

**PACHY DEV'S**

# Gabriel Napolião

## Frontend Developer



[/gabrielndr](https://github.com/gabrielndr)



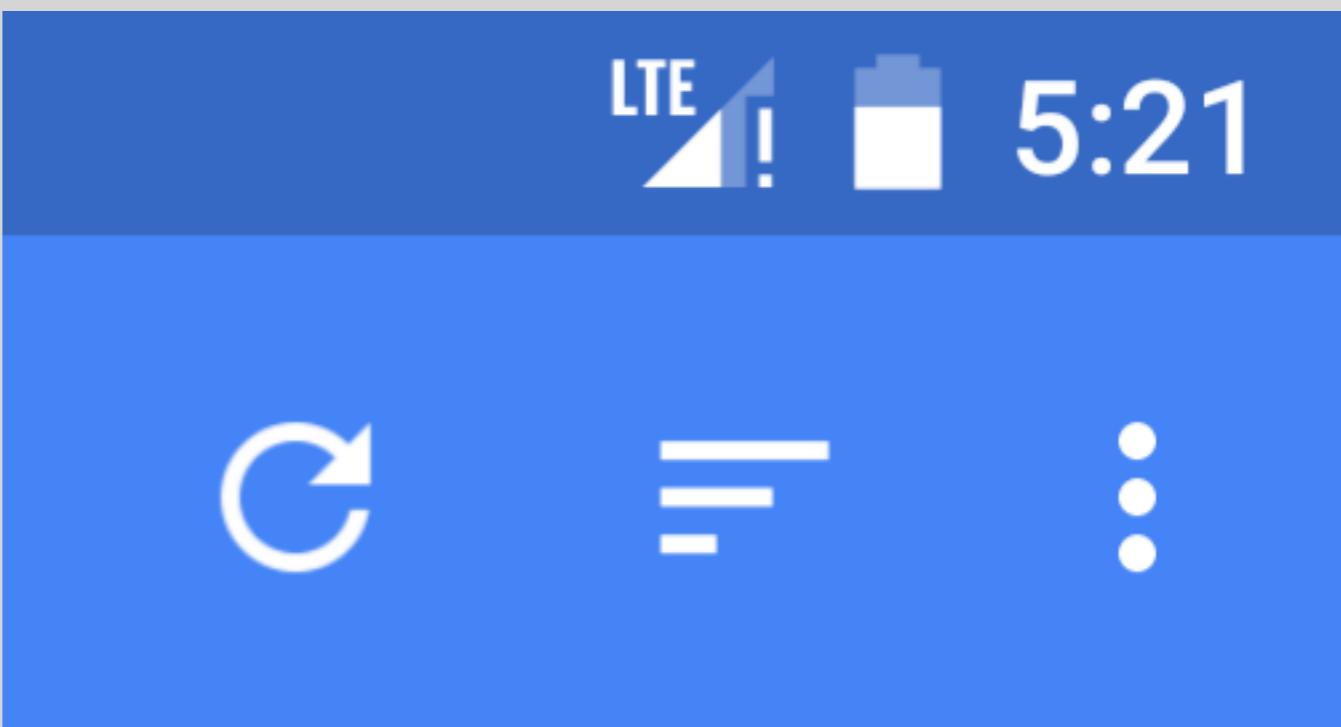
[gabrielndr.github.io](https://gabrielndr.github.io)

# Service Workers

## Offline Web



# Offline?



Nos primórdios do  
HTML5...

*Application Cache API*

# index.html

```
<html manifest="arquivo.appcache">
```

# O Manifesto:

CACHE MANIFEST

/index.html

/img/logo.png

/css/style.css

/js/app.js

NETWORK:

<https://google-analytics.com/ga.js>

FALLBACK:

/feed.html

# Simples...



- Tem que acertar o mime-type no servidor
- Não pode esquecer nenhuma URL
- Sempre cacheia a página
- Nada pode dar erro 404 ou 500
- **CUIDADO PARA NÃO CACHEAR O MANIFESTO**
- Usuário não pode controlar nada
- Impossível escolher o que cachear
- Potencial para detonar o 3G do usuário
- Remover do cache é um parto
- Não posso impedir update automático
- Terrível para desenvolver e debugar

AppCache é chato,  
limitado e  
complicado

É declarativo e mágico

# SERVICE WORKERS

# ENTRAM EM AÇÃO





```
<!DOCTYPE html>
<html lang="en">
  <head>

  </head>
  <body>
    <h1>Página Offline</h1>
  </body>
</html>
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script>
      navigator.serviceWorker.register('offline.js');
    </script>
  </head>
  <body>
    <h1>Página Offline</h1>
  </body>
</html>
```



```
this.onfetch = event => {  
  console.log(event.request.url);  
}
```



```
this.onfetch = event => {
  event.respondWith(
    new Response("<h1>Página Offline!</h1>");
  )
}
```

# Service Workers

É um **Worker** orientado a **eventos**, que **controla** as páginas em **background**. Nele tudo é **assíncrono**, e ele pode interceptar chamadas de **rede** e usar um **cache** de recursos

# JAVASCRIPT COMUM

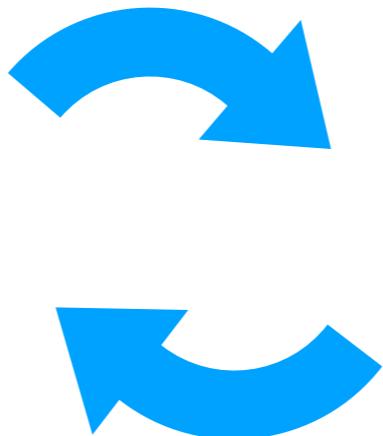
```
<script src="script.js" async></script>
```



