CREATING YOUR OWN IMAGE FORMAT IN THE BROWSER

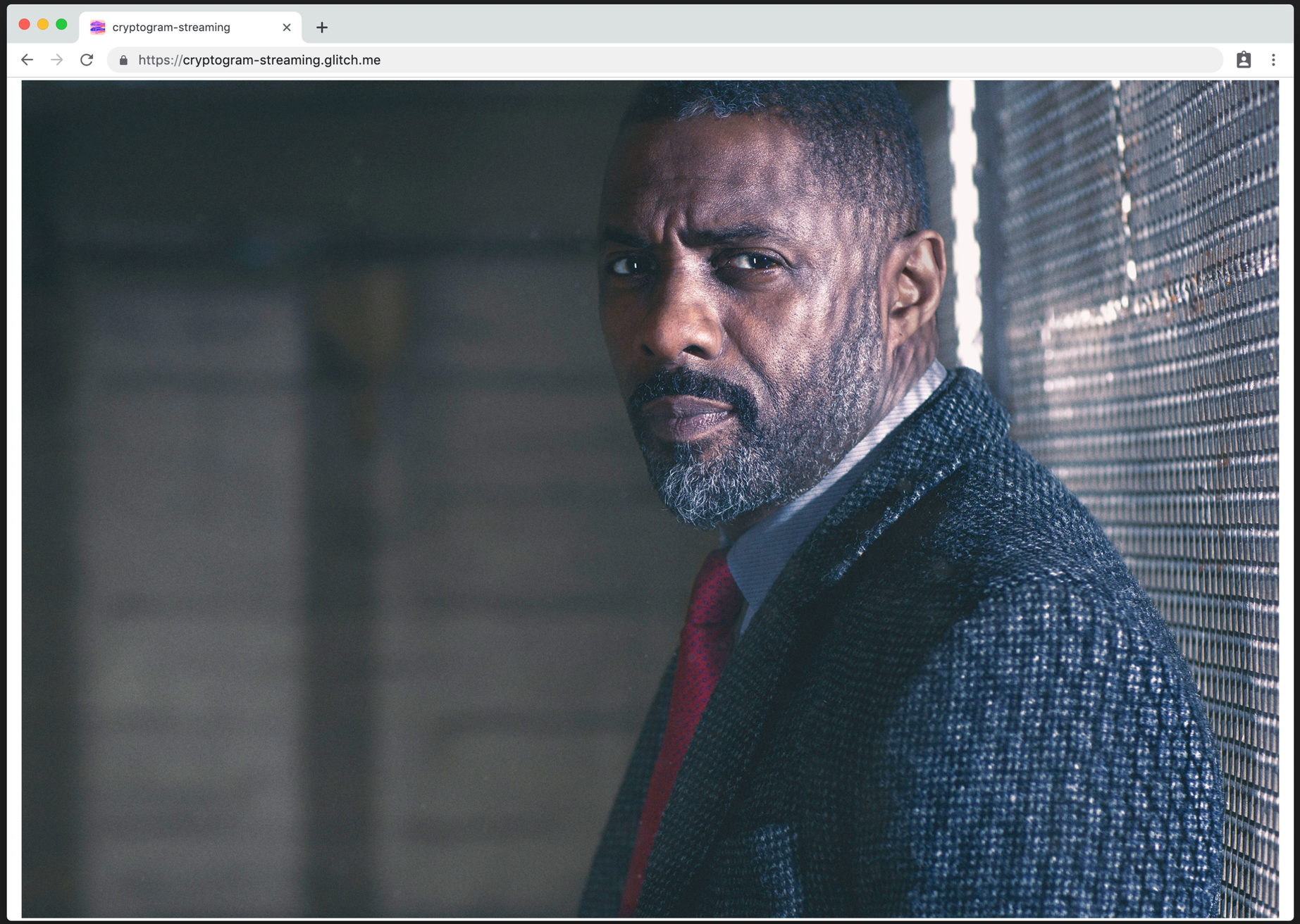
ATWOOD'S LAW

*"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


DoppioJVM: A JVM in JavaScri
https://plasma-umass.org/doppio-demo/
Fork me on GitHub
DoppioJVM
A Java Virtual Machine written in 100% JavaScript.
Read the academic paper »
Browse the code »
Shell Demo
Ruby.js
Not Secure
ruby-in-the-browser.my-codeworks.com
Ruby.js
Fork me on GitHub
Ruby in the browser
This is a collection of demos of what you can currently do in the area of using ruby as the scripting language of the browser. As a replacement, kinda, of javascript if you will.
This is an area where a lot of people have done a lot of work over the years


