HTML5
FOR DIGITAL ADVERTISING

VERSION 2

RELEASED APRIL 11, 2016

This document has been developed by the IAB Tech Lab and the Mobile Marketing Center of Excellence

The HTML5 for Digital Advertising (HTML5_DA1) document was created by a working group of volunteers from IAB member companies.

The HTML5 Digital Advertising Working Group was led by:

- Sarah Hunt (Adobe)
- Luke Wilson (Millenial)
- Cory Hudson (AOL)

The following IAB member companies contributed to this document:

- Adobe
- Amazon
- AOL
- CBS Interactive
- Celtra
- Flashtalking
- Google
- Monotype
- Pointroll

The following IAB members were part of the HTML5 Digital Advertising Working Group:

AOL	Flite	PGA TOUR
Aarki	FreeWheel	PointRoll
Adobe	Gamut	SpinMedia
Amazon	Google	SpotXchange
AOL	Hearst Magazines Digital Media	The Business Insider
Belo	Hipcricket	The Weather Channel
BrightRoll	InMobi	Time Inc.
CBS Interactive	Mansueto Ventures	Tribune Company
Celtra	Medialets	TruEffect
Complex Media	MediaShift	Turner Broadcasting System
Cox Media Group	Monotype	Vdopia
ESPN.com	Mixpo	Xaxis
FindTheBest.com	NBCUniversal	Yahoo
Flashtalking	Pandora	YuMe

The IAB lead on this initiative was Shailley Singh.

Contact shailley@iab.com to comment on this document. Please be sure to include the version number of this document (found on the bottom right corner on this page).

About the IAB's Tech Lab and Mobile Marketing Center of Excellence

The IAB Tech Lab spearheads the development of innovative and scalable technical standards, creates and maintains a code library to assist in rapid, cost-effective implementation of IAB standards, and establishes test platforms for companies to evaluate the compatibility of their technology solutions with IAB standards, which for almost 20 years have been the foundation for interoperability and profitable growth in the digital advertising supply chain. More details can be found at: http://www.iab.com/organizations/iab-tech-lab/

The IAB Mobile Marketing Center of Excellence, an independently funded and staffed unit inside the IAB, is charged with driving the growth of the mobile marketing, advertising and media marketplace. The Mobile Center devotes resources to market and consumer research, mobile advertising case studies, executive training and education, supply chain standardization, creative showcases and best practice identification in the burgeoning field of mobile media and marketing. Our agenda focuses on building profitable revenue growth for companies engaged in mobile marketing, communications and advertising, and helping publishers, marketers and agency professionals understand and leverage interactive tools and technologies in order to reach and influence the consumer. More information can be found at: http://www.iab.com/topics/mobile/ http://www.iab.net/mobile_advertising_committee

This document is on the IAB website at:
http://www.iab.com/HTML5http://www.iab.net/HTML5_Digital

TABLE OF CONTENTS

Executive Summary

HTML5 for Digital Advertising is a guide to help ad developers produce ads in an HTML5 format that meet IAB creative guidelines for desktop and mobile display ads.

Historically, ad developers have used Adobe Flash™ tools to create an interactive ad experience online. Flash allowed developers to create animated ads with interactive elements without having to worry too much about code. These ads required a browser plug-in to execute Flash ads. However, in recent years, many browsers have announced reducing support for Flash content to some degree or another. Some browsers are discontinuing support altogether. In addition, some ad networks have announced discontinuation of Flash Ads timelines.

This increasing lack of support for Flash-formatted ads and the release of HTML5 in late 2012 has prompted the digital advertising industry to look at working with HTML5 as a replacement technology for Flash. However, ad developers who are used to working with Flash visual development tools find moving to the more code-heavy HTML5 format a daunting transition. In addition, the overhead in ad operations is overwhelming as ad technologists work with ad developers to format HTML5 ads to meet publisher requirements.

While working with HTML5 is complex, it's not necessarily more complex than working with Adobe Creative Cloud (Flash and other Adobe animation tools). In fact, a number of visual development tools have been released in recent years to produce interactive content in an HTML5 format without having to deal much with the code. Despite the options available for ad developers, working with HTML5 is different than working with the traditional animation tools that use Flash plug-ins for execution.

This document provides guidance on ad development in HTML5, including tips for generating, packaging, and testing ads as well as how to work with ad operations to ensure ads will load and work as expected in a live campaign.

Seamless operation with HTML5 ads continues to prove challenging, but adhering to the practices outlined in the document will improve ad development and reduce ad operation overhead.

Audience

This document is designed for ad designers, but other parties in the ad supply chain, especially those who manage file uploads and ad trafficking, may benefit from reviewing the guidance provided.

1 Overview

Released as a Final recommendation by the World Wide Web Consortium (W3C) in October 2014, HTML5 is the latest update to the Hypertext Markup Language (HTML) that includes new semantic tags for features such as video, audio, canvas, and other design features. HTML5 has grown into an industry buzzword that has come to encompass all the various web technologies and APIs that work together to execute animation and other interactions.

While HTML5 is code-heavy, several visual development tools have been released to aid designers in developing interactive content without dealing with code. Previously known as "Flash," the Adobe Creative Cloud animating tools also generate HTML content, which is an alternative to the Flash output that many browsers no longer accept. Despite the growing popularity of these tools and increased ease of use, special considerations are necessary in the ad development process to meet publisher and industry standards.

This document provides an overview of the IAB Display Creative Guidelines for Desktop and Mobile, tips on ad development that help meet these guidelines, and guidance packaging and testing your ad.

1.1 Designer's role vs. ad technologist

As a designer, your role is to create a compelling message that gets people's attention. But if your message can't break through the technology to reach enough people, then it loses a great deal of effectiveness. Designing to overcome some of these technical barriers is part of your job as the designer.

However, an ad technologist can work with you to prepare the ad for its journey through the ad storage, delivery, and ad trafficking execution systems. Ideally, designers and ad technologists work together to create an ad package that works as it was designed to.

2 IAB Display Creative Guidelines

In 2015 the IAB Display Creative Guidelines (Creative Guidelines) was updated to reflect support for HTML5 ads. While the Creative Guidelines provide a reference for details, an overview is provided here in the context of HTML5 ad development.

2.1 Display in Desktop vs. Mobile

Three key factors to be aware of when designing ads that might display in either a desktop or mobile environment are: screen size, connection data rate, and run-time environment.

