Save Credentials from Forms

By Eiji Kitamura
Developer Advocate in Tokyo

By Meggin Kearney
Meggin is a Tech Writer

0:00

Save Credentials from sign-in forms

Keep your registration and sign-in forms as simple as possible.

Save credentials from sign-in forms so users won't have to sign in again when they return.

To store user credentials from forms:

1. Include `autocomplete` in the form.
2. Interrupt the form submission event.
3. Authenticate by sending a request.
4. Store the credential.
5. Update the UI or proceed to the personalized page.

Include `autocomplete` in the form

Before moving forward, check if your form includes `autocomplete` attributes. This helps the Credential Management API find the `id` and `password` from the form and construct a credential object.

This also helps browsers not supporting the Credential Management API to understand its semantics. Learn more about autofill in this article by Jason Grigsby.

```
<form id="signup" method="post">
 <input name="email" type="text" autocomplete="username email">
 <input name="display-name" type="text" autocomplete="name">
 <input name="password" type="password" autocomplete="new-password">
 <input type="submit" value="Sign Up!">
</form>
```

Interrupt the form submission event

Interrupt the form submission event when the user presses the submit button, and prevent the default behavior.

```
var f = document.querySelector('#signup');
f.addEventListener('submit', e => {
 e.preventDefault();
```

By preventing a page transition, you can retain the credential information while verifying its authenticity.

Authenticate by sending a request

To authenticate the user, deliver credential information to your server using AJAX.

On the server side, create an endpoint (or simply alter an existing endpoint) that responds with HTTP code 200 or 401, so that it’s clear to the browser whether the sign-up/sign-in/change password is successful or not.

For example:

```
// Try sign-in with AJAX
fetch('/signin', {
  method: 'POST',
  body: new FormData(e.target),
  credentials: 'include'
})
```

Store the credential

To store a credential, first check if the API is available, then instantiate a `PasswordCredential` with the form element as an argument either synchronously or asynchronously. Call `navigator.credentials.store()`. If the API is not available, you can simply forward the profile information to the next step.

Synchronous example:

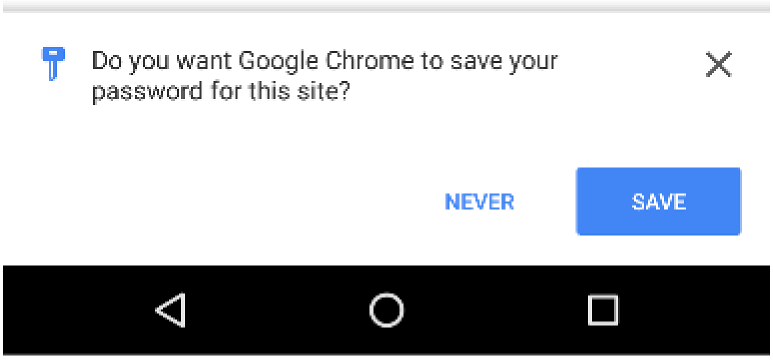
```
if (window.PasswordCredential) {
   var c = new PasswordCredential(e.target);
   return navigator.credentials.store(c);
 } else {
   return Promise.resolve(profile);
 }
```

Asynchronous example:

```
if (window.PasswordCredential) {
   var c = await navigator.credentials.create({password: e.target});
   return navigator.credentials.store(c);
 } else {
   return Promise.resolve(profile);
 }
```

Once the request succeeds, store the credential information. (Don't store the credentials information if the request failed as doing so confuses returning users.)

When the Chrome browser obtains credential information, a notification pops up asking to store a credential (or federation provider).



Do you want Google Chrome to save your password for this site?

Notification for an auto signed-in user

Update the UI

If everything went well, update the UI using the profile information, or proceed to the personalized page.

```
 }).then(profile => {
   if (profile) {
     updateUI(profile);
   }
 }).catch(error => {
   showError('Sign-in Failed');
 });
});
```

Full code example

```
// Get form's DOM object
var f = document.querySelector('#signup');
f.addEventListener('submit', e => {

  // Stop submitting form by itself
  e.preventDefault();

  // Try sign-in with AJAX
  fetch('/signin', {
    method: 'POST',
    body: new FormData(e.target),
    credentials: 'include'
  }).then(res => {
    if (res.status == 200) {
      return Promise.resolve();
    } else {
      return Promise.reject('Sign in failed');
    }
  }).then(profile => {

    // Instantiate PasswordCredential with the form
    if (window.PasswordCredential) {
      var c = new PasswordCredential(e.target);
      return navigator.credentials.store(c);
    } else {
      return Promise.resolve(profile);
    }
  }).then(profile => {

    // Successful sign in
    if (profile) {
      updateUI(profile);
    }
  }).catch(error => {

    // Sign in failed
    showError('Sign-in Failed');
  });
});
```

Last updated July 2, 2018.