Theme Developer Tutorial: 6. Using the JS API

- Common API methods
  - getCurrentUser()
  - headerIcons
  - replaceIcon()
  - decorateCookedElement()
  - registerValueTransformer()
- Finding more JS API methods

In the last couple of chapters, we've explored how to use the JavaScript API to render content into outlets. `renderInOutlet` is the most commonly-used API, but there are a ton more! In this chapter we'll try out a few of them, and show you how to discover more.

Common API methods

getCurrentUser()

`api.getCurrentUser()` will return information about the current user, or `null` if nobody is logged in. This can be used for all sorts of things, including per-group logic, or rendering a user's username into the UI.

For group membership checks backed by theme settings, use the `resolve_group_membership` option in your theme's `settings.yml` file instead of checking `currentUser.groups`. This avoids leaking or missing hidden group memberships.

```
import { apiInitializer } from "discourse/lib/api";


export default apiInitializer((api) => {
```

```
  console.log("Current user is", api.getCurrentUser());
});
```

headerIcons

`api.headerIcons` will allow you to add, remove and re-arrange icons in the header. For example, to add a new icon before the search icon, you'd do something like

```
import DButton from "discourse/components/d-button";
import { apiInitializer } from "discourse/lib/api";

export default apiInitializer((api) => {
  const sayHello = () => {
    alert("Hello, world!");
  };
  api.headerIcons.add(
    "my-unique-icon-name",
    <template>
      <li>
        <DButton
          @action={{sayHello}}
          @icon="wand-magic"
          class="icon btn-flat"
        />
      </li>
    </template>,
    { before: "search" }
  );
});
```

replaceIcon()

```
api.replaceIcon(source, destination);
```

With this method, you can easily replace any Discourse icon with another. For example, we have a theme component that replaces the heart icon for like with a thumbs-up icon

decorateCookedElement()

`api.decorateCookedElement()` allows you to customize the rendered content of Discourse posts. This can be used for anything from simple formatting changes, all the way up to advanced integrated UIs like the built-in 'poll' plugin.

The API should be passed a callback function which will be run for every post when it's rendered to the screen. The first argument to the callback will be the post's root HTML element, and the second will be a helper.

A simple example which appends content to every post would look like:

```
api.decorateCookedElement((element, helper) => {
  const myNewParagraph = document.createElement("p");
  myNewParagraph.textContent = "Hello, this is appended to every post!";
  element.appendChild(myNewParagraph);
});
```

Or for a more advanced UI, you can render a glimmer component into a post. For example, to render the counter component we authored earlier into every post, you could do something like this:

```
import { apiInitializer } from "discourse/lib/api";
import CustomWelcomeBanner from "../components/custom-welcome-banner";

export default apiInitializer((api) => {
  api.decorateCookedElement((element, helper) => {
    const counterWrapper = helper.renderGlimmer(
      "div.my-counter",
      CustomWelcomeBanner
    );
    element.appendChild(counterWrapper);
  });
});
```

`helper.getPost()` will return the current post, and can be used to build conditional logic into these `decorateCookedElement` callbacks. `console.log` the post to see what's available.

registerValueTransformer()

`api.registerValueTransformer` allows you to inject logic into predefined parts of the Discourse JavaScript application. For example, you can add a `"home-logo-href"` transformer to link the logo to `example.com`:

```
api.registerValueTransformer("home-logo-href", () => "https://example.com");
```

For more information on Transformers, check out the dedicated guide

Finding more JS API methods

All the available APIs are listed in the `plugin-api.gjs` source code in Discourse core, along with a short description and examples.

That's it for this chapter, and almost the end of the tutorial. Let's wrap things up in the conclusion.

□□□ #2
□ □□□ □□□ 2026-10-31 13:32:28 UTC
□ □□□ □□□ 2026-10-31 13:32:29 UTC