
Google™
09
I/O

Using Google Data APIs and OAuth to Create an OpenSocial Gadget

Monsur Hossain && Eric Bidelman
5/28/2009

Post your questions for this talk on Google Moderator:
http://bit.ly/io-oauth-gadgets

Google 09 IO

GOAL

Build an iGoogle gadget that can read and write user data

ISSUES

Traditional JavaScript development practices are inadequate

ISSUES

- Cross-domain communication (Iframe hacks)

- Data format
  - `json-in-script` only offers read capabilities

```
<script src="http://www.google.com/calendar/feeds/
    developer@google.com/public/full
    ?alt=json-in-script&callback=handleSuccess">
</script>
```

- Authentication
  - AuthSub doesn't offer a good user experience
  - ClientLogin exposes user credentials

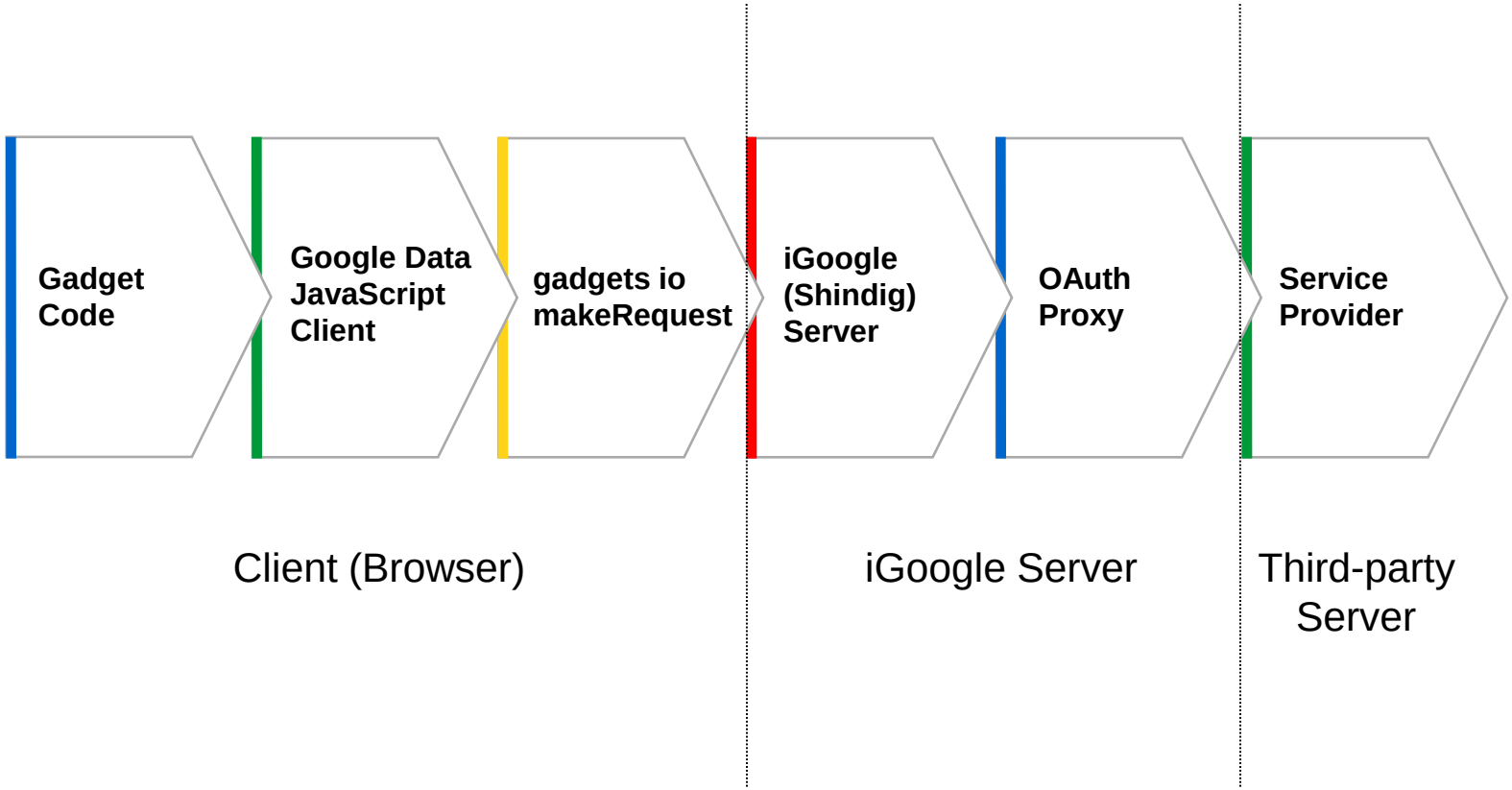
SOLUTION

Use the Google Data JavaScript Client with the OAuth Proxy

SOLUTION

- Authenticate users with OAuth
- Read and write data using Google Data JavaScript client

Technology Stack



OAuth

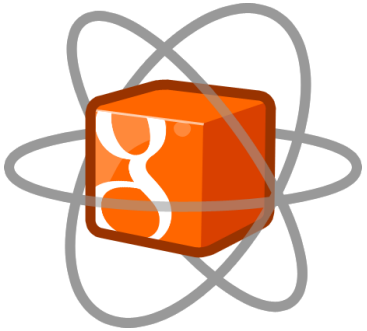
OAuth: Basics

- Open standard for sharing user's private data with another website or gadget.
- Eliminates the need to share passwords with various sites.
- All data requests are digitally signed.
- Similar to AuthSub's secure mode.
- More info: http://oauth.net

OAuth: The Players

USER
You

CONSUMER
The application trying to access the user's data (on behalf of the gadget).



