
Google™
09
I/O

Powering Mobile Apps With Social Data

Ryan Boyd
May 28th, 2009

Post your questions for this talk on Google Moderator:
http://tinyurl.com/mobile-social

Post your feedback for this talk:
http://haveasec.com/io/

139,287,900

Smartphones sold in 2008 (Gartner)

Agenda

Agenda

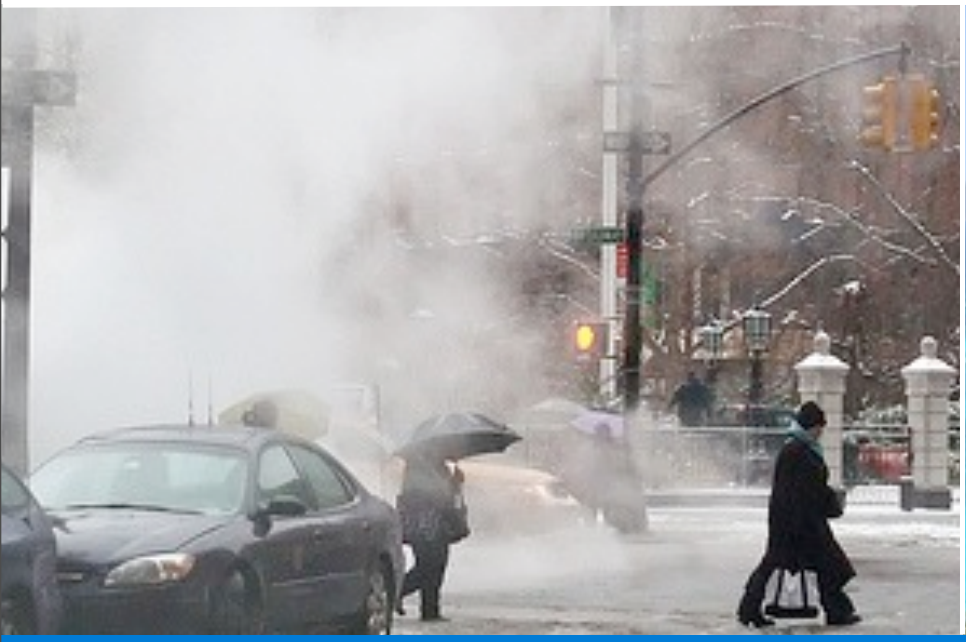
- Every app is better when ...
- Enabling technology
- Choosing the right road to success
- Authorization with OAuth
- Demonstration and code
  - Web app
  - Web app + native functionality
  - Native app
- Other devices and the future
- Questions and Answers

Every app is better when…

it’s Social!

Google 09 I/O

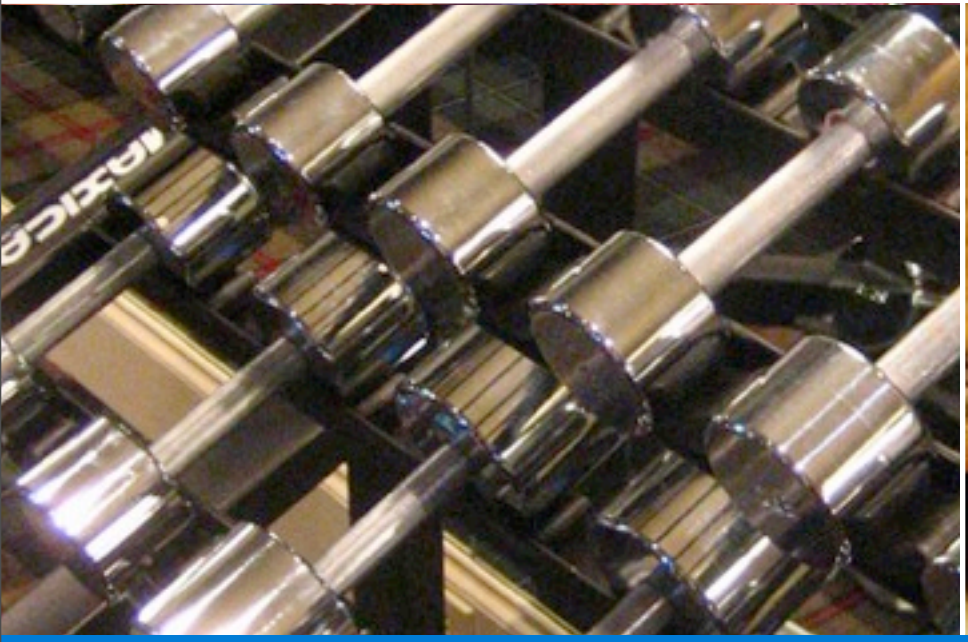





Weather

Google I/O 09






Exercise & Weight Loss

Google I/O 09

Every app is better when it's social

Every app is better when it's social

- Shopping

Every app is better when it's social

- Shopping
- Restaurant directory

Every app is better when it's social

- Shopping
- Restaurant directory
- Sports

Every app is better when it's social

- Shopping
- Restaurant directory
- Sports
- Movies

Every app is better when it's social

- Shopping
- Restaurant directory
- Sports
- Movies
- Subway Map

Every app is better when it's social

- Shopping
- Restaurant directory
- Sports
- Movies
- Subway Map
- Stocks & Finance

