routing.registerRoute(
  new RegExp('/api/(.*)'),
  workbox.strategies.networkOnly({})
);
```

侍 Cache Only



```
workbox.routing.registerRoute(
    /\.(?:png|gif|jpg|jpeg)$/,
    workbox.strategies.cacheOnly({
    cacheName: workbox.core.cacheNames.runtime
  })
);
```


PWA
Service Worker
PreCache
RunTimeCache

```
workbox.precaching.precacheAndRoute([
  ‘index.html’,
  ‘otherPage.html’,
  '/assets/app.27afb7e661ed43f51738.js',
  '/assets/vendor.27afb7e661ed43f51738.js’,
   '/assets/styles.d62ba9e609ca8215905e.css’
]);

workbox.routing.registerRoute(
 /\.(?:png|gif|jpg|jpeg)$/,
  workbox.strategies.cacheFirst({
    cacheName: runtimeCache,
  })
);
```

侍 Custom Offline page

```
workbox.precaching.precacheAndRoute([
  ‘offline.html',
]);

workbox.routing.registerRoute(
  ({ event }) => event.request.mode === 'navigate',
  ({ url }) => fetch(url.href).catch(() => caches.match('/offline.html'))
);
```

“Los plugins permiten extender la funcionalidad de las estrategias de ‘cacheo’ o de una cache completa”

侍 Expiration Plugin

```
workbox.routing.registerRoute(
  /\.(?:jpg|png)$/,
  workbox.strategies.cacheFirst({
  cacheName: runtimeCache,
  plugins: [
    new workbox.expiration.Plugin({
      maxEntries: 50,
      maxAgeSeconds:  24 * 60 * 60
    })
  ]
}));
```

侍 Cacheable Response Plugin

```
workbox.routing.registerRoute(
  new RegExp('/api/products/'),
  workbox.strategies.cacheFirst({
    plugins: [
     new workbox.expiration.Plugin({
       maxAgeSeconds: 3 * 60 * 60
     }),
     new workbox.cacheableResponse.Plugin({
       statuses: [200]
     })]
}));
```

侍 Sincronización en segundo plano

```
navigator.serviceWorker.register('/serviceWorker.js');

navigator.serviceWorker.ready.then( (swRegistration) => {
    return swRegistration.sync.register('mySyncTag');
  });
});

 self.addEventListener('sync', (event) =>  {
  if (event.tag == 'mySyncTag') event.waitUntil(backgroundStuff());
});
```

侍 BackgroundSync Plugin

```
const bgSyncPlugin = new workbox.backgroundSync.Plugin('SyncQueue', {
  maxRetentionTime: 60 * 60
});

workbox.routing.registerRoute(
  new RegExp('/api/shoppingCart/(.*)'),
  workbox.strategies.networkOnly({
    plugins: [ bgSyncPlugin ]
   }),
   'POST'
);
```

```
class MyPlugin  {

  async requestWillFetch({ request }) {
    return request
  }

  async cachedResponseWillBeUsed({ request, matchOptions, cachedResponse }) {
   return cachedResponse
}

  async cacheWillUpdate({ request, response }) {
   return response
  }

}
```

¨Workbox dispone de un CLI que permite automatizar la generación del service worker¨

侍 Worbox Wizard

$ workbox wizard

```
? What is the root of your web app? build/
? Which file types would you like to precache? css, html, js
? Where would you like your service worker file to be saved? build/sw.js
? Where would you like to save these configuration options? workbox-config.js
To build your service worker, run

  workbox generateSW workbox-config.js

as part of a build process. See https://goo.gl/fdTQBf for details.
You can further customize your service worker by making changes to workbox-config.js.
```

```
module.exports = {
  globDirectory: 'build/',
  globPatterns: [ '**/*.{ html, js, css }' ],
  swDest: 'build/sw.js',
  runtimeCaching: [{
    urlPattern: /\.(?:png|jpg|jpeg|svg)$/,
    handler: 'cacheFirst',
    options: {
      cacheName: 'myDynamicCache',
      expiration: { maxEntries: 10, },
    },
  }],
};
```

```
$ workbox generateSw path/to/workbox-config.js

...


importScripts("https://storage.googleapis.com/workbox-cdn/3.5.0/workbox-sw.js");


self._precacheManifest = [
  '/assets/app.7f9418722b42fd55b5e4.js',
  '/assets/vendor.7f9418722b42fd55b5e4.js',
  '/assets/styles.d62ba9e609caae10782159dc7b3a905e.css'
]

workbox.precaching.precacheAndRoute(self._precacheManifest, {})

workbox.strategies.cacheFirst({
    cacheName: runtimeCache,
  })
);

...
```

侍 Workbox Webpack Plugin

```
// Wepack.config.js
const { GenerateSW } = require('workbox-webpack-plugin');

module.exports = {
  module: { rules : [...] }
  plugins: [
    new GenerateSW({
      include: [/\.html$/, /\.js$/, /\.css$/],
      ...
    })]
};
```

Gracias.
¿Preguntas?

Slides: https://pablomagaz.com/#about

twitter.com/pablo_magaz