Grundlagen der Programmierung von Progressive Web Apps

Service Worker

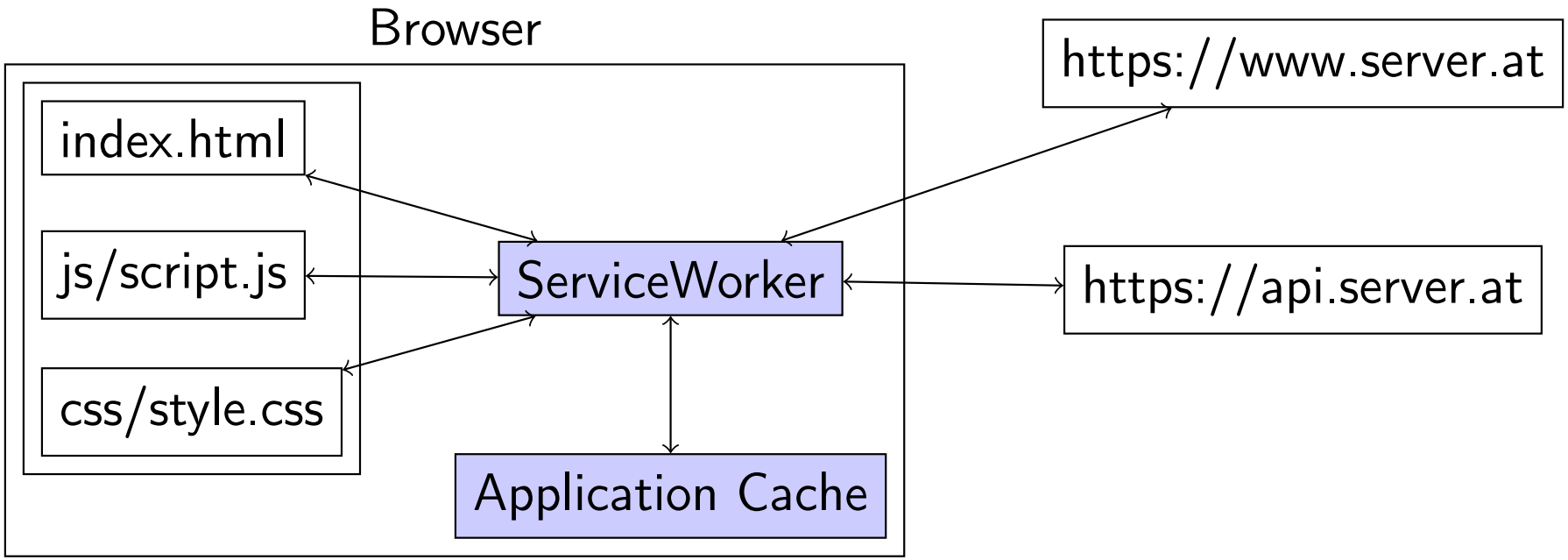
Alexander Miller
alles.mil@gmail.com

Ausbildung Junior Software-Entwickler
WIFI Salzburg

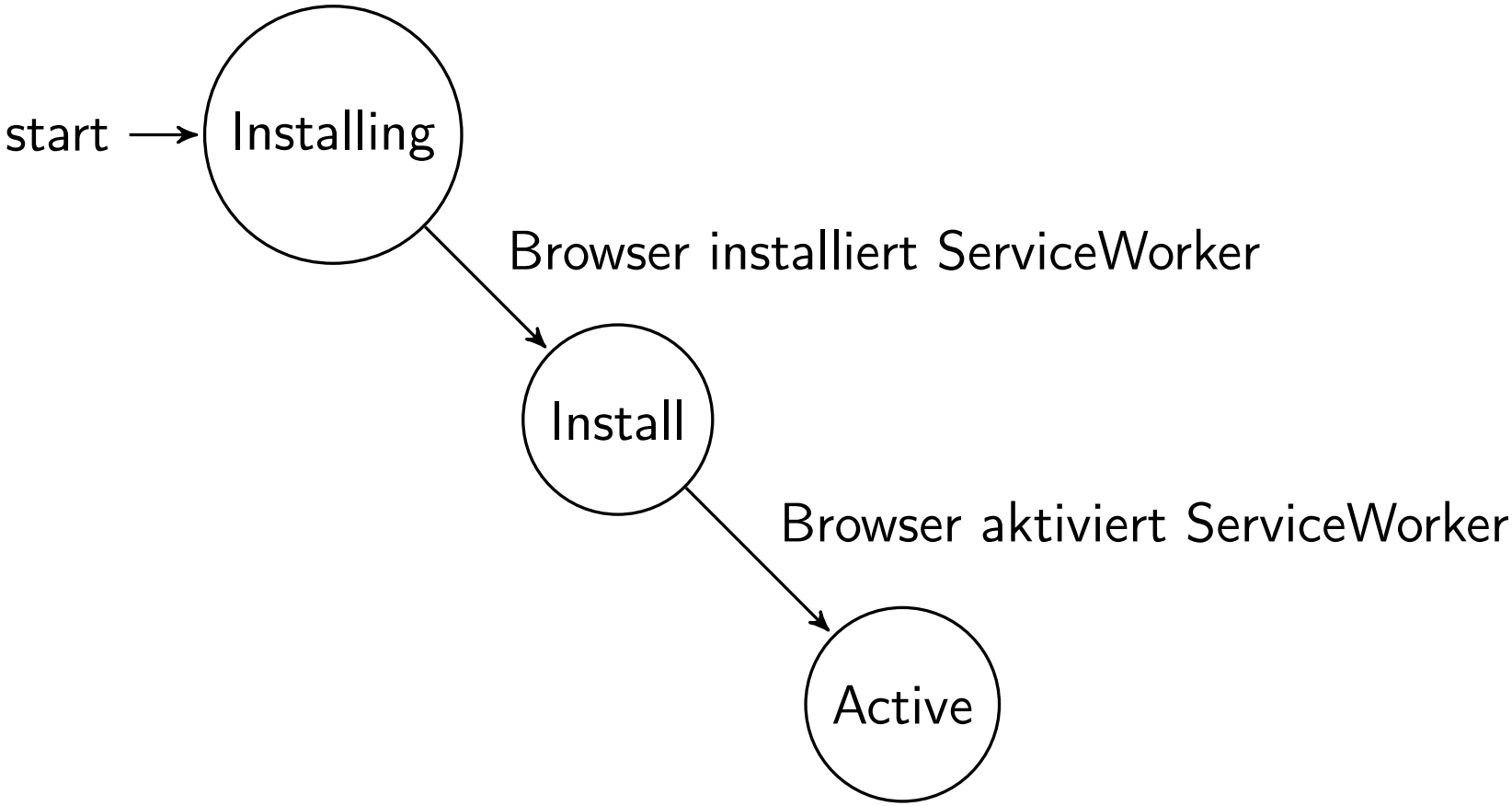
Frühjahr 2021

Version 20. April 2021

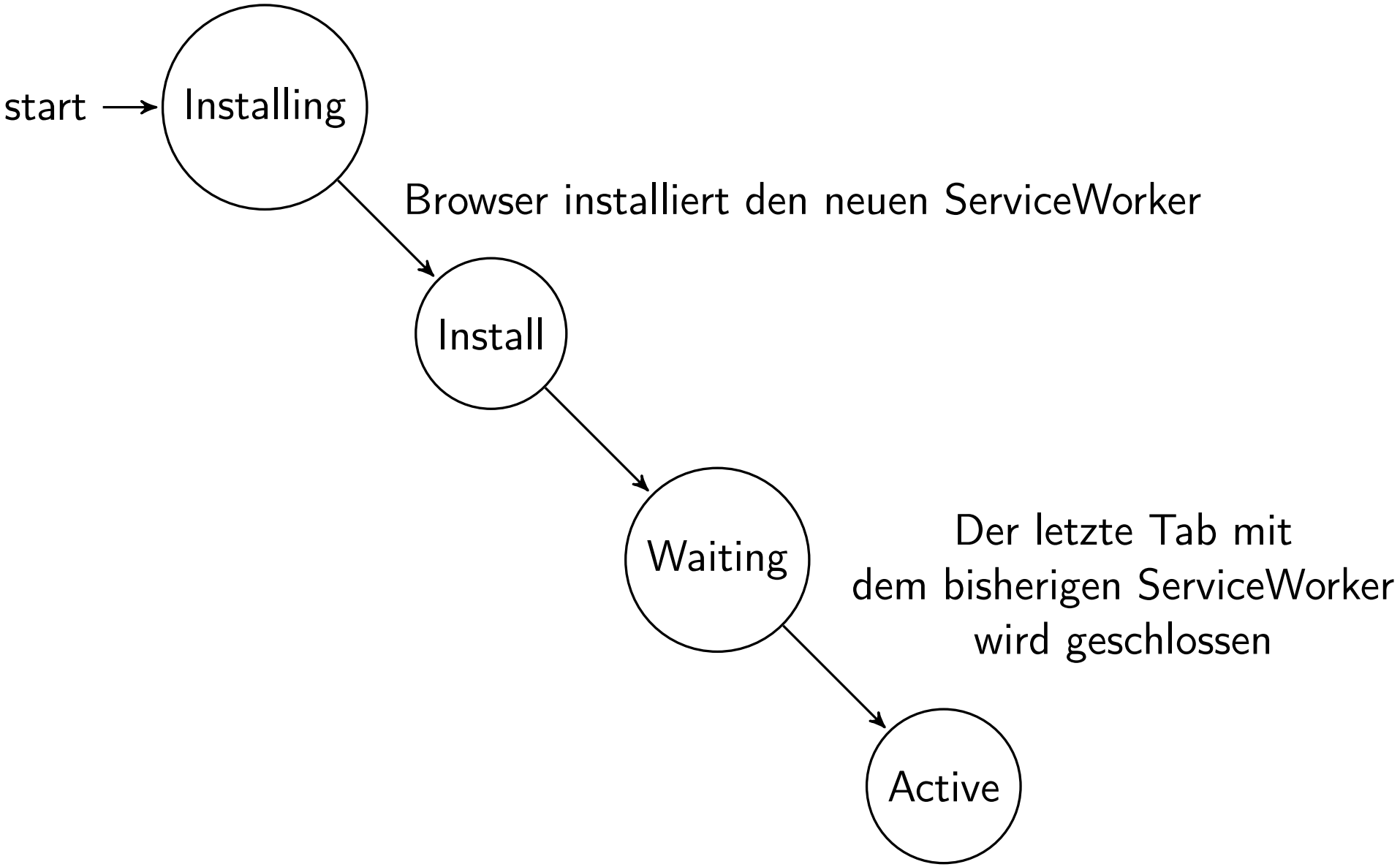
Aufbau


Browser
index.html
js/script.js
css/style.css
ServiceWorker
Application Cache
https://www.server.at
https://api.server.at

1. ServiceWorker registrieren



Nächsten ServiceWorker registrieren



Feature Detection

```
// js/script.js

if (navigator.serviceWorker) {
  // Browser unterstützt ServiceWorker
}
```

Registrierung

```
// js/script.js

navigator.serviceWorker.register('./sw.js')
  .then(reg => {
    console.log('ServiceWorker erfolgreich registriert');
  })
  .catch(error => {
    console.log('Fehler beim Registrieren', error);
  });
```

Events im ServiceWorker

```
// sw.js

self.addEventListener('install', event => { });

self.addEventListener('active', event => { });

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

Dateien im Cache speichern

```
// sw.js

const CACHE_NAME = 'sw-cache-v1';
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(cache => {
        return cache.addAll([
          './',
          './index.html',
          './js/script.js'
        ]);
      })
  );
});
```

Response im Cache speichern