cryptogram-streaming
https://cryptogram-streaming.glitch.me

CREATING YOUR OWN IMAGE FORMAT IN THE BROWSER

@TRENTMWILLIS

LOVE THE WEB
LOVE JAVASCRIPT

JAVASCRIPT CONTINUES TO EVOLVE

WEB HYPERTEXT APPLICATION TECHNOLOGY WORKING GROUP

WHATWG

WHATWG STREAMS SPEC

STREAMS SPEC

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


LISTEN, I SHIT YOU NOT
THAT BRIDGE IN LONDON IS FALLING DOWN
quickmeme.com

PHOTOS ARE IMPORTANT

PHOTOS CAN BE EXPERIMENTED WITH

WEB CRYPTO API


I HAVE AN IDEA.
A GOOD ONE?
LET`S NOT GET AHEAD OF OURSELVES.
poorlydrawnlines.com

CRYPTO + INSTAGRAM

CRYPTOGRAM

Encrypted Image

Encrypted Image

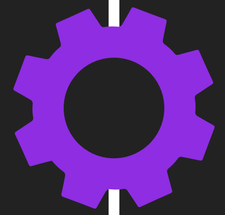

```
<img src="./encrypted-image.epng">
```

Encrypted Image

```
<img src="./encrypted-image.epng">
```

**Can Display PNG/JPEG, Not EPNG*

Encrypted Image



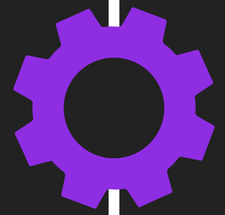
Service Worker

```
<img src="./encrypted-image.epng">
```

**Can Display PNG/JPEG, Not EPNG*

Encrypted Image

.epng



Service Worker

.png

```
<img src="./encrypted-image.epng">
```

**Can Display PNG/JPEG, Not EPNG*

ALL CODE WILL BE AVAILABLE ONLINE

FETCHEVENT

FETCHEVENT
.RESPONDWITH()

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

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

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

  if (isEncryptedImageRequest(event.request)) {
  }

});
```

```
const isEncryptedImageRequest = request => request.url.endsWith('.epng');
```

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

  if (isEncryptedImageRequest(event.request)) {
  }

});
```

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

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

```
const pngFromEncryptedImageRequest = async (request) => {
  const response = await fetch(request);
};
```

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();