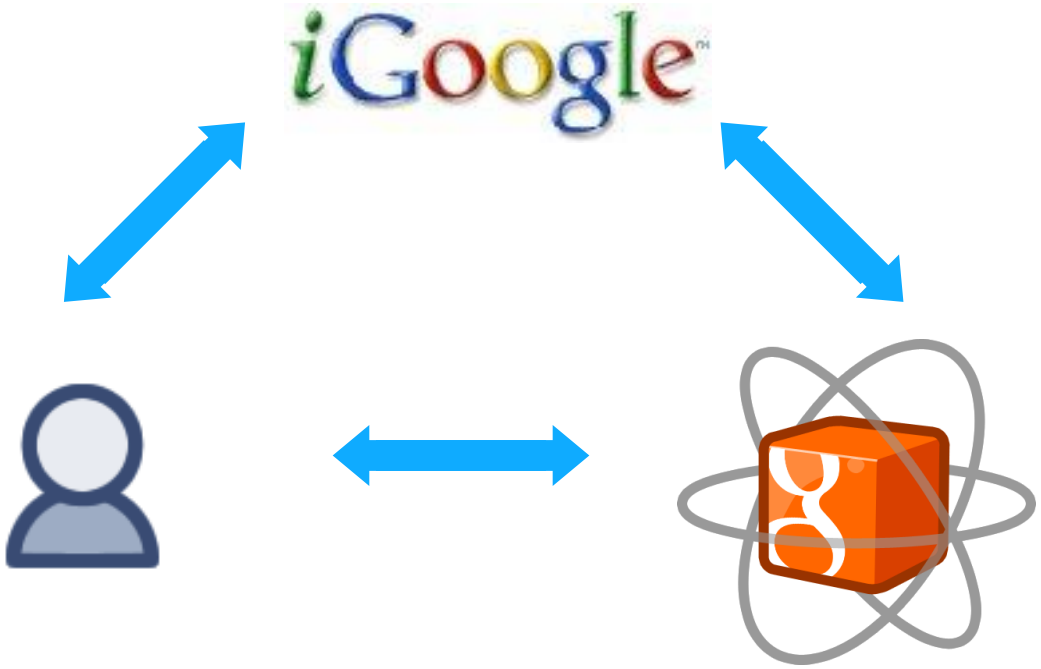
SERVICE PROVIDER
Keeper of user's data

OAuth URLs

Get Request Token

Authorize Request Token

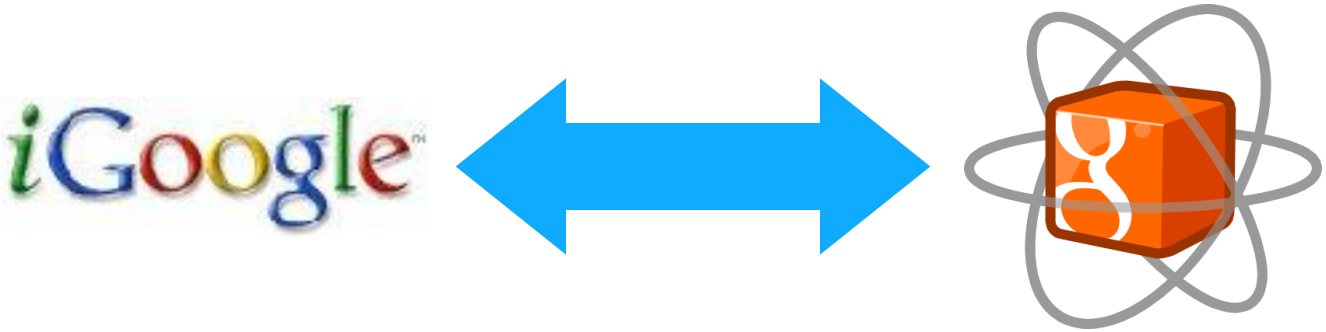
Get Access Token



OAuth URLs

Get Request Token | Authorize Request Token | Get Access Token

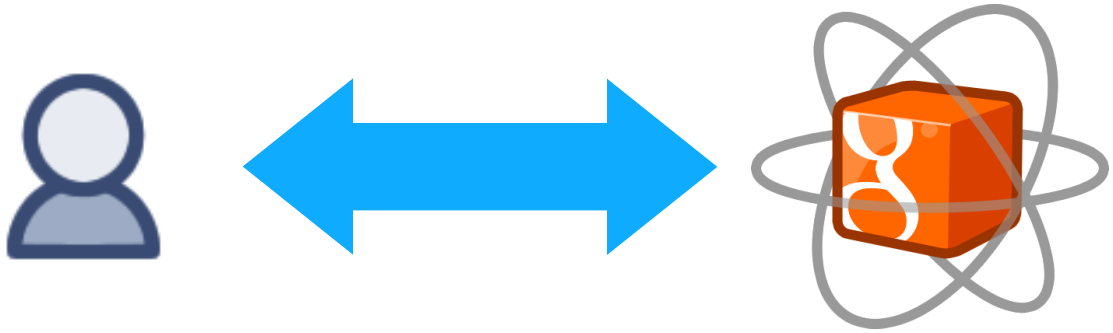
Behind-the-scenes request from consumer to service provider to obtain the initial request token.



