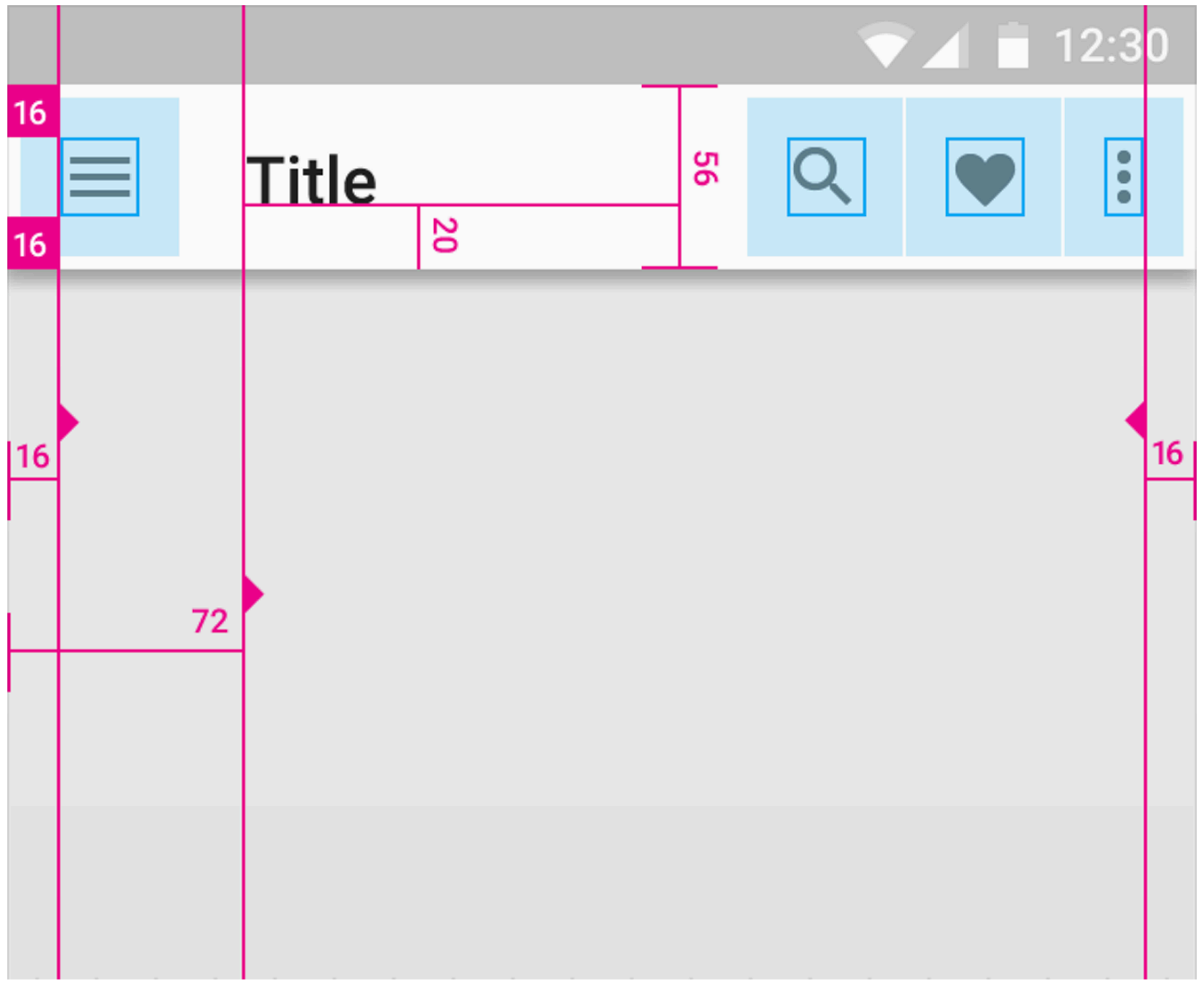
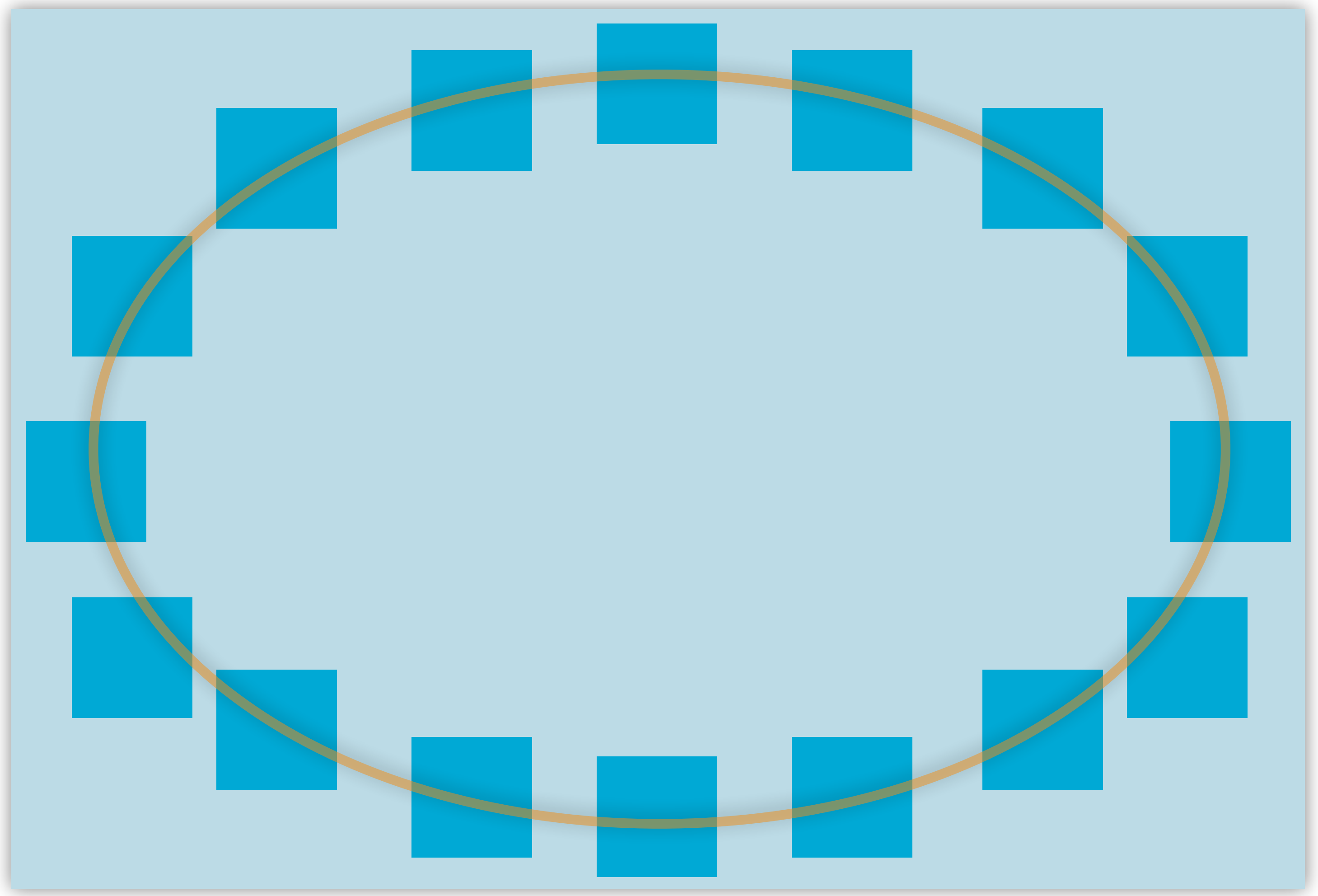


MOBILE APPLICATION DEVELOPMENT

ANDROID (2017)

LECTURE 08: CUSTOM LAYOUTS

CUSTOM LAYOUT CLASSES



- Sometimes, standard Android layouts are not capable of expressing the intentions of the programmer.
- Custom layouts allow for flexible and precise definition of non-standard layouts.
  - Useful when the layout of Views needs to be flexible, particularly when the number of Views to be laid out is not known at compile-time.
  - Example (pictured): what if the programmer wants Views to be arranged in an oval within the contents of a ViewGroup?

CUSTOM LAYOUT RESPONSIBILITIES

- Custom layouts are expected to behave in the following ways:
  - Act as a View, in the sense that they can (but often don't) draw content.
  - Respond to measure passes from parent layouts by reporting preferred measurements (as with custom Views).
  - Provide layout to child Views, attempting to respect their measured dimensions after performing a measure pass on them.
  - Define a structure that the programmer can rely on, including mechanisms for the programmer to specify layout parameters for the layout.

CREATING CUSTOM LAYOUTS

- Layout classes, including custom layouts defined by the programmer, are subclasses of the ViewGroup class which generally do the following:
  - Override constructors to provide custom initialization.
  - Override `onMeasure()` to define measurements for the layout based on the parent layout's restrictions AND the measurements reported by child Views.
  - Override `onLayout()` to provide layout to all enclosed child Views.
  - Define a nested class which subclasses LayoutParams and is used to provide layout information to the layout class.

ON MEASURE FUNCTION

- The `onMeasure()` function is called on custom layouts just like it is called on custom Views, and the way layouts must respond is similar.
- The function has the following objectives:
  - Uses the `getChildAt()` and `measureChild()` functions to obtain the sizings of all child Views, and calls the `resolveSizeAndState()` function for each child View to ensure sizings are set correctly. Uses size information from child Views to calculate the layout's size within its parent.
  - Should still use `suggestedMinimumHeight` / `suggestedMinimumWidth` to respect suggested minimum sizes for the layout itself, should still use `resolveSize()` before setting layout dimensions, and must still call `setMeasuredDimension()`.

ON LAYOUT FUNCTION

- The `onLayout()` function is called on custom layouts when the system has finished measuring Views, and wants the measurements of those views to be applied.
- The function has the following objectives:
  - Uses the `childCount` property and `childAt` function to iterate over child Views, obtaining their measurements with `measuredHeight` / `measuredWidth`.
  - Calculates layout rectangles for each child View, and applies gravity if needed.
  - Applies the layout to each child View by calling that child View's `layout()` function with the calculated layout rectangle for that child View.

NESTED LAYOUTPARAMS CLASS

- A LayoutParams class is needed for custom layouts if those layouts need more than the default functionality provided by LayoutParams.
  - By default, LayoutParams have width, height, padding, and gravity properties.
- If additional properties are desired, the default LayoutParams must be subclassed and should be added as a nested class inside the custom layout class.
  - The subclass is defined as a nested class within the custom layout so that the two are always associated with one another.