In a desktop environment, you can count on a screen size that is generally large enough to display any ad, but in a mobile environment, screen size varies from device to device. Depending on campaign strategy and ad serving technology, you may need to design more than one version of the ad to work in multiple environments.

Connection data rates may be limited with mobile devices, so file weight is an important factor to consider when an ad will display in mobile devices. Even in desktop environments with high data rates, heavy file-weighted ads may delay page load and cause user drop-off. The Creative Guidelines provide file size limits that help balance ad load and page load performance.

In desktop display, most ads are executed in a browser. However, in a mobile device, ads may be executed within a native application. In this case, ads developed with rich interaction may need an API like MRAID to be executed.

Developing an ad that accounts for screen size, connection data rate, and run-time environment results in an ad that can display almost anywhere. The following overview of ad units covered in the Creative Guidelines provides general information to help you account for these key factors.

2.2 Ad Units Overview

The ad units covered in the Creative Guidelines include ad formats for desktop display and mobile environments. Both categories include ad formats for simple, non-rich media ads and ads designed for more complex interactions. Guidance is provided for file weights at three different load stages, animation and video lengths, video formats, mobile considerations, general ad requirements and other unit-specific notes.

The ad units are broken down as follows:

2.2.1 Simple ads

In general, simple ads are your basic image ads that may include light, 15-second animation. While initial file load is close to that of rich media units, file weights are limited to only what is allowed upon initial file load. No subsequent loads, expansion, or video is allowed. A variety of simple ad formats are available in the following ad unit groups:

- Universal Ad Package (UAP)
- Other Ad Units
- Image Ads (mobile)

2.2.2 Rich media ads

Ads that require expansion, rich interactions, video, and additional file load weights fall under the selection of ad units defined in the following ad groups:

- Display Rising Stars
- Rich Media Guidance
- Mobile Rising Stars (mobile)
- Rich Media/Expand (mobile)

Before designing an HTML5 ad, review the Creative Guidelines, check with the ad serving vendor or publisher for additional ad requirements, and consider all the guidance offered in this document so that you can plan for a smooth development process and ultimately a more successful ad experience.

3 Developing Your HTML5 Ad

Developing in HTML5 requires a different skill set than developing a Flash ad. Flash ads are executed using a plug-in that users install in their browsers. Because of this plug-in, you could generate one file and expect a consistent experience across browsers. With HTML5, you are generating a collection of files with data that browsers interpret directly.

Because browsers are all designed differently, HTML5 ads display differently in each browser. These variations in ad execution require knowledge of the feature set you use to design your ad and plan accordingly for execution in different browsers. Even though your creative development software may generate HTML content, you need to know how browsers handle that content so that you can make adjustments that ensure that ad functions as it's supposed to.

The Creative Guidelines offer a baseline set of file requirements your ad should meet, but this section provides more information on how to meet those requirements.

3.1 Working with HTML5

HTML5 Ads use code that a browser or embedded web view (mobile app) processes. A number of software tools that use an integrated development environment (IDE) will allow you to work with a visual interface for producing interactions. You can then use the IDE to generate the HTML5 files needed for ad execution. Visit the IAB HTML5 Wiki for list of HTML5 IDE tools.

3.1.1 The index.html File

Within the HTML5 files generated for the ad, there should be one HTML5 document with the file name: index.html. The index file precedes all content code with the `<!DOCTYPE html>` tag. This tag identifies the file as an HTML5 file. Tags to call out the `<html>` code and `<body>` content code are also included in the file.

While some vendors use a specific naming convention for the name of the file that references other files for the ad, "index.html" is the common convention. The more this file name is used as an HTML file name convention for ads, the easier it is for most systems to find the relevant file and execute accordingly.

The index.html file naming convention is only necessary in a .zip file that represents a single ad and only in cases where multiple HTML files are included. No recommendations are made for file naming in a package of multiple ads or for an ad file that is contained within only one HTML file.

3.1.2 The Code

An HTML5 IDE generates code as you develop the ad creative. You should have access to a panel or window that displays the code, and you should be able to adjust this code in order to meet certain ad standards that the IDE cannot produce.

For example, declaring ad dimensions is not something the IDE will do. In the Dimensions section of this document, a recommendation is given for declaring ad dimensions in a standard way that most systems will recognize. Another example includes a standard way to provide

multiple video files in ads that use video. Wherever relevant, this document provides sample HTML code when you need to inspect or adjust the generated code.

3.2 Dimensions

One of the powerful characteristics of HTML5 is how easy it is to generate ads that adjust to the container size dynamically, similar to a web-page viewed on browsers of different window sizes. However, in most cases the ad dimensions should match the placement dimensions. When the HTML ad content is contained within an HTML `<head>` tag, you can define ad dimensions by adding a `<meta>` tag named `"ad.size"` within the HTML `<head>` tag as in the example below:

```
<meta name="ad.size" content="width=300,height=250">
```

If the width and height are left undefined, the publisher may either disregard the ad or consider the creative to be responsive in size and react according to its policy on responsive ad sizes. Check with the ad server or publisher on their policy for ad unit dimensions.

3.3 Expanding ads

Expanding ads involve some extra technical effort to display. When an ad serves to a web page, it is often served into an iframe (an html window within the page). Within the iframe, ad functionality is limited, and the communication to the webpage that is required to make an expanding ad work is blocked. Expanding ads and other ads that involve complex interactions must work around this iframe limitation.

In most cases, the ad serving vendor you work with provides a platform and possibly a template that handles the expansion operation. For example, in some cases you may provide two creative: the collapsed version and the expanded version. You would then upload those two versions and the ad server would handle the expansion. However, each vendor may handle this operation differently. Check in with your ad serving technology vendor about how to design an expanding ad that works with their setup.

3.4 Text and Fonts

Text content can be presented as part of the creative in different ways, using either text and fonts, by encapsulating text as part of a scalable vector graphic (SVG), or in rasterized images. With advances in CSS and widespread adoption of webfonts, representing text content using HTML5 text elements and webfonts, provides the most flexible option.