};
```

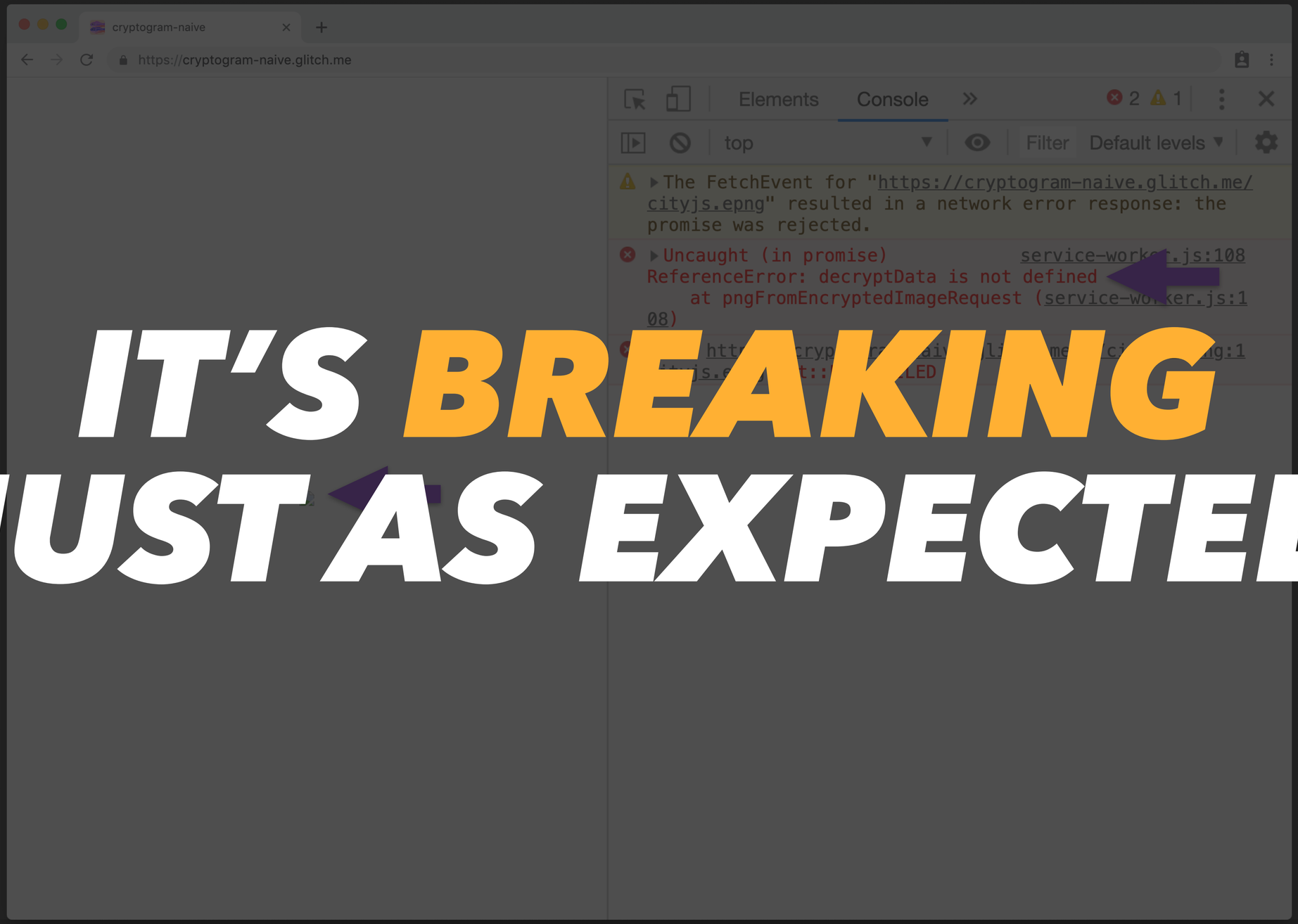
```
const pngFromEncryptedImageRequest = async (request) => {

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);

};
```

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

};
```

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

  return new Response(pngBlob);

};
```

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

  return new Response(pngBlob);

};
```

```
const pngFromCustomImageRequest = async (request, transform) => {

  const response = await fetch(request);
  const rawData = await response.arrayBuffer();
  const pngData = await transform(rawData);
  const pngBlob = new Blob([pngData], { type: 'image/png' });

  return new Response(pngBlob);

};
```



IT'S BREAKING JUST AS EXPECTED

WEB CRYPTO API

SUBTLECRYPTO

SUBTLECRYPTO

*"IT IS NAMED **SUBTLECRYPTO** TO REFLECT THE FACT THAT MANY OF THESE ALGORITHMS HAVE SUBTLE USAGE REQUIREMENTS IN ORDER TO PROVIDE THE REQUIRED ALGORITHMIC SECURITY GUARANTEES."*

CRYPTOKEY

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

  return new Response(pngBlob);

};
```

```
const decryptData = async (encryptedData) => {
};
```

```
const decryptData = async (encryptedData) => {

  const cryptoKey = await getCryptoKey();

};
```

```
const decryptData = async (encryptedData) => {

  const cryptoKey = await getCryptoKey();
  const decryptionOptions = {
    name: ‘AES-CTR',
    counter: new Uint8Array(16),
    length: 128
  };

};
```

```
const decryptData = async (encryptedData) => {

  const cryptoKey = await getCryptoKey();
  const decryptionOptions = {
    name: 'AES-CTR',
    counter: new Uint8Array(16),
    length: 128
  };
  return crypto.subtle.decrypt(
    decryptionOptions,
    cryptoKey,
    encryptedData
  );

