Google Web Designer + Dynamic Ads in AdWords: Build Guide

Overview

This document outlines the steps necessary for the creation of AdWords Dynamic Remarketing ads using the new dynamic features and templates in Google Web Designer.

Dynamic Remarketing ads are those which show users the most relevant offer for each impression. The offers are pulled from a data feed and populate the dynamic ads based on remarketing and contextual user signals.

Contents

First Steps

Getting Started with AdWords

If you're unfamiliar with AdWords, see below for an introduction to get started

- AdWords Dynamic Remarketing Overview

Getting Started with Google Web Designer

If you're unfamiliar with using Google Web Designer, here are some helpful resources to get started

- Download Google Web Designer
- Google Web Designer Help Center
- Google Web Designer Youtube channel

Feed Types

AdWords ads are served under different **"Business Types".** Each business type has a unique feed schema and available dynamic attributes which the creative can access to display data and define settings. The developer must become familiar with these attributes and decide which ones the creative should use.

Each business type has a different schema. The schema defines the list of possible feed attributes which the creative can bind to. Binding to an attribute means linking an element in the creative to a feed value, for example: a div element may bind to a feed attribute which determines its color value, or a text element binds to an attribute which determines its text content.

Each feed comes in 3 sections:

Design

Defines colors and styling. Set when creating the ad in AdWords.

Feed Attribute	Mapping Value
1. bgAlpha	1. Background Alpha
2. bgColor	2. Background Color
3. bgColorAlt	3. Background Color 2
4. bgGradient	4. Background Gradient
5. bgImageUrl	5. Background Image
6. borderColor	6. Border Color
7. btnBevel	7. Button Bevel
8. btnColor	8. Button Color
9. btnRollColor	9. Button Mouseover Color
10. btnShad	10. Button Shadow
11. btnStyle	11. Button Style (round or square)
12. cornerStyle	12. Corner Style (round or square)
13. logoImageUrl	13. Logo Image
14. txtColorCta	14. Button Text Color
15. txtColorDescription	15. Description Text Color
16. txtColorDisc	16. Disclaimer Text Color
17. txtColorPricePrefix	17. Price Prefix Text Color
18. txtColorPriceSuffix	18. Price Suffix Text Color
19. txtColorPrice	19. Price Text Color
20. txtColorProduct	20. Name Text Color
21. txtColorSubTitle	21. Sub Title Text Color
22. txtColorTitle	22. Headline Text Color

23. txtShadow	23. Text Shadow

Headline

Text fields and miscellaneous settings which are set by the user when creating the ad in AdWords.
This is separate to the text which comes from the product feed.

Feed Attribute	Mapping Value
1. txt	1. Headline Text
2. cta	2. CTA Button Text
3. disclaimer	3. Disclaimer Text
4. pricePrefix	4. Price Prefix Text
5. priceSuffix	5. Price Suffix Text
6. showPrice	6. Show the Price
7. productClickOnly	7. Always Navigate to the Feed Url (See "Handling Exits")

Product

The product feed data.
This is an array of up to 6 feed items.
Each Use case contains a different set of attributes, each mapping to different values.

Retail (GMC Feed)	
Feed Attribute	**Mapping Value**
1. name	1. Product Summary
2. description	2. Product Details
3. price	3. Lowest Price
4. regularPrice	4. Regular Price
5. salePrice	5. Sale Price
6. salePercentDiscount	6. Price Discount (as a decimal)
7. installmentCount	7. Installment Price Count
8. installmentPrice	8. Installment Price
9. imageUrl	9. Image Url
10. url	10. Item landing page

Custom Use Case	
Feed Attribute	**Mapping Value**
1. name	1. Item Title
2. subTitle	2. Item Subtitle
3. description	3. Item Description
4. price	4. Regular Price
5. salePrice	5. Sale Price
6. salePercentDiscount	6. Price Discount (as a decimal)
7. imageUrl	7. Image Url
8. url	8. Item landing page

Travel	
Feed Attribute	**Mapping Value**
1. name 2. subTitle 3. description 4. price 5. salePrice 6. salePercentDiscount 7. imageUrl 8. url	1. Title 2. Destination Name 3. Origin Name 4. Regular Price 5. Sale Price 6. Price Discount (as a decimal) 7. Image Url 8. Item landing page

Flight	
Feed Attribute	**Mapping Value**
1. name 2. price 3. salePrice 4. salePercentDiscount 5. imageUrl 6. url	1. Flight Description 2. Regular Price 3. Sale Price 4. Price Discount (as a decimal) 5. Image Url 6. Item landing page

Hotel and Rental	
Feed Attribute	**Mapping Value**
1. name 2. subTitle 3. description 4. price 5. salePrice 6. salePercentDiscount 7. rating 8. imageUrl 9. url	1. Property Name 2. Destination Name 3. Description 4. Regular Price 5. Sale Price 6. Price Discount (as a decimal) 7. Star Rating (0 - 5) 8. Image Url 9. Item landing page

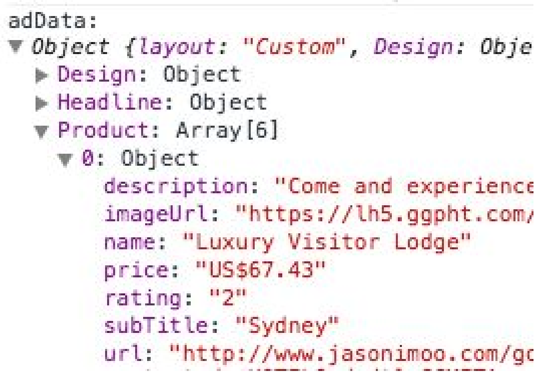
Jobs	
Feed Attribute	**Mapping Value**
1. name 2. subTitle 3. description 4. price 5. salePrice 6. salePercentDiscount 7. imageUrl 8. url	1. Job Title 2. Job Subtitle 3. Job Description 4. Regular Price 5. Sale Price 6. Price Discount (as a decimal) 7. Image Url 8. Item landing page

Local Deals	
Feed Attribute	**Mapping Value**
1. name	1. Deal Name
2. subTitle	2. Deal Subtitle
3. description	3. Deal Description
4. price	4. Regular Price
5. salePrice	5. Sale Price
6. salePercentDiscount	6. Price Discount (as a decimal)
7. imageUrl	7. Image Url
8. url	8. Item landing page

Real Estate / Property	
Feed Attribute	**Mapping Value**
1. name	1. Listing Name
2. subTitle	2. City Name
3. description	3. Description
4. price	4. Regular Price
5. salePrice	5. Sale Price
6. salePercentDiscount	6. Price Discount (as a decimal)
7. imageUrl	7. Image Url
8. url	8. Item landing page

Education	
Feed Attribute	**Mapping Value**
1. name	1. Program Name
2. subTitle	2. Area of Study
3. description	3. Program Description
4. imageUrl	4. Image Url
5. url	5. Item landing page

Tip: *You can inspect each feed when previewing the creative if you open your browser developer tools (Chrome: Settings > More Tools > Developer Tools). Below shows how this appears, when expanding the “**adData Object**”.*



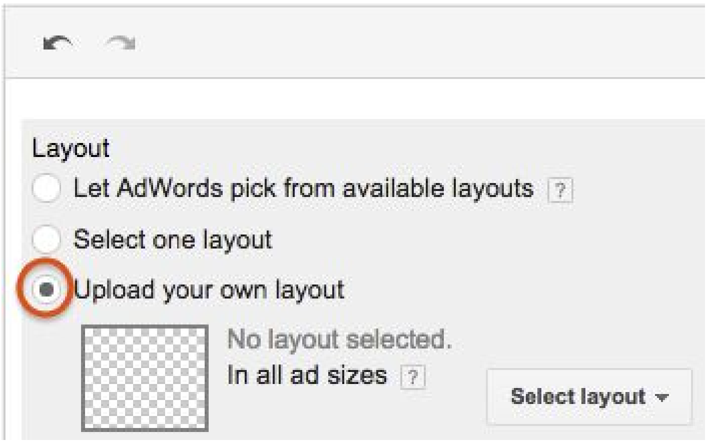
```
adData:
▼ Object {layout: "Custom", Design: Obje
  ▶ Design: Object
  ▶ Headline: Object
  ▼ Product: Array[6]
    ▼ 0: Object
        description: "Come and experience
        imageUrl: "https://lh5.ggpht.com/
        name: "Luxury Visitor Lodge"
        price: "US$67.43"
        rating: "2"
        subTitle: "Sydney"
        url: "http://www.jasonimoo.com/gc
```

Saving Client Feeds

Once your feeds are set up in AdWords, you can save some samples in the form of downloadable JSON files to use during development. This allows you to preview using your client's feed data instead of the generic sample feeds which come with each **Template**.

