

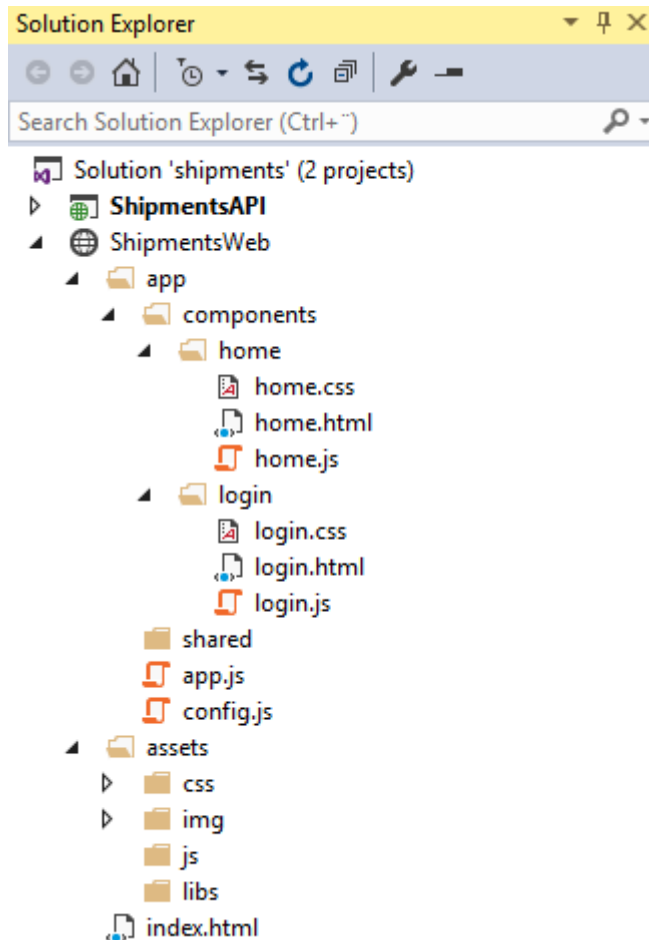
Building the Front-end SPA



Ajden Towfeek

@ajtowf | www.towfeek.se

Project Structure

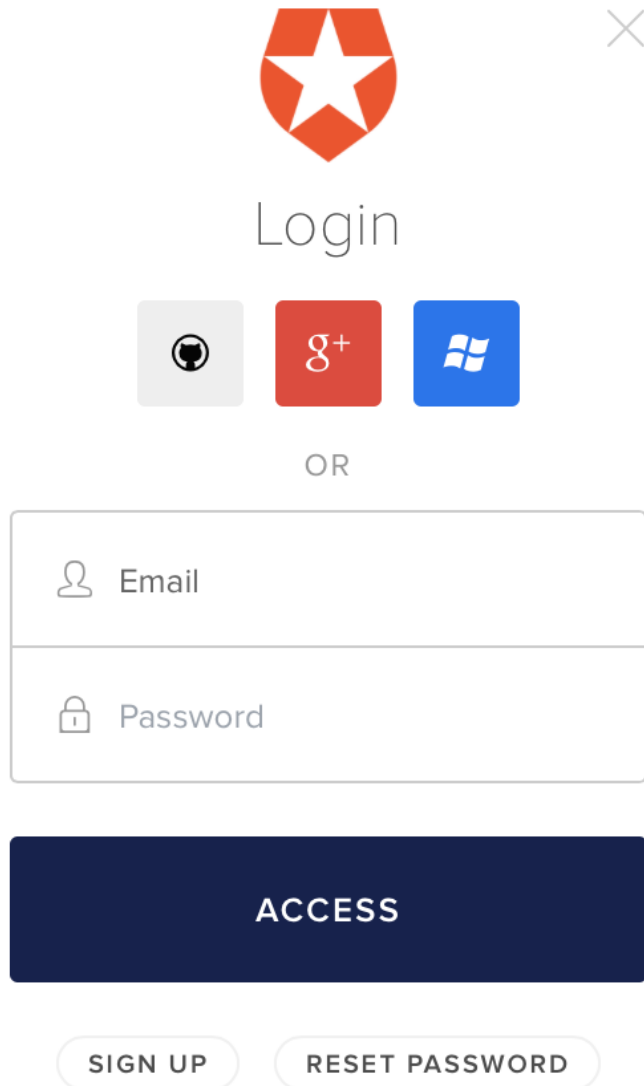


- index.html – Handles loading
- Assets Folder – Not angular related
- App Folder
 - app.js – Handles setup
 - config.js – Contains variables
 - Components – Home and Login
 - Shared

