Tutorial 18: Going for Speed

Up next!

09:00	**Introduction**
09:15	Part 1: Efficient Frontend Design
09:40	Q&A
09:50	Coffee Break
10:00	**Part 2: High-Performance Networking**
10:40	Q&A
10:50	Coffee Break
11:00	**Part 3: Scalable Backend Architectures**
11:40	Q&A
11:50	Coffee Break
12:00	**Part 4: Performance Tracking & Analysis**
12:00	The Core Web Vitals (Google Guest Speaker!)
12:30	Measuring Web Performance
12:50	Q&A

Baqend

Going for Speed

Full-Stack Performance Engineering in Modern Web-Based Applications

The Web Conference
April 15, 2021
Tutorial

Baqend

Who We Are

Wolfram Wingerath
Data Engineering (Baqend)

Felix Gessert
CEO (Baqend)

Benjamin Wollmer
Data Engineering (Baqend, UHH)

Ralf Ohlenbostel
Conversion Specialist E-Commerce – Central Europe (Google, Guest Speaker)

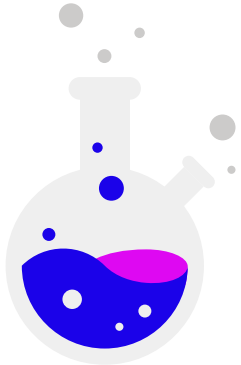
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Baqend

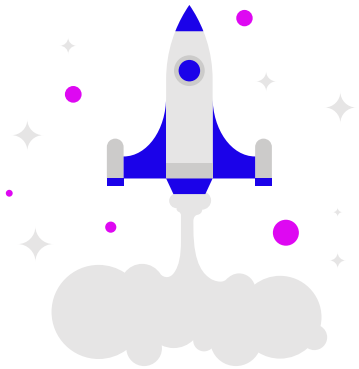
We Are Baqend

We bring performance research to practice.

Years of **web performance research**

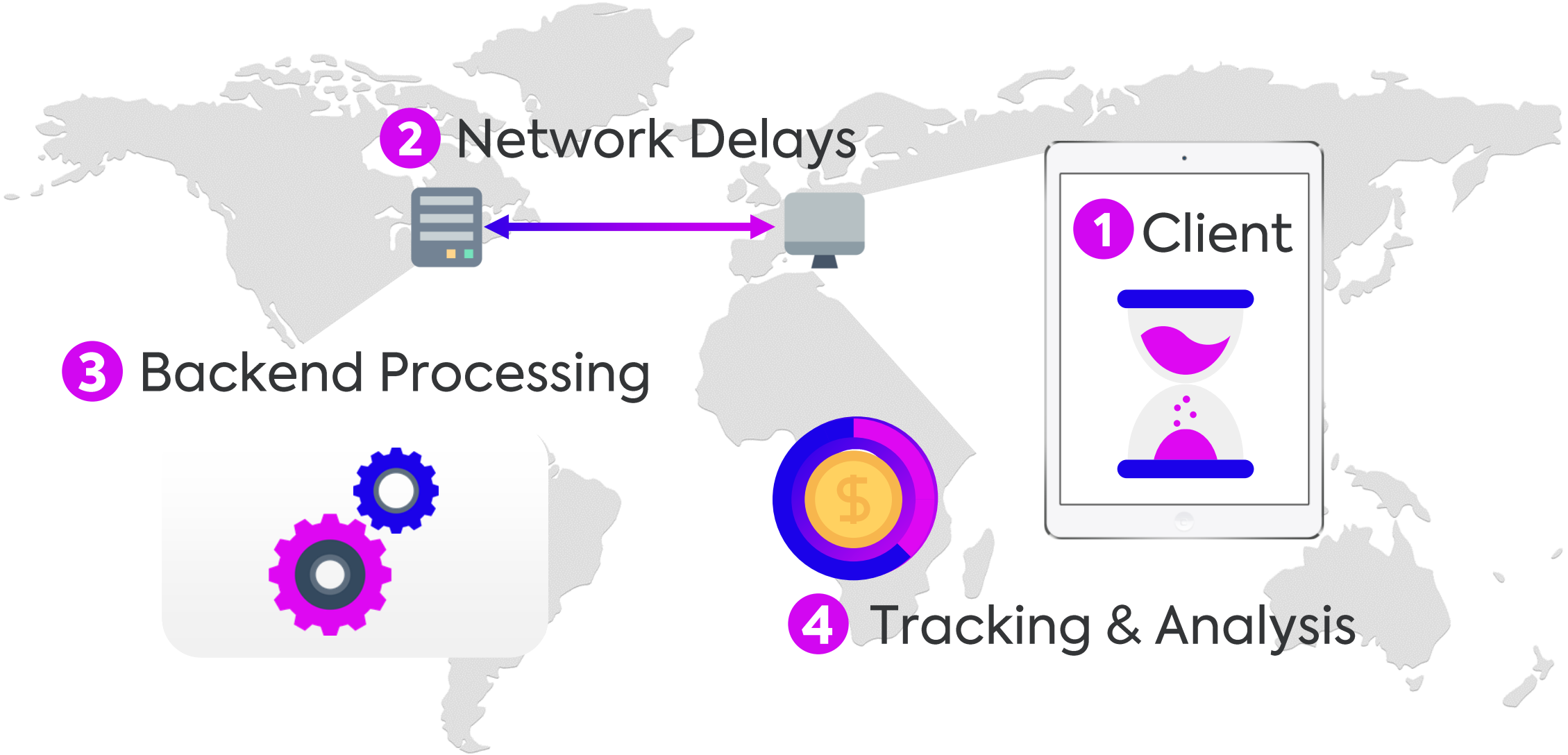


Novel approach for caching **dynamic data**

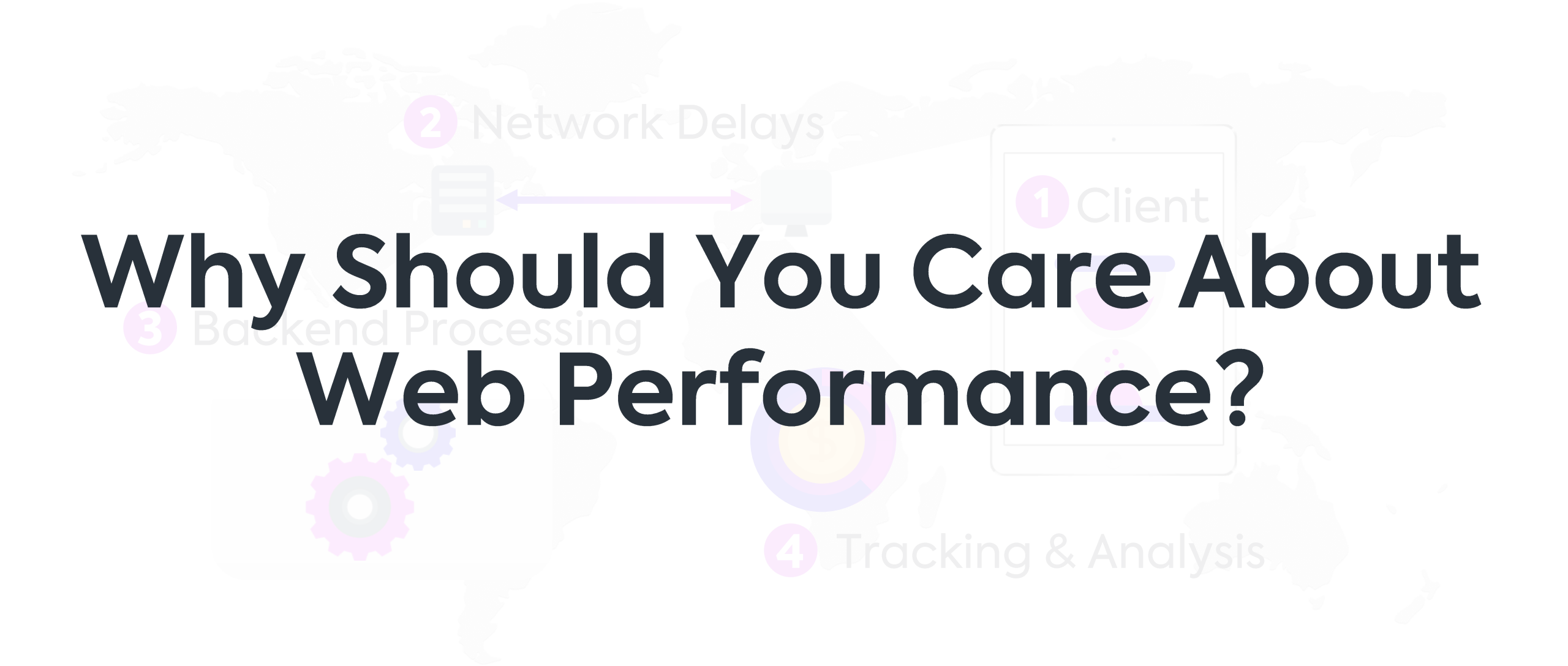


Speed Kit – SaaS for e-commerce speed

The 4 Challenges of Web Performance



The 4 Challenges of Web Performance



Why Should You Care About Web Performance?

Tutorial 18: Going for Speed

Up next!

09:00	**Introduction**
09:15	Part 1: Efficient Frontend Design
09:40	Q&A
09:50	Coffee Break
10:00	**Part 2: High-Performance Networking**
10:40	Q&A
10:50	Coffee Break
11:00	**Part 3: Scalable Backend Architectures**
11:40	Q&A
11:50	Coffee Break
12:00	**Part 4: Performance Tracking & Analysis**
12:00	The Core Web Vitals (Google Guest Speaker!)
12:30	Measuring Web Performance
12:50	Q&A

Baqend

Going for Speed

Motivation:
Who Cares About Web Performance?

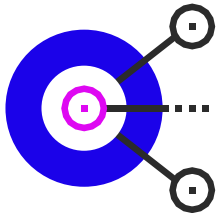
The Web Conference
April 15, 2021
Tutorial

Wolfram Wingerath, Benjamin Wollmer, Felix Gessert, Stephan Succo, Norbert Ritter

Baqend

How Do Users Perceive Web Performance?

User Profile



Gender

Young women are less likely to buy on slow pages

Region

Speed influences New Yorkers more than Californians

Nationality

Japanese have the highest expectations

Age

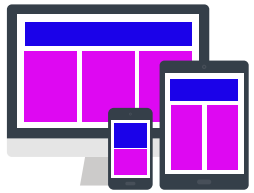
18-to-24-year olds tend to be more demanding

Think Fast, The 2019 Page Speed Report Stats & Trends for Marketers, Unbounce, 2019.

Brain Food, Speed Matters, Designing for Mobile Performance, Google, 2017.

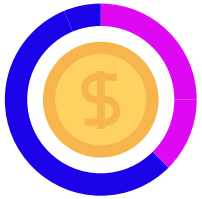
Performance Matters, 9 Key Consumer Insights, Akamai, 2014

Device Distribution



iOS vs. Android

iOS users are more demanding



Purchases

Mobile users buy more frequently than desktop users

Market Share

iOS dominant in US/UK, Android in other markets

Comparing

Mobile is also preferred for comparing products

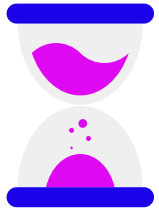
Think Fast, The 2019 Page Speed Report Stats & Trends for Marketers, Unbounce, 2019.

Brain Food, Speed Matters, Designing for Mobile Performance, Google, 2017.

Performance Matters, 9 Key Consumer Insights, Akamai, 2014

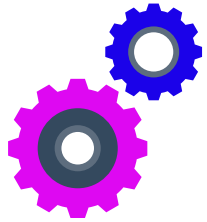
Android v iOS market share 2019 Q2, DeviceAtlas, 2019

Context Matters



Situation

Pages feel slower
when on the move

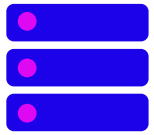


State of Mind

Pages feel faster
when relaxed

Sale

Pages feel slow
during sales



Availability

Slow eventually
becomes unavailable

Brain Food, Speed Matters, Designing for Mobile Performance, Google, 2017.

Performance Matters, 9 Key Consumer Insights, Akamai, 2014

Delay Psychology : Rules of Thumb

Delay	User Perception
0 – 100 ms	Instant
100 – 300 ms	Small perceptible delay
300 – 1000 ms	Machine is working
1+ s	Mental context switch
10+ s	Task is abandoned

Stay under 1000 ms to keep users' attention

I. Grigorik, High performance browser networking, O'Reilly Media, 2013

Jakob Nielsen, Usability Engineering, Morgan Kaufmann, 1994

How Do Businesses Measure Web Performance?

