Getting started with tiny.pictures

Setting up your account and deploying real-time image optimization to your website in an instant

Simple HTML integration

We provide your new image optimization infrastructure under your personal sub domain at **https://demo.tiny.pictures/main**. The **main** directory works as a shortcut to your image source servers at **https://your.website/**.

Additionally, it adds all the image optimization functionality you need and delivers your images through a Content Delivery Network.

You are now able to apply image operations by simply swapping the host name of your image URLs and adding some query parameters:

```
<img
    alt="Your original image"
    src="https://your.website/example1.jpg">
<img
    alt="The same image in grayscale (delivered by tiny.pictures)"
    src="https://demo.tiny.pictures/main/example1.jpg?grayscale=true">
```

Responsive images

Thanks to HTML5's `srcset` and `sizes` attributes you can now speed up your website by delivering perfectly-sized images to your users – no matter what type or size of device they use. These attributes tell the browser that there are a couple of differently sized image candidates to choose from and how large the image will be displayed on the web page. Together with other information already known to the browser – e. g. viewport size, device pixel ratio, network bandwidth, and supported file formats – the browser automatically chooses the best image candidate.

```
<img
    alt="Image candidates width widths of 200, 500, and 800 pixels,
        displayed at 50% of the viewport width"
    srcset="https://demo.tiny.pictures/main/example1.jpg?width=200 200w,
        https://demo.tiny.pictures/main/example1.jpg?width=500 500w,
        https://demo.tiny.pictures/main/example1.jpg?width=800 800w"
    sizes="50vw">
```

That's quite a lot of code, right? That's why we offer both client and server libraries to make implementing such advanced HTML a whole lot easier.

Lazyloading and automatic srcset and sizes calculation with our JavaScript client library

Coding `srcset` and `sizes` attributes over and over again can be a tedious and error prone task. That's why you as a good developer are probably also a lazy developer… Take a look at how our JavaScript client library may help you:

```
<img
    alt="This image will be lazyloaded and automatically wrapped by an HTML5 picture element"
    class="tp-lazyload"
    data-tp-src="https://your.website/example1.jpg"
    data-tp-sizes="auto">

<script src="https://cdn.jsdelivr.net/npm/tiny.pictures-js@4.1.1/dist/browser.js"></script>
<script>
    var tinyPictures = new TinyPictures({
        window: window,
        user: "demo",
        namedSources: [{"name":"main","url":"https://your.website/"}],
        srcsetWidths: [200,500,800]
    })
    tinyPictures.lazyload()
</script>
```

Our library automatically converts your `img` element to a comprehensive `picture` element with lazyloading, automatic image candidate selection and support for the WebP image format.

```
<picture>
    <!--[if IE 9]><video style="display: none"><![endif]-->
    <source type="image/webp" srcset="https://demo.tiny.pictures/main/example1.jpg?format=webp&width
    <source srcset="https://demo.tiny.pictures/main/example1.jpg?width=200 200w,https://demo.tiny.pi
    <!--[if IE 9]></video><![endif]-->
    <img class="tp-lazyloaded" src="https://your.website/example1.jpg" srcset="https://demo.tiny.pic
</picture>
```

Server-side libraries

One drawback of the client-side JavaScript technique is that the browser starts loading images only after the JavaScript is loaded and executed. For "above the fold" images you might want to use a server-side solution. That's why we also offer several server-side libraries, e. g. for Node.js.

```
const TinyPictures = require('tiny.pictures-js')
const tinyPictures = new TinyPictures({
    user: 'demo',
    namedSources: [{"name":"main","url":"https://your.website/"}],
    srcsetWidths: [200,500,800]
})
const imageUrl = "https://your.website/example1.jpg"

const middleware = (req, res) => {
    return res.send(`
        <img
            alt="The image in grayscale (delivered by tiny.pictures)"
            src="${tinyPictures.url(imageUrl, {grayscale: true})}">
        <img
            alt="Image candidates width widths of 200, 500, and 800 pixels,
                displayed at 50% of the viewport width"
            srcset="${tinyPictures.srcsetArray(imageUrl, {grayscale: true}).join(', ')}"
            sizes="50vw">
    `)
}
```

For more information, please take a look at our documentation for the Node.js and PHP library.