OAuth URLs

Get Request Token	**Authorize Request Token**	Get Access Token

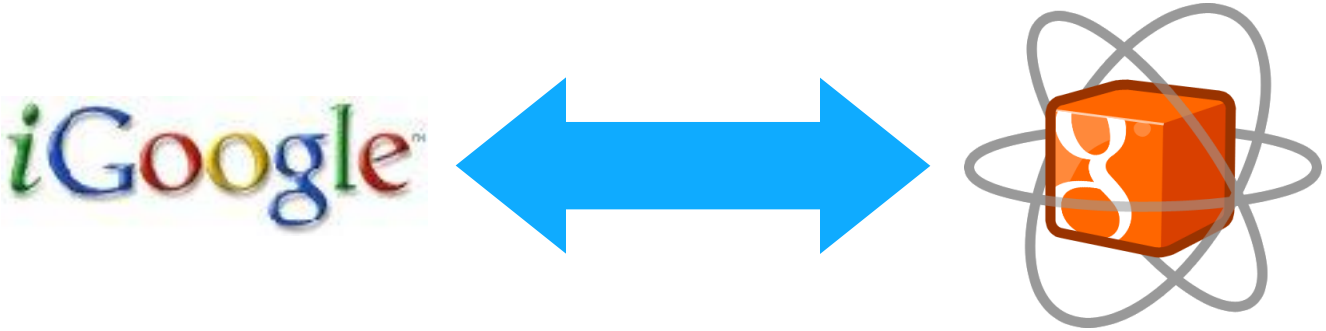
The url to which the user is redirect to authorize the request token.



OAuth URLs

Get Request Token | Authorize Request Token | **Get Access Token**

Behind-the-scenes request from consumer to service provider to exchange the authorized request token for an access token.



OAuth: Issues with JavaScript

- ✘ Crypto
  - There are no secrets in JavaScript!
- ✘ Cross-domain communication
  - Difficult in JavaScript
- ✘ Storage
  - Where to store tokens? Cookies?

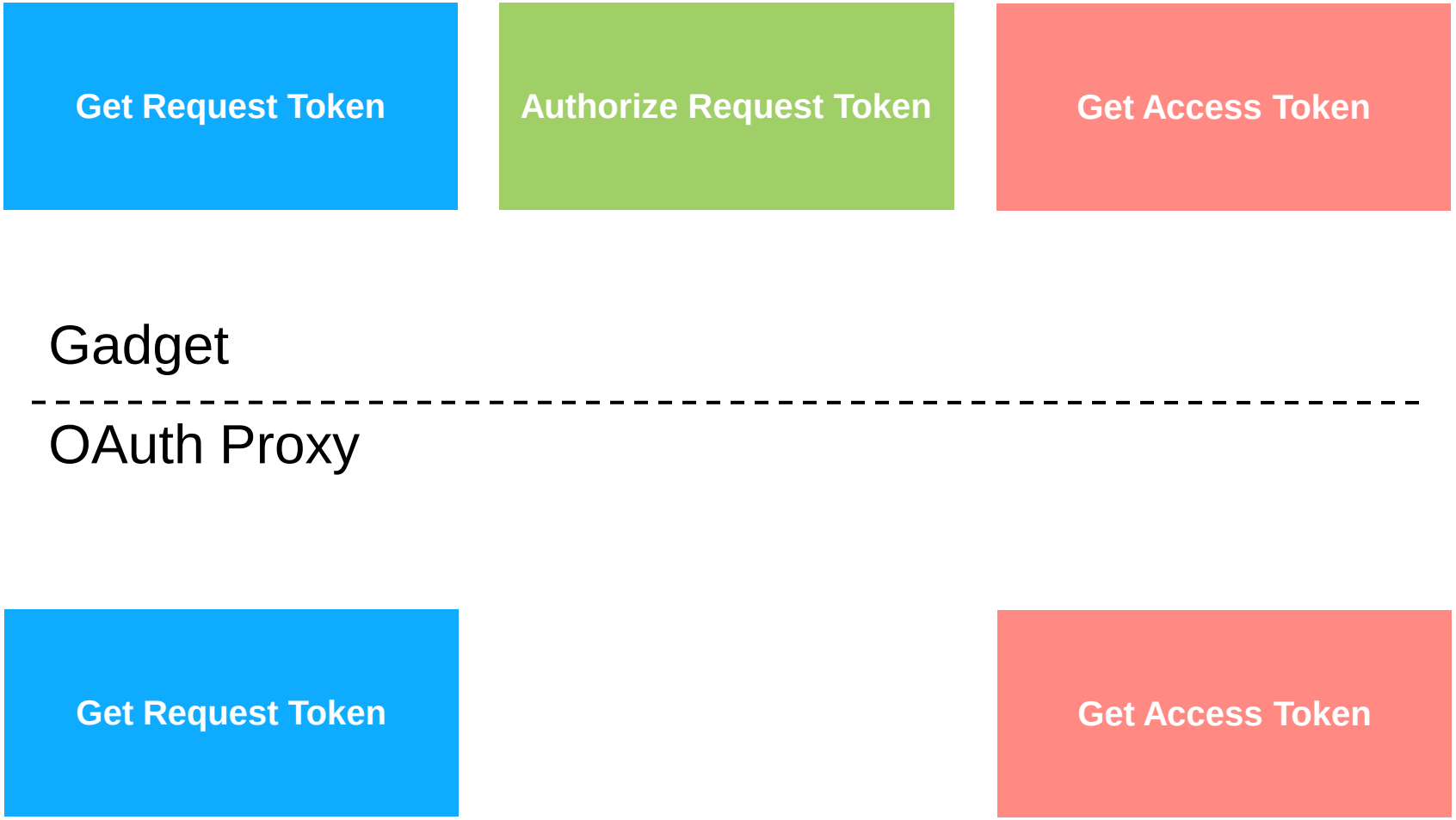
OAuth for Gadgets: The OAuth Proxy

- Open source, part of the iGoogle (Shindig) server
- Manages the “OAuth Dance” for gadgets