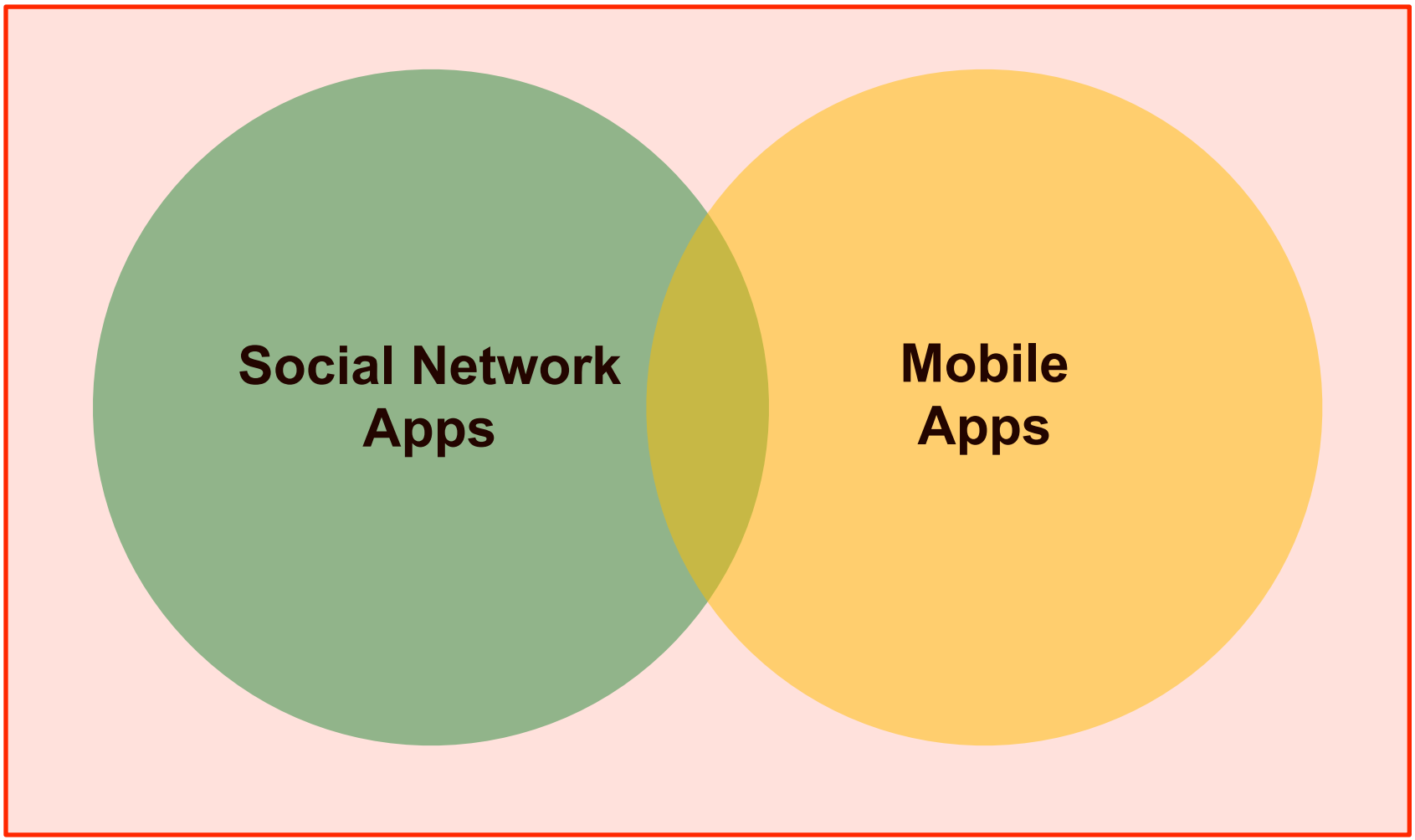
Every app is better when it's social

- Shopping
- Restaurant directory
- Sports
- Movies
- Subway Map
- Stocks & Finance
- Tic Tac Toe

Every app is better when it's social

- Shopping
- Restaurant directory
- Sports
- Movies
- Subway Map
- Stocks & Finance
- Tic Tac Toe
- Solitaire!

Types of Apps



Enabling Technology

Enabling Technology

- OpenSocial 0.8.1
  - REST and RPC Protocols
- OAuth for authorization
- OpenSocial Client Libraries for Java, PHP, Python, Objective-C and Ruby
- Sample Code for Android and iPhone
- The web!

Choosing the right road

Choosing the right road to success

Choosing the right road to success

- Web application

Choosing the right road to success

- Web application
- Web application with native functionality

Choosing the right road to success

- Web application
- Web application with native functionality
- Native application

Choosing the right road to success

Choosing the right road to success

- Web application

Choosing the right road to success

- Web application
  - Single app

Choosing the right road to success

- Web application
  - Single app
  - Open standards

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms
- Native application

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms
- Native application
  - Higher performance

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms
- Native application
  - Higher performance
  - Access to full-suite of native APIs

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms
- Native application
  - Higher performance
  - Access to full-suite of native APIs
  - Can still ‘call the web,’ when needed

Choosing the right road to success

- Web application
  - Single app
  - Open standards
  - Not in market/app store
- Web application with native functionality
  - Can have improved performance over standard web apps
  - Can be made available offline
  - Access to native APIs - such as geolocation
  - Standards emerging, may need to fork for different platforms
- Native application
  - Higher performance
  - Access to full-suite of native APIs
  - Can still ‘call the web,’ when needed
  - Requires building app for each target platform

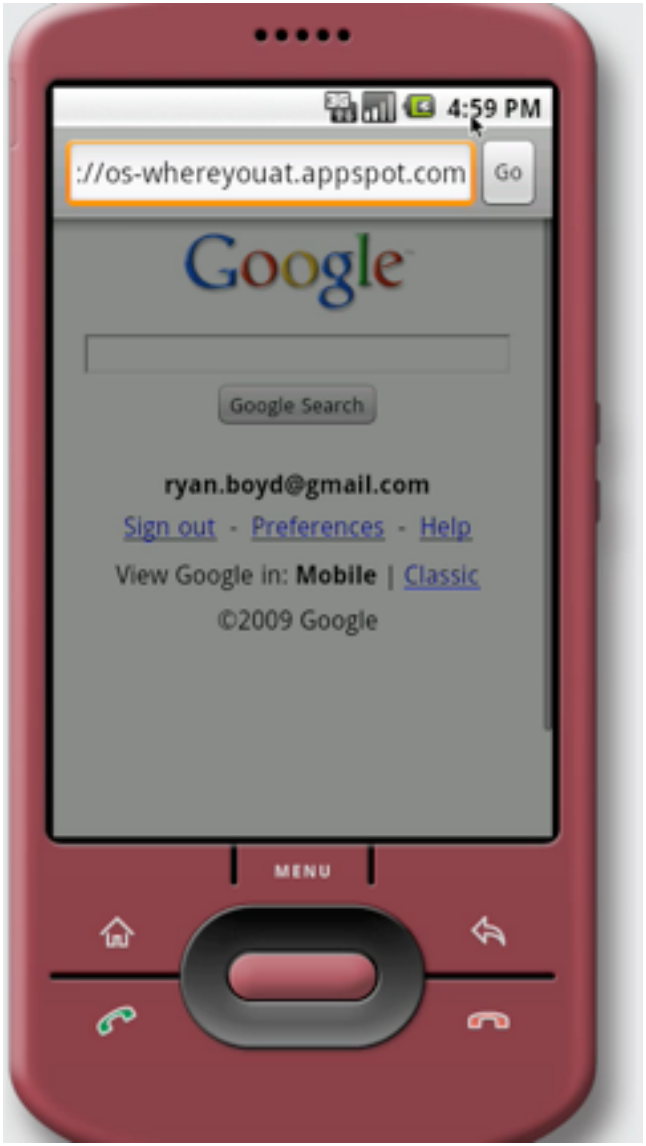
http://www.flickr.com/photos/caveman_92223/3185534518

