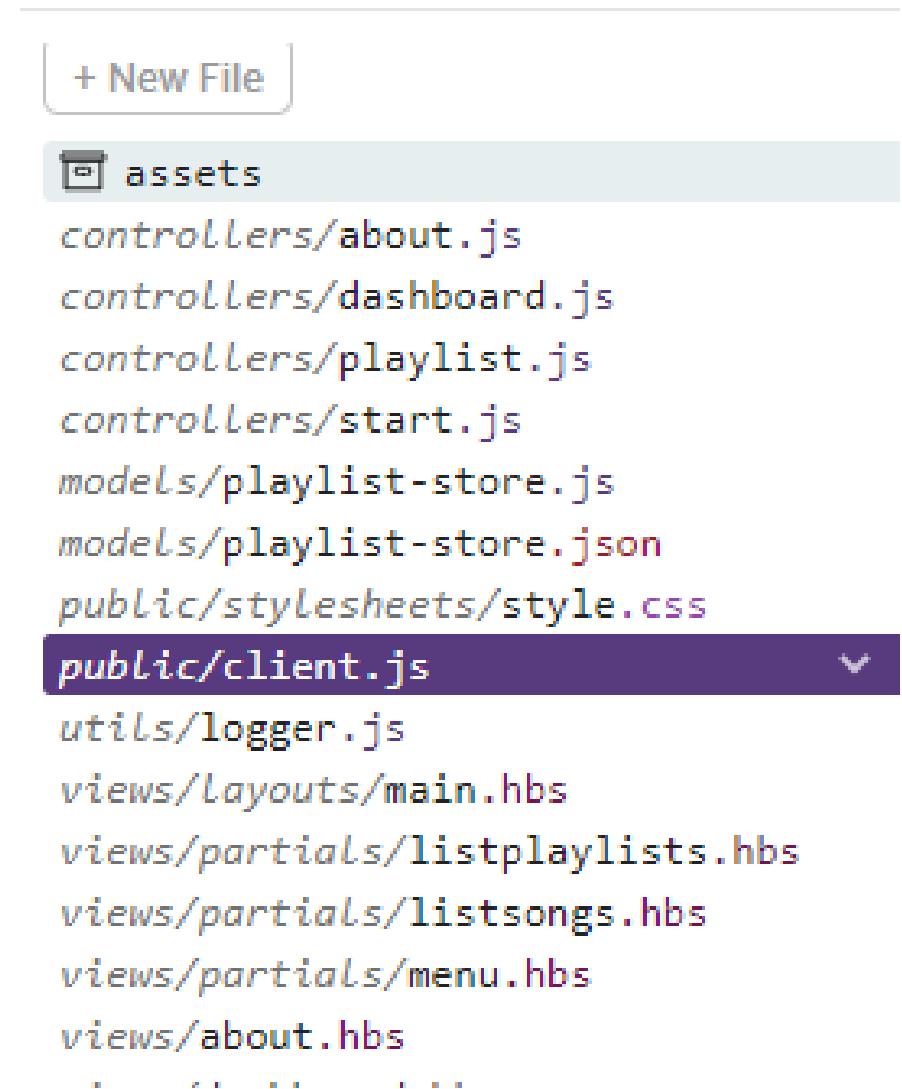


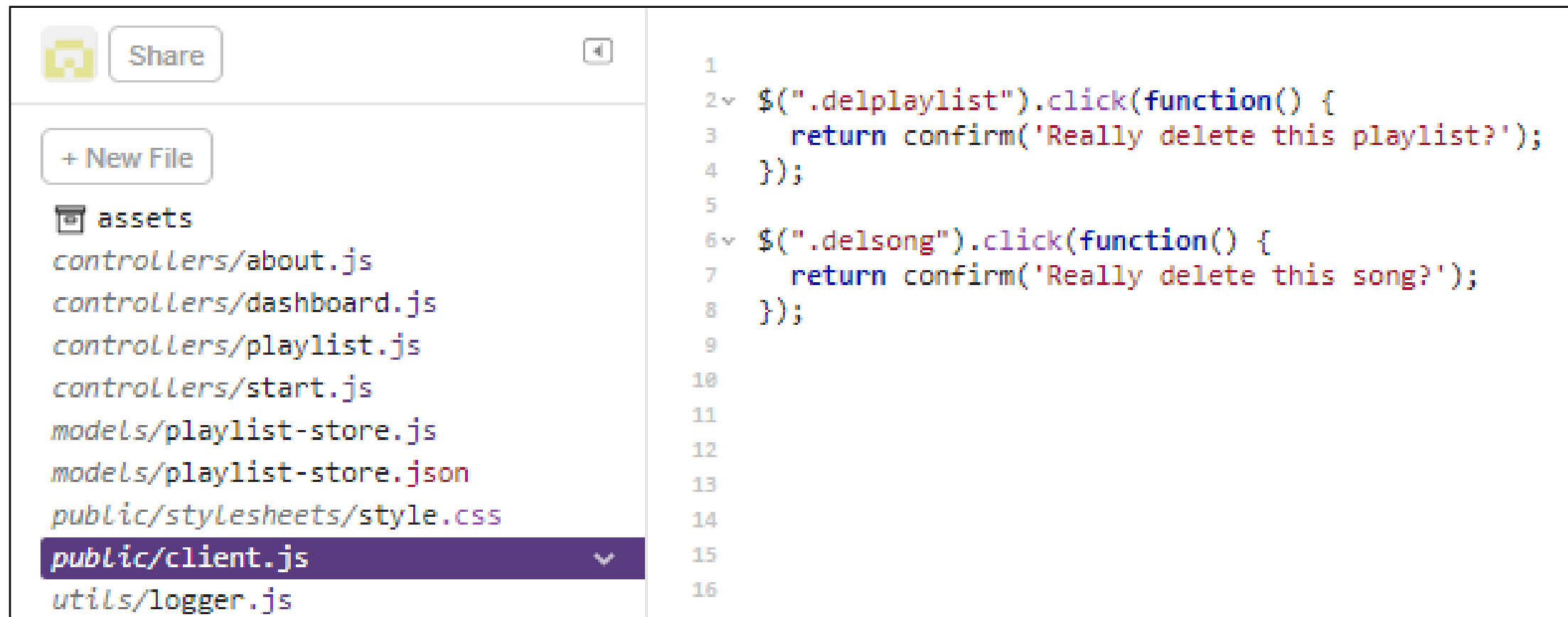
Adding client-side JavaScript

Add your own client-side JavaScript

- Create a file in the public/ directory
- Add your client-side JavaScript to this file



Add your own client-side JavaScript



The screenshot shows a code editor interface. On the left is a file explorer with a 'Share' button at the top. Below it is a '+ New File' button. The file list includes:

- assets
- controllers/about.js
- controllers/dashboard.js
- controllers/playlist.js
- controllers/start.js
- models/playlist-store.js
- models/playlist-store.json
- public/stylesheets/style.css
- public/client.js** (selected)
- utils/logger.js

On the right is the code editor for 'public/client.js', showing the following JavaScript code:

```
1
2 ✓ $(".delplaylist").click(function() {
3     return confirm('Really delete this playlist?');
4 });
5
6 ✓ $(".delsong").click(function() {
7     return confirm('Really delete this song?');
8 });
9
10
11
12
13
14
15
16
17
```

Add your own client-side JavaScript

- Link to /client.js in layouts/main.hbs file

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title> {{title}} </title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.css" type="text/css">
    <link rel="stylesheet" type="text/css" href="/stylesheets/style.css">
  </head>
  <body>
    <section class="ui container">
      {{{body}}}
    </section>
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/semantic-ui/2.4.1/semantic.min.js"></script>
    <script type="text/javascript" src="/client.js"></script>
  </body>
</html>
```

Add your own client-side JavaScript/jQuery

- This example:
 - Confirms that the user does want to delete the playlist/song

```
$(".delplaylist").click(function() {  