You Heard The Stories

amazon — **100 ms** slower → **-1%** Conversion Rate

zalando — **100 ms** faster → **+0.7%** Revenue Per Session

Walmart — **100 ms** faster → **+1%** Revenue

Greg Linden. Make Data Useful. Stanford Data Mining Class CS345A, 2006

Shuhei Kagawa, Jeff Cybulski, David Martin Jones, et. al.. Loading Time Matters. Zalando Tech Blog, 2018

C. Crocker, A. Kulick, B. Ram. Real-User Monitoring at Walmart, SF & SV Web Performance Group, 2012.

Load Time & SEO

GQ — From **7 s** to **2 s** Loads → **+80%** Traffic

Google — **500 ms** Slower Loads → **-20%** Traffic

Pinterest — **40%** Faster Loads → **+15%** SEO Traffic

Lucia Moses. How GQ Cut Its Webpage Load Time By 80 Percent, Digiday, 2015.

Marissa Mayer, Conference Keynote, Web 2.0, 2006

Sam Meder, Vadim Antonov, Jeff Chang. Driving User Growth With Performance Improvements, Pinterest Blog, 2017

Load Time & User Engagement

FORRESTER®

-80% load time → **+60%** Session Length (Mobile)

OTTO

-42% time to FCP → **+25%** Session Length

Akamai

+2s load time → **+103%** Bounce Rate

Forrester. The Total Economic Impact ™ Of Accelerated Mobile Pages, 2017

Lars Bognar. Mobile Speed Race der Otto Group Verbessert Mobile Ladezeiten, TWG, 2019

Akamai. Akamai Online Retail Performance Report: Milliseconds Are Critical, Akamai Blog, 2017

Load Time & User Satisfaction

radware

+500 ms network delay → **+26%** peak frustration

Aberdeen Group

+1 s delay in response times → **-16%** customer satisfaction

Imperial College London

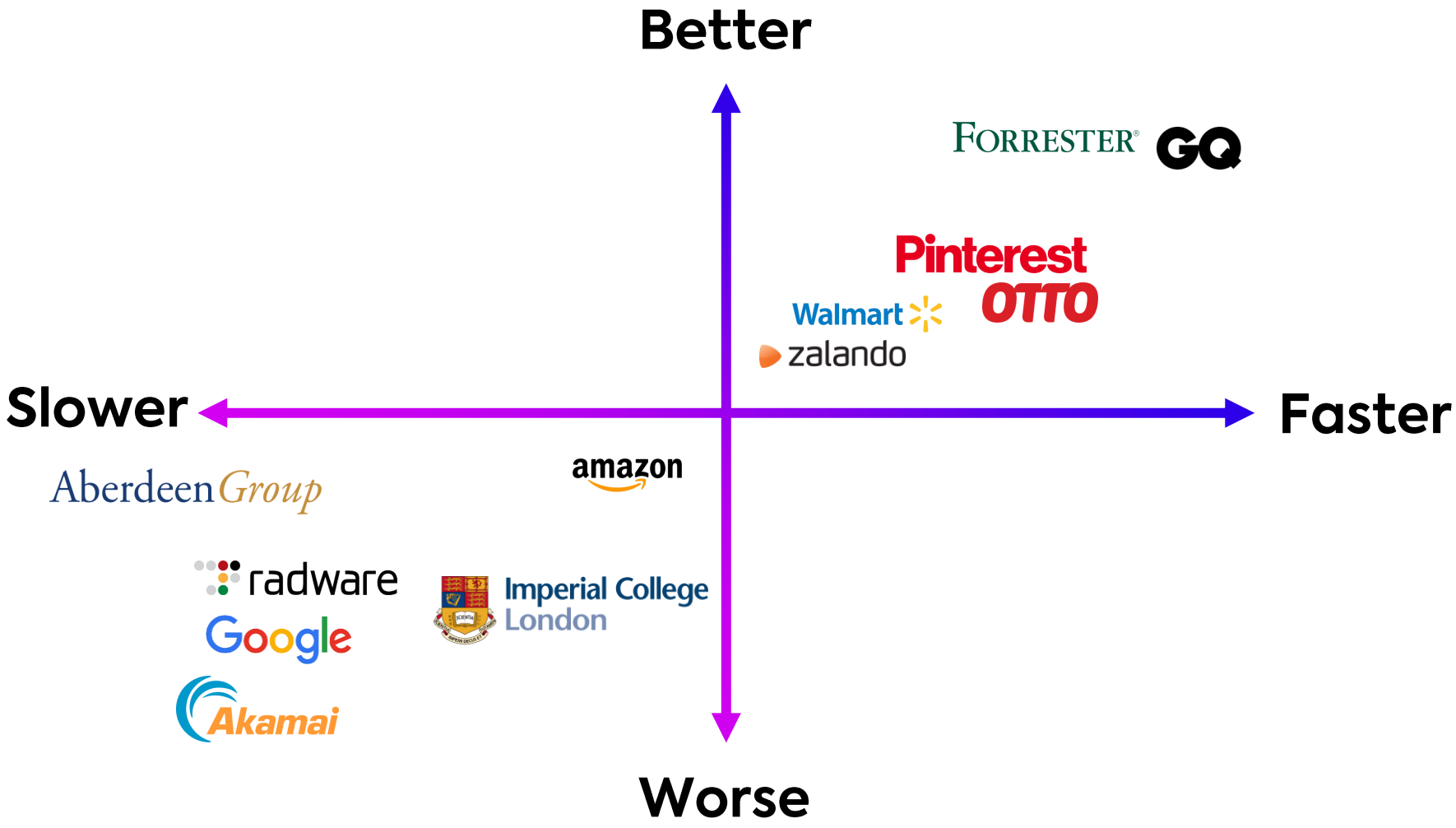
+50% response time → **-50%** productivity

Tammy Everts. Mobile Web Stress: The Impact of Network Speed on Emotional Engagement and Brand Perception, Radware, 2013

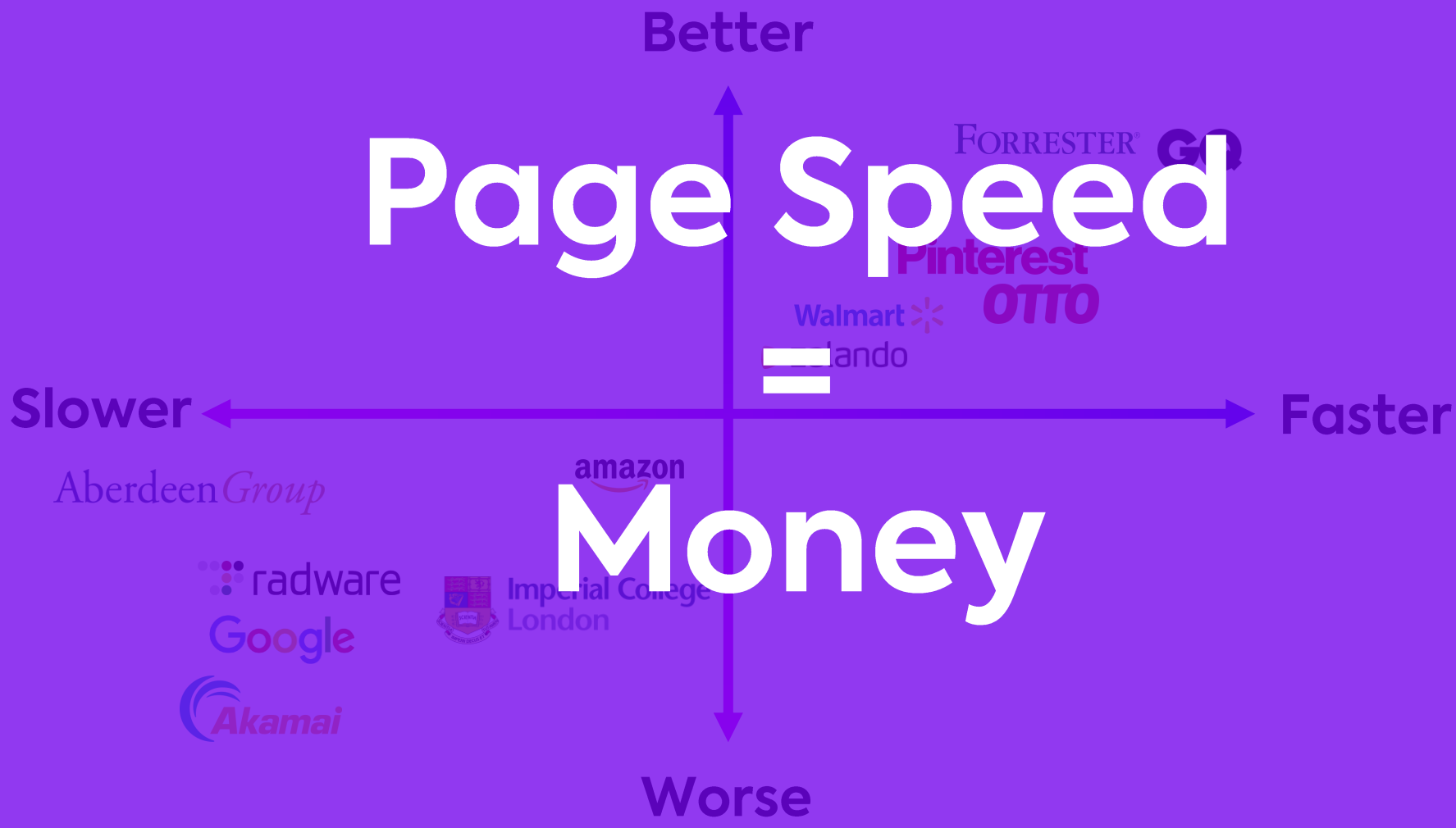
The Performance of Web Applications: Customers Are Won or Lost in One Second, Aberdeen Group, 2008.

