The HTML5 Picture Element: WordPress

What is the <picture> element?

The HTML5 `<picture>` element provides an alternative method of handling images on a webpage.

Much like the `<video>` and `<audio>` elements, the `<picture>` element contains any number of `<source />` elements as well as an `<img />` element that acts as a fallback for browsers which do not support the `<picture>` element

The `<picture>` element provides new and improved ways of handling responsive images, but most importantly, it can help in curating art-directed images. This essentially means that developers can serve multiple versions of the same image, potentially with different image formats,crops, or different content across devices.

How does it work?

How does the `<picture>` element actually work? There are a couple of behaviors to note:

1. The `<picture>` element itself behaves more like a block-level element container. It can accept an `id` and `className` but limited styling should be applied to this element that is intended to affect the `<img />` element.
2. According to MDN, the picture element handles fallbacks in a somewhat unique way. The picture and `source` elements are not capable of rendering an actual image. The picture element itself is simply just a wrapper around source and img elements. When a browser encounters a source element and the conditions match, the `src` of the `img` element is overwritten (dynamically, no replacement in the browser).
3. If a condition is met, the user agent serves the url to the image attached to `<source />` as the `src` of the `<img />` element. This change is not rendered to the user in the browser (no change in the DOM).
4. If no conditions are met, the original `<img />` src is used.

`<source />` elements do not contain width and height attributes, thus dimensions and aspect ratio are controlled on the `<img />` element itself.

A user can also implement `object-position` and `object-fit` when it comes to using the `<picture />` element. These 2 CSS properties provide greater flexibility in ensuring images scale correctly across devices, especially with regards to aspect-ratio. All CSS properties should be applied to the `<img />` element, not the `<picture />` element.

The <source /> element

Each `<source />` element should contain the following attributes depending on the use case:

1. `media` - this is an inline media query, e.g `min-width (800px)`
2. `type` - mime type of the image, e.g. `image/webp`
3. `srcset` - this attribute is used to provide image descriptors of different widths that match breakpoint conditions.

How can it improve WordPress?

Historically, WordPress has been slow to adopt the picture element for a number of reasons:

1. It adds complexity to an already complex ecosystem
2. There is no clear way to handle the amount of default and custom image sizes WP generates on the fly
3. There's been no support for modern image formats (up until now)
4. A change in markup and classes could lead to millions of themes breaking.
5. Lack of browser support & support of Internet Explorer

Perhaps the biggest benefit of using the `<picture>` element is that WordPress could support art-directed images. Cropping and editing images have been a feature in WordPress for quite some time. However, users have lacked the ability to be able to assign different crops based on device or responsive breakpoints.

The other benefit exists around performance and responsive images.

Some major advancements include:

- Support of next-gen formats such as AVIF.

- More adaptable user experiences, allowing the browser to determine the most appropriate format and image size to serve.
- Proggresive Enhancement using fallbacks inherent with <picture> functionality.

WordPress has supported responsive images for a while now - using the `srcset` and `sizes` attributes on `<img />` elements.

Using the `<picture>` element, each `<source />` can specify a number of image descriptors based on the `w` descriptor in separate formats. Considering `<source />` uses `srcset` and `media` we can utilize the best of responsive images, modern formats AND art-directed crops across devices.

WordPress never destructively edits the original uploaded image, so generating multiple `.webp` images in different crops and widths to account for mobile devices, along with art-directed `.webp` or `.jpeg` images is a possibility of note.

How do we handle styles?

The `<picture>` element, in terms of styling, behaves rather like the `<figure>` element - a block level element. The only real use-case for the `<picture>` element itself is to provide a containing wrapper around the `<picture>` element's own API. Thus any CSS that is applied in order to treat images should be applied to the `<img />` element itself and not the `<picture>`

How do we handle layout shifts?

No different than how we would handle CLS on normal images. `width` and `height` attributes are applied to the `<img />` nested within the `<picture>` element itself. The `aspect-ratio` CSS property which is applied to all images by the `user-agent` stylesheet should keep dimensions in check.

Can we still use loading=”lazy” on the <picture> element?

Yes. The difference being that the loading attribute is still placed on the `<img />` element. Not the `<picture>` element itself.

How can it improve performance?

The biggest performance benefit with regards to the `<picture>` element revolves around its ability to selectively serve appropriate images based on device type. The API allows developers to provide a well-thought out strategy when it comes to:

- serving images in modern formats and serving fallback image types when a modern format isn't supported by the browser
- serving optimized images based on device or breakpoint width
- a combination of both which impacts not only performance but user-experience too.

Markup Examples

https://github.com/dainemawer/performant-html

Opt-in Support

```
add_theme_support( “picture” );
```

The above method which has more than likely been mentioned in previous tickets, would provide a far more versatile way to add picture support to a theme. The notion of replacing all image markup in `post_content` / templates could create a fair amount of regression in already built themes. Adding the above call to `add_theme_support` will allow developers to disable, enable, or provide support for the picture element in a holistic and well-supported / incremental manner.

Another option is to include picture support in `add_theme_support( 'html5' )`

```
add_theme_support( “html5”, array( “caption”, “gallery”, “picture” ) );
```

Important WP Core Functions

The below list of functions would probably be a useful place to start in terms of building the `<picture>` markup and rendering it out in post content.

Function	Reference	Purpose
`wp_get_attachment_image_src`	Codex	Retrieves an image to represent an attachment.
`wp_get_attachment_image_srcset`	Codex	Retrieves the value for an image attachment's 'srcset' attribute.
`wp_calculate_image_srcset`	Codex	A helper function to calculate the image sources to include in a 'srcset' attribute.
`wp_get_attachment_image_sizes`	Codex	Retrieves the value for an image attachment's 'sizes' attribute.
`wp_image_src_get_dimensions`	Codex	Determines an image's width and height dimensions based on the source file.
`wp_get_image_mime`	Codex	Returns the real mime type of an image file.

Browser Support

This table conveys the cross-browser testing results of the examples contained in https://github.com/dainemawer/performant-html

Browser	E.g 1	E.g 2	E.g 3	E.g 4	E.g 5
Safari	✅	✅	⛔	⛔	✅
Chrome	✅	✅	✅	✅	✅
FireFox	✅	✅	✅	✅	✅

WP Plugins Cross Reference

See: Plugins Conversion to Webp

Plugin	Picture Support	Approach
WebP Express	✅	1. Removes all `<img />` tags and any `<picture>` tags that already exist. 2. Processes all `<img />` tags found using a regex. 3. All attributes are parsed with `DOMDocument`. 4. Re-inserts elements into DOM. 5. Filtered through `the_content`
WPvivid Imgoptim Free	⛔	No support for `<picture>` element
Autoptimize	⛔	Processes `<picture>` elements in terms of amending optimized images based on settings but does no conversions from `<img />` to `<picture>`
EWWW Image Optimizer	✅	View `filter_page_output` function in `ewww-image-optimizer/classes/class-eio-picture-webp.php`
Compress Images with Squeeze Img	⛔	No support for `<picture>` element.
Easy WebP	⛔	No support for `<picture>` element.
Femora Compress	⛔	No support for `<picture>` element.
Flying Images by WP Speed Matters	⛔	No support for `<picture>` element.

WP Media Optimizer	⛔	No support for `<picture>` element.
Auto Webp Image Converter	⛔	No support for `<picture>` element.
Imagify	✅	1. Removes existing `<picture>` tags using `preg_replace` 2. Rebuilds `<picture>` tags. 3. Replaces all `<img />` tags with `<picture>` 4. Adds Gutenberg support for `<picture>` tags. 5. Runs on `template_redirect` hook.