Web Applications

Google 09 I/O

What you doing?

What you doing?



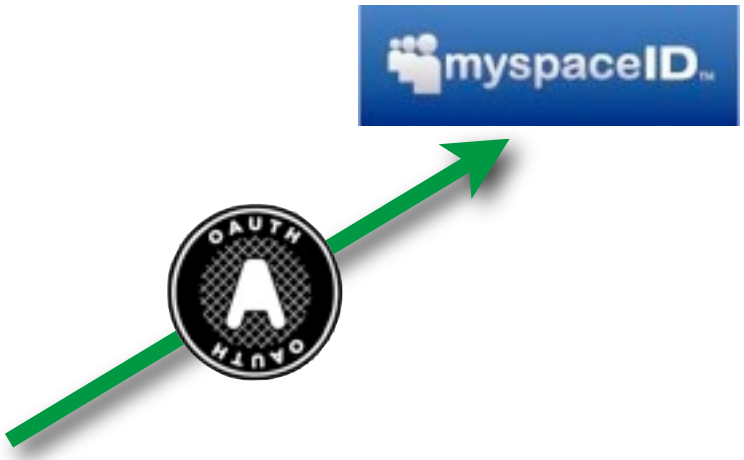
Note: videos edited

What you doing? - data flowing over the net

What you doing? - data flowing over the net

Mobile Device

What you doing? - data flowing over the net

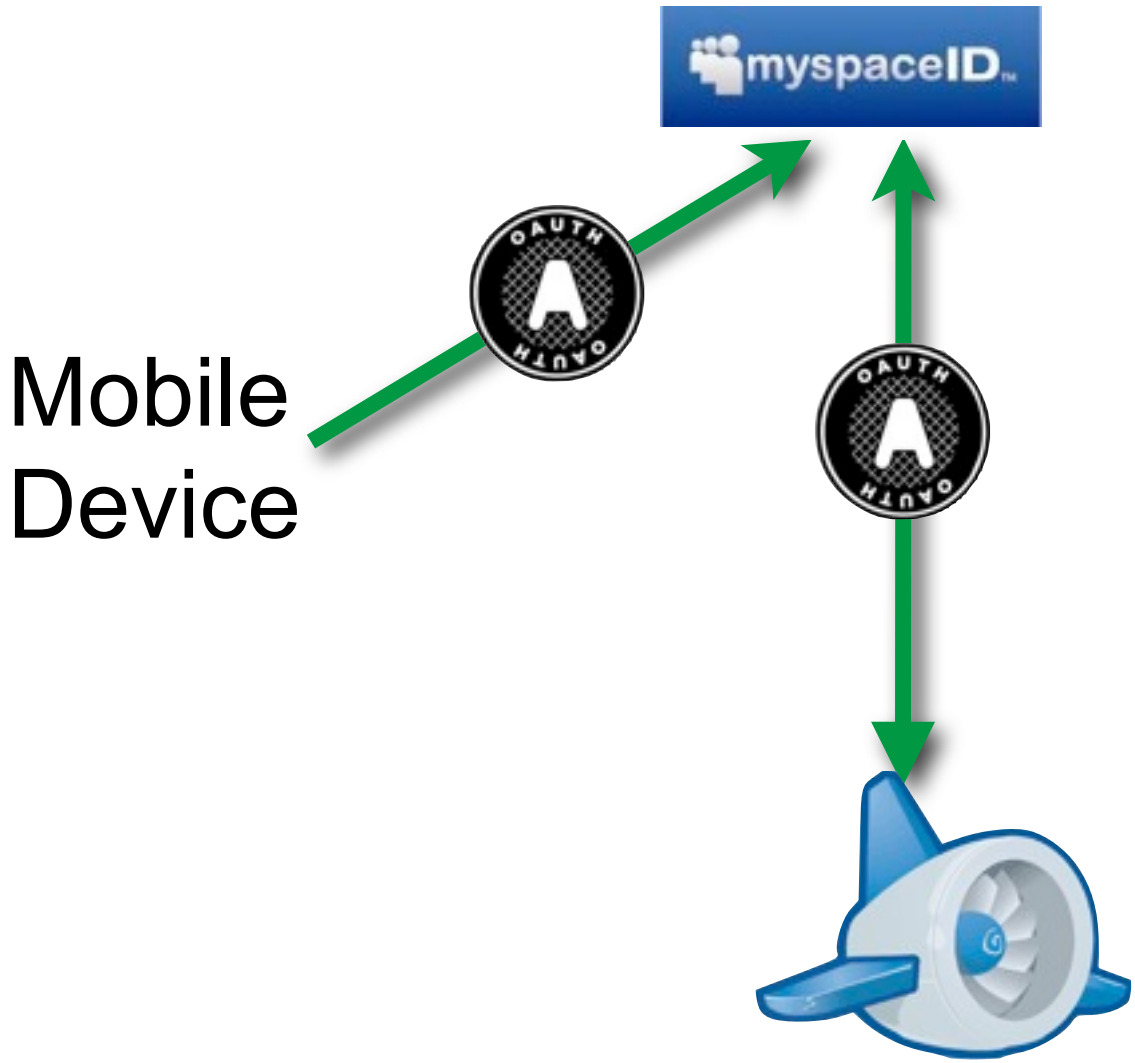


Mobile Device

myspace.com
a place for friends.