OAuth for Gadgets: The OAuth Proxy

- ✓ Crypto
  - Signs requests on behalf of the gadget
- ✓ Cross-domain communication
  - Lives on the same server as the gadget
- ✓ Storage
  - Stores token information server-side

OAuth URLs


Get Request Token
Authorize Request Token
Get Access Token
Gadget
OAuth Proxy
Get Request Token
Get Access Token

Building the Gadget: Adding OAuth Endpoints

```
<ModulePrefs>
...
<OAuth>
  <Service name="google">




  </Service>
</OAuth>
...
</ModulePrefs>
```

Building the Gadget: Adding OAuth Endpoints

```
<ModulePrefs>
...
<OAuth>
  <Service name="google">
    <Request method="GET“
      url="https://www.google.com/accounts
           /OAuthGetRequestToken
           ?scope=http://www.blogger.com/feeds" />
  </Service>
</OAuth>
...
</ModulePrefs>
```

Building the Gadget: Adding OAuth Endpoints

```
<ModulePrefs>
...
<OAuth>
  <Service name="google">
    <Request method="GET" url="https://www.google.com/accounts/OAuthGetRequestToken
                               ?scope=http://www.blogger.com/feeds" />
   <Authorization
    url="https://www.google.com/accounts
      /OAuthAuthorizeToken?oauth_callback=
      https://oauth-opensocial.googleusercontent.com/
      gadgets/oauthcallback" />

  </Service>
</OAuth>
...
</ModulePrefs>
```

Building the Gadget: Adding OAuth Endpoints

```
<ModulePrefs>
...
<OAuth>
  <Service name="google">
    <Request method="GET" url="https://www.google.com/accounts/OAuthGetRequestToken
                               ?scope=http://www.blogger.com/feeds" />
    <Authorization url="https://www.google.com/accounts/OAuthAuthorizeToken?
                        oauth_callback=https://oauth-opensocial.googleusercontent.com
                        /gadgets/oauthcallback" />
    <Access method="GET"
            url="https://www.google.com/accounts
                 /OAuthGetAccessToken" />
  </Service>
</OAuth>
...
</ModulePrefs>
```

Building the Gadget: Adding OAuth Endpoints

```
<ModulePrefs>
...
<OAuth>
  <Service name="google">
    <Request method="GET"
     url="https://www.google.com/accounts/OAuthGetRequestToken
          ?scope=http://www.blogger.com/feeds" />
    <Authorization url="https://www.google.com/accounts
          /OAuthAuthorizeToken?oauth_callback=
          https://oauth-opensocial.googleusercontent.com
          /gadgets/oauthcallback" />
    <Access method="GET" url="https://www.google.com/accounts
         /OAuthGetAccessToken" />
  </Service>
</OAuth>
...
</ModulePrefs>
```

Gadget Design

Building the Gadget: Starter Code

```
<Module>
  <ModulePrefs>
    <OAuth>…</OAuth>
    <Require feature="opensocial-0.8" />
    <Require feature="oauthpopup" />
  </ModulePrefs>
  <Content type="html">
    …
  </Content>
</Module>
```

Building the Gadget: Adding Content

```
<Content type="html">
 <div id="errors" style="display:none;">
   <!-- A good idea to have. -->
 </div>
 <!-- 1. Unauthenticated state. -->
 <div id="approval" style="display:none;">
  <a href="#" id="personalize">Sign in to Blogger</a>
 </div>
 <!-- 2. Pending state. Waiting for user to grant access. -->
 <div id="waiting" style="display:none;">
  <a href="#" id="approvalLink">I've approved access</a>
 </div>
 <!-- 3. Authenticated. Normal operating state. -->
 <div id="main" style="display:none;"></div>
 <!-- Another nice to have. -->
 <div id="loading">
  <h3>Loading...</h3>
  <p><img src="ajax-loader.gif"></p>
 </div>
</Content>
```

Building the Gadget: Adding Content

```
<Content type="html">
 <div id="errors" style="display:none;">
   <!-- A good idea to have. -->
 </div>
 <!-- 1. Unauthenticated state. -->
 <div id="approval" style="display:none;">
  <a href="#" id="personalize">Sign in to Blogger</a>
 </div>
 <!-- 2. Pending state. Waiting for user to grant access. -->
 <div id="waiting" style="display:none;">
  <a href="#" id="approvalLink">I've approved access</a>
 </div>
 <!-- 3. Authenticated. Normal operating state. -->
 <div id="main" style="display:none;"></div>
 <!-- Another nice to have. -->
 <div id="loading">
  <h3>Loading...</h3>
  <p><img src="ajax-loader.gif"></p>
 </div>
</Content>
```

