fgl.com

Implementation Guide: FGL SDK for HTML5

The FGL SDK for HTML5 allows you to easily tap into all of the value that FGL has to offer for your HTML5 game.

Version 1.6 August 7th 2014

Obtaining the SDK

Download the SDK package from FGL.com/html5

Package contents:

sample-game
: This directory contains a fully functional sample game. See `index.html` and `game.js` for example integration.

scaling-reference
: View `landscape.html` and `portrait.html` for a reference of how your game should handle scaling. It's a requirement of the FGL SDK that your game scales properly.

fgl.js
: The FGL HTML5 SDK - include this in your game

Implementation Guide.pdf
: This Guide

SDK Documentation.pdf
: Contains a technical documentation of the SDK functionality

Initial Setup

First you need to include the `fgl.js` file in your project. It can be included in the same way as a normal JS file - include it in your project before any game setup or logic.

```
<script type="text/javascript" src="fgl.js"></script>
```

The title of the index.html file should be set to your game's title; it will be used when your game is submitted to FGL.

index.html:

```
<html>
<head>
    <title>My Game Title</title>
</head>
<body>
    <script type="text/javascript" src="fgl.js"></script>
    <script type="text/javascript" src="your-game.js"></script>

    <canvas id="game"></canvas>

    <script type="text/javascript">
        myGame.start();
    </script>
</body>
</html>
```

Handling Screen Resolution

Your game must fill to fit all of the available screen area. You can see a reference implementation in the `scaling-reference` directory which is included with the SDK.

Black bars at the top and bottom, or left and right, can be added when your game needs to retain aspect ratio to remain playable.

More information on scaling can be found at the following links:

1. https://github.com/fglmobile/html5-game-scaling
2. https://mpstatic.com/html5/static/scaling-reference/landscape-guide.html

Displaying Ads

You can call a function to display a takeover ad at any time. The advert will take focus of your game and will display a close button for the user.

We can control which ad networks run on your game, and we are constantly optimizing the inventory on your behalf to get the best possible return. The ads are player-friendly and can be immediately dismissed.

Call the following function when appropriate, for example when the user reaches the main menu, or between levels on your game:

```
fgl.showAd();
```

Cross Promotion

To take advantage of our powerful cross-promotion system, add a *"More Games"* button to your main menu.

When tapped, this button will display an overlay which will cross promote to other games in our network, allowing your game to get significant promotion at launch when it needs it most.

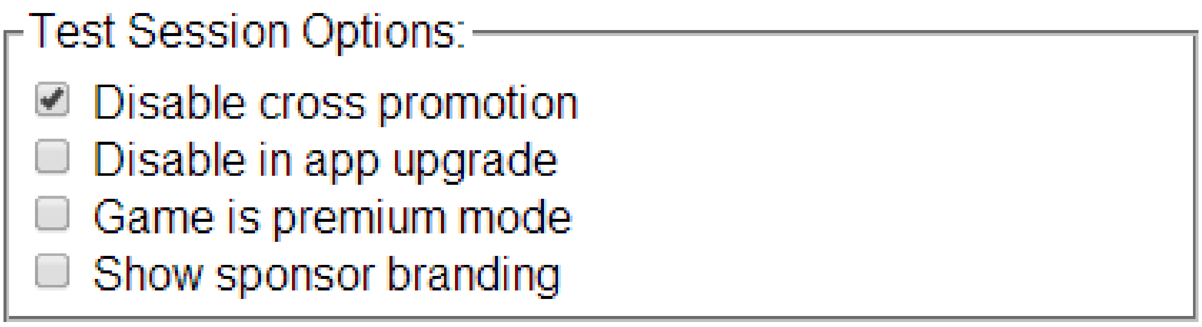
The More Games button should invoke the following code:

```
fgl.showMoreGames();
```

Not all target platforms allow cross promotion, so your More Games button must only display when cross promotion is enabled. You should check using this code before adding or drawing your button:

```
if(fgl.crossPromotionEnabled){
    // Button must be displayed
} else {
    // Button must be hidden
}
```

You can test your implementation by using the Test Session Options on the intro screen of the FGL SDK:


Test Session Options:

- [✓] Disable cross promotion
- [ ] Disable in app upgrade
- [ ] Game is premium mode
- [ ] Show sponsor branding

Premium Model

To most effectively monetize your game on the various platforms, we offer a simple set of features to allow us to sell a full or premium version of your game.

The system works by allowing you to check if the game is running in premium or free mode:

```
if(fgl.isPremium()){
    // Allow premium content
}
```

You should use this feature to restrict content or features when your game is running in free mode. This will help your game to upsell to premium.

On many platforms your game will be unlockable to premium by a simple in-app purchase. You should take advantage of this by implementing an unlock button in various places within your game. The unlock buttons should only be displayed if the premium upgrade feature is available:

```
if(fgl.unlockEnabled){
    // Render or display unlock button
}
```

By using this method we can still sell a premium version of your game on markets which don't support in app purchases, as well as distributing a free ad-supported version of your game.

