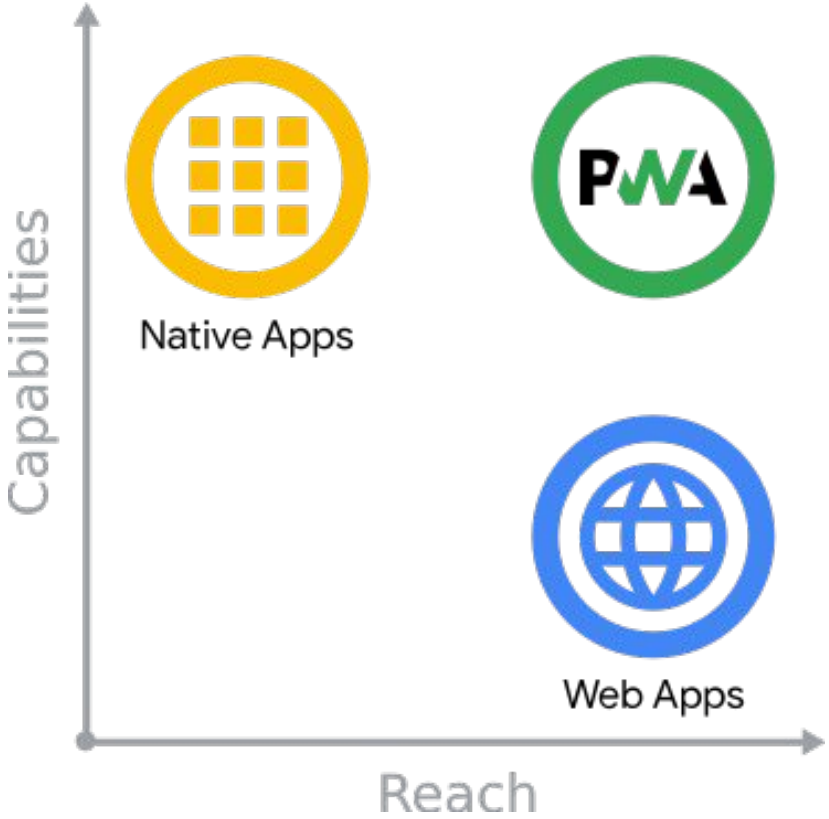
Lecture 9

Additional Technologies

Frontend Web Development

Application Architecture

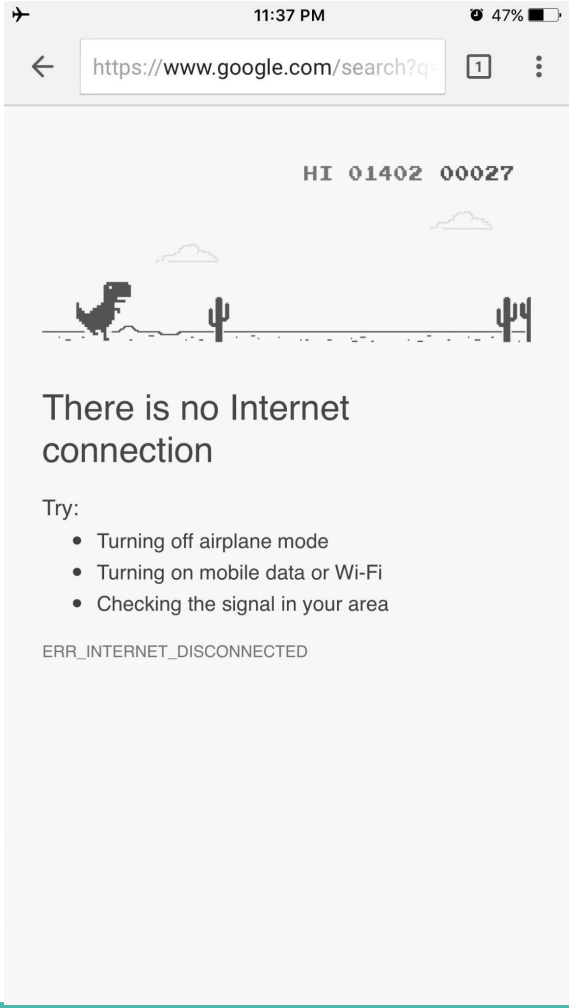
Progressive Web Apps



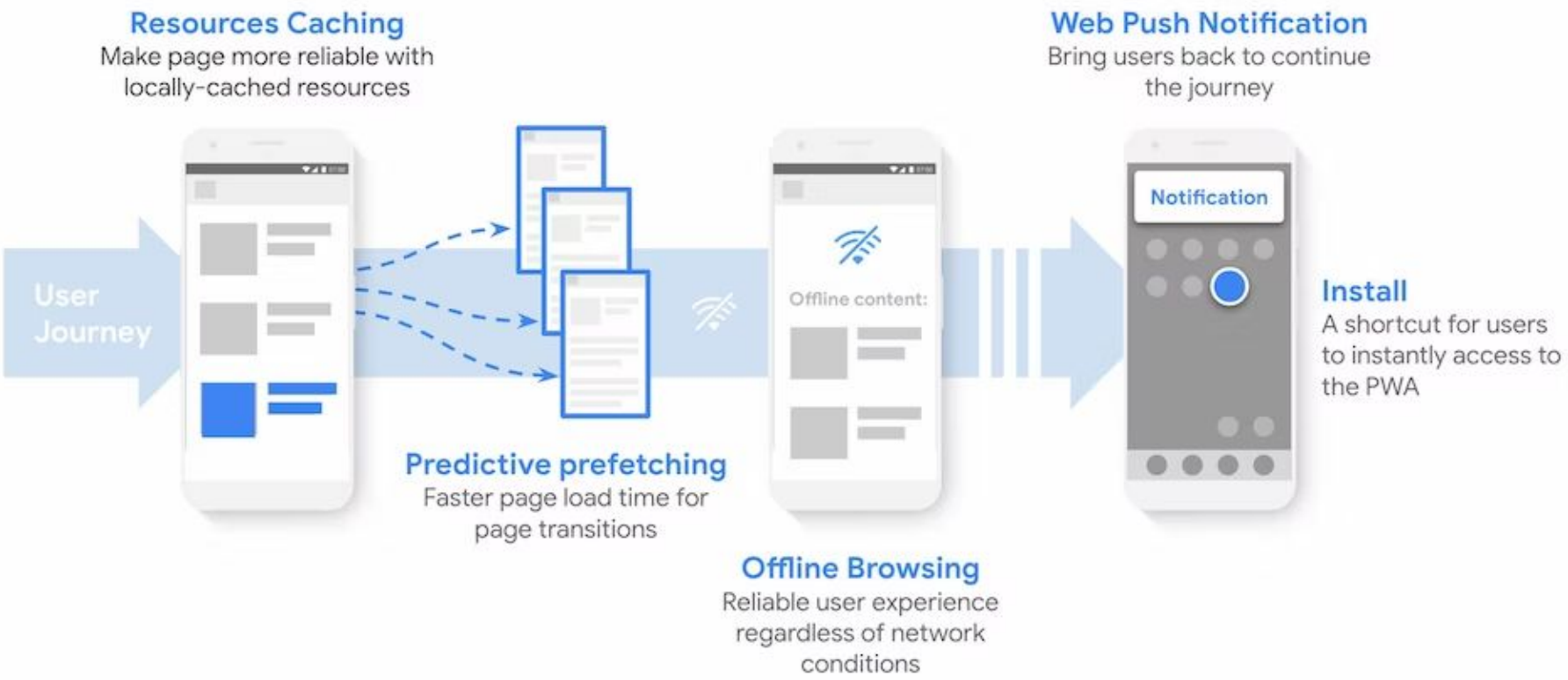
Progressive Web Apps

The 3 pillars of PWAs:

- **Capable**: Using modern APIs, they can really do a lot
- **Reliable**: Works fast even on slow (or no internet)
- **Installable**: Can be added to the device's apps shelf



Progressive Web Apps - Offline first experience



Add to home screen (A2HS) - WD

The ability for a user to "install" a website and use it as if it was a natively installed app. To enable this behaviour, a website must serve a valid Web App Manifest and load it's assets through a Service Worker.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mobile *	Android Browser *	Firefox for Android
	12-16		2-75							
4-38	[1] 17-18		[3] 76-85			3.2-11.2				
39-110	79-110	3.1-16.3	86-110	6-10		[2] 11.3-16.3	4-19.0	12-12.1	2.1-4.4.4	
111	111	16.4	111	11	111	16.4	20	73	111	110
112-114		16.5-TP	112-113			16.5				

Notes | Test on a real browser | Known issues (0) | Resources (8) | Feedback

WebKit status: Partially Supported

[1] A manifest could be used to list apps in the Microsoft Store, which would use Edge as a back-end.