Some of benefits of using text include:

- Improved legibility and readability of text displayed on devices with various screen sizes and resolutions
- Reduced ad unit weight - the text element accompanied by a small web font (or one using a device font or a cached font resource) would be universally scalable using less bandwidth than a set of images
- Guaranteed accessibility of the ad content (e.g. when using screen readers)
- Guaranteed text look and feel preserving a brand identity of advertised products and services

- Preserving text and content layout
- Enabling dynamic content updates and personalization
- Giving content authors greater typographic controls and language choices outside of what's available with default system fonts;
- Sharing web font resources among multiple ad units, and
- Using automatic browser caching of font resources thereby lowering the ad unit load times and improving end-user experience by reducing ad unit weights and the number of HTTP requests.

3.4.1 Webfont support and availability

Webfonts are supported using the Web Open Font Format (WOFF) or (in IE) Embedded OpenType (EOT). These standards are file formats for supplying font rendering details for your ad's text. WOFF is supported in 2 versions, 1.0 and 2.0. WOFF 2.0 is gaining adoption, but until it is more universally supported (and until old browsers such as IE are phased out), the CSS@font-face declarations should include font formats for all three versions:

- WOFF 2.0
- WOFF 1.0
- EOT (if support for legacy IE clients is desired)

Fonts represent an additional file in your ad package and contribute to overall file weight. While inclusion of all three formats listed is recommended for compatibility on all browsers, only one web font file is downloaded. The added file weight for including these three formats is based on the average file size of the three formats, since only one is downloaded. Publishers are encouraged to use shared font libraries and exempt any file weight that would otherwise be added for included webfonts.

3.4.2 Font usage considerations

For all the benefits that webfonts offer, a few details to ensure proper font rendering should be considered:

- **Font substitutes:** If a font is not supported or is otherwise missing during ad load and display, a fallback font substitute is used. To avoid skewing the ad appearance, declare suitable font substitutes the browser can use instead of allowing the browser to select a default font substitute. Fallback font definitions play an important role in minimizing the "Flash Of Unstyled Text" (FOUT) effect.
- **Font subsetting:** Browsers may use a "fallback" font to render the text visible while the requested webfont is being loaded. Depending on the size of the webfont and connection bandwidth, this process causes a brief "flash of unstyled text" or FOUT. To minimize the FOUT effect in ad units with static text content, you can subset the web font to reduce the glyph set of a font to only include those glyphs that are used in the creative. For really small size font subsets you may also consider using data URI for inline font data embedding. While this option increases CSS file size, it eliminates the file request for the font, improves load time, and avoids the FOUT effect.
- **Server-side support:** The fonts used by your ad unit(s) may be hosted online by third party servers (such as ad servers and publishers or webfont service providers). While

server-side supported fonts must still be included in the overall file weight of the ad, the hosted fonts allow for long-term browser caching, which improves the ad load time by reducing the size of transferred data and the number of file requests needed to deliver repeat ad impressions.

- **Shared web font resources:** Using static locations, such as with a vendor or partner that hosts web fonts, allows for long-term browser caching, reducing both the data transfer size and the number of HTTP requests. Shared web fonts also allows for font use among different ad units. When hosted on a shared server but cached locally, fonts remain available without requiring any additional data transfer or load requests, which also eliminates FOUT effect on subsequent ad impressions.

If you're not sure how to best support the fonts used in your ad, check with your ad server or publisher regarding your options.

3.4.3 Licensing Considerations

Fonts come with a EULA that specifies how and where the font can be used. Many font foundries provide licenses and font formats specifically for distribution on the web. Be mindful of the terms defined by the EULA, as there can be requirements and limitations on embedding, view count, number of domains hosted, and other considerations which may not be appropriate for the conditions of the ad publisher. Contact the providing font foundry to learn more about the requirements and terms for the font to be included in your ad creative.

3.5 Backup Image

In HTML5, compatibility depends on which tags the ad uses and whether the browser supports each of those tags. Since HTML5 is specification defined by the W3C, browsers can choose which parts of the specification they implement, without any particular order or priority. In short, whether a browser supports "HTML5" or not is undefined and ad servers are not designed to detect HTML5 compatibility in browsers. Ads can, however, check for feature support. Visit the IAB HTML5 wiki for a list of resources that can be used to check feature compatibility in different browsers.

3.5.1 HTML5 Fallback Recommendations:

Fallback functionality is a technique that implements alternate functionality when certain HTML5 features are not supported. Fallback preparation should include testing ad features across browsers to ensure acceptable rendering as well as static images when required by publishers.

The following recommendations help ensure HTML5 ads work as designed:

- **Note HTML5 features used**. Depending on the API usage, an ad can be compatible with all existing browsers or only a certain subset. There are several tools available to help show which browsers support which features, many of which may be available in the animation software used. For more information, see the IAB HTML5 Wiki. To determine which features your ad uses, review HTML5 development tool specifications. For example, for animation, the software may output the result in CSS, JavaScript or Canvas. Each of these outputs have certain pros and cons.
- **"Graceful degradation" is highly recommended**. The lack of one feature or another in a browser does not mean the ad is incompatible. For example, if getting the geo location is not available in the browser, the ad can provide a field for typing in the zip code or city name. Libraries such as Modernizr help with browser feature detection at run-time.
- **`<noscript>` Tag** should be applied, this provides a path to an alternate image when users have scripts disabled in their browser, or don't support client-side scripting.

Ad designers may be challenged as they design fallback functionality for each HTML5 feature used, but the effort involved defines the consumer experience and therefore the effectiveness of the ad.

3.6 Labeling

Since the webpage content and HTML5 ad share the same rendering technology (the browser) and resources, the ad unit must be clearly distinguishable from the normal webpage content as per the IAB Display Advertising Guidelines. To protect the aesthetics of the banner and brand message, a border should be applied in the ad creative rather than impose upon the ad server to produce a border.

3.7 Animation

In general, there are two ways to animate HTML5 banners: using CSS or JavaScript. Each has its strengths and weaknesses. When selecting a tool to generate your ads, understanding which technologies it uses to construct and animate your ads is essential for maximizing compatibility and performance.

3.7.1 CSS

CSS offers "transitions" and "animations" that can be applied directly in style sheets (e.g. "#element { transition: all 2s; }"). Instead of manually applying values on each frame like you would in JavaScript, CSS defines rules that the browser will implement to render the animation. Their native nature is both a strength and a weakness: it means that performance is usually great but it also means that browser implementation quirks are much more difficult (or impossible) to work around. The browser can also do smart optimizations, like ignore rendering the animation when it's not visible.