```
// sw.js

self.addEventListener('fetch', event => {
  return fetch(event.request)
    .then(response => {
      // Response-Objekte können nur einmal verwendet werden:
      const responseClone = response.clone();
      caches.open(CACHE_NAME)
        .then(cache => {
          cache.put(event.request, responseClone);
       });
      return response;
    });
});
```

Requests mit dem Cache beantworten

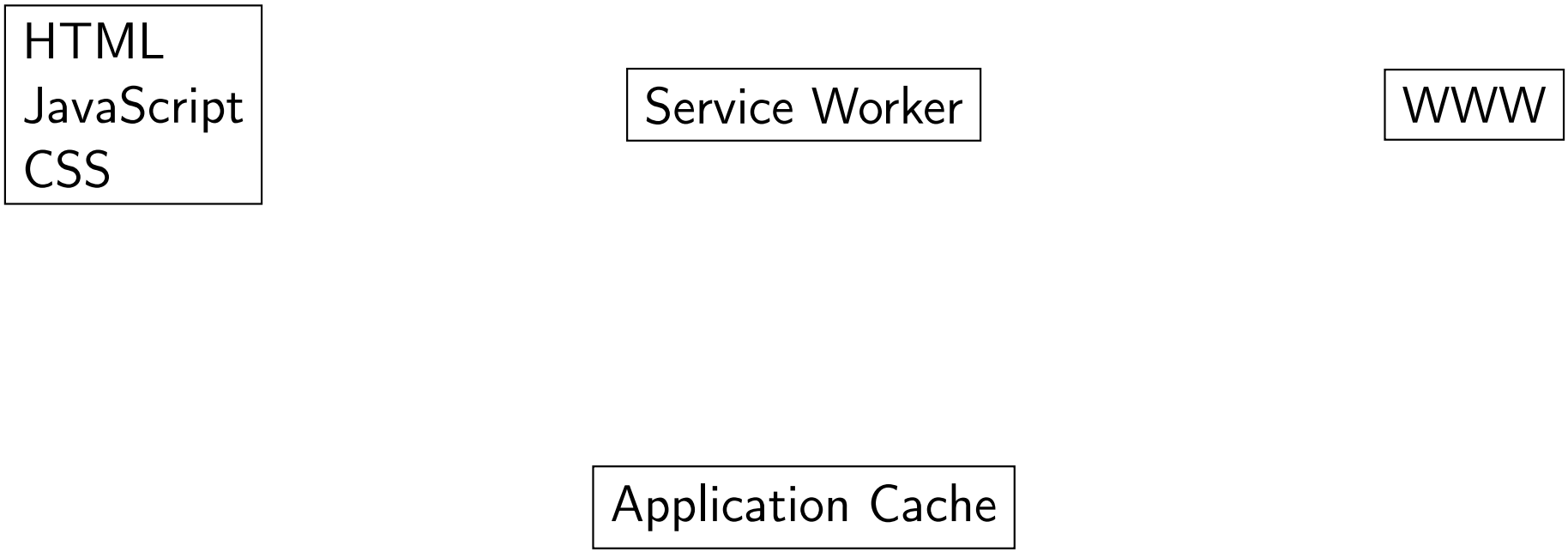
