FUN WITH SERVICE WORKERS

@TRENTMWILLIS

{CODEMOTION}

ATWOOD'S LAW

*"ANY APPLICATION THAT **CAN** BE WRITTEN IN JAVASCRIPT,*
***WILL EVENTUALLY** BE WRITTEN IN JAVASCRIPT."*




Winamp2-js
Secure
https://webamp.org
WINAMP
02:26
1. CAKE - OPERA SINGER
128 kbps 44 kHz
mono stereo
EQ PL
SHUFFLE
WINAMP EQUALIZER
ON AUTO
PRESETS
+12 db
+0 db
-12 db
PREAMP 60 170 310 600 1K 3K 6K 12K 14K 16K
WINAMP PLAYLIST
1. Cake - opera singer 4:06
2. Cake - meanwhile, rick james 3:57
3. Cake - shadow stabbing 3:07
4. Cake - short skirt long jacket 3:24
5. Cake - commisioning a symphony 2:59
6. Cake - arco arena 1:31
7. Cake - comfort eagle 3:40
8. Cake - long line of cars 3:23
9. Cake - love you madly 3:57
10. Cake - pretty pink ribbon 3:08
11. Cake - world of two 3:40
ADD REM SEL MISC
4:06/36:56
02:26
LIST OPTS


PWA

FUN WITH
SERVICE WORKERS

@TRENTMWILLIS

LOVE THE WEB
LOVE JAVASCRIPT

JAVASCRIPT CONTINUES TO EVOLVE

WEB HYPERTEXT APPLICATION TECHNOLOGY WORKING GROUP

WHATWG

STREAMS SPEC

*"IF INSTALLED INSIDE THE FETCH HOOK OF A SERVICE WORKER, THIS WOULD ALLOW DEVELOPERS TO **TRANSPARENTLY POLYFILL** NEW IMAGE FORMATS."*

POLYFILLING FILE FORMATS

New Image Format

New Image Format

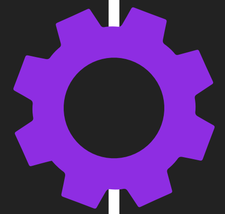

```
<img src="./my-image.nif">
```

New Image Format


```
<img src="./my-image.nif">
```

**Can Display PNG/JPEG/GIF, Not NIF*

New Image Format



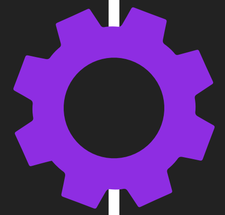
Service Worker

```
<img src="./my-image.nif">
```

**Can Display PNG/JPEG/GIF, Not NIF*

New Image Format

.nif



Service Worker

.gif

```
<img src="./my-image.nif">
```

*Can Display PNG/JPEG/GIF, Not NIF

ALL CODE WILL BE AVAILABLE ONLINE

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

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

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

  event.respondWith();

});
```

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

  event.respondWith(validFileFromPolyfilledFile(event.request));

});
```

```
self.addEventListener('fetch', (event) => {
  if (isPolyfilledFile(event.request)) {
    event.respondWith(validFileFromPolyfilledFile(event.request));
  }
});
```

