SMART IMAGES

with Service Workers

Hey!

Vlad Stawizki
@vlad_stawizki

Florian Unglaub
@florian_unglaub

Paul Straetmans
@paul_straetmans

Bastelvorlage

github.com/vdstawizki/ing-webweek-service-worker


WHAT IS A
PWA?

Progressive Web App

- Offline
- Sync in Background
- Installable
- Geolocation
- Web Bluetooth
- Stream API
- ...

PWA Requirements

- Service Worker
- Manifest
- https://

SW Support

Service Workers - WD

Method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline.

Usage: Global

% of all users: 91.48% + 0.35% = 91.83%

Current aligned | Usage relative | Date relative | Apply filters | Show all | ?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsung Internet
		2-32													
		1 33-43													
		44													
		3 45													
		46-51													
		3 52													
	12-14	53-59	4-39		10-26										
	2 15-16	3 60	40-44	3.1-11	27-31	3.2-11.2									
6-10	17	61-67	45-74	11.1-12	32-60	11.3-12.1		2.1-4.4.4	7	12-12.1			10		4-8
11	18	68	75	12.1	62	12.3	all	67	10	46	75	67	11	11.8	9.2
	76	69-70	76-78	13-TP		13									

Feature detection

```
const serviceWorkerSupported = 'serviceWorker' in navigator;
const cacheSupported = 'caches' in window;
const geoSupported = 'geolocation' in navigator;
```

SMART IMAGE

LOADING

Default images

- Render Blocking
- Request Blocking
- Huge traffic

```
<img
    src="https://source.unsplash.com/random/300x300"
    alt="Image desc"
/>
```

Lazy loading

- Don’t block rendering
- Skeleton loading
- Only define Aspect Ratios

Lazy loading

- Example: ing.de

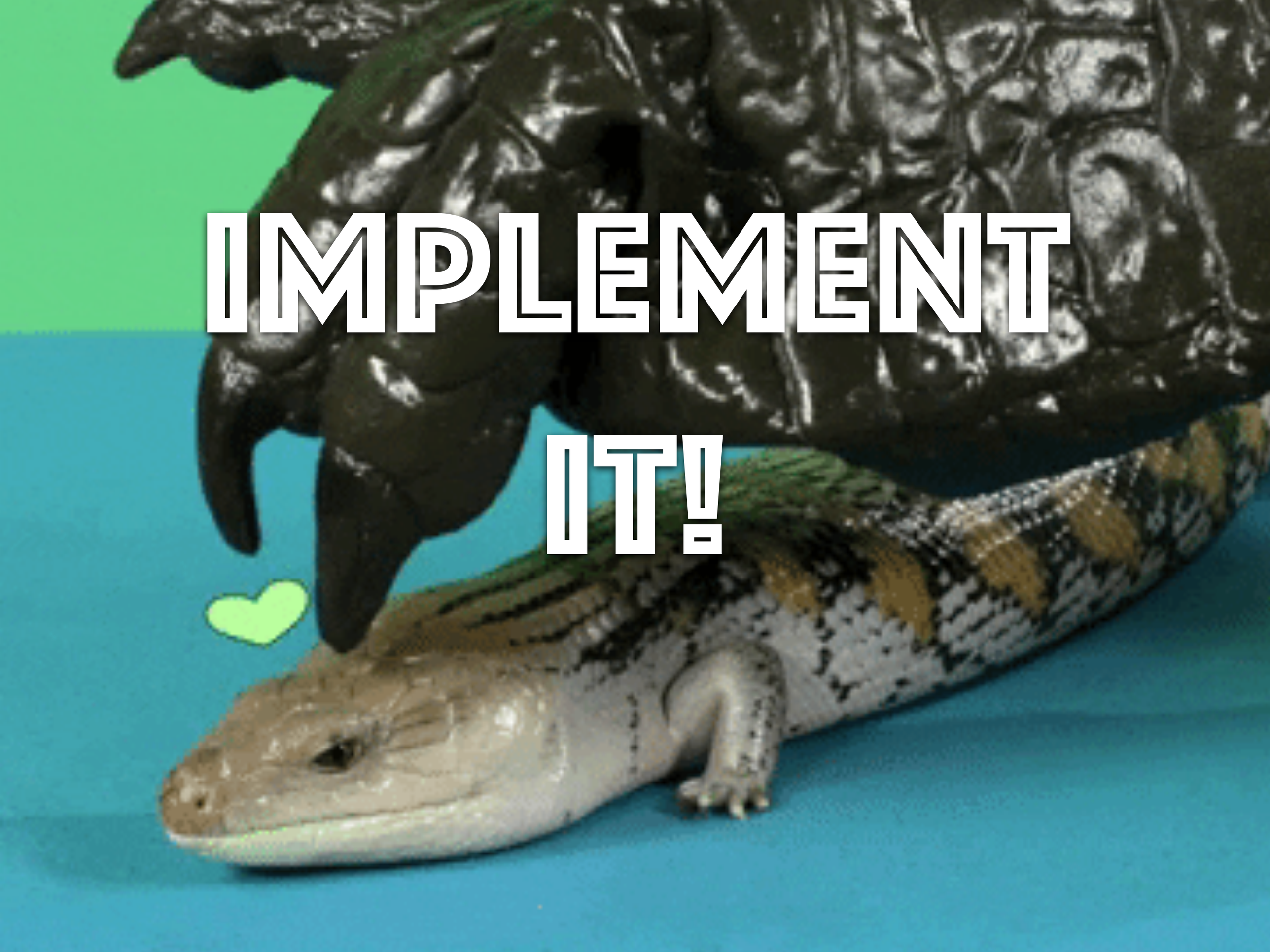
```
<img
    src=""
    data-src="https://source.unsplash.com/random/300x300"
    alt="Image desc" />

<script>
    const images = document.querySelectorAll('img[data-src]');

    images.forEach(image => {
        image.setAttribute('src', image.getAttribute('data-src'));
    });
</script>
```