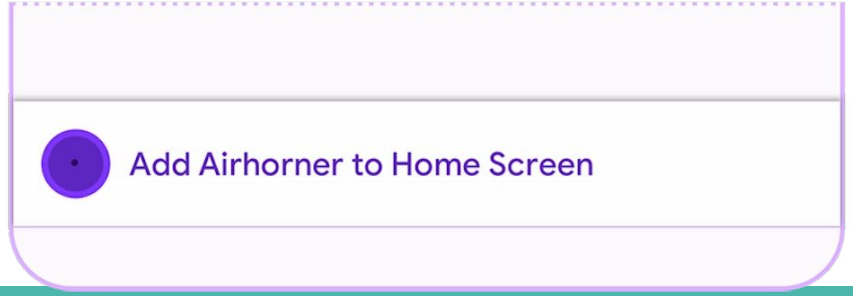
[2] Safari on iOS does not support A2HS in WebViews like Chrome and Firefox.

[3] Firefox is experimenting with desktop support behind the `browser.ssb.enabled` flag.

Progressive Web Apps - Requirements

Installability criteria:

- Meets user engagement heuristics
  - User interacted with the page (click or tap at least)
  - User spent 30+ seconds viewing the page
- Served over HTTPS
- Includes a web manifest file
- Registers a service worker with a `fetch` handler



Web Manifest

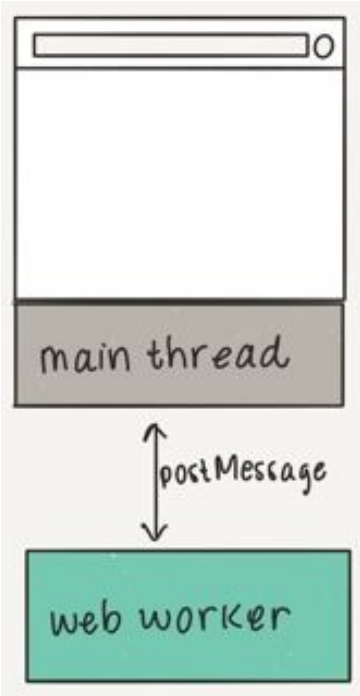
```html
<link rel="manifest" href="/manifest.json">
```

```json
{
  "short_name": "Weather",
  "name": "Weather: Do I need an umbrella?",
  "icons": [{
    "src": "/images/icons-512.png",
    "type": "image/png",
    "sizes": "512x512"
  }],
  "start_url": "/?source=pwa",
  "background_color": "#3367D6",
  "display": "standalone",
  "scope": "/",
  "theme_color": "#3367D6",
  "description": "Weather forecast information",
}
```

Web Workers

- Separate threads that do not interact with the UI
- No shared memory
- Asynchronous communication (via `postMessage`)
- Can be utilized for computationally-heavy tasks
  - Fetching large files
  - Processing images

```
const worker = new Worker("./worker.js");
```

