Meme Generator

What is the Meme Generator?

Meme Generator allows user to upload image and add customizations to images. It operates in HTML5 supported, so your images are created instantly.

How can I customize my meme?

* You can move and resize images dragging them around.
* You can customize the font color, outline color, and outline width just to the right of where you type your text.
* You can rotate, flip and also zoom in /zoom-out the meme. You can create overlay of another image as overlay over the uploaded image.

Yes! The Meme Generator is flexible for many purposes. By uploading images and using all the customizations.

Below are the working examples of the Meme-Generator:

[http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/filter-meme-poc/#](http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/filter-meme-poc/)

<http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/meme-generator-poc/>

https://github.com/WEAD2C/meme-generator/tree/postcard-meme

Pre-Requisites:

Basic knowledge on html5 canvas,javascript and jQuery

Developers Guide:

Step 1: Copy the code from the stage url [http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/filter-meme-poc/#](http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/filter-meme-poc/) or clone/download from the below mentioned git path:

<https://github.com/WEAD2C/meme-generator/tree/filter-meme-poc>.

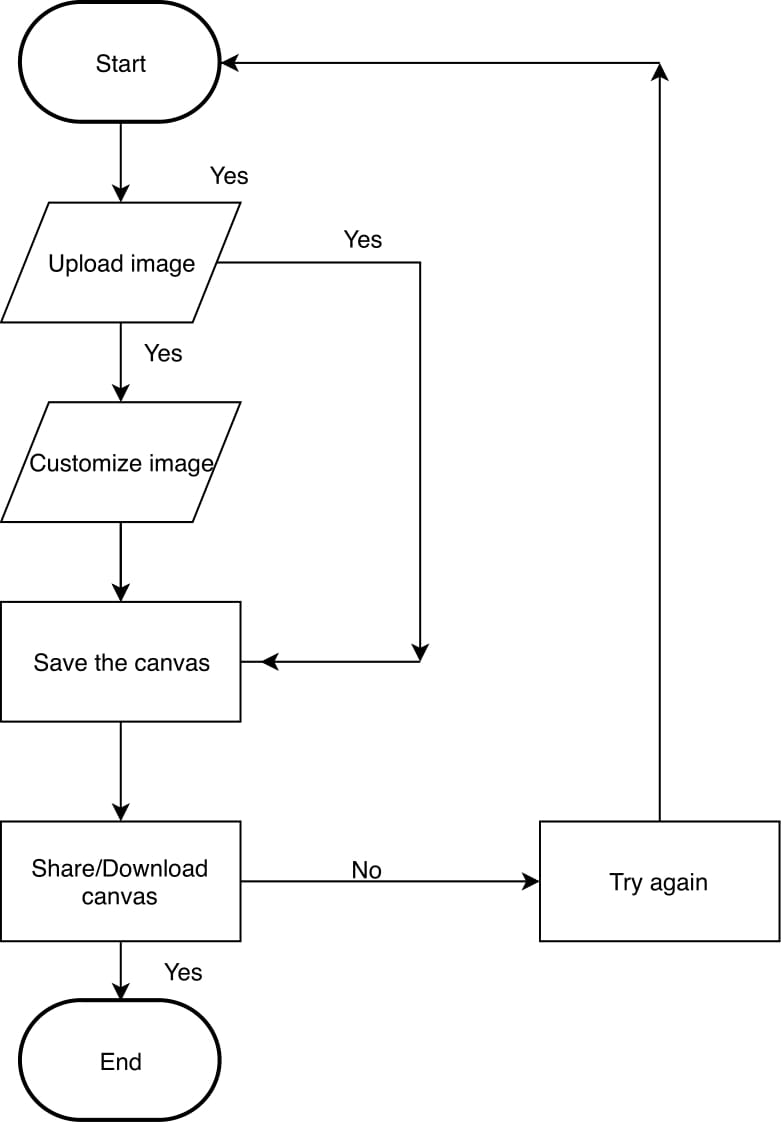
Step 2: Change the HTML as per the provided wireframe.

Step 3: Update the config.php file for the og params and metadata.

Step 4: site-script.js holds all the functionality of the meme. Customize as per the requirement.