Google I/O 09

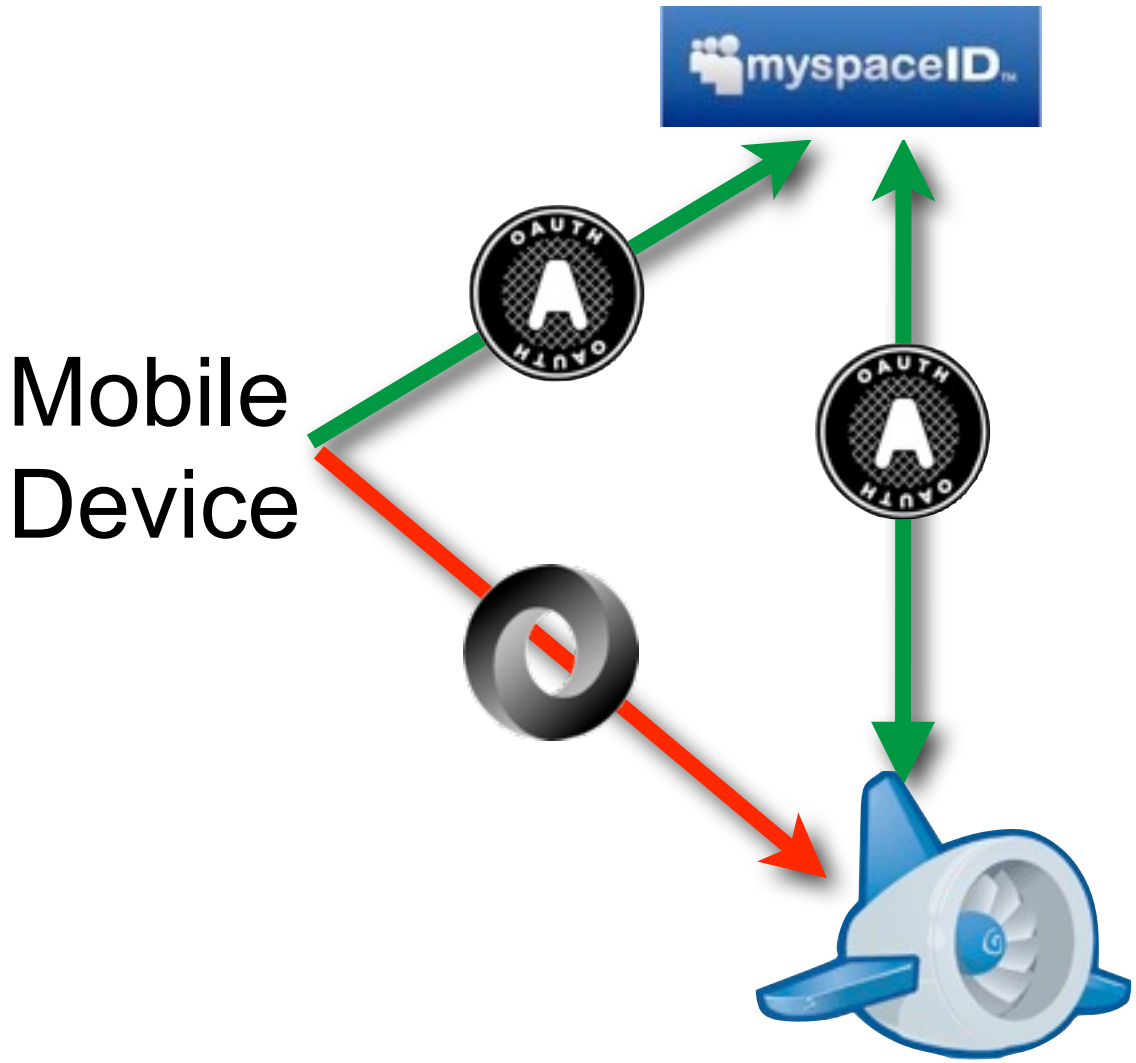
What you doing? - data flowing over the net



myspace.com
a place for friends.

Google 09 I/O

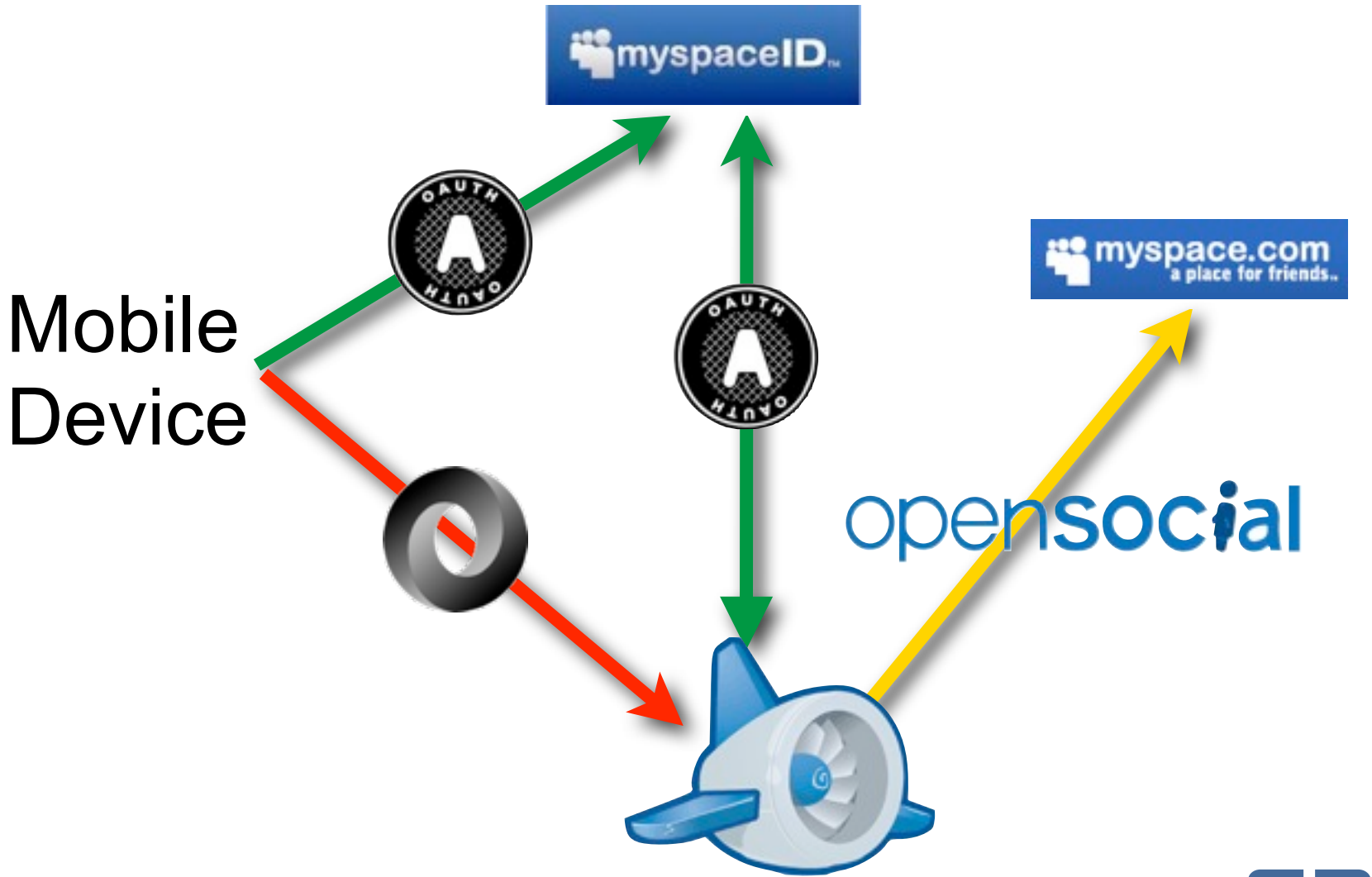
What you doing? - data flowing over the net