# WEB WORKERS

```
<script> new Worker('worker.js'); </script>
```

DOM  
CSSOM  
LAYOUT  
EVENTOS  
SCROLL

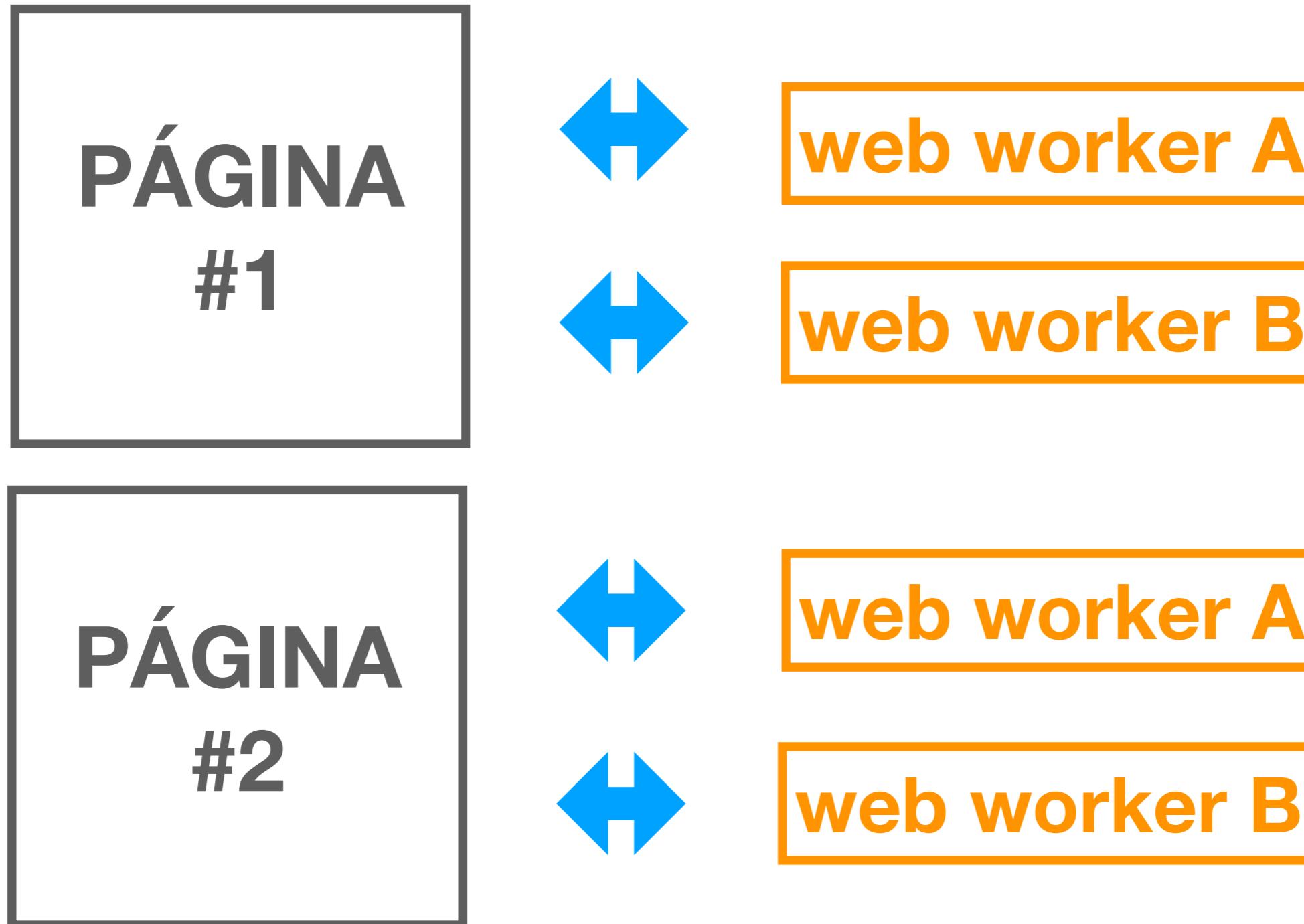


script  
script  
script  
script

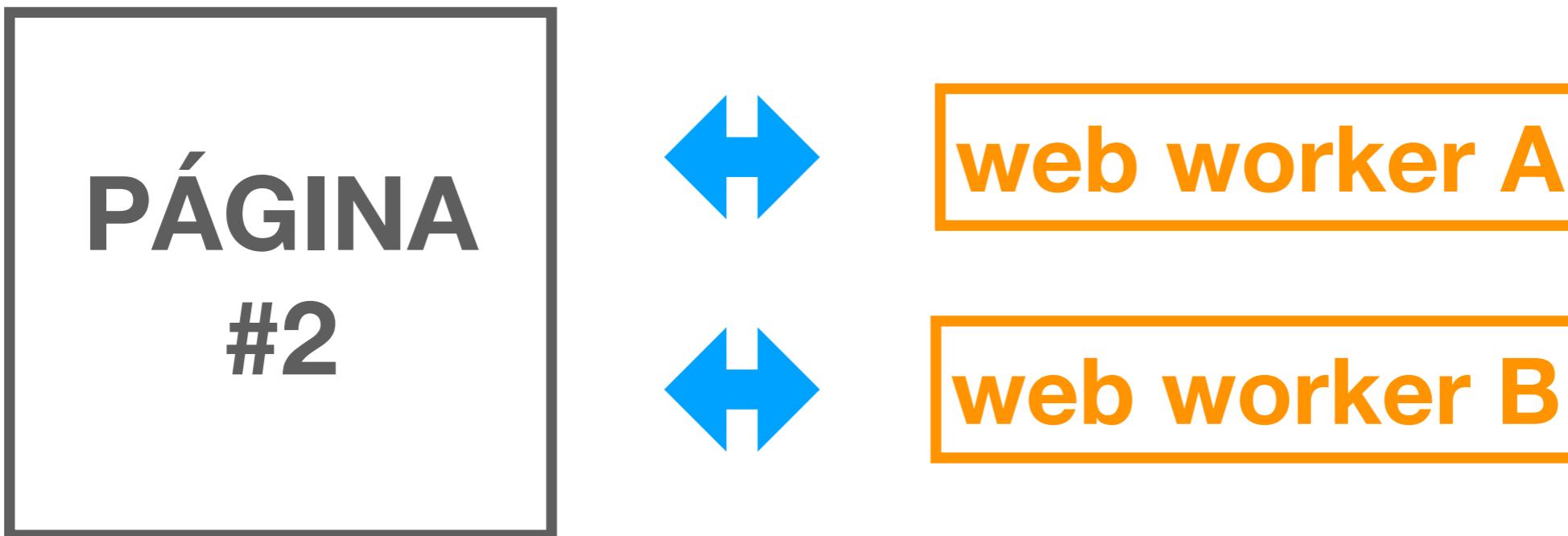
# WEB WORKERS



# WEB WORKERS



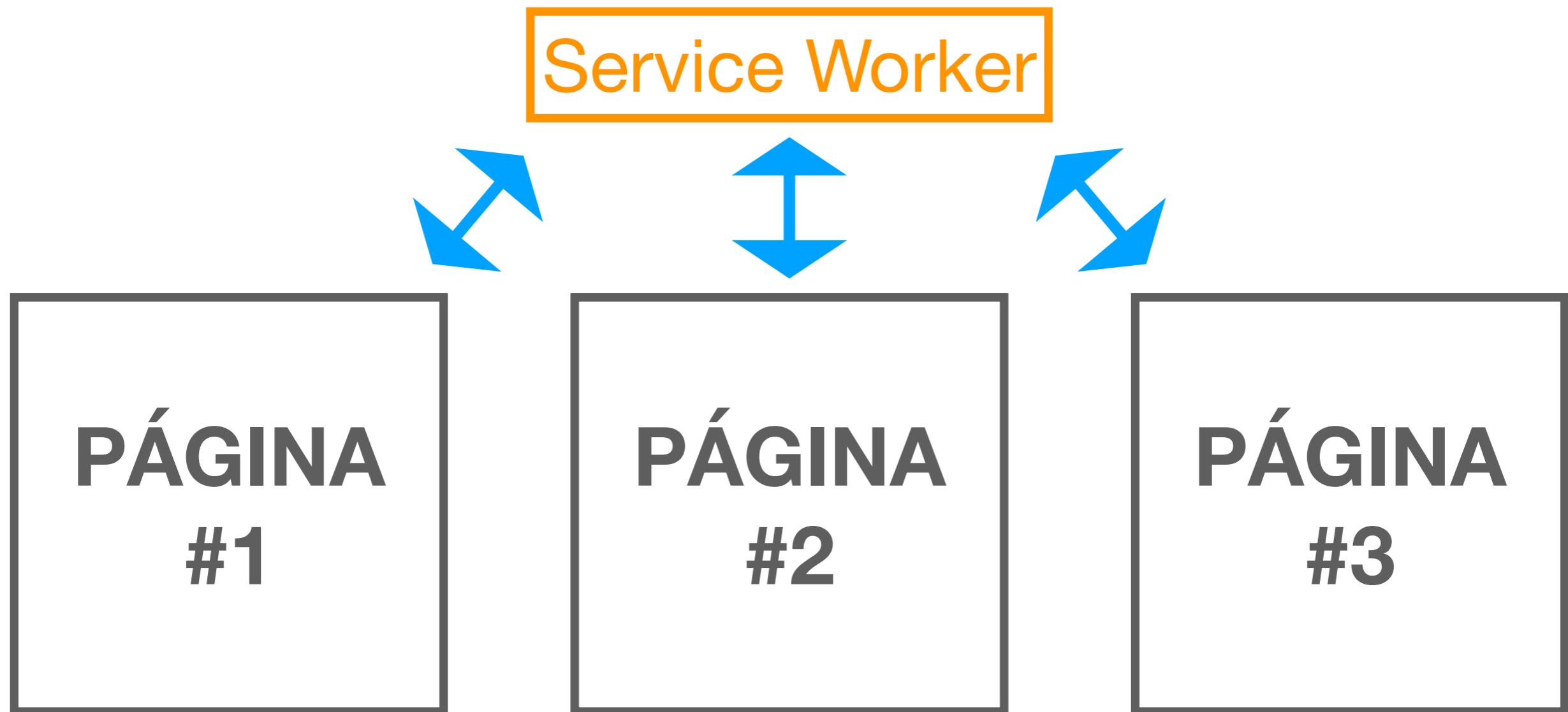
# WEB WORKERS



# WEB WORKERS

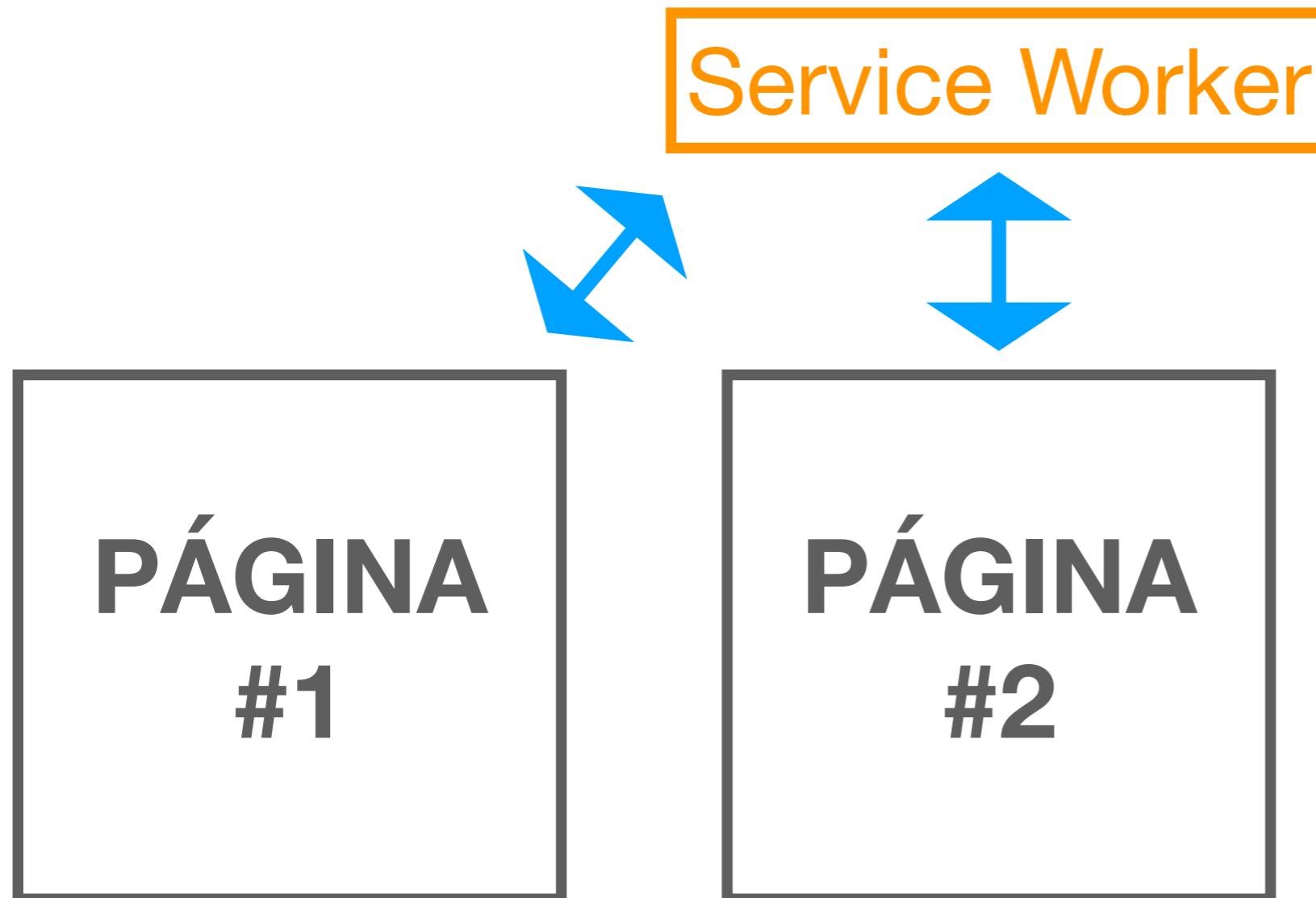
# SERVICE WORKER

```
<script>  
navigator.serviceWorker.register('sw.js');  
</script>
```



# SERVICE WORKER

```
<script>  
navigator.serviceWorker.register('sw.js');  
</script>
```



# SERVICE WORKER

```
<script>  