Note: This feature is currently available to Whitelisted accounts only

1. Go into AdWords and create a new ad.
2. In the Ad builder tool, choose the option to **Upload your own layout**

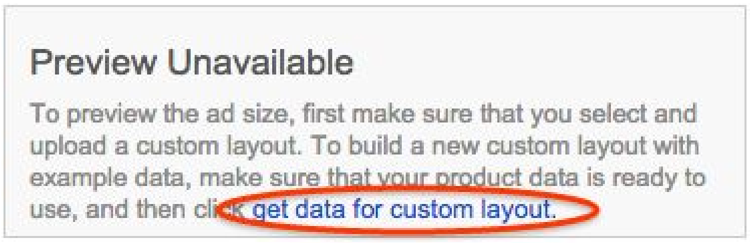


Layout

- Let AdWords pick from available layouts
- Select one layout
- [✓ Upload your own layout]

No layout selected.
In all ad sizes

3. This brings up a blank preview page including a link to "**get data for custom layout**"



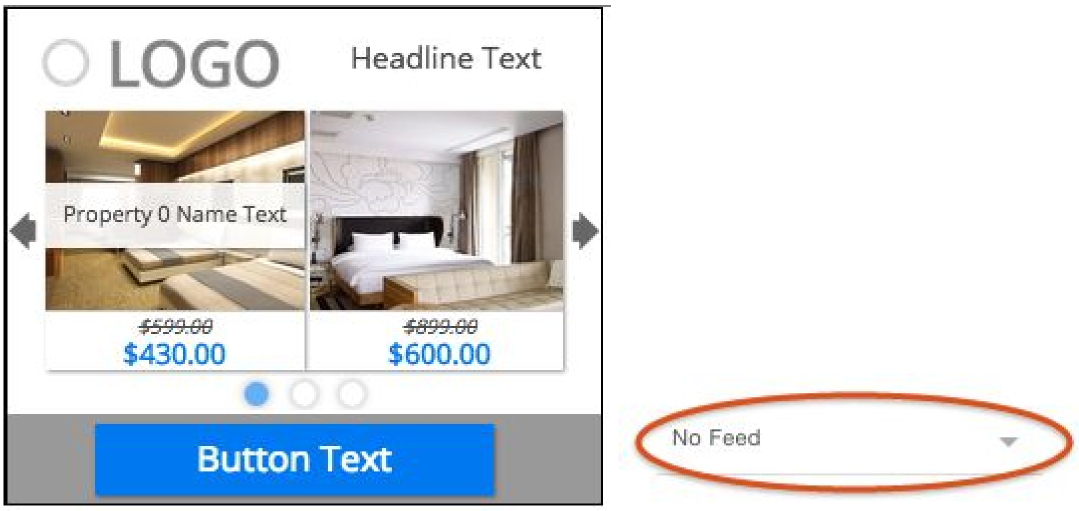
Preview Unavailable

To preview the ad size, first make sure that you select and upload a custom layout. To build a new custom layout with example data, make sure that your product data is ready to use, and then click get data for custom layout.

4. Click this link and save the resulting page as a JSON file into you creative's **feeds** folder. This is a folder inside your main creative folder named "feeds" and is automatically added only once a binding has been created, or if starting with a Template.

Preview with Client Feed

Once you've saved a new json file, you can preview by opening your ad in Google Web Designer and clicking preview. You can then select your feed from the dropdown at the bottom right of the page.



Data Binding Overview

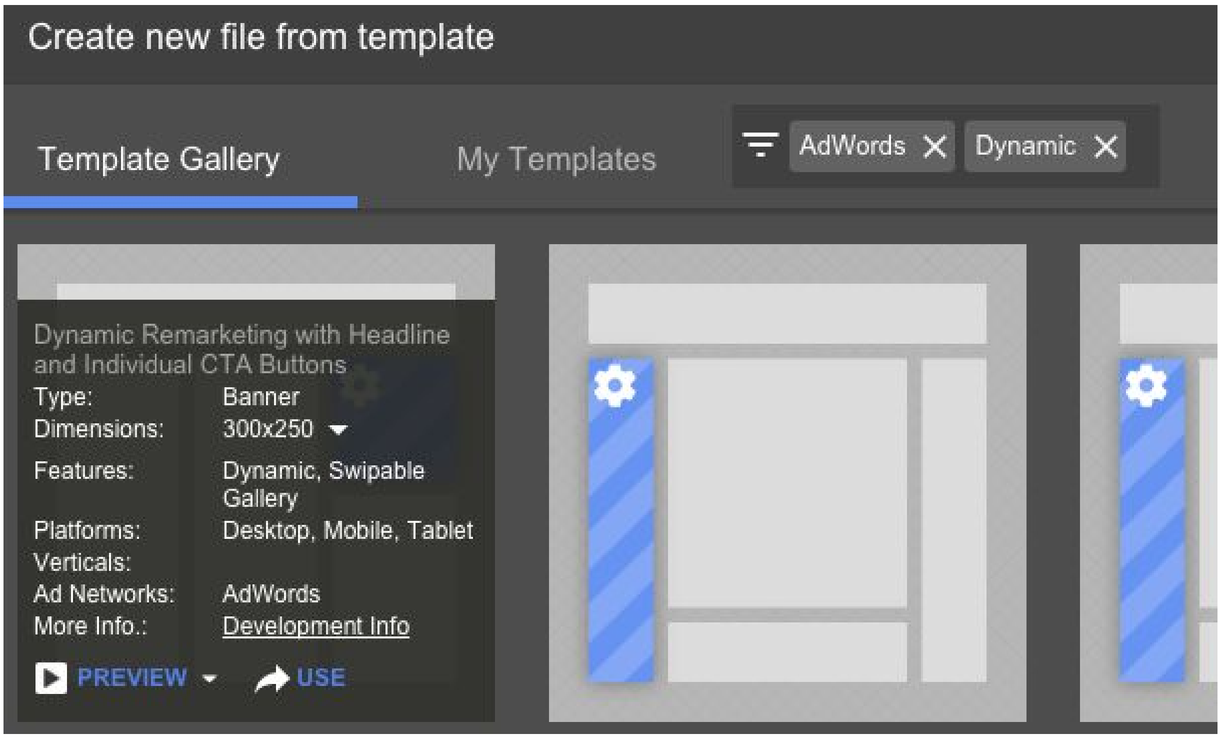
This page offers instructions on binding your creative's elements to your feed attributes: https://support.google.com/webdesigner/answer/6212374

Using the Templates

All **AdWords Dynamic Ads** currently need to start from a Template. Starting with a Template is important in order to ensure the necessary base code is included into your creative.

Note: If you want to start from scratch and not have any predesigned elements, there is a **Blank slate** template also available which only includes the base code.

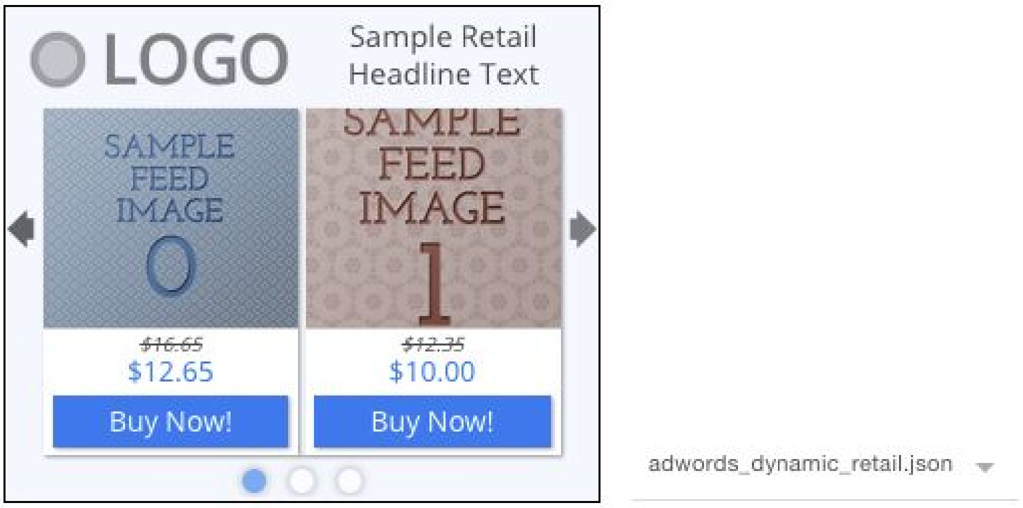
1. Open Google Web Designer and choose File > **New from template** to get the templates library.
2. To show only the AdWords Dynamic Remarketing Templates, filter on **Network: AdWords** and **Features: Dynamic**



Dynamic Remarketing with Headline and Individual CTA Buttons

Type:	Banner
Dimensions:	300x250
Features:	Dynamic, Swipable Gallery
Platforms:	Desktop, Mobile, Tablet
Verticals:	
Ad Networks:	AdWords
More Info.:	Development Info

3. Choose your size and click **Use**
4. Change the document size and adjust elements accordingly if you require a different ad size.
5. Save and preview by selecting either a sample feed or one you've saved into the **feeds** folder *(see "Saving Client Feeds" section)*



Required AdWords Dynamic Ads functionality

The **AdWords Dynamic Ads** Templates include separate JS and CSS files.

These files provide base code: required and optional javascript and CSS to help build your creatives.

Note: *All code shown below already exists in the templates.*

These files are accessed throughout all adsizes you create and are found in your creative's **custom** folder.

- **Common code**: applies to all Templates.
  - **utils**.js
    - Utilities to help with commonly used functionality.
  - **common**.js
    - Required functionality.
  - **common**.css
    - Common CSS and widely used classes.

- **Custom code**: add any custom functionality here.
  - **custom**.js
    - Custom functionality you need to apply to all ad sizes.
  - **custom**.css
    - Custom styling, transitions, and other CSS you need to apply to all adsizes.

