How to upload HTML5 banner to gDE.

New approach towards delivering HTML5 banners

Currently it is necessary to pack all creative assets inside an html file. It means that all JS, CSS and images will be included inside one file. To do that we use tool called Inliner. Further in the text you will find information how to do it.

Soon we plan to implement a solution which would allow for uploading zip file with index.html file and all assets, so it won't be necessary to make inline creatives anymore.

The file created in this way, you should treat similarly to swf files.

HTML file has to contain our code for proper click tracking measurement. This situation is similar to the implementation of clickTag in swf files.

HTML5 codes are delivered asynchronously and we encourage the use of asynchronous placements in gDE to deliver them (Placement parameters, Enviroment -> Browser (async))

Implementation codes

> Warning
> Implementation codes depend on URL fragment identifier, it means that your creative must not depend on this property, at least not on document load.

A tag, which should surround clickable place in banner
 	
ID of the element is important, if you modify it, you need to change it also in succeeding functions
	`<a id="creativelink" target="_blank">`
	`  <img src="..." />`
	`</a>`

Function which should be added in a creative to measure clicks before </body>	`<script type="text/javascript">`
	`  var parsed = (document.location.href.split('#')[1]||'').split('&');`
	`  var params = parsed.reduce(function (params, param) {`
	`  var param = param.split('=');`
	`  params[param[0]] = decodeURIComponent(param.slice(1).join('='));`
	`  return params;`
	`  }, {});`
	`</script>`

Click measurement implemented after previous script but before </body> Be careful about the variable in brackets, it has to be this same as given in the 1st row in this case it's 'creativelink'	`// change link href` `document.getElementById('creativelink').href = params.clickTag;`

MultiClicktag option (you need to make multiple surrounds of <a> tag) and then properly use it with this function	document.getElementById('creativelink0').
	href = params.clickTag0;
	document.getElementById('creativelink1').
	href = params.clickTag1;

Interaction implementation. You need to define a proper interaction-trigger	document.getElementById('interaction-trigger').addEventListener('mouseenter',
	function() {
	
	window.parent.postMessage('my_interaction_function', '*');
	});

Short example:

CreativeCode:

```
<html>
  <head>
  </head>
  <body>
    <img src="link-to-picture.png" />
  </body>
</html>
```

Surrounding clickable element:

```
<html>
  <head>
  </head>
  <body>
     <a id="creativelink" target="_blank">
        <img src="..." />
     </a>
  </body>
</html>
```

Inserting proper function to measure clicks:

```
<html>
  <head>
  </head>
  <body>
     <a id="creativelink" target="_blank">
        <img src="..." />
     </a>
     <script type="text/javascript">
        var parsed = (document.location.href.split('#')[1]||'').split('&');
        var params = parsed.reduce(function (params, param) {
           var param = param.split('=');
           params[param[0]] = decodeURIComponent(param.slice(1).join('='));
           return params;
        }, {});

        // change link href
        document.getElementById('creativelink').href = params.clickTag;
     </script>
  </body>
</html>
```

Case with measuring interactions (be aware that there is no interaction-trigger element, which should be responsible for interaction measurement)

```
<html>
  <head>
  </head>
  <body>
     <a id="creativelink" target="_blank">
        <img src="..." />
     </a>
     <script type="text/javascript">
        var parsed = (document.location.href.split('#')[1]||'').split('&');
        var params = parsed.reduce(function (params, param) {
           var param = param.split('=');
           params[param[0]] = decodeURIComponent(param.slice(1).join('='));
           return params;
        }, {});

        // change link href
        var creativeLink = document.getElementById('creativelink');
        creativeLink.href = params.clickTag;
        // add interaction
        document.getElementById('interaction-trigger').addEventListener("mouseenter", function() {
           window.parent.postMessage('my_interaction_function', '*');
        });
     </script>
  </body>
</html>
```

Requirement for ZIP file:

- Must have at least one HTML file
- Should have index.html file
- Main HTML file must be at top level or in root folder
- Can have root folder
- Must not have multiple root folders
- Can have multiple HTML files

Example ZIP folder structure with root folder:

```
+ creative.zip
+-+ /rootfolder
  +-+ index.html
    + image.png
    + style.css
```

