```
const CACHE_NAME = 'booktracker-debug-v1.4';
const urlsToCache = [
  '/',
  '/index.html',
  '/manifest.json',
  '/icon-192.png',
  '/icon-512.png',
  'https://cdnjs.cloudflare.com/ajax/libs/jsqr/1.4.0/jsQR.min.js'
];

// Install service worker
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        console.log('Opened cache');
        return cache.addAll(urlsToCache);
      })
      .catch(error => {
        console.error('Cache installation failed:', error);
      })
  );
  // Force the waiting service worker to become the active service worker
  self.skipWaiting();
});

// Fetch event - serve from cache when offline
self.addEventListener('fetch', event => {
  // Skip caching for API calls (they need internet anyway)
  if (event.request.url.includes('googleapis.com') ||
    event.request.url.includes('openlibrary.org')) {
    return;
  }

  event.respondWith(
    caches.match(event.request)
      .then(response => {
        // Return cached version or fetch from network
        if (response) {
          console.log('Serving from cache:', event.request.url);
          return response;
        }

        // Fetch from network and cache for next time
```

```
                return fetch(event.request)
                    .then(response => {
                        // Check if we received a valid response
                        if (!response || response.status !== 200 || response.type !== 'basic') {
                            return response;
                        }

                        // Clone the response since it can only be consumed once
                        const responseToCache = response.clone();
                        caches.open(CACHE_NAME)
                            .then(cache => {
                                cache.put(event.request, responseToCache);
                            });

                        return response;
                    })
                    .catch(error => {
                        console.error('Fetch failed:', error);
                        // Return a custom offline page if you have one
                        // return caches.match('/offline.html');
                    });
            })
    );
});

// Activate service worker
self.addEventListener('activate', event => {
    event.waitUntil(
        caches.keys().then(cacheNames => {
            return Promise.all(
                cacheNames.map(cacheName => {
                    if (cacheName !== CACHE_NAME) {
                        console.log('Deleting old cache:', cacheName);
                        return caches.delete(cacheName);
                    }
                })
            );
        })
    );
    // Take control of all clients immediately
    self.clients.claim();
});
```