};
```

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

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

  return new Response(pngBlob);

};
```

CRYPTOGRAM-NAIVE.GLITCH.ME

MAKE IT WORK ✓
MAKE IT RIGHT ✓
MAKE IT FAST ✗

Encrypted Data


Encrypted Data
Decrypted Data

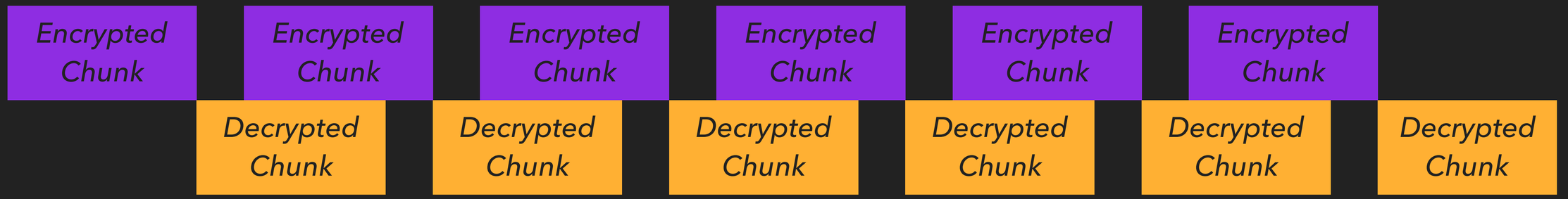
BATCH PROCESSING

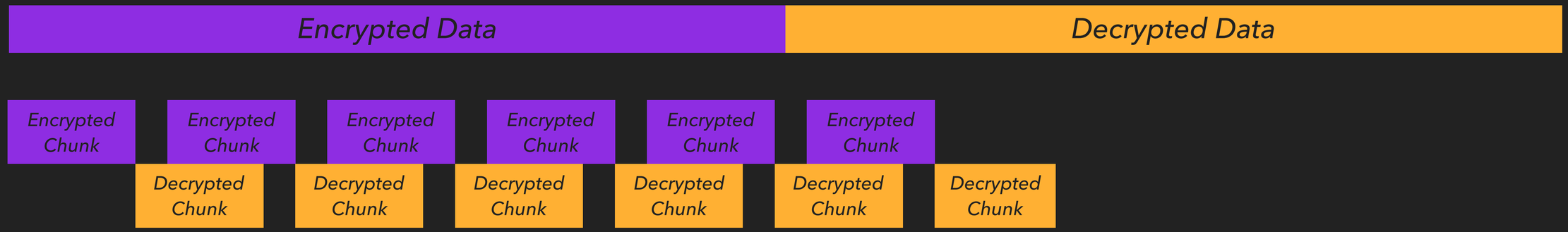
STREAM PROCESSING

Encrypted Chunk


Encrypted
Chunk
D


Encrypted Chunk
Encrypte Chunk
Decrypted Chunk

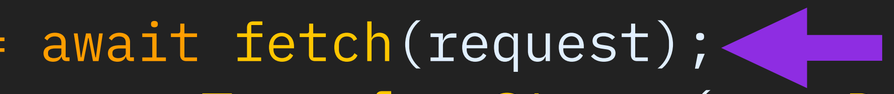

Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk


Encrypted Data
Decrypted Data
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Encrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk
Decrypted Chunk

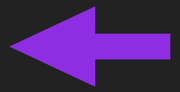
```
const pngFromEncryptedImageRequest = async (request) => {

  const response = await fetch(request);
  const encryptedData = await response.arrayBuffer();
  const decryptedData = await decryptData(encryptedData);
  const pngBlob = new Blob([decryptedData], { type: 'image/png' });

  return new Response(pngBlob);

};
```

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

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```

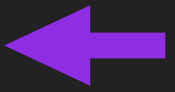
```
const pngFromEncryptedImageRequest = async (request) => {

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```



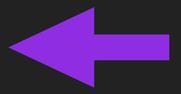
```
const pngFromEncryptedImageRequest = async (request) => {