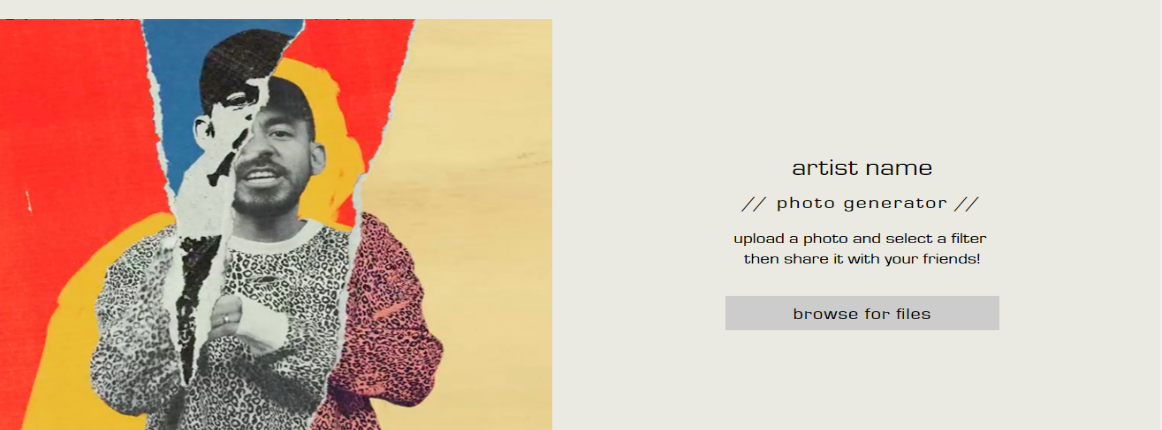
Note: We need write permissions of the output folder in order to save the final image. Use iswritabble.php to check whether the folder has read/write permissions.

Flowchart of basic Meme-Generator:



Meme Generator Code Walk-through:

**STEP 1**:

User uploads the picture. 

Here when user clicks on the browse for files button, user will be allowed to open the file upload window.

Dev Side Implementation:

Specify that the server accepts only image files in the file upload:

<form action="/action\_page.php">  
   <input type="file" name="pic" accept="image/\*">  
   <input type="submit">  
</form>

**STEP 2:**

The uploaded image is read using the FileReader instance and then saved in new Image instance using new Image() constructor.

The **FileReader.onload** property contains an event handler executed when the [load](https://developer.mozilla.org/en-US/docs/Web/Events/load) event is fired, when content read with [readAsArrayBuffer](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readAsArrayBuffer), [readAsBinaryString](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readAsBinaryString), [readAsDataURL](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readAsDataURL) or [readAsText](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readAsText) is available.

Here we use readAsDataURL():

The readAsDataURL method is used to read the contents of the specified [Blob](https://developer.mozilla.org/en-US/docs/Web/API/Blob) or [File](https://developer.mozilla.org/en-US/docs/Web/API/File). When the read operation is finished, the [readyState](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/readyState) becomes DONE, and the [loadend](https://developer.mozilla.org/en-US/docs/Web/Events/loadend) is triggered. At that time, the [result](https://developer.mozilla.org/en-US/docs/Web/API/FileReader/result) attribute contains  the data as a [data: URL](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Data_URIs) representing the file's data as a base64 encoded string.

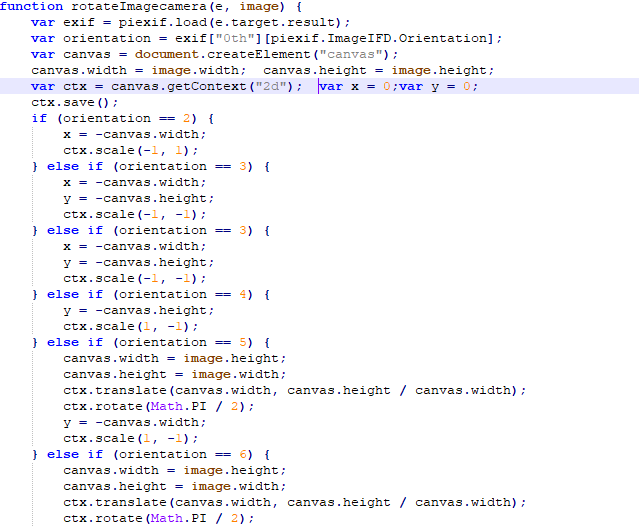


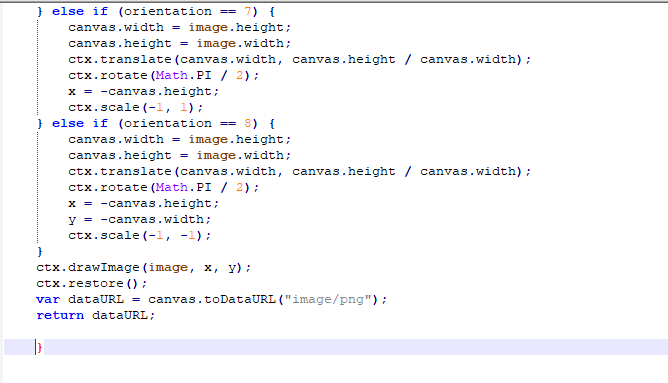
Step 2.1

RotateImageCamera is used for devices to avoid the unwanted rotation of the images.For eg:When user captures an image in ios devices the image orientation is Changed horizontally once uploaded.So based on the uploaded image orientation we uniform the image to vertical orientation using piexif.js and exif.js

