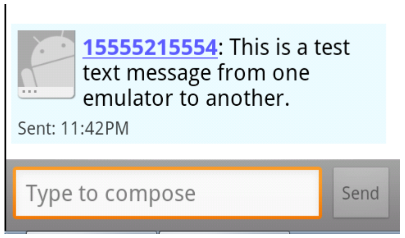
CS 193A

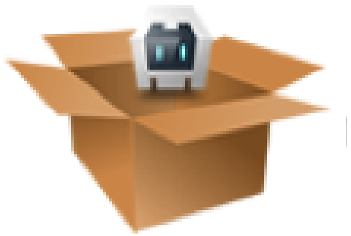
What's Next?
(Other Topics We Didn't Cover)

Outline

- Sending and Receiving SMS messages
- Testing
- Games and Graphics
- Android Wear
- PhoneGap (multi-platform app development)


15555215554: This is a test text message from one emulator to another.

Sent: 11:42PM



Sending and Receiving SMS

- To send/receive SMS messages, need permissions:
  - in AndroidManifest.xml:

```
<uses permission
      android:name="android.permission.SEND_SMS" />
<uses permission
      android:name="android.permission.RECEIVE_SMS" />
```

- Sending an SMS message:

```
SmsManager mgr = SmsManager.getDefault();
mgr.sendTextMessage("phoneNumber", null,
                    "msg text", null, null);
```

- Receiving an SMS message:
  - have to set up a broadcast receiver (see next slide)

Receiving SMS

- Declare a broadcast receiver in AndroidManifest.xml:

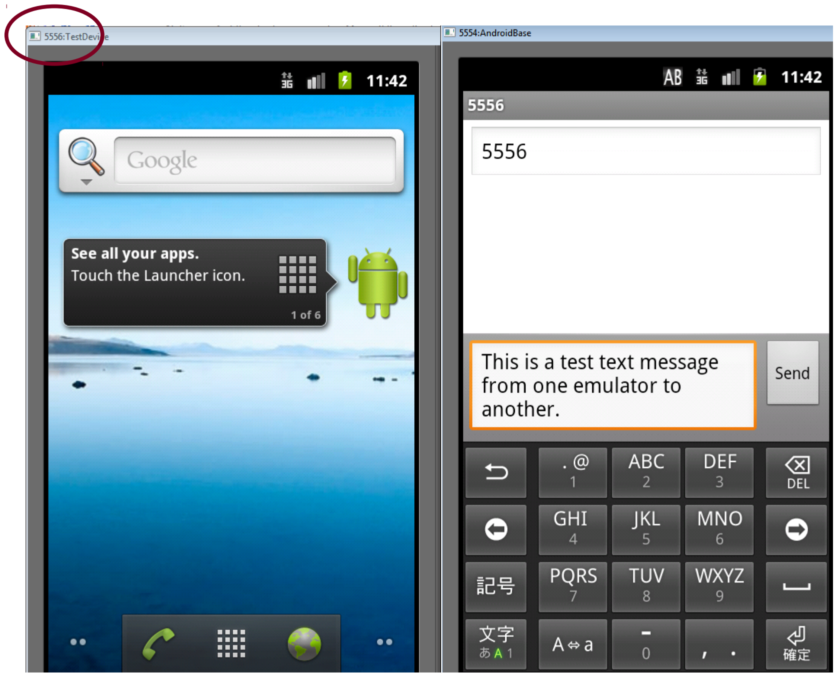
```
<receiver android:name=".SmsBroadcastReceiver" android:exported="true">
    <intent-filter android:priority="999" >
        <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

- Write the code for your broadcast receiver:

