LAYOUT

Cristian Bogdan
cristi@kth.se

The basic building blocks

COMPONENTS

Label

HTML `<span>Some text</span>`

Java `javax.swing.JLabel`

Android `android.widget.TextView`

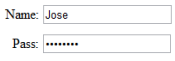
Button

HTML `<button />`

Java `javax.swing.JButton`

Android `android.widget.Button`

Text input



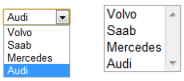
HTML
```
<input type="text"/>
<input type="password"/>
```

Java `javax.swing.JTextField`
`javax.swing.JPasswordField`

Android `android.widget.EditText`

`android:password="true"`

Dropdown/List



HTML
```
<select size=4>
<option>Volvo</option>
…
<select/>
```

Java `javax.swing.JComboBox`
`javax.swing.JList`

Android `android.widget.Spinner`
`android.widget.ListView`

Radio/check button

HTML
```
<input type="checkbox" />
<input type="radio" />
```

Java
```
javax.swing.JCheckBox
javax.swing.JRadioButton
```

```
android.widget.CheckBox
android.widget.RadioButton
```

Tooltip

HTML
```
<element title="Your tooltip" />
```

Java
```
javax.swing.JToolTip
```

```
N/A
```

Menu

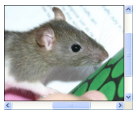
HTML
```
N/A
```

Java
```
javax.swing.JMenu
```

```
Done a bit differently
```

http://developer.android.com/guide/topics/ui/menus.html

Scroll container



HTML
```
<div style="height:100px; weight:
100px overflow:scroll"/>
```

Java
```
javax.swing.JScrollPane
```

```
android.widget.ScrollView
```

Tabs

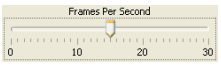
jQuery
```
$("#tabs").tabs();
```

http://jqueryui.com/demos/tabs/

Java
```
javax.swing.JTabbedPane
```

```
android.widget.TabHost
└android.widget.TabWidget
```

Slider



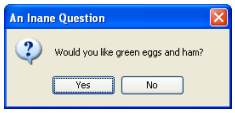
jQuery
```
$("#slider").slider();
```

http://jqueryui.com/demos/slider/

Java
```
javax.swing.JSlider
```

```
android.widget.SeekBar
```

Dialog




```
$("#dialog").dialog();
```

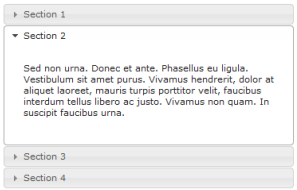
http://jqueryui.com/demos/dialog/

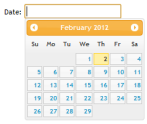
```
javax.swing.JDialog
```

```
android.app.AlertDialog
```

http://developer.android.com/guide/topics/ui/dialogs.html

Other

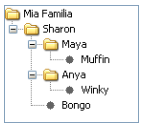





```
$("#accordion").accordion();
```

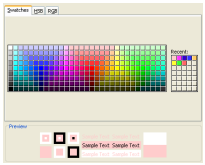
```
$("#datepicker").datepicker();
```

Other

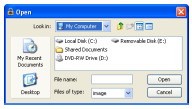


javax.swing.**JTree**

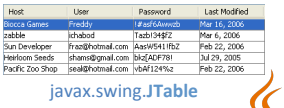
javax.swing.**JSplitPane**



javax.swing.**JColorChooser**



javax.swing.**JFileChooser**

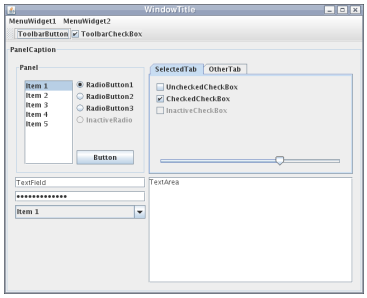


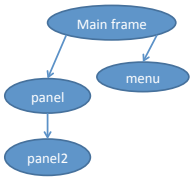
javax.swing.**JTable**

Putting components together

BUILDING THE INTERFACE

View Tree

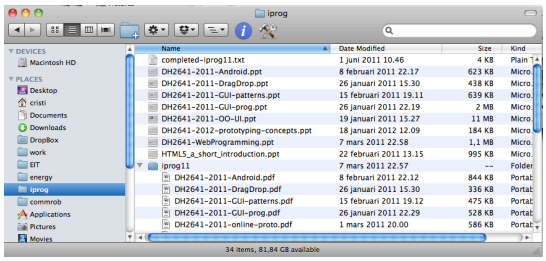




Tree terminology

- Trees: leaves (terminal nodes) and inner (parent) nodes
- Leaves in the tree are usually interface widgets
- Parent nodes group other nodes
  - other parent nodes or leaves
  - arranged (laid out) in specific ways
- HTML: “nodes” (Document Object Model, DOM)
- Swing; node: JComponent, parent node e.g. JPanel
- Android: node: View, parent node: ViewGroup