Example ZIP without root folder:

```
+ creative.zip
+-+ index.html
  + image.png
  + style.css
```

Example of invalid ZIP with multiple root folders:

```
+ creative.zip
+-+ /rootfolder1
| +-+ index.html
|   + image.png
|   + style.css
|
+-+ /rootfolder2
  +-+ index.html
    + image2.png
    + style2.css
```

Working templates

Currently we have 3 templates available which are able to deliver HTML5 file. Those are:

- Toplayer [N] [ASYNC]
- Expand [N] [ASYNC]
- Billboard/Banner/Box [N] [ASYNC]

Real case example:

Example banner

First we have to open index.html file, and find the place which is a clickable area.

In this case we can find:

```
<div id="tui-animation"><div id="tui-ct"></div>
<div id="tui-8"></div>
<div id="tui-7"><div class='tuirect'></div></div>
</div>
```

So first we surround with our <a> tag.

```
<a id="creativelink" target="_blank">
 <div id="tui-animation"><div id="tui-ct"></div>
 <div id="tui-8"></div>
 <div id="tui-7"><div class='tuirect'></div></div>
 </div>
</a>
```

Then we need to implement other necessary scripts.

In this case we need to include before </body> the following code:

```
<script type="text/javascript">
    var parsed = (document.location.href.split('#')[1]||'').split('&');
    var params = parsed.reduce(function (params, param) {
    var param = param.split('=');
        params[param[0]] = decodeURIComponent(param.slice(1).join('='));
        return params;
    }, {});

    // change link href
    var creativeLink = document.getElementById('creativelink');
    creativeLink.href = params.clickTag;
</script>
```

Full code of our index will look like this:

```
<!doctype html>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<title></title>

<!-- TWEENUI INCLUDES START -->
<script type="text/javascript" src="js/plugins/CSSPlugin.min.js"></script>
<script type="text/javascript" src="js/easing/EasePack.min.js"></script>
<script type="text/javascript" src="js/TimelineLite.min.js"></script>
<script type="text/javascript" src="js/TweenLite.min.js"></script>
<script type="text/javascript" src="js/TweenUI.js"></script>
<link type="text/css" href="css/TweenUI.css" rel="stylesheet">
<!-- TWEENUI INCLUDES END -->

</head>
<body style="margin:0px;padding:0px;">

<!-- TWEENUI ANIMATION START  -->
<a id="creativelink" target="_blank">
<div id="tui-animation"><div id="tui-ct"></div>
<div id="tui-8"></div>
<div id="tui-7"><div class='tuirect'></div></div>
</div>
</a>
<img src="img/1388d5a5b2837dfd84daaded3b1efc52.png" onLoad="tweenui.imgcntd(this)"
id="tui-img-8" />
<noscript>Activate javascript to view TweenUI <a href="http://tweenui.com">mobile banner
creator</a> animations.</noscript>
<script type="text/javascript">
tweenui.init();
</script>
<!-- TWEENUI ANIMATION END -->


<script type="text/javascript">
     var parsed = (document.location.href.split('#')[1]||'').split('&');
     var params = parsed.reduce(function (params, param) {
        var param = param.split('=');
        params[param[0]] = decodeURIComponent(param.slice(1).join('='));
        return params;
     }, {});

// change link href
document.getElementById('creativelink').href = params.clickTag;

</script>

</body>
</html>
```

We can now pack whole creative into folder and upload it to gDE. Link to updated creative

Part connected with Inliner is obsoleted.

Currently we can simlpy upload zip file.

Hint for swiffy creatives:

Given fragment:

```
<script>
  var stage = new swiffy.Stage(document.getElementById('swiffycontainer'),
      swiffyobject, {});
  stage.start();
</script>
```

Change to:

```
<script>
  var stage = new swiffy.Stage(document.getElementById('swiffycontainer'),
  swiffyobject, {});
  stage.setFlashVars(document.location.hash.split('#')[1]);
  stage.start();
</script>
```

> Remember to check the name of the clickTag variable in you swiffy-converted creative and set the clickTag name in gDE interface to be the same. For example in swiffy the variable for clicks can be "clickTAG" and since the code is case sensitive you need to make sure that the name in gDE is also "clickTAG" and not "clickTag"

Inliner (obsolete)

Currently we have possibility to upload zip files to gDE, that's why there is no need to inline creative anymore.