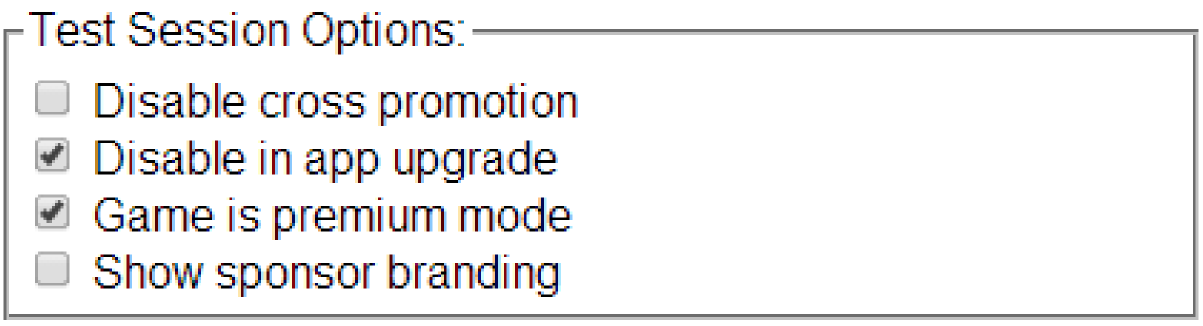
(Continued on next page)

Premium Model (cont.)

When tapped/clicked, your unlock button should call the following code:

```
fgl.inApp.initiateUnlockFunction(

    // On Success:
    function(){
        // Success! Unlock your game's premium content here.
    },

    // On Failure:
    function(){
        // Make sure premium content is locked here
        // Do not notify the user, that is handled by the SDK
    }
);
```

You can test your implementation by using the Test Session Options on the intro screen of the FGL SDK:


Test Session Options:

- [ ] Disable cross promotion
- [✓] Disable in app upgrade
- [✓] Game is premium mode
- [ ] Show sponsor branding

Publisher Branding

To further monetize your game, sometimes we may have the option to sell a "sponsorship branded" version of your game to a publisher, much like on the FGL Game Shop. If you would like your game to be able to make money in this way, then you must implement our simple publisher branding functionality.

You should display the image on your main menu, and in any other menus where it make sense. Publishers will be able to see where their branding will show up in your game before they purchase, so placing the branding somewhere non-prominent such as only in the credits menu may result in fewer sales.

You can check to see if branding is enabled using the `brandingEnabled` property. The image asset `src` attribute is determined by the `getBrandingLogo()` method. And when the player clicks or taps the branding logo it should invoke the `handleBrandingClick()` method.

Example Implementation:

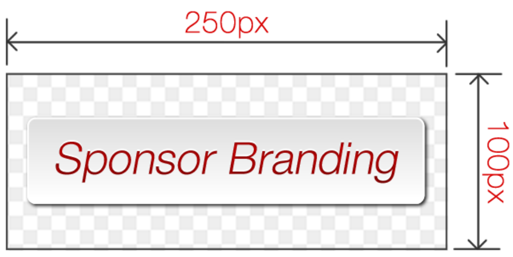
```
if(fgl.brandingEnabled){
    var img = document.createElement('img');
    img.src = fgl.getBrandingLogo();
    img.onclick = fgl.handleBrandingClick;

    myGame.appendChild(img);
}
```

(Continued on next page)

Publisher Branding (cont.)

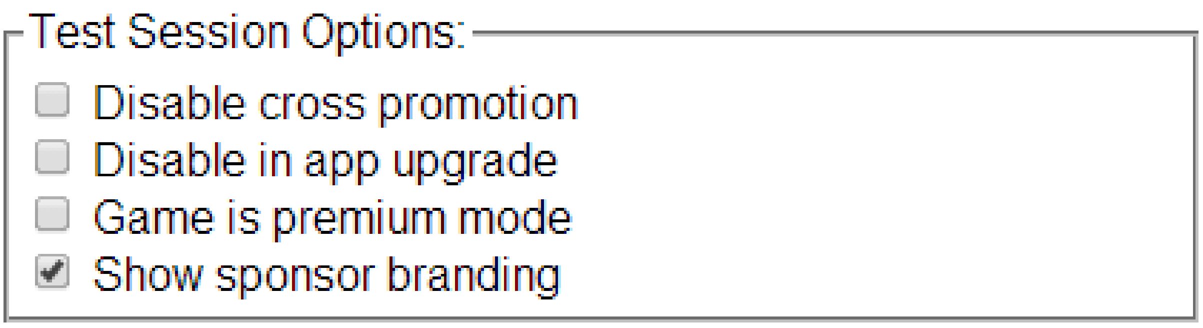
When you implement publisher branding, you should see a test branding placeholder. This will help you to properly position the logo on your menu screens.

The image will always be 250px by 100px in size and will have a transparent background



Example sponsor logo image

You can test your implementation by checking the "Show sponsor branding" option from the Test Session Options on the intro screen of the FGL SDK:



Test Session Options:

- ☐ Disable cross promotion
- ☐ Disable in app upgrade
- ☐ Game is premium mode
- ☑ Show sponsor branding

If the branding image doesn't work, you may be experiencing issues with cross-domain image loading. To fix this, set the 'crossorigin' attribute to 'anonymous' to enable cross-origin resource sharing (CORS).

Scoreboards

Our generic leaderboard functions allow you to implement one simple set of leaderboard calls - we can then do all the work to make your HTML5 game use the best leaderboard system for each platform. This gives your game an advantage as many platforms will give additional promotion or exposure to games which integrate with their proprietary systems.

The following functions are available:

```
// Submit the score in response to a user pressing a submit button:
fgl.submitScore(score);

// Show the scoreboard UI. This happens automatically after submitScore:
fgl.displayScoreboard();
```

See the API documentation for more details.

Achievements

As with leaderboards, the generic achievement functions give you an advantage of being able to use many proprietary game systems without modifying your code.

The following functions are available:

```
// Grant a named achievement to the player:
fgl.grantAchievement(name);

// Check if the player has previously been granted an achievement:
boolean fgl.hasAchievement(name);
```

See the API documentation for more details.

In-app purchases

In-app purchases are not currently implemented, but will be included in future versions of the FGL SDK.