navigator.serviceWorker.register('sw.js');  
</script>
```

Service Worker



PÁGINA  
#1

# SERVICE WORKER

```
<script>  
navigator.serviceWorker.register('sw.js');  
</script>
```

Service Worker

# Service Workers

É um **Worker** orientado a **eventos**, que **controla** as páginas em **background**. Nele tudo é **assíncrono**, e ele pode interceptar chamadas de **rede** e usar um **cache** de recursos



```
navigator.serviceWorker.register('/sw.js').then(registration => {
  console.log('Service worker registrado com sucesso:');
}).catch(function(error) {
  console.log('Falha ao Registrar o Service Worker:', error);
})
```



```
if ('serviceWorker' in navigator) {  
    navigator.serviceWorker.register('/sw.js').then(registration => {  
        console.log('Service worker registrado com sucesso:');  
    }).catch(function(error) {  
        console.log('Falha ao Registrar o Service Worker:', error);  
    })  
} else {  
    console.log('Service workers não suportado!');  
}
```

# EVENTOS



```
this.oninstall = event => {
  console.log('Instalou');
}

this.onactivate = event => {
  console.log('Ativou');
}

this.onfetch = event => {
  event.respondWith(
    new Response("<h1>Olá, você está offline!</h1>");
  );
}
```