Initializing AdWords Dynamic Ads code

Looking in Code View, you'll see each of the custom and common javascript files initialized in the `handleAdInitialized` function:

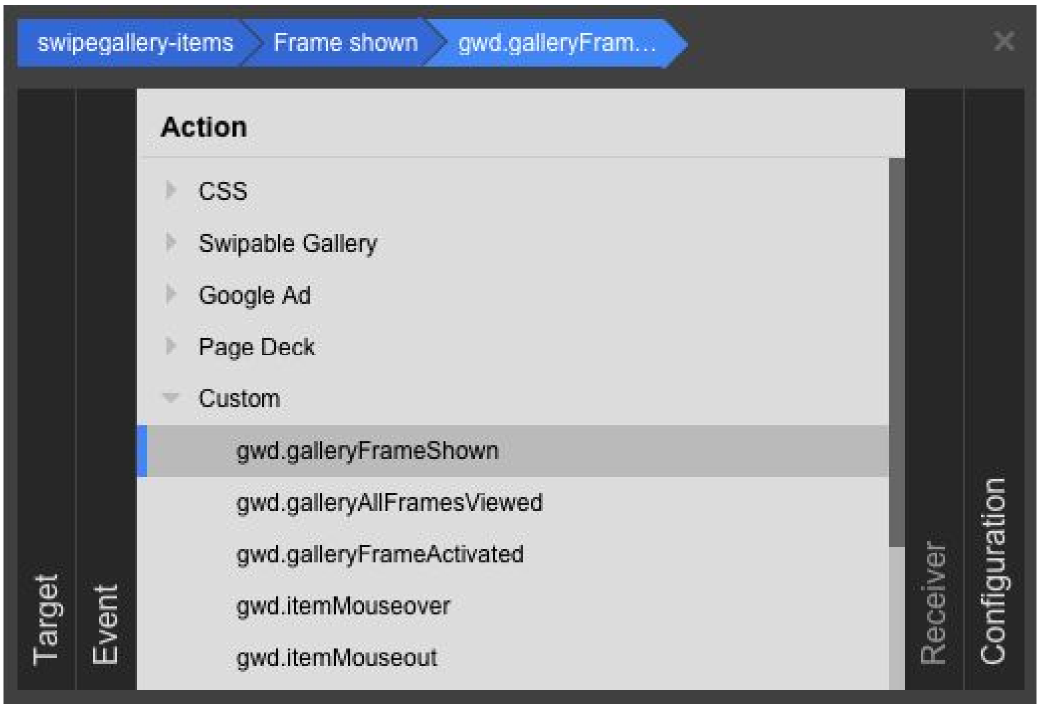
- `common.init();`
- `custom.init();`

Default AdWords Dynamic Ads event handlers

There are default functions specific to AdWords Dynamic Remarketing ads:

- 3 functions to handle **exits**
- 3 functions to handle common **Swipe-gallery events** (relevant if the Template uses the swipegallery)
- 2 functions to handle on **mouse interaction events**

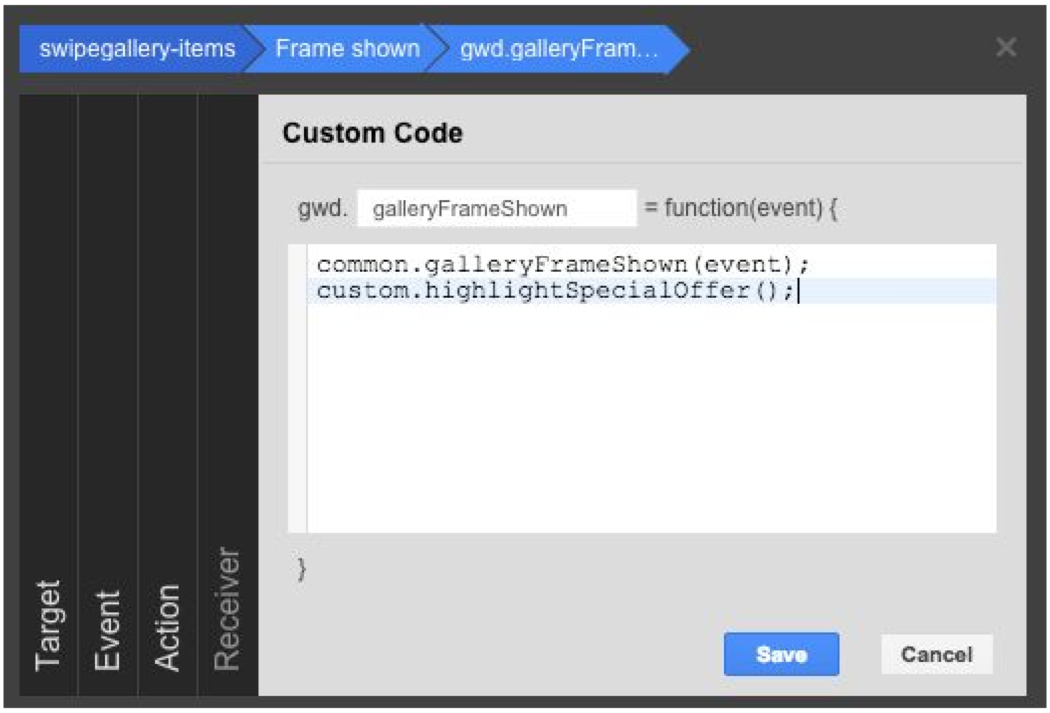
These functions are accessible through the **Events panel** when editing / adding an event. Expanding the **Custom Actions** section reveals the common actions, alongside any new actions you create



The functionality for these event handlers comes from functions of the same name in **common.js**

If you want to add some custom functionality to any of these events, just create a new function in your **custom.js** and then add it the function name, prefixed with "custom." (since it's coming from the custom module).

Example adding a custom function when each swipe-gallery frame becomes visible: "custom.highlightSpecialOffer()"



```
gwd. galleryFrameShown = function(event) {
common.galleryFrameShown(event);
custom.highlightSpecialOffer();
}
```

Looking in Code View you'll see these event handler functions under the **Event Handlers** section
`<script type="text/javascript" gwd-events="handlers">`

Handling Feed Items (Products)

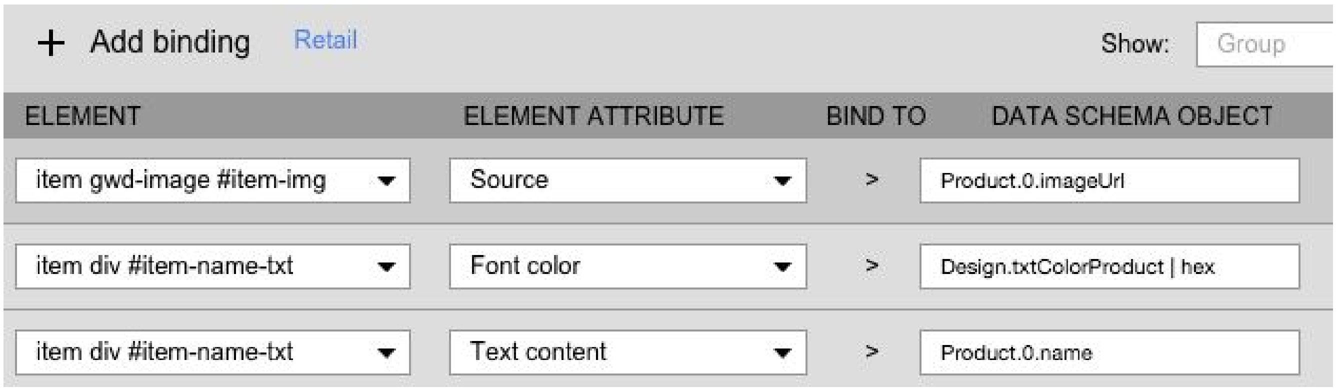
If your creative will display more than one feed item (more than one item in the **Product** section of the feed), then you need to track user interaction. User interaction includes mousing over each item or navigating the gallery to update the currently visible item. Keeping track of these interactions is required in order for the creative to know which item in the feed is currently active. Keeping track of the currently active item allows the creative to update itself, either by visually highlighting the active item or by updating the current landing page url and directing the user to the correct page once they click on the creative.

Each feed item is displayed in the creative as a "**group"**, allowing each of the feed attributes (image, name, price, etc.) to be replicated easily as a single element and linked to each item in the Product feed. SInce each group is linked to each item in the Product feed, the group is called "**item**".

The feed here refers to the Product array, and each item refers to each object in that array. See "Feed Types" section for more detail).

Create the item group

Create your new group, calling it "item", then create its child element and bind each to the relevant feed attributes

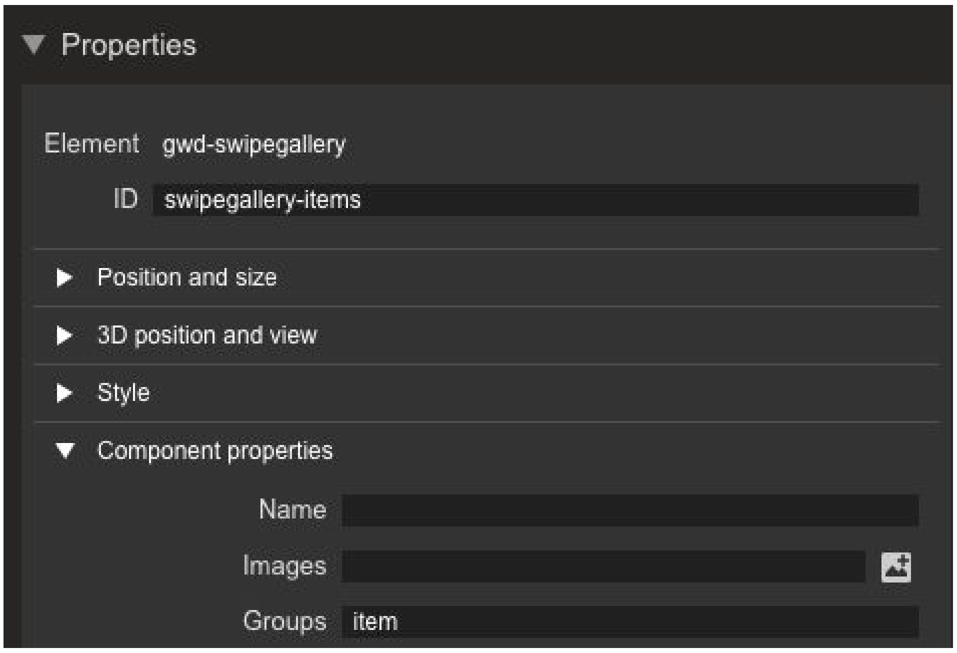


ELEMENT	ELEMENT ATTRIBUTE	BIND TO	DATA SCHEMA OBJECT
item gwd-image #item-img	Source	>	Product.0.imageUrl
item div #item-name-txt	Font color	>	Design.txtColorProduct | hex
item div #item-name-txt	Text content	>	Product.0.name

Display items through a swipe-gallery

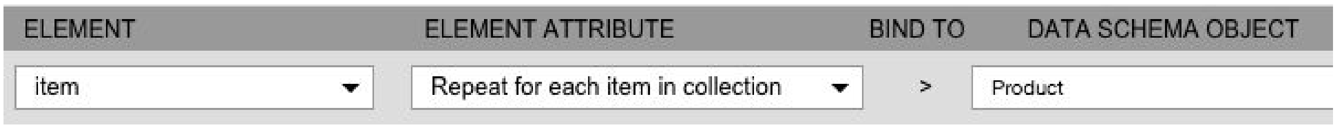
If you're adding a swipe-gallery to your creative, you need to follow a few steps to access each item with the common and custom code.