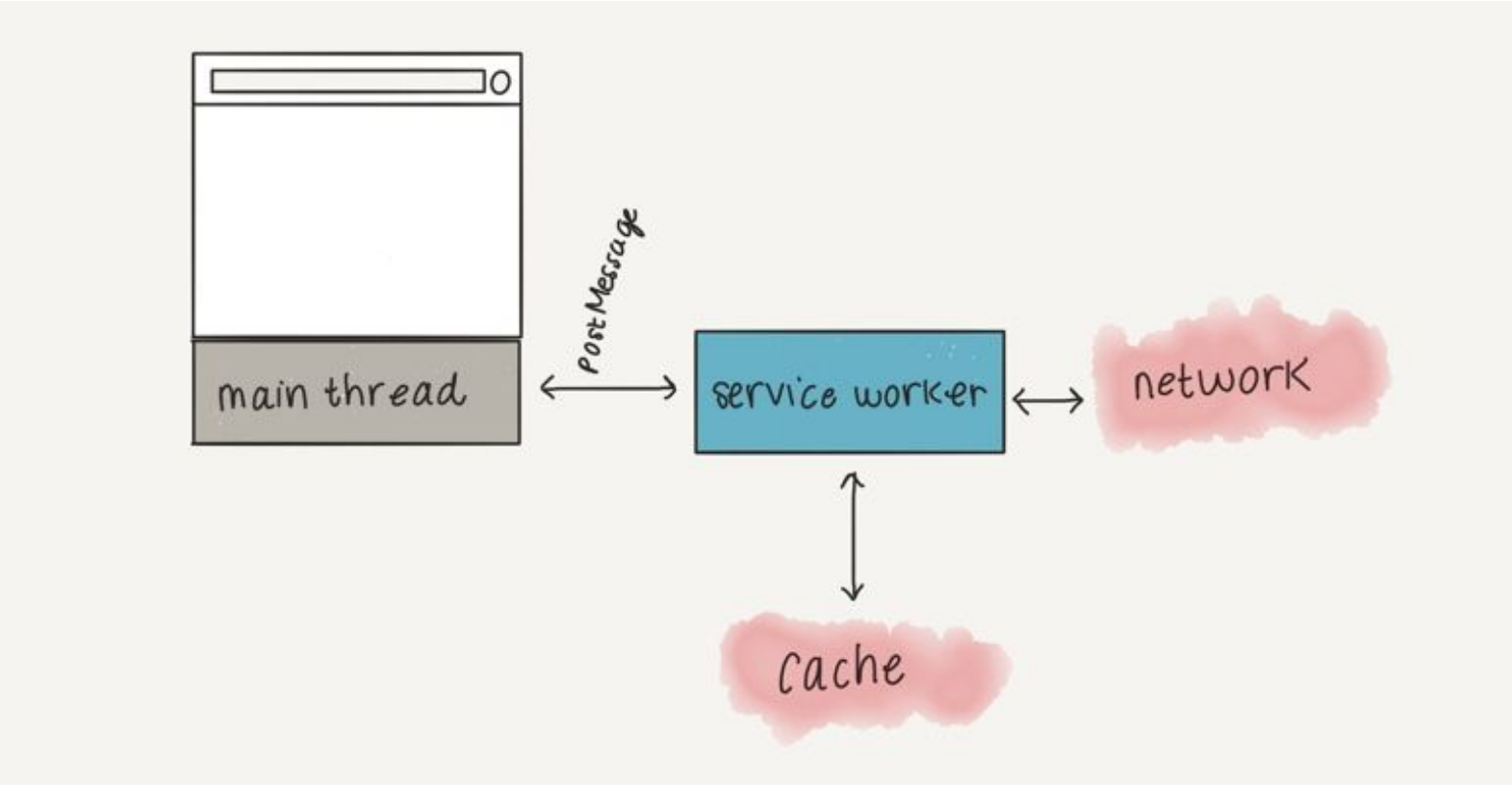


Service Worker

- A more specific kind of a Web Worker
- Acts as a proxy between the browser and the network
- Only one SW registered at a time
- Utilized for providing an offline-first experience

```
navigator.serviceWorker.register('/service-worker.js');
```

Service Worker



Service Workers vs Web Workers

	Web Workers	**Service Workers**
Tab Control	Many per tab	One for all tabs
Lifespan	Same as tab	Independent
Good for	Parallelism	Offline support

Service Worker - Example

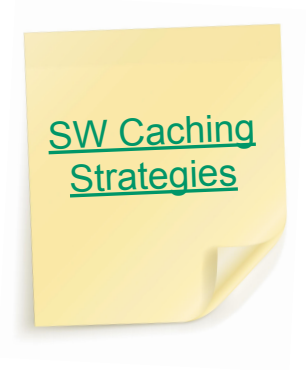
```
self.addEventListener('install', event => {
  console.log('sw install');
  event.waitUntil(
    caches.open('static-v1').then(cache => {
      cache.addAll([ '/', '/static/styles.css' ])
    })
  );
});

self.addEventListener('activate', () => {
  console.log('sw activate');
  clients.claim();
  // TODO: Delete old caches
});
```

Service Worker - Example

```
self.addEventListener('fetch', event => {
  event.respondWith((async () => {
      console.log(`Fetching resource: ${event.request.url}`);
      const r = await caches.match(event.request);
      if (r) { return r; }
      const response = await fetch(event.request);
      const cache = await caches.open(cacheName);
      console.log(`Caching resource: ${event.request.url}`);
      cache.put(e.request, response.clone());
      return response;
    })()
  );
});
```

Caching strategies

- Cache first, then network
- Network first, then cache
- Stale-while-revalidate
- Cache only
- Network only



Web Components

3 technologies/APIs combined together:

- **Custom Elements API**
- **Shadow DOM**: A DOM tree encapsulated to an element
- **HTML templates**: The `<template>` and `<slot>` elements

Typical usage: Define a template in the HTML, clone it into the element's shadow DOM, then register the element using the custom elements API.