# CACHE API



```
this.oninstall = event => {
  console.log('Instalou');
}

this.onactivate = event => {
  console.log('Ativou');
}

this.onfetch = event => {
  event.respondWith(
    new Response("<h1>Olá, você está offline!</h1>");
  );
}
```



```
cache.open('aplicacao')
```



```
cache.open('aplicacao').then(cache => {  
})
```



```
cache.open('aplicacao').then(cache => {  
    cache.add('pg.html');  
    cache.add('style.css');  
})
```



```
cache.open('aplicacao').then(cache => {
  cache.addAll([
    '/index.html',
    '/style.css',
    'logo.png',
    '/contato.html',
    'https://www.google-analytics.com/analytics.js'
  ]);
})
```



```
cache.open('aplicacao').then(cache => {
    return cache.addAll([
        '/index.html',
        '/style.css',
        'logo.png',
        '/contato.html',
        'https://www.google-analytics.com/analytics.js'
    ]);
})
```



```
this.oninstall = event => {

    cache.open('aplicacao').then(cache => {
        return cache.addAll([
            '/index.html',
            '/style.css',
            'logo.png',
            '/contato.html',
            'https://www.google-analytics.com/analytics.js'
        ]);
    })
}
```



```
this.oninstall = event => {
  event.waitUntil(
    cache.open('aplicacao').then(cache => {
      return cache.addAll([
        '/index.html',
        '/style.css',
        'logo.png',
        '/contato.html',
        'https://www.google-analytics.com/analytics.js'
      ]);
    })
  );
}
```

# CACHE Programático & Controlável

Cacheio URLs como eu quero

Gero endereços num for com certa regra

Recursos diferentes dependendo do browser

Levo em conta alguma preferência do usuário

Mudo de acordo com o hardware e contexto

# RESPOSTA OFFLINE



```
this.onfetch = event => {  
  console.log(event.request.url);  
}
```



```
this.onfetch = event => {  
  event.respondWith(  
    new Response('Contéudo');  
  )  
}
```



```
this.onfetch = event => {  
  event.respondWith(  
    caches.match(event.request)  
  )  
}
```

E SE NÃO EXISTIR NO  
CACHE?



```
this.onfetch = event => {  
  event.respondWith(  
    caches.match(event.request)  
  )  
}
```



```
this.onfetch = event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      })
    )
}
```



```
this.onfetch = event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || event.default();
    })
  )
}
```

**SE NÃO EXISTIR NA  
REDE NEM NO CACHE?**

# FALLBACK DE URLs



```
this.onfetch = event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || event.default();
    })
  )
}
```



```
this.onfetch = event => {
  event.respondWith(
    caches.match(event.request).then(response => {
      return response || event.default();
    }).catch(error => {
      return caches.match('/contato.html');
    })
  )
}
```

# CACHE Programático & Controlável

Busco no cache

Busco na rede

Devolvo fallback

Construo a resposta na mão

Tudo com lógica e a sequência que você quiser

# O CENTRAL DO SERVICE WORKER É

Ele dá duas vezes mais trabalho que o Application  
Cache para escrever

E 200 mil vezes menos trabalho, para o offline  
funcionar

# ATUALIZAÇÕES

## Mudar o `worker.js`

Detecta na próxima navegação

Dispara instalação (`oninstall`) em background  
(worker original ainda comanda a página)

Fecho a página

Worker velho é desativado

Novo worker é ativado (`onactivate`)  
(novo worker em ação)

Abro a página





```
this.oninstall = event => {
  event.waitUntil(
    cache.open('aplicacao').then(cache => {
      return cache.addAll([
        '/index.html',
        '/style.css',
        'logo.png',
        '/contato.html',
        'https://www.google-analytics.com/analytics.js'
      ]);
    })
  );
}
```



```
this.oninstall = event => {
  event.waitUntil(
    cache.open('aplicacao-v2').then(cache => {
      return cache.addAll([
        '/index.html',
        '/style.css',
        'logo.png',
        '/contato.html',
        'https://www.google-analytics.com/analytics.js'
      ]);
    })
  );
}
```



```
this.onactivate = event => {  
    event.waitUntil(  
        caches.delete('aplicacao-v1');  
    )  
}
```

