Article

Loading the latest version of the Apple Pay JS SDK

Link to the most recent autoupdating version of the Apple Pay JS SDK or a version of your choice.

Overview

Load the most recent version of the Apple Pay JS SDK available whenever possible. With the latest version, you get the most recent features and bug fixes. Depending on what features your website uses, you may want to load a specific version.

Load the auto-updating version of the SDK

The recommended mechanism to load the Apple Pay JS SDK is through backward-compatible, autoupdate URLs, as shown in the following example. This approach downloads the latest version of the SDK with the most recent bug fixes and backward-compatible features. The example shows the auto-updating version of the Apple Pay JS SDK:

```
<head>
    <script crossorigin
      src="https://applepay.cdn-apple.com/jsapi/1.latest/apple-pay-sdk.js" crossorig
    </script>
</head>
```

Get specific versions of the SDK

Specific SDKs following semantic versioning are also available with each release. This means that you, as the developer, have full control over when to upgrade to a new version of the Apple Pay JS SDK. Each version follows the standard semantic versioning pattern MAJOR.MINOR.PATCH, which conveys the following information:

- MAJOR — Incompatible API changes
- MINOR — New functionality, but backward-compatible
- PATCH — Backward-compatible bug fixes

A backward-compatible new release that contains only bug fixes increases the patch version; a backward-compatible new release that contains new features or functionality increases the minor version. A new release that changes the API, so it's no longer backward-compatible, increases the major version.

Use integrity checks

If you're using a semantic version of the SDK, adding an integrity check is a best practice when you load libraries from a third-party source. Subresource Integrity (SRI) uses the integrity and cross-origin attributes to ensure that information hosted on third-party servers isn't tampered with. For more information on SRI, see srihash.org. The following example shows the script for loading Apple Pay JS SDK version `1.3.2`:

```
<head>
    <script crossorigin
        src="https://applepay.cdn-apple.com/jsapi/v1.3.2/apple-pay-sdk.js" integrity
    </script>
</head>
```

Important

Only use the integrity attribute for semantic versions of the SDK, not the autoupdate SDK (`1.latest`), because the contents of the autoupdate SDK are subject to change and break the integrity checks. For a record of changes to the SDK, see the Apple Pay JS change log.

Implement a Content Security Policy

To help mitigate possible security attacks like data injecting or cross-site scripting (XSS), consider implementing Content Security Policy (CSP) in your website. For example, the following code shows possible CSP directives in HTML to load the resource types for the `image-src`, `frame-src`, and `script-src`:

```
img-src https://applepay.cdn-apple.com;
frame-src https://applepay.cdn-apple.com;
script-src https://applepay.cdn-apple.com;
```

The following sample is a JavaScript example for loading Apple Pay JS SDK dynamically:

```javascript
// This promise resolves when Apple Pay JS downloads and evaluates.
const applePayJSLoadedPromise = new Promise(resolve => {
    const element = document.createElement("script");
    element.addEventListener("load", resolve, { once : true });
    element.src = "https://applepay.cdn-apple.com/jsapi/1.latest/apple-pay-sdk.js"
    element.crossOrigin = "anonymous";
    document.head.appendChild(element);
});
```