myspace.com
a place for friends.

Google I/O 09

What you doing? - data flowing over the net



What you doing? - the (relevant) JS code

- `getRecentUpdates`
  - Retrieves recent public updates
- `getFriendUpdates`
  - Retrieves public and private updates for friends of the current user

What you doing? - the backend code

- `Update`
  - Represents a list of updates including posted time, content of post and the OpenSocial ID of the author
- `User`
  - Represents the currently authenticated user and holds the ID, container, name

What you doing? - the social code

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   consumerKey);
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   consumerKey);
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   consumerSecret);
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   consumerKey);
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   consumerSecret);

client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken);
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   consumerKey);
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   consumerSecret);

client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken);
client.setProperty(
               OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
               accessTokenSecret);
```

What you doing? - the social code

```
provider = OpenSocialProvider.MYSPACE;
consumerKey = MYSPACE_KEY;
consumerSecret = MYSPACE_SECRET;

client = new OpenSocialClient(provider);
client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   consumerKey);
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   consumerSecret);

client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken);
client.setProperty(
               OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
               accessTokenSecret);

Collection<OpenSocialPerson> friends = client.fetchFriends();
```

Where you at?

Where you at? - Enhanced with native functionality

Where you at? - the (relevant) JS code

- `getNearbyUpdates`
  - Retrieves updates within map boundary, using geohashing
- `watchPosition`
  - Uses Google Gears to get current location
- `updatePosition`
  - Re-centers map based on current location, re-calculates the boundary and re-fetches nearby updates

Native App - Divide and Conquer

Google I/O 09

Divide and Conquer - a social game

Divide and Conquer - data flowing over the net

Divide and Conquer - data flowing over the net

Google 09 I/O

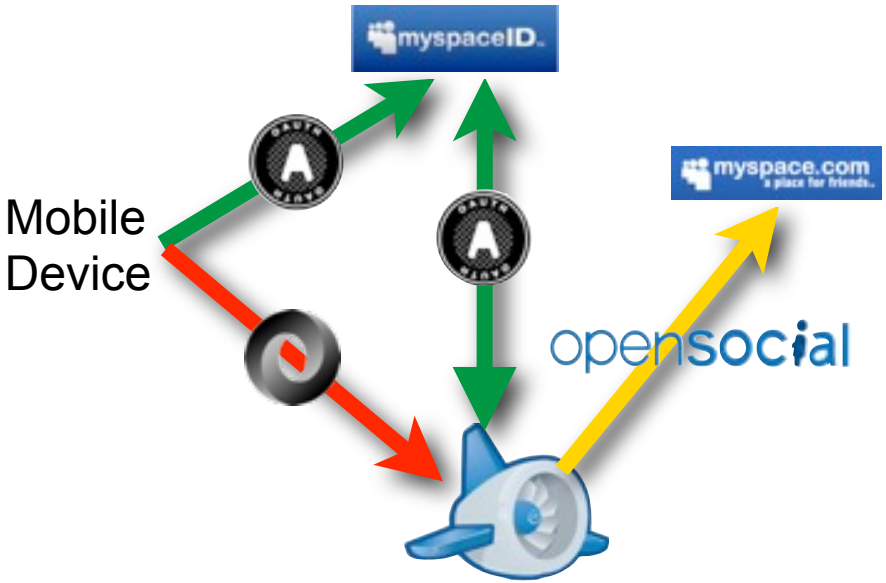
Divide and Conquer - data flowing over the net

Divide and Conquer - data flowing over the net

Divide and Conquer - data flowing over the net

Architectural Differences - Web vs Native

Web App



Native App

Google I/O 09

Divide and Conquer - the (relevant) mobile code

Divide and Conquer - the (relevant) mobile code

- **`DivideAndConquerActivity`**
  - Most game play - modified to store scores, initiate the `SocialHandler` and display the `TopScoresDialog`

Divide and Conquer - the (relevant) mobile code

- **`DivideAndConquerActivity`**
  - Most game play - modified to store scores, initiate the `SocialHandler` and display the `TopScoresDialog`
- **`SocialNetworkChooserListPreference`**
  - List of social networks, handles selection

Divide and Conquer - the (relevant) mobile code

- **`DivideAndConquerActivity`**
  - Most game play - modified to store scores, initiate the `SocialHandler` and display the `TopScoresDialog`
- **`SocialNetworkChooserListPreference`**
  - List of social networks, handles selection
- **`SocialHandler`**
  - Interaction with the SNSs and App Engine backend
  - Storage of cached data

Divide and Conquer - the (relevant) mobile code

- **`DivideAndConquerActivity`**
  - Most game play - modified to store scores, initiate the `SocialHandler` and display the `TopScoresDialog`
- **`SocialNetworkChooserListPreference`**
  - List of social networks, handles selection
- **`SocialHandler`**
  - Interaction with the SNSs and App Engine backend
  - Storage of cached data
- **`SocialHandler.SocialThread`**
  - Retrieval of social data

Divide and Conquer - the (relevant) mobile code

- **`DivideAndConquerActivity`**
  - Most game play - modified to store scores, initiate the `SocialHandler` and display the `TopScoresDialog`
- **`SocialNetworkChooserListPreference`**
  - List of social networks, handles selection
- **`SocialHandler`**
  - Interaction with the SNSs and App Engine backend
  - Storage of cached data
- **`SocialHandler.SocialThread`**
  - Retrieval of social data
- **`TopScoresDialog`**
  - UI for displaying social scoreboard

Divide and Conquer - the backend code

Divide and Conquer - the backend code

- `MainHandler`
  - `post()`
    - Accepts HTTP posts requests, calls appropriate RPC method

Divide and Conquer - the backend code

- `MainHandler`
  - `post()`
    - Accepts HTTP posts requests, calls appropriate RPC method
- `RPCMethods`
  - `GetTopScores()`
    - Returns 5 top scores for the app

Divide and Conquer - the backend code

- `MainHandler`
  - `post()`
    - Accepts HTTP posts requests, calls appropriate RPC method
- `RPCMethods`
  - `GetTopScores()`
    - Returns 5 top scores for the app
  - `GetTopScoresForFriends()`
    - Given a list of friends, returns top score for each

Divide and Conquer - the backend code

- `MainHandler`
  - `post()`
    - Accepts HTTP posts requests, calls appropriate RPC method
- `RPCMethods`
  - `GetTopScores()`
    - Returns 5 top scores for the app
  - `GetTopScoresForFriends()`
    - Given a list of friends, returns top score for each
  - `SetScore()`
    - Sets the score, if top score achieved

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Google I/O 09

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - Step by Step

Divide and Conquer - In the background

Divide and Conquer - getTopScoreList()

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
  returnList = getScoreInfoFromJson(callBackendRPC(args));
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
  returnList = getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
  returnList = getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
  ex.printStackTrace();
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
  returnList = getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
  ex.printStackTrace();
}
```

