ASYNCHROUNOUS JAVASCRIPT

# SERVICE-WORKER

# PENGERTIAN SERVICE-WORKER

- Thread yang berjalan mandiri (asynchronous), diciptakan oleh Browser
- Service Worker tidak mengenal DOM, karena berjalan terpisah dari Browser
- Service Worker dapat mengintersepsi jaringan, menulis dan membaca Cache
- Wervice Worker menyimpan data di indexDB (bukan localStorage)

# SERVICE-WORKER KOMPONEN DI NAVIGATOR

```
if (!('serviceWorker' in navigator)) {
  console.log('Browser tidak mendukung Service worker');
}
else {
   navigator.serviceWorker.register('service-worker.js')
   .then(function() {
     console.log('Service Worker terdaftar (registered)');
   })
   .catch(function(error) {
     console.log('Error: Gagal melakukan registrasi service workder:', error);
   });
}
```

# SCOPE UNTUK SERVICE-WORKER

```
navigator.serviceWorker.register('/service-worker.js', {
  scope: '/app/'
});
```

# TUJUAN SERVICE-WORKER

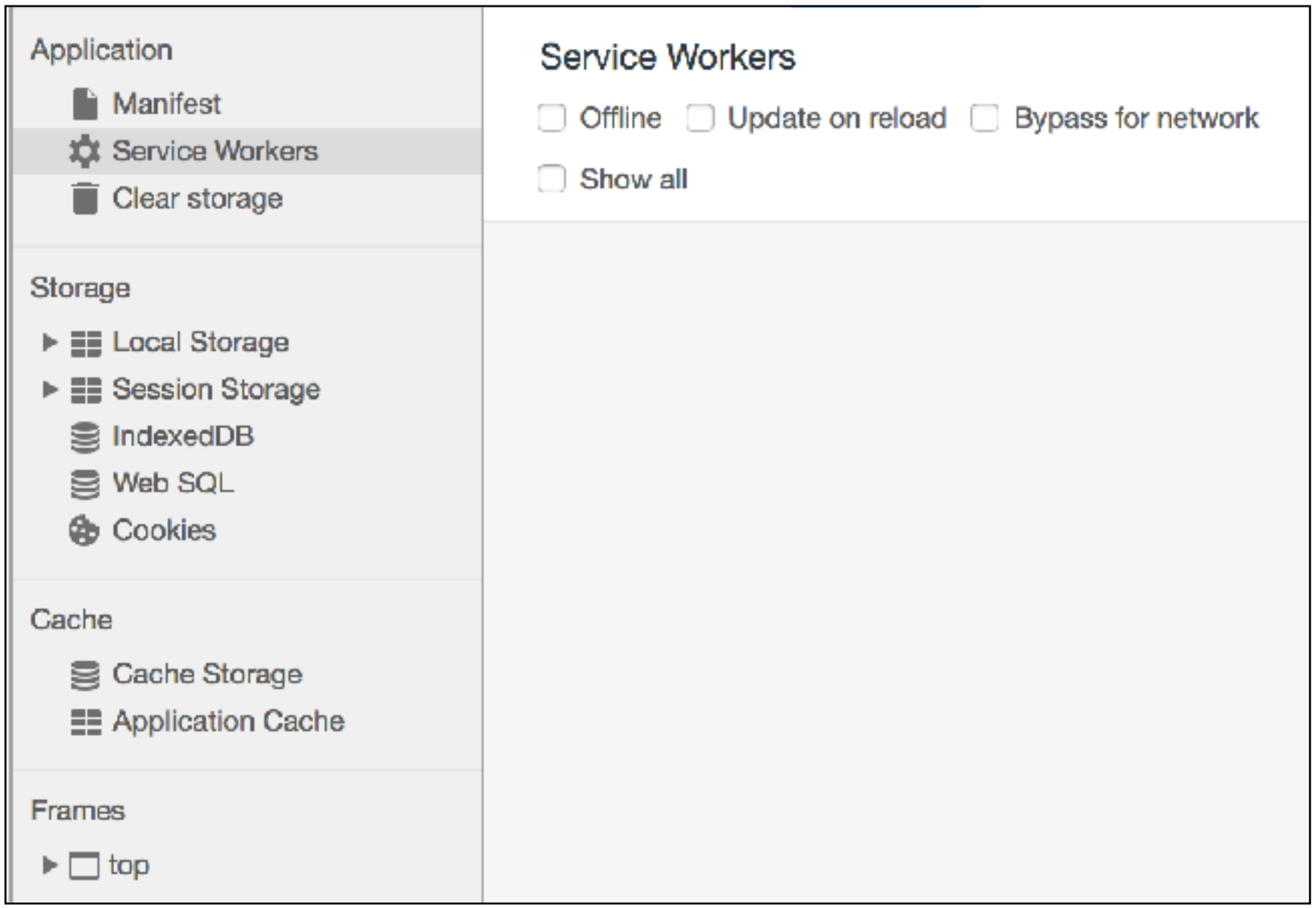
- Mengendalikan *network request*, kemudian menyimpan untuk sementara request tersebut di cache
- Memberikan fasilitas off-line dengan mengakses data dari cache.
- Mengintersepsi jaringan, kemudian memodifikasi response ke web-client, atau request ke web-server

# EVENT “INSTALL”

```
// Siapkan fungsi Callback untuk event "install"
self.addEventListener('install', function(event) {
    // Lakukan beberapa pekerjaan

});
```

# Melalui Developer's Tool, Application -> Service-Workers



# EVENT “ACTIVATE”

Setelah instalasi Service-Worker selesai, lakukan aktifasi (activate)

```
// Siapkan fungsi Callback untuk event "install"
self.addEventListener('install', function(event) {
    // Lakukan beberapa pekerjaan

});


self.addEventListener('activate', function(event) {
  // Lakukan aktifasi

});
```

# CONTOH: INSTALASI CACHE PADA SAAT INSTALL

```
self.addEventListener('install', function(event) {
    // Lakukan beberapa pekerjaan
    event.waitUntil(
       caches.create('static-v2').then(function(cache) {
          return cache.add({
              'myapp/f1',
              'myapp/f2',
              ....
          })
    )
});
```

# CONTOH: INTERSEPSI JARINGAN

```
self.addEventListener('fetch', function(event) {
  console.log('Fetching:', event.request.url);
  console.log(event.request);
  console.log(event.request.method);
  console.log(event.request.headers);

});
```

# CONTOH: HIJACK JARINGAN

```
// hijack the response
self.addEventListener('fetch', function(event) {
   event.respondWith(response);
});

//Construct self response
self.addEventListener('fetch', function(event) {
   event.respondWith(
      new Response("hi there")
   )
});
```

# AKSES CACHE

```
//response dari cache
self.addEventListener('fetch', function(event) {
   event.respondWith(
      caches.match(event.request)
   )
})

//Jika tidak ada match
self.addEventListener('fetch', function(event) {
   event.respondWith(
      caches.match(event.request)
      .catch(function() {
          return event.default;
      })
   )
})
```