Add the component to the stage and name it, eg "swipegallery-items". Add the group you want to populate the gallery with in the Component Properties.



Element gwd-swipegallery
ID swipegallery-items
Name
Images
Groups item

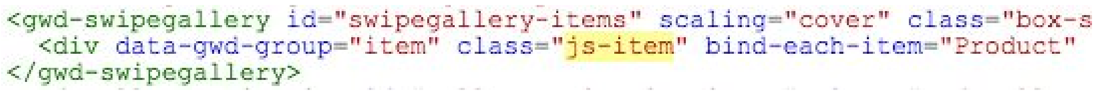
Bind each group in the swipe-gallery with each item in the feed by binding the item to "**Repeat for each item in the collection**", where the collection is the **Product** array.



ELEMENT	ELEMENT ATTRIBUTE	BIND TO	DATA SCHEMA OBJECT
item	Repeat for each item in collection	>	Product

Then you need to add a classname to your item in order for the common code to target it as a feed item.

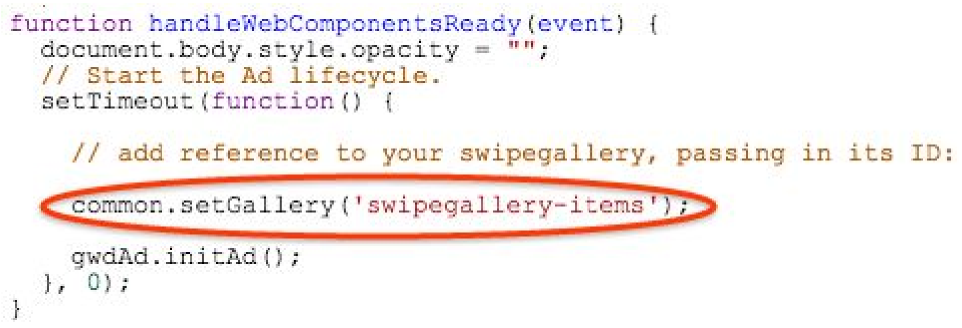
In **Code view:** go to the swipe-gallery where you'll see your "item" element. Add a class named: ″js-item″



```
<gwd-swipegallery id="swipegallery-items" scaling="cover" class="box-s
  <div data-gwd-group="item" class="js-item" bind-each-item="Product"
</gwd-swipegallery>
```

Then inside the handleWebComponentsReady function (which is called when the components have been registered), add the common.setGallery function, passing in the swipe-gallery's ID, eg. "'swipegallery-items".

This is already included in the templates, and so you'll just need to uncomment this line when adding your gallery.

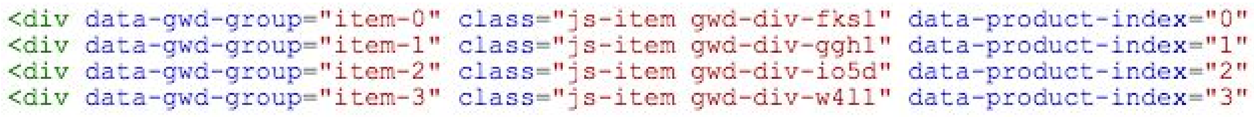


```
function handleWebComponentsReady(event) {
  document.body.style.opacity = "";
  // Start the Ad lifecycle.
  setTimeout(function() {

    // add reference to your swipegallery, passing in its ID:

    common.setGallery('swipegallery-items');

    gwdAd.initAd();
  }, 0);
}
```

Display items without a swipe-gallery

If you're adding items directly to the stage, you only need to do the following:

1. Add a class named: ″js-item″ to each item.
2. Add a reference to the index in the feed which you want the item to links to by adding an attribute called data-product-index

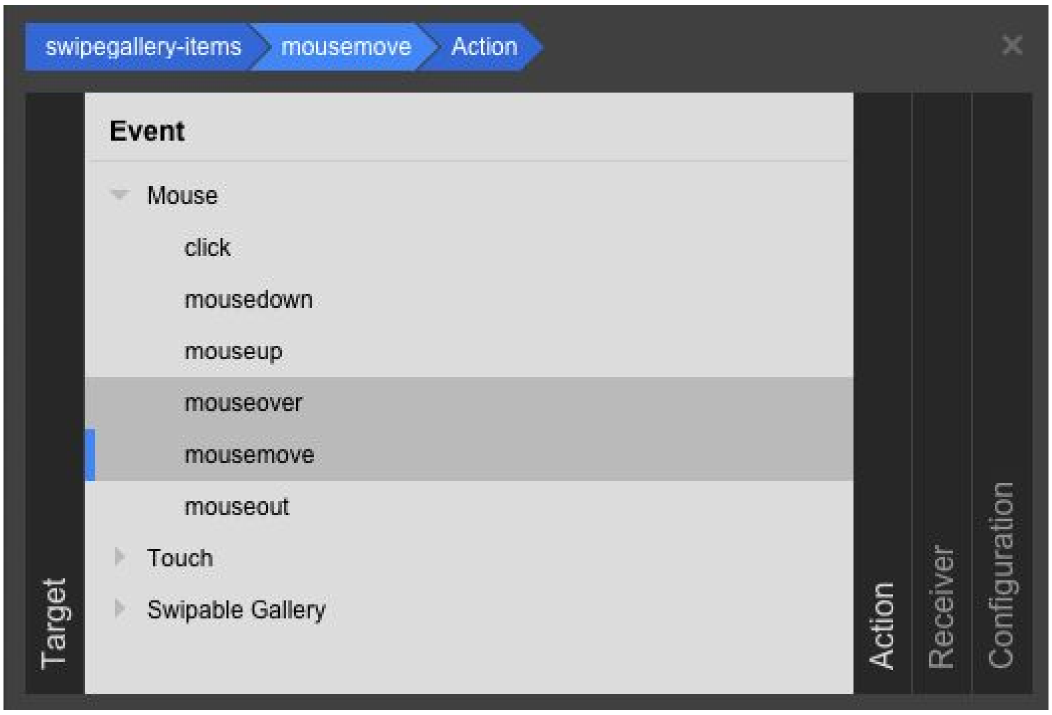


```
<div data-gwd-group="item-0" class="js-item gwd-div-fksl" data-product-index="0"
<div data-gwd-group="item-1" class="js-item gwd-div-gghl" data-product-index="1"
<div data-gwd-group="item-2" class="js-item gwd-div-io5d" data-product-index="2"
<div data-gwd-group="item-3" class="js-item gwd-div-w4l1" data-product-index="3"
```

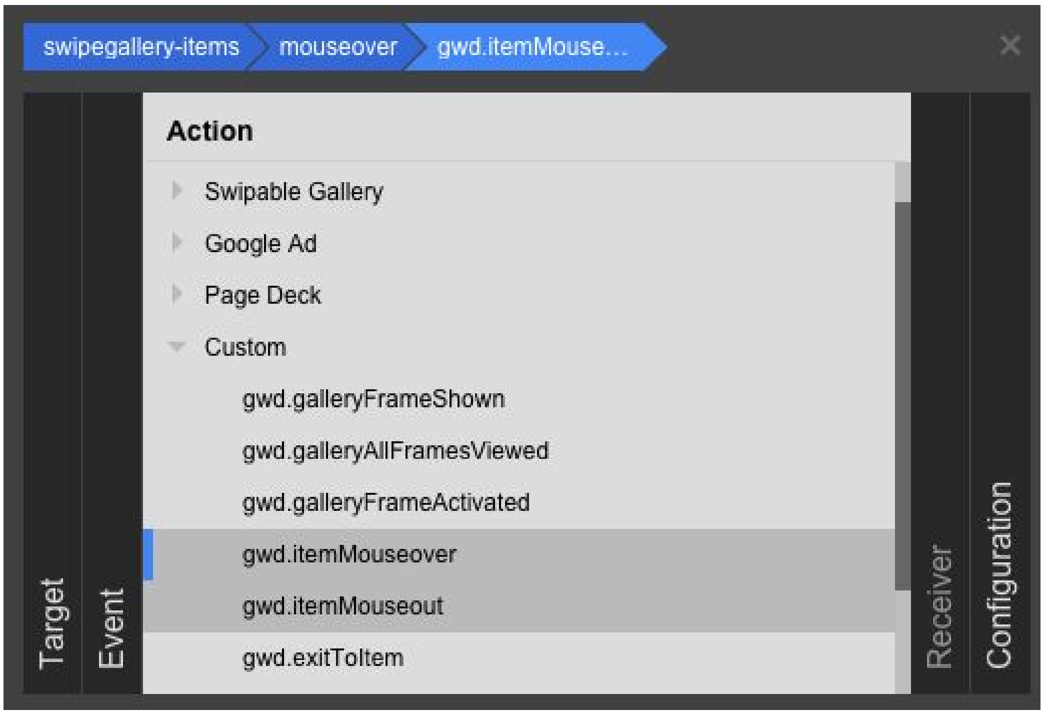
Adding user interaction events to feed items

All items must include mouseover events to ensure the creative keeps track of the currently active item, and the functionality to do this is included in the Template's common code. You therefore only need to link each item's mouseover and mouseout events to the default mouseover and mouseout functions.