    return confirm('Really delete this playlist?');  
});  
  
$(".delsong").click(function() {  
    return confirm('Really delete this song?');  
});
```

JavaScript/jQuery references

- <https://www.w3schools.com/jsref/default.asp>
- <https://developer.mozilla.org/bm/docs/Web/JavaScript/Reference>
- https://www.w3schools.com/jquery/jquery_get_started.asp

Link to styled-playlist repository

- Import from GitHub: rbirney/styled-playlist

The screenshot shows the Glitch editor interface for a project named 'rb-styled-playlist'. The browser address bar displays the URL: `https://glitch.com/edit/#!/rb-styled-playlist?path=views/layouts/main.hbs:18:3`. The left sidebar shows a file explorer with a tree structure including `assets`, `controllers`, `models`, `public`, `utils`, and `views`. The main editor area displays the content of `views/layouts/main.hbs`, which is an HTML template with a head section containing meta tags and links, and a body section containing a script tag for `client.js`.

A modal dialog box titled 'glitch.com says' is open, providing instructions for importing from an existing GitHub repository. The dialog text reads: 'Import from an existing GitHub repository - please note that this will overwrite your current project. Enter the GitHub username, repository name, and optionally a relative path to a specific folder to import using the following format: username/repository:path.' Below the text is a text input field containing the text 'rbirney/styled-playlist'. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

In the bottom-left corner, the 'Git, Import, Export' menu is open, showing options for 'Read' and 'Write' access, a URL field with the text 'https://api.glitch.com/rb-s', and buttons for 'Import from GitHub' and 'Export to GitHub'.