Building the Gadget: Adding Content

```
<Content type="html">
 <div id="errors" style="display:none;">
   <!-- A good idea to have. -->
 </div>
 <!-- 1. Unauthenticated state. -->
 <div id="approval" style="display:none;">
  <a href="#" id="personalize">Sign in to Blogger</a>
 </div>
 <!-- 2. Pending state. Waiting for user to grant access. -->
 <div id="waiting" style="display:none;">
  <a href="#" id="approvalLink">I've approved access</a>
 </div>
 <!-- 3. Authenticated. Normal operating state. -->
 <div id="main" style="display:none;"></div>
 <!-- Another nice to have. -->
 <div id="loading">
  <h3>Loading...</h3>
  <p><img src="ajax-loader.gif"></p>
 </div>
</Content>
```

Building the Gadget: Adding Content

```
<Content type="html">
 <div id="errors" style="display:none;">
   <!-- A good idea to have. -->
 </div>
 <!-- 1. Unauthenticated state. -->
 <div id="approval" style="display:none;">
  <a href="#" id="personalize">Sign in to Blogger</a>
 </div>
 <!-- 2. Pending state. Waiting for user to grant access. -->
 <div id="waiting" style="display:none;">
  <a href="#" id="approvalLink">I've approved access</a>
 </div>
 <!-- 3. Authenticated. Normal operating state. -->
 <div id="main" style="display:none;"></div>
 <!-- Another nice to have. -->
 <div id="loading">
  <h3>Loading...</h3>
  <p><img src="ajax-loader.gif"></p>
 </div>
</Content>
```

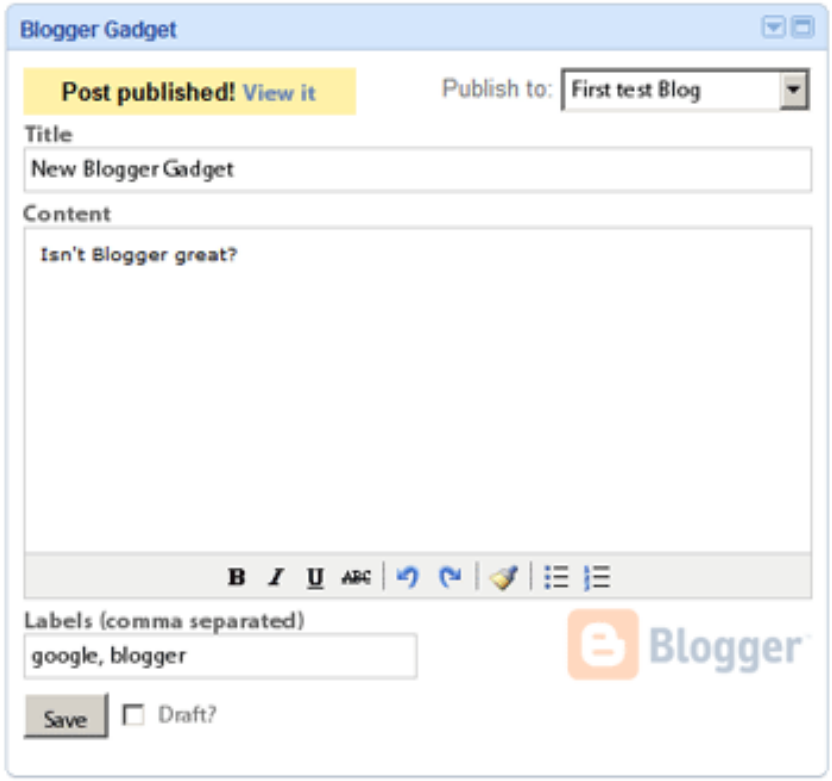
Building the Gadget: Adding Content

```
<Content type="html">
 <div id="errors" style="display:none;">
   <!-- A good idea to have. -->
 </div>
 <!-- 1. Unauthenticated state. -->
 <div id="approval" style="display:none;">
  <a href="#" id="personalize">Sign in to Blogger</a>
 </div>
 <!-- 2. Pending state. Waiting for user to grant access. -->
 <div id="waiting" style="display:none;">
  <a href="#" id="approvalLink">I've approved access</a>
 </div>
 <!-- 3. Authenticated. Normal operating state. -->
 <div id="main" style="display:none;"></div>
 <!-- Another nice to have. -->
 <div id="loading">
  <h3>Loading...</h3>
  <p><img src="ajax-loader.gif"></p>
 </div>
</Content>
```

Read / Write JS library

- Power of library becomes apparent with POSTs / PUTs
- DEMO: Blogger gadget

http://bit.ly/blogger_gadget



What about loading data?

- `gadgets.io.makeRequest(url, callback, params)` is responsible for all communication with third-party servers.
  - `url`: url to make the request to
  - `callback`: function to call once response is received
  - `params`: (optional) other parameters related to the request (i.e. headers, http methods, etc)
- Response is returned to the callback function as a json object.
- OAuth Authorization header is automatically added to the request for data

Gadget Design - OpenSocial JS Setup

```
<script type="text/javascript">

function initGadget() {
  fetchData();
}

gadgets.util.registerOnLoadHandler(initGadget);

</script>
```