Note: Even if your items are groups within the **swipe-gallery** component, you must add interaction via the **Mouse events**, not the **Swipeable Gallery events.**



Choose the itemMouseover and itemMouseout AdWords Dynamic Ads default actions, listed under “Custom”.



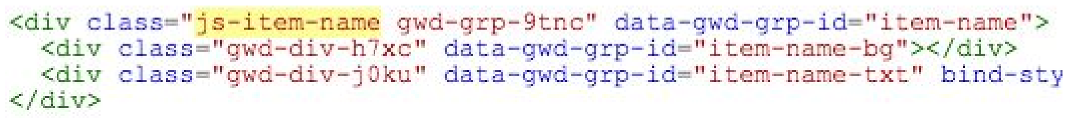
Note: This is the only function in the common code which calls the same function in the custom code, so you don’t need to add your own function to this; as long as you have a function in custom.js with the same name, it will also be called.

Adding custom user interaction

If you want to add custom behaviour to child elements within your item, add a class to each element enabling the custom code to identify each one.

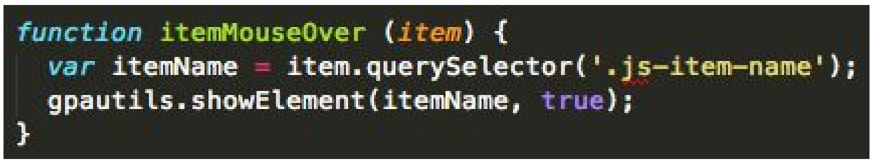
This class is only used as a hook for your custom javascript. It should therefore begin with “js-” and end with the name of the element, eg. the item name field’s element: “js-item-name”.

Add this to the beginning of the element’s class list:



```
<div class="js-item-name gwd-grp-9tnc" data-gwd-grp-id="item-name">
  <div class="gwd-div-h7xc" data-gwd-grp-id="item-name-bg"></div>
  <div class="gwd-div-j0ku" data-gwd-grp-id="item-name-txt" bind-sty
</div>
```

In your **custom.js** reference each item via its classname, by using a querySelector on the **item** element, which is passed into the **itemMouseOver** and **itemMouseOut** functions.



```
function itemMouseOver (item) {
  var itemName = item.querySelector('.js-item-name');
  gpautils.showElement(itemName, true);
}
```

Initializing custom behaviour for each item

If you want to apply some custom functionality to each item once the creative loads, either to items inside the swipe-gallery or added directly to the page, the Template's custom.js code includes some useful functions to easily do this.

The custom code starts with a function `init()` which is called from the creative's `handleAdInitialized()` function. This is where functionality is applied to the items once the ad has finished loading.

There are two functions already in here, and you should use one depending on where or not the gallery is used:

1. `initItemsUsingGallery()` *(uncomment if you're using the swipe-gallery)*
2. `initItemsWithoutGallery()` *(uncomment if you're **not** using the swipe-gallery)*

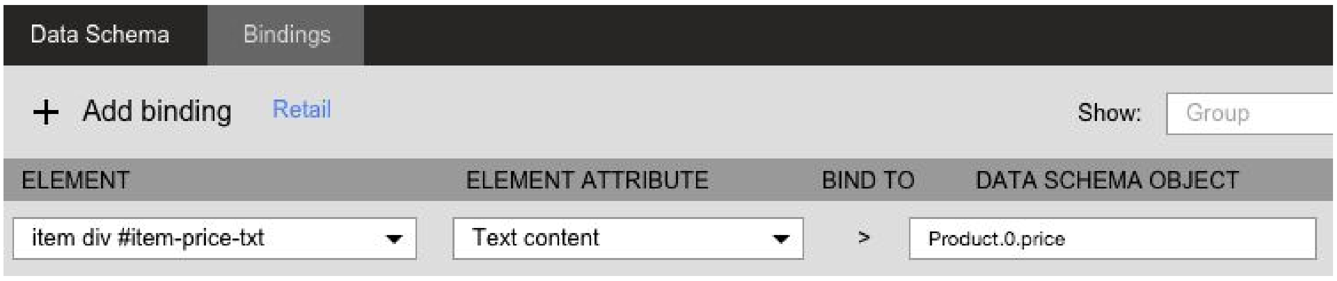
Each of these functions works by calling the `initItemDisplay()` function for each of the item elements. Since this function passes in a reference to each item, you can set any custom default behaviour to each item here. Common functionality to ad is simply including a call to the `itemMouseOut()` function, so the creative starts with each item being in the mouseout state.

Displaying Prices

AdWords feeds contain different types of prices, and the AdWords Dynamic Ads Templates include useful functionality to determine which of these prices should display for each feed item.

Displaying a single price only

If your feed only contains single prices you simply bind the single price text field to the "**price**" attribute



ELEMENT	ELEMENT ATTRIBUTE	BIND TO	DATA SCHEMA OBJECT
item div #item-price-txt	Text content	>	Product.0.price

Displaying the sales price with the regular price

For feeds which contain sales prices, which are often only available to some feed items, the AdWords Dynamic Ads common code provides functionality to determine how to display this alongside the default price.

Create three text fields, for two scenarios:

A. **default price**
B. **regular price** with **sales price**

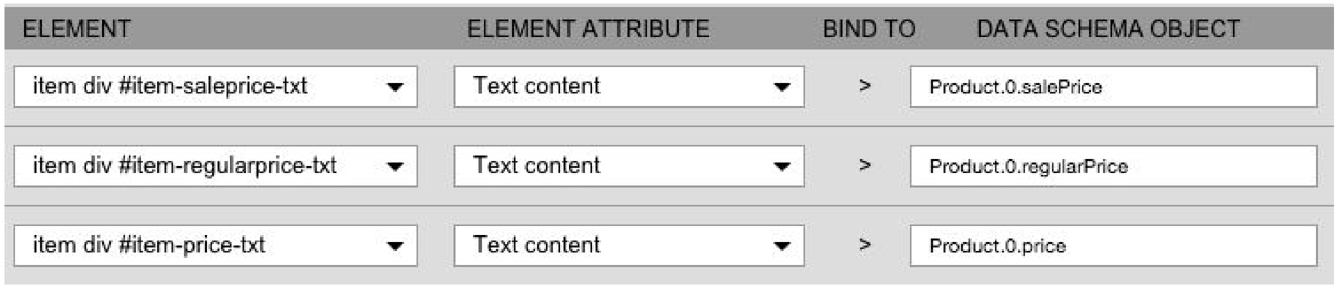
Example:

1. `item-price-txt` (default price: displays alone)
2. `item-regularprice-txt` (regular price: often smaller with italics with strikethrough)
3. `item-saleprice-txt` (sale price: larger and positioned alongside the regular price)

Bind to the following attributes:

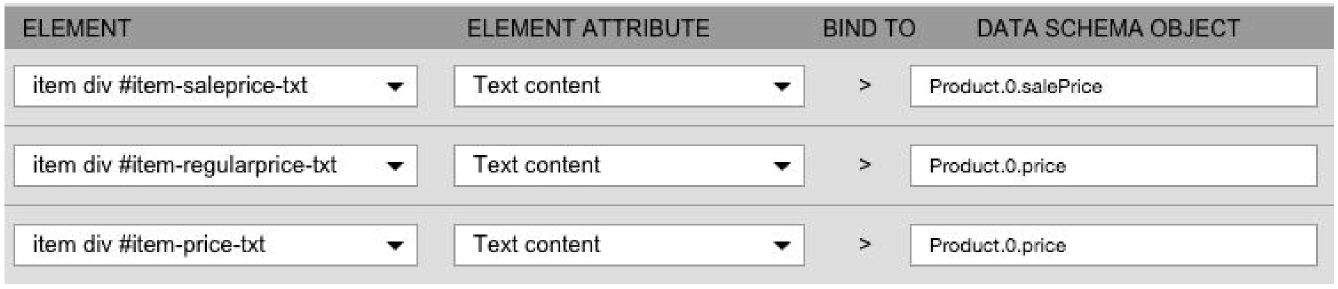
- item-price-txt = Product.0.price
- item-saleprice-txt = Product.0.salePrice
- ***For GMC (retail) feeds only:*** item-regularprice-txt = Product.0.regularPrice
- ***For all other feeds:*** item-regularprice-txt = Product.0.price

Example GMC feed:



ELEMENT	ELEMENT ATTRIBUTE	BIND TO	DATA SCHEMA OBJECT
item div #item-saleprice-txt	Text content	>	Product.0.salePrice
item div #item-regularprice-txt	Text content	>	Product.0.regularPrice
item div #item-price-txt	Text content	>	Product.0.price

Example other feed:



ELEMENT	ELEMENT ATTRIBUTE	BIND TO	DATA SCHEMA OBJECT
item div #item-saleprice-txt	Text content	>	Product.0.salePrice
item div #item-regularprice-txt	Text content	>	Product.0.price
item div #item-price-txt	Text content	>	Product.0.price

Open **custom.js** and in the `initItemDisplay` function (which is called for each feed item), pass a reference to each of these three text fields into this function on common.js:

```
common.displayCorrectPrices(itemPrice, itemSalePrice, itemRegularPrice);
```

This will ensure the correct price text fields are displayed, depending on whether an item has a sales price available and whether its value is unique to the regular price.