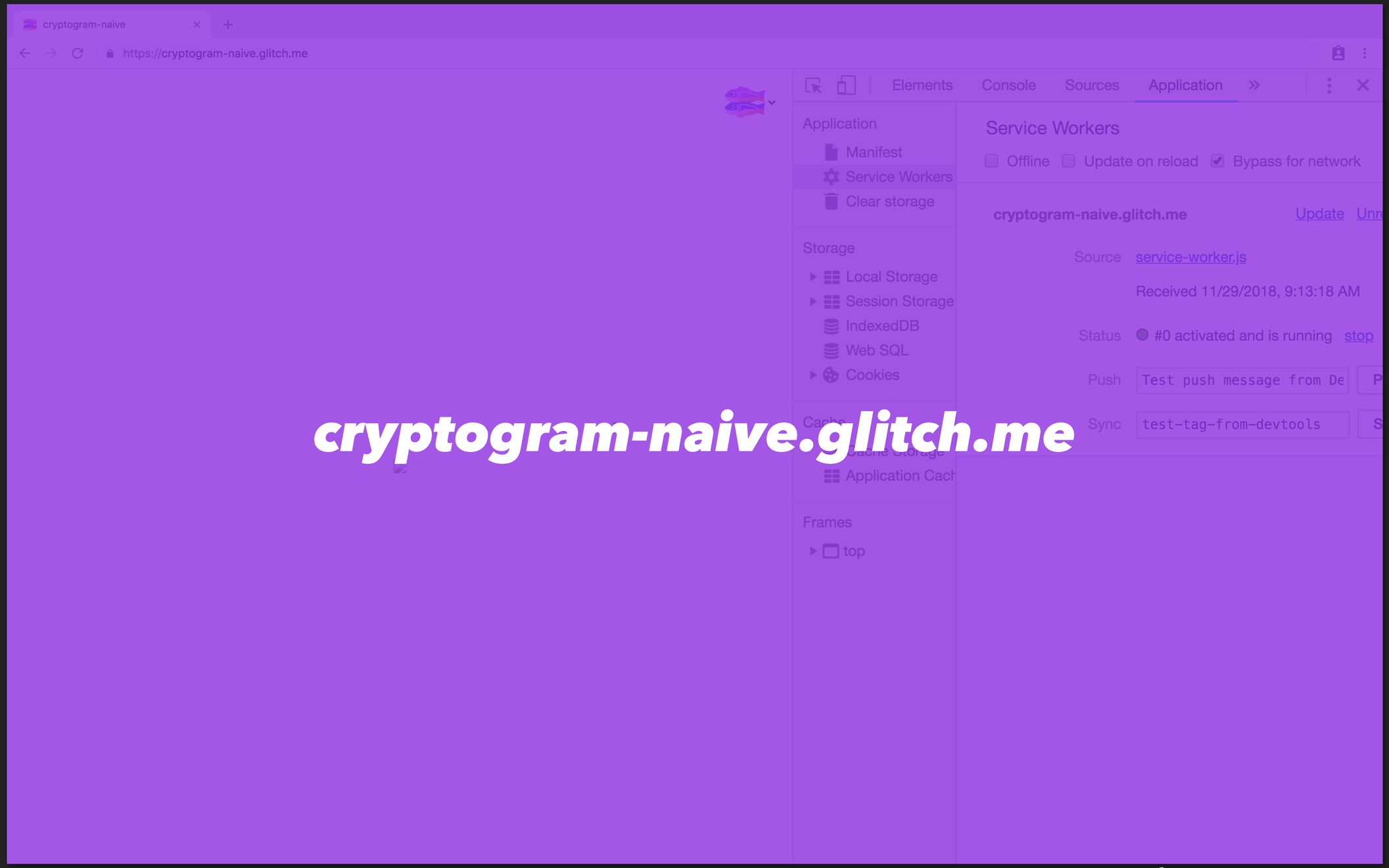
```
const validFileFromPolyfilledFile = async (request) => {
};
```

```
const validFileFromPolyfilledFile = async (request) => {

  const originalResponse = await fetch(request);

};
```

```
const validFileFromPolyfilledFile = async (request) => {

  const originalResponse = await fetch(request);
  const originalData = originalResponse.blob(); // .text(), .json(), etc.

};
```

```
const validFileFromPolyfilledFile = async (request) => {

  const originalResponse = await fetch(request);
  const originalData = originalResponse.blob(); // .text(), .json(), etc.
  const modifiedData = polyfillTransform(originalData);

};
```

```
const validFileFromPolyfilledFile = async (request) => {

  const originalResponse = await fetch(request);
  const originalData = originalResponse.blob(); // .text(), .json(), etc.
  const modifiedData = polyfillTransform(originalData);
  const modifiedResponse = new Response(modifiedData);
  return modifiedResponse;

};
```



cryptogram-naive.glitch.me


hello-world.html – html-modul
https://glitch.com/edit/#!/html-modules-polyfill?path=hello-world.html:3:0
html-modules-polyfill
Show
Live
Sign in
Share
Join
Remix to Edit
1 <h1>Hello, world!</h1>
2 Coming at you live, from an HTML Template imported as an ES Module!
3
assets
README.md
app.js
boot.js
hello-world.html
hello-world.js
index.html
service-worker.js
html-modules-polyfill.glitch.me

COMPILING CODE AT RUNTIME

TYPESCRIPT AT RUNTIME

```
<script src="my-module.ts"></script>
```

```
<script src="my-module.ts" type="module"></script>
```

CACHE API

ONLY COMPILE CHANGED FILES

```
const compileWithCache = async (response, compile) => {
};
```

```
const compileWithCache = async (response, compile) => {

  const cache = await caches.open(‘compile-cache');

};
```

```
const compileWithCache = async (response, compile) => {

  const cache = await caches.open(‘compile-cache');

  if (response.headers.get('status') === '304') {
  }

};
```

```
const compileWithCache = async (response, compile) => {
  const cache = await caches.open(‘compile-cache');
  if (response.headers.get('status') === '304') {
    return cache.match(request.url);
  }
};
```