Gadget Design - OpenSocial `fetchData()`

```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    if (response.oauthApprovalUrl) {
      createApprovalPopupHandler(response.oauthApprovalUrl);
      showDiv('approval');
    } else if (response.feed) {
      showResults(response);
      showDiv('main');
    } else {
      $('errors').innerHTML = 'Yikes!! ' + response.oauthErrorText;
      showDiv('errors');
    }
  };
  var params = {
    gadgets.io.RequestParameters.OAUTH_SERVICE_NAME: 'google‘,
    params[gadgets.io.RequestParameters.METHOD: gadgets.io.MethodType.GET,
    params[gadgets.io.RequestParameters.CONTENT_TYPE: gadgets.io.ContentType.JSON,
    params[gadgets.io.RequestParameters.AUTHORIZATION: gadgets.io.AuthorizationType.OAUTH,
    params[gadgets.io.RequestParameters.OAUTH_USE_TOKEN: 'always'
  };
  var url = 'http://www.blogger.com/feeds/default/blogs';
  gadgets.io.makeRequest(url, callback , params);
}
```

Gadget Design - OpenSocial `fetchData()`

```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    if (response.oauthApprovalUrl) {
      createApprovalPopupHandler(response.oauthApprovalUrl);
      showDiv('approval');
    }
    else if (response.feed) {
      showResults(response);
      showDiv('main');
    }
    else {
      $('errors').innerHTML = response.oauthErrorText;
      showDiv('errors');
    }
  };
}
```

Gadget Design - OpenSocial `fetchData()`

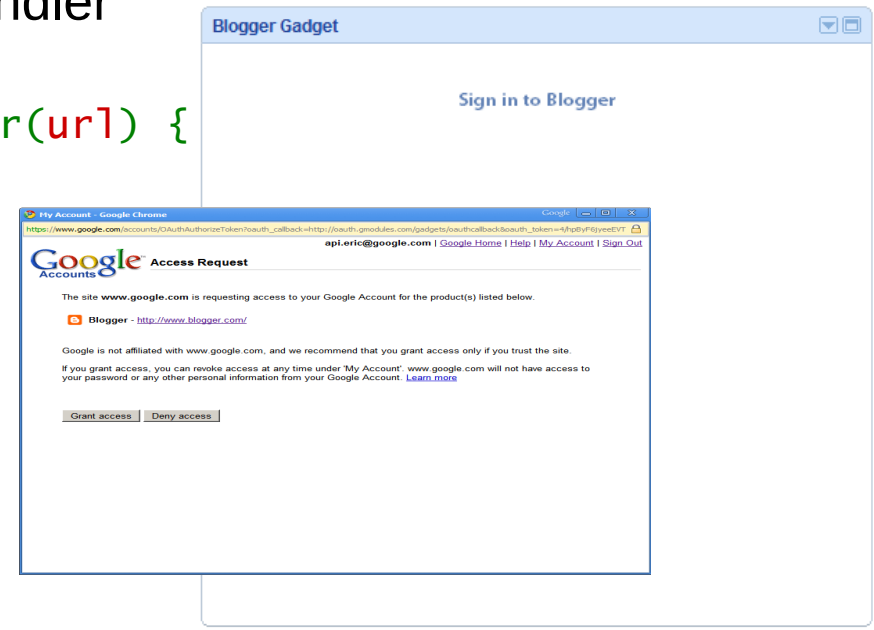
```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    ...
  };

  var rp = gadgets.io.RequestParameters;
  var params = {
    rp.OAUTH_SERVICE_NAME: 'google',
    rp.METHOD: gadgets.io.MethodType.GET,
    rp.CONTENT_TYPE: gadgets.io.ContentType.JSON,
    rp.AUTHORIZATION: gadgets.io.AuthorizationType.OAUTH,
    rp.OAUTH_USE_TOKEN: 'always'
  };
}
```

Gadget Design - OpenSocial `fetchData()`

```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    ...
  };

  var rp = gadgets.io.RequestParameters;
  var params = {
    ...
  };

  var url = 'http://www.blogger.com/feeds/default/blogs' +
            '?alt=json';
  gadgets.io.makeRequest(url, callback, params);
}
```

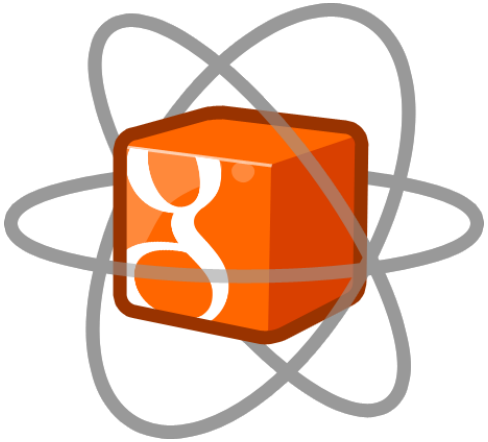
Gadget Design – Popup handler

```
function createApprovalPopupHandler(url) {
  var onOpen = function() {
    showDiv('waiting');
  };

  var onClose = function() {
    showDiv('loading');
    fetchData();
  };

  var popup = new gadgets.oauth.Popup(url, 'height=600,width=800',
                                      onOpen, onClose);

  $('personalize').onclick = popup.createOpenerOnClick();
  $('approvalLink').onclick = popup.createApprovedOnClick();
}
```



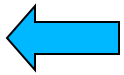
Making Things Easier: Google Data JS Client

- Handles the details of loading data from Google
- Supports read and write operations
- Supports multiple authentication methods, including AuthSub, ClientLogin, and the OAuth Proxy
- Works cross-domain using iframes
- Wraps the `gadgets.io.makeRequest()` function (in gadget environments)
- More info: http://bit.ly/js-client