T. Goodman, R. Spence. The Effect of System Response Time on Interactive Computer Aided Problem Solving, ICL, 1978

Summary: The Business Impact of Site Speed



Summary: The Business Impact of Site Speed



Going for Speed

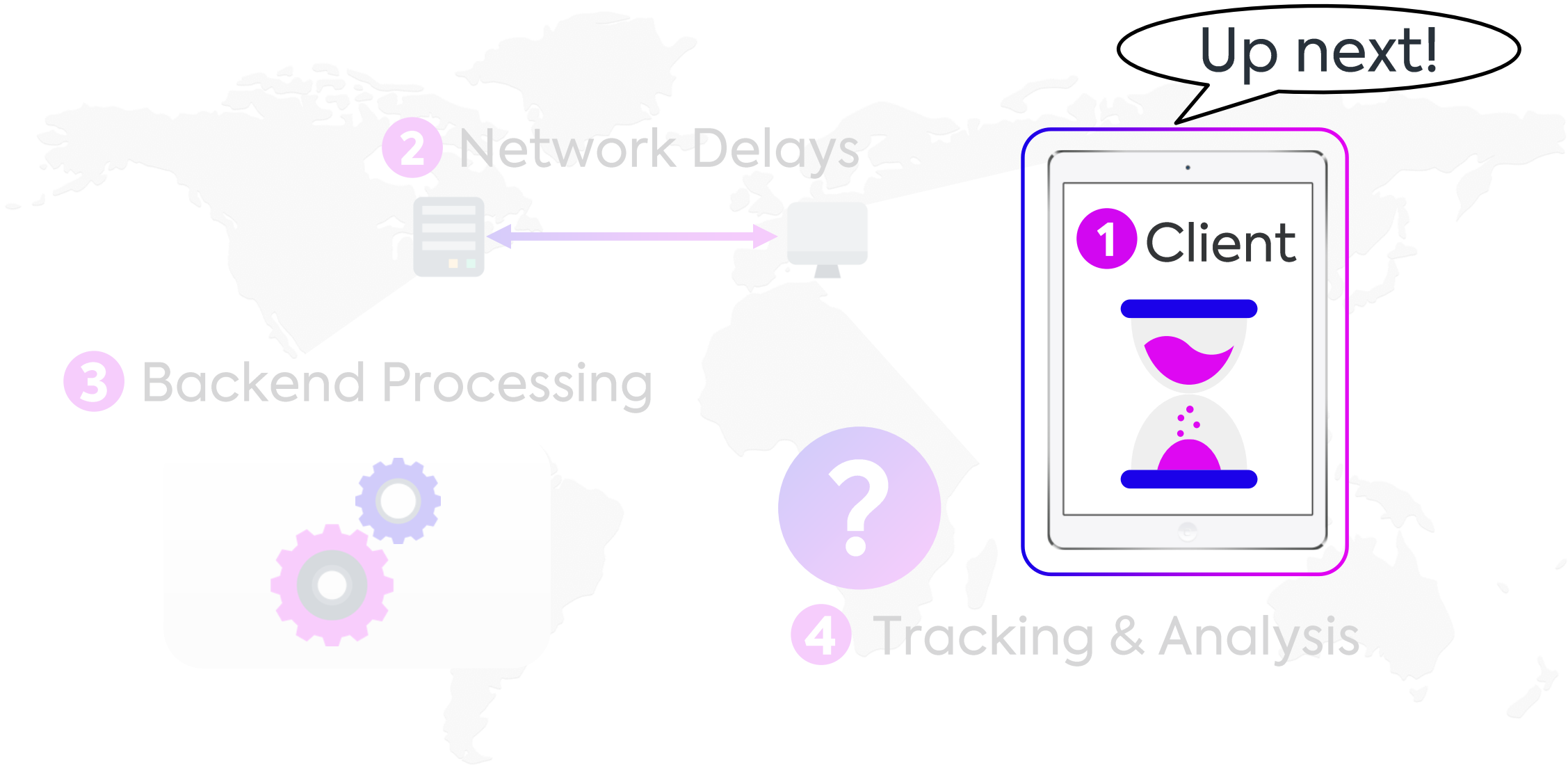
Part 1:
Efficient Frontend Design

The Web Conference
April 15, 2021
Tutorial

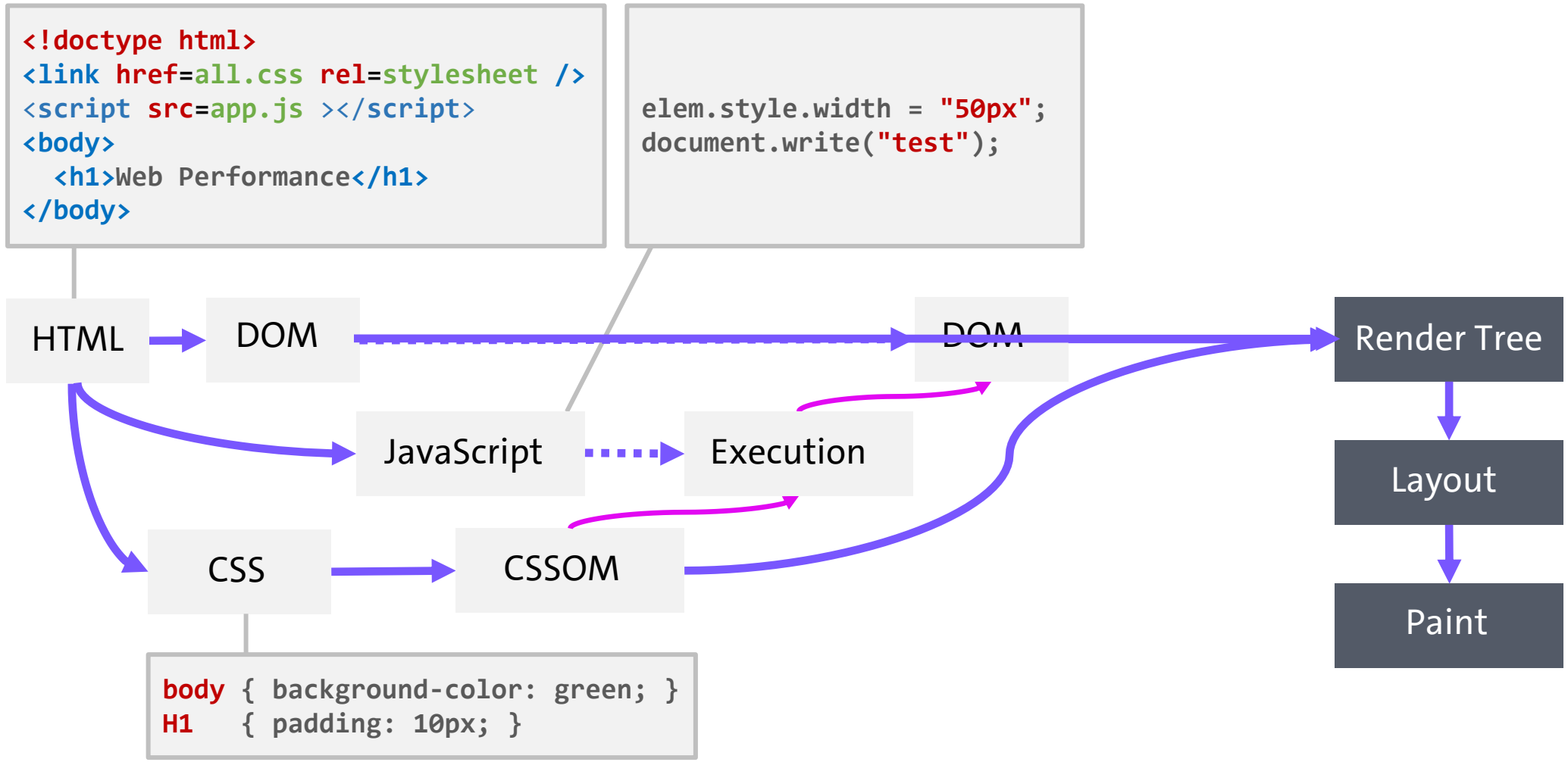
Wolfram Wingerath, Benjamin Wollmer, Felix Gessert, Stephan Succo, Norbert Ritter

Baqend

The 4 Challenges of Web Performance



Critical Rendering Path (CRP)



Critical Rendering Path: Best Practices

1. Minimize **Length** (*Round-Trips*)
2. Minimize **Size** (*Critical Resources*)
3. Minimize **Weight** (*Critical Bytes*)

Inlining cricital CSS and JS “**above the fold”**

Minify and **concatenate** CSS and JS

Single-page application

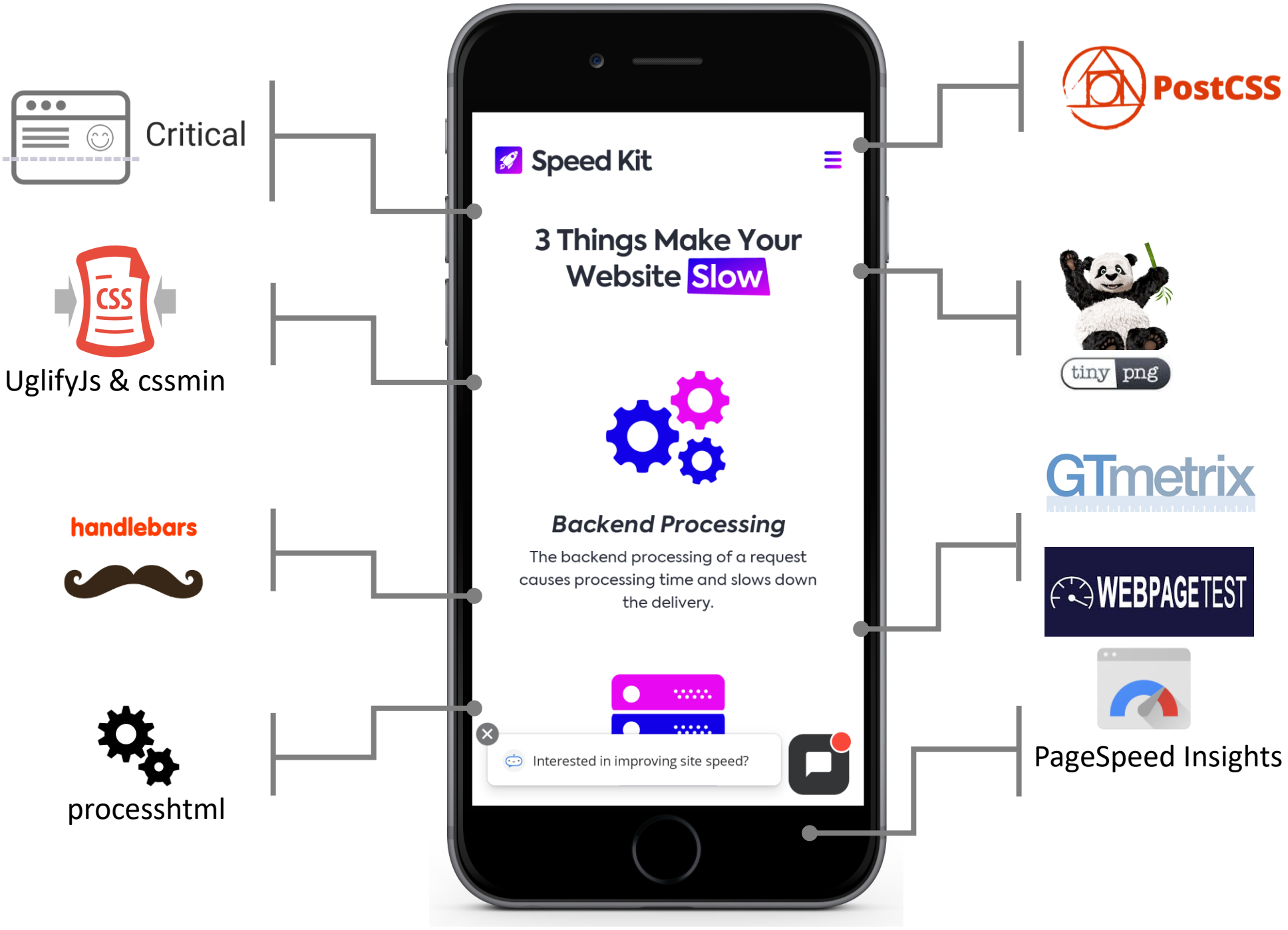
Progressive rendering

Load **CSS** first**, JS** last

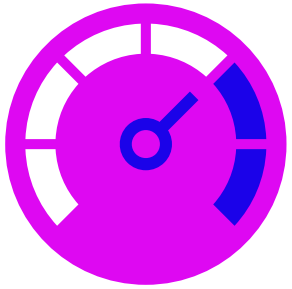
Compress images

Test your performance

Load non-critical CSS and JS **asynchronously**

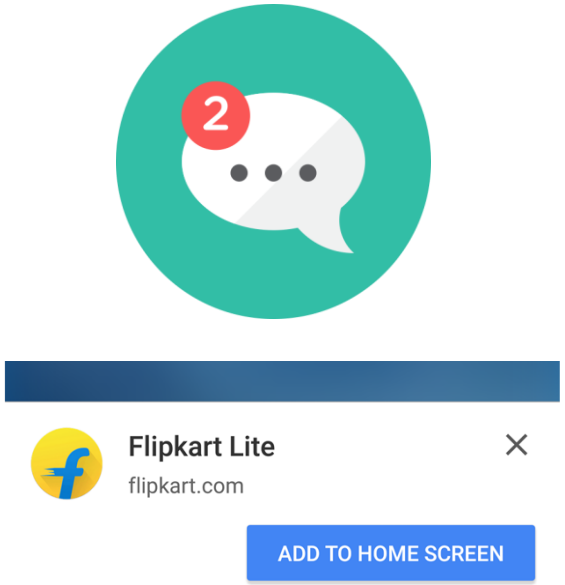

Critical
UglifyJs & cssmin
handlebars
processhtml
Speed Kit
3 Things Make Your Website Slow
Backend Processing
The backend processing of a request causes processing time and slows down the delivery.
Interested in improving site speed?
PostCSS
tiny png
GTmetrix
WEBPAGETEST
PageSpeed Insights

Progressive Web Apps (PWA)



+

+



Fast **Loads** through Caching

Offline Mode (Synchronization)

Add-to-**Homescreen** and **Push Notifations**

Advantages of PWAs

Discoverable
E.g. in search engines

Installable
Easy access from home screen

Linkable
Link into apps through URLs

Network independant
Offline mode

Progressive
Enhance on capable browsers

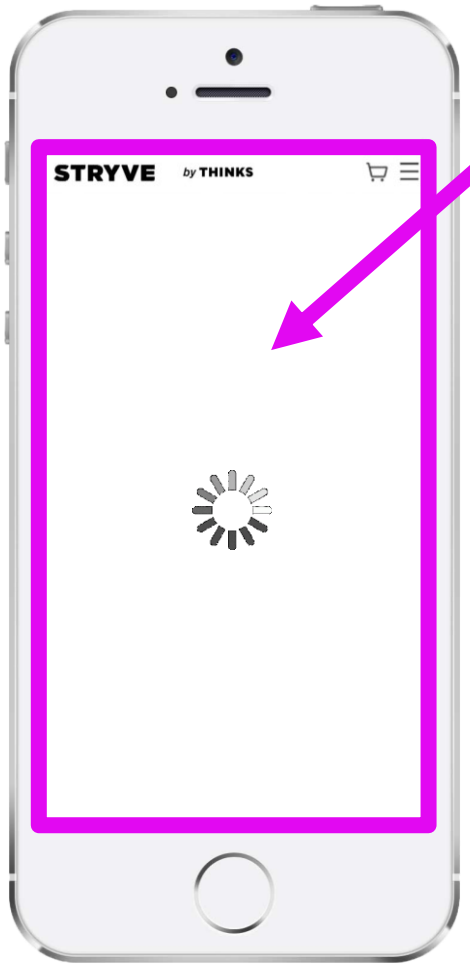
Re-engageable
Engage through Web Push

Responsive
Fit any form factor

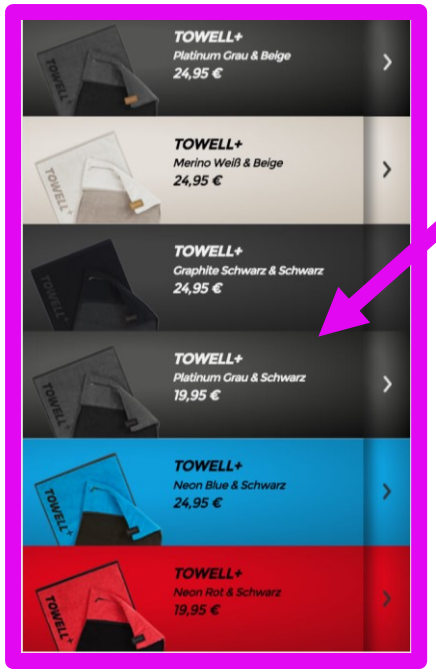
Safe
HTTPS & recognizable URLs

https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive/Advantages

Typical Architecture: App Shell Model



App Shell: **HTML**, **JS**, **CSS**, **images** with app logic & layout

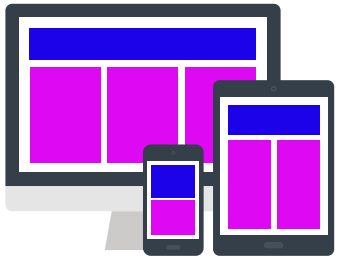


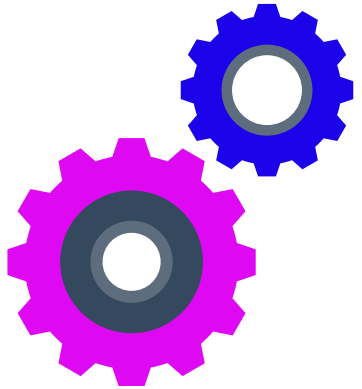
Content: Fetched on demand & may change more often