Divide and Conquer - getTopScoreList()

```
ArrayList<Hashtable<String,String>> returnList = null;
List<Object> args = new ArrayList<Object>();
args.add("GetTopScores");
try {
  returnList = getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
  ex.printStackTrace();
}
return returnList;
```

Divide and Conquer - getSelfAndFriends()

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
client.setProperty(
              OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
client.setProperty(
               OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
               accessToken.secret);
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
client.setProperty(
                OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
                accessToken.secret);
client.setProperty(OpenSocialClient.Property.REST_BASE_URI,
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   "anonymous");
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   "anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
client.setProperty(
                OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
                accessToken.secret);
client.setProperty(OpenSocialClient.Property.REST_BASE_URI,
                   provider.restEndpoint);
```

Divide and Conquer - getSelfAndFriends()

```
List<OpenSocialPerson> friends = new
ArrayList<OpenSocialPerson>();

OpenSocialPerson self = null;
OpenSocialClient c = new OpenSocialClient();
OpenSocialProvider provider = util.getProvider();


client.setProperty(OpenSocialClient.Property.CONSUMER_KEY,
                   “anonymous”);
client.setProperty(OpenSocialClient.Property.CONSUMER_SECRET,
                   “anonymous);
client.setProperty(OpenSocialClient.Property.ACCESS_TOKEN,
                   accessToken.token);
client.setProperty(
                OpenSocialClient.Property.ACCESS_TOKEN_SECRET,
                accessToken.secret);
client.setProperty(OpenSocialClient.Property.REST_BASE_URI,
                   provider.restEndpoint);
client.setProperty(OpenSocialClient.Property.RPC_ENDPOINT,
                   provider.rpcEndpoint);
```

Divide and Conquer - getSelfAndFriends()

Divide and Conquer - getSelfAndFriends()

```
friends = c.fetchFriends();
self = c.fetchPerson();
```

Divide and Conquer - getSelfAndFriends()

```
friends = c.fetchFriends();
self = c.fetchPerson();


for (OpenSocialPerson person : friends) {
  returnHash.put(provider.toString() + "_" +
                 person.getId(), person);
}
if (self != null) {
  returnHash.put(provider.toString() + "_" +
                 self.getId(), self);
}
```

Divide and Conquer - getTopScoresForFriends()

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
    returnList =
getScoreInfoFromJson(callBackendRPC(args));
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
    returnList =
getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
    returnList =
getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
    ex.printStackTrace();
```

Divide and Conquer - getTopScoresForFriends()

```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
    returnList =
getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
    ex.printStackTrace();
}
```

Divide and Conquer - getTopScoresForFriends()