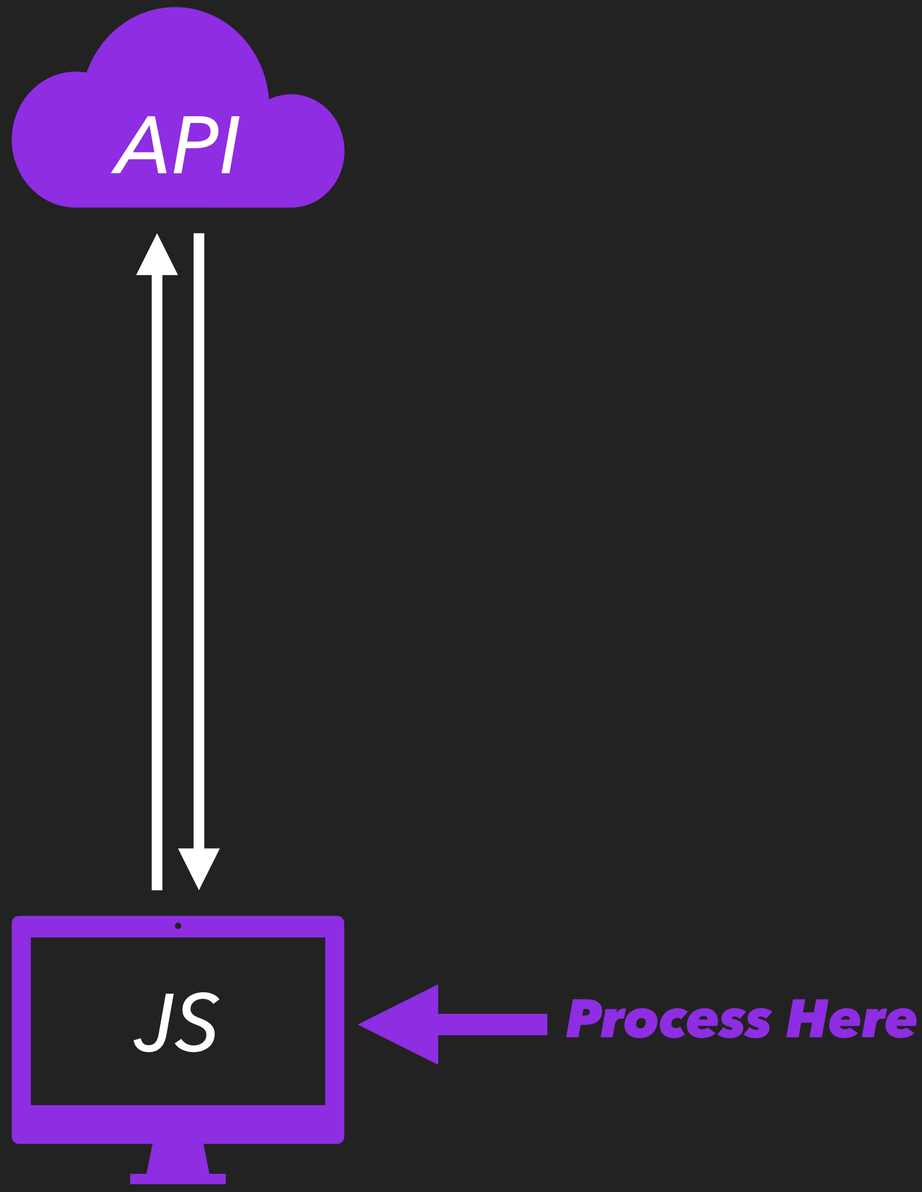
```
const compileWithCache = async (response, compile) => {
  const cache = await caches.open(‘compile-cache');
  if (response.headers.get('status') === '304') {
    return cache.match(request.url);
  }
  const compiledResponse = compile(response);
};
```

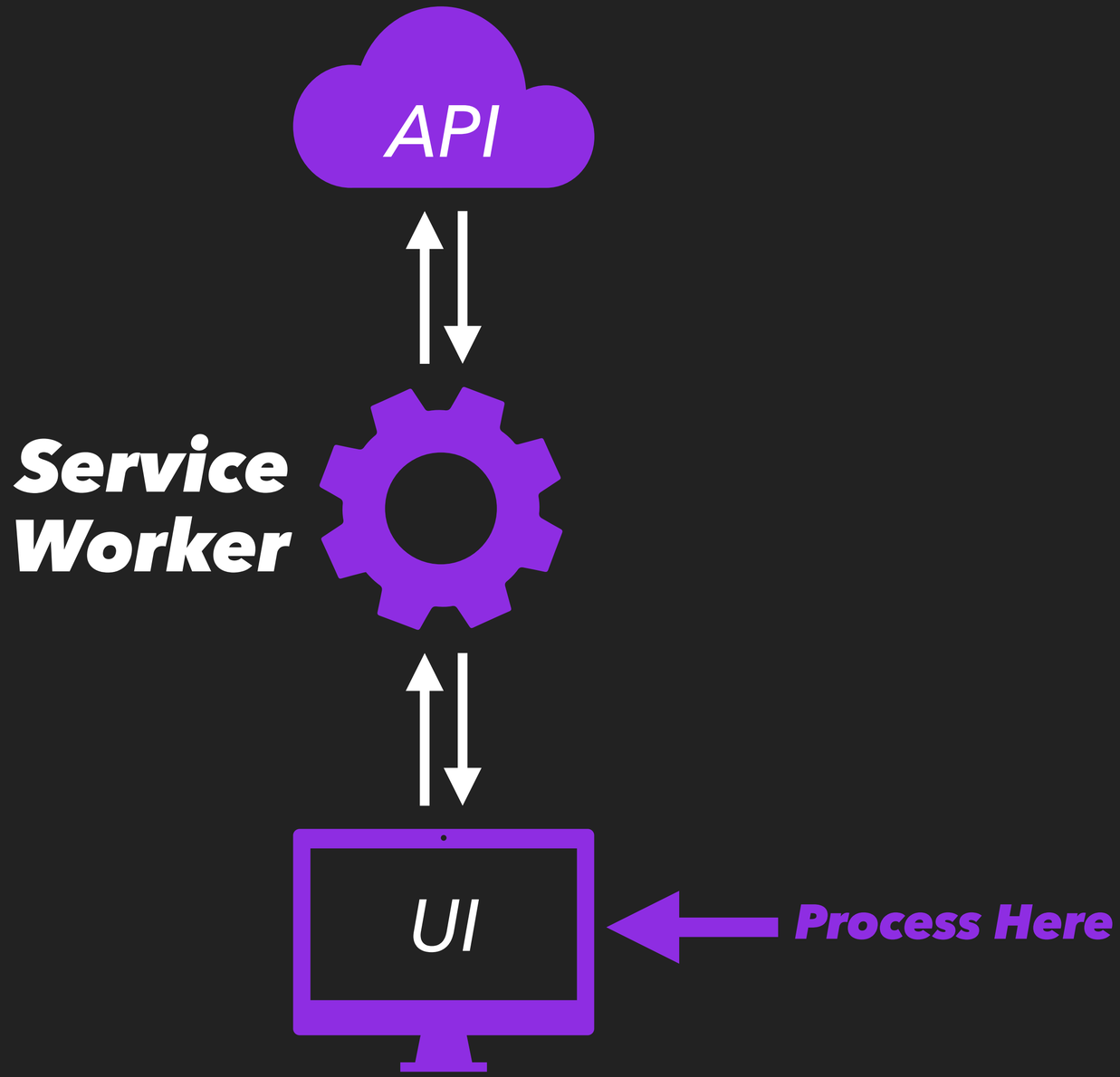
```
const compileWithCache = async (response, compile) => {
  const cache = await caches.open(‘compile-cache');
  if (response.headers.get('status') === '304') {
    return cache.match(request.url);
  }
  const compiledResponse = compile(response);
  cache.put(request.url, compiledResponse.clone());
};
```