Handling Exits

AdWords Dynamic Remarketing Ads have two landing page urls which the creative can exit to:

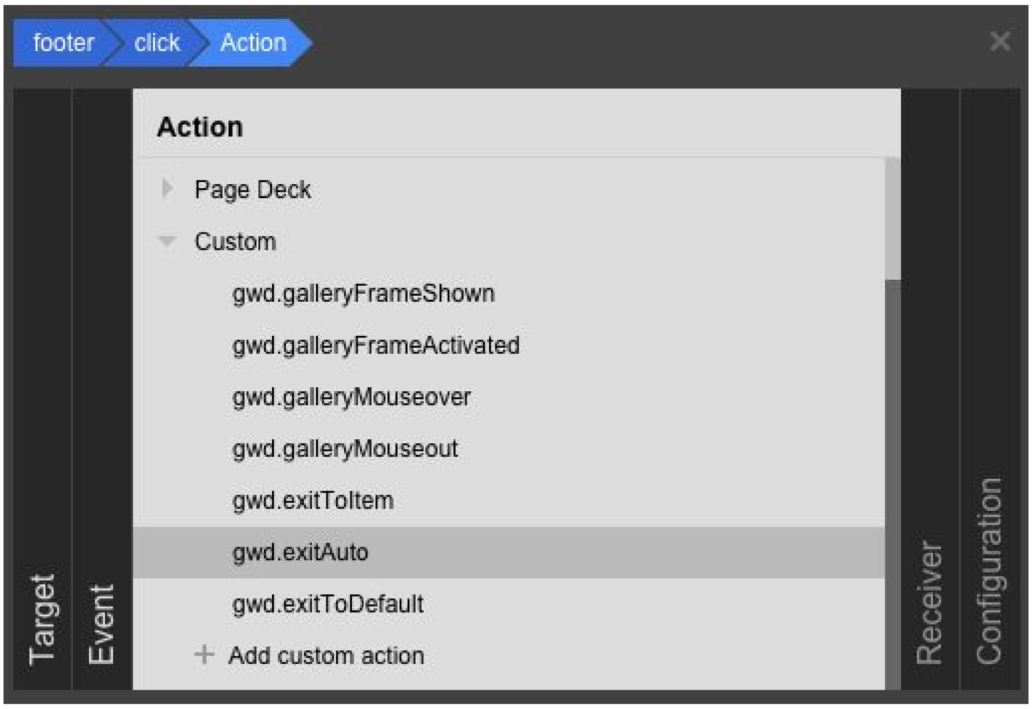
1. Exit to the **feed item's url**
    - The url for each item in the feed.
2. Exit to the **Default url**
    - The url which the user defines while setting up the ad in AdWords (eg. home page url)

The AdWords Dynamic Remarketing Ads Templates include code which provides functionality to easily hook into each of these exit types.

Below are instructions for 3 types of exit functionality, using the Template code (*option A*), and an explanation on how to do this yourself (*option B*).

It's advised to use the template code option, since this enables the creative to easily keep track of the current item's index url via the gwd.**galleryMouseOver** and gwd.**galleryMouseOut** events, which are added to each item *(refer to "Displaying feed items" section for instructions).*

These events are handled from the **Custom** events in the Actions panel:



Note: *Do not use the default Exit functions in the "Google Ad" Action in the Events panel.*

1. Exit to the current feed item's url

A. Use the Template code by setting the click event action to the predefined Custom event: `gwd.`**`exitToItem`**, which calls the common.js function: `common.`**`exitToItem`**`();`

B. ***Or*** create your own custom click event handler which calls `Enabler.exit`, passing in a reference to the current feed item's url attribute, eg. `Enabler.exit('Product_3_url');`

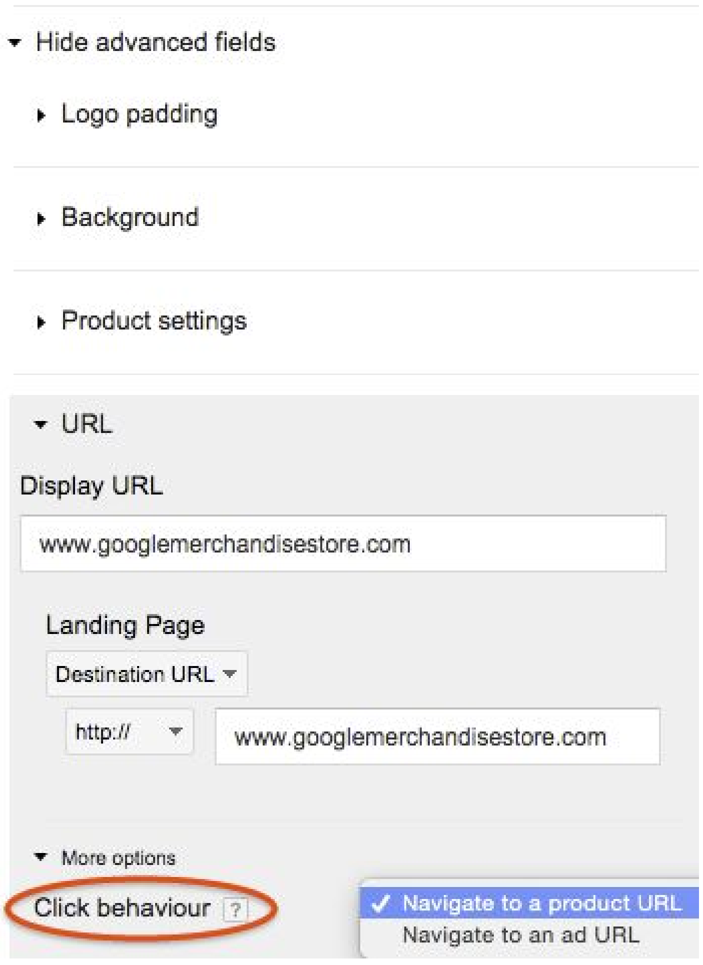
2. Exit to the default url

A. Use the Template code by setting the click event action to the predefined Custom event: `gwd.`**`exitToDefault`**, which calls the common.js function: `common.`**`exitToDefault`**`();`

B. ***Or*** create your own custom click event handler which calls `Enabler.exit('default');`

3. Exit to either the default url or the feed item's url based on user preference

When setting up each ad in AdWords, the user has the option to determine which url a non feed item area (like the logo, background, etc) exits to. This option is found under **Show advanced settings > Url > More options > Click behaviour:**


Hide advanced fields

Logo padding

Background

Product settings

URL

Display URL: www.googlemerchandisestore.com

Landing Page: Destination URL

http:// www.googlemerchandisestore.com

More options

Click behaviour: [✓ Navigate to a product URL] Navigate to an ad URL

The two options are read by the creative via the **productClickOnly** boolean attribute.

1. **Navigate to a product URL**:
   - *productClickOnly = true*
   - Non feed item areas click to the current feed item's url
2. **Navigate to an ad URL**:
   - *productClickOnly = false*
   - Non feed item areas click through to the default Landing Page url

A. Use the Template code by setting the click event action to the predefined Custom event: `gwd.exitAuto`, which calls the common.js function: `common.exitAuto();`

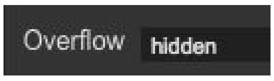
B. Or create your own custom click event handler which either calls Enabler.exit('default') if `productClickOnly` is false (for the Default Url), or pass in a reference to the current feed item's url attribute when `productClickOnly` is true, eg. `Enabler.exit('Product_3_url')`

Working with text

Text fields which have been bound to feed data will automatically resize until the text fits into the available space, or until it has reached 10pt.

If the text is still too long after being resized it will continue to extend beyond the text field boundaries.

To ensure this overflowing text is not visible, all text fields which are bound to dynamic text must also have overflow set to "hidden".


Overflow: hidden

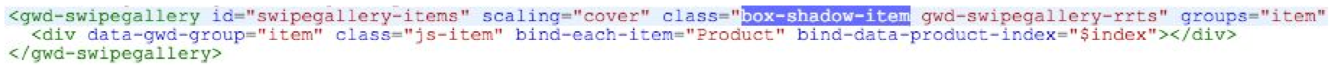
To ensure the overflowing lines of text are not cut off after resizing, it's important to ensure the text field is sized to a height which displays the text at 10pt with the last line of characters being fully visible.