The orientation of the camera relative to the scene, when the image was captured. The relation of the '0th row' and '0th column' to visual position is shown as below.

|  |  |  |
| --- | --- | --- |
| Value | 0th Row | 0th Column |
| 1 | top | left side |
| 2 | top | right side |
| 3 | bottom | right side |
| 4 | bottom | left side |
| 5 | left side | top |
| 6 | right side | top |
| 7 | right side | bottom |
| 8 | left side | bottom |

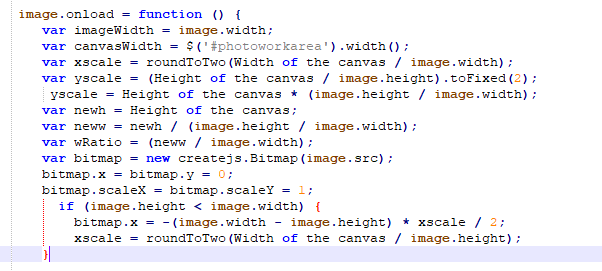




We use CreateJS which is:  
**A suite of modular libraries and tools which work together or independently to enable rich interactive content on open web technologies via HTML5.**

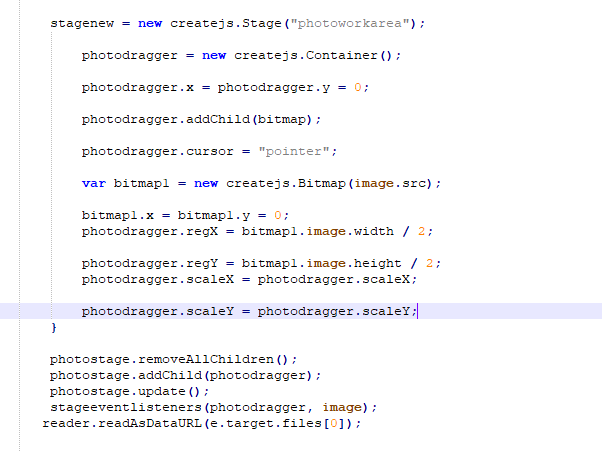
On loading the image we create a Bitmap instance. A Bitmap represents an Image, Canvas, or Video in the display list. A Bitmap can be instantiated using an existing HTML element, or a string.

In the below code snippet we are scaling the image based on the canvas height and width in order to avoid the unnecessary cropping of the image loaded.

Step 2.2

Stage is the root level container and A Container is a nestable display list that allows you to work with compound display elements which allows to move the individual parts relative to each other adding the drag feature.

Please refer <https://createjs.com/docs/easeljs/classes/Stage.html> , <https://createjs.com/docs/easeljs/classes/Container.html> for more on Stage and container classes.



 stage.update is used redraw the stage

**STEP 3:**

User can customize the image by adjusting the zoom, rotate and flipping the image.

Here in the example we are trying to add a mask/overlay filter image to the user uploaded image.

filterimage = new Image();

filterimage.src = $(this).find('img').attr('src');

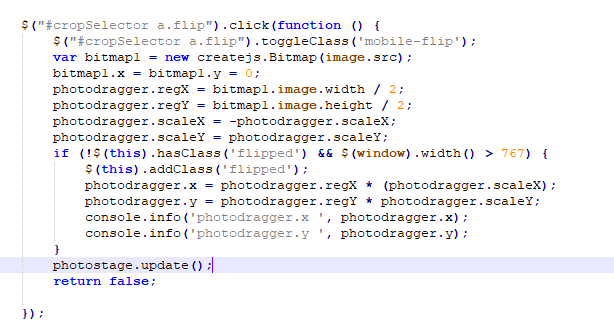
var maskImageObj = new Image();

maskImageObj.src = filterimage.src;

maskImage = new createjs.Bitmap(maskImageObj.src);

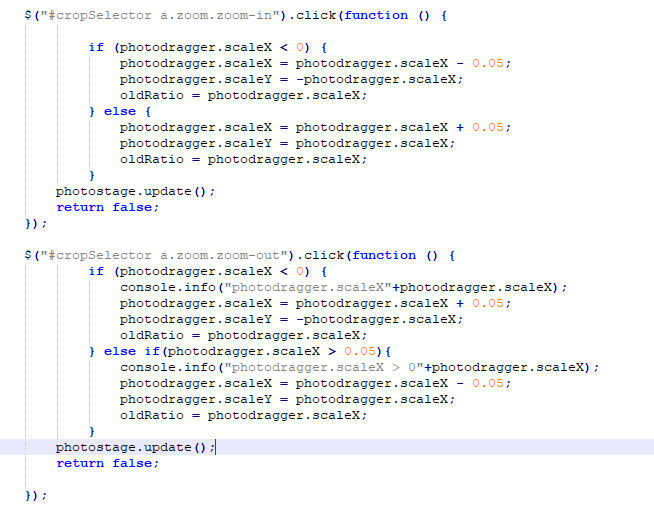
Once the customizations are applied to the image we need to update the stage using  
 photostage.addChild(maskDragger);

photostage.update();