Gadget Design - Google Data JS Setup

```
<script type="text/javascript“ src="http://www.google.com/jsapi"></script>
<script type="text/javascript">
var blogger = null;

function initGadget() {
  google.load('gdata', '1.x', {packages: ['blogger', 'calendar']});

  google.setOnLoadCallback(function () {
    var appName = 'google-ExampleGadget-v1.0';
    blogger = new google.gdata.blogger.BloggerService(appName);
    blogger.useOAuth('google');
    // blogger.setGadgetsAuthentication('OAUTH', {'hd': 'example.com'});
    fetchData();
  });
}

gadgets.util.registerOnLoadHandler(initGadget);
</script>
```



Gadget Design - Google Data `fetchData()`

```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    ...
  };
  var rp = gadgets.io.RequestParameters;
  var params = {
    rp.OAUTH_SERVICE_NAME: 'google',
    rp.METHOD: gadgets.io.MethodType.GET,
    rp.CONTENT_TYPE: gadgets.io.ContentType.JSON,
    rp.AUTHORIZATION: gadgets.io.AuthorizationType.OAUTH,
    rp.OAUTH_USE_TOKEN: 'always'
  };
  var url = 'http://www.blogger.com/feeds/default/blogs';
  blogger.getBlogFeed(url, callback, callback);
}
```

Blogger Gadget – Creating data

```
blogger.getBlogFeed('http://www.blogger.com/feeds/default/blogs', function(resp) {
  postData(resp.feed.getEntries()[0]);
}, handleError);

function postData(blog) {
  var newEntry = new google.gdata.blogger.BlogPostEntry({
    title: { type: 'text', text: 'New Blogger Gadget' },
    content: { type: 'text', text: "Isn't Blogger great?" },
    categories: [
      {scheme: 'http://www.blogger.com/atom/ns#', term: 'io'},
    ],
    control: { draft: {value: google.gdata.Draft.VALUE_NO} }
  });
  blogger.insertEntry(blog.getEntryPostLink().getHref(),
                      newEntry, handleInsert, handleError);
}
```

- Raw OS calls would require more code.
- Helper methods transform JSON → Atom XML (req & resp)

Blogger Gadget – callback handlers

```
function handleInsert(entryRoot) {
  var href = entryRoot.entry.getHtmlLink().getHref();
  $('main').innerHTML = '<a href="' + href +
                        '" target="new">View post</a>';
}

function handleError(e) {
  var msg = e.cause ? e.cause.statusText + ': ' : '';
  msg += e.message;
  alert('Error: ' + msg);
}
```

Behind the scenes

Dissecting a gadget's request

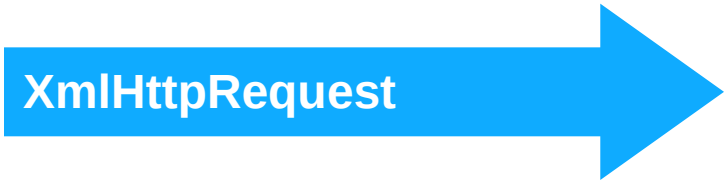
Google Data JavaScript Client

```
blogger.getBlogFeed(url, successHandler, errorHandler);
```

Dissecting a gadget's request

gadgets.io.makeRequest

```
gadgets.io.makeRequest(url,modified_callback, {
  rp.OAUTH_SERVICE_NAME: 'google',
  rp.METHOD: gadgets.io.MethodType.GET,
  rp.CONTENT_TYPE: gadgets.io.ContentType.JSON,
  rp.AUTHORIZATION: gadgets.io.AuthorizationType.OAUTH,
  rp.OAUTH_USE_TOKEN: 'always'
});
```



Dissecting a gadget's request

XmlHttpRequest (HTTP POST)

`http://XXX.gmodules.com/gadgets/makeRequest`

Url	Url to make the request to. i.e. http://blogger.com/feeds/…
httpMethod	GET, POST, etc…
postData	Data to send to the url
authz	oauth
gadget	url of the gadget
OAUTH_SERVICE_NAME	google

Dissecting `gadgets.io.makeRequest()`


```
{"URL OF REQUEST" : {
"oauthState" : "SOME LONG TOKEN",
"rc" : 200,
   "body" : "",
   "oauthApprovalUrl" :
        "https://www.google.com/accounts/
         OAuthAuthorizeToken?oauth_callback=
         http://oauth.gmodules.com/gadgets/
         oauthcallback&oauth_token=TOKEN"
   }
}
```

Dissecting `gadgets.io.makeRequest()`

Success from Server

```
{"URL OF REQUEST" : {
"oauthState" : "SOME LONG TOKEN",
"rc" : 200,
"body" : ”
   { \”version\" : \"1.0\",
   \"encoding\" : \"UTF-8\",
   \"feed\" : \"JSON OF FEED\” }
"}
```

Demos!