PWA Building Blocks

PWAs are **best practices** and **open web standards**

Progessively enhance when supported





1. Manifest

2. Service Worker

PWA Implementation

PWAs are **best practices** and **open web standards**

Progessively enhance when supported

1. Manifest declares Add-to-Homescreen:

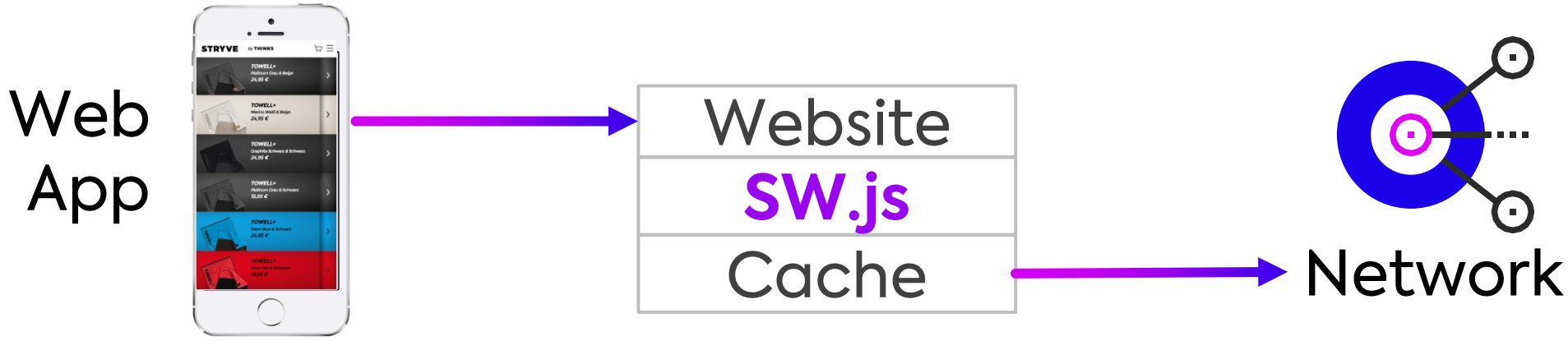
```
<link rel="manifest" href="/manifest.json">
{
  "short_name": "WWW PWA",
  "icons": [
    {"src": "icon-1x.png", "type": "image/png", "sizes": "48x48"}],
  "start_url": "index.html?launcher=true"
}
```

PWA Implementation

PWAs are **best practices** and **open web standards**

Progessively enhance when supported

2. Service Workers for caching & offline mode:

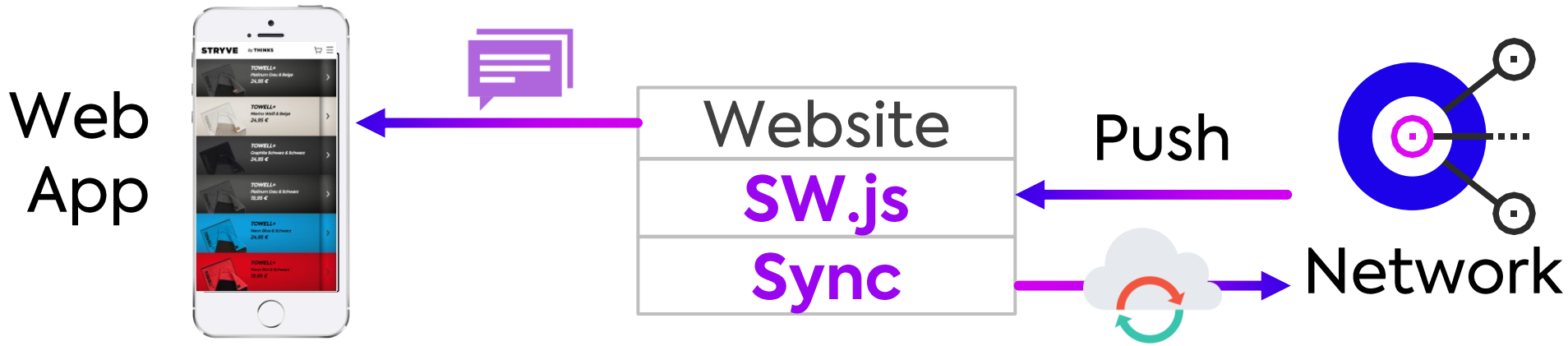


PWA Implementation

PWAs are **best practices** and **open web standards**

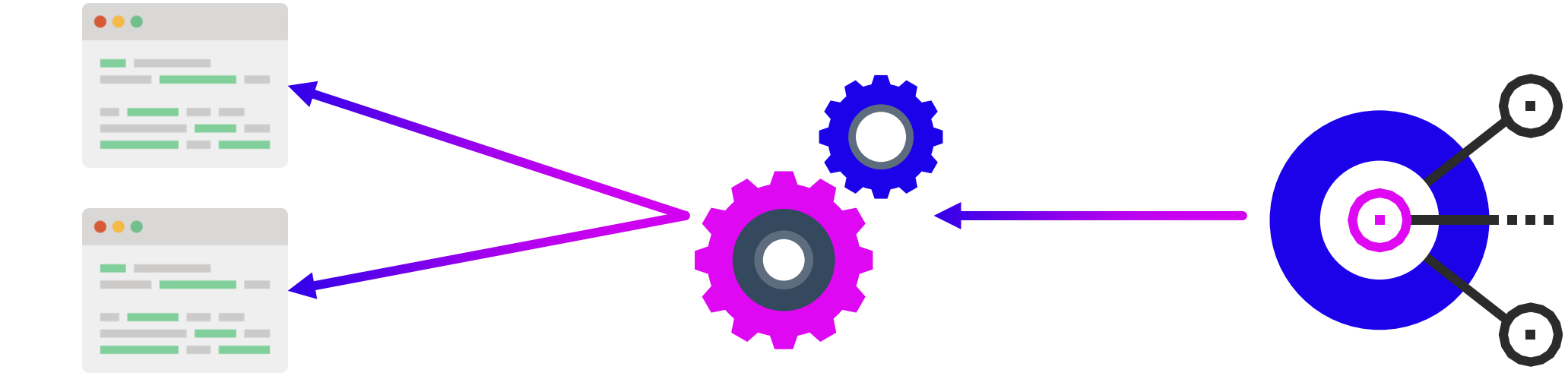
Progessively enhance when supported

3. Add Web Push and Background Sync:



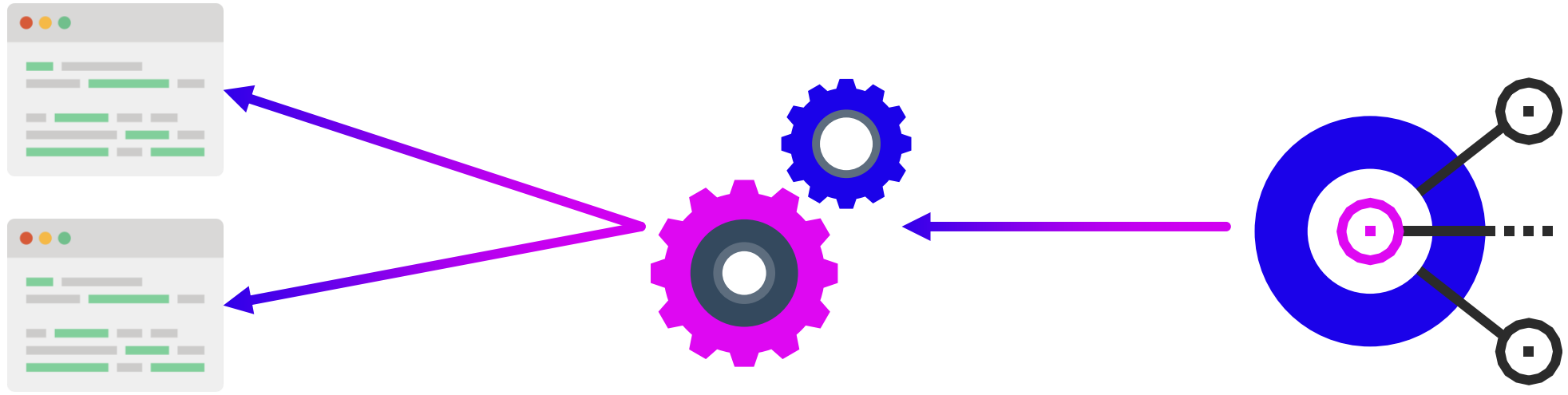
What are Service Workers?

What Are Service Workers



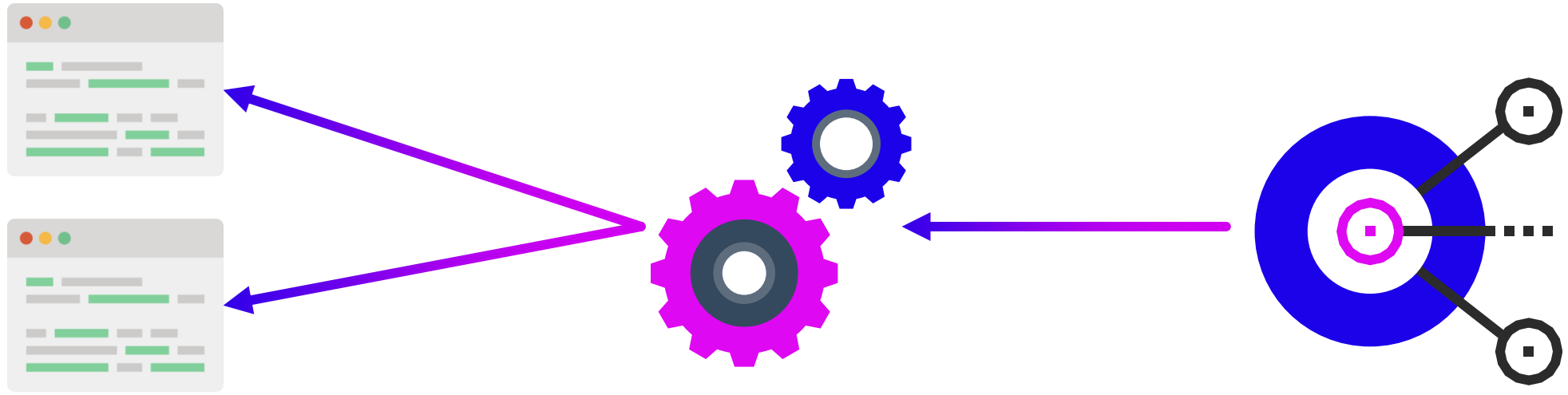
Programmable **Network Proxy**, running as a separate **Background Process**, without any **DOM Access**.