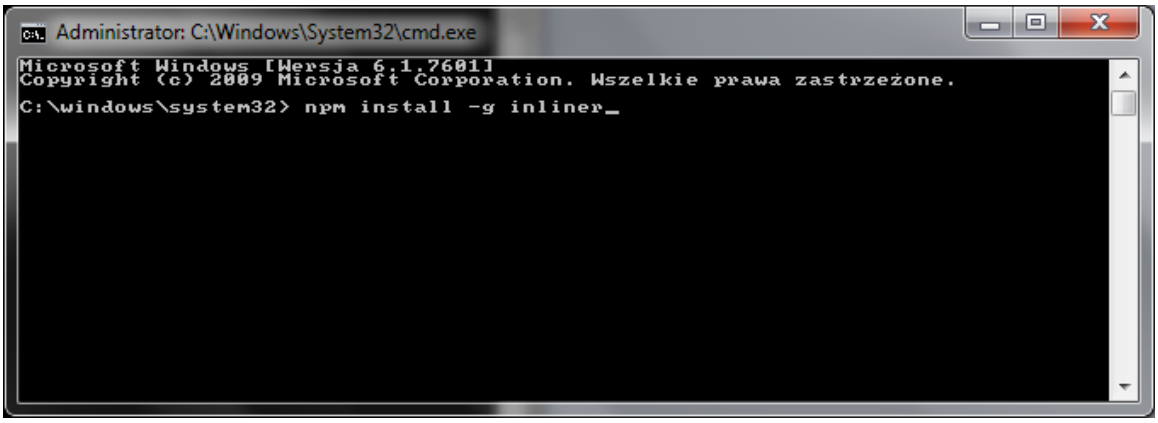
Inliner (https://github.com/remy/inliner) is a program which turns your page into a single html file.

To use it. first you need to install NodeJS. https://nodejs.org/download/

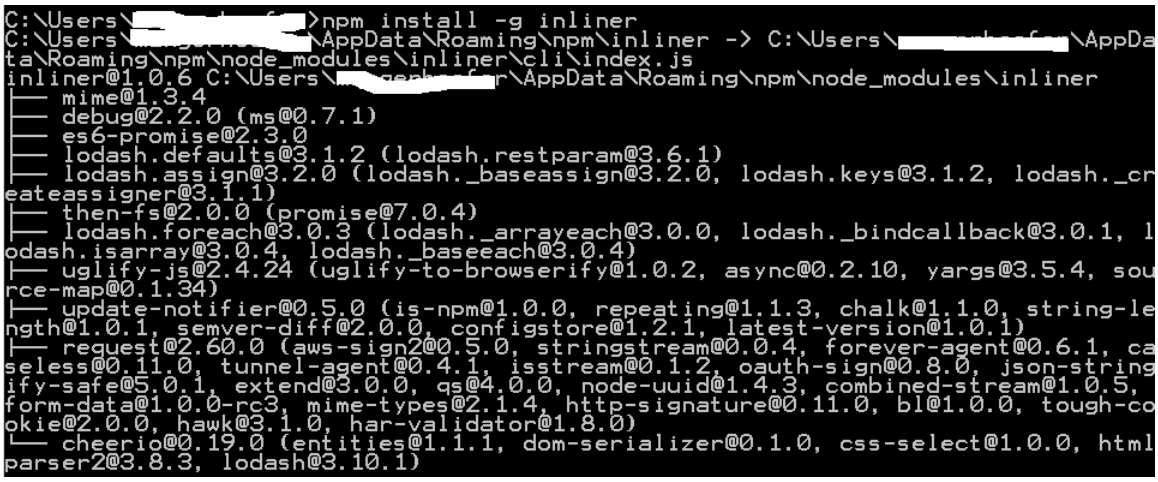
Then in Windows you need to open Command Line Terminal with administrator rights.

After that execute command:

```
npm install -g inliner
```



After clicking enter you should see the following output:



After proper installation you can check if inliner is working by executing command:

```
inliner -h
```

You should see the following output: