Bài giảng

Phát triển ứng dụng web

Lê Đình Thanh

VNU-UET

Email: thanhld@vnu.edu.vn

Mobile: 0987.257.504

Website: https://uet.vnu.edu.vn/~thanhld

PWA

Progressive Web Application

Giới thiệu PWA

- PWA là ứng dụng web có khả năng khai thác các tính năng hiện đại của trình duyệt nhằm cải thiện dần trải nghiệm của người dùng
- Ba yếu tố trong khái niệm PWA
  1. PWA là ứng dụng web

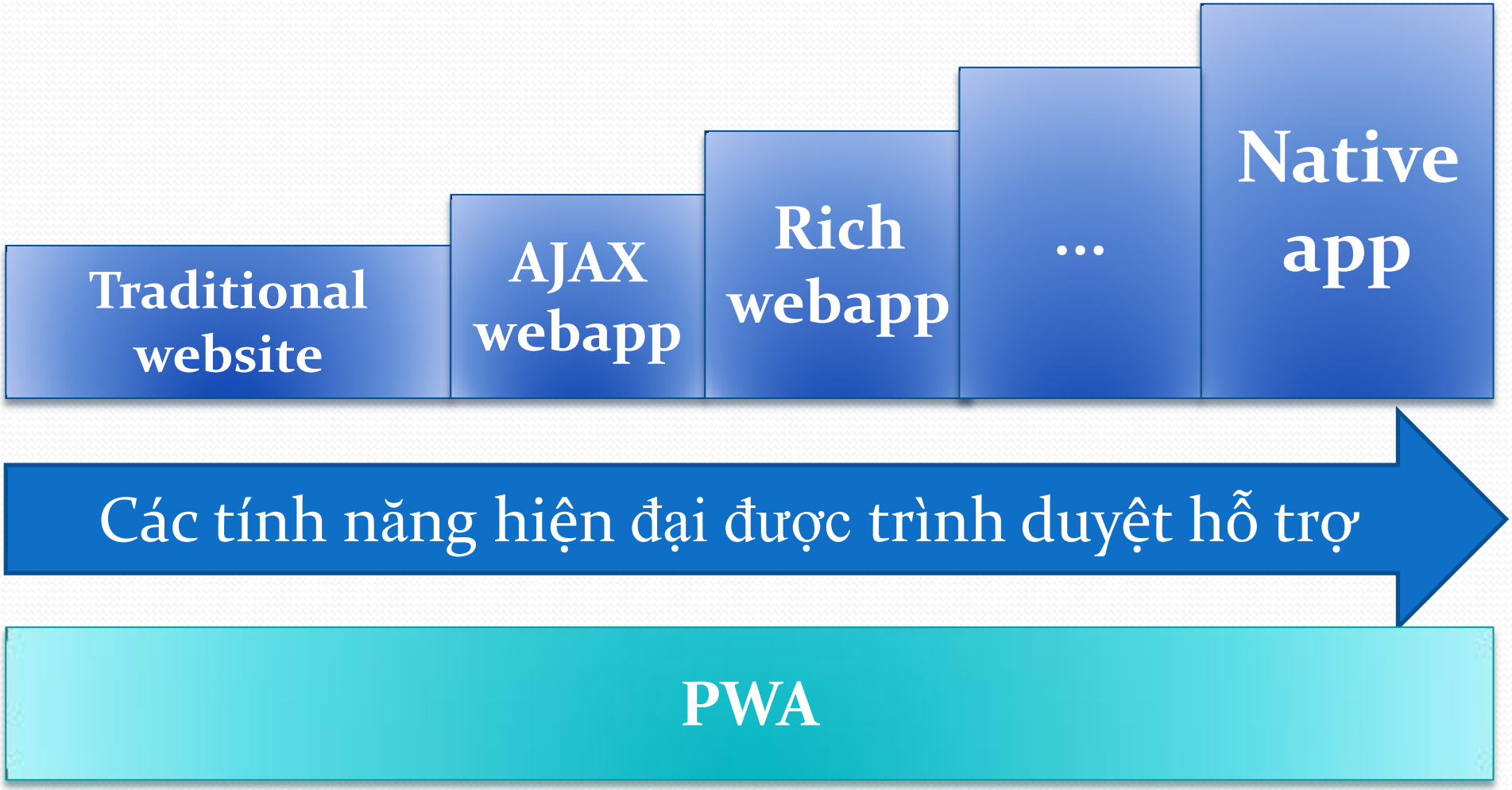
     PWA được phát triển bằng công nghệ web (HTML, JavaScipt, CSS) như ứng dụng web thông thường
  2. khai thác tính năng hiện đại của trình duyệt

     Service worker, fetch API, client cache, push notification, IndexedDB… gần đây mới được W3C đưa vào đặc tả
  3. cải thiện dần trải nghiệm của người dùng

Giới thiệu PWA

- Ba yếu tố trong khái niệm PWA (tiếp)
  3. cải thiện dần trải nghiệm của người dùng
     - Trình duyệt không hỗ trợ các tính năng hiện đại
       - PWA hoạt động như website thông thường
     - Trình duyệt hỗ trợ một phần các tính năng hiện đại
       - PWA tận dụng tính năng được hỗ trợ để làm cho trải nghiệm tốt hơn
     - Trình duyệt hỗ trợ tất cả các tính năng hiện đại
       - PWA hoạt động như native app

Trải nghiệm người dùng



Lợi ích của PWA

- Cho người dùng
  - Hiệu năng cao, hấp dẫn và đàn hồi
  - Như native app
    - đặt trên Home Screen, làm việc offline, ...
  - Không phải cài đặt
- Cho nhà phát triển
  - Một source code duy nhất
  - Không phải học công nghệ khác

… và

- Có thể chuyển website bất kỳ đang có trở thành PWA
  - bằng cách áp dụng các công nghệ web hiện đại

Dean Alan Hume, "Progressive Web Apps", page 6

"... Let's say your online business is a newspaper that people visit to discover more about their local area. If you know that people regularly visit your site and read multiple pages, why not cache those pages ahead of time for them so they can read the information completely offline? Or imagine your web app is for a charity that has volunteers working in areas with limited or no connectivity. The features of a PWA would allow you to build an offline app that lets them collect information in the field with no network connection. As soon as they come back to the office or to an area with connectivity, the data can sync back to the server.

Service Worker (SW)

SW

- Là thành phần quan trọng nhất để tạo PWA
- Là đối tượng JavaScript
- Chạy trong ngữ cảnh của worker toàn cục
  - chế độ nền
  - Luồng (thread) riêng độc lập với luồng chính của trang
  - Không đồng bộ (async)
  - Không gắn với trang web cụ thể nào
  - Không thể thay đổi DOM của trang web
- Chỉ chạy nếu sử dụng HTTPS

SW

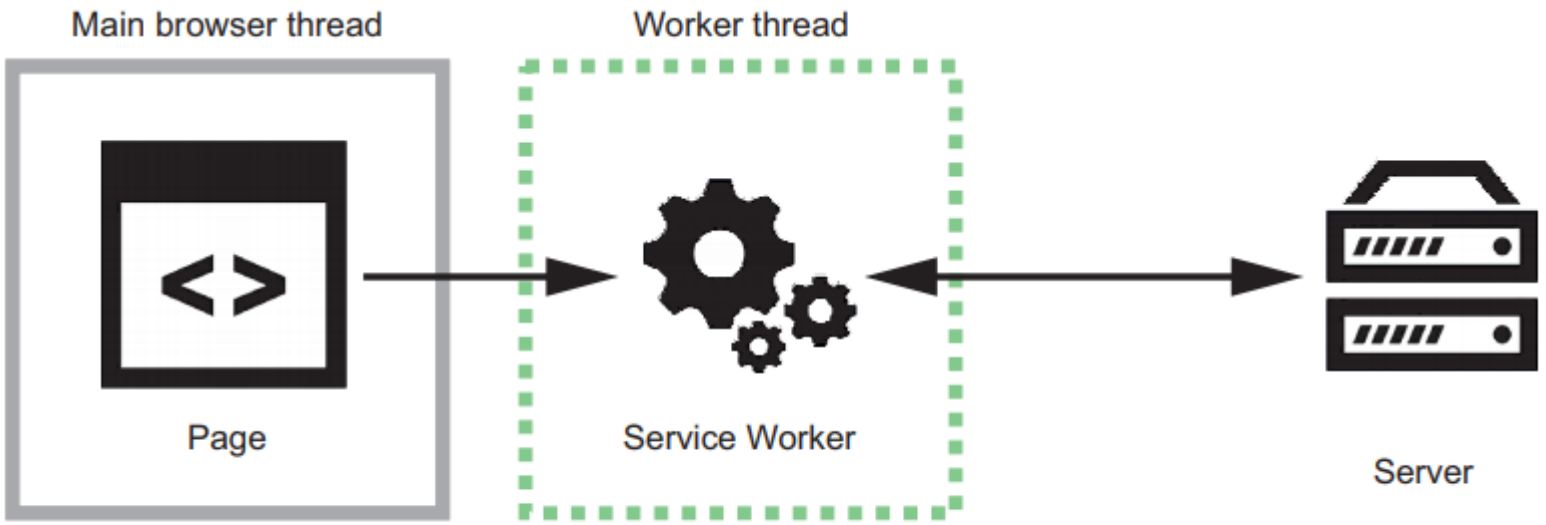


Figure 1.3 Service Workers can intercept incoming and outgoing HTTP requests, giving you total control of your website.

Nếu trình duyệt không hỗ trợ Service Workers, PWA hoạt động như website thông thường.

Đăng ký SW

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/serviceworker.js")
    .then(function(registration) {
      console.log("Service Worker registered with scope:", registration.scope);
    }).catch(function(err) {
      console.log("Service worker registration failed:", err);
    });
}
```

```
// These two commands will have the exact same scope:
navigator.serviceWorker.register("/sw.js");
navigator.serviceWorker.register("/sw.js", {scope: "/"});

// These two commands will register two service workers
// each controlling a different directory:
navigator.serviceWorker.register("/sw-ginnos.js", {scope: "/Ginnos"});
navigator.serviceWorker.register("/sw-ralphs.js", {scope: "/Ralphs"});
```

Vòng đời của SW

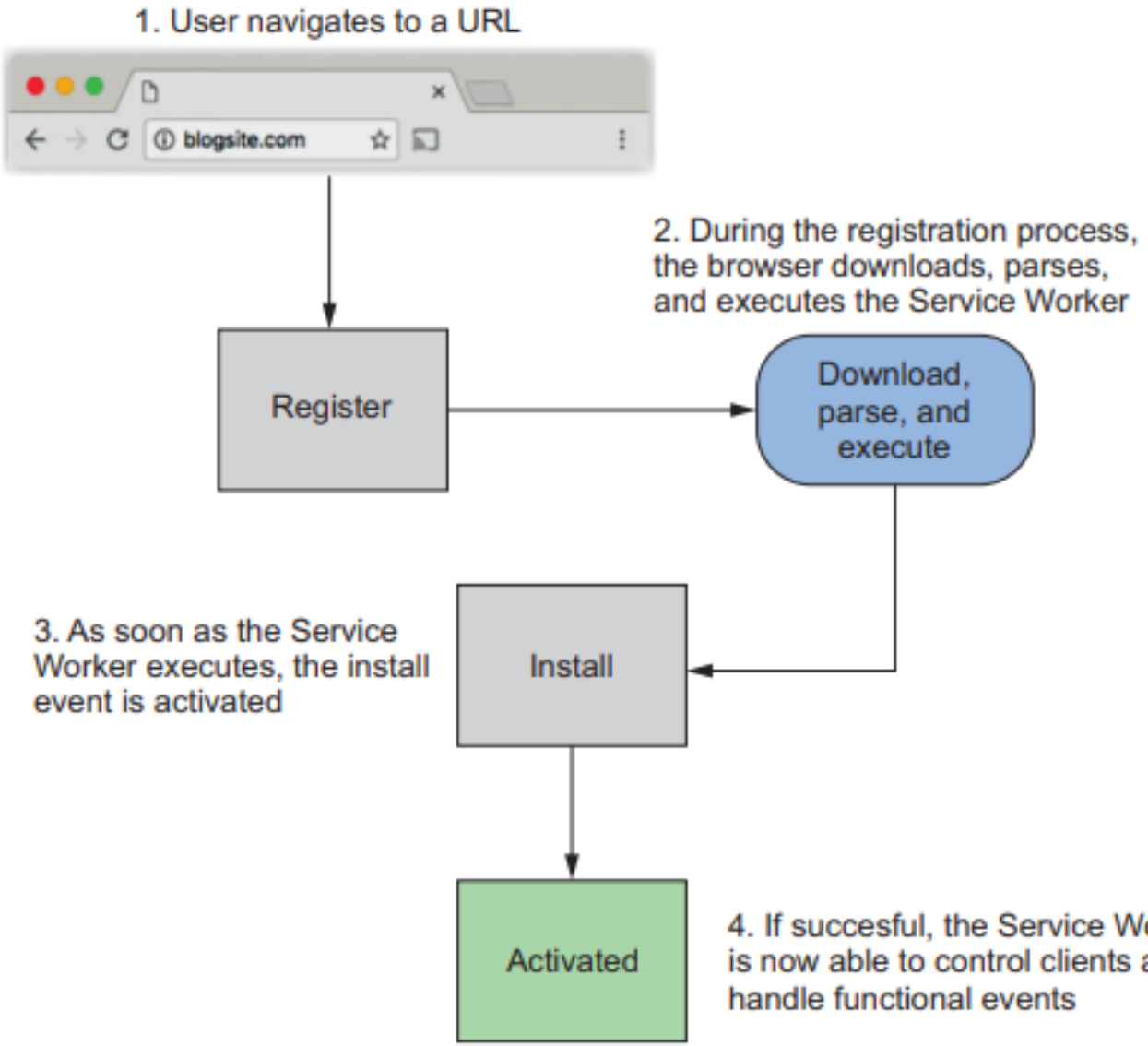


Figure 1.4 The Service Worker lifecycle

Response **Caching**

PWA Caching

- Một trong những đặc trưng quan trọng của PWA là lập trình được cache, do vậy hoàn toàn kiểm soát được cache và cho duyệt offline như native app
  - **CacheStorage** là cơ chế cho phép thực hiện điều này
- Kết hợp với SW, CacheStorage cho phép lập trình để
  - tạo và mở nhiều đối tượng cache
  - lưu, truy xuất và xóa các **response** trong cache
  - quyết định response nào lấy trong cache, response nào cần lấy trên server

Thời điểm cache

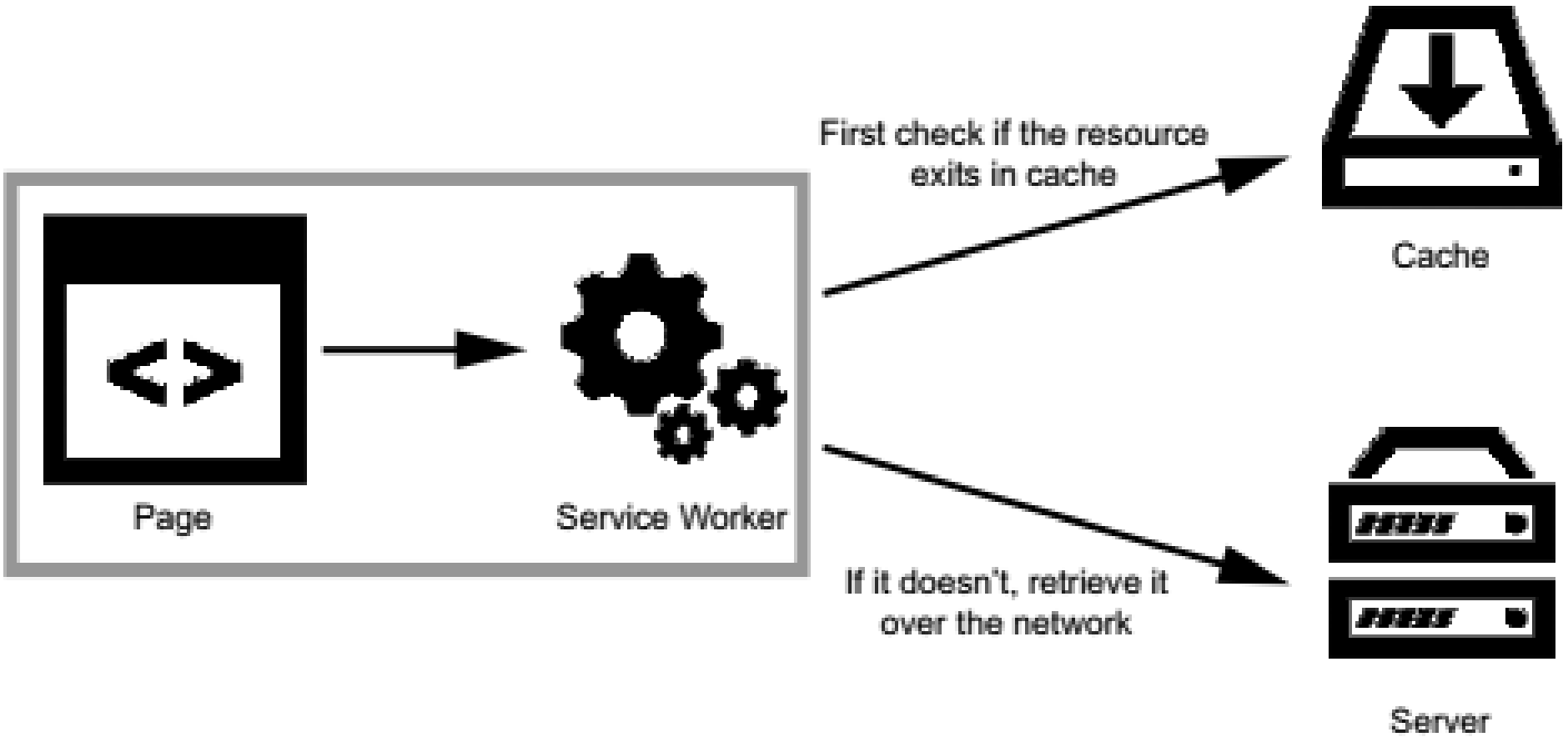
- Precache
  - Download và cache những files cần cho SW và khởi chạy ứng dụng
  - Được thực hiện trước khi SW active và nắm điều khiển
  - Tại sự kiện install của SW
- Interncept and Cache
  - Cache các response trong quá trình người dùng tương tác
  - Tại sự kiện fetch

Precache

```
var CACHE_NAME = "gih-cache";
var CACHED_URLS = [
  "/index-offline.html",
  "https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
  "/css/gih-offline.css",
  "/img/jumbo-background-sm.jpg",
  "/img/logo-header.png"
];

self.addEventListener("install", function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME).then(function(cache) {
      return cache.addAll(CACHED_URLS);
    })
  );
});
```

Interncept and Cache



- Cache-first <> Network-first

Interncept and Cache

```
// Cache any new resources as they are fetched
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request, { ignoreSearch: true })
    .then(function(response) {
      if (response) {
        return response;
      }
      var requestToCache = event.request.clone();

      return fetch(requestToCache).then(
        function(response) {
          if(!response || response.status !== 200) {
            return response;
          }

          var responseToCache = response.clone();
          caches.open(cacheName)
          .then(function(cache) {
            cache.put(requestToCache, responseToCache);
          });

          return response;
        });
    })
  );
});
```

Listen for the fetch event.

Ignore any querystring parameters so you don't get any cache misses.

If you found a successful match, return it at this point and go no further.

If you didn't find anything in cache, make the request

Store it in cache so we won't need to make that request again

Cập nhật cache

- Tài nguyêncó thể được cập nhật phía server
  - => Cần cập nhật cache ở browser
- Cập nhật tệp cài đặt SW
  - SW tự động re-download và cập nhật chính nó (tệp js cài đặt logic của SW) ở chế độ nền nếu tệp cài đặt SW thay đổi.
- Cập nhật các tài nguyên
  - Sử dụng tên khác cho cache

Can thiệp HTTP request

Can thiệp request

Listing 4.9 Service Worker Code to Return WebP Images if the Browser Supports It

```
"use strict";

// Listen to fetch events
self.addEventListener('fetch', function(event) {

  if (/\.jpg$|.png$/.test(event.request.url)) {

    var supportsWebp = false;
    if (event.request.headers.has('accept')) {
      supportsWebp = event.request.headers
       .get('accept')
       .includes('webp');
    }

    if (supportsWebp) {
      var req = event.request.clone();

      var returnUrl = req.url.substr(0, req.url.lastIndexOf(".")) + ".webp";

      event.respondWith(
        fetch(returnUrl, {
          mode: 'no-cors'
        })
      );
    }
```

Check whether the incoming request is for an image of type JPEG or PNG.

Inspect the accept header for WebP support.

Does the browser support WebP?

Can thiệp HTTP request

Listing 4.10 Service Worker code to check for the save-data HTTP header

```
"use strict";

this.addEventListener('fetch', function (event) {

  if(event.request.headers.get('save-data')){
    // We want to save data, so restrict icons and fonts
    if (event.request.url.includes('fonts.googleapis.com')) {
        // return nothing
        event.respondWith(new Response('', {status: 417, statusText: 'Ignore
     fonts to save data.' }));
    }
  }
});
```

Duyệt offline

Offline

- Khi mất kết nối đến máy chủ, HTTP request fails

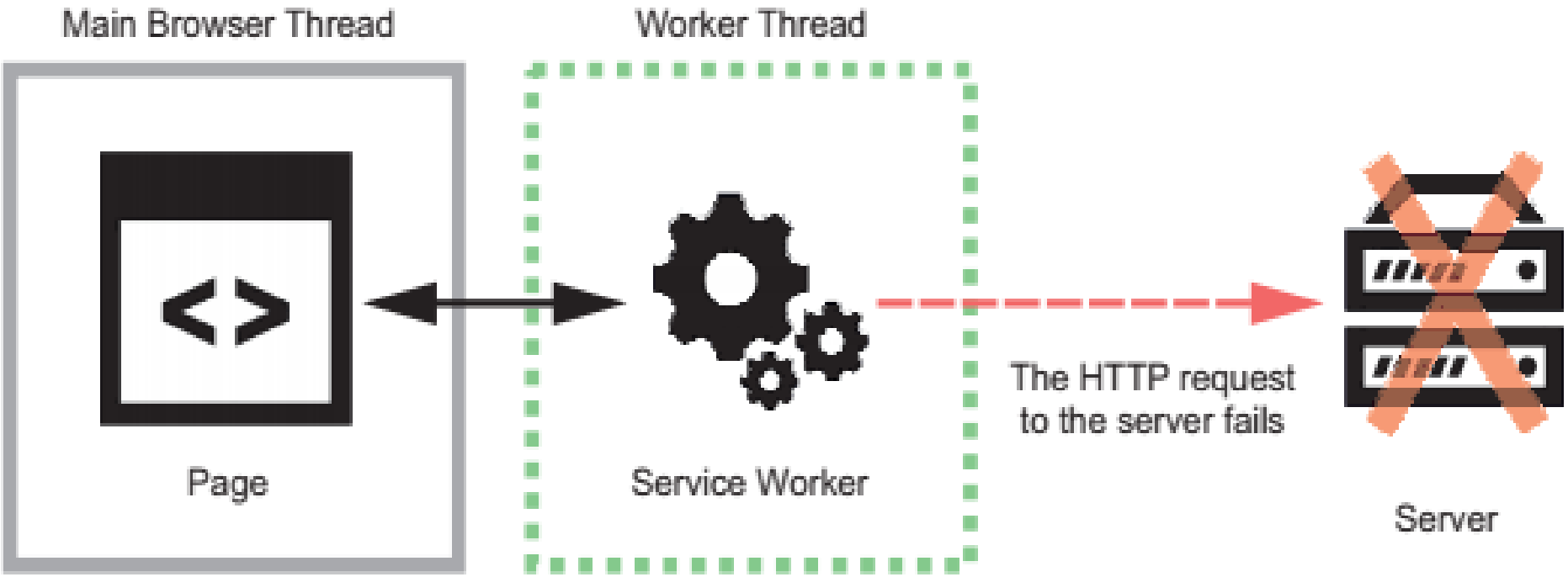
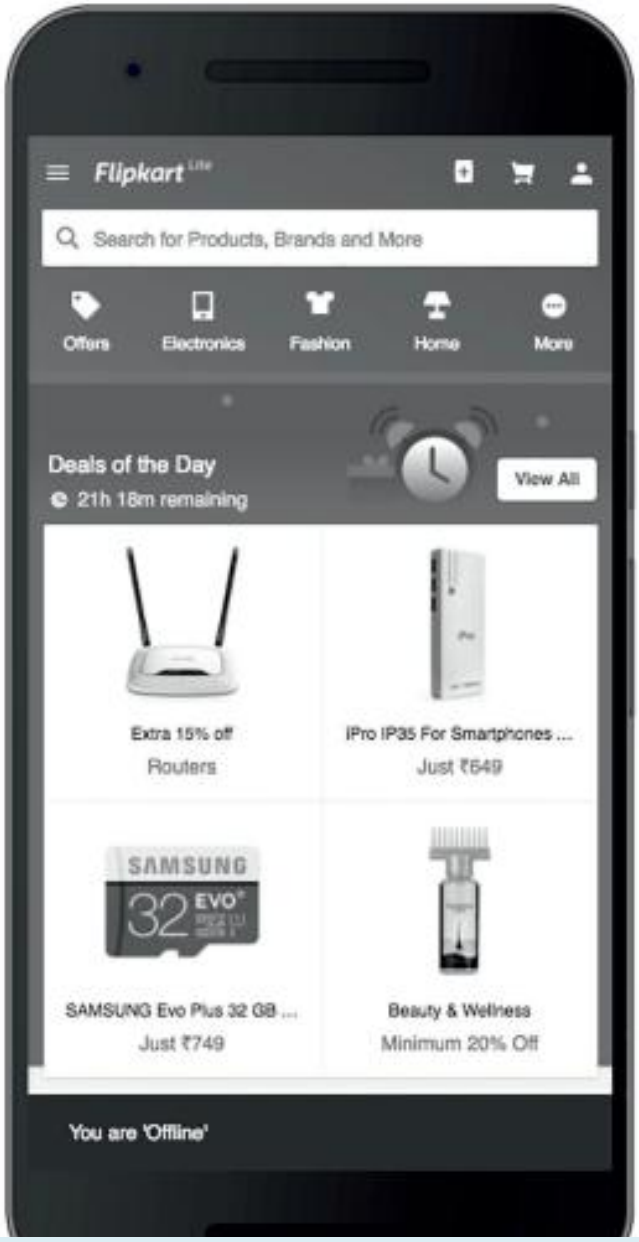


Figure 7.1 Using a Service Worker, you can determine whether the user is attempting to fetch a resource while they're offline.

Các tình huống offline

- Tìm được response trong cache
  - Hiển thị trang theo response tìm thấy + **thông báo tình trạng offline**
- Không tìm được bản response trong cache
  - Hiển thị trang offline
    - Trang offline phải được precache từ lúc cài đặt

Tìm được response trong cache



Thông báo tình trạng offline

```
var offlineNotification = document.getElementById('offline');

function showIndicator() {
  offlineNotification.innerHTML = 'You are currently offline.';
  offlineNotification.className = 'showOfflineNotification';
}

function hideIndicator() {
  offlineNotification.className = 'hideOfflineNotification';
}

window.addEventListener('online',  hideIndicator);    #C
window.addEventListener('offline', showIndicator);    #D
```

This function is used to show an offline notification when the user is offline.

This function is used to hide the offline notification when the user is back online.

Không có response trong cache

```
self.addEventListener('fetch', event => {

  event.respondWith(caches.match(event.request).then(function (response) {
    if (response) {
      return response;
    }
    var fetchRequest = event.request.clone();

    return fetch(fetchRequest).then(function (response) {
      if (!response || response.status !== 200) {
        return response;
      }

      var responseToCache = response.clone();
      caches.open(cacheName).then(function (cache) {
        cache.put(event.request, responseToCache);
      });

      return response;
    }).catch(error => {
      if (event.request.method === 'GET' &&
     event.request.headers.get('accept').includes('text/html')) {
        return caches.match(offlineUrl);
      }
    });
  }));
```

Event listener for the fetch event

If fetch event failed for any reason, check if it was for an HTML page and if user was trying to navigate to another page.

Return offline page you stored in cache during the Service Worker install step

Precache trang offline

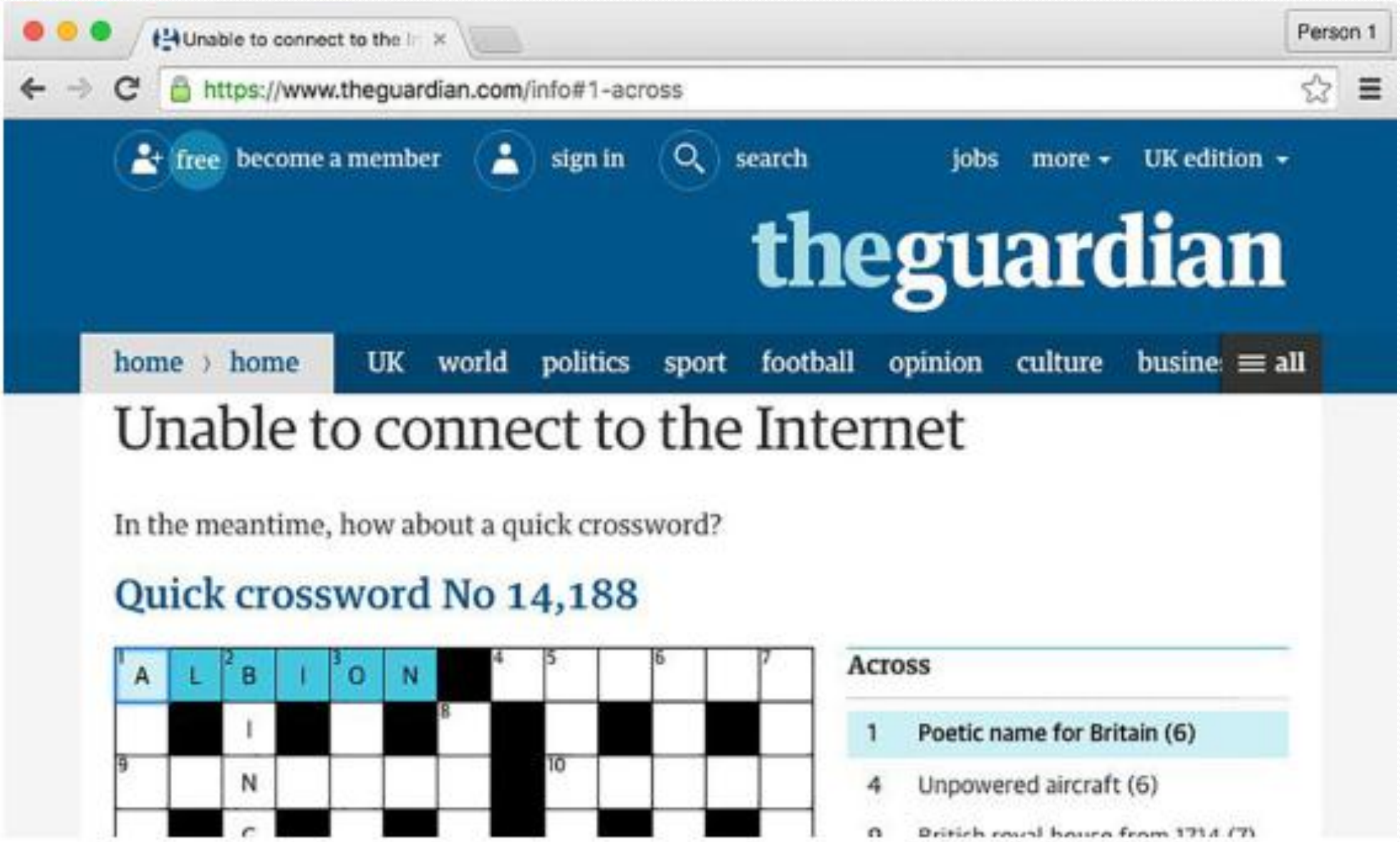
```
const cacheName = 'latestNews-v1';
const offlineUrl = 'offline-page.html';

self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(cacheName)
    .then(cache => cache.addAll([
      './js/main.js',
      './js/article.js',
      './images/newspaper.svg',
      './css/site.css',
      './data/latest.json',
      './data/data-1.json',
      './article.html',
      './index.html',
      offlineUrl
    ]))
  );
});
```

URL of the offline web page

Add in the offline page during Service Worker install.

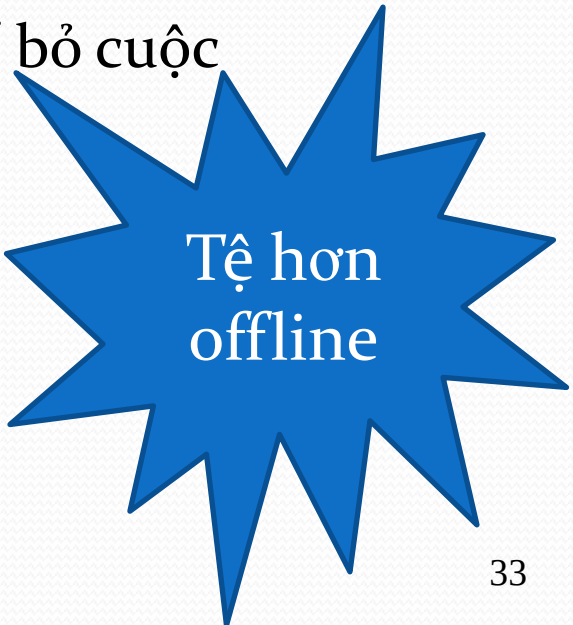
Làm cho trang offline bớt buồn tẻ



Xử lý lỗi kết nối

Lỗi kết nối

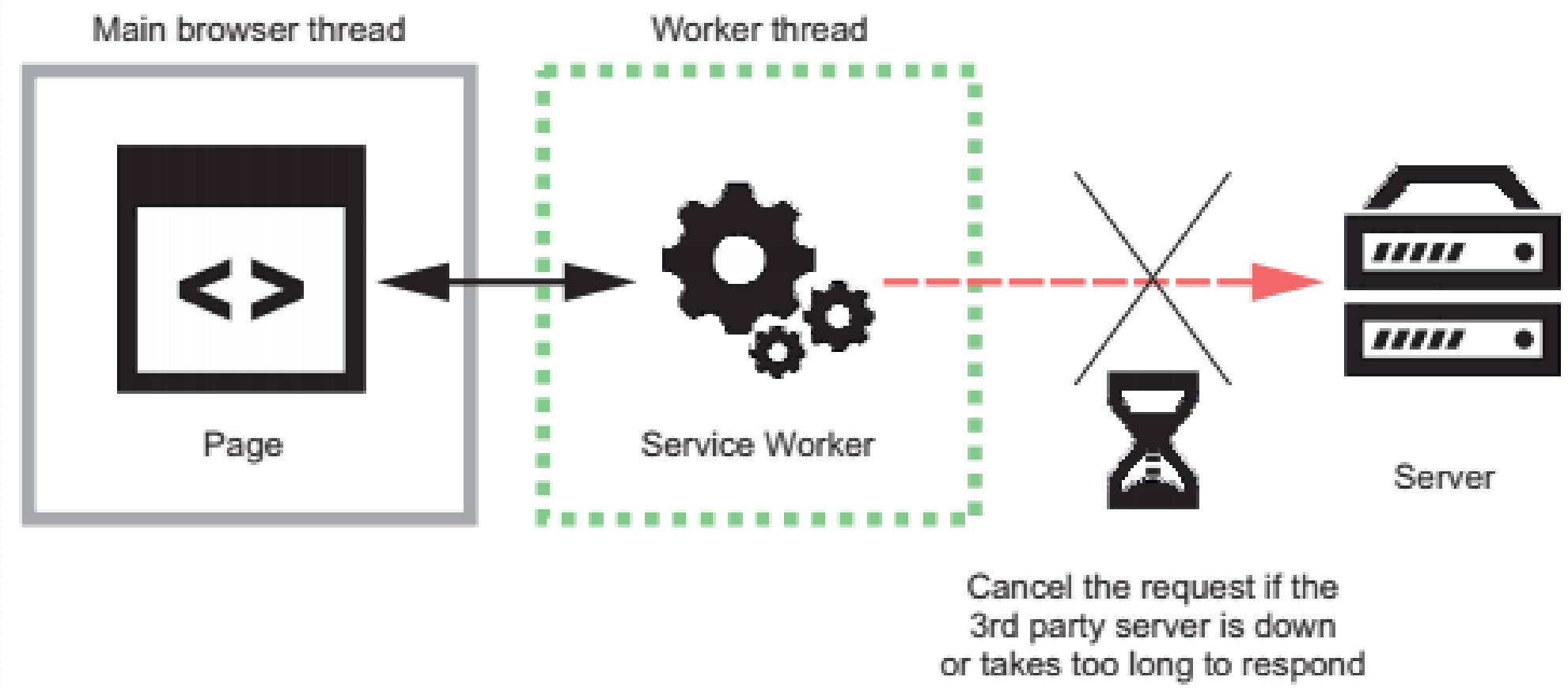
- Các tình huống lỗi kết nối
  - Lie-fi
    - Mạng chập chờn
    - Sóng wifi mạnh, LAN tốt nhưng không kết nối được Internet
  - Server down
- Tác hại của lỗi kết nối
  - Trình duyệt vẫn cố chờ để nhận response
    - Thời gian đáp ứng chậm, người dùng có thể bỏ cuộc
  - SPOF
    - Ứng dụng không thể chạy tiếp
  - Mất dữ liệu người dùng đã nhập
    - Khi người dùng submit form



Cách thức xử lý

- Chậm đáp ứng hoặc SPOF
  - Cho service worker hủy request nếu phải chờ quá lâu, tạo response 408 thay thế.
- Mất dữ liệu khi submit form
  - Lưu dữ liệu cần submit vào **IndexedDB**
  - Sử dụng **BackgroundSync** gửi POST request với dữ liệu trong IndexedDB

Hủy request nếu phải chờ lâu



Hủy request nếu phải chờ lâu

```
function timeout(delay) {
    return new Promise(function (resolve, reject) => {
        setTimeout(function(){
          resolve(new Response('', {
              status: 408,
              statusText: 'Request timed out.'
          }));
        }, delay);
    });
};

self.addEventListener('fetch', function(event) {
if (/googleapis/.test(event.request.url)) {
     event.respondWith(Promise.race([timeout(3000),fetch(event.request.url)]));
} else {
     event.respondWith(fetch(event.request));
}
});
```

Classic timeout function that has been updated to return as a promise

Execute the timeout function using the passed in delay (amount in milliseconds).

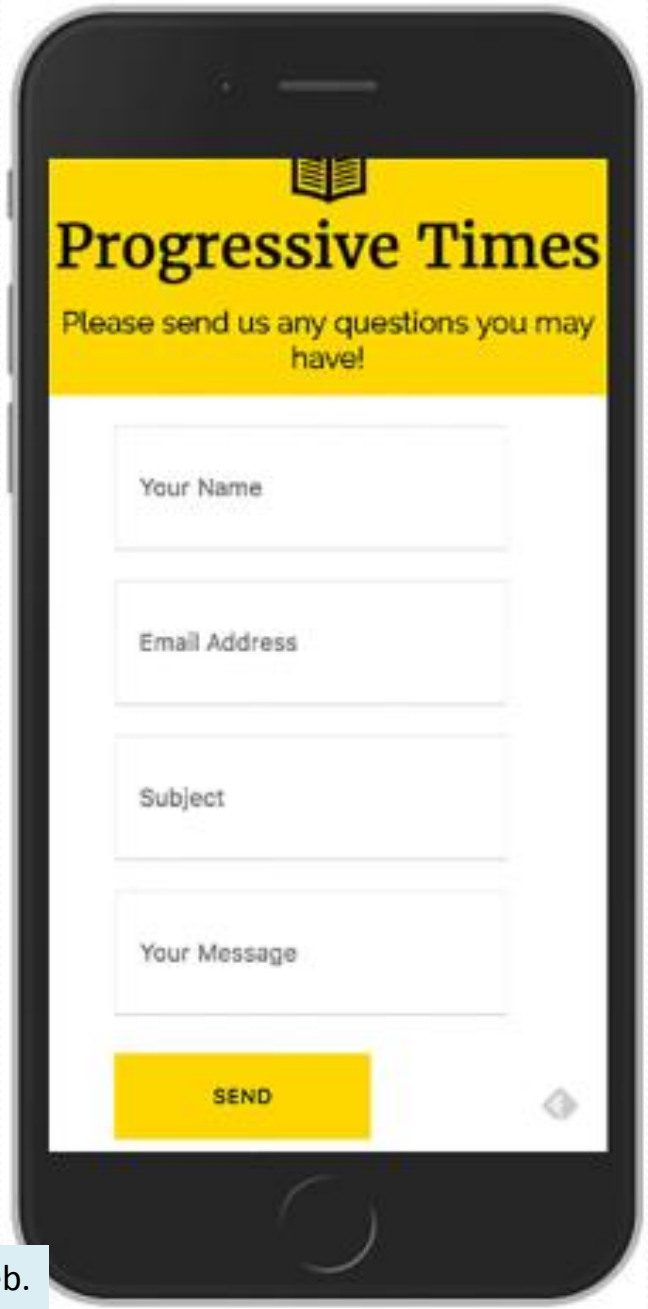
If timeout occurs, return a new Response of status code 408 and provide a message.

Use a Promise.race condition to fire both the timeout and fetch function at the same time.

Otherwise make the request as expected.

Only run this check for requests to the domain **googleapis.com** .

Mất kết nối khi submit form



POST request trực tiếp từ trang

```
document.getElementById('submit').addEventListener('click', () => {

var payload = {

  name: document.getElementById('name').value,
  email: document.getElementById('email').value,
  subject: document.getElementById('subject').value,
  message: document.getElementById('message').value,
};

fetch('/sendMessage/',
{
    method: 'post',
    headers: new Headers({
      'content-type': 'application/json'
    }),
    body: JSON.stringify(payload)
})
.then(displayMessageNotification('Message sent'))
.catch((err) => displayMessageNotification('Message failed'));
```

Try to send message using traditional method

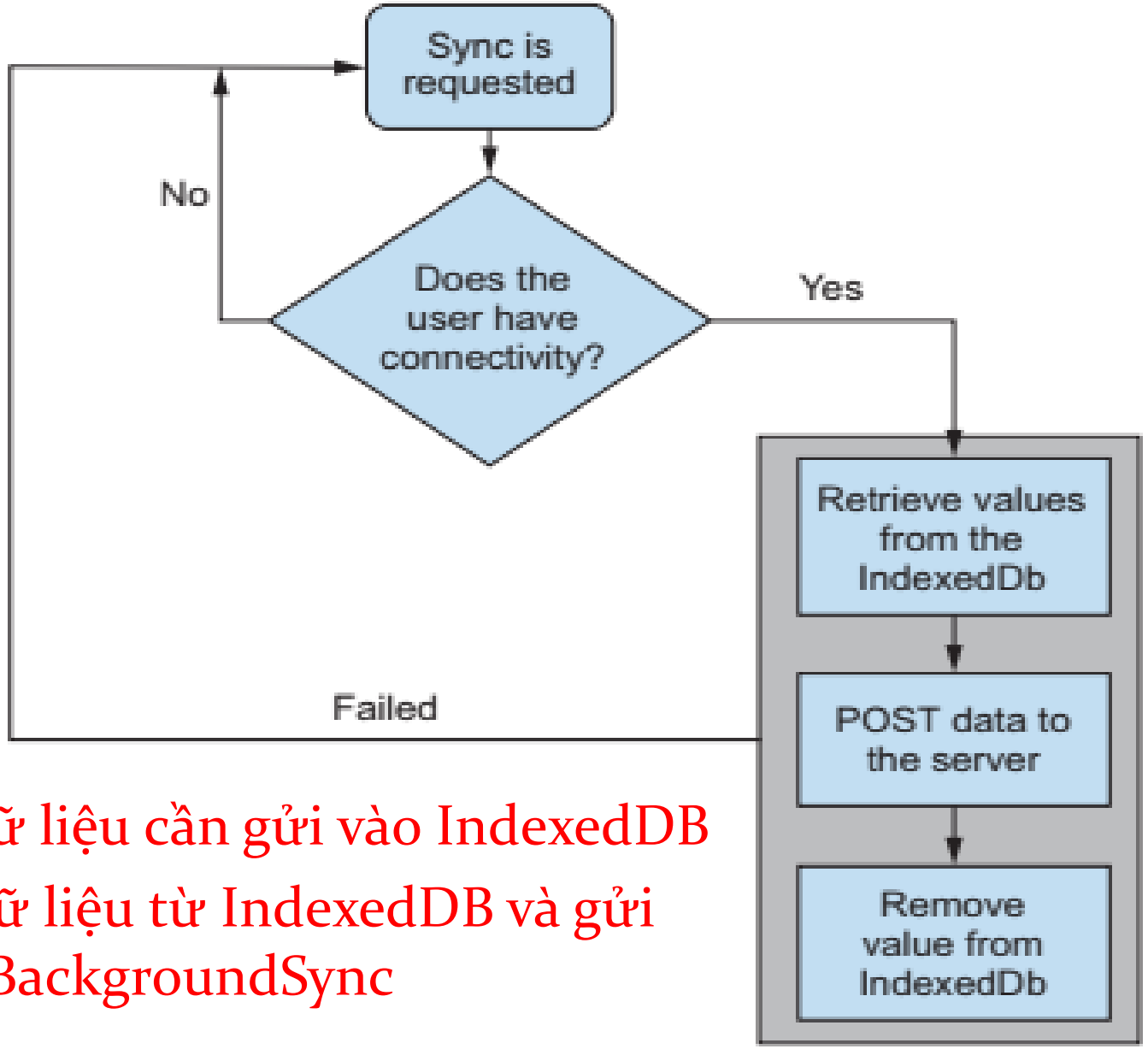
Show notification and let user know the notification was sent

- Mất dữ liệu của người dùng => Ứng dụng không tin cậy

Đảm bảo POST request thành công

- Không submit form trực tiếp từ trang mà
  - Lưu dữ liệu form vào IndexedDB
  - Sử dụng BackgroundSync để gửi POST request với dữ liệu từ IndexedDB
- IndexedDB
  - Lưu bền vững dữ liệu ngay tại client
    - Sẽ xóa dữ liệu sau khi gửi thành công (nếu cần)
- BackgroundSync
  - chạy ở chế độ nền, chạy ngay cả khi người dùng chuyển trang hoặc đóng trình duyệt
  - tự gửi lại request cho đến khi request thành công
    - **-> ĐẢM BẢO 100% gửi thành công kể cả khi mạng chập chờn hoặc offline -> TIN CẬY**

Đảm bảo POST request thành công



- Lưu dữ liệu cần gửi vào IndexedDB
- Đọc dữ liệu từ IndexedDB và gửi bằng BackgroundSync

IndexedDB

- Là CSDL giao tác
  - Mọi thao tác CSDL đều thực hiện theo giao tác
    - Giao tác có tính ACID (Atomicity, Consistency, Isolation, Durability)
- Lưu trữ các đối tượng
  - Dữ liệu được lưu trữ là đối tượng: Bất kỳ thứ gì mà JavaScript xử lý được như JavaScript objects, booleans, numbers, blobs, …
  - Tương tự NoSQL
- Được đánh chỉ mục
  - Các cặp <key, value>
- Chạy hoàn toàn trong trình duyệt

IndexedDB (tiếp)

- Có thể tạo nhiều CSDL
- Mỗi CSDL có nhiều object store
- Object store lưu các cặp <key, value>
- value có thể là bất kỳ dữ liệu gì trong JavaScipt: objects, numbers, booleans, strings, dates, arrays, regular expressions, undefined, and null.
- Tuân thủ SOP (Same Origin Policy)
- Hầu hết các thao tác đều không đồng bộ
  - sử dụng callback để nhận kết quả

Thao tác trên IndexedDB

- Bước 1. Mở kết nối CSDL
- Bước 2. Mở giao dịch
- Bước 3. Mở object store
- Bước 4. Đọc hoặc ghi dữ liệu vào các object store
- Bước 5. Hoàn tất giao dịch

Mở CSDL và tạo object store

```
var db;

function getDB() {
  if (!db) {
    db = new Promise(function(resolve, reject) {
      var openreq = indexedDB.open('keyval-store', 1);
      openreq.onerror = function() {
        reject(openreq.error);
      };
      openreq.onupgradeneeded = function() {
        // Create an empty object store
        openreq.result.createObjectStore('keyval');
      };
      openreq.onsuccess = function() {
        resolve(openreq.result);
      };
    });
  }
  return db;
}
```

Mở giao dịch và object store

```
function withStore(type, callback) {
  return getDB().then(function(db) {
    return new Promise(function(resolve, reject) {
      var transaction = db.transaction('keyval', type);
      transaction.oncomplete = function() {
        resolve();
      };
      transaction.onerror = function() {
        reject(transaction.error);
      };
      callback(transaction.objectStore('keyval'));
    });
  });
}
```

Thao tác dữ liệutrên object store

```
var idbKeyval = {
  get: function(key) {
    var req;
    return withStore('readonly', function(store) {
      req = store.get(key);
    }).then(function() {
      return req.result;
    });
  },
  set: function(key, value) {
    return withStore('readwrite', function(store) {
      store.put(value, key);
    });
  },
  delete: function(key) {
    return withStore('readwrite', function(store) {
      store.delete(key);
    });
  },
  clear: function() {
    return withStore('readwrite', function(store) {
```

Thao tác dữ liệutrên object store

```
    keys: function() {
      var keys = [];
      return withStore('readonly', function(store) {
        // This would be store.getAllKeys(), but it isn't supported by Edge or
Safari.
        // And openKeyCursor isn't supported by Safari.
        (store.openKeyCursor || store.openCursor).call(store).onsuccess =
function() {
          if (!this.result) return;
          keys.push(this.result.key);
          this.result.continue();
        };
      }).then(function() {
        return keys;
      });
    }
  };
```

Lưu dữ liệu vào IndexedDB và đăng ký sync

Check to see whether current browser supports Service Workers and SyncManager.

