Reviewing Lazy Loading in WordPress

In the not to distant future WordPress will have lazy loading by default in WordPress core enabled by default. I believe that this will be merged into core for the 5.5 release; until then, development is done on a feature plugin for people to continue working on it.

But the problem is that this is enabled by default. As with many of the later decisions by the WordPress core team, this is good for beginners but I can think of at least two instances where lazy loading is not what I want to use:

- Header images at the top of the page. If we want to minimize the load time then we shouldn't lazy load them
- Images and iframes above the fold that the user will see when the page first loads

There are several plugins that already do image and iframe lazyloading in my WordPress installation I have decided to use Jetpack and W3 Total Cache so I won't loose lazy loading if I deactivate it in core.

Another thing to keep in mind is how does lazy loading affect existing image manipulation plugins such as Cloudinary?

I was thinking about enabling Jetpack's lazy loading feature but while researching it I realized that it uses `data-src` attributes to handle lazy loading so I don't know how it will interact with Cloudinary and the `srcset` attribute.

W3 Total Cache's documentation and support leave a lot to be desired for the free version

Incorporating lazy loading without core

The first thing to do if you want to keep your code working as before is to disable lazy loading in WordPress Core with the `wp_lazy_loading_enabled` filter.

```
add_filter( 'wp_lazy_loading_enabled', '__return_false' );
```

We can then alter the content that gets inserted into the (classic) editor and will be published with the post.

The function will take the image from the attachment and the HTML from the image tag and use it to build a figure element with captioning, and alt attributes built from the content of the attachment page.

It also adds lazy loading that you can modify or remove based on your neds. As far as I know you can't remove the attribute once it's been added to image.

```
<?
function html5_insert_image($html, $id, $caption, $title, $align, $url, $s
  $src  = wp_get_attachment_image_src( $id, $size, false );
  $html = get_image_tag($id, '', $title, $align, $size);
  $html5 = "<figure>";
  $html5 .= "  <img src="<figure>"'$url' alt='$alt' class='size-$size'lazy
  if ($caption) {
    $html5 .= "  <figcaption class='wp-captio";
  if ($caption) {
    $html5 .= "'wp-caption-text'gure>";
  return $html5;
}
add_filter( 'image_send_to_editor', 'html5_insert_image', 'image_send_to_e
```

The drawback of this method is that it only handles new images when using the classic editor, it will not go back and retroactively add the `loading` attribute to images already used in posts, if you need that done you'll have to do it manually.

Good or Bad?

Whether native lazy loading is good or bad depends on what you need and what you're using for image management and lazy loading (if anything). Having lazy loading in core is not the only way to lazy load images, just what appears to be the most convenient because people just want things to work.

If you already use lazy loading from other sources you have to test carefully how they interact with your code and any image manager that you have installed.