```
public class SmsBroadcastReceiver extends BroadcastReceiver {
    public void onReceive(Context context, Intent intent) {
        SmsMessage[] msgs =
            Telephony.Sms.Intents.getMessagesFromIntent(intent);
        String msgText = "";
        for (int i = 0; i < msgs.length; ++i) {
            String smsBody = msgs[i].getMessageBody().toString();
            String address = msgs[i].getOriginatingAddress();
            msgText += "SMS From: " + address + "\n" + smsBody + "\n";
        }
        Toast.makeText(context, msgText, Toast.LENGTH_SHORT).show();
    }
}
```

Testing SMS

- need to set up two emulators to test it
  - emulator's "phone number" is port number (e.g. 5556)


5556:TestDevice

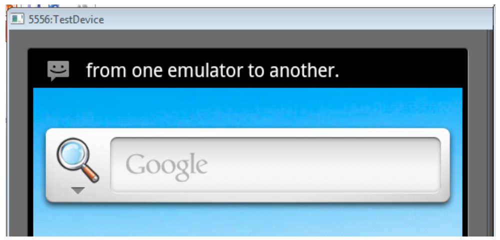
See all your apps.
Touch the Launcher icon.
1 of 6

5554:AndroidBase

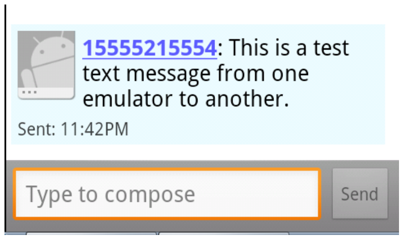
5556

5556

This is a test text message from one emulator to another.

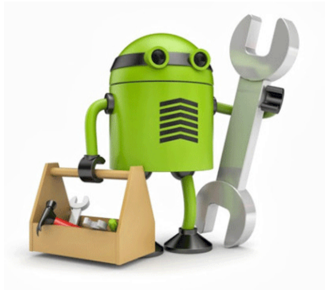

5556:TestDevice

from one emulator to another.


15555215554: This is a test text message from one emulator to another.

Sent: 11:42PM

Testing an app



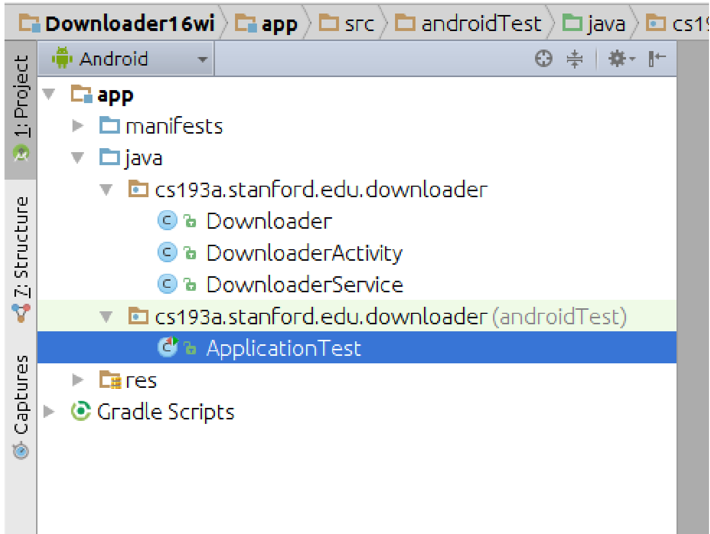
- There are many kinds of testing you might want to do:
  - **unit testing**: Small tests of individual classes and methods.
  - **functional / UI testing**: Simulate running the entire app.
  - **load testing**: See how the app handles many actions/requests.
  - **security testing**: Check that the app cannot be compromised.
  - ...
- Luckily, Android has testing "baked in" as a core feature.
  - Several classes, libraries, systems, etc. for testing out of the box.
  - Many third-party libraries to add further testing features.

Unit testing

- Every Android Studio app creates an `androidTest` src folder where you can put test cases.

- **JUnit**: Popular library for unit testing Java apps.