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```



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

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```



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

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```



WRITABLESTREAM

WRITABLESTREAM
READABLESTREAM

WRITABLESTREAM
+
READABLESTREAM
=
TRANSFORMSTREAM

TRANSFORMER

```
TRANSFORMER
  .START()
  .TRANSFORM()
  .FLUSH()
```

```
TRANSFORMER
  .START()
  .TRANSFORM()
  .FLUSH()
```

```
TRANSFORMER
  .START()
  .TRANSFORM()
  .FLUSH()
```

```
TRANSFORMER
  .START()
  .TRANSFORM()
  .FLUSH()
```

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

  const response = await fetch(request);
  const decrypter = new TransformStream(new Decrypter());
  const pngStream = response.body.pipeThrough(decrypter);

  return new Response(pngStream);

};
```

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

  async transform() {
  }

  async flush() {
  }
}
```

```
class Decrypter {
  async start() {
    this.counter = new Uint8Array(16);
  }

  async transform() {
  }

  async flush() {
  }
}
```

```
class Decrypter {
  async start() {
    this.counter = new Uint8Array(16);
    this.key = await getCryptoKey();
  }

  async transform() {
  }

  async flush() {
  }
}
```

```
async transform(chunk, controller) {
  let data = chunk;

  const length = data.length;
  const blocks = Math.floor(length / 16);
}
```

