* Course Overview
* Intro
  + Json Web Tokens(JWT)
* Why Use an Auth Provider
  + Authentication: Who are you?
    - Login with email and password
  + Authorization: What are you allowed to do?
    - Check user rights
  + Auth0, Okta
    - OAuth provider
* OAuth 2.0
  + Auth0 is security provider
  + OAuth is security protocol
  + OAuth 2.0 is about authorization
  + Scope is permission
    - String they denotes type of access
    - Like read only, write, etc
  + OAuth
    - Authorization protocol
    - Authorize a user without their password
    - Get info from a third party
  + OAuth setup
    - Register with service(google, facebook, etc)
    - Give app name, website, callback url
    - Auth0 handles this for us
  + OAuth roles
    - Resource owner: user
    - Client: app that want to access user account
    - Auth Server: auth logic, provides access token
    - Resource Server: api application want to access, user data
* Choosing an OAuth Flow
  + Client ap sent auth request to user
  + User accepts request and sends back auth grant to app
  + App sends grant to auth server
  + Auth server sends app access token
  + App use access token to call api in resource server
  + Resource server returns data
  + Grant(aka flow)
    - A way to receive an access token
  + Implicit Flow
    - Your app directs the browser to the Auth0 sigin-in
    - Auth0 redirects your app, at the callback url you chose
    - Your app reads the tokens from the URL
  + OAuth is for authorization not authentication
  + Use OpenID Connect for authentication
* OpenID Connect
  + OAuth is for authorization
    - No standard for scopes and user info requests
  + Nice to avoid managing passwords ourselves
  + OpenID Connect
    - Authenticate users without managing passwords
    - Sits on top of OAuth2.0
    - OAuth2.0 sits on top of http
    - Adds ID Token(JWT), userinfo endpoint, standard scopes(permissions)
  + Using Auth0 we don’t have to implement OpenID connect from scratch
* JSON Web Tokens