Adding Custom Styles and Transitions

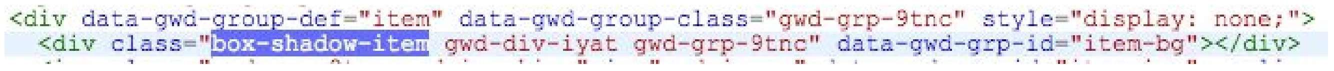
Apart from the styling available in Google Web Designer Design view, you can also add custom styling or effects by adding new classes to custom.css and applying each element using Code View. Some examples:

Adding Shadows to the gallery:

- add box-shadow-gallery to the item's base: item-bg
- add box-shadow-gallery to <gwd-swipegallery>

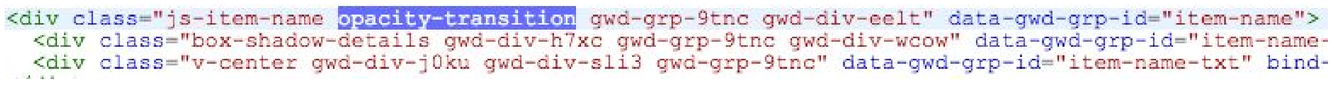


```
<gwd-swipegallery id="swipegallery-items" scaling="cover" class="box-shadow-item gwd-swipegallery-rrts" groups="item"
  <div data-gwd-group="item" class="js-item" bind-each-item="Product" bind-data-product-index="$index"></div>
</gwd-swipegallery>
```



```
<div data-gwd-group-def="item" data-gwd-group-class="gwd-grp-9tnc" style="display: none;">
  <div class="box-shadow-item gwd-div-iyat gwd-grp-9tnc" data-gwd-grp-id="item-bg"></div>
```

Adding transitions:

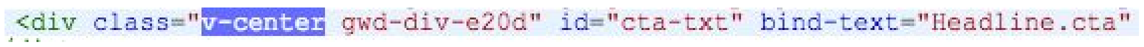


```
<div class="js-item-name opacity-transition gwd-grp-9tnc gwd-div-eelt" data-gwd-grp-id="item-name">
  <div class="box-shadow-details gwd-div-h7xc gwd-grp-9tnc gwd-div-wcow" data-gwd-grp-id="item-name-
  <div class="v-center gwd-div-j0ku gwd-div-sli3 gwd-grp-9tnc" data-gwd-grp-id="item-name-txt" bind-
```

Vertically Centering an Element

This is as yet unavailable in Design view and so some extra CSS needs to be added to an element to allow it to vertically center inside its parent.

1. Create a div and place the element you need vertically centered inside
2. Add your element and give it a position and size using the properties panel
3. Open the CSS panel and make the following changes:
    a. delete the **top** property
    b. change the **height** property to **max-height** and set the value to **100%**
    c. delete **position** property
4. Go into code view and find the element. Add **v-center** to the beginning of its list of classes:



```
<div class="v-center gwd-div-e20d" id="cta-txt" bind-text="Headline.cta"
```

QA and Testing Overview

GWD provides everything you need to run a thorough QA (quality assurance) on your creative. It's important to QA your creatives before finalising development and uploading to AdWords.

QA is enabled through the creation of test feeds which provide examples of possible variations the ad may encounter when serving live. It's therefore up to the developer to create these samples to test the creative against, ensuring the ad functions correctly with all possible edge cases.

Adjust Sample Feed for QA

Format the feed

The downloaded feed samples are in the form of JSON text.

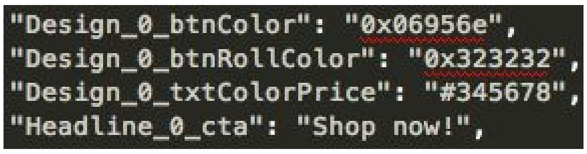
This is often hard to read after downloading a sample, and so the first step to creating alternative versions is to format the JSON text to become more readable: formatting the text so that each attribute displays on a separate line and are ordered to easily find and adjust each value.

There are many text editors available which can format JSON either natively or by added plugins or packages, as well as many free online tools.

Simplify the data

Once formatted it's advised to also strip out all the attributes which the creative is not making use of, and so include only those which the creative is designed to bind to. Delete each unused attribute from the Design, Headline and Product sections. This will usually significantly reduce the number of attributes in the JSON file.

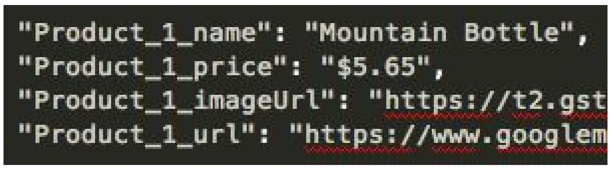
For example, a creative which only allows the user to change the text for the CTA button and the colors for the CTA button and price, should include the **Design** and **Headline** attributes which look like this:



```
"Design_0_btnColor": "0x06956e",
"Design_0_btnRollColor": "0x323232",
"Design_0_txtColorPrice": "#345678",
"Headline_0_cta": "Shop now!",
```

Then go through the **Product** attributes and do the same for as many feed items the creative is designed to serve.

For example, if the creative does not make use of the subTitle or description attributes, remove these and keep only the name, price, image, and url. If it's displaying 1-4 items only, then include all Product attributes from 0 - 3.



```
"Product_1_name": "Mountain Bottle",
"Product_1_price": "$5.65",
"Product_1_imageUrl": "https://t2.gst
"Product_1_url": "https://www.googlem
```

Create New Test Feeds for QA

In order to test the creative's handling of each attribute it maps to, the developer should create duplicates of this sample. The new sample should include attributes and values which are relevant to certain edge cases the live ad may encounter. These cases will be unique to each creative and campaign.

Each sample is then saved as a new JSON file and you should end up with a number of JSON files, each simulating a different edge case which the ad may encounter when serving live.

See the documentation on Feed Types above for more details about these attributes and what each one means.

Design and Headline sections

This is the part of the feed with provides the various options which the user inputs when uploading the creative into a new Ad in AdWords. These values will show as a preview when the ad is uploaded, and so can be verified by the user before saving.

Some examples

1. The creative handles not showing the CTA button
   - `"Headline_0_cta": ""`

2. The creative handles long variations of CTA button text
    - `"Headline_0_cta": "Book This Flight!"`
3. The price color is editable
    - `"Design_0_txtColorPrice": "#345678"`

Product section

This is the business feed data. Each feed item is represented by Product attributes, and the creative is served between 1 and 6, depending on various campaign settings.

A feed may contain up to a few thousand items, and a different selection of items are sent to the creative each time the ad is served.

This means there may be certain edge cases to the business feed data which need to be simulated with sample feeds, since the AdWords preview will only show the user the first **6 feed items**.

For example, testing a feed item without a price would have the price data removed for at least one of the items.

Common Edge Cases

1. Empty values
    - Remove the values.
2. Very long values
    - Add the longest possible data you expect to display, depending on the feed(s) the creative will be serving under.
3. Different prices
    - If your feed contains sales prices for only some items, the sample feed may not include this for any of the 6 items. This means it may be necessary to manually add the salePrice and regularPrice attributes.
4. Variable number of feed items
    - All Business Types may serve between 1 and 6 items, except for GMC (retail) with always serve 6.
    - This means any creative which is designed to display multiple items should properly display a variable number, and so samples should also be created with some **Product** attributes deleted.

Some examples

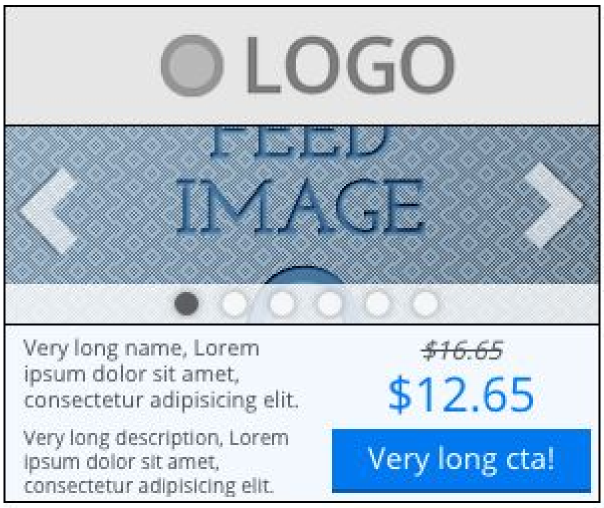
- Very long names:
    - `"Product_0_name": "Lorem ipsum dolor sit amet, consectetur adipisicing elit.”`
- Empty names:
    - `"Product_0_name": ""`
- Very long prices (perhaps serving with alternative currencies):
    - `"Product_0_price":"руб 12,149.90"`
- No sales price:
    - `"Product_0_salePrice": ""`
- Serve 1 - 5 items:
    - only include attributes starting with `“Product_0”` *(serves single item only)*
    - include attributes starting with `“Product_3”, “Product_2”, “Product_1”, “Product_0”` *(serves 4 items only)*

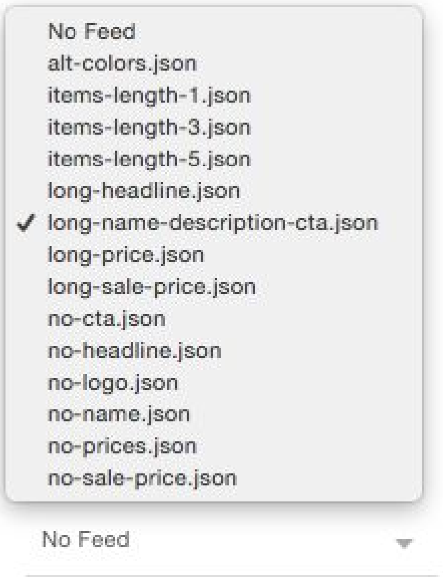
Testing Feeds for QA

Once you’ve created samples which cover all edge cases for your feed, QA is as simple as previewing the ad using each one and involves confirming that the relevant data is displaying correctly for each feed loaded.

Start by hitting the Preview button.

Then select each feed in the dropdown. This reloads with ad with the new test.


LOGO
FEED
IMAGE
Very long name, Lorem ipsum dolor sit amet, consectetur adipisicing elit.
~~$16.65~~
$12.65
Very long description, Lorem ipsum dolor sit amet, consectetur adipisicing elit.
Very long cta!



Testing Exits for QA

For an explanation on how the exit logic works for AdWords Dynamic Remarketing, see instructions on Handling Exits above.

