Permissions we can access

User Data

Read Permissions - User Attributes

- email
- public_profile
- read_custom_friendlists
- user_about_me
- user_birthday
- user_education_history
- user_friends
- user_hometown
- user_location
- user_relationship_details
- user_relationships
- user_religion_politics
- user_work_history

Read Permissions - User Activity

user_actions.books
user_actions.fitness
user_actions.music
user_actions.news
user_actions.video
user_games_activity
user_likes
user_photos
user_posts
user_tagged_places
user_videos
user_website

Read Permissions - User Events and Groups

user_events
user_managed_groups

Write Permissions

publish_actions
rsvp_event

Pages and Business Assets

Read Permissions

ads_management
ads_read
business_management
read_audience_network_insights
read_insights
manage_pages
pages_manage_cta
pages_manage_instant_articles
pages_show_list
read_page_mailboxes

Write Permissions

publish_pages

Messenger Platform Permissions

pages_messaging
pages_messaging_payments
pages_messaging_phone_number
pages_messaging_subscriptions

Instagram Platform Permissions

instagram_basic
instagram_manage_comments
instagram_manage_insights

Deprecated Permissions

manage_notifications
read_mailbox
read_stream
user_actions:{app_namespace}
user_groups
user_status

The problem we face:

user_friends does not grant permissions to access someones total friend list. Just the people that use the same app and have enabled other friends to see that they are using the app. THIS SAME PERMISSION DOES NOT ALLOW TO INVITE FRIENDS TO THE APP NEITHER.

Read_custom_friendlists is the permission that allows us to see CUSTOM friend lists NAMES not who is in it. We can't see their friends.

The facebook docs say they have removed being able to see friends. The only type of app that they allow a user to invite their friends to are games. They say "If your app is categorized as a game and you have a presence on Facebook Canvas, you can use the Invitable Friends API to render a custom invites dialog within your game. " so we are out of luck.

-

1. Enter Your Redirect URL

In the App Dashboard, choose your app and go to **Products** > **Facebook Login** > **Settings**. Under the **Client OAuth Settings**, enter your redirect URL in the **Valid OAuth redirect URIs** field for successful authorization.

2. Check Login Status

The first step when loading your webpage is figuring out if a person is already logged into your app with Facebook Login. You start that process with a call to `FB.getLoginStatus`. That function will trigger a call to Facebook to get the login status and call your callback function with the results.

Taken from the sample code below, here's some of the code that's run during page load to check a person's login status:

```
FB.getLoginStatus(function(response) {
   statusChangeCallback(response);
});
```

The `response` object that is provided to your callback contains a number of fields:

```
{
   status: 'connected',
   authResponse: {
      accessToken: '...',
      expiresIn:'...',
      signedRequest:'...',
      userID:'...'
   }
}
```

`status` specifies the login status of the person using the app. The `status` can be one of the following:

- `connected` - The person is logged into Facebook, and has logged into your app.
- `not_authorized` - The person is logged into Facebook, but has not logged into your app.
- `unknown` - The person is not logged into Facebook, so you don't know if they've logged into your app. Or FB.logout() was called before, and therefore, it cannot connect to Facebook.
- `authResponse` is included if the status is `connected` and is made up of the following:
  - `accessToken` - Contains an access token for the person using the app.
  - `expiresIn` - Indicates the UNIX time when the token expires and needs to be renewed.
  - `signedRequest` - A signed parameter that contains information about the person using the app.
  - `userID` - The ID of the person using the app.

Once your app knows the login status of the person using it, it can do one of the following:

- If the person is logged into Facebook and your app, redirect them to your app's logged in experience.
- If the person isn't logged into your app or isn't logged into Facebook, prompt them with the Login dialog with `FB.login()` or show them the Login Button.