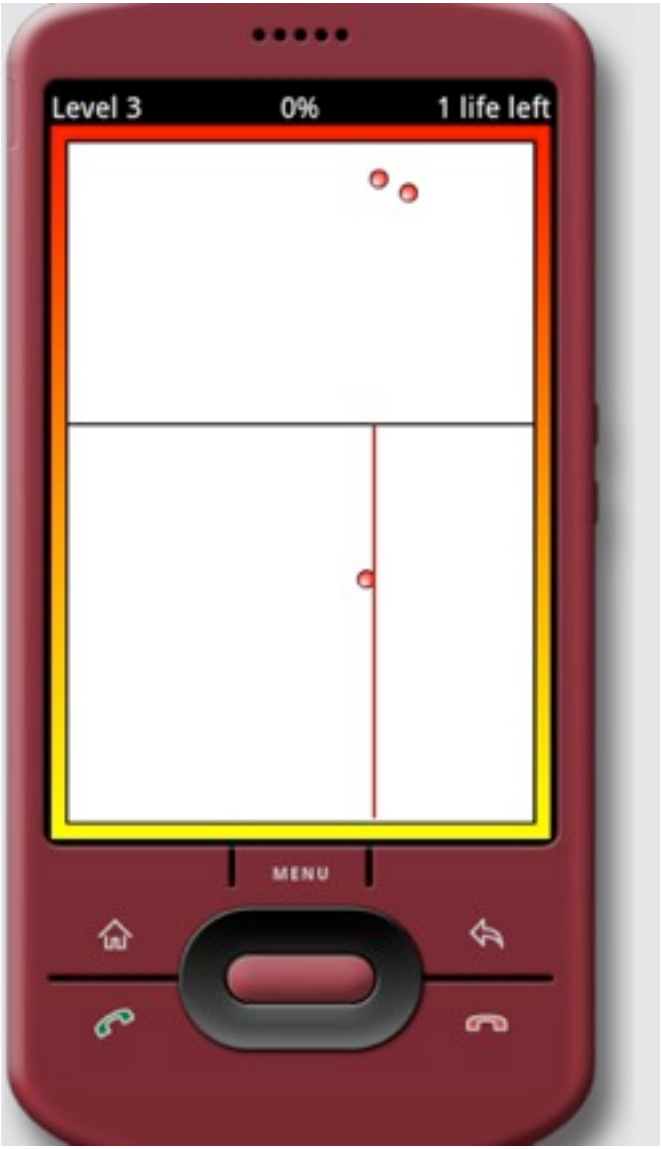
```
ArrayList<Hashtable<String,String>> returnList = null;
OpenSocialClient c = osClient;

List<Object> args = new ArrayList<Object>();
args.add("GetTopScoresForFriends");

while (friends.hasMoreElements()) {
    args.add(friends.nextElement());
}
try {
    returnList =
getScoreInfoFromJson(callBackendRPC(args));
} catch (Exception ex) {
    ex.printStackTrace();
}
return returnList;
```

Divide and Conquer - getPicturesForFriends()

Divide and Conquer - getPicturesForFriends()

```
Hashtable<String,String> userPictures = new Hashtable....
for (Hashtable<String,String> score : topScoreList) {
  String user = score.get("user");
  OpenSocialPerson person = friends.get(user);
  if (person.getField("thumbnailUrl") != null) {
    thumbnailUrl = person.getField("thumbnailUrl").
                   getStringValue();
    // ** download thumbnail here, save to file **
    userPictures.put(user, user + ".img");
  }
}
```

Divide and Conquer - When you die



Divide and Conquer - postUpdate()

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
args.add(provider.toString() + "_" + person.getId());
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
args.add(provider.toString() + "_" + person.getId());
args.add(level);
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
args.add(provider.toString() + "_" + person.getId());
args.add(level);
args.add(timeTaken);
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
args.add(provider.toString() + "_" + person.getId());
args.add(level);
args.add(timeTaken);
args.add(person.getDisplayName());
```

Divide and Conquer - postUpdate()