Tree Exercise

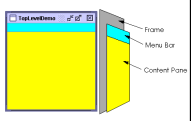


Tree APIs

- In all OO frameworks the parent node class inherits from the node class
  - Not unexpected, a parent node is a node ☺
- Node APIs allow getting and setting node attributes
- Parent node APIs provide ways to arrange (lay out) the nodes contained
- Parent node APIs allow traversing and changing the tree
  - getting the parent of a node
  - listing children of a parent node
  - adding and removing

Root nodes

- How to start, where to hang the tree?
- Easy in HTML, the <html> tag
  - New browser frames, dialogs
- Swing
  - interaction in a window (JFrame), a dialog box (JDialog), an applet (JApplet)
  - Each has a "content pane"
    - root parent node
    - an empty node is provided by default, just add to it *(code later)*
  - and can have a menu bar
- Android
  - Each Activity has a "content view", needs to be set *(code later)*



Node attributes

- Or properties
- Generic:
  - color, background color, font, name, …
  - size, position in the parent node
  - also "layout constraints"
  - e.g. alignment in the parent node
- Specific: depends on the node type
  - E.g. number of visible elements in a list widget
- In HTML and Android, attributes can be set by style rules

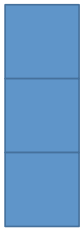
Constructing the interface

- Constructing a tree
  - Procedural
  - Declarative
- Styling rules to apply to a tree
- Graphically arranging children of a node
  - "layout"
  - Manual: set the size and position of each node manually
  - Automatic: use layout policies (managers) and layout constraints

Constructing a tree

Declarative

A tower of 3 blocks



Procedural

1. Put down block A.
2. Put block B on block A.
3. Put block C on block B.

Declarative - HTML

```
<body>
  <div>What is your name?</div>
  <div><input type="text"></div>
  <div><button>Send</button></div>
</body>
```

What is your name?

Send

HTML

Procedural - HTML

```
var div1 = $("<div/>").html("What is your name?");
$("body").append(div1);
var div2 = $("<div/>").append($("<input/>"));
$("body").append(div2);
var div3 = $("<div/>").append($("<button/>").html("Send"));
$("body").append(div3);
```

What is your name?

Send

HTML

Declarative - Android

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

res/layout/main.xml

Declarative - Android

```
package se.kth.csc.iprog;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroid extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

src/se/kth/csc/iprog/HelloAndroid.java

Procedural - Android

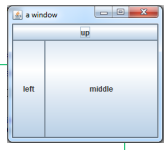
```
package se.kth.csc.iprog;

import android.app.Activity;
import android.os.Bundle;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;

public class HelloAndroid extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        LinearLayout layout = new LinearLayout(this);
        TextView tv = new TextView(this);
        tv.setText("Hello I'm a TextView");
        Button bt = new Button(this);
        bt.setText("Hello I'm a Button");
        layout.addView(tv);
        layout.addView(bt);
        setContentView(layout);
    }
}
```

src/se/kth/csc/iprog/HelloAndroid.java

Procedural – Java Swing




```
package se.kth.csc.iprog;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class GUISimple{
        public static void main(String[] args) {
                JFrame f = new JFrame("a window");
                JPanel panel = new JPanel();

                panel.setLayout(new BorderLayout());
                panel.add(new JButton("up"), BorderLayout.NORTH);
                panel.add(new JButton("middle"), BorderLayout.CENTER);
                panel.add(new JButton("left"), BorderLayout.WEST);

                f.getContentPane().add(panel);
                f.setSize(300, 500);
                f.setVisible(true);
        }
}
```

Java

Why automatic layout

- Window size changes
- Font changes
- Widget set/theme/skin changes
- Labels (e.g. internationalization)
- Dynamic changes in the tree

The automatic layout process

- A parent node has a layout *policy* for arranging its children
  - Some generic parents allow the policy to change
    - May have a default policy
    - e.g. JPanel default is BorderLayout
  - Example: a grid (table) layout: arrange children nodes in lines and columns
    - Number of lines and number of columns are layout attributes
- Children nodes have layout *constraints*
  - "wishes" towards the parent
  - "align me to the right and to the top"
  - "I want to eat up all the space left on this line"
- Since parent nodes have parents, and children can also contain children, the process is complex!

Flow Layout - HTML

