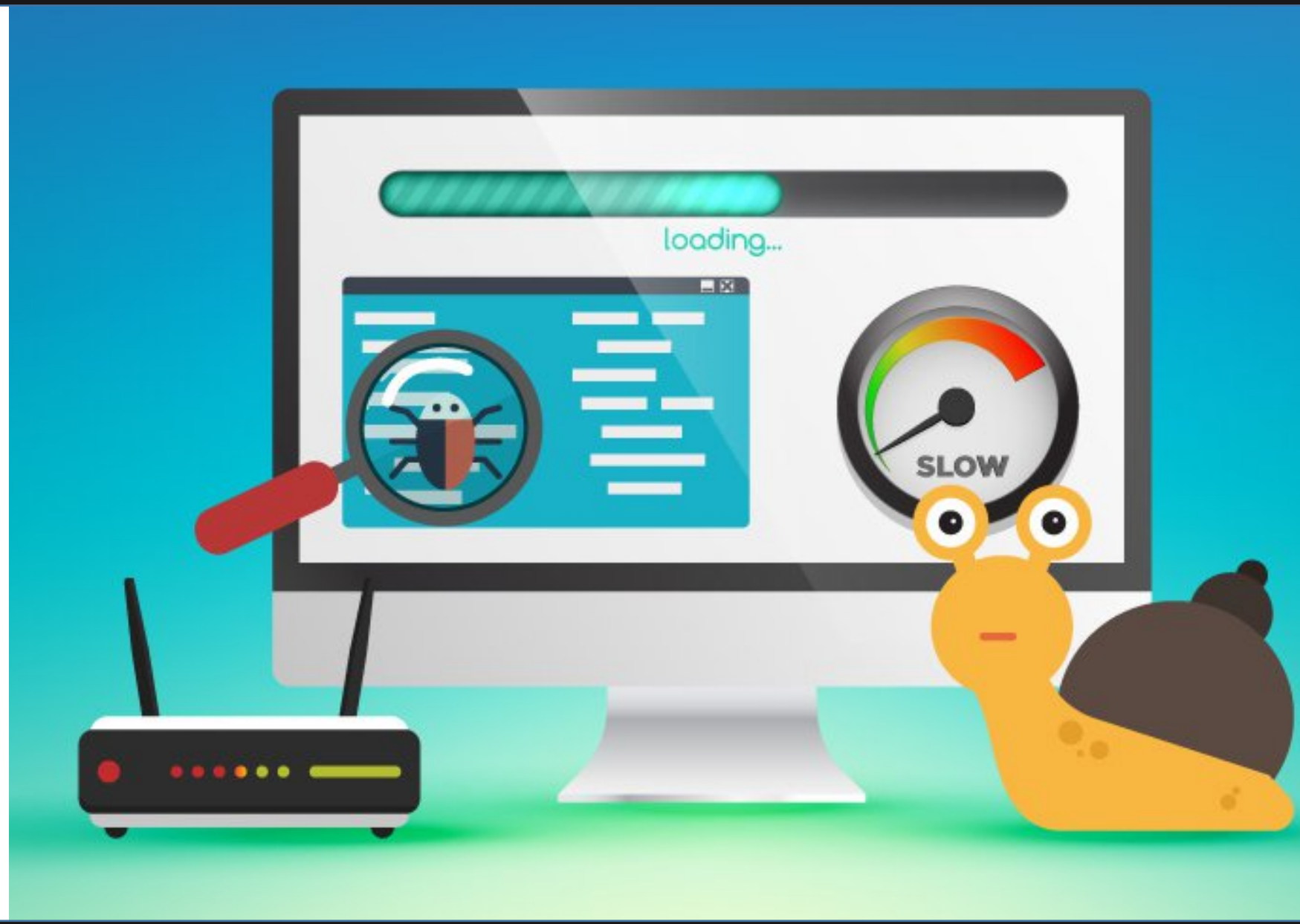
TESTES E PERFORMANCE WEB

TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

POLITÉCNICO DO PORTO
ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN

M04 – ASSETS OPTIMIZATION

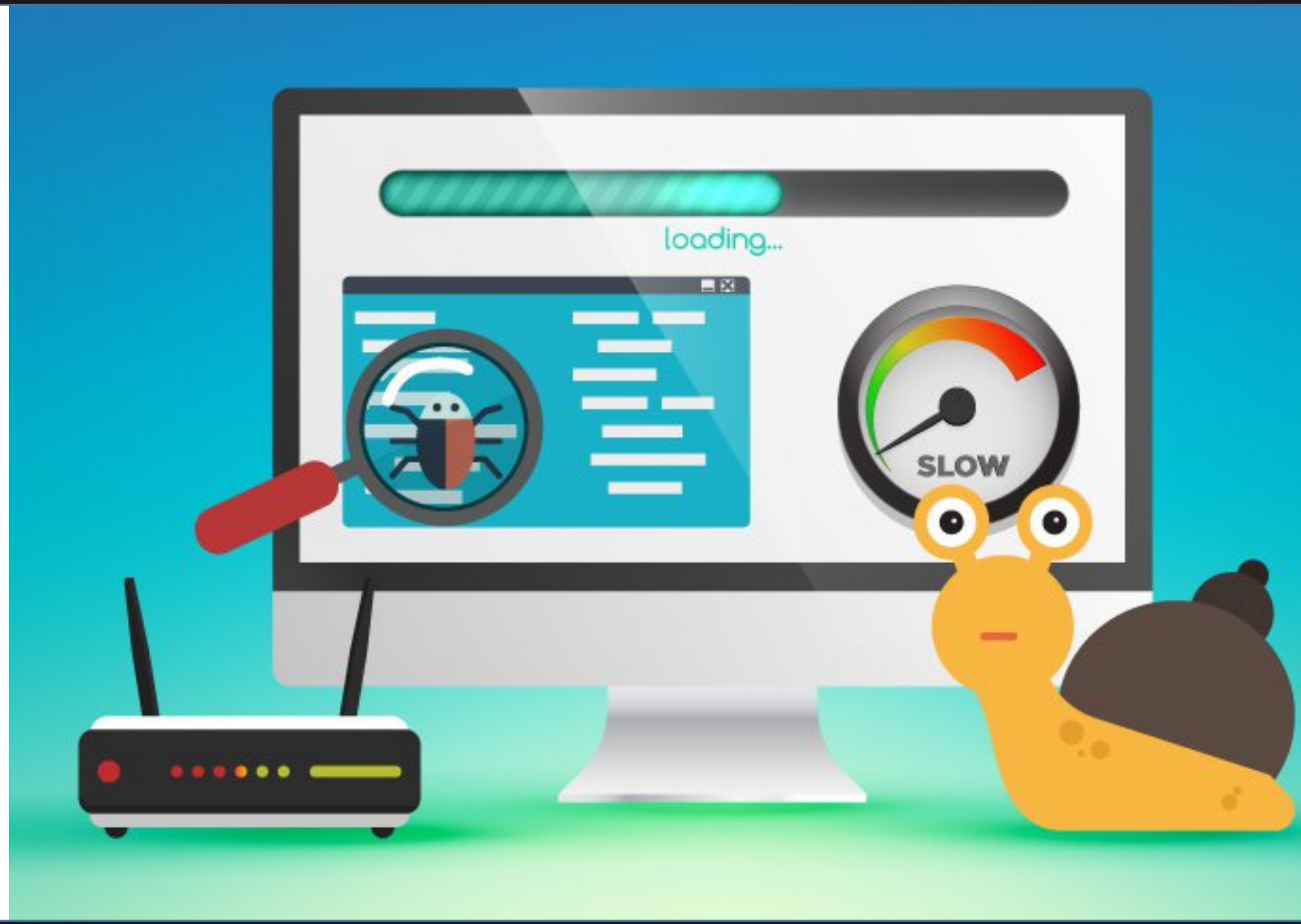