Web Components - Example

```
class AppDrawer extends HTMLElement {
  constructor() {
    this.addEventListener('click', e => {
      if (this.disabled) return;
      this.toggleDrawer();
    });
  }
  get disabled() { return this.hasAttribute('disabled'); }
  toggleDrawer() { … }
}
window.customElements.define('app-drawer', AppDrawer);
```

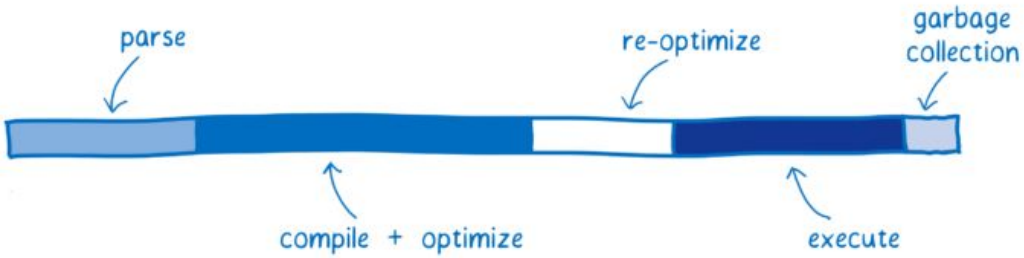
```
<app-drawer></app-drawer>
```

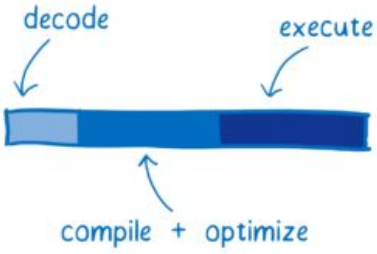
WebAssembly (WASM)

A portable binary instruction format for a stack-based virtual machine.

Runs with near-native performance.

WebAssembly is designed to **complement** and run **alongside** JavaScript.





WebAssembly - Example

adder.wasm

```
(module
  (type $t0 (func (param i32 i32) (result i32)))
  (func $_add (type $t0) (param $p0 i32) (param $p1 i32) (result i32)
    get_local $p0
    get_local $p1
    i32.add)
  (export "_add" (func $_add)))
```

adder.c

```c

int add(int a, int b) {
    return a + b;
}
```

WebAssembly - Language support

Language	Browser	Other	WASI	Notes
JavaScript	⏳	⏳	⏳	
Python	⏳	✅	✅	
Java	✅	✅	❌	
C# and .NET	✅	✅	✅	Covers .NET as well
C++	✅	✅	✅	
TypeScript	❌	⏳	❌	Consider AssemblyScript
C	✅	✅	✅	
R	✅	❌	❌	
Go	✅	✅	✅	Via Go and TinyGo
Kotlin	✅	⏳	⏳	
Rust	✅	✅	✅	
Dart	⏳	❌	❌	

WASM Language Support

WASI

Communication

WebSockets

A 2-way interactive communication session between the browser and server.

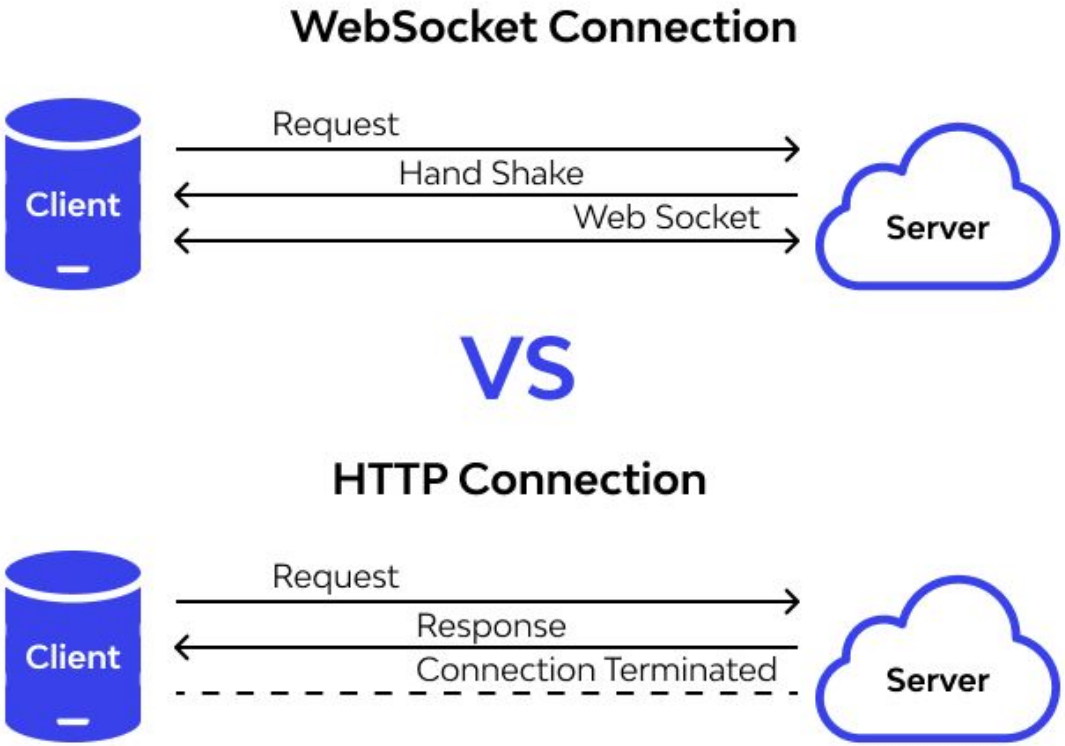
Like HTTP, it builds on TCP (both are Layer 7 in OSI model).