# Atualização

Totalmente em background

Não incomoda o usuário

Só troca no próximo acesso

*Chrome-like*

# DETALHES

**HTTPS Only**

Tudo assíncrono

Pode ser morto a  
**QUALQUER MOMENTO**

MUITO MAIS PODEROSO

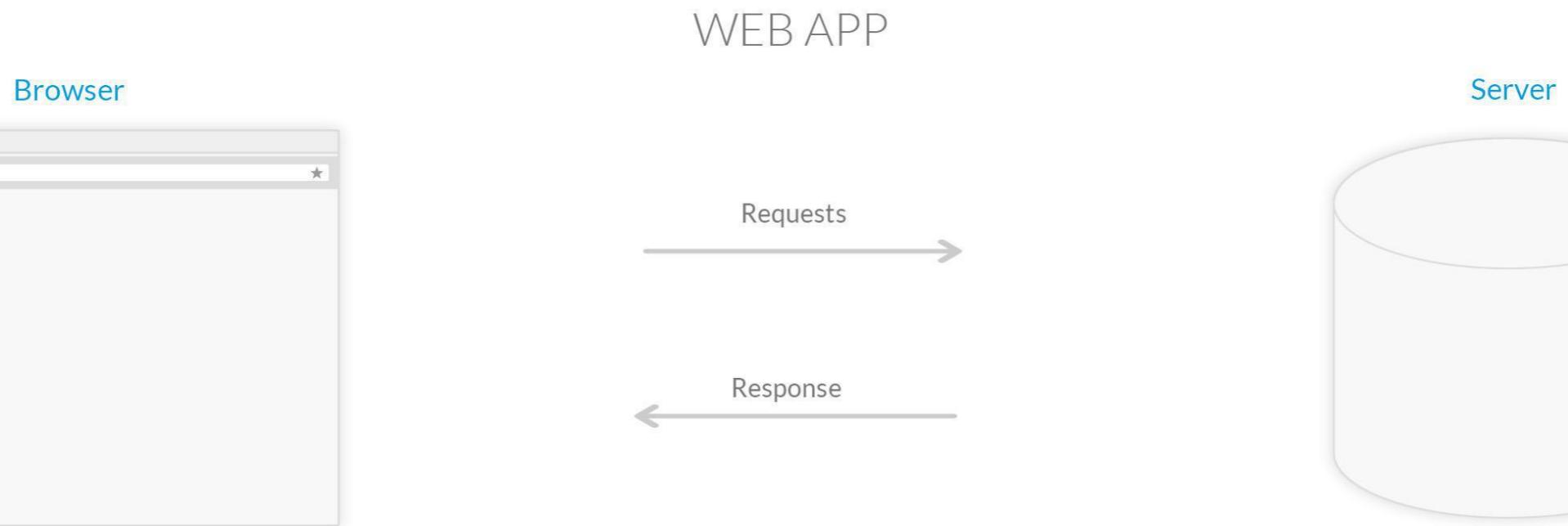
MUITO MAIS  
COMPLICADO



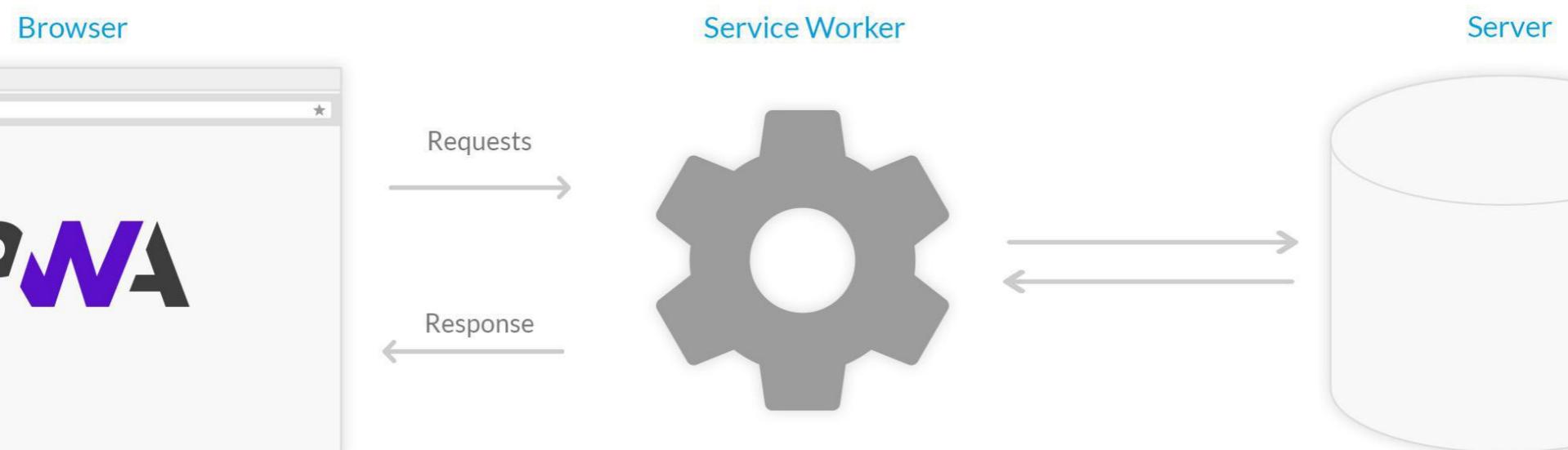
Google

moz://a

SAMSUNG

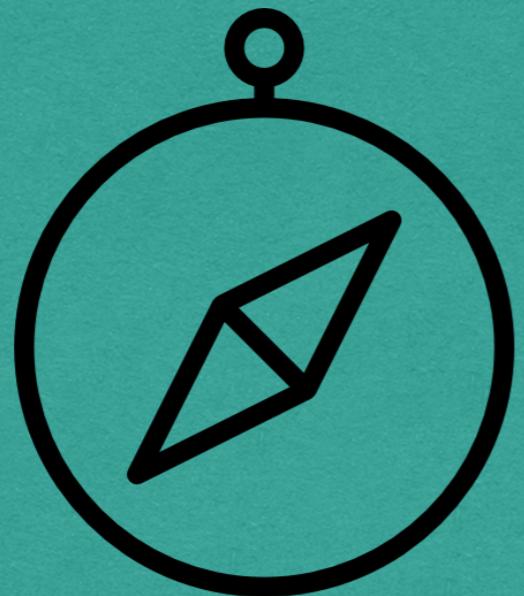
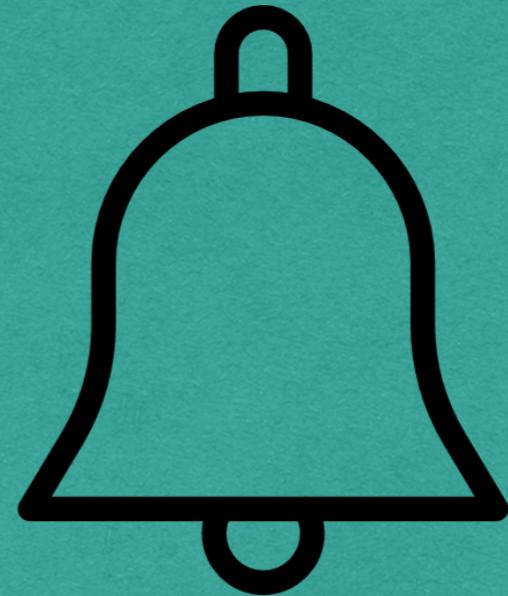
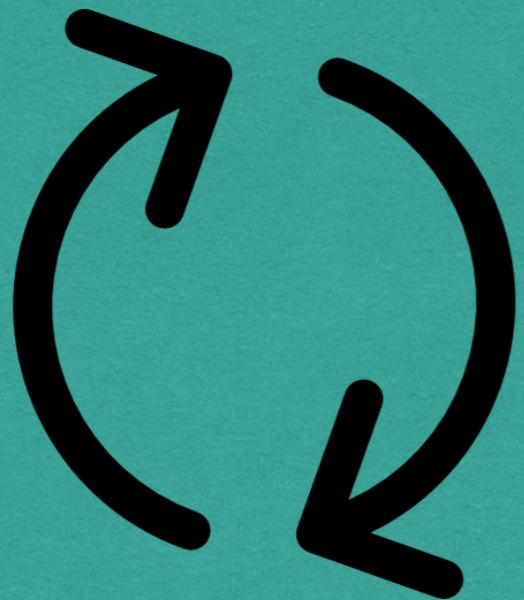