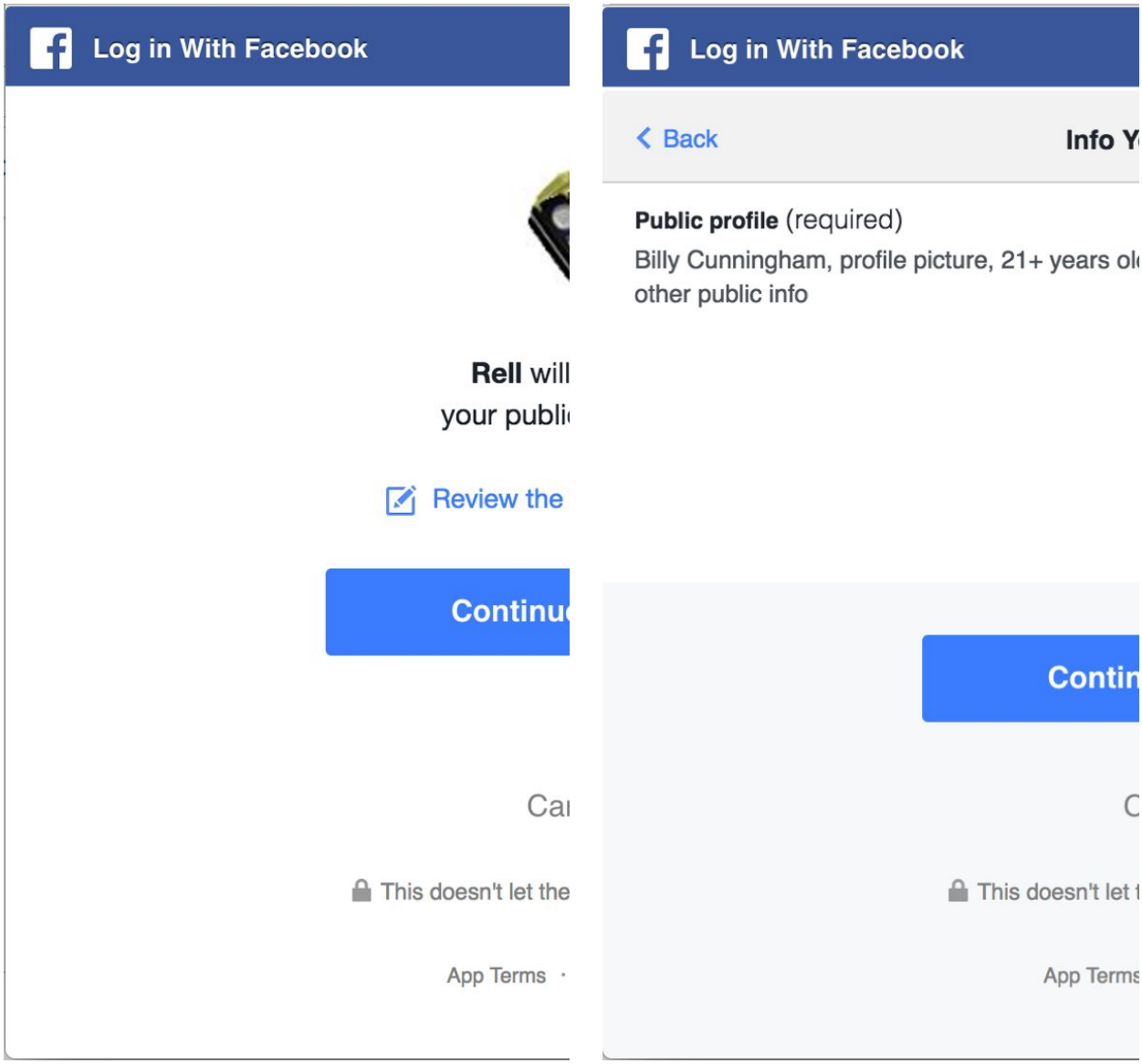
3. Log People In

If people using your app aren't logged into your app or not logged into Facebook, you can use the Login dialog to prompt them to do both. Various versions of the dialog are shown below.

If they aren't logged into Facebook, they'll first be prompted to log in, then move on to logging in to your app. The JavaScript SDK automatically detects this, so you don't need to do anything extra to enable this behavior.

There are two ways to log someone in:

- Use the Login Button.
- Use `FB.login()` from the JavaScript SDK.



A. Log People in with the Login Button

Including the Login Button on your page is easy. For information on customizing the Login Button, see Login Button. To get the code for the basic button, enter the values you want in the following configurator and click **Get Code**.

Plugin Configurator

Width

Maximum Rows of Photos

Button Size

large

Button Text

continue_with

Show Friends' Faces

Enable Logout Button

Include name and profile picture when user is signed into Facebook

Get Code

Note that the example at the end of this topic uses the `onlogin` attribute on the button to set up a JavaScript callback that checks the login status to see if the person logged in successfully:

```
<fb:login-button scope="public_profile,email" onlogin="checkLoginState();">
</fb:login-button>
```

This is the callback. It calls `FB.getLoginStatus()` to get the most recent login state. (The `statusChangeCallback()` function is part of the example that processes the response.)

```
function checkLoginState() {
  FB.getLoginStatus(function(response) {
    statusChangeCallback(response);
  });
}
```

B. Log People in with the Login Dialog from the JavaScript SDK

For apps that want to use their own button, you can invoke the Login Dialog with a simple call to `FB.login()`:

```
FB.login(function(response){
  // Handle the response object, like in statusChangeCallback() in our demo
```

```
  // code.
});
```

As noted in the reference docs for this function, it results in a pop-up window showing the Login dialog, and therefore should only be invoked as a result of someone clicking an HTML button (so that the pop-up isn't blocked by browsers).

There is an optional `scope` parameter that can be passed along with the function call that is a comma-separated list of permissions to request from the person using the app. Here's how you would call `FB.login()` with the same `scope`as the Login Button we used above. In this case, it would ask for a person's email address and a list of friends who also use the app:

```
FB.login(function(response) {
  // handle the response
}, {scope: 'public_profile,email'});
```

Handle Login Dialog Response

At this point in the login flow, your app displays the login dialog, which gives people the choice of whether to cancel or to enable the app to access their data.