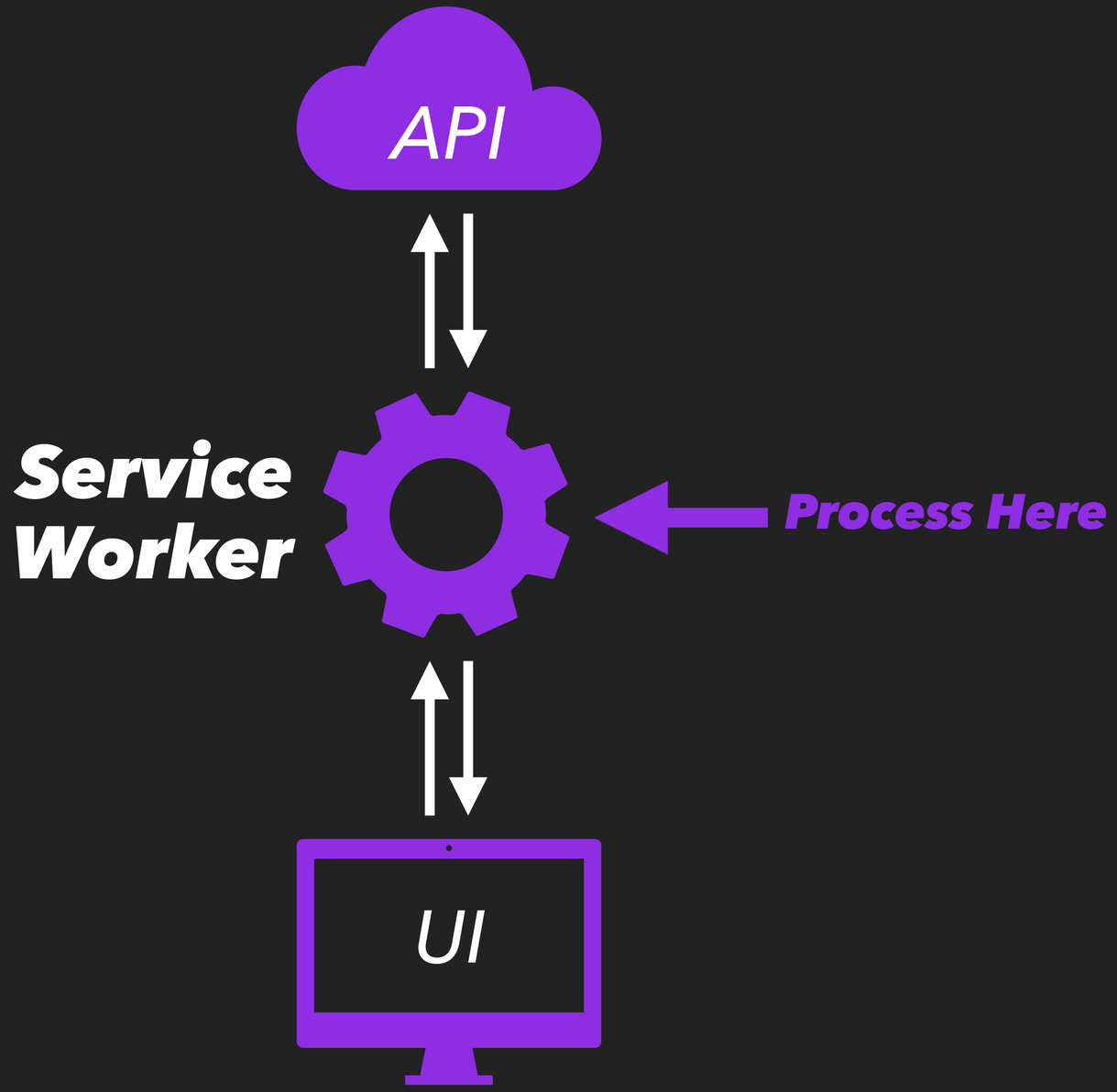
```
const compileWithCache = async (response, compile) => {
  const cache = await caches.open(‘compile-cache');
  if (response.headers.get('status') === '304') {
    return cache.match(request.url);
  }
  const compiledResponse = compile(response);
  cache.put(request.url, compiledResponse.clone());
  return compiledResponse;
};
```