### PROGRESSIVE WEB APP

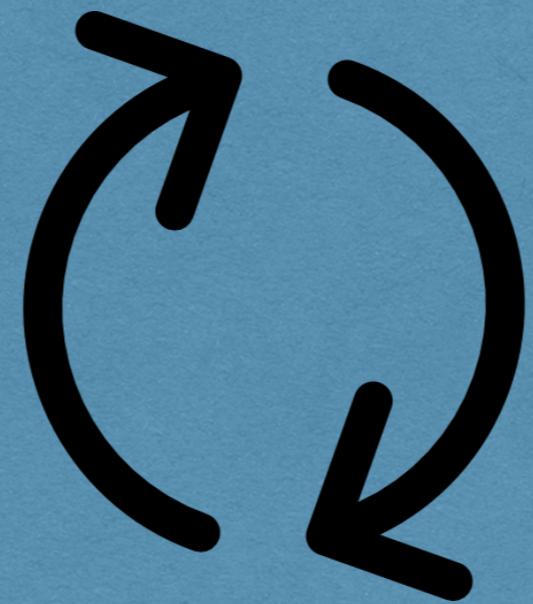




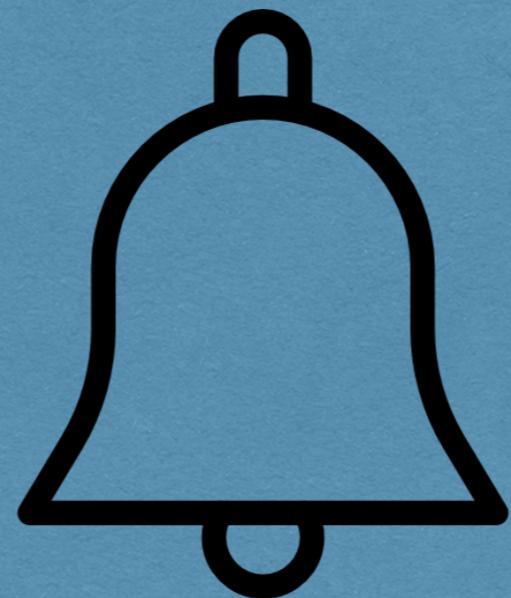
**POSSIBILIDADES**



# BACKGROUND SYNC



# PUSH NOTIFICATION





```
this.onpush = event => {  
}  
}
```



```
this.onpush = event => {  
    if(event.message.data == 'nova-mensagem') {  
    }  
}
```



```
this.onpush = event => {
  if(event.message.data == 'nova-mensagem'){
    event.waitUntil(
      atualizaMensagens().then(() => {
        }
      )
    }
}
```



```
this.onpush = event => {
  if(event.message.data == 'nova-mensagem'){
    event.waitUntil(
      atualizaMensagens().then(() => {
        new Notification("Você tem uma nova mensagem!");
      })
    )
  }
}
```

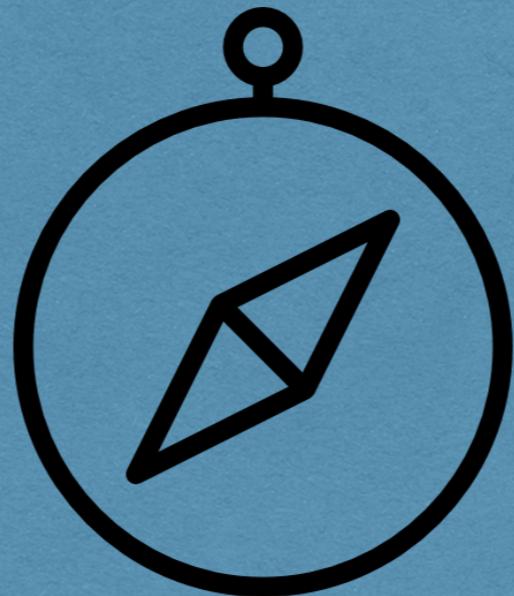


```
this.onnotificationclick = evento => {  
}
```