User is allowed to rotate the image, flip the image,zoom in/zoom out.  
a)For Flipping the image horizontally ,Have a look at the snippet  


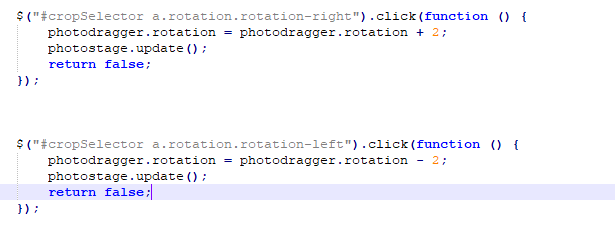
By using the negative scaleX we can flip the image horizontally.

b)Zoom in/ Zoom-out:



Allows user to zoom in or zoom out by scale of 0.05.

c) Rotate clockwise/Anticlockwise: Allows user to rotate clockwise or anticlockwise by 2 degree’s.

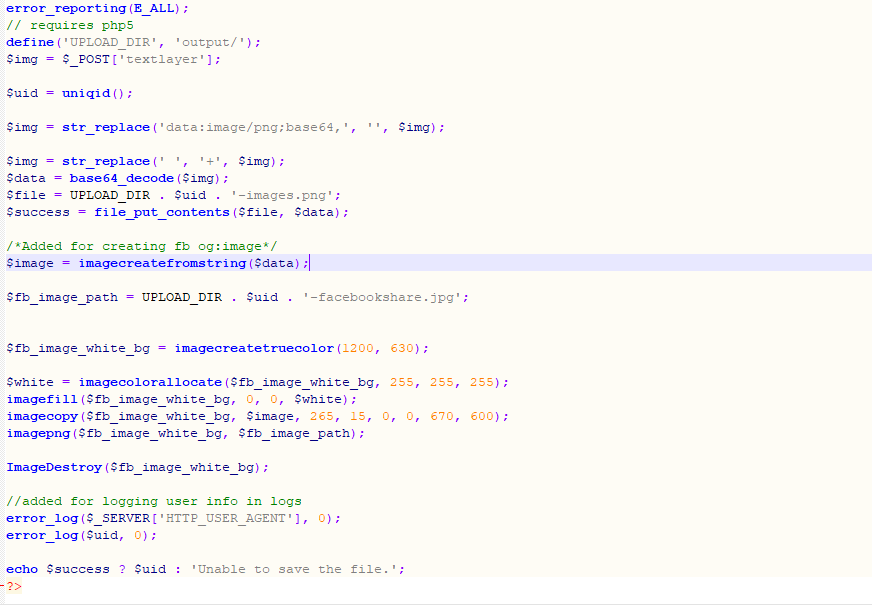


d) Custom message on the canvas:  
 var c=document.getElementById('myCanvas');  
 var ctx=c.getContext('2d');  
 ctx.font='30px Arial';  
 ctx.fillText(**'Hello World',20,50,80**);

**STEP 4:**

Create the final image after the customizations and store the final image in the output folder of the application.

Here is the glimpse of the save.php file which creates unique id and posts the folder with the unique name created by using the uniqid().





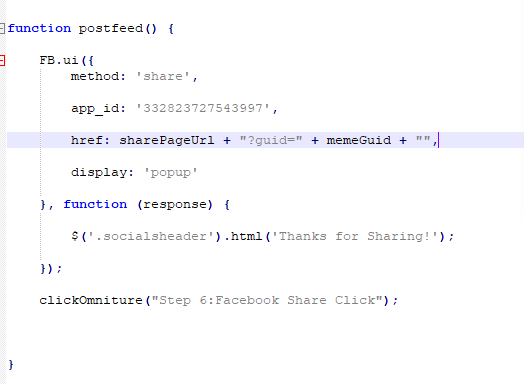
The response gives us the unique id which is the name of the folder created in the output folder. Here we can update the share links and share our meme across platforms like Facebook, twitter etc. and also can download the image to the local by using below mentioned simple code

<a href=”http://staging.weate.ch.stage1.535e.blackmesh.com/wbr/filter-meme-poc/output/5bffbbbd68873-images.png” target=”\_blank" download=”download.png" >download</a>

The download attribute on the ‘a’ tag causes the browser to open a save dialog instead of navigating to the image before the browser has had time to activate the link

**STEP 5**. User can now share the canvas across social networks like Facebook, twitter.



  
Above function is used to share the generated image on Facebook platform. The key parameters are the app id which needs to be created in developers.facebook.com.