```
public class EmailValidatorTest {
    @Test
    public void testEmailValidator() {
        assertTrue(MyEmail.isValid("name@email.com"));
    }
    ...
}
```


Downloader16wi > app > src > androidTest > java > cs19

```
Android
▼ app
  ▶ manifests
  ▼ java
    ▼ cs193a.stanford.edu.downloader
        Downloader
        DownloaderActivity
        DownloaderService
    ▼ cs193a.stanford.edu.downloader (androidTest)
        ApplicationTest
  ▶ res
▶ Gradle Scripts
```

Mocks and Stubs

- Testing parts of an app in isolation is hard.
  - Use **Mockito** framework to make a fake "mock" version of other parts of the app.
  - http://mockito.org/

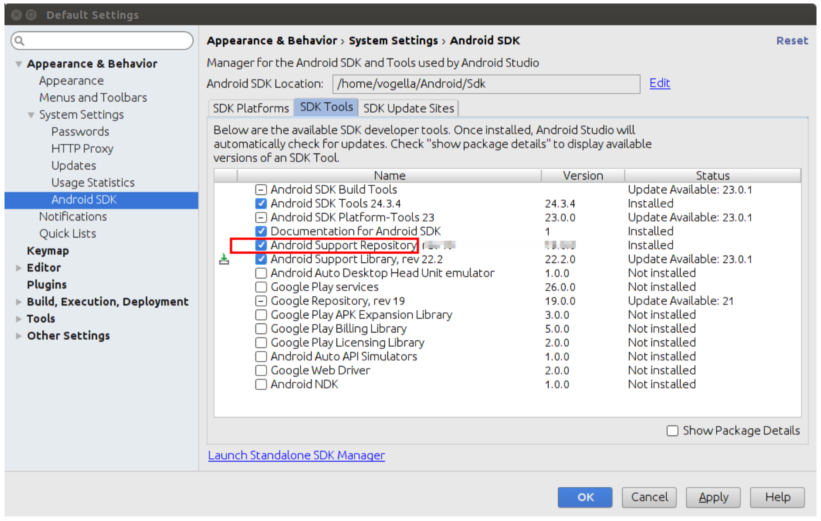
```
@RunWith(MockitoJUnitRunner.class)
public class UnitTestSample {
    private static final String FAKE_STRING = "HELLO WORLD";

    @Mock Context mMockContext;

    @Test
    public void readStringFromContext_LocalizedString() {
        when(mMockContext.getString(R.string.hello_word))
                .thenReturn(FAKE_STRING);
        ClassUnderTest myObjectUnderTest =
                       new ClassUnderTest(mMockContext);
        String result = myObjectUnderTest.getHelloWorldString();
        assertEquals(FAKE_STRING, result);
    }
}
```

UI testing with Espresso

- **Espresso**: Library for automated testing of an app's UI.
  - http://developer.android.com/training/testing/ui-testing/espresso-testing.html
  - Add to project's build.gradle:

```
dependencies {  ...
    androidTestCompile
      'com.android.support.test.espresso:espresso-core:2.2.1'
}
```

  - Install Android Support Repository / Library in Android Studio



Default Settings

Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

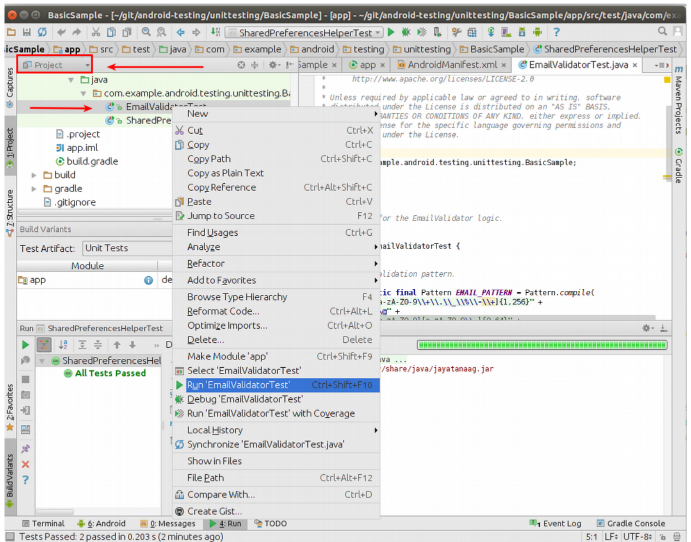
Android SDK Location: /home/vogella/Android/Sdk

SDK Platforms | SDK Tools | SDK Update Sites

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
▣ Android SDK Build Tools		Update Available: 23.0.1
✓ Android SDK Tools 24.3.4	24.3.4	Installed
▣ Android SDK Platform-Tools 23	23.0.0	Update Available: 23.0.1
✓ Documentation for Android SDK	1	Installed
✓ Android Support Repository [illegible]	[illegible]	Installed
✓ Android Support Library, rev 22.2	22.2.0	Update Available: 23.0.1
☐ Android Auto Desktop Head Unit emulator	1.0.0	Not installed
☐ Google Play services	26.0.0	Not installed
▣ Google Repository, rev 19	19.0.0	Update Available: 21
☐ Google Play APK Expansion Library	3.0.0	Not installed
☐ Google Play Billing Library	5.0.0	Not installed
☐ Google Play Licensing Library	2.0.0	Not installed
☐ Android Auto API Simulators	1.0.0	Not installed
☐ Google Web Driver	2.0.0	Not installed
☐ Android NDK	1.0.0	Not installed

☐ Show Package Details



BasicSample - [~/git/android-testing/unittesting/BasicSample] - [app] - ~/git/android-testing/unittesting/BasicSample/app/src/test/java/com/exa