No overhead of HTTP headers, handshakes for every request, ...

Persistent full-duplex connection.

Enables server to initiate a data transfer without the use polling.

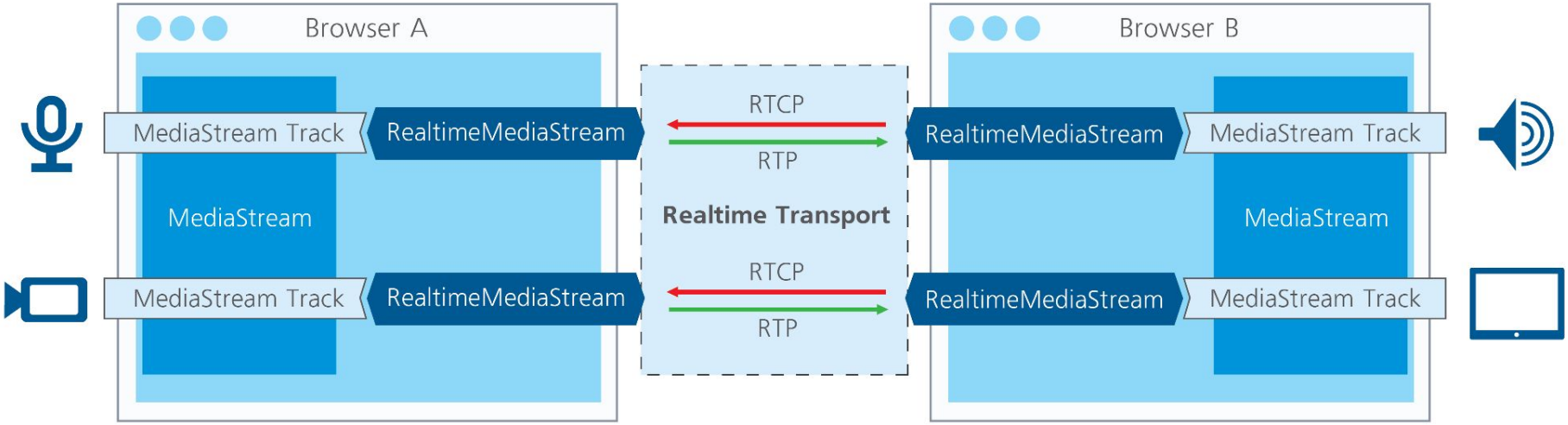
WebSockets


WebSocket Connection
Client
Request
Hand Shake
Web Socket
Server
VS
HTTP Connection
Client
Request
Response
Connection Terminated
Server