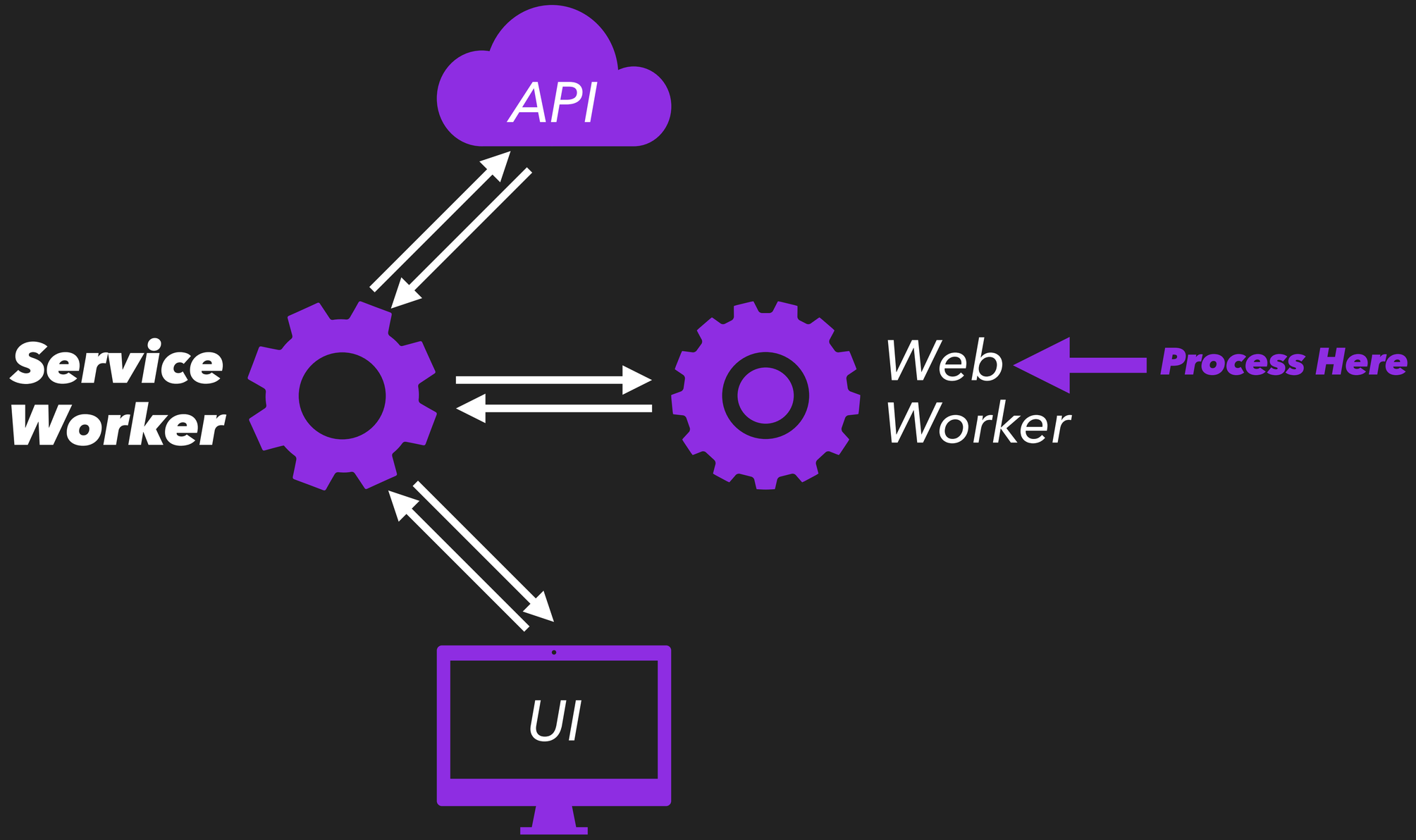

hello-world-template.ts – type
TypeScript In Browser Demo
https://glitch.com/edit/#!/typescript-in-browser?path=hello-world-template.ts:2:27
typescript-in-browser
Show
Live
Share
Status
+ New File
assets
README.md
app.js
hello-world-template.ts
hello-world.ts
index.html
service-worker.js
1 export default function template(): string {
2 return '<h1>Hello, world!</h1> This module is written in TypeScript with no build! ';
3 }
typescript-in-browser.glitch.me
New