```
<script src="/js/idb-keyval.js" ></script>
<script>
  if ('serviceWorker' in navigator && 'SyncManager' in window) {
    navigator.serviceWorker.register('./sw.js')
    .then(registration => navigator.serviceWorker.ready)

    .then(registration => {
      document.getElementById('submit').addEventListener('click', () => {

        registration.sync.register('contact-email').then(() => {

  var payload = {
    name: document.getElementById('name').value,
    email: document.getElementById('email').value,
    subject: document.getElementById('subject').value,
    message: document.getElementById('message').value,
  };

  idbKeyval.set('sendMessage', payload);

        });
```

Gửi trực tiếp nếu trình duyệt không hỗ trợ BackgroundSync

```
if ('serviceWorker' in navigator && 'SyncManager' in window) {

} else {

document.getElementById('submit').addEventListener('click', () => {
var payload = {
  name: document.getElementById('name').value,
  email: document.getElementById('email').value,
  subject: document.getElementById('subject').value,
  message: document.getElementById('message').value,
};

fetch('/sendMessage/',
{
    method: 'POST',
    headers: new Headers({
      'content-type': 'application/json'
    }),
    body: JSON.stringify(payload)

  });
});
```

submit button.

Get the values from the input fields on the contact form.

Use the fetch API to post the values to the server.

Bắt và xử lý sự kiện sync trong SW

```
importScripts('./js/idb-keyval.js');
self.addEventListener('sync', (event) => {
  if (event.tag === 'contact-email') {
    event.waitUntil(
      idbKeyval.get('sendMessage').then(value =>
        fetch('/sendMessage/', {
          method: 'POST',
          headers: new Headers({ 'content-type': 'application/json' }),
          body: JSON.stringify(value)
        })));

        idbKeyval.delete('sendMessage');
    }
});
```

Add an event listener for the sync event.

Check the tag of the current sync to ensure that you fire the correct code.

Get the payload values from the IndexedDB .

Use fetch API to POST to the server.

Pass in the payload values that you retrieved from the IndexedDB.

Remove the payload values from the IndexedDB.

Look and feel

Web app manifest

- Tệp manifest.json
- Cung cấp thông tin về ứng dụng: Tên, icon, mô tả, ...
- Cho phép cài đặt ứng dụng trên Home Screen của thiết bị

manifest.json

```
{
  "name": "Progressive Times web app",
  "short_name": "Progressive Times",
  "start_url": "/index.html",
  "display": "standalone",
  "theme_color": "#FFDF00",
  "background_color": "#FFDF00",
  "icons": [
    {
      "src": "homescreen.png",
      "sizes": "192x192",
      "type": "image/png"
    },
    {
      "src": "homescreen-144.png",
      "sizes": "144x144",
      "type": "image/png"
    }
  ]
```

Bao hàm manifest.json

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Progressive Times</title>
    <link rel="manifest" href="/manifest.json">
  </head>
```

The web app manifest file is referenced between the head tags in a web page.

Cài đặt ứng dụng lên Home Screen

Các thuộc tính manifest

- name: Tên của ứng dụng xuất hiện tại màn hình khởi động
- short_name: Tên xuất hiện trên Home Screen
- icons: Ảnh đại diện xuất hiện trên Home Screen
- theme_color: Màu tùy biến cho thanh địa chỉ của trình duyệt
- start_url: Trang đầu tiên được mở khi chạy ứng dụng từ Home Screen
- background_color: Màu nền được sử dụng tại màn hình khởi động
- display: Kiểu hiển thị ứng dụng

Tùy biến icons

- Có thể cung cấp nhiều icons để ứng dụng chọn cho phù hợp với thiết bị và n

```
"icons": [
  {
    "src": "homescreen.png'
    "sizes": "192x192",
    "type": "image/png"
  },
  {
    "src": "homescreen-144.
    "sizes": "144x144",
    "type": "image/png"
  }
]
}
```

Màn hình khởi động ứng dụng

- name: Tên của ứng dụng xuất hiện tại màn hình khởi động
- icons: Ảnh đại diện xuất hiện tại màn hình khởi động
- background_color: Màu nền được sử dụng tại màn hình khởi động

Kiểu hiển thị ứng dụng

- "display":
  - fullscreen
  - standalone
  - minimal-ui
  - browser

Trải nghiệm người dùng liên quan cài đặt Home Screen

- Nếu người dùng chỉ cần sử dụng ứng dụng một hoặc ít lần trong thời gian ngắn
  - Cài đặt Home Screen chỉ gây phiền hà => Cần hủy tính năng cài đặt Home Screen
- Người dùng có thể đồng ý cài đặt hoặc không
  - Thống kê để phân tích sở thích của người dùng
- Người dùng bị động với lời nhắc. Nếu người dùng không quan tâm, lời nhắc cài đặt sẽ gây phiền hà cho người dùng
  - Trì hoãn lời nhắc để người dùng chủ động tiến hành cài đặt khi họ thấy cần thiết

Hủy tính năng cài đặt Home Screen

```
window.addEventListener('beforeinstallprompt', function(e) {
  e.preventDefault();
  return false;
});
```

Thống kê sử dụng Home Screen

```
window.addEventListener('beforeinstallprompt', function(event) {
  event.userChoice.then(function(result) {

    console.log(result.outcome);

    if(result.outcome == 'dismissed') {
      // Send to analytics
    }
    else {
      // Send to analytics
    }
  });
});
```

Determine the user's choice - returned as a Promise

Based on the user's choice, decide how to proceed

Trì hoãn/Cất lời nhắc cài đặt

```
  <button id="btnSave" disabled>Click this to show the prompt</button>
  </body>
<script>
      window.addEventListener('load', function() {

        var btnSave = document.getElementById('btnSave');
        var savedPrompt;


  window.addEventListener('beforeinstallprompt', function(e) {

    e.preventDefault();

    btnSave.removeAttribute('disabled');

    savedPrompt = e;

    return false;
  });
```

Add an event listener to the beforeinstallprompt event.

Stash the event in a variable so it can be triggered later.

Khôi phục lời nhắc cài đặt khi người dùng có yêu cầu

```
btnSave.addEventListener('click', function() {
  if (savedPrompt !== undefined) {
    savedPrompt.prompt();

    savedPrompt.userChoice.then(function(result) {

      if (result.outcome == 'dismissed') {
        console.log('User dismissed homescreen install');
      }
      else {
        console.log('User added to homescreen');
      }

      savedPrompt = null;
```

Add an event listener to the click event of the button.

Follow what the user has selected.

You no longer

---------- Hết ----------