Whatever choice people make, the browser returns to the app and response data indicating whether they're connected or if they cancelled is sent to your app. When your app uses the JavaScript SDK, it returns an `authResponse` object to the callback specified when you made the `FB.login()` call:

This response can be detected and handled within the `FB.login` call, like this:

```
FB.login(function(response) {
  if (response.status === 'connected') {
    // Logged into your app and Facebook.
  } else {
    // The person is not logged into this app or we are unable to tell.
```

```
  }
});
```

4. Log People Out

You can log people out of your app by attaching the JavaScript SDK function `FB.logout` to a button or a link, as follows:

```
FB.logout(function(response) {
  // Person is now logged out
});
```

Note: This function call may also log the person out of Facebook.

Consider the 3 scenarios below:

1. A person logs into Facebook, then logs into your app. Upon logging out from your app, the person is still logged into Facebook.
2. A person logs into your app and into Facebook as part of your app's login flow. Upon logging out from your app, the user is also logged out of Facebook.
3. A person logs into another app and into Facebook as part of the other app's login flow, then logs into your app. Upon logging out from either app, the user is logged out of Facebook.

Additionally, logging out is not the same as revoking login permission (i.e., removing previously granted authentication), which can be performed separately. Because of this your app should be built in such a way that it doesn't automatically force people who have logged out back to the Login dialog.

Full Code Example

This code will load and initialize the JavaScript SDK in your HTML page. Use your app ID where indicated.

```
<!DOCTYPE html>
<html>
```

```
<head>
<title>Facebook Login JavaScript Example</title>
<meta charset="UTF-8">
</head>
<body>
<script>
  // This is called with the results from from FB.getLoginStatus().
  function statusChangeCallback(response) {
    console.log('statusChangeCallback');
    console.log(response);
    // The response object is returned with a status field that lets the
    // app know the current login status of the person.
    // Full docs on the response object can be found in the documentation
    // for FB.getLoginStatus().
    if (response.status === 'connected') {
      // Logged into your app and Facebook.
      testAPI();
    } else {
      // The person is not logged into your app or we are unable to tell.
      document.getElementById('status').innerHTML = 'Please log ' +
        'into this app.';
    }
  }

  // This function is called when someone finishes with the Login
  // Button.  See the onlogin handler attached to it in the sample
  // code below.
  function checkLoginState() {
    FB.getLoginStatus(function(response) {
      statusChangeCallback(response);
    });
  }

  window.fbAsyncInit = function() {
    FB.init({
      appId      : '{your-app-id}',
```

```
    cookie     : true,  // enable cookies to allow the server to access
                        // the session
    xfbml      : true,  // parse social plugins on this page
    version    : 'v2.8' // use graph api version 2.8
  });

  // Now that we've initialized the JavaScript SDK, we call
  // FB.getLoginStatus().  This function gets the state of the
  // person visiting this page and can return one of three states to
  // the callback you provide.  They can be:
  //
  // 1. Logged into your app ('connected')
  // 2. Logged into Facebook, but not your app ('not_authorized')
  // 3. Not logged into Facebook and can't tell if they are logged into
  //    your app or not.
  //
  // These three cases are handled in the callback function.

  FB.getLoginStatus(function(response) {
    statusChangeCallback(response);
  });

 };

 // Load the SDK asynchronously
 (function(d, s, id) {
   var js, fjs = d.getElementsByTagName(s)[0];
   if (d.getElementById(id)) return;
   js = d.createElement(s); js.id = id;
   js.src = "https://connect.facebook.net/en_US/sdk.js";
   fjs.parentNode.insertBefore(js, fjs);
 }(document, 'script', 'facebook-jssdk'));

 // Here we run a very simple test of the Graph API after login is
 // successful.  See statusChangeCallback() for when this call is made.
 function testAPI() {
```

```
    console.log('Welcome!  Fetching your information.... ');
    FB.api('/me', function(response) {
      console.log('Successful login for: ' + response.name);
      document.getElementById('status').innerHTML =
        'Thanks for logging in, ' + response.name + '!';
    });
  }
</script>

<!--
  Below we include the Login Button social plugin. This button uses
  the JavaScript SDK to present a graphical Login button that triggers
  the FB.login() function when clicked.
-->

<fb:login-button scope="public_profile,email" onlogin="checkLoginState();">
</fb:login-button>

<div id="status">
</div>

</body>
</html>
```

Now you can test your app by going to the URL where you uploaded this HTML. Open your JavaScript console, and you'll see the `testAPI()` function display a message with your name in the console log.

Congratulations, at this stage you've actually built a really basic page with Facebook Login. You can use this as the starting point for your own app, but it will be useful to read on and understand what is happening in the code above.

TLDR:

We can’t have access to a user’s friends list. Not even to invite their friends to use the app because we are not a game app. Best thing we can do is have the names of custom friends lists that the user has made and that is it. No real access to people at all.