```
*      http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software
ANTIES OR CONDITIONS OF ANY KIND, either express or implied.
nse for the specific language governing permissions and
 under the License.

ample.android.testing.unittesting.BasicSample;

for the EmailValidator logic.

ailValidatorTest {

lidation pattern.

ic final Pattern EMAIL_PATTERN = Pattern.compile(
-zA-Z0-9\\+\\.\\_\\%\\-\\+]{1,256}" +
@" +
```

All Tests Passed

Tests Passed: 2 passed in 0.203 s (2 minutes ago)

SharedPreferencesHelperTest
icSample › app › src › test › java › com › example › android › testing › unittesting › BasicSample › SharedPreferencesHelperTest
Project
ample × app × AndroidManifest.xml × EmailValidatorTest.java ×
java
com.example.android.testing.unittesting.Ba
EmailValidatorTest
SharedPre
.project
app.iml
build.gradle
build
gradle
.gitignore
Build Variants
Test Artifact: Unit Tests
Module
app
Run SharedPreferencesHelperTest
SharedPreferencesHel
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy as Plain Text
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Jump to Source F12
Find Usages Ctrl+G
Analyze
Refactor
Add to Favorites
Browse Type Hierarchy F4
Reformat Code... Ctrl+Alt+L
Optimize Imports... Ctrl+Alt+O
Delete... Delete
Make Module 'app' Ctrl+Shift+F9
Select 'EmailValidatorTest'
Run 'EmailValidatorTest' Ctrl+Shift+F10
Debug 'EmailValidatorTest'
Run 'EmailValidatorTest' with Coverage
Local History
Synchronize 'EmailValidatorTest.java'
Show in Files
File Path Ctrl+Alt+F12
Compare With... Ctrl+D
Create Gist...
Captures
1: Project
7: Structure
2: Favorites
Build Variants
Maven Projects
Gradle
Terminal
6: Android
0: Messages
4: Run
TODO
Event Log
Gradle Console
5:1 LF÷ UTF-8÷

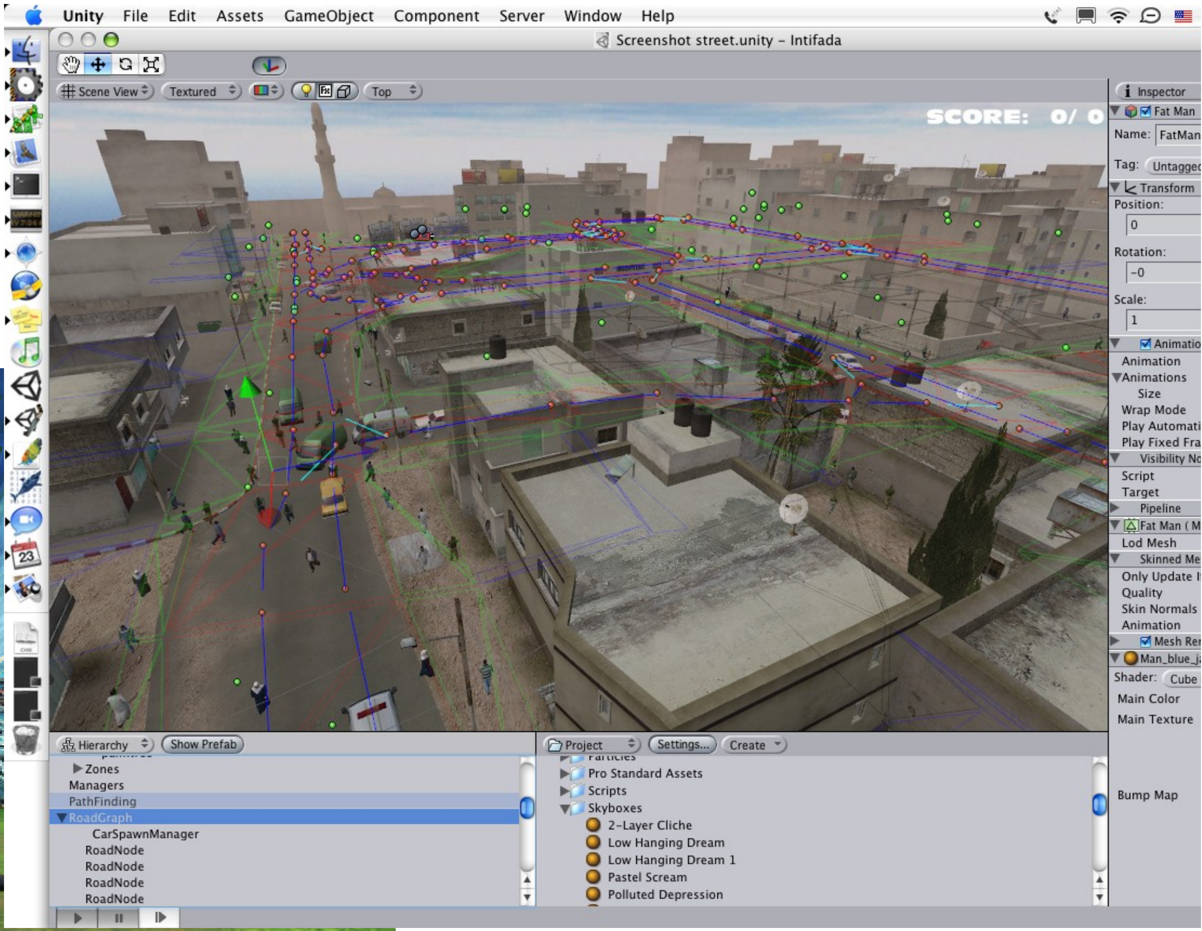
An Espresso test class

```
@RunWith(AndroidJUnit4.class)
@LargeTest
public class ChangeTextBehaviorTest {
    private String expected = "Espresso";   // string we expect

    @Rule
    public ActivityTestRule<MainActivity> mActivityRule =
            new ActivityTestRule<>(MainActivity.class);

    @Test
    public void changeText_sameActivity() {
        // type text and then press a button
        Espresso.onView(withId(R.id.my_edit_text))
                .perform(typeText(expected), closeSoftKeyboard());
        Espresso.onView(withId(R.id.my_button)).perform(click());

