Hello Scout JS with a CDN

Version 11.0

Table of Contents

Looking for something else? Visit https://eclipsescout.github.io for all Scout related documentation.

Introduction

A typical approach to work with Scout JS is to use a package manager (e.g. pnpm) to download the modules and a bundler (e.g. webpack) to build them. Thanks to the Scout CLI this task is straightforward.

The main advantage is the simplified development process. You can use Less variables from Scout (e.g. the color palette). Individual source files can be imported directly which facilitates code completion. It also enables you to use modern code but still target older browsers by the usage of Babel.

If you don’t need all that and just want to include Scout as a script in your html page (as in the early days), you can do so, too! This article shows how to achieve this. A live demo of the app we’ll create is published on CodePen.

Preparation

First, you need to get the prebuilt Scout assets like scripts, stylesheets, fonts etc. These assets are part of the Scout npm modules and located in the `dist` folder.

To get them you can either install the Scout modules using a package manager and take the necessary resources from the dist folder. Or you could use a CDN and download them manually from there or even link to that CDN in your html files. In this example we are going to use a CDN directly.

There are several CDNs out there that serve the content of all npm modules. A popular one is jsDelivr. Using this CDN you can easily access all Scout assets: @eclipse-scout/core/dist.

Include Assets

Now let's create a new html file called `index.html` and paste the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Hello Scout CDN</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@eclipse-
scout/core@11.0.12/dist/eclipse-scout-core-theme.css" />①
</head>
<body>
  <div class="scout"></div>③
  <script src="https://code.jquery.com/jquery-3.5.1.js"></script>②
  <script src="https://cdn.jsdelivr.net/npm/sourcemapped-
stacktrace@1.1.11/dist/sourcemapped-stacktrace.js"></script>②
  <script src="https://cdn.jsdelivr.net/npm/@eclipse-scout/core@11.0.12/dist/eclipse-
scout-core.js"></script>①
  <script src="hello.js"></script>④
</body>
</html>
```

① As you can see, we include the Scout assets, namely `eclipse-scout-core-theme.css` and `eclipse-scout-core.js`.

② Additionally, we need to include all dependencies, which are `jquery` and `sourcemapped-stacktrace.js`. These are the dependencies listed in the package.json of @eclipse-scout/core.

③ Finally, we need to add an empty scout `<div>` where the html content generated by Scout will be placed.

④ The script `hello.js` contains the code of our application.

Add your Code

Now create a file called `hello.js` and paste the following code:

```
class Desktop extends scout.Desktop {

  constructor() {
    super();
  }

  _jsonModel() {
    return {
      objectType: 'Desktop',
      navigationHandleVisible: false,
      navigationVisible: false,
      headerVisible: false,
      views: [
        {
          objectType: 'Form',
          displayHint: 'view',
          modal: false,
          rootGroupBox: {
            objectType: 'GroupBox',
            borderDecoration: scout.GroupBox.BorderDecoration.EMPTY,
            fields: [
              {
                id: 'NameField',
                objectType: 'StringField',
                label: 'Name'
              },
              {
                id: 'GreetButton',
                objectType: 'Button',
                label: 'Say Hello',
                keyStroke: 'enter',
                processButton: false
              }
            ]
          }
        }
      ]
    };
  }

  _init(model) {
    super._init(model);
    this.widget('GreetButton').on('click', event => {
      let name = this.widget('NameField').value || 'stranger';
      scout.MessageBoxes.openOk(this.session.desktop, `Hello ${name}!`);
    });
```

```
  }
}

scout.addObjectFactories({
  'Desktop': () => new Desktop()
});

new scout.App().init({
  bootstrap: {
    textsUrl: 'https://unpkg.com/@eclipse-scout/core@11.0.12/dist/texts.json'
  }
});
```

As you can see, there are no imports at the top. Instead, we are using the global variable `scout`, that is automatically put on the `window` object, to reference Scout classes.

Furthermore, we have to include the `texts.json`. This file needs to be included to make sure the texts used by Scout can be resolved for the language the user is using. In this case it is necessary for the text `Ok` which is visible on the message box when you click the button.

If you like to present dates and numbers using the format the user is used to, you could also include the `locales.json`. We do not need it for now so it is not included.

The rest of the code looks pretty similar to regular Scout JS code but be aware that some newer JS features are used like `class` or the template syntax that may not be supported by every browser.

Summary

That’s it. This is all you need to do to use Scout in a plain html site without the need of build tools.

If you like you could adjust the example to use the dark theme by using `eclipse-scout-core-theme-dark.css` instead of `eclipse-scout-core-theme.css`.

Or you could try to add a Chart by including the `@eclipse-scout/chart` module. The procedure is the same: link to the `@eclipse-scout/chart` assets (script, stylesheet, texts) and include its dependencies, that are referenced by its `package.json`.

The result could look like this: Eclipse Scout Chart on CodePen.

Have fun!

Do you want to improve this document? Have a look at the sources on GitHub.