Lazy loading with SW

- Only load needed resolution
- Smart network detection
- Smart cache client side
- Offline mode


IMPLEMENT
IT!

Install SW

```
// client.js
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('service-worker.js');
}

// service-worker.js
self.addEventListener('install', event => {
    console.log('The service worker is being installed.');
});
```

LIVE DEMO

Caching

```
// service-worker.js
const PRECACHE = 'precache-v1';

// A list of local resources we always want to be cached.
const PRECACHE_URLS = [
    './',
    'main.js',
    'manifest.json',
    '...'
];

self.addEventListener('install', event => {
    console.log('The service worker is being installed.');
    event.waitUntil(
        caches
            .open(PRECACHE)
            .then(cache => cache.addAll(PRECACHE_URLS))
            .then(self.skipWaiting())
    );
});
```

Caching

```
// service-worker.js
self.addEventListener('fetch', event => {
    if (event.request.url.startsWith(self.location.origin)) {
        event.respondWith(
            caches.match(event.request).then(cachedResponse => {
                if (cachedResponse) {
                    return cachedResponse;
                }
            })
        );
    }
});
```

Offline

```
// service-worker.js
self.addEventListener('fetch', event => {
    /*
        previous caching implementation
    */

    if (event.request.destination === 'image') {
        if (!navigator.onLine) {
            event.respondWith(caches.match('assets/kitty.jpeg'));
        }
    }
});
```

LIVE DEMO

Network Information

```
// service-worker.js
const imageQuality = navigator => {
    if (navigator.connection) {
        const networkType = navigator.connection.effectiveType;

        switch(networkType) {
            case '3g':
                return 60;
            case '2g':
                return 40;
            case 'slow-2g':
                return 20;
        }
    }
    return 80;
};
```

Network Information

```
// service-worker.js
self.addEventListener('fetch', event => {
    /*
      previous implementation...
    */

    if (event.request.destination === 'image') {
        if (!navigator.onLine) {
            event.respondWith(caches.match('assets/kitty.jpeg'));
        } else {
            const quality = imageQuality(navigator);
            const newRequestUrl = event.request.url
                .replace(/q=\d+/, `q=${quality}`);
            console.log(`fetching ${newRequestUrl}`);

            event.respondWith(fetch(newRequestUrl));
        }
    }
});
```

Breakpoint Check

```
// client.js
let initialMq = 0;

const cbTablet = e => {
    if (e.matches) {
        navigator.serviceWorker.ready.then(registration => {
            if (registration.active) {
                registration.active.postMessage(560);
            }
        });
    }
};

const mqTablet = window.matchMedia('(min-width: 780px) and (max-width: 1199px) ');

if (mqTablet.matches) {
    initialMq = 560;
}

mqTablet.addListener(cbTablet);
```

Breakpoint Check

```
// client.js
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('service-worker.js');

    navigator.serviceWorker.ready.then(registration => {
        if (registration.active) {
            registration.active.postMessage(initialMq);
        }
    });

    navigator.serviceWorker.addEventListener('message', event => {
        console.log('Client Received Message: ' + event.data);
        loadImages();
    });
};
```

Breakpoint Check

```
// service-worker.js
self.addEventListener('message', evt => {
    const client = evt.source;
    client.postMessage(evt.data);
    clientImageWidth = evt.data;
});

self.addEventListener('fetch', event => {
  /* previous implementation ... */

    if (event.request.destination === 'image') {
        const quality = imageQuality(navigator);
        const newRequestUrl = event.request.url
            .replace(/w=\d+/, `w=${clientImageWidth}`)
            .replace(/h=\d+/, `h=${clientImageWidth}`)
            .replace(/q=\d+/, `q=${quality}`);

        event.respondWith(fetch(newRequestUrl));
    }
});
```

LIVE DEMO


MOAR!

Few ideas

- Energy Mode (battery)
- Push Notifications
- Geolocation (Location based content / currency)
- Background Color (lazy load)
- saveData (API/NetworkInformation)

QUESTIONS?

github.com/vdstawizki/ing-webweek-service-worker