Creating a Dynamic Web App with Google Apps Script: A Step-by-Step Guide

This guide will walk you through creating a basic web app that dynamically loads HTML content without refreshing the page, enhancing your site's interactivity and speed.

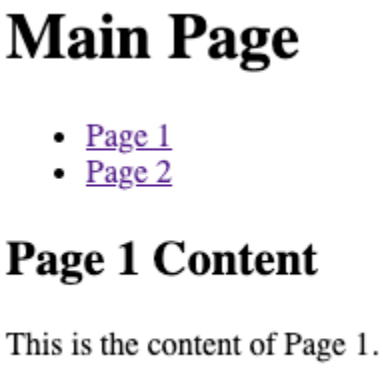

Image: Page load with links to load additional content pages

To update your Google Apps Script web app to dynamically load HTML content without reloading the page, and to incorporate JavaScript addEventListener for better handling of user interactions, follow the instructions below. This guide will show you how to modify your HTML to use event listeners and prevent default link behavior, ensuring a smoother user experience.

Step 1: Create or Update Your Google Apps Script Project

If you haven't already set up your Google Apps Script project, follow the initial steps to create one. If you have your project ready, proceed to the next steps.

Step 2: Update the Main Web App Script

Ensure your Code.gs file contains the necessary functions to serve your web app and to include HTML files dynamically. If you haven't done so, here's the script to use:

```
function doGet(e) {
 return HtmlService.createHtmlOutputFromFile('Index')
 .setTitle('Dynamic Web App')
 .setXFrameOptionsMode(HtmlService.XFrameOptionsMode.ALLOWALL);
```

```
}

function include(filename) {
 return HtmlService.createHtmlOutputFromFile(filename)
 .getContent();
}
```

Step 3: Update the Main HTML Page

Modify your main HTML page (Index.html) to use addEventListener for handling clicks on links. Replace its existing content with the following code:

```
<!DOCTYPE html>
<html>
  <head>
    <base target="_top">
  </head>
  <body>
    <h1>Main Page</h1>
    <nav>
      <ul>
        <li><a href="#" id="linkPage1">Page 1</a></li>
        <li><a href="#" id="linkPage2">Page 2</a></li>
      </ul>
    </nav>
    <div id="content">
      <!-- Additional content will be loaded here -->
    </div>
    <script>
      document.addEventListener('DOMContentLoaded', function() {
        document.getElementById('linkPage1').addEventListener('click', function(event) {
          event.preventDefault();
          loadPage('Page1');
        });
        document.getElementById('linkPage2').addEventListener('click', function(event) {
          event.preventDefault();
          loadPage('Page2');
        });
      });
      function loadPage(pageName) {
        google.script.run
```

```
        .withSuccessHandler(newPage)
        .include(pageName);
    }
    function newPage(html) {
      document.getElementById('content').innerHTML = html;
    }
  </script>
 </body>
</html>
```

Step 4: Create or Update Additional HTML Pages

If you haven't already created additional HTML pages (e.g., Page1.html and Page2.html), follow the previous instructions to create these files with the content you wish to load dynamically.

Step 5: Deploy Your Updated Web App

After updating your code, deploy your web app:

1. Click on Deploy -> New deployment or Manage deployments if updating.
2. Choose Web app and fill out the necessary fields.
3. Click Deploy and authorize any permissions if asked.
4. Use the provided URL to access your web app.

Your web app should now be capable of dynamically loading content when the "Page 1" and "Page 2" links are clicked, without reloading the entire page. This is achieved through the use of JavaScript's addEventListener and preventDefault methods, offering an improved user experience.

You've just created a dynamic web app using Google Apps Script! This simple example illustrates the power of Google Apps Script for building interactive web applications. By loading content dynamically, your app becomes more engaging and responsive, providing a better experience for your users. Explore further to discover how Google Apps Script can be used to create even more complex and feature-rich applications