        // check that the text was changed
        Espresso.onView(withId(R.id.my_text_view_output))
                .check(matches(withText(expected)));
    }
}
```

Making Games with Unity

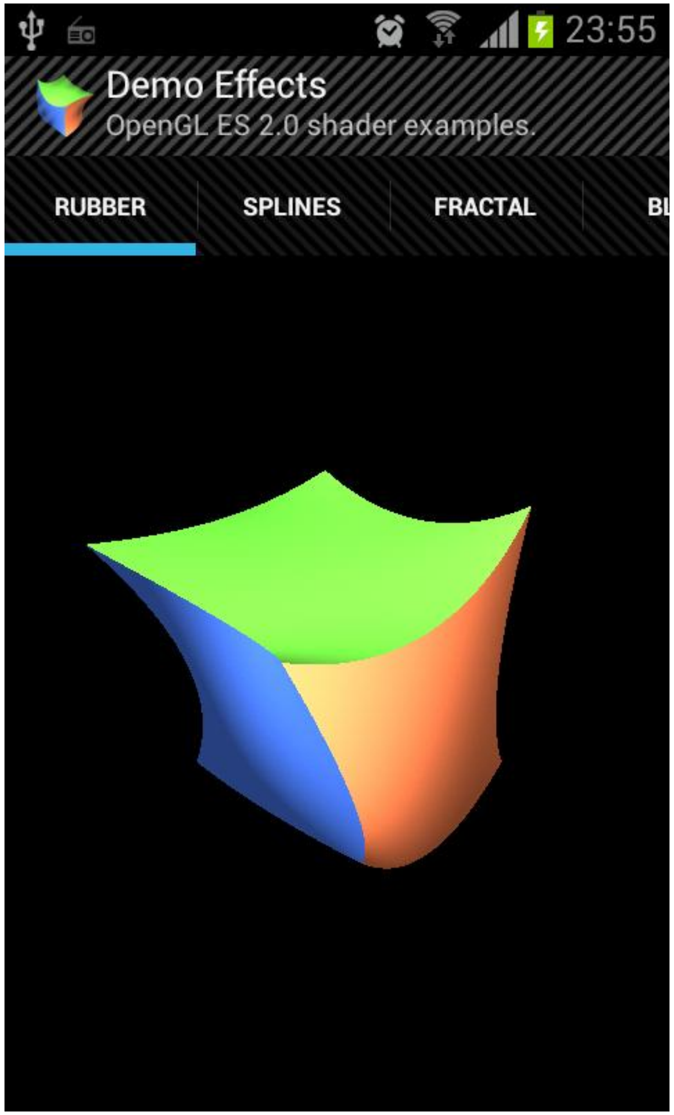
- https://unity3d.com/

Use [illegible] to choose the right treat for the dragon. Use [illegible] to give.



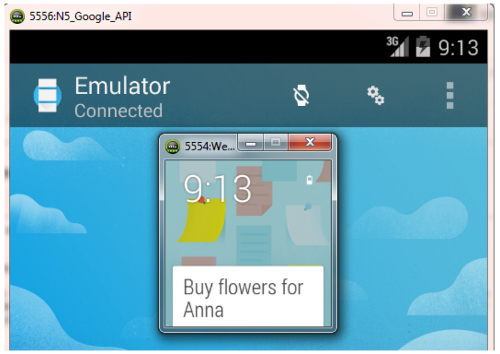
3D Graphics with OpenGL ES

- http://developer.android.com/guide/topics/graphics/opengl.html

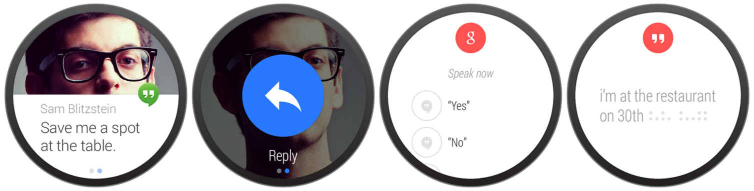


Android Wear (watches)

- **Android Wear**: Version of Android that runs on watches and other wearable devices.
  - http://developer.android.com/training/building-wearables.html



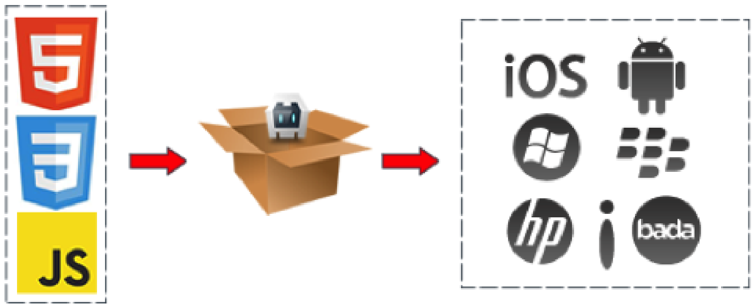
- Modifying an app to run on Wear
  - Probably re-do your UI and layout
    - new set of widgets specific to Wear
  - Heavy emphasis on notifications
  - Greater use of voice input over typing/clicking
  - Testing: Create an emulator device to represent the watch
    - tricky; must be **paired** with another device/emulator representing a phone

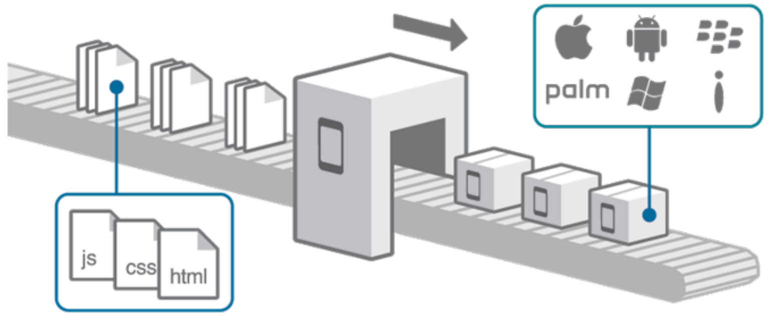


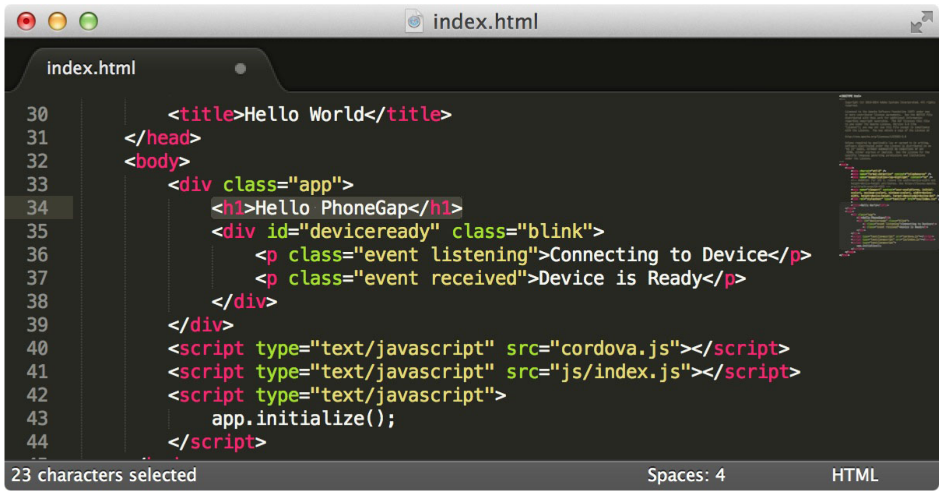
PhoneGap

- PhoneGap: Cross-platform app development framework.
  - based on HTML and JavaScript
  - write code once, deploy to web app, Android, iOS, more
  - http://phonegap.com/

- Similar frameworks
  - Xamarin, React Native, Qt, AppInventor, Appcelerator







```
        <title>Hello World</title>
    </head>
    <body>
        <div class="app">
            <h1>Hello PhoneGap</h1>
            <div id="deviceready" class="blink">
                <p class="event listening">Connecting to Device</p>
                <p class="event received">Device is Ready</p>
            </div>
        </div>
        <script type="text/javascript" src="cordova.js"></script>
        <script type="text/javascript" src="js/index.js"></script>
        <script type="text/javascript">
            app.initialize();
        </script>
```

23 characters selected   Spaces: 4   HTML

Thank you!

- Thanks for a great class!
  - Special thanks to our TAs, May, Dae Hyun, and Senthilnathan!
- Please give me your feedback about CS 193A:
  - fill out our anonymous survey
  - let me know what you'd like in future quarters

- CS 193A will be offered again next year (Winter 2018)
  - Tell your friends!
- Coming soon: HW7