3.7.2 JavaScript

Animation is achieved with JavaScript by changing property values many times per second, typically in a requestAnimationFrame loop. The most commonly animated properties are CSS styles which can be set using JavaScript (e.g. element.style.width = “100px”). JavaScript can also be used to animate `<canvas>` content, WebGL, scroll position, SVG attributes, and almost anything in the browser.

JavaScript delivers maximum flexibility and compatibility. It is capable of stellar performance, but it is also more performance-sensitive; poorly-written JavaScript can have a much bigger impact on performance than poorly-written CSS animation.

For tips on leveraging better JavaScript for animation, visit the wiki.

3.7.3 Weighing the Options for HTML5 Development

Developing an ad using HTML5 is a lot like developing a webpage. The process can involve hand-coding or using tools that leverage HTML5 technologies such as JavaScript and CSS.

There are four factors to consider when assessing which technology to use for developing your HTML5 ads:

- Performance
- Compatibility
- Flexibility
- Workflow/Ease-of-use

Let’s look at CSS and JavaScript in light of these four factors.

3.7.4 Performance

Well-written JavaScript animation and CSS animation both perform pretty well, but JavaScript animation is sensitive to the quality of code used to generate the animation. The software-generated code is often poorly written, but unless you’re an expert coder, you’ll have to work with automated JavaScript that is likely to under perform.

JavaScript also requires more code because of the libraries needed to support execution. However, some tools use commonly shared libraries that most ad networks host on their CDNs. If you can make use of these shared libraries, then you may be able to have them exempted from your ad’s file size. When this is the case, performance falls in favor of JavaScript because the ad-specific code that references a shared library can more concisely define the animation than CSS can.

JavaScript can also use more CPU than CSS-animations if there are multiple JavaScript-animations on the same page.

See the HTML5 wiki for a list of commonly shared JavaScript libraries.

3.7.5 Compatibility

All browsers support JavaScript but there is varying support for individual CSS3 (the latest version of CSS) properties on DOM elements (this applies to both CSS and JS). For example, IE8 doesn't support transforms, border-radius, box-shadow and SVG. Some browsers require prefixes for certain properties (like "-webkit-transform" instead of "transform") so you'll need to be mindful of these differences.

JavaScript provides better backward-compatibility and more robust workarounds, which may be required by certain publishers and ad servers. In IE, CSS animations require IE10 or later.

See the HTML5 wiki for a list of resources for identifying browser compatibility.

3.7.6 Flexibility

Since JavaScript can access any property of any object in the browser, it is more flexible than CSS. The following list includes some examples of animation effects that you can accomplish with JavaScript but not with CSS:

- Physics or complex easing like elastic or bounce
- Dynamic effects based on user interaction (react to user mouse position)
- Independent control of transform components (position, scale, rotation, and skew)
- Run-time controls like altering speed or checking remaining time
- Scroll position
- SVG shape tweening
- Motion along a Bezier (curved) path
- Set an FPS cap

Despite the limitations of CSS, many projects don't require this level of flexibility. CSS is adequate for most simple animations.

3.7.7 Workflow

Building eye-catching, beautiful animations is an inherently experimental process filled with tweaking, testing, and revising. It's wise to select a technology that fits your build process well and maximizes your productivity.

For simple animations, CSS animations are easy to drop into your style sheets. However, even moderately complex animations can quickly become unwieldy and verbose when attempted in pure CSS. Managing all the delays gets cumbersome and hampers experimentation.

There are JavaScript libraries that allow you to modularize your animation logic and stitch things together dynamically, making it much easier to create complex sequenced animations that foster experimentation. Fortunately, your animation tools will generate the code for you. Hand coding can be time-consuming and make troubleshooting more difficult.

3.8 Animation Rendering Technologies

JavaScript and/or CSS apply changes over time to various properties, but the rendering technology is what actually displays those changes visually in the browser, and it often accounts for the majority of the processing load. These technologies are DOM, Canvas, SVG, and WebGL, and the rendering process is a little different for each one.

- **DOM (document object model):** encompasses HTML elements like and <img>. DOM elements are highly accessible and universally compatible. Complex layout rules baked into the browser enable responsive designs, text is handled well, and most browsers support 3D transforms on DOM elements.

- **Canvas:** included in the HTML5 spec and requires JavaScript (no CSS support). Instead of manipulating DOM elements, it animates by updating pixels within a <canvas> element. In modern browsers, Canvas is processed using the graphic processing unit (GPU), which results in faster rendering than with animation that uses the central processing unit (CPU).

- **SVG (scalable vector graphics):** enables sharp vector-based, graphic rendering at any size, which is useful for developing scalable and responsive creative. SVGs are DOM elements that can be styled with CSS, but they require more processing power to render because pixels are dynamically calculated as the graphic is resized.

- **WebGL:** high-performance technology for rendering high quality graphics, especially in 3D. It can be leveraged in a <canvas> tag, but only the most modern browsers support it at this time. Avoid WebGL animations for creative that needs to be served to a wide array of browsers and devices.

3.9 Recommendation

As Flash support dwindles, the number and quality of HTML5 animating tools increases. Some of these tools leverage CSS to do the run-time animation, and others use JavaScript. Depending on the ad campaign's requirements for performance and reach, some tools might be better than others. When assessing visual development HTML5 tools, be sure to assess performance, compatibility, and file size considerations associated with the exported content.

Look for HTML5 animation tools that leverage the following technologies:

- CSS for simple animation.
- JavaScript for more complex animation and superior compatibility, flexibility, and workflow.
- DOM manipulation for basic animations and to achieve maximum compatibility and accessibility.
- SVG and Canvas technologies if the ad campaign can avoid serving to IE8 or earlier. However, you may also include a simple fallback to an image for display in IE8.
- SVG for graphics that scale, but limit animation to simple groups of SVG elements to avoid performance issues.
- Canvas uses GPU processing and is recommended when multiple elements are animated simultaneously, but well-coded CSS and JavaScript can offer similar results.

Also, leverage tools to use well-established JavaScript and Canvas libraries that are likely to be hosted on your ad networks' CDNs so that you can take advantage of exempting those libraries from the file size of the ad (if approved).

3.10 Interactions

Most HTML5 creative development software provides an interface for creating interactions. The software generates the code necessary for execution in an HTML5 runtime environment. Depending on the ad campaign goals, the agency or advertiser marketing department may want to track certain interactions. In most cases, you may need to work with the ad technology vendor to format interactions in a specific way that aligns with their technology and methods.