```
List<Object> args = new ArrayList<Object>();
args.add("SetScore");
args.add(provider.toString() + "_" + person.getId());
args.add(level);
args.add(timeTaken);
args.add(person.getDisplayName());
callBackendRPC(args);
```

OAuth Authorization

OAuth authorization

- Common standard used by many OpenSocial containers and other data providers
- Enables container-specific methods of authentication, authorization and account security
- Two types: 2-legged and 3-legged
- Caveats
  - Installed apps cannot identify themselves securely
  - Requires web browser for authorization flow - for both web apps and native apps

Types of OAuth

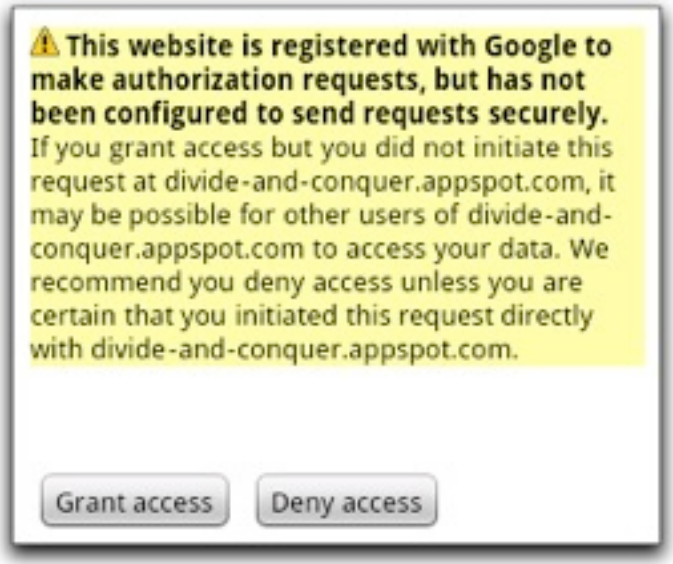
- 3-legged OAuth (the “standard” form)
  - End-user is using native app or web app and **hasn’t necessarily installed an app** by the same developer on the SNS
  - If necessary, end-user is asked to **authenticate to data provider**
  - End-user is sent to a **webpage to authorize access** to their data
  - After authorization is granted, end-user returns to the app
  - Access can sometimes be revoked by the end-user using an account settings/preferences page
- 2-legged OAuth
  - End-user **installs social network application**, which authorizes app developer to have access to their data
  - App developer is given a consumer key and consumer secret to **access the data of any user who has installed their app**, without any additional authorization
  - Access is revoked by the end-user when they remove the app

Supporting 3-legged OAuth and REST or RPC

- Google contacts & iGoogle
- Plaxo
- MySpace
- Netlog
- Partuza
- More...

Caveats: Installed app identification not secure

- Apps are typically identified by
  - Consumer Key
  - Consumer Secret
- Installed apps compromise the secret
- Impersonation possible
- Google's solution
  - Key: `anonymous`
  - Secret: `anonymous`
  - Identification: `xoauth_displayname`



Web apps

Native apps

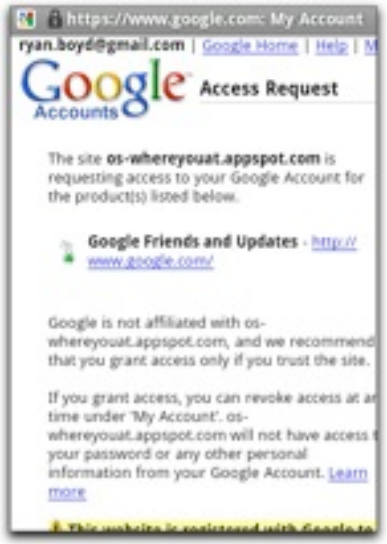
Google

Web apps

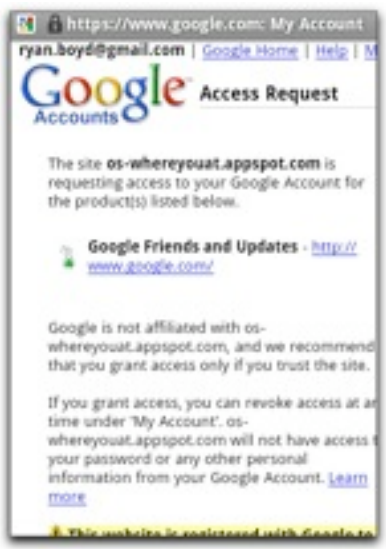
Native apps

Google

Web apps



Native apps

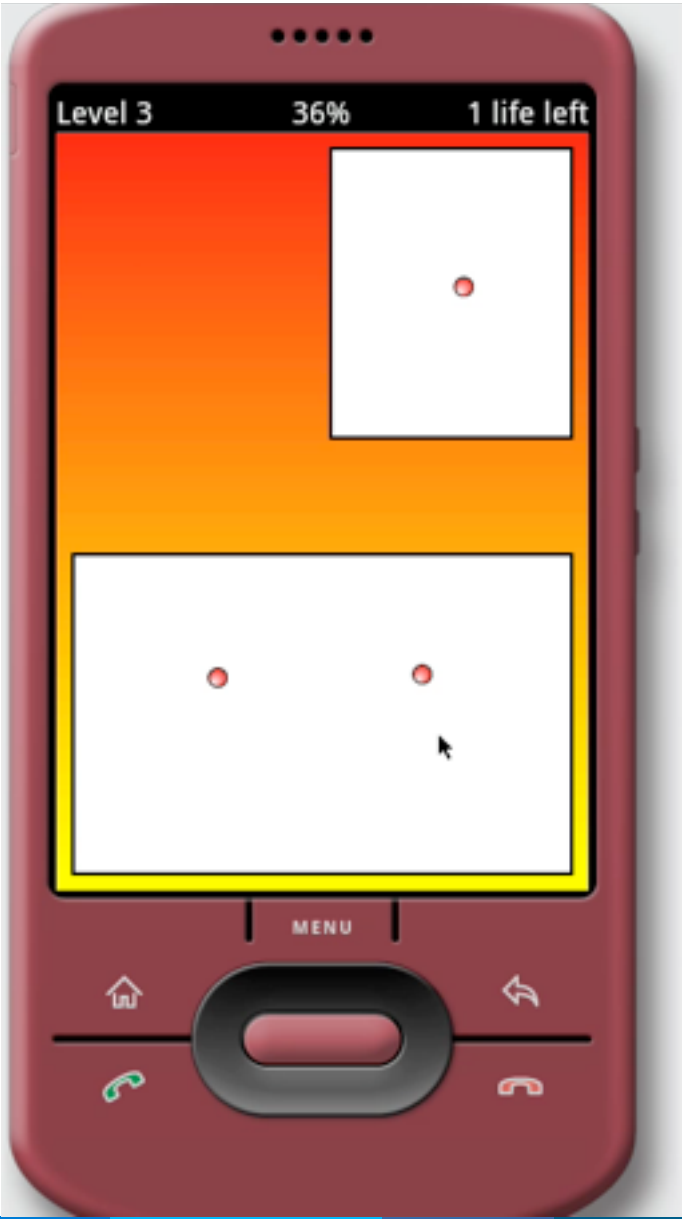


Google

Web apps

Native apps

OAuth Authorization – Example for native app



Divide and Conquer - Selecting a network

Divide and Conquer - onDialogClosed()

Divide and Conquer - onDialogClosed()

```
token = OpenSocialOAuthClient.getRequestToken(
            client, provider);
persistRequestToken(token, providerString);
String url = OpenSocialOAuthClient.
                  getAuthorizationUrl(provider, token,
                  ANDROID_SCHEME + "://opensocial");
```

Divide and Conquer - onDialogClosed()

```
token = OpenSocialOAuthClient.getRequestToken(
            client, provider);
persistRequestToken(token, providerString);
String url = OpenSocialOAuthClient.
                  getAuthorizationUrl(provider, token,
                  ANDROID_SCHEME + "://opensocial");


// Browse to webpage
Intent i = new Intent(Intent.ACTION_VIEW);
i.setData(Uri.parse(url));
getContext().startActivity(i);
```

http://www.flickr.com/photos/2litros/1332097012/

What about the iPhone?

Google 09 I/O

The iPhone ™

Google 09 I/O

iPhone is a trademark of Apple Inc

The iPhone ™

iPhone is a trademark of Apple Inc

The Future

Google I/O 09

What's Next?

- HTML5
  - Canvas
  - Database
  - App Cache
- W3C GeoLocation APIs
- Improved browser performance

Additional Credits for CC-licensed Photos

All referenced photos in the presentation are licensed under the Creative Commons. Here's some additional CC-licensed photos that were used in slides 8 and 9:

- http://www.flickr.com/photos/dandeluca/3235128655/
- http://www.flickr.com/photos/lexrex/63737829/
- http://www.flickr.com/photos/nileguide/3153216404/
- http://www.flickr.com/photos/vshioshvili/187988222
- http://www.flickr.com/photos/samdogs/2430430593/
- http://www.flickr.com/photos/allyrose18/3306534175
- http://www.flickr.com/photos/spine/546150140/
- http://www.flickr.com/photos/frenkieb/4615847

Q & A

Post your questions for this talk on Google Moderator:
http://tinyurl.com/mobile-social


Google™
09
I/O