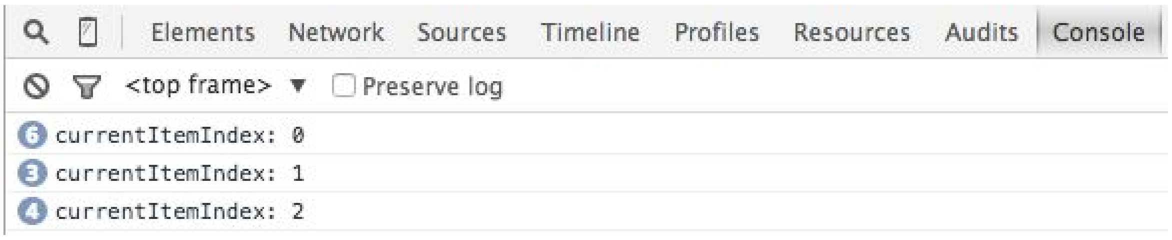
Depending on how the creative is designed, there are up to 3 types of exists to test for.

Items exit to the correct feed item url

For creatives designed to serve multiple items, the user must verify that clicks on the item areas direct to the correct item's url.

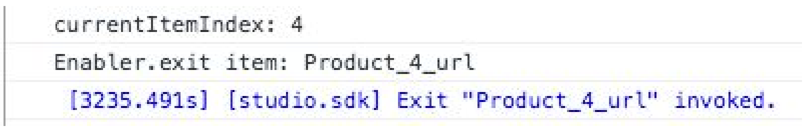
Open the developer tools in the browser and view the Console to check the logs.

Mouseover each item and ensure the index of each item is appearing in the **currentItemIndex** log.



```
6 currentItemIndex: 0
3 currentItemIndex: 1
4 currentItemIndex: 2
```

Clicking on each item should then log the current item's url attribute in the console.



```
currentItemIndex: 4
Enabler.exit item: Product_4_url
 [3235.491s] [studio.sdk] Exit "Product_4_url" invoked.
```

Failing this could be due to the following:

1. The item element does not have the 'js-item' class added
2. The common code's **init** function is not placed inside the page's **handleAdInitialized** function
3. The click event has not been added to the item element or swipegallery
4. The click event is not pointing to the **common.exitToItem** or **common.exitAuto** functions
5. Something is blocking the item element from receiving click events. This can be debugged using the developer

tools.

Non-item areas using exitAuto function: exit to the default url AND current item url

If the creative is making use of the **productClickOnly** attribute, and so allows the user control over the destination of the non-item areas, then the areas which do not display the feed items should click to the default url or the url for the currently active item depending on the value of **productClickOnly.**

This requires these areas have click events pointing to the **common.exitAuto** function. This should be tested by the following:

1. Create a new sample feed with Headline_0_productClickOnly to “false” and one with the value set to “true”.
2. Preview the creative and switch between these feeds. Click on the non item areas and check the console logs out the following:
    a. productClickOnly = false: **“Enabler.exit default”**
    b. productClickOnly = true: **“Enabler.exit item: Product_0_url”**

Non-item areas using exitToDefault function: exit to the default url ONLY

If instead the creative does not allow the user control over this option and the creative is not making use of the **productClickOnly** attribute, these areas should point to the **common.exitToDefault** function. Clicking these areas should therefore only log: **“Enabler.exit default”**.

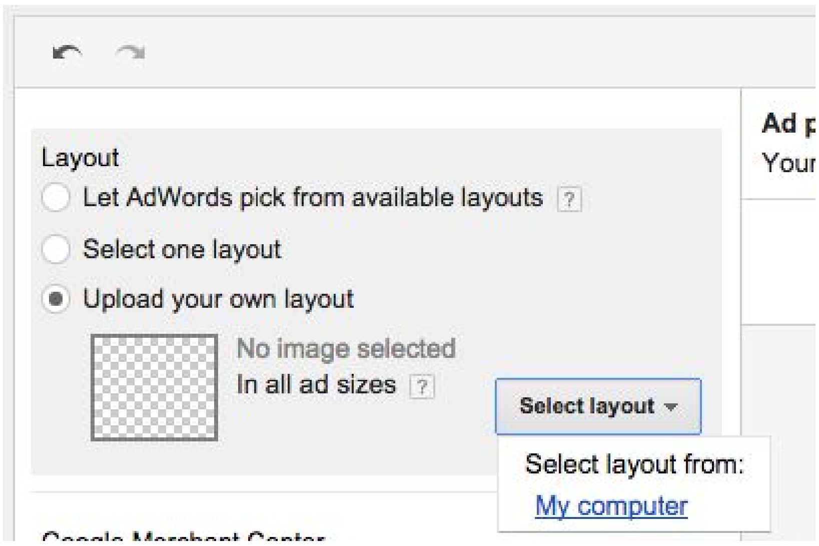
Publish from Google Web Designer

1. Click the Publish button

2. Select Publish locally instructions for settings.

Upload into AdWords

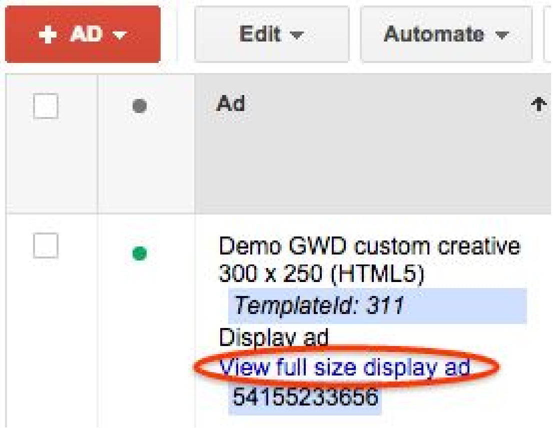
1. Go into AdWords and create a new ad.
2. In the In the Ad builder tool, choose the option to **Upload your own layout**


Layout
Let AdWords pick from available layouts ?
Select one layout
[✓ Upload your own layout]
No image selected
In all ad sizes ?

3. Browse to your published zip and upload. Repeat for each size.

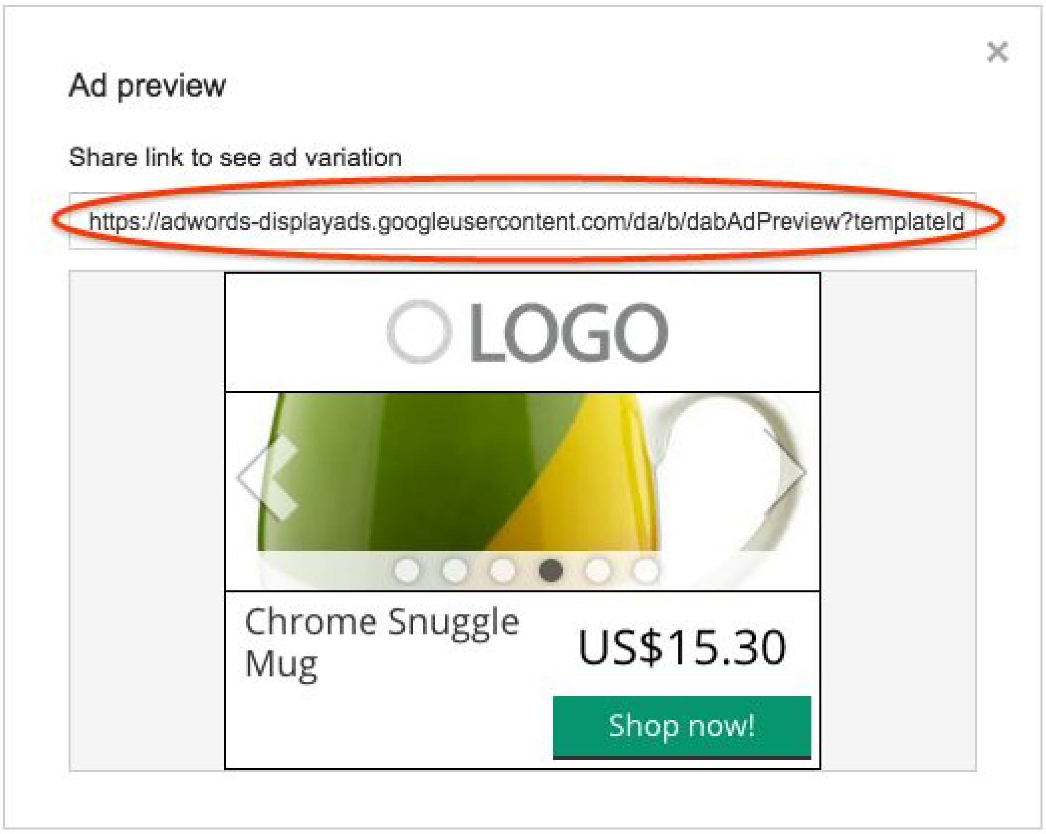
External previews

Note that the exit will not fire when previewed in the upload tool. To see the exits function, you must save the ad and preview using the “View full size display ad” external preview link.



Demo GWD custom creative
300 x 250 (HTML5)
TemplateId: 311
Display ad
View full size display ad
54155233656

This provides a preview link which can be copied and viewed directly in the browser where the ad functions as it will when being served, including the exits.

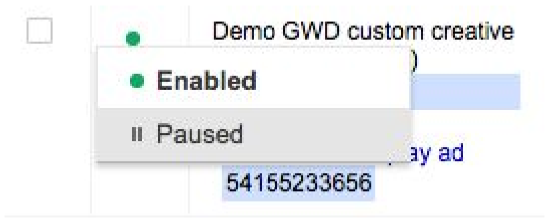


Ad preview

Share link to see ad variation

https://adwords-displayads.googleusercontent.com/da/b/dabAdPreview?templateId

Once uploaded and saved, the ad can be paused and re-enabled..


Demo GWD custom creative
● Enabled
II Paused
ay ad
54155233656