TSIW 2023/2024



AGENDA

1. Images;
2. Video;
3. Fonts.

IMAGES



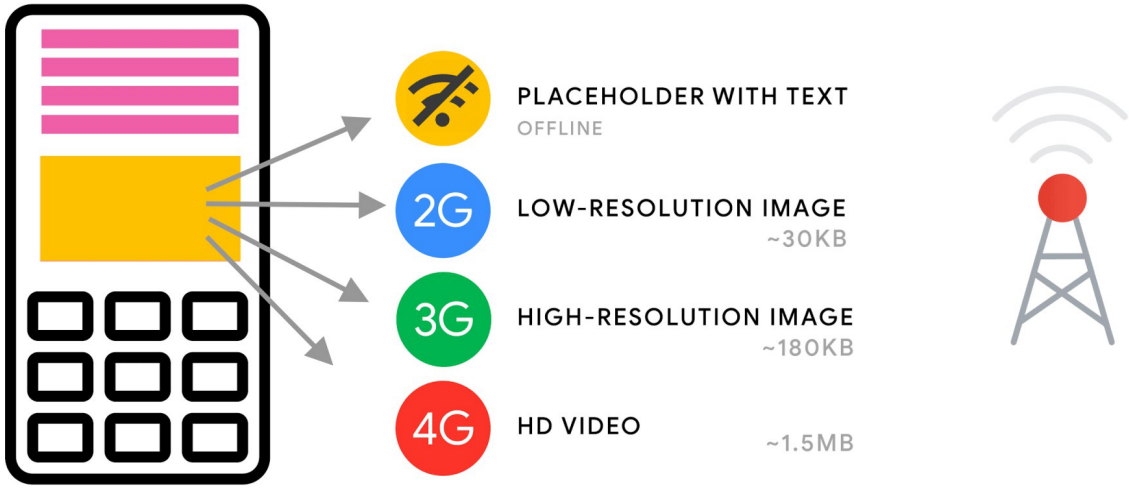
1. IMAGES

- Main Actions:
  - **Properly size images;**
  - Efficiently encode images;
  - Serve images in next-gen formats;
  - Defer off-screen images.