```
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        if (response !== undefined) {
          return response;
        }
        else {
          // Antwort ist nicht im Cache,
          // der ServiceWorker löst den tatsächlichen
          // fetch zum Server aus.
          return fetch(event.request);
        }
      })
  );
});
```

Alte Caches löschen

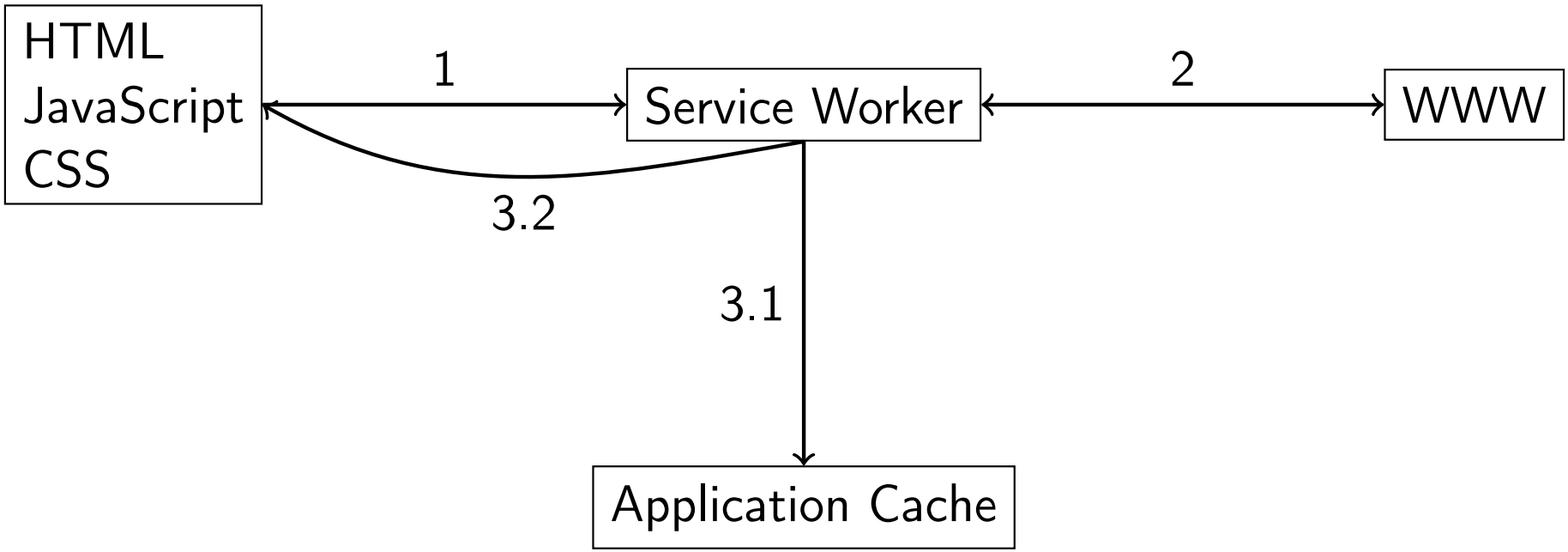
```
// sw.js

