Web Share APIs

Sharing content from your web application takes two different APIs: Web Share make the content shareable and the Web Share Target API makes your app elegible to receive shared data.

Getting Started

We will first look at adding share capabilities to a web page.

> The basic shape of the API looks like this:

```
navigator.share({ title: title, url: url })
  .then(function() {
    console.log("Share success!");
  })
  .catch(function() {
    console.log("Share failure!");
  });
```

And here's the code of my fist attempt to add it to a webpage. This will make the links with a `.share` class use the operating system's share functionality.

```
  document.addEventListener('DOMContentLoaded', function () {
    var shareLinks = [].slice.call(document.querySelectorAll('.share'));
    shareLinks.forEach(function (link) {
      link.addEventListener('click', function (event) {
        if (typeof navigator.share !== 'undefined') {
          event.preventDefault();
          var pageTitle = document.querySelector('.post-title').textConter
          var url = window.location.href;

          navigator.share({
              title: pageTitle,
              url: url
            })
```

```
            .then(function () {
              console.log("Share success!");
            })
            .catch(function () {
              console.log("Share failure!");
            });
        } else {
          console.log("Share not supported, revert to something else");
        }
      });
    });
  })
```

The script does the following:

1. Wrap the code on a `DOMContentLoaded` event listener
2. Select all links with the class “share”
3. Add a `click` listener to each link
4. When a share link is clicked, check for the existence of `navigator.share`
5. If it does exist, stop the link from doing anything
6. Get the URL from `window.location.href`
7. Get the page title from the element with the class `post-title`
8. Share the title and url

This will work in all kinds of web content when using a supported browser in a supporting operating system.

When implementing in a CMS

When working with WordPress or any other CMS, there is an additional consideration

Instead of using `window.location.href` you should use the canonical url for the document, if one is defined. This will make for better SEO by sharing the authoritative URL for the page.

Note:

Web Share must be initiated by a user action. There is no way to activate it programmatically.

Web Share Target API

The other sided of sharing to and from your web application is the Web Share Target API, that allows your App to add itself as a handler for shared items.

When working with Chrome, an app must meet the following criteria to become a share target:

- It must meet Chrome's installability criteria
- The user must add your app to their home screen

If these conditions are met then you have to decide what do you want to users to share with your application.

This post will cover the simplest case where you just want to share basic information about the page you're visiting. For other, more indepth cases, check out the Receiving shared data with the Web Share Target API.

To share basic information with your app, add the following code to your web app manifest file:

```
"share_target": {
  "action": "/share-target/",
  "/share-target/""method": "GET",
  "params" "title""title""title""text",
    "ur"text""text": "text",
    "url": "url"
  }
}
```

Once your app has the data, it can process it and respond to it appropriately.

If the user selects your application, and your method is “GET” (the default), the browser opens a new window at the action URL. The browser then generates a query string using the URL-encoded values supplied in the manifest.

For example, if the sharing app provides title and text, the query string is `?title=hello&text=world`. To process this, use a `DOMContentLoaded` event listener in your foreground page and parse the query string:

```
window.addEventListener('DOMContentLoaded', () => {
  const parsedUrl = new URL(window.location);
  console.log('Title shared: ' + parsedUrl.searchParams.get('title'));
  console.log('Text shared: ' + parsedUrl.searchParams.get('text'));
  console.log('URL shared: ' + parsedUrl.searchParams.get('url'));
});
```

What you do with the data you got from the shared item depends on your app. For example, you could use it to create a new document to edit or you could store it in an IndexedDB database.

To handle POST requests, check the Web.dev article referenced earlier.

Links, Specs and Resources

- Web Share API
  - Integrate with the OS sharing UI with the Web Share API — web.dev
  - Web Share API Explainer
  - The Web Share API — Phil Nash
  - How to Use the Web Share API to Trigger the Native Dialog to Share Content & Pull Quotes — Chris Love
  - Web Share API brings the native sharing capabilities to the browser — Serg Hospodarets
- Web Share Target API
  - Receiving shared data with the Web Share Target API — web.dev
  - Web Share Target API Explainer