Demo

- Created the Web Application
- Bootstrap Angular
- Configure Routes









The image shows a login interface for Auth0 Lock. At the top is the Auth0 logo (a red shield with a white star) and a close button (an 'X'). Below the logo is the word 'Login'. There are three social login buttons: GitHub (grey), Google+ (red), and Windows (blue). Below these is the text 'OR'. There are two input fields: 'Email' with a person icon and 'Password' with a lock icon. Below the input fields is a large dark blue button labeled 'ACCESS'. At the bottom are two buttons: 'SIGN UP' and 'RESET PASSWORD'.


✕

Login

OR

 Email

 Password

ACCESS

SIGN UP RESET PASSWORD

Auth0 Lock

- User login & Signup
- Libraries
 - lock-7.js
 - auth0-angular-4.js
- Module dependencies
- Sign in popup
- Show user information

Demo

- Signing in using Auth0 Lock



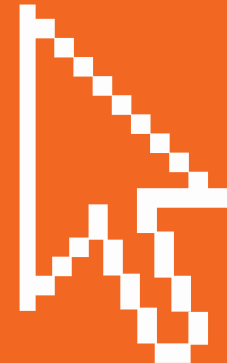
Storing Information

```
function Controller(auth, $location, store, $scope)
  $scope.login = function() {
    auth.signin({}, function (profile, id_token, access_token,
      store.set('profile', profile);
      store.set('token', id_token);
      $location.path('/');
    }, function () {
      // TODO Handle when login fails
    });
  }
}
```

- Include angular-storage dependency
- Save the user info after login
- Authenticate on page refresh

Demo

- Store information



Calling the API

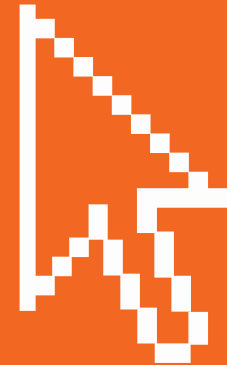
```
angular.module('myApp', ['auth0', 'angular-jwt'])
  .config(function($httpProvider, jwtInterceptorProvider) {
    jwtInterceptorProvider.tokenGetter = function(auth) {
      return auth.idToken;
      // or
      // return store.get('token');
    }

    $httpProvider.interceptors.push('jwtInterceptor');
  });
```

- Include angular-jwt dependency
- Configure the jwtInterceptor

Demo

- Call the API



Refresh Tokens

```
auth.signin({  
  authParams: {  
    scope: 'openid offline access',  
  }  
});
```

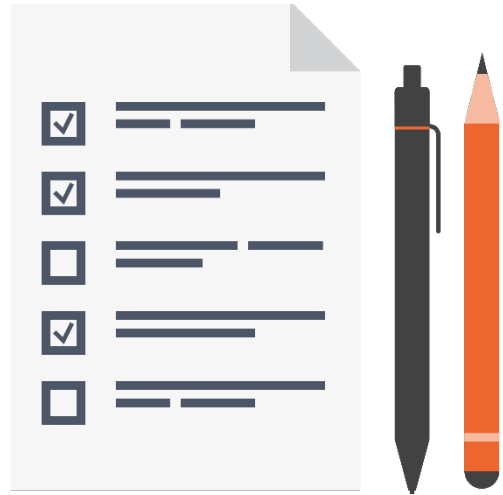
- Getting the refresh token
- Storing the refresh token
- Use refresh token to get a new JWT
- Use refresh token on page refresh

Demo

- Refresh Tokens



Summary



- Setup scalable Angular app structure
- Consumes Auth0 Angular libraries
- Called secured API endpoints
- Refresh Tokens