How are Service Workers Registered



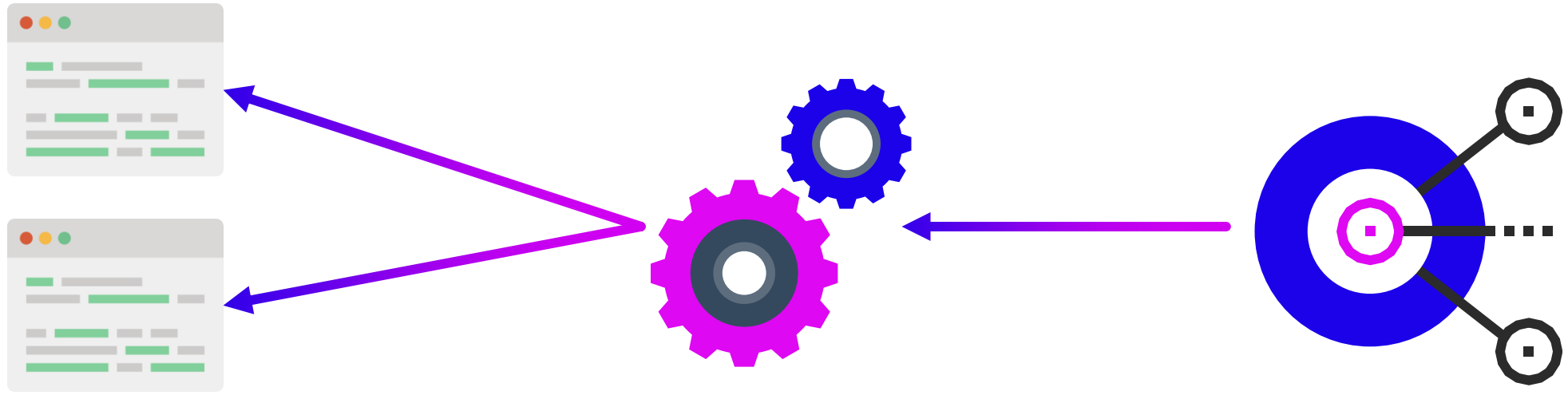

```
<script>
    navigator.serviceWorker.register('/sw.js');
</script>
```

What do Service Workers Do?



- **Cache** Data (CacheStorage)
- **Store** Data (IndexedDB)
- Receive **Push**
- Respond when **Offline**

What do Service Workers Do?



- **Intercept** HTTP Requests
- **Sync** Data in Background
- Hide **Flaky Connectivity** from the User

Service Workers Browser Support

IE	Edge *	Firefox	Chrome	Safari	Safari on iOS *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			87						
	88	86	88	13.1	13.7				
11	89	87	89	14	14.5	all	89	12.12	13.0
		88	90	TP					
		89	91						
			92						

Supported by **~95%** of browsers.

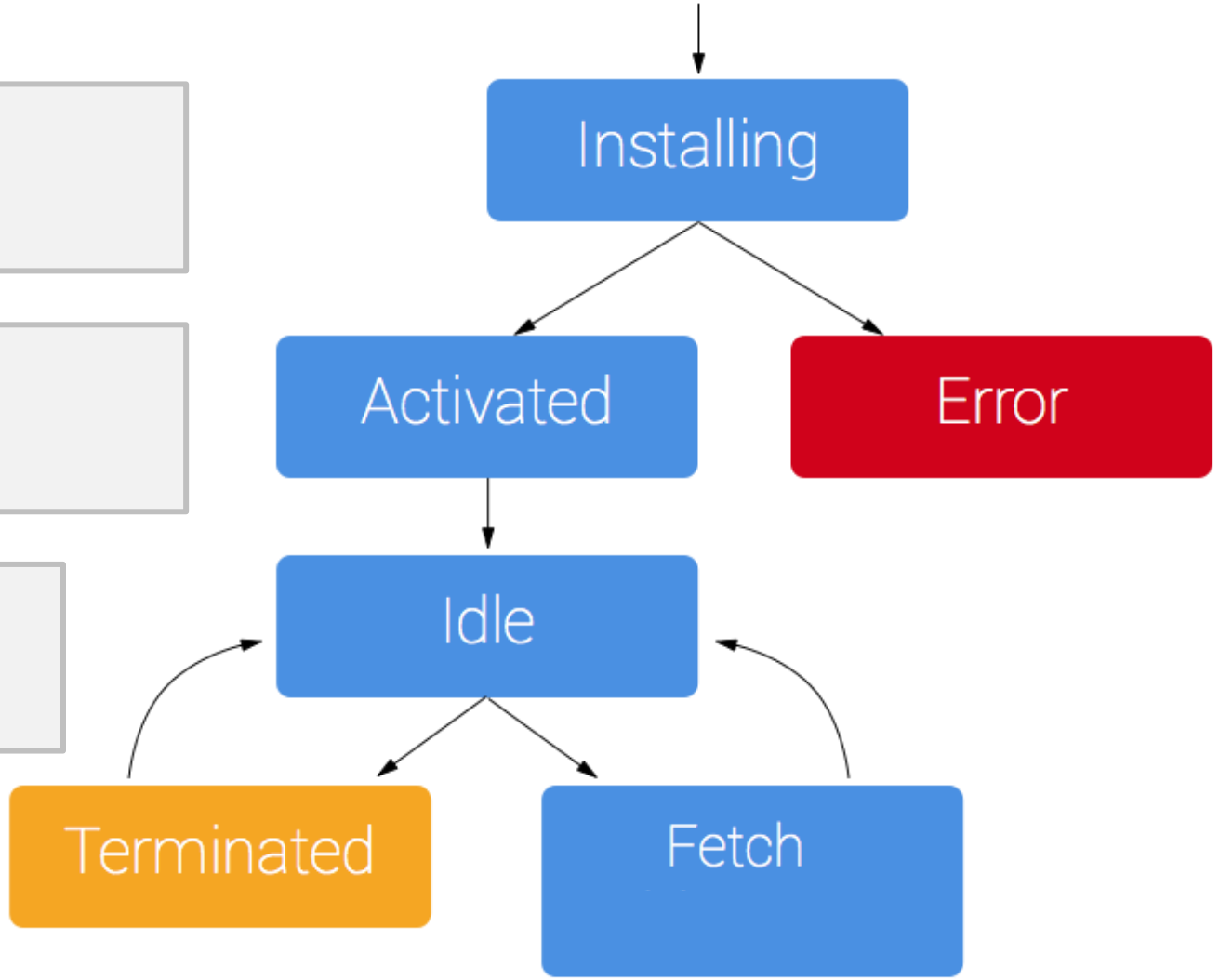
Requires **TLS Encryption**.

Service Worker Lifecycle

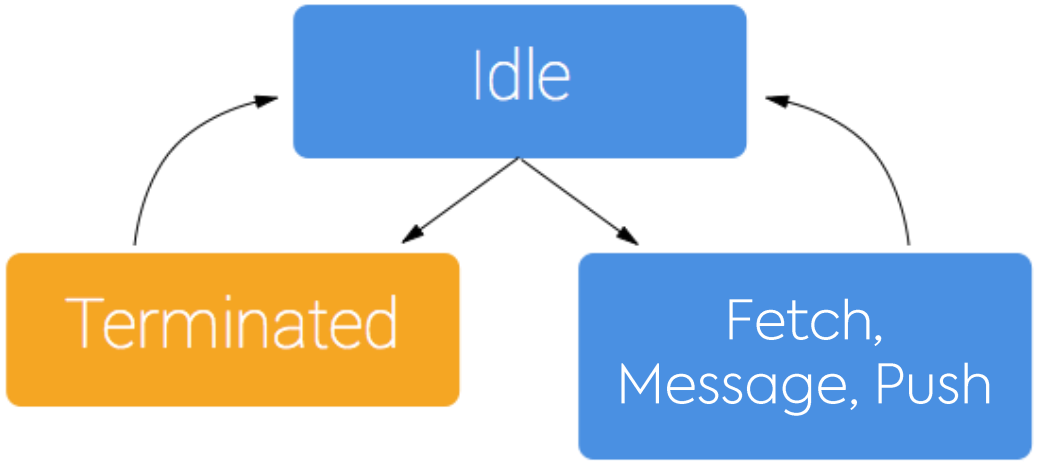
```
self.addEventListener('install', (event) => {
  // Perform install steps
});
```

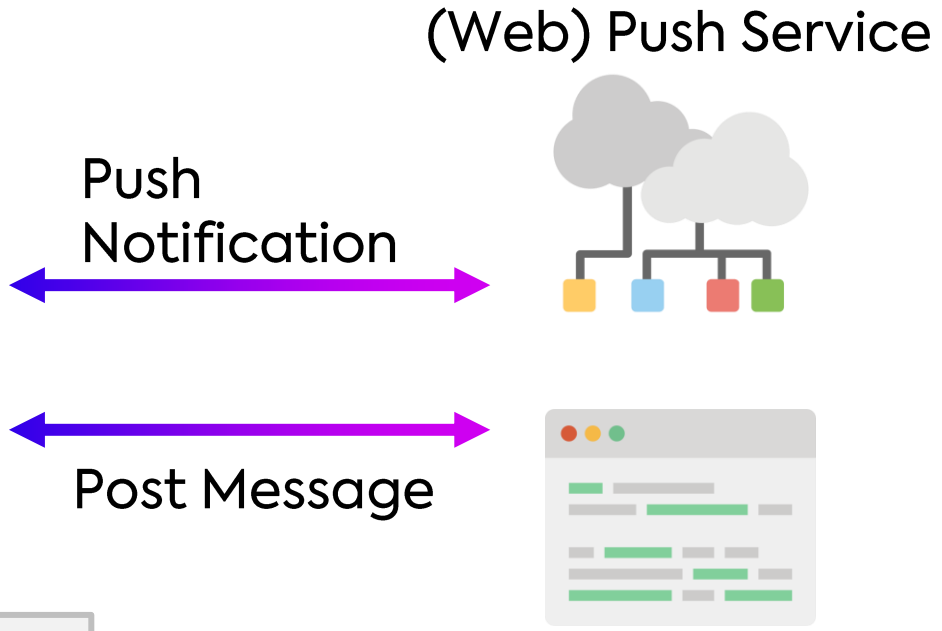
```
self.addEventListener('activate', (event) => {
  // Perform activate steps
});
```

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



Service Worker Communication



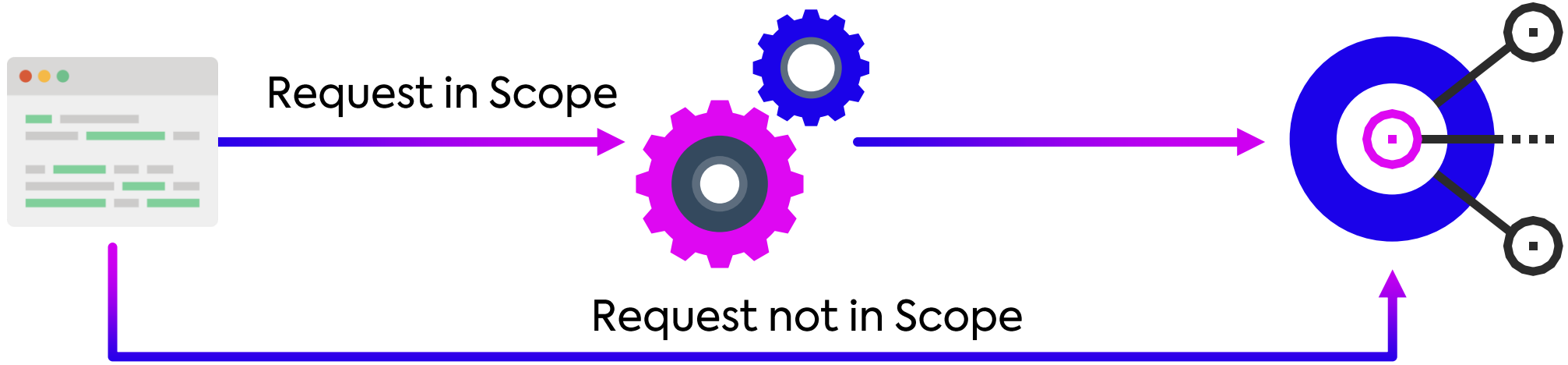



```
self.addEventListener('message', (event) => {
  // Receive message
});
```

```
// Send message to browser tab
const client = await clients.get('id');
client.postMessage(someJsonData);
```

```
self.addEventListener('push', (event) => {
  // Receive push notification
});
```

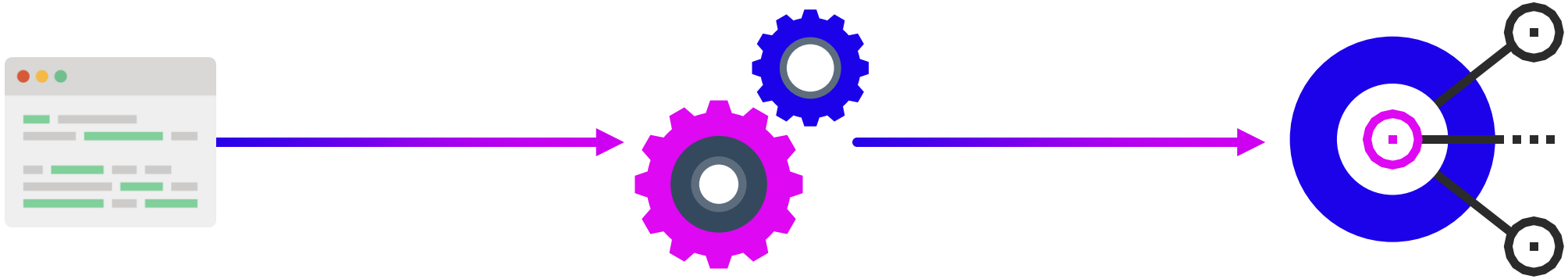
Service Worker Scope



Scope determines which requests go to the Service Worker

```
// Default (and maximum) scope is location of Service Worker
// Gets all requests starting with '/path/'
navigator.serviceWorker.register('/path/sw.js');
```