OFF THE MAIN THREAD DATA PROCESSING


API
JS
Process Here


API
Service Worker
UI
Process Here


API
Service Worker
Process Here
UI


API
Service Worker
Web Worker
Process Here
UI

```
const transform = async (request, clientId) => {
  const [response, port] = await Promise.all([
    fetch(request),
    getPortToWebWorkerFromClient(clientId)
  ]);
};
```

```
const getPortToWebWorkerFromClient = async (clientId) => {
  const client = await self.clients.get(clientId);
};
```

```
const getPortToWebWorkerFromClient = async (clientId) => {
  const client = await self.clients.get(clientId);
  return new Promise(resolve => {
  });
};
```

```
const getPortToWebWorkerFromClient = async (clientId) => {
  const client = await self.clients.get(clientId);
  return new Promise(resolve => {
    client.postMessage({});
  });
};
```

```
const webWorker = new Worker('web-worker.js');

navigator.serviceWorker.onmessage = () => {
};
```

```
const webWorker = new Worker('web-worker.js');

navigator.serviceWorker.onmessage = () => {

  const channel = new MessageChannel();

};
```

```
const webWorker = new Worker('web-worker.js');

navigator.serviceWorker.onmessage = () => {

  const channel = new MessageChannel();
  const serviceWorker = navigator.serviceWorker.controller;
  serviceWorker.postMessage({port: channel.port1}, [channel.port1]);

};
```

```
const webWorker = new Worker('web-worker.js');

navigator.serviceWorker.onmessage = () => {

  const channel = new MessageChannel();
  const serviceWorker = navigator.serviceWorker.controller;
  serviceWorker.postMessage({port: channel.port1}, [channel.port1]);
  webWorker.postMessage({port: channel.port2}, [channel.port2]);

};
```

```
const getPortToWebWorkerFromClient = async (clientId) => {
  const client = await self.clients.get(clientId);
  return new Promise(resolve => {
    client.postMessage({});
  });
};
```

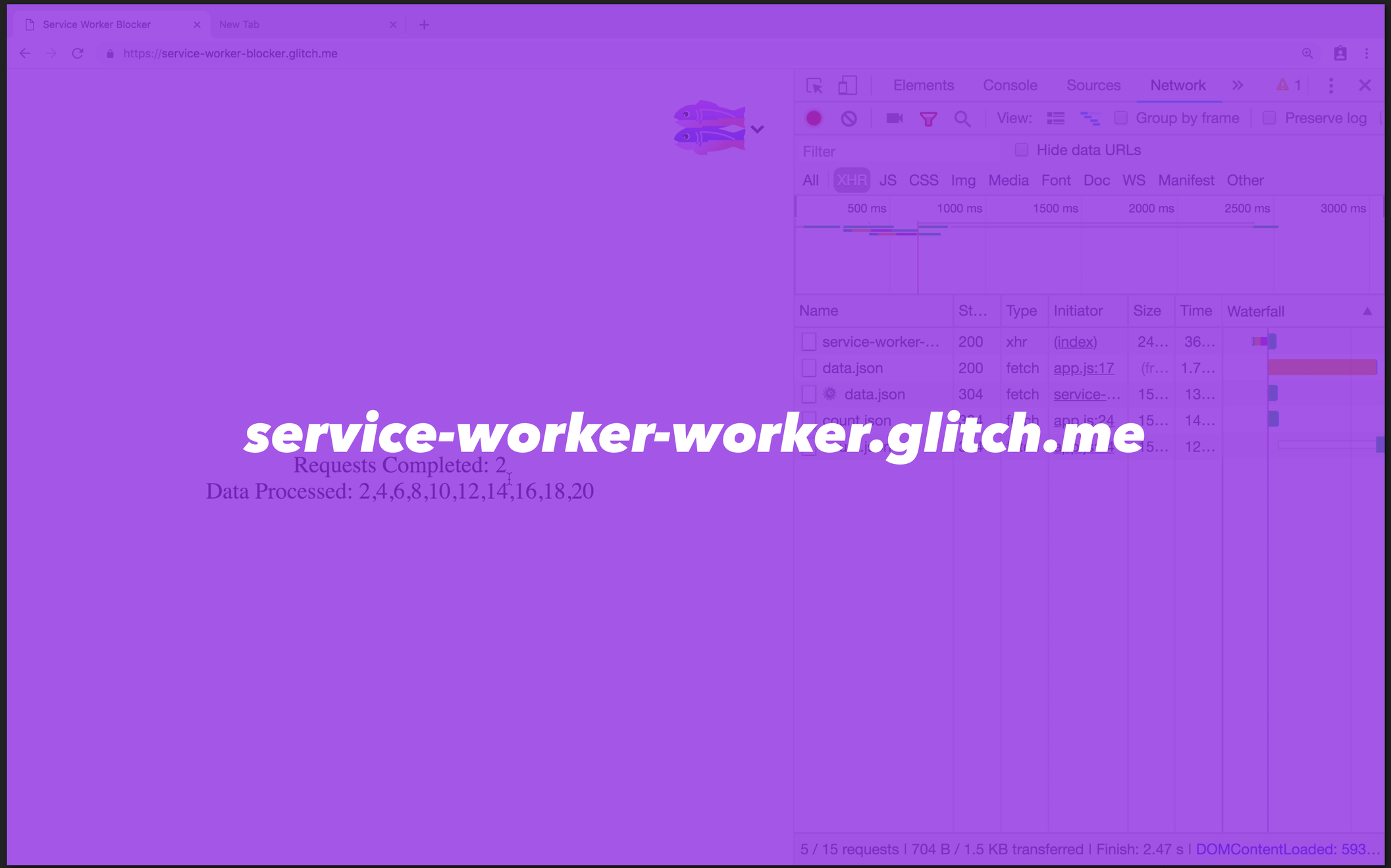
```
const getPortToWebWorkerFromClient = async (clientId) => {
  const client = await self.clients.get(clientId);
  return new Promise(resolve => {
    self.onmessage = (msg) => {
      self.onmessage = null;
      resolve(msg.data.port);
    };

    client.postMessage({});
  });
};
```

```
const transform = async (request, clientId) => {
  const [response, port] = await Promise.all([
    fetch(request),
    getPortToWebWorkerFromClient(clientId)
  ]);
};
```

```
const transform = async (request, clientId) => {
  const [response, port] = await Promise.all([
    fetch(request),
    getPortToWebWorkerFromClient(clientId)
  ]);

  return new Promise(async (resolve) => {
  });
};
```

```
const transform = async (request, clientId) => {
  const [response, port] = await Promise.all([
    fetch(request),
    getPortToWebWorkerFromClient(clientId)
  ]);

  return new Promise(async (resolve) => {
    port.postMessage(await response.json());
  });
};
```

```
const transform = async (request, clientId) => {
  const [response, port] = await Promise.all([
    fetch(request),
    getPortToWebWorkerFromClient(clientId)
  ]);

  return new Promise(async (resolve) => {
    port.onmessage = (msg) => {
      resolve(new Response(`[${msg.data.toString()}]`));
    };
    port.postMessage(await response.json());
  });
};
```