3.10.1 The Clickthrough

All ads provide some kind of clickthrough used to provide valuable insight on campaign performance. Ad serving vendors have to find your implementation of the clickthrough and replace it with something that their systems can track. This process varies from vendor to vendor, but the use of code that can find the clickthrough in a click URL simplifies this process for the parties involved. The following example is a JavaScript function that locates the clickthrough and sets the JS variable `clickTag` to the clickthrough used.

```
function getParameterByName(name) {
  var match = RegExp('[?&]' + name +
'=([^&]*)').exec(window.location.search);
 return match && decodeURIComponent(match[1].replace(/\+/g, ' '));
}

var clickTag = getParameterByName('clickTag');
```

The above example would be placed in the header for the HTML ad, but is only a suggestion. Other code snippets may be used to accomplish the same task.

With a code snippet similar to the above example, the ad can use the same click code as if it was built using `var clickTag = "http://www.example.com";` line that explicitly defines the click URL in the header (see below example). The ad designer uses the JS method `window.open` or similar method to go to the URL in the `clickTag` variable, such as in the following example:

```
window.open(clickTag, '_blank');
```

Either the ad server (or ad developer) puts the ad on a CDN and invokes with a line of code similar to the following:

```
<iframe
src="http://d2o307dm5mqftz.cloudfront.net/1539675001/1445477600222/a.html?
clickTag=ENCODED_URL_MACRO"></iframe>
```

To reduce friction in providing a trackable clickthrough, use a function designed to locate the click URL and set it to the variable, `clickTag`, with appropriate camel-case capitalization. For multiple clickthroughs, use the format: `clickTag1`, `clickTag2`, `clickTag3`, etc.

In HTML, this variable should always be in the <head> tag as in the following example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
<script type="text/javascript">
var clickTag = " ";
</script>
</head>
```

In iframe cases where the clickthrough is included in the iframe src tag, use the variable as follows:

```
<iframe
src="http://d2o307dm5mqftz.cloudfront.net/1539675001/1445477600222/
a.html?clickTag=$$MACRO$$"></iframe>
```

Using the variable `clickTag` and including a function designed to set the click URL to this variable reduces a lot of overhead when it comes to setting the clickthrough in HTML ads. The content of `clickTag` can vary from being a combination of landing page URL, Tracking URL, Ad server Macro etc. No specific recommendation on the content of the clickTag is provided except that all of them should be inside of the click tag variable. As always, check with your partners regarding the best way to implement the clickthrough or any other interactions that will be tracked.

3.10.2 Mobile Rich Interactions

Rich ads with functions that leverage mobile in-app features, like geo-location and tilt, should leverage the IAB Mobile Rich Media Ad Interface Definition (MRAID). MRAID offers an API that enables your ad to interact with MRAID-compliant mobile apps. Using MRAID involves building standard calls and functions into the ad so that MRAID-compliant apps can recognize them and respond accordingly. This communication not only enables rich user interactions in mobile devices, it also enables standardized tracking used to track campaign and ad performance.

For more information on MRAID, please visit http://www.iab.com/guidelines/mobile-rich-media-ad-interface-definitions-mraid/.

3.10.3 Display Rich Interactions

Rich media interactions in display advertising require the browser to execute code on the page. Publishers have a rigid certification process for ad vendors who can serve ad code to their pages because it introduces a security risk. Because of this risk, many ads are served into an iframe, a mini webpage that is isolated from other page content. While more secure, iframes prohibit any ad-browser interactions, and ultimately any rich user experience that involve ad expansion and other user-initiated interactions.

The IAB developed SafeFrame to enable browser-ad interactions using an API that allows for communication between the page and the iframe. Like MRAID, SafeFrame involves building in standard calls and functions that a webpage can recognize and respond accordingly. Despite

the benefits of SafeFrame, publisher implementation is complex and adoption has been slow. However some publishers support SafeFrame. If you're developing a rich media ad for display with concerns about whether the ad will function properly, check with your partners about any special requirements for designing user-interaction features.

More details on SafeFrame are available here http://www.iab.com/guidelines/safeframe/ and http://safeframes.net/ and https://github.com/InteractiveAdvertisingBureau/safeframe.

3.11 Video

The following sections establish a set of guidelines for HTML5 creative that make use of the video tag so that creative designers can produce ads with video that function correctly across multiple browsers.

3.11.1 Display Video vs. In-Stream Video

Display video is video that is served into a display ad space, either as part of a banner ad or on its own. Several ad formats that use video are included in the display video definition.

Display video includes but is not limited to the following formats:

- In-banner video
- In-page video
- Rich media
- Interstitial
- Incentivized video

This document addresses these forms of display video.

In-stream video requires different resources and technology and is not covered in this document. Ads served in-stream are served to a player or app before, during, or after some form of streaming content. Guidelines for formatting ads for digital video in-stream ads are provided in the IAB Digital Video In-Stream Ad Format Guidelines.

3.11.2 File Format and Video Codecs

For an HTML5 video ad to play correctly in most browsers, you may need a few different file formats. While MP4 and WebM formats are typically sufficient for most browsers, some additional formats may be necessary for support in certain browsers.

A few common formats and codecs are:

- MPEG-4(MP4)/H.264 with AAC audio. Use the H.264 Base profile to ensure smooth working on lower end devices.
- Ogg/Theora with Vorbis audio.
- WebM/Vp8 with Vorbis audio.

Check the IAB wiki for resources that provide information on browser support.

These video files can be included in the HTML for the creative using the <video> tag. The browser uses the first recognized file format. Note the file weight of the video. The file weight used contributes to the overall ad file size and is subject to the Creative Guidelines limits.

The following code sample includes the suggested files using the HTML5 <video> tag:

```
<video width="320" height="240" controls>
  <source src="pr6.mp4"  type="video/mp4; codecs=avc1.42E01E,mp4a.40.2">
  <source src="pr6.webm" type="video/webm; codecs=vp8,vorbis">
  <source src="pr6.ogv"  type="video/ogg; codecs=theora,vorbis">
</video>
```

3.11.3 Aspect Ratios

For ad campaigns targeted to mobile devices or in the case of expanding rich media, the preferred aspect ratio for video is 16:9. For in-banner videos, an aspect ratio of 4:3 is commonly accepted.

Commonly accepted video dimensions for in-banner video include:

- 300x600 (large banner)
- 300x250 (medium rectangle)
- 300x170 (small rectangle)
- 728x90 (cropped to display main content)

3.11.4 HD versus SD

Standard definition (SD) may be used for display in low-res phones and small devices with resolution up to 960x640 px. High definition (HD) video may be used for Retina screens and devices with resolution up to 1024x769 px. Most in-banner video ads don't display on big screens, but if your campaign includes them as part of their strategy, ultra-high definition video may be generated for serving to devices with resolutions of 2048x1536 px or more.

3.11.5 Compression and file preparation tips

Provide a version of the video in which you reduce the audio to mono that can serve to devices that don't support stereo. Also, if the video contains no audio, be sure to generate the video without an audio track.

Also, use a "web-optimized" format for compression, which moves the movie (MOOV) atom to the beginning. This setting ensures that the video plays as it downloads. Video is often encoded without adjusting the MOOV atom, and if it's listed last the browser waits for complete download before playing the video.

3.11.6 Progressive vs Adaptive Streaming

By default HTML5 supports progressive video download, which means the browser downloads the video like an image or any other page resource, instead of playing it from where it's served (streaming). Adaptive bitrate streaming methods have been developed to simulate streaming video, which improves performance and smooth playback.

Adaptive bitrate streaming techniques break up a video file into multiple short 2-3 second clips, each at multiple playback qualities. These segments are added to playlist file, such as M3U8. The player parses this playlist at each 2-3 second interval and selects the best quality file to match bandwidth levels during that interval. The result is seamless playback in environments where bandwidth is inconsistent.

To generate a video in adaptive bitrate format, the following options are available:

- HLS – Most commonly accepted, especially in the US (required in iOS devices and Safari)
- MPEG-DASH – Also commonly accepted, with higher rates outside the US

3.12 Audio

For ads that include audio (without video), several options exist. MP3 format for audio is the most universally accepted (excluding IE8 and earlier). If you plan to use other formats check with your ad partners about whether audio formats are accepted. You may be able to include a backup format if you have a preference for audio that may not be accepted everywhere.

Also, while audio is a great attention-grabber, the IAB Creative Guidelines only allow audio upon intentional user interaction. At a minimum, include a mute control with any ads that include audio and check with your ad partners on their requirements for audio.

3.13 Controls

Standard controls are required in HTML5 ads just as they are in any ads. In some cases the ad serving vendor adds things like the close X control for floating or expanding ads. With video, defining the controls using the HTML5 video tag instead of designing your own reduces file weight. Check with your vendor or publisher about specific control requirements.

3.13.1 Autoplay

When using a data connection, video ads should only play upon user-initiation. Autoplay should be prevented in mobile, data-connected environments to avoid incurring unplanned costs to the user.

4 Packaging and Testing Your HTML5 Ad

Emerging HTML5 creative design tools export all the files needed to execute the ad. In most cases you'll work with an ad serving vendor that will format and package your ad for delivery, but some files may not be necessary or there may be a few special circumstances for your ad based on it's functionality and the target market. For example, Flash leverages jQuery libraries in its HTML5 export; however, jQuery libraries increase file size and/or may not be supported by some publishers.

The Creative Guidelines limit file size and number of file requests allowed at different load stages. Consult with your ad serving or rich media vendor regarding any ad submission

requirements. Discuss your plans for ad design and functionality and whether there are any alternate design implementations necessary for the best performance and least amount of friction in the ad trafficking work flow.

The following sections outline some of the HTML5 properties to consider as you design your ad and prepare it for submission.

4.1 File size and file load

The Creative Guidelines offer file size loads in three phases: initial load, host-initiated subload, and user-initiated load.

- **Initial load:** file weight limit includes all files necessary for first complete view of the ad.
- **Host-initiated subload:** additional file weight allows rich media ads to include animation or other rich functionality that occurs before any user interaction. Host-initiated subload is defined as "the additional file limit allowed for rich media and Rising Star units that is auto-initiated at least one second after the `domContentLoaded` occurs (web page content has been loaded) on the host computer or device.
- **User-initiated load:** unlimited file weight allowed after the user interacts with the ad. (User experience, bandwidth limitations and other factors should be considered when designing any user-initiated ad creative.)

These file-load phases are described in more detail in the IAB Display Creative Guidelines.

While the ad serving vendor packages the ad for delivery into the system, consider designing your ad with these load phases in mind. Use fewer features for the first visual experience of the ad to support lower file weights at that phase. Increased files weights are allowed for the host-initiated subload, but use the allotted file weight to draw attention and invite user interaction. Save the richest portion of the ad design for when the user interacts.

Designing with these load phases in mind simplifies the ad packaging and delivery process.

4.1.1 Asset Compression

Creative development tools for graphics, video, and audio offer a wide array of options for balancing quality with file size. Asset compression is increasingly important with high-density displays. To compress assets, use PNG "crushers" and other forms of image compression, and leverage formats such as scalable vector graphics (SVG) whenever applicable. SVGs can scale indefinitely to high-resolution displays without increasing file size.

4.1.2 Shared Libraries

Some of the files included in the HTML export reference shared libraries. These libraries support ad execution for several ads by referencing files from a shared server. Features like fonts or JavaScript are examples of features that reference shared libraries. Although shared, a common library may be hosted on a CDN, on the publisher's server, on the rich media vendor's server, or other locations. Some publishers host some shared libraries while not hosting others. Currently, there is no standard way to leverage shared libraries, and some lesser-known but possibly better libraries may not be hosted anywhere.

Publishers and ad servers are encouraged to take advantage of browser caching functionality by allowing use of common and popular shared libraries in HTML5 Ads. Doing so improves ad load and page load performance since the previously used shared library will not need to be downloaded again.

Publishers and ad servers are encouraged to exempt certain shared libraries from the initial file weight calculation of the gzipped HTML ad. During the publisher certification process, publishers should approve both the shared library and its source before the library can be exempted from ad file size. If shared libraries cannot be exempted, the file weights for these libraries must be included as part of the ad's initial file weight.

Check the IAB wiki on HTML5 resources for a list of common libraries.

4.1.3 Zip File Contents

Before compressing all files for your creative, check the following guidelines to ensure the ad is ready for the ad server to handle:

- ✓ Each .zip file should represent one ad. Multiple ads may be zipped into a package, but each file in that package should be a zipped file representing one ad with labels to distinguish each ad. A common rule for labeling ads is to include the width and height in the ad file name.
- ✓ There must be at least one index.html file (the starting point of the ad) in the .zip for each ad. If multiple .html files exist for one ad, the first file to be loaded should be the index.html file. This index file should include the meta data for declaring your creative dimensions as explained in *Section 3.2 Dimensions*.
- ✓ Structure files as needed. No specific rules are outlined for the folder structure of the .zip file, but consider combining like files together (images, fonts, libraries, etc.). Files may be organized in subfolders or may be present solely within the root folder. Also, consider using a common structure for each zipped ad in a package.
- ✓ All code and assets should be relatively referred to by the index.html file, meaning that the index.html file should reference files that are in the same folder or within a folder where the index.html file is. File references higher up in the folder hierarchy cannot be loaded once served and will render a broken ad.
- ✓ Minimize the number of files included within the .zip file. For performance reasons, the Creative Guidelines limit file requests to 15 for initial file load.
- ✓ All code and assets needed for initial display and the host-initiated subload (for rich ads) should be contained in the .zip file. Exceptions include files such as JavaScript libraries or Web fonts that are exempted by the publisher; however, if not exempted, the file size of these external files count toward the initial file load size. Additionally, exemptions exist where the JS tag is used to call non-standard dynamic ads. Check the IAB wiki for HTML5 ads for tools that help to determine whether the ad meets file weight requirements.

4.1.4 File requests

HTML5 ads operate on a collection of files. During execution, each file request increases ad and page load time. As part of the 2015 Creative Guidelines, all file requests during initial file load and subsequent host-initiated file load, including tracking files and pixels, are limited to 15. After user initiation, unlimited file requests are allowed. Despite file request guidelines and limits, keep in mind that ad performance is improved with fewer ad requests.

To reduce file request in the ad, limit the number of creative assets, libraries, and other resources used to execute the ad. For example, combine multiple image assets into one image or as few images as possible.

Serving assets over HTTP/2 from the same hostname will reduce the problem of many assets, but it might take some time before browser and server support is good enough. Check current status of HTTP/2 implementations at https://github.com/http2/http2-spec/wiki/Implementations.

4.2 CPU and GPU usage

Computing devices use two different types of processors: the central processing unit (CPU) and the graphics processing unit (GPU). Ad functionality that leverages the GPU reduces the need for the CPU and improves performance. You have little control over how your ad is processed on the device where it displays, but certain HTML5 components like CSS and Canvas are designed to be executed by the GPU if one is found.

5 Tools

There are a number of tools, utilities, and companies that can help you with HTML5 creation as well as adhering to the suggestions in these guidelines. The IAB Wiki provides a living resource that is updated as requests are submitted.

In general, the toolset for designers and developers differs. Designers need tools that focus on producing the creative assets and developers need tools for modifying the CSS, Javascript, HTML5 and other files.

5.1 For Designers

The following list provides an overview of the different types of resources available to designers who need to develop ads in an HTML5 format.

5.1.1 HTML5 mobile rich media

Some of the most powerful tools are available from rich media ad vendors. These often provide a complete environment for ad creation using templates or components to build your ad and include ad-friendly features like video, geo-location, and mobile calls to action.

As advertising specialists, rich media ad vendors already adhere to many of the best practices outlined in this document. Look for tools that:

- Generate universal tags for cross-platform delivery
- Support animations and transformations
- Import animations and custom HTML from other tools
- Provide components to build ads without hand-coding
- Support web standards and IAB MRAID
- Deliver ads are lightweight and optimized for platforms

5.1.2 Graphic Development Tools

In general, the tools you use today to develop image assets can still be leveraged in your HTML5. The most important aspect is generating the output. Creative should be saved as PNG, JPG, GIF, or SVG with attention to file size and clarity. Expect to use other tools for image compression, optimization, and sprite creation.

GIF note: Browsers handle animated GIFs very poorly, especially in mobile. One loop animated GIFs don't play upon refresh, they're file-weight heavy, image quality is low (especially if scaled), and you have no control over frame start. HTML5 animations offer several benefits over animated GIFs. Consider avoiding GIFs and use HTML5 animation instead.

5.1.3 HTML5 Animation Tools

HTML5 animation tools attempt to replace the Flash building experience with a frame-based animation builder using HTML5. When choosing an animation tool, give special consideration to

the generated output. Some tools require additional JavaScript libraries, which make the output too heavy.

HTML5 animation tools met the following minimum criteria:

- Can be used on your platform (Mac, PC, online)
- Generates small files without external JavaScript
- Provides cross-platform support, especially for mobile
- Follows current standards for HTML5 and CSS3
- Follows advertising standards like the IAB's MRAID for advertising in-app

5.1.4 Flash converters

There are tools that attempt to convert Flash files to HTML5. These tools have limitations—not all features can be converted — and optimization may be limited. Look at these tools as a possibility to convert simple Flash files or to "jump-start" a larger conversion effort.

While useful during transitions from Flash ads to HTML5, Flash converters are not a viable long-term solution. Use the recommendations in this guide to help you move to a long-term HTML5 solution.

5.1.5 HTML5 website tools

Many website building tools are incorporating HTML5 features and provide a designer-friendly layout for drag-and-drop creation.

Because these are general-purpose tools, the generated HTML5 lacks some best practices for ad creation. For these tools, you should expect to do some hand-coding after the initial designs.

Look for tools that:

- Support responsive layouts
- Provide mobile device previews
- Include web fonts
- Have a code-view

5.2 For Developers (Ad Technologists)

The following list provides an overview of the different types of resources available to traffickers and other ad operations engineers who handle campaigns that use HTML5 ads.

5.2.1 Browser Reference

As you consider cross-platform and cross-browser support, having a reference is essential. HTML5 is an emerging standard and support for CSS3 is not complete for all vendors. Check the IAB wiki for resources that provide relatively up-to-date information on browser support for select HTML5 features.

5.2.2 JavaScript Libraries

Using JavaScript libraries is a common practice in web-site building, but is not always the right approach for ad building. Look for libraries that do not conflict with publisher sites and can be optimized to include only the methods you use.

5.2.3 Code compression

To serve quickly, look for code compression tools that can reduce file size. Better tools also provide obfuscation and can be incorporated with build tools to run automatically.

5.2.4 Image compression

Image optimization is often the most effective approach to reducing file size. Look for tools that provide lossless optimization, can work on a batch of files, and allow you to choose a compressed image from a preview.

5.2.5 Sprite Generators

These tools create a single image from several smaller images which is a great technique for limiting the number of connections and improving load times. Look for a tool that not only combines images to a single sprite, but also provides the CSS to display the portion of sprite needed in your ads.

5.2.6 Web fonts technology

Font foundries also provide fonts for the web and tools to include them. Look for plug-in support that make it easy to include web fonts in your other creative tools, and support for font sub-setting where only the letters needed for your creative are downloaded to the device.

6 Terminology

animation: A programmatically generated display of sequential frames or transitioning images, creating the illusion that objects in the image are moving. Not digital video, as it relates to this document.

application Programming Interface (API): Application Programming Interface is a set of commands, the language that programmers or developers use to communicate with a specific piece of software or hardware. For example, mobile ads delivered in apps use an API to communicate with an SDK (like MRAID) that is built into the app.

base64 encoding: Is when you convert image information into text (radix-64 representation) and paste it into a document instead of making additional server request to load the image. Note: The Image weight increases by ⅓ when using Base64.

bitrates: a measure of bandwidth that indicates how much data is traveling from one place to another on a computer network. Bitrate is usually expressed in kilobits per second (kbps) or megabits per second (Mbps).

canvas: an HTML5 element that is a resolution-dependent bitmap container used for rendering graphics, interactivity and animation dynamically through JavaScript directly within the browser and without the need for any 3rd party plug-ins. Canvas provides a set of functions ("the canvas API") for drawing shapes, defining paths, creating gradients, applying transformations and more.

character: a unit of text content represented by its code point. For example, the character "a" can be processed by a screen reader as part of the text string but it doesn't have any associated visual representation.

code Minification: practice of removing unnecessary characters from code to reduce its size, removing unnecessary spacing, and optimizing the code; thus improving load times.

central Processing Unit (CPU): the key component of a computer system, which contains the circuitry necessary to interpret and execute program instructions.

cascading Style Sheets (CSS): language used for describing the presentation semantics (the look and formatting) of a document written in a markup language.

degrade Gracefully: When a developer codes an ad unit using the latest HTML features and it is viewed by a less feature-rich browser the ad should "degrade" in a way that is still functional, but with fewer features. (Also referred to as Graceful Degradation**)**

document object model (DOM): The DOM is a W3C standard for accessing documents like XML and HTML. The HTML DOM defines the objects and properties of all HTML elements, and the methods (interface) to access them. JavaScript can be used to move and manipulate DOM elements to create animation.

font: a resource that provides means and additional data to render and visualize text.

font subsetting: a process by which only select characters of a font are included in a file so that text displays as intended without embedding the entire font.

frame rate: The rate at which video frames or animated images display as the video or animated file executes, measured as the number of frames per second (fps).

frames per second (FPS): metric used to indicate the frame rate of animated or video creative content.

glyph: a unit of text display. Glyphs are defined by a font chosen for each particular text fragment, a font will provide a collection of glyphs that share the same design style and unified metrics (a.k.a. a typeface).

graphics processing unit (GPU): GPU handles graphical processing, decreasing the processing burden handled by the CPU.

gzip: When activated and configured, Gzip offers automatic compression of creative assets for an ad when delivering from an ad server to a web page or application.

high resolution displays: screen displays that uses more pixels to cover the same physical area, also known as retina display.

JavaScript libraries: a library of pre-written JavaScript which typically include functions for common tasks like animations, DOM manipulation, and Ajax handling. (Often called JavaScript frameworks)

kilobyte (KB): A multiple of the unit 'byte' for digital information, used to quantify computer memory or storage capacity roughly equal to a 1,000 bytes (or technically, 2^10 = 1,024 bytes).

k-weight: weight of a file measured in kilobytes.

megabyte (MB): A multiple of the unit 'byte' for digital information, used to quantify computer memory or storage capacity roughly equal to 1,000 kilobytes (or technically, 2^20 = 1,048,576 bytes).

Mobile Rich Media Ad Interface Definitions (MRAID): standardized set of commands, designed to work with HTML5 and JavaScript, that developers creating rich media ads will use to communicate what those ads do (expand, resize, get access to device functionalities such as the accelerometer, etc) with the apps they are being served into. http://www.iab.com/guidelines/mobile-rich-media-ad-interface-definitions-mraid/

progressive enhancement: when a creative developer uses features that are widely supported across browsers, but also develops an enhanced version using the newest HTML5 features for browsers that are compatible.

SafeFrame: SafeFrame 1.0 offers a solution that prevents external HTML content from accessing the website and its sensitive data by framing and rendering the content from within a secondary domain. An API enables communication between the webpage and the external content to allow for any rich interactions. http://www.iab.com/guidelines/safeframe/ and http://safeframes.net/ and https://github.com/InteractiveAdvertisingBureau/safeframe

software developer's kit (SDK): is a pre-packaged piece of code that developers can incorporate into their application to avoid having to develop it from scratch. For example SDKs

from rich media vendors and networks are often implemented into the publisher's mobile app to handle advertising.

sprite sheets: Is a large image filled with smaller images. Putting all image assets for a creative into one sprite sheet reduces the server calls needed to load the images. Another use of sprite sheets is to create animation by displaying each of the smaller images in the correct order.

subsetting: see font subsetting

scalable vector graphics (SVG): Defines graphics in XML format and can scale indefinitely to high-resolution displays without increasing file size.

text: an element of ad content that has a semantical meaning but does not have predefined visual appearance (as opposed to pre-rendered images where text is present as a visual element only). The visual appearance of a text (it's size, layout and look) is determined by font choices.

transcoded: is the direct digital-to-digital data conversion of one encoding to another, such as for movie data files or audio files. Files such as video assets may need to be transcoded into a few different formats to ensure cross browser functionality.

Video Suite (VSuite): a set of technical specifications and protocols for in-stream video ad formats that allow compliant ads to seamlessly play across multiple compliant publisher sites. This included IAB's Video Ad-Serving Template (VAST), Video Player-Ad Interface Definition (VPAID) and Video Multiple Ad Playlist (VMAP). http://www.iab.com/guidelines/digital-video-suite/

WebViews: A container with a rendering engine that displays web content within an app environment. This is sometimes referred to as a "micro-browser".