Service Worker Request Interception

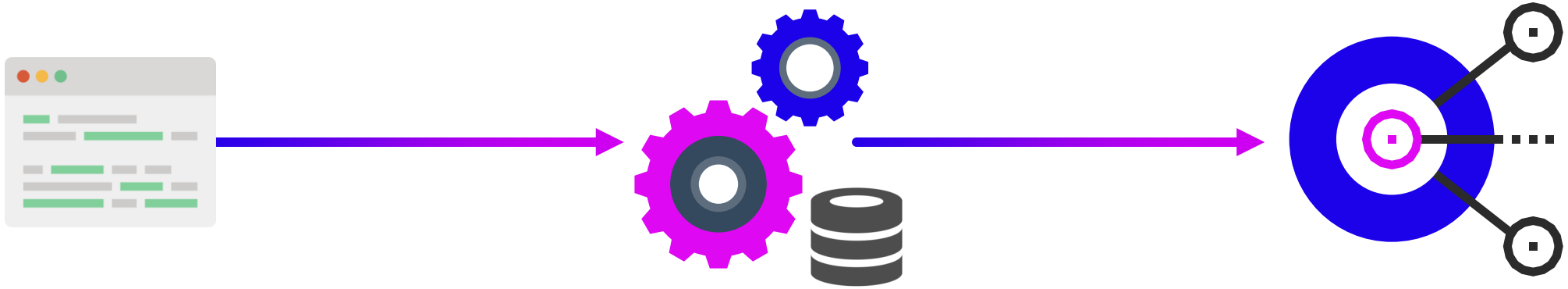


```
self.addEventListener('fetch', (event) => {
  // React to fetch event
  const { url } = event.request;
  event.respondWith((async () => {
    const request = new Request(url.replace('.com', '.de'))
    const response = await fetch(request);
    const text = await response.text();
    const newText = text.replace('Goethe', 'Schiller');
    return new Response(newText, { status: 200 });
  })());
});
```

There is so much you can do:

- **Rewrite** Requests
- **Change** Responses
- **Concat** Responses
- **Cache** Responses
- **Serve** Cached Data
- …

Service Worker Persistence



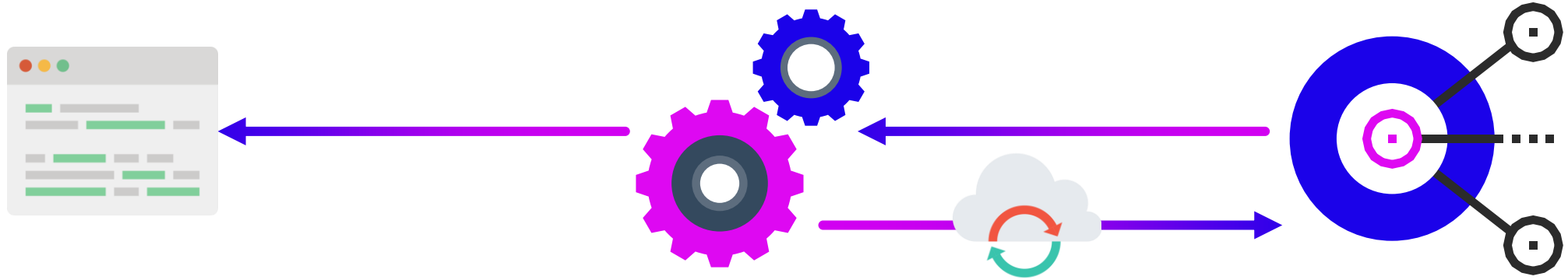
IndexedDB

an actual database in the browser

- Stores Data **Persistently**
- Stores **Structured** Data
- Supports **Range Queries**
- **Browser Support** 94%

Experimental

Service Worker Background Sync



One-off Sync

- executed when user is **online**
- **retried** when failed (exponential backoff)

Use Cases

- Save **file** when online again
- Send **email** when online again

Periodic Sync

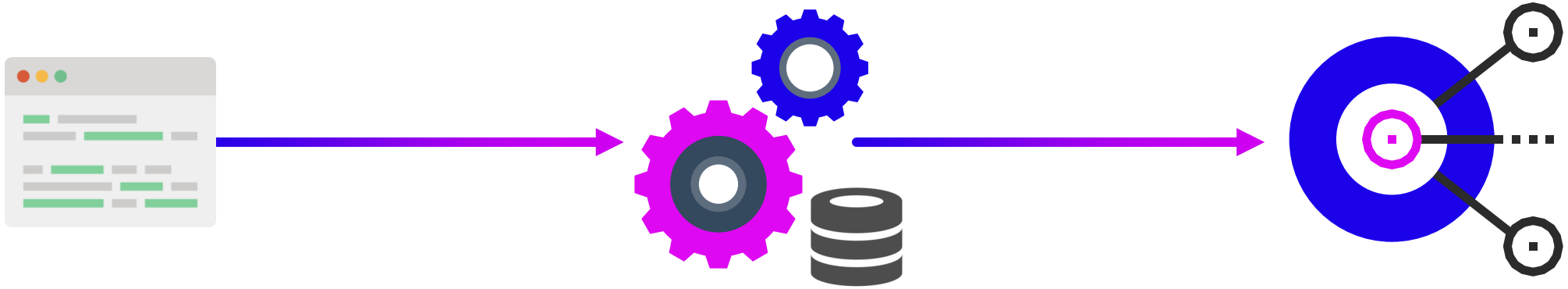
- executed when online, according to **period options**

Use Cases

- Load updates to **social media time-line** when browser closed

Service Worker Caching Strategies

Service Worker Caching



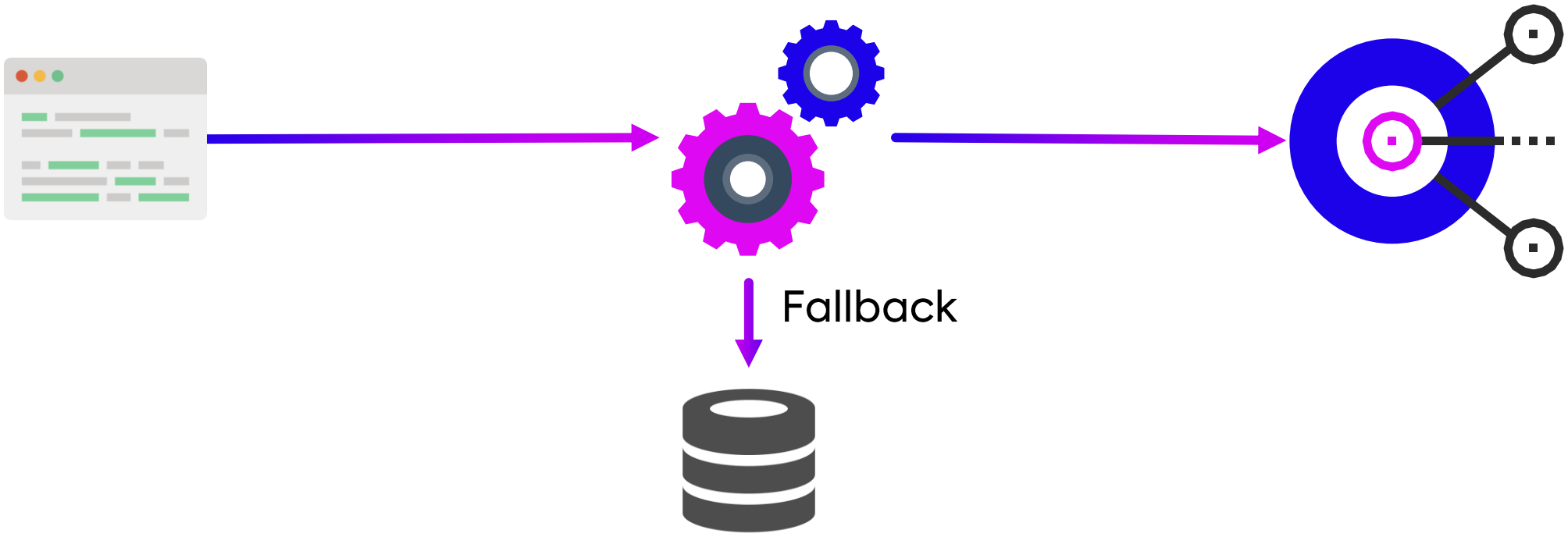
Cache Storage

Stores Request/Response pairs

Cache Storage

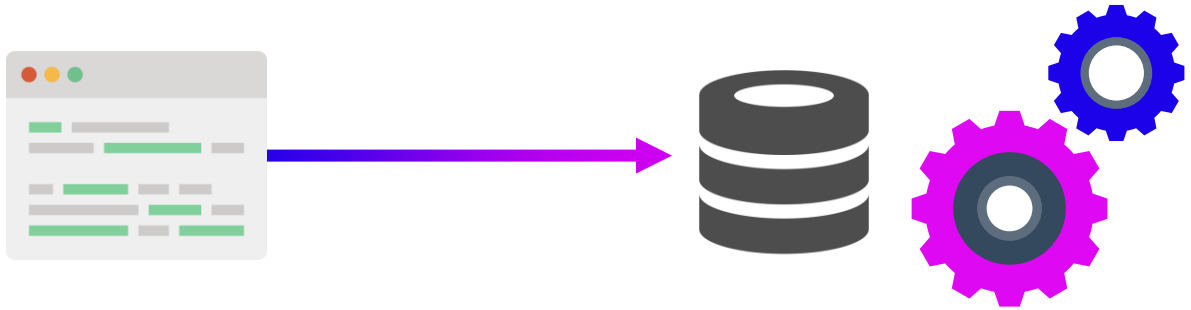
- **Programmatically** managed
- **Persistent** and non-expiring
- Supports only **HTTP**
- Only caches **GET** requests (no HEAD)

Caching Strategies Network, Cache Fallback



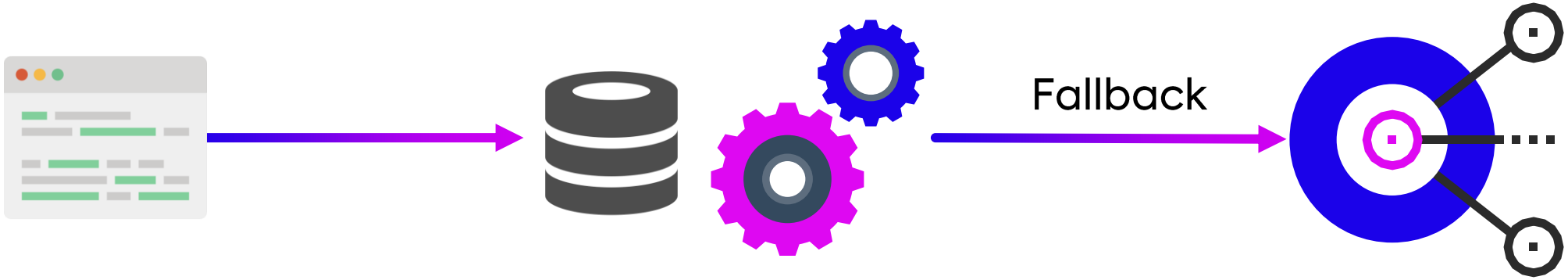
Gets requests from network, the cache acts as fallback (offline mode).

Caching Strategies Cache Only



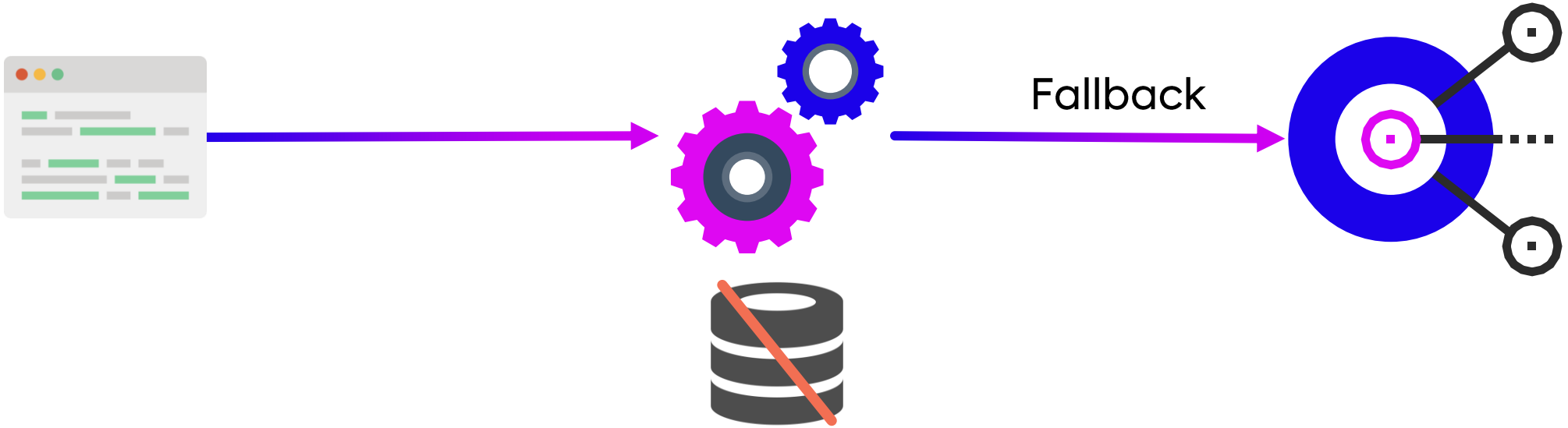
Gets all requests from cache or fails.