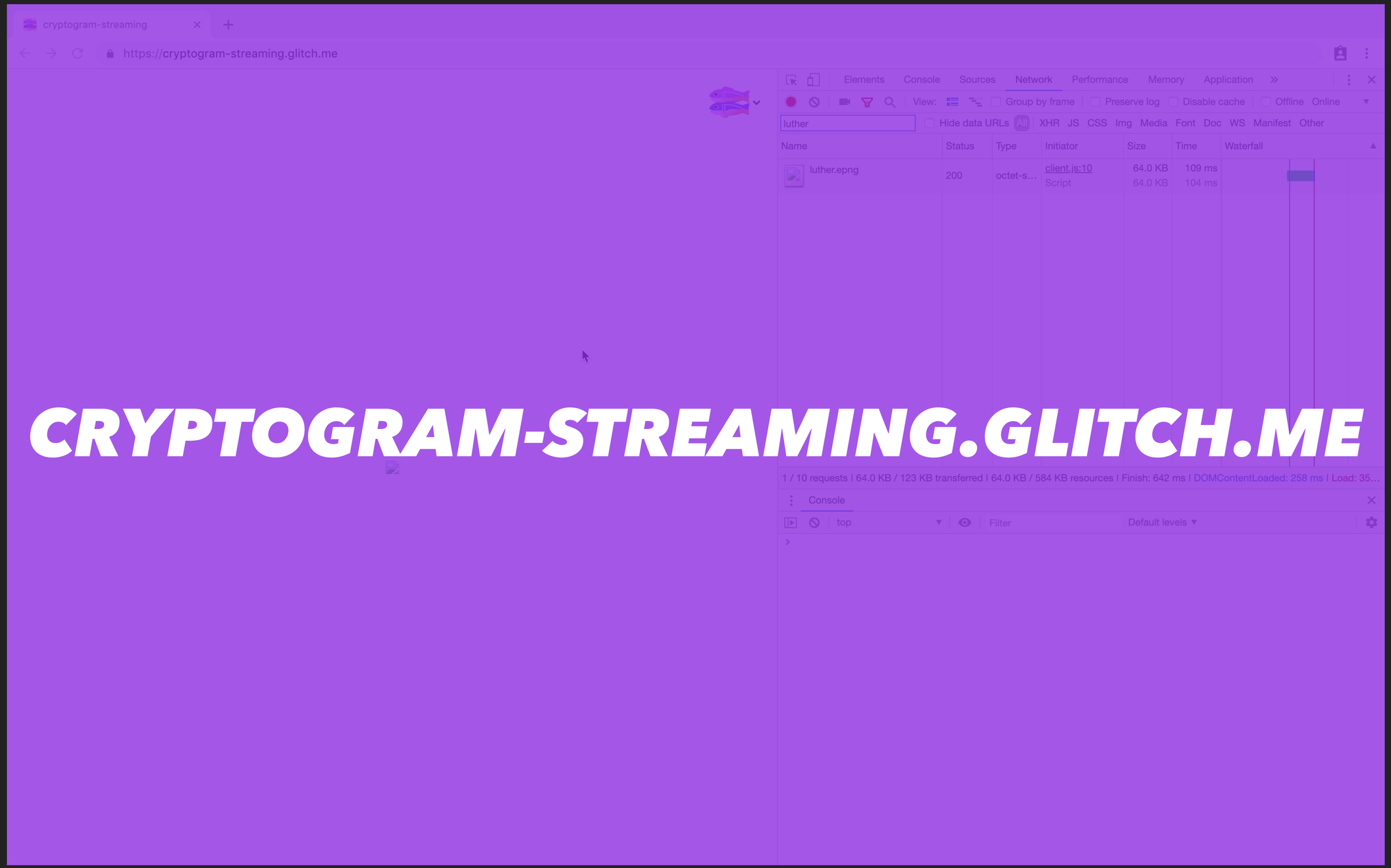
```
async transform(chunk, controller) {
  let data = chunk;

  const length = data.length;
  const blocks = Math.floor(length / 16);

  const decryptedData = await decrypt(this.key, data, this.counter);
  controller.enqueue(decryptedData);

  for (let i = 0; i < blocks; i++) {
    incrementUint8ArrayCounter(this.counter);
  }
}
```

```
async transform(chunk, controller) {
  let data = chunk;

  const length = data.length;
  const blocks = Math.floor(length / 16);
  const remainder = length % 16;

  if (remainder) {
    this.remainderData = data.slice(length - remainder);
    data = data.slice(0, length - remainder);
  }

  const decryptedData = await decrypt(this.key, data, this.counter);
  controller.enqueue(decryptedData);

  for (let i = 0; i < blocks; i++) {
    incrementUint8ArrayCounter(this.counter);
  }
}
```

```
async transform(chunk, controller) {
  let data = chunk;

  if (this.remainderData) {
    data = concatUint8Arrays(this.remainderData, data);
    this.remainderData = null;
  }

  const length = data.length;
  const blocks = Math.floor(length / 16);
  const remainder = length % 16;

  if (remainder) {
    this.remainderData = data.slice(length - remainder);
    data = data.slice(0, length - remainder);
  }

  const decryptedData = await decrypt(this.key, data, this.counter);
  controller.enqueue(decryptedData);

  for (let i = 0; i < blocks; i++) {
    incrementUint8ArrayCounter(this.counter);
  }
}
```

```
class Decrypter {
  async start() {
    this.counter = new Uint8Array(16);
    this.key = await getCryptoKey();
  }

  async transform() {
    // It's too long!
  }

  async flush() {
  }
}
```

```
class Decrypter {
  async start() {
    this.counter = new Uint8Array(16);
    this.key = await getCryptoKey();
  }

  async transform() {
    // It's too long!
  }

  async flush(controller) {
    if (this.remainderData) {
      const decryptedData = await decrypt(
        this.key, this.remainderData, this.counter);
      controller.enqueue(decryptedData);
    }
  }
}
```



CRYPTOGRAM-STREAMING.GLITCH.ME

WEBASSEMBLY

FOR LIGHTNING FAST TRANSFORM STREAMS

wasm-streams.glitch.me

RUNTIME
STREAMING
COMPILATION
FOR DEVELOPMENT

HTML MODULE POLYFILL

html-modules-polyfill.glitch.me

PACKAGE NAME MAPS POLYFILL

package-name-maps.glitch.me

RUNTIME
VIDEO EFFECTS

RUNTIME STREAMING COMPILATION FOR DEVELOPMENT

HTML MODULE POLYFILL

PACKAGE NAME MAPS POLYFILL

RUNTIME VIDEO EFFECTS

SERVICE WORKERS

STREAMS

WEB CRYPTO

WEB ASSEMBLY

ESNEXT

WEB COMPONENTS

AND MORE!

WRITE EVERYTHING*
IN JAVASCRIPT!

**WITHIN REASON, BUT DEFINITELY HAVE FUN!*

***ALSO, GLITCH IS GREAT FOR EXPERIMENTS!*