Demos

Base Data API (unauthenticated)

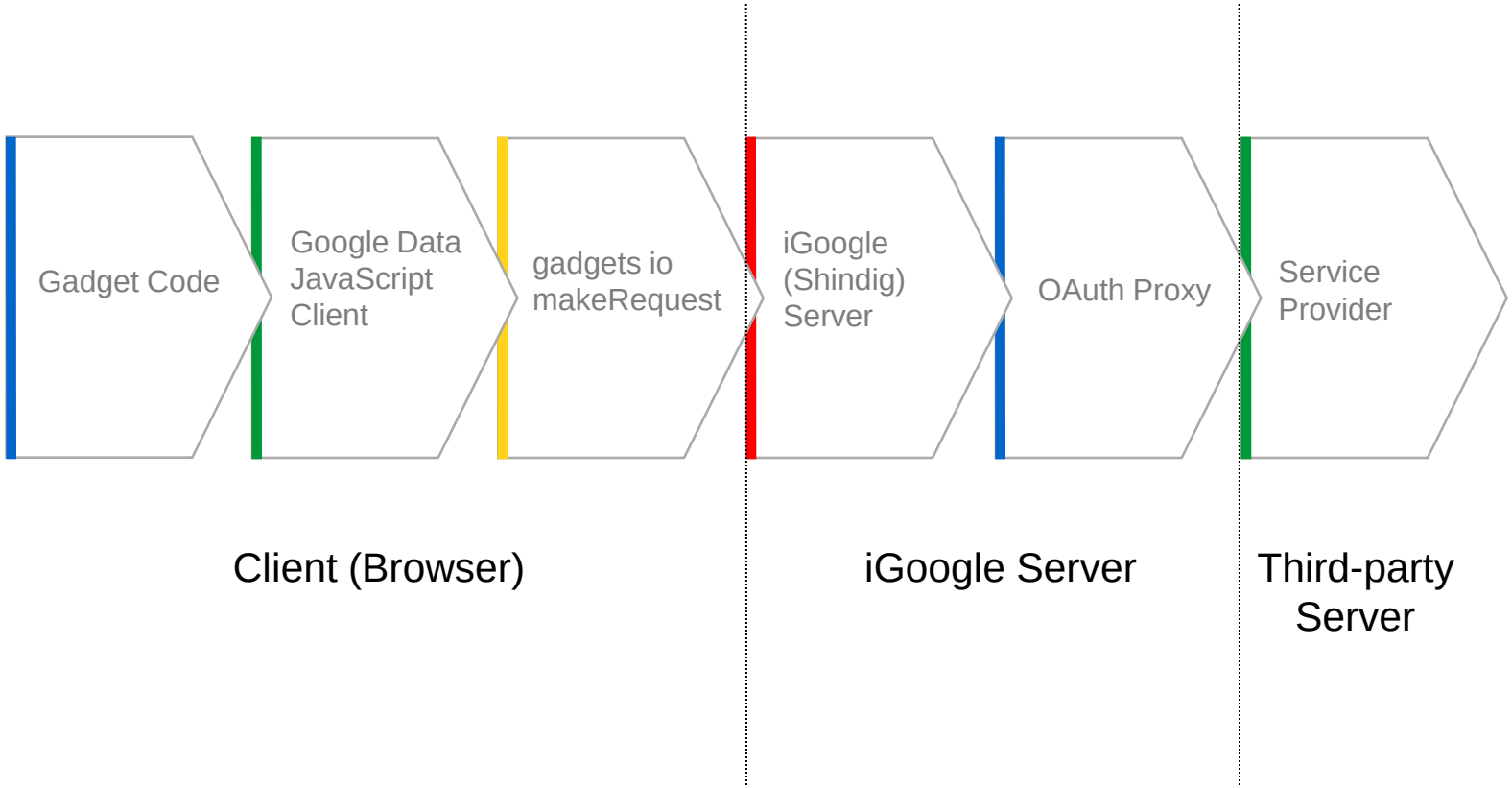
- http://bit.ly/base-gadget

Calendar Data API

- http://bit.ly/cal-gadget

Conclusion

Conclusion - Technology Stack



Conclusion - Resources

- Using the Google Data JS client library: http://bit.ly/js-client-docs
- 'Writing OAuth Gadgets': http://bit.ly/oauth-gadgets
- Additional code samples: http://bit.ly/code-samples

Q & A

Post your questions for this talk on Google Moderator:
http://bit.ly/io-oauth-gadgets


Google™
09
I/O

Gadget Design – Utility methods

```
function showResults(feedRoot) {
  var entries = feedRoot.feed.getEntries();
  var html = [];
  for (var i = 0, entry; entry = entries[i]; ++i) {
    var title = entry.getTitle().getText();
    html.push(title + '<br>');
  }
  $('main').innerHTML = html.join('');
}
```

Gadget Design – Google Data `fetchData()`

```
function fetchData() {
  showDiv('loading');
  var callback = function(response) {
    if (response.oauthApprovalUrl) {
      createApprovalPopupHandler(response.oauthApprovalUrl);
      showDiv('approval');
    } else if (response.feed) {
      showResults(response);
      showDiv('main');
    } else {
      $('errors').innerHTML = 'Yikes!! ' + response.oauthErrorText;
      showDiv('errors');
    }
  };
  var url = 'http://www.google.com/calendar/feeds/default/allcalendars/full';
  calendar.getAllCalendarsFeed(url, callback, callback);
}
```

- Notice the dropped **`?alt=json`** on the feed URI
- What is the Google library doing?....calling **`gadgets.io.makeRequest()`** behind the scenes.

Gadget Design – Utility methods

```
function $(id) {
  return document.getElementById(id);
}

function showDiv(id) {
  var sections = ['main', 'approval', 'waiting', 'loading', 'errors'];
  for (var i = 0, section; section = sections[i]; ++i) {
    $(section).style.display = section === id ? 'block' : 'none';
  }
}
```