Caching Strategies Cache, Network Fallback



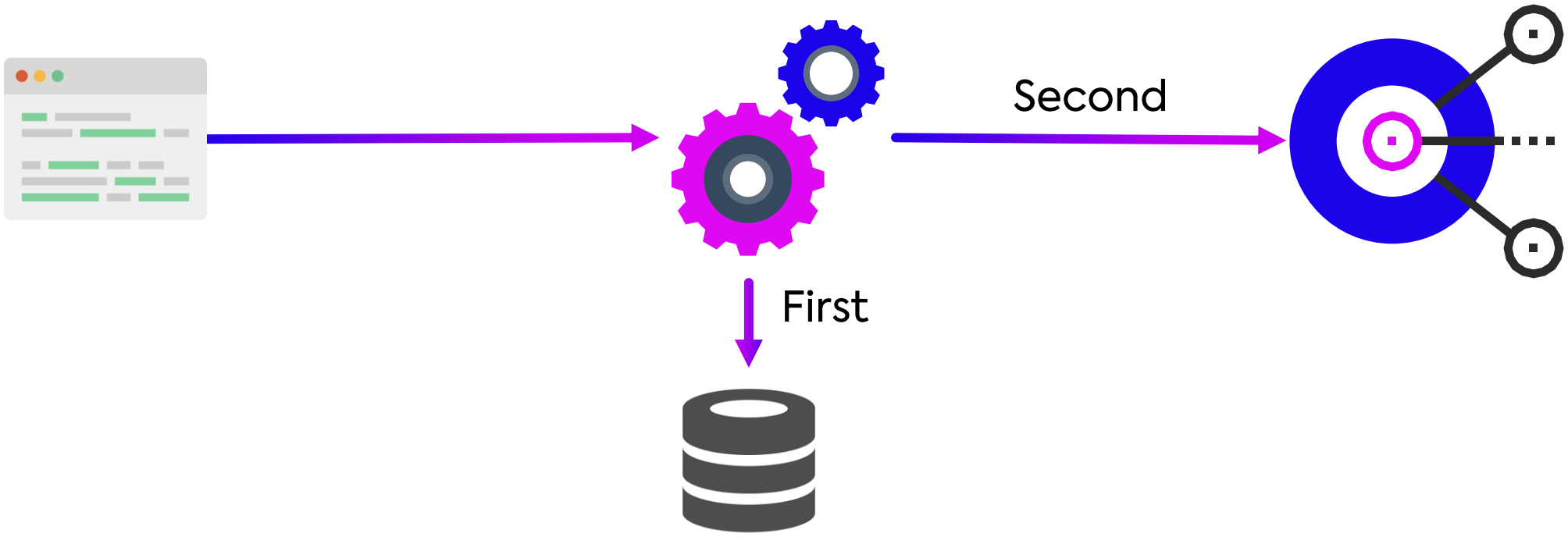
Gets requests from cache & uses network as fallback.

Caching Strategies Network Only



Gets requests from network only.

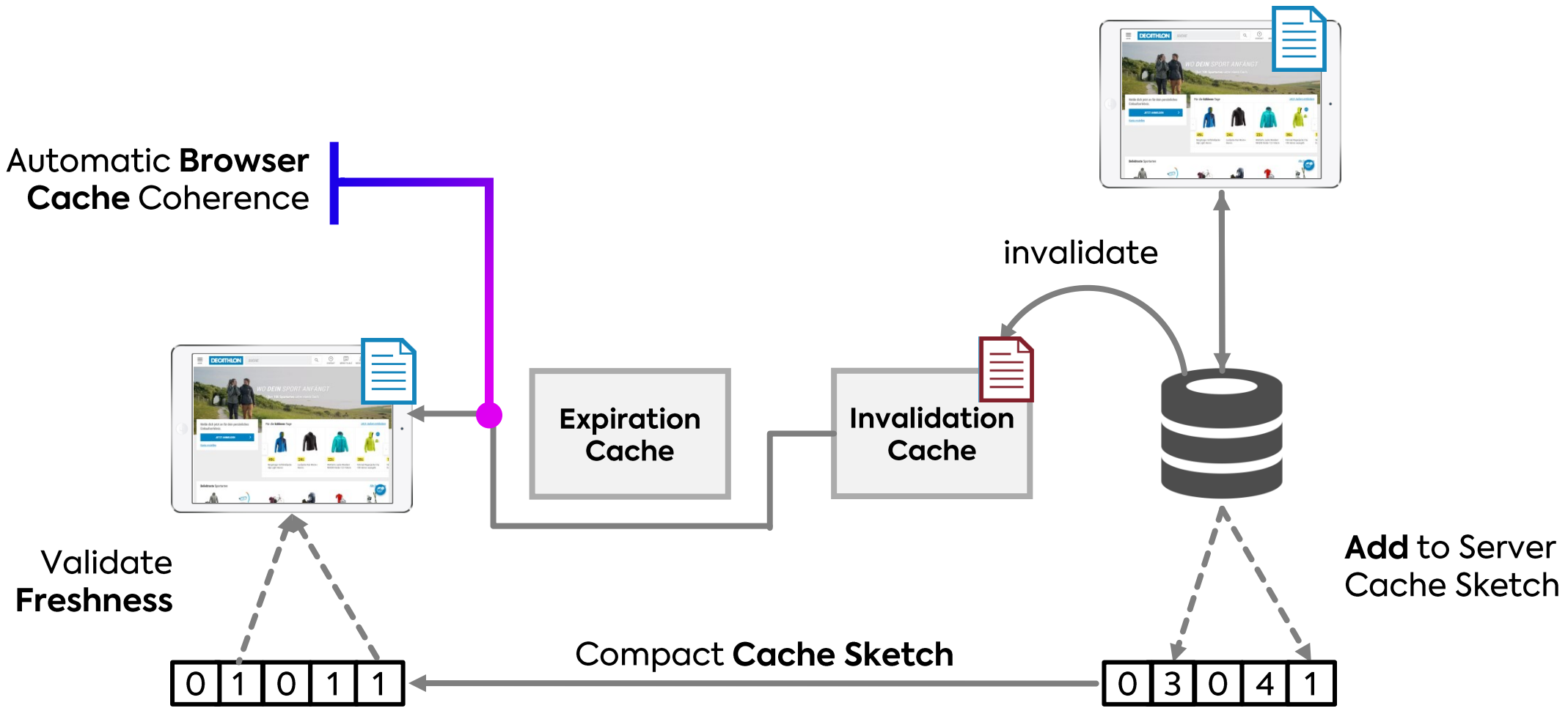
Caching Strategies Cache, Then Network



Gets requests from cache first and from network in background.

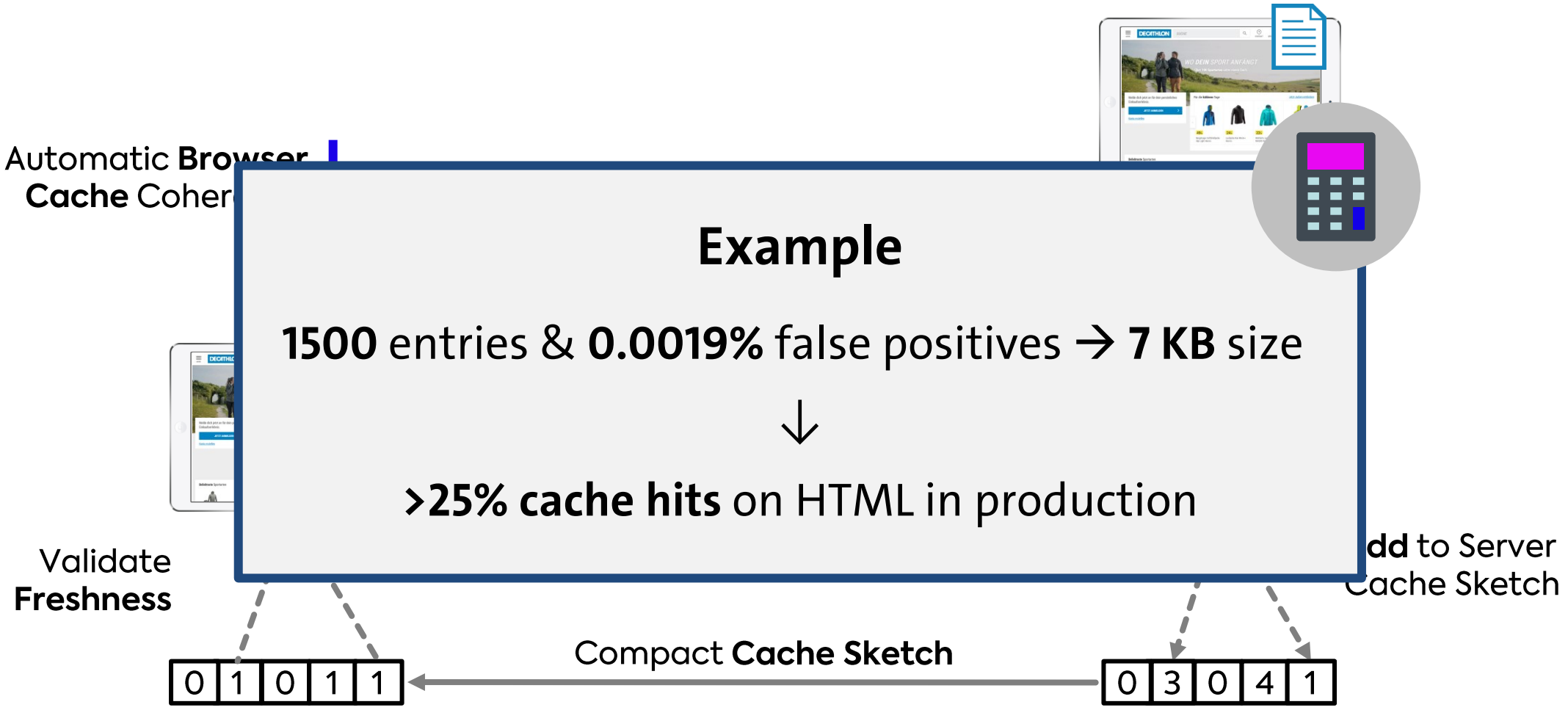
Main Challenges: Cache Coherence & Personalization

How Speed Kit Solves Cache Coherence



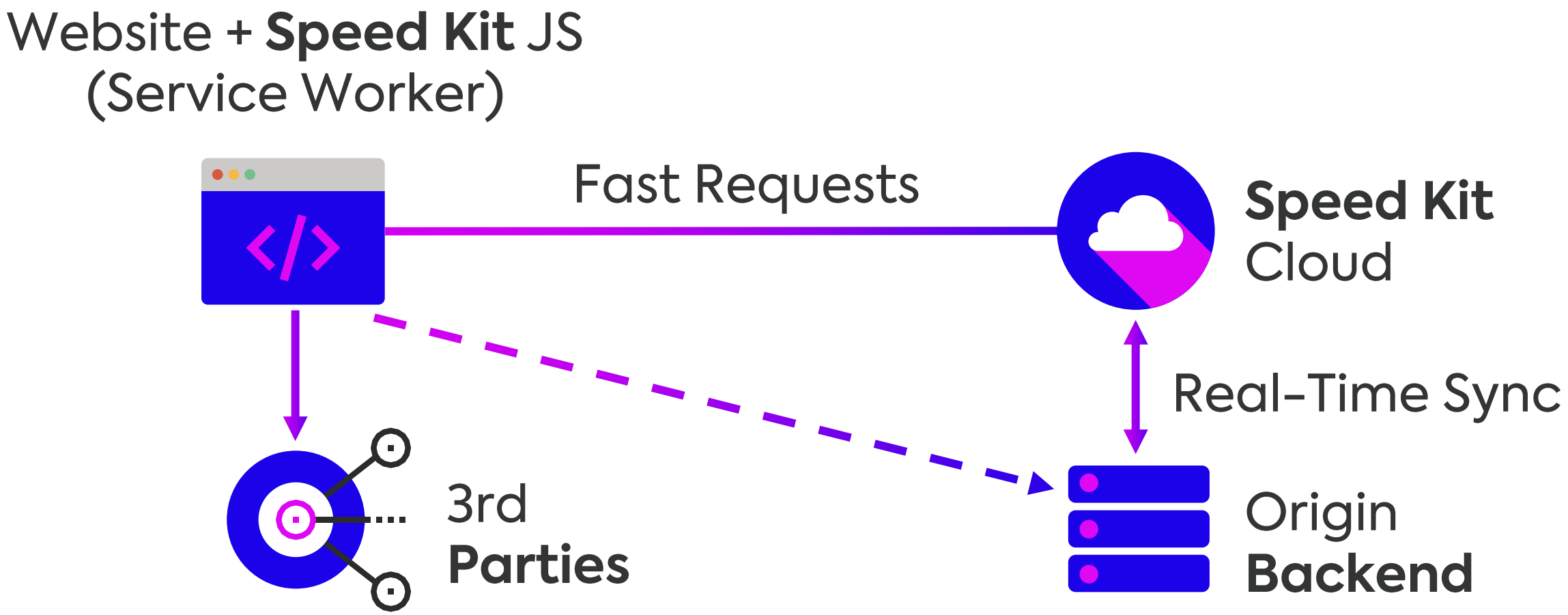
Felix Gessert, Michael Schaarschmidt, Wolfram Wingerath, et al.: *The Cache Sketch: Revisiting Expiration-Based Caching in the Age of Cloud Data Management*. BTW 2015