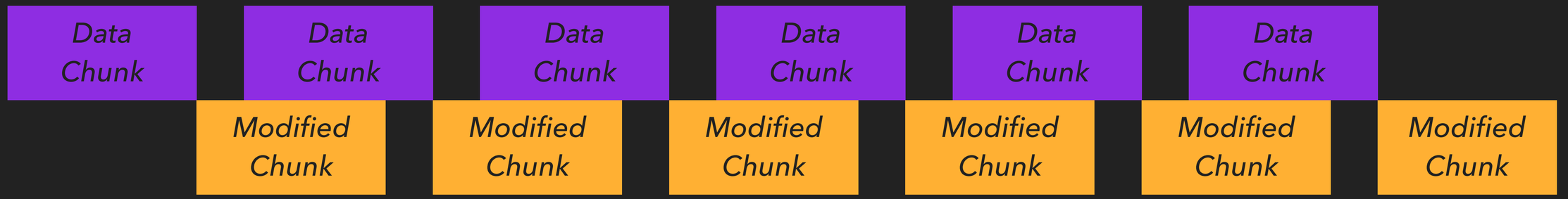

Service Worker Blocker
New Tab
https://service-worker-blocker.glitch.me
Elements
Console
Sources
Network
View:
Group by frame
Preserve log
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
Name
St…
Type
Initiator
Size
Time
Waterfall
service-worker-…
200
xhr
(index)
24…
36…
data.json
200
fetch
app.js:17
(fr…
1.7…
data.json
304
fetch
service-…
15…
13…
count.json
app.js:24
15…
14…
15…
12…
5 / 15 requests | 704 B / 1.5 KB transferred | Finish: 2.47 s | DOMContentLoaded: 593…
service-worker-worker.glitch.me
Requests Completed: 2
Data Processed: 2,4,6,8,10,12,14,16,18,20

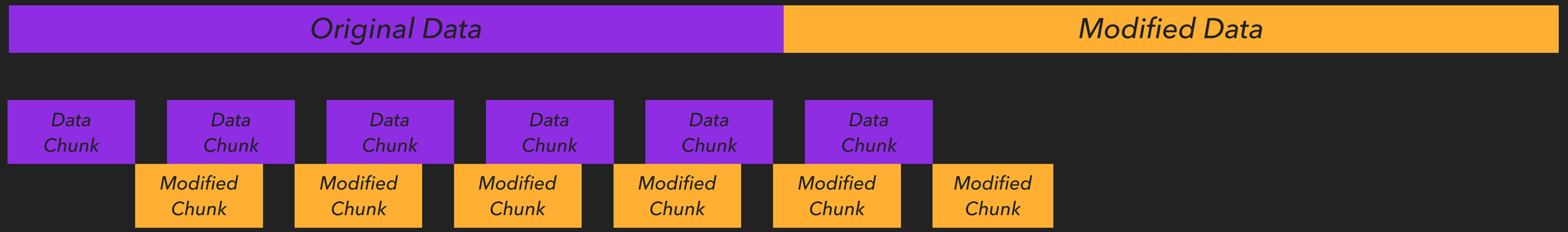
STREAM DATA PROCESSING


Original Data
Modified Data

BATCH PROCESSING

STREAM PROCESSING


Data Chunk
Data Chunk
Data Chunk
Data Chunk
Data Chunk
Data Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk

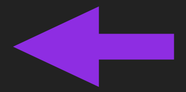

Original Data
Modified Data
Data Chunk
Data Chunk
Data Chunk
Data Chunk
Data Chunk
Data Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk
Modified Chunk

```
const validFileFromPolyfilledFile = async (request) => {

  const originalResponse = await fetch(request);
  const originalData = originalResponse.blob(); // .text(), .json(), etc.
  const modifiedData = polyfillTransform(originalData);
  const modifiedResponse = new Response(modifiedData);
  return modifiedResponse;

};
```

```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```