```
<element style="display:inline"/>
```

```
<body>
   <button>Button 1</button>
   <button>Button 2</button>
   <button>Button 3</button>
   <button>Long-Named Button 1</button>
   <button>5</button>
</body>
```

Flow Layout – Java Swing

java.awt.**FlowLayout**

```
JPanel controls = new JPanel();
controls.setLayout(new FlowLayout());

//Add buttons to the experiment layout
controls.add(new JButton("Button 1"));
controls.add(new JButton("Button 2"));
controls.add(new JButton("Button 3"));
controls.add(new JButton("Long-Named Button 4"));
controls.add(new JButton("5"));
```

Flow Layout - Android

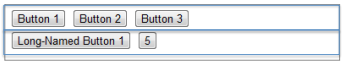
android.widget.**LinearLayout**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Block Layout - HTML

```
<element style="display:block"/> (default for div)
```

```
<body>
   <div>
     <button>Button 1</button>
     <button>Button 2</button>
     <button>Button 3</button>
   </div>
   <div>
     <button>Long-Named Button 1</button>
     <button>5</button>
   </div>
</body>
```



Grid Layout - HTML

```
<table>
  <tr>
    <td>Name:</td>
    <td><input /></td>
  </tr>
  <tr>
    <td>Surname:</td>
    <td><input /></td>
  </tr>
</table>
```

Name:
Surname:

Grid Layout – Java

```
JPanel controls = new JPanel();
controls.setLayout(new GridLayout(0,2));

//Add buttons to the experiment layout
controls.add(new JLabel("Name:"));
controls.add(new JTextField());
controls.add(new JLabel("Surname:"));
controls.add(new JTextField());
```

Also GridBagLayout (more complex), BoxLayout (simpler)
BorderLayout, particular case of GridBagLayout

Grid Layout - Android

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://
schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="1">
    <TableRow>
        <TextView
            android:text="@string/open"
            android:padding="3dip" />
        <TextView
            android:text="@string/open_shortcut"
            android:gravity="right"
            android:padding="3dip" />
    </TableRow>

    …
</TableLayout>
```

Views/Layouts/TableLayout/04. Stretchable
Open... Ctrl-O
Save As... Ctrl-Shift-S

Float Layout - HTML

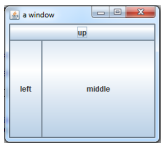
```
<element style="float:left"/> (or right)
```

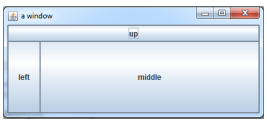
```
<body>
<button style="height:100px;float:left"> I'm a button </button>
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam
nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat
volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation
ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo
consequat.
</body>
```

I'm a button Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

I'm a button Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Border Layout – Java Swing

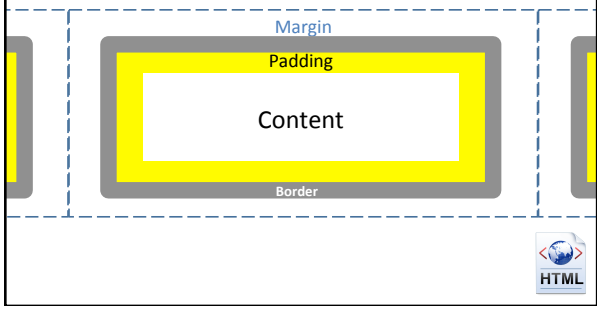




Card Layout

- Contains several children arranged in a stack
- Can change the order of the children to bring any of them on top (i.e. visible)
- Swing/AWT: java.awt.CardLayout
- Android: android.widget.FrameLayout
- HTML: hide and show various parts of the tree

Spacing - HTML



Adding some style

STYLYING

Why?

Coherent look and feel

Separation of the **structure** (view tree) from the details of **presentation**

Cascading Style Sheet (CSS)

Color
Font
Size
Position
Borders
…

CSS in HTML

As attribute

```
<button style="font-weight:bold">Save</button>
```

As style sheet

```
<style>
  button {
    font-weight:bold
  }
</style>
```

Selectors

```
<button class="menuButton" id="b1">Save</button>

<style>
  #b1 { … }                    By id
  .menuButton                  By class
     { padding:10;… }
  button { … }                 By tag

  #c1 div { … }                By element paths
</style>
```

PRIORITY

Style - Android

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

res/value/CodeFont.xml

Layout Exercise