* SLDS
  + Salesforce lightning design system
  + A way of providing salesforce styling to our components
    - Lighting styling
  + It’s a group of prebuilt salesforce-made component blueprints
  + Basically html blueprints, css classes and other css stylings
  + Css classes
    - Accordion
      * Panels can expand and collapse
      * We can use the css class slds-accordion to give it this styling
    - Grid
      * slds-grid
      * Creates a flexible device-agnostic layout system
    - Icons
      * slds-icons
      * Not quite a class
    - Can use these prebuilt libraries of icons to enhance page accessibility and add context
    - Can be included in both visualforce and lighting component framework
* Single page application (SPAs)
  + Lightning component framework is made of two different frameworks
    - Lightning aura framework
    - Lightning web components
  + UI framework for designing and making dynamic, responsive, single page applications for both mobile and desktop users
  + With visualforce, we were using multi-page applications
    - Whenever you visited a page or made a call to the server, the server sends the markup to the user
  + Have many benefits in comparison to multi-page applications
    - Designed to work well on mobile devices
      * This is because they include responsive styling and they require fewer server calls
    - With aura, we minimize calls to the server
      * Uses Javascript to handle client side interaction
    - We make asyncronous calls to the server
* Aura
  + Promotes code reusability through use of components
    - Component – a self-contained reusable unit of code
  + Offers event-driven architecture using the MVCC
    - Model view controller controller
  + Aura pages cannot be rendered as pdfs
  + Has prebuilt components that we can use out of the box
  + Component bundles
    - A series of related files to give functionality to our custom components
  + When we create a component, we actually create the component bundle
    - It is made of 8 files
    - Component
      * This one you will always have to create
      * Basically the markup of our componenet
      * Can contain html and aura-specific markup
        + Similar to vf
    - Controller
      * This is a javascript controller to provide client side functionality to our component
      * Do not have to have it
    - Helper
      * Is a helper file for our Javascript controller
      * Also written in Javascript
      * Helps keep our controller logicless for readability
      * All instances of a component share a helper reducing the amount of information we are creating
    - Style
      * Css file to add in styling for our component
    - Documentation
      * Auradoc file associated with the component and is used to provide documentation for a component
      * Done in HTML markup
    - Design
      * Helps add our component to lightning pages and inside the app itself
      * Will likely not use it that much
    - Render
      * This is a Javascript file
      * Used to create custom rendering for our components’ elements
      * Easier to use render attribute
    - SVG
      * Scalable vector graphics
      * Used to define a custom icon for a lighting component
  + Creating a component
    - They are enclosed in <aura:component> tags and everything inside of them is the markup describing what will be displayed in that component
  + We need an application to view a component
    - Use <namespace:componentName>
    - Namespace will usually be c
* Lightning Applications
  + We cannot preview an aura component in the dev console
  + This is because an aura component is just a small piece of a bigger puzzle
  + Salesforce doesn’t provide a URL for our components
  + That’s where lightning apps come into play
  + Naming practices to call it the harness app
    - Is a top layer URL addressable component that will instantiate any component you wish to view
* Component Accessibility Interfaces
  + How can I add aura components to salesforce
  + We have a series io interfaces that tell our component where it can be used
  + Lightning:availableForFlowActions
  + Lightning:availableForFlowScreens
    - These make it viewable for flow actions and flow screens
  + Flexipage:availableForAllPageTypes
  + Flexipage:availableRecordHome
    - These make our component available for lightning app builder
  + Force:appHostable
    - This can be used in custom tab or mobile application
  + Force:lightningQuickAction
    - The component can be used for quick action
* Component Attributes
  + <aura:attribute>
    - Aura component that stores values within the component
    - Each attribute will have a name, data type and can have a default value
  + You can use the application to pass information into a component when the component is instantiated
* Expression syntax
  + {!v.myVar}
  + Value providers
    - Tell the file that contains them where to find a variable or a method
    - We have 3
      * V
        + Look in the view
        + Can be used in both the component markeup and the javascript controller
        + Refers to the aura attrubite name as a parameter
      * C
        + Controller
        + Refers to actions in the javascript controller and can only be used in the view
      * C
        + Refers to our apex controller
        + Can only be used in the javascript controller/helper files
* General notes
  + <aura:if>
    - Requires an ifTrue attribute to determine if something should be displayed
  + Lightning button
    - Another component from the lighting component library
    - Creates a button
    - Has a label attribute
      * Text that will be displayed on our button
    - Has a value attribute
      * Text that will be displayed in a pop-up when a user hovers over it
    - onClick
      * Listens for a click event and performs the action bound by the attribute
  + Javascript controller
    - Actions take in 3 parameters
      * Component
      * Event
      * Helper
        + Reference to the helper actually
    - We can reference our aura attributes from the component file by using the component.get(v.attribute) method
    - We will pass in the v value provided and the name of the attribute we want to grab
    - We can modify component aura attributes by grabbing the current value and changing it to what we want it to be
    - Then we can use the component.set(v.attribute, newValue) method to set a new value on the attribute
  + Application needs to extend force:slds