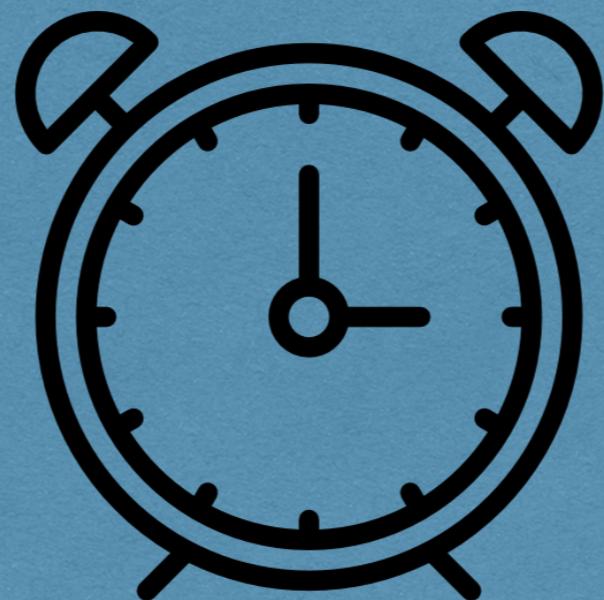


```
this.onnotificationclick = evento => {  
    new ServiceWorkerClient('/mensagens.html');  
}
```

# GEOFENCING



# ALARMES TEMPORAIS



Use AppCache como  
fallback

Progressive Enhancement



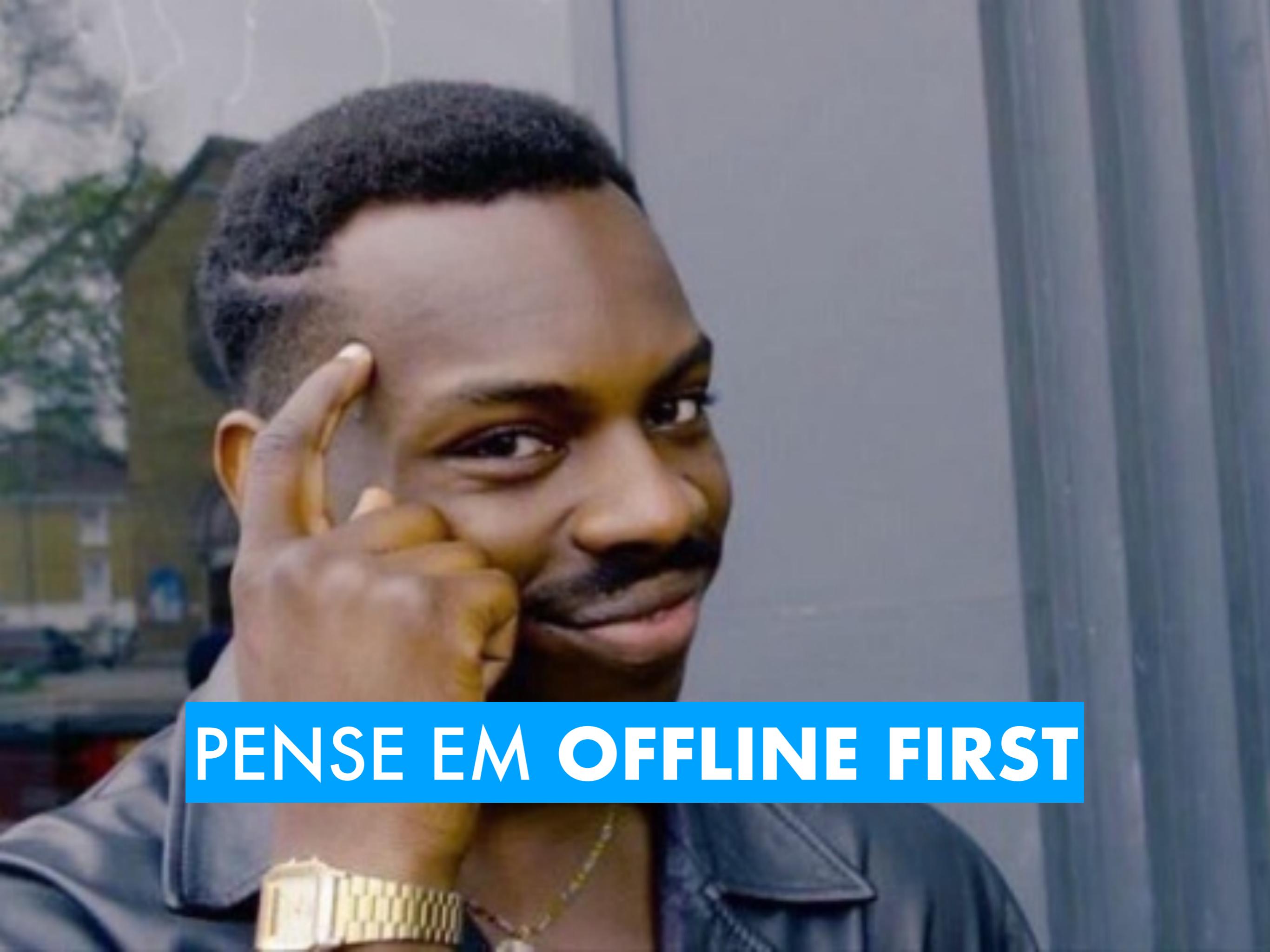
```
if ('serviceWorker' in navigator) {  
  
}
```



```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script>
      navigator.serviceWorker.register('offline.js');
    </script>
  </head>
  <body>
    <h1>Página Offline</h1>
  </body>
</html>
```



```
if ('serviceWorker' in navigator) {  
    navigator.serviceWorker.register('/sw.js').then(registration => {  
        console.log('Service worker registrado com sucesso:');  
    }).catch(function(error) {  
        console.log('Falha ao Registrar o Service Worker:', error);  
    })  
} else {  
    console.log('Service workers não suportado!');  
}
```



**PENSE EM OFFLINE FIRST**