self.addEventListener('active', event => {
  caches.keys().then(cacheNames => {
    for (let cacheName of cacheNames) {
      if (cacheName !== CACHE_NAME) {
        caches.delete(cacheName);
      }
    }
  });
});
```

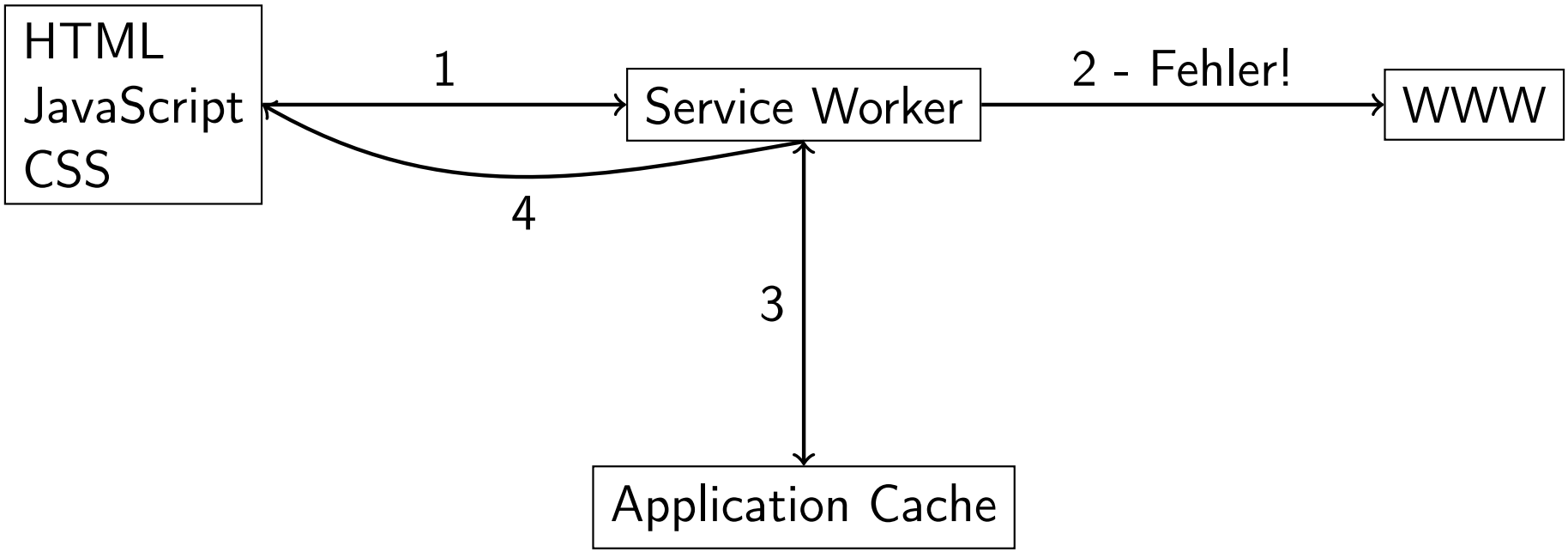
Service Worker als Request Proxy



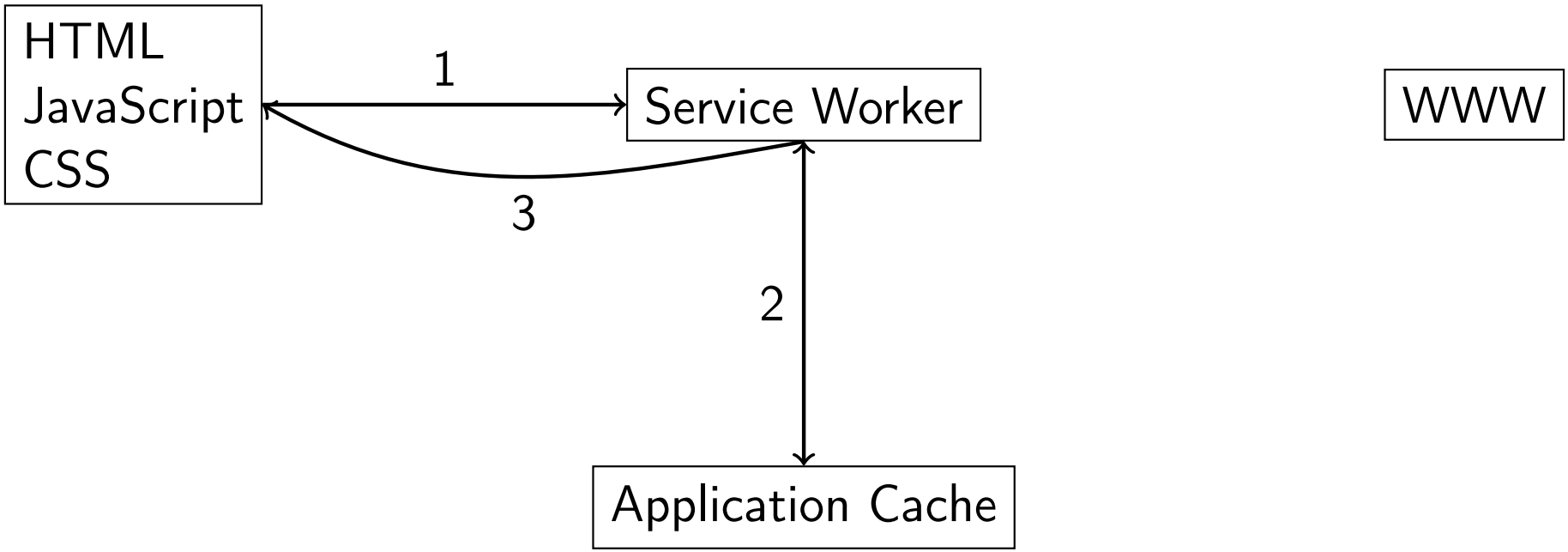
Network First



Network First (Offline)



Cache First


HTML
JavaScript
CSS
1
Service Worker
WWW
3
2
Application Cache

Cache First (Cache Miss)