How Speed Kit Solves Cache Coherence



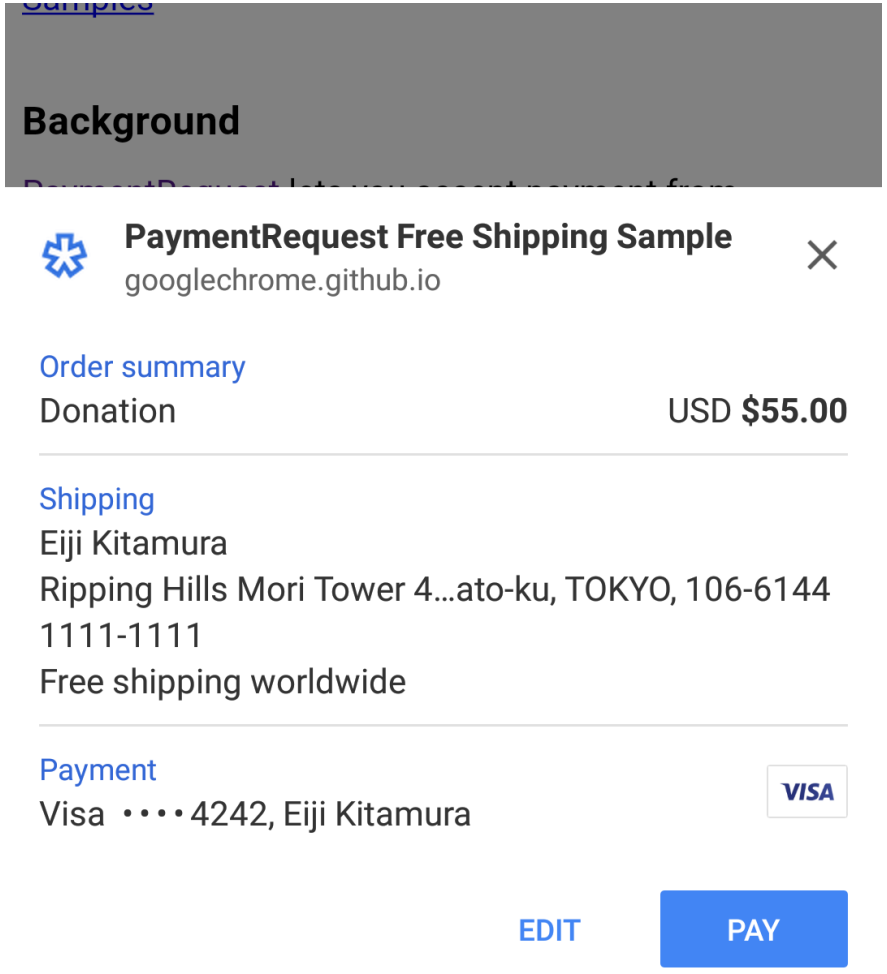
Felix Gessert, Michael Schaarschmidt, Wolfram Wingerath, et al.: *The Cache Sketch: Revisiting Expiration-Based Caching in the Age of Cloud Data Management*. BTW 2015

How Speed Kit Works



Fancy PWA Features

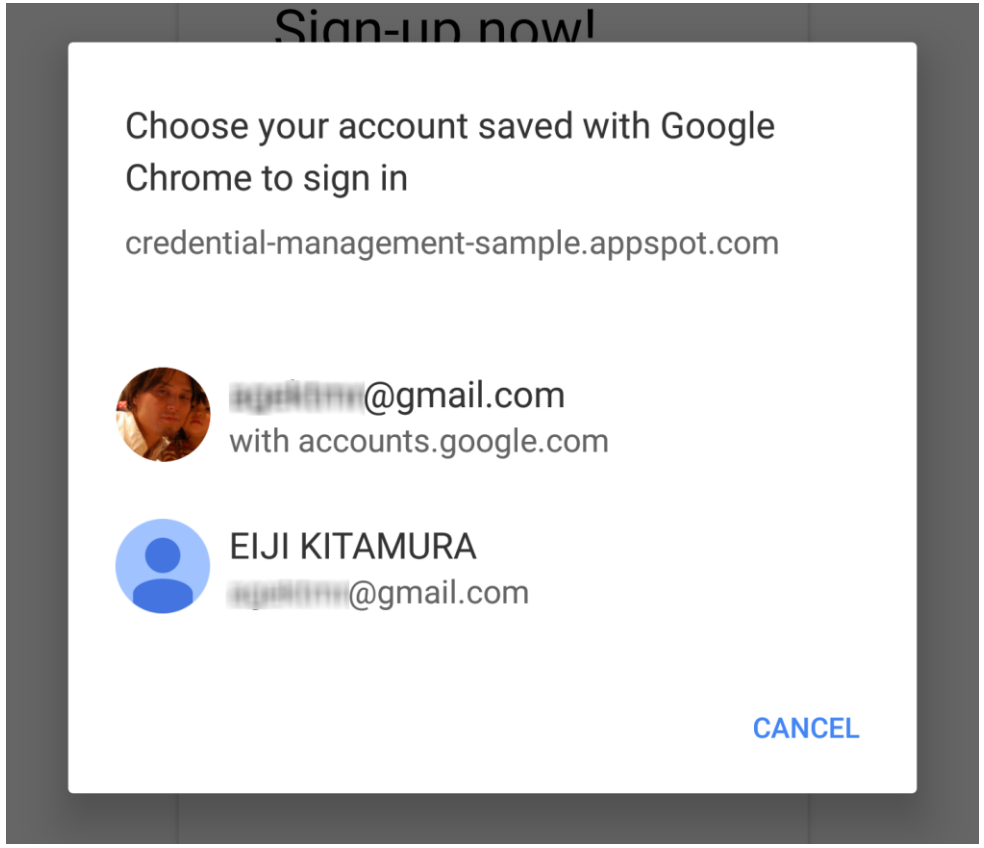
Integrated Payment



Web Payment APIs

- Goal: replace traditional **checkout** forms
- Just ~10 LOC to implement **payment**
- Vendor- & Browser-**Agnostic**

User Management



Credentials Management API

1. Click **Sign-in** → Native Account Chooser
2. Credentials API **stores** information for future use
3. **Automatic** Sign-in afterwards

Conversational Interfaces

Web Speech API

Native Speech Recognition in the Browser:

```
annyang.addCommands({
  'Hello Code Talks': () => {
    console.log('Hello you.');
  }
});
```

Seamless Sharing

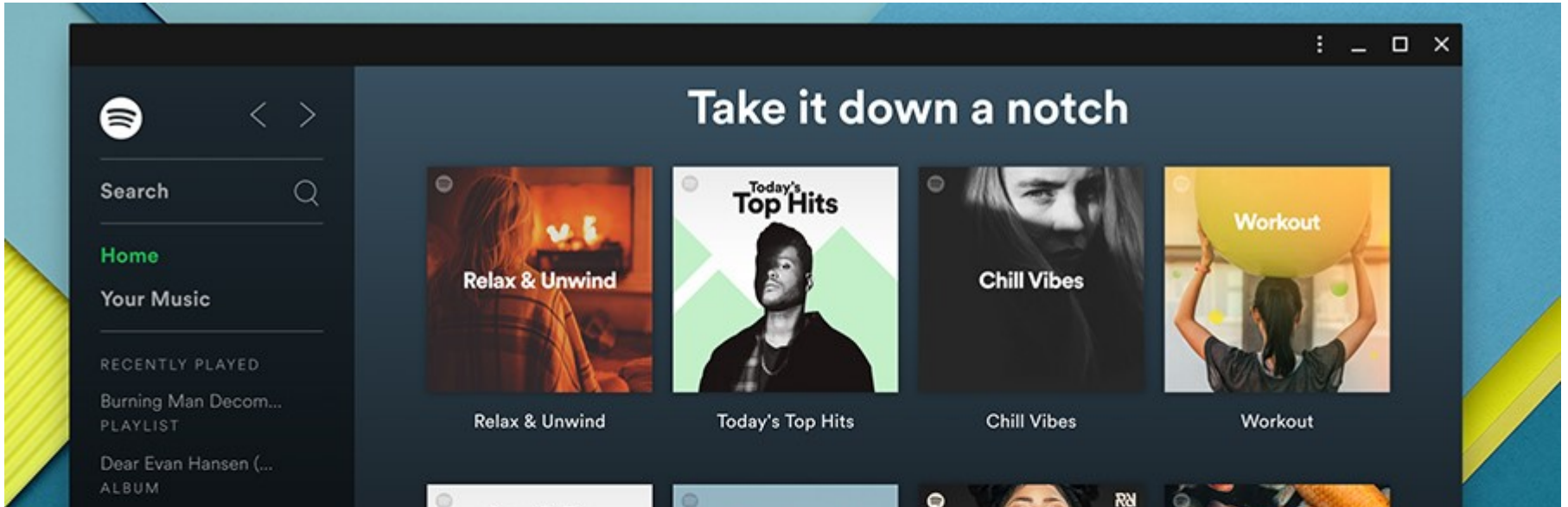
Web Share API

- **Share** site through native share sheet UI
- Service Worker can register as a **Share Target**

Cross-Device Support Through Desktop PWAs

Chrome >73 now supports **Desktop PWAs** on every platform

Customizable installation process/UI (with Event beforeinstallprompt)

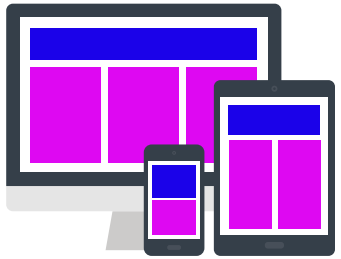


Wrapup: Frontend Design

Traditional Optimization

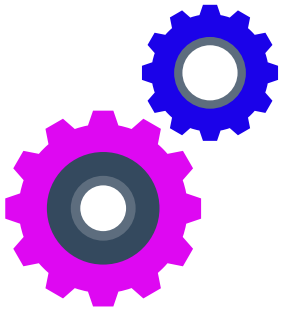
Minimize page weight, blocking & critical resources

PWAs



Novel alternative to native apps

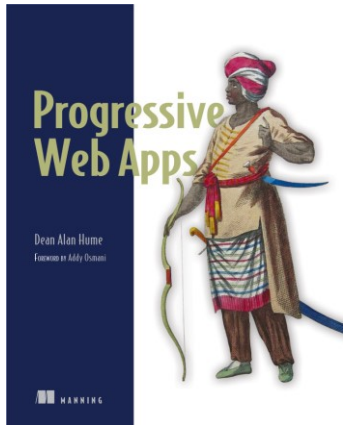
Service Workers

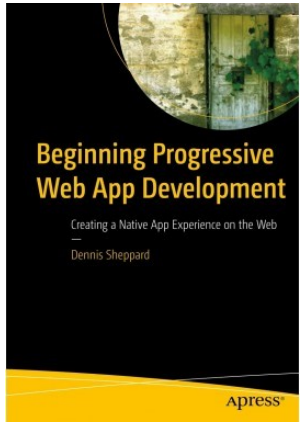


Powerful programmable network proxy

Learn More About PWAs & Service Workers

Recommended Books





Blogs

Baqend Blog
On Building a Faster Web

https://blog.baqend.com/

Ilya Grigorik
Internet plumber

https://www.igvita.com/

Jake Archibald wrote...
I discovered a browser bug

https://jakearchibald.com/

Guides & Tutorials

Progressive Web Apps

A new way to deliver amazing user experiences on the web.

https://developers.google.com/web/progressive-web-apps/

Progressive web apps

Jump to: PWA advantages Core PWA guides Technology guides

https://developer.mozilla.org/en-US/docs/Web/Apps/Progressive

Tutorial 18: Going for Speed

09:00	Introduction
09:15	Part 1: Efficient Frontend Design
09:40	Q&A
09:50	Coffee Break
10:00	**Part 2: High-Performance Networking**
10:40	Q&A
10:50	Coffee Break
11:00	Part 3: Scalable Backend Architectures
11:40	Q&A
11:50	Coffee Break
12:00	Part 4: Performance Tracking & Analysis
12:00	The Core Web Vitals (Google Guest Speaker!)
12:30	Measuring Web Performance
12:50	Q&A

Baqend