275 CDN locations

Cloudflare Workers

Workers can be deployed via the CLI tool called Wrangler. They run in the Cloudflare Worker Runtime based on V8 that supports JavaScript/Wasm.

> The core Workers runtime was recently open-sourced under the name Workerd (meaning you can self-host Workers).

A default worker looks like the following:

```
addEventListener('fetch', event => {

        event.respondWith(handleRequest(event.request));

});

async function handleRequest(request) {

        return new Response('Hello worker!', { status: 200 });

}
```

The `fetch` event is any incoming HTTP request (ex. `*.workers.dev` or a custom origin). The handlers can be written in a Service Worker (seen above) or Module Worker format. The `event`
type is always `fetch`. You can handle the request with:

- `respondWith` lets you send a custom response. In most cases you will want to send a response. If the worker is run via `*.workers.dev`, you must send a response

- `waitUntil` lets you asynchronously run a process (ex. logging or caching) without blocking the response.
- `passThroughOnException` allows you to not send back a runtime error response in the case of an unhandled exception.

The response is typically a `Response` object. This can be the result of a `fetch` event or a custom response object.

Cloudflare Pages Functions

You can automatically deploy Workers from a Cloudflare Pages site by placing Worker files in the `functions` directory (source). This is currently in beta but is coming out of beta any day now and billed the same as Workers.

Routing of Pages Functions allows for dynamic and catchall routes.

```
├── ...
├── functions
|   └── api
|       ├── [[path]].ts
|       ├── [username]
|       |   └── profile.ts
|       ├── time.ts
|       └── todos
|           ├── [[path]].ts
|           ├── [id].ts
|           └── index.ts
└── ...
```

Cloudflare recently announced that they added support for Next.js edge runtime for Cloudflare Pages.

Middleware

They also support middleware similar to Next.js (Vercel). Just create a `_middleware.js` file in the `functions` folder or in specific subdirectories of this folder for middleware that applies only to specific routes.

You can chain middleware handlers and even pass data between them.

Edge Data

Cloudflare has multiple offerings for storing data at the edge depending on your requirements:

1. **KV** - A simple key value store that replicates to the edge. Eventually consistent.
2. **Durable objects** - Object storage (the stored objects are an instance of a class in JavaScript). Objects can have storage attached and exist in one location at any one time, giving them strong transactional consistency. This is available only on paid accounts.
3. **R2** - This is S3 compatible object storage. Available free with a pay-as-you-go option if you exceed the free limits.
4. D1 - A SQLite-based SQL database with read-only replication to the edge. Currently only available in private beta.

Worth noting that you can also store things in cache using the cache API.

HTMLRewriter

A special class that lets you parse and modify the HTML response body (like DOM manipulation except the DOM doesn't exist). The HTML response is modified via Element or Document handlers and then the updated HTML is returned.

Element Handlers

These are called on an instance of a selector within the response HTML. The available selectors are a subset of the full CSS selectors. Element handler classes can implement three methods:

- `element` (an incoming "DOM" element)
- `comments`
- `text`

Elements returned by an `element` handler have a ton of methods that can be used to mutate the response.

Text is streamed in chunks, meaning you may not get the full contents of an element at once.

Document Handlers

Rather than a specific DOM element, the document handler is for the full document response. An instance of a document handler also has the `comments` and `text` methods but also:

- `doctype` to modify the doctype of the document
- `end` which is triggered when the document is finished. Useful for appending content to the end of a document.