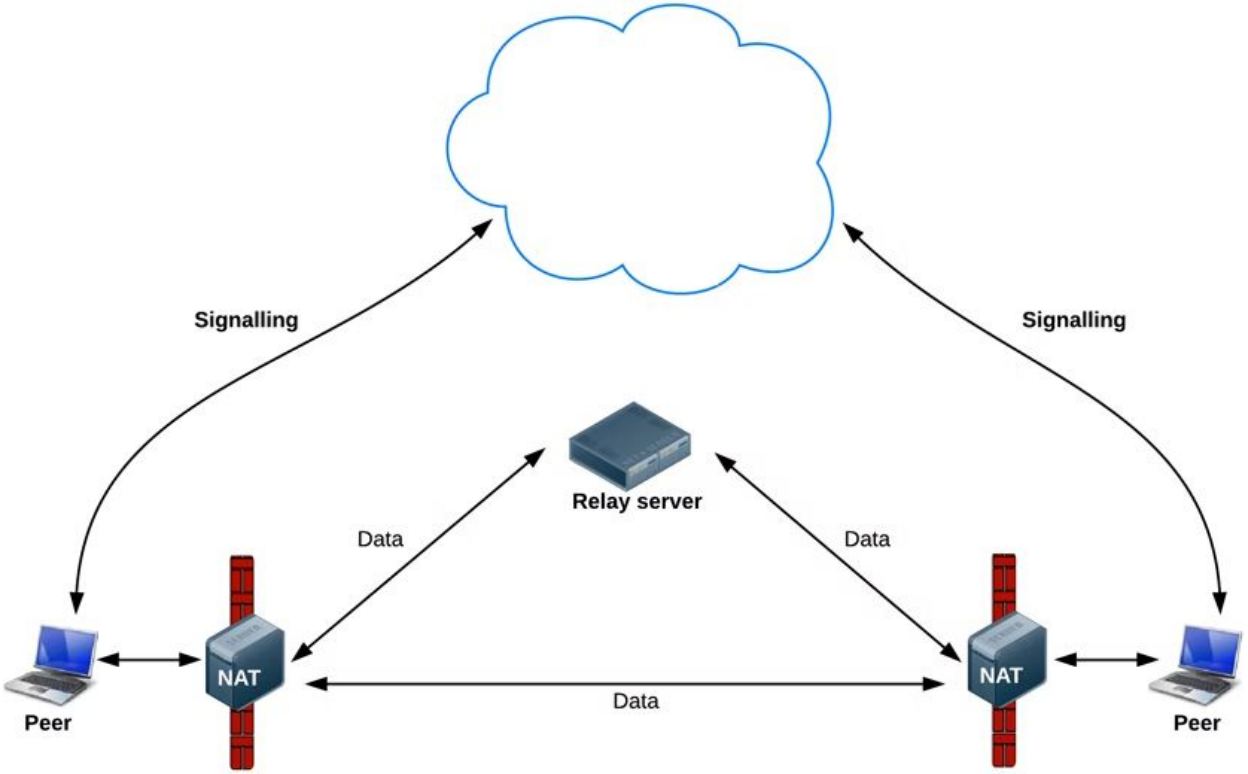
WebRTC

Web Real-**T**ime **C**ommunication

Peer-to-peer real-time streaming solution.



WebRTC


Signalling
Signalling
Relay server
Data
Data
NAT
NAT
Data
Peer
Peer

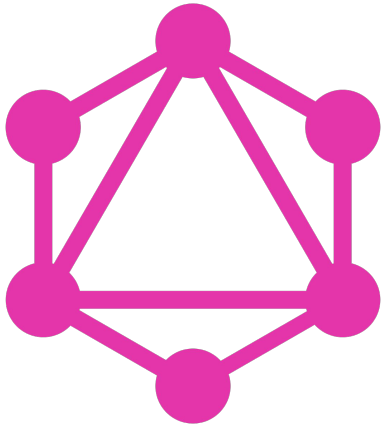
GraphQL

An alternative to RESTful APIs where we have just one (`POST`) endpoint.

Requests describe the format of data needed for the *query*, or how to *mutate*.

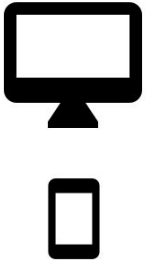
Can query multiple resources or parts of a resource in the same request.

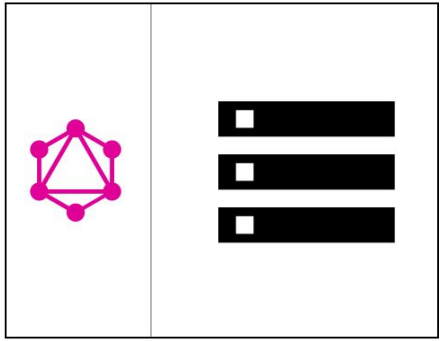
Strongly typed.



GraphQL - Example

```
query {
  User(id: “er3tg439frjw”) {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```





```
{
  “data”: {
    “User”: {
      “name”: “Mary”,
      “posts”: [
        { title: “Learn GraphQL today” }
      ],
      “followers”: [
        { name: “John” },
        { name: “Alice” },
        { name: “Sarah” },
      ]
    }
  }
}
```

Data Persistence

Web Storage API

`localStorage` & `sessionStorage`

Simple key-value pairs. Only strings.