```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```

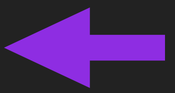
```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```



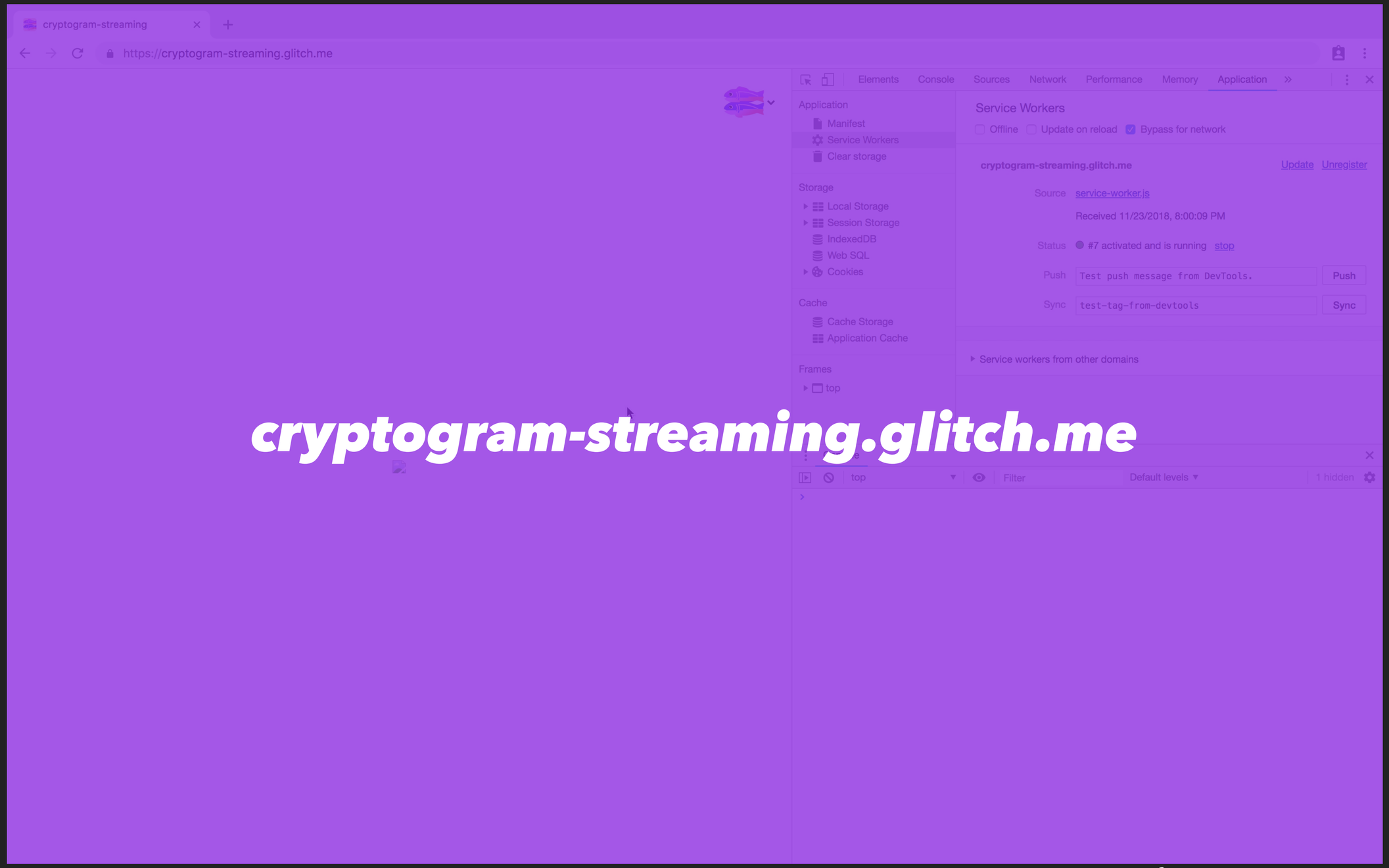
```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```

```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```

```
const validFileFromPolyfilledFile = async (request) => {;

  const originalResponse = await fetch(request);
  const transformStream = new TransformStream(new Transformer());
  const transformedBody = originalResponse.body.pipeThrough(
    transformStream
  );
  const transformedResponse = new Response(transformedBody);
  return transformedResponse;

};
```



```
class Transformer {
  async start() {}

  async transform() {}

  async flush() {}
}
```



cryptogram-streaming.glitch.me

POLYFILLING FILE FORMATS

COMPILING CODE AT RUNTIME

OFF THE MAIN THREAD DATA PROCESSING

STREAM DATA PROCESSING

SERVICE WORKERS

STREAMS

WEB ASSEMBLY

ESNEXT

WEB COMPONENTS

AND MORE!

EXPERIMENT & HAVE FUN WITH JAVASCRIPT!

*ALSO, GLITCH IS GREAT FOR EXPERIMENTS!

QUESTIONS?

@TRENTMWILLIS {codemotion}