```
localStorage.setItem("name", "Human");
localStorage.getItem("name"); // Human
```

Name	Value
ads.adserverDownvote	{"t3_5lo2wf":"https://engine.a.redditmedia.com/
ads.adserverUpvotePix	{"t3_5lo2wf":"https://engine.a.redditmedia.com/
ads.recent	["t3_5lo2wf"]
newsletterbar.seen	true
plugin.RES.state	INACTIVE
ui.shown.welcome	true

Web Storage API - Comparison

	Cookies	LocalStorage	SessionStorage
Capacity	4kb	5-10 Mbs(Browser Dependent)	5 Mbs
Accessibility	All windows	All windows	Private to tab
Expiration	Manually Set	Never expires	On tab close.
Passed in request	Yes	No	No
Storage	Browser and Server	Browser Only	Browser Only

Web SQL

An SQLite database inside the browser.

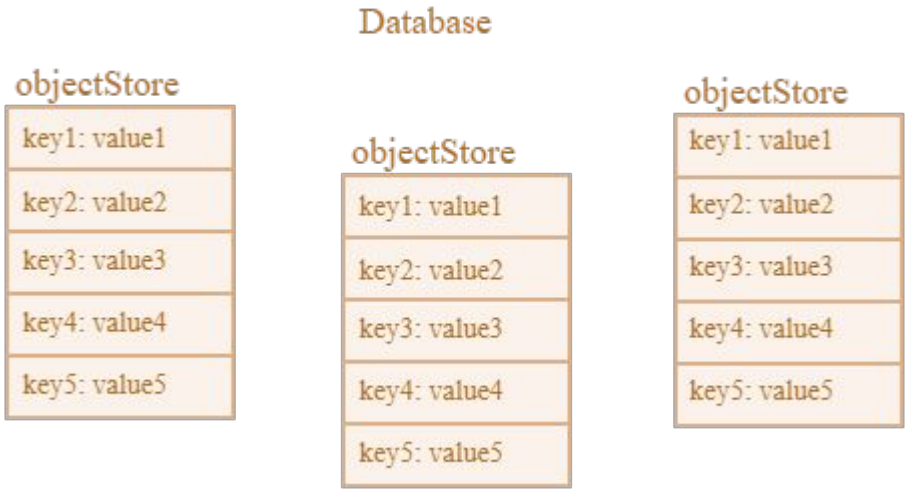
Deprecated in 2010 (for good reason)

IndexedDB

"A low-level API for client-side storage of significant amounts of structured data, including files/blobs" - *MDN*

Basically, an in-browser SQL-*like* RDBMS, except that it uses objects.

Supports transactions and indexes

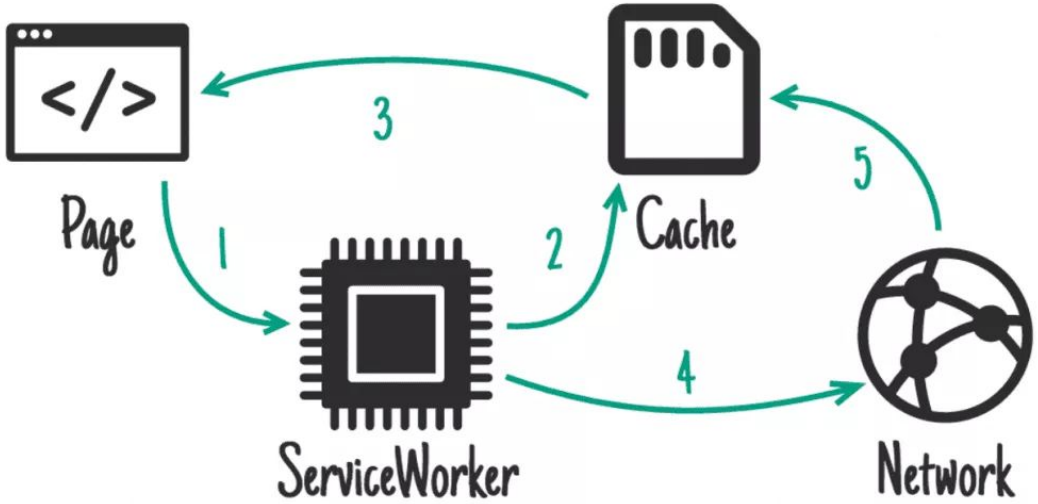


Cache Storage API

Accessible only in the Service Worker.

Can store “Response” objects or files.

Typically used for static files.



References

- https://web.dev/progressive-web-apps/
- https://web.dev/service-worker-lifecycle/
- https://www.smashingmagazine.com/2021/06/web-workers-2021/
- https://developer.mozilla.org/en-US/docs/Web/Web_Components
- https://web.dev/custom-elements-v1/
- https://hacks.mozilla.org/2017/02/what-makes-webassembly-fast/
- https://web.dev/webrtc-basics/
- https://graphql.org/
- https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
- https://developer.mozilla.org/en-US/docs/Web/API/CacheStorage