1. IMAGES

Properly size images

- Ideally your page should never serve images that are larger than the version that is rendered on the user screen;
- Anything larger just results in wasted bandwidth and slows down the page load time.



1. IMAGES

Properly size images

- Lighthouse opportunities section

▲ Properly size images — 9.77 s

Serve images that are appropriately-sized to save cellular data and improve load time. Learn more.

Show 3rd-party resources (0)

URL	Size	Potential Savings
 /flower_photo.jpg (correct-dimensions.glitch.me)	1,953 KB	1,918 KB
 /flower_logo.png (correct-dimensions.glitch.me)	71 KB	69 KB

1. IMAGES

Properly size images

- How lighthouse calculates oversize images:
  - For each image on the page, lighthouse compares the size of the rendered image against the size of the actual image;
  - The rendered size also accounts for device pixel ratio. If the rendered size is at least 4KB smaller than the actual size, then the image fails the audit.
- Strategies:
  - Serve responsive images;
  - Use images CDN's;
  - Use vector based image formats (e.g. SVG).

1. IMAGES

Properly size images

- Serve responsive images:
  - Serving desktop-sized images to mobile devices can use 2 to 4 times mores data than is needed;
  - Instead of a one-size-fits-all approach to images, serve different image sizes to different devices;
  - It is common to serve 3 to 5 different size of images;
  - Specify multiple image versions so browser will choose the best one to use.

```
<!-- Before -->
<img src="flower-large.jpg">

<!-- After -->
<img
    src="flower-large.jpg"
    srcset="flower-small.jpg 480w,
            flower-large.jpg 1080w"
    sizes="50vw"
>
```

1. IMAGES

Properly size images

- Serve responsive images:
  - SRC
    - Default option used if browser do not support srcset attribute;
    - Should be large enough to work well on all device sizes.
  - SRCSET
    - Comma-separated list of images filenames and their width or density descriptions;
    - Example: 480w tells browser that “flower-small.jpg” is 480px wide;
    - Just a way to tell browser the width of an image. This saves the browser from needing download the image to determinate its size.

1. IMAGES

Properly size images

- Serve responsive images:
  - SIZES
    - Tells the browser how wide the image will be when it is displayed;
    - The browser uses this information, along with what it knows about the user's device (screen size and resolutions), to determinate which image to load.

1. IMAGES

Properly size images

EXAMPLE 1

1. IMAGES

Properly size images

- Use images CDN's:
  - A content delivery network can provide a performance boost to any website by distributing the load and serving assets from locations geographically closer to users;
  - If your site is hosted on servers in California, and you don't have a CDN, all users must connect to the server in California directly;
  - Using a CDN service allows for example a user from Australia to access from closer servers hosted in Melbourne instead of servers in California.

1. IMAGES

Properly size images

- Use images CDN's:
  - CDN's may provide additional services regardless of your host's server facilities and limitations, such as:
    - SSL certificates for HTTPS encryption;
    - Load balancing, data compression, and HTTP/2 protocol for faster transmission;
    - Automatic file minification, image optimization, video transcoding, and email obfuscation.

1. IMAGES

Properly size images

- Use images CDN's:
  - Image CDN's can be used in addition to or instead of a standard CDN;
  - Popular options includes:
    - Cloudimage;
    - Cloudinary;
    - ImageEngine;
    - Imgix;
    - Imagekit.io;
    - Pixboost;
    - Uploadcare.

1. IMAGES

Properly size images

- Use images CDN's:
  - Main benefits:
    - **Assets management:** allow you to upload original images - perhaps directly from users - where they can be stored , optimized, and managed via a user interface or API;
    - **Optimal formatting and compressions:** regardless of the media uploaded, an image CDN can serve the file in the most optimal format. For example, you could upload a JPG image but have it served to Chrome and Firefox users in the more efficient WebP format. Browsers without WebP support would receive the next most appropriate image format.

1. IMAGES

Properly size images

- Use images CDN's:
  - Main benefits:
    - **Art direction, sizing, and effects:** offer an API that allows you to crop, resize, transform, or apply filters without affecting the original image.

1. IMAGES

Properly size images

- Use images CDN's:
  - Beyond the existence of image CDN's which let you generate multiple versions, either when you upload an image, or request it from your page;
  - There are some tools to convert images in multiple formats;
  - Some can help automate the process of converting:
    - gulp-responsive;
    - responsive-images-generator.

1. IMAGES

Properly size images

EXAMPLE 2

1. IMAGES

Properly size images

- Use vector based image formats:
  - Scalable Vector Graphics (SVG) define points, lines, and shapes as vectors in XML;
  - Unlike bitmaps, SVG images can be scaled to any dimensions without increasing the file size or losing quality. This makes them ideal for logos, charts, and diagrams;
  - It is possible to create and manipulate SVGs manually on client/server;
  - However, more complex images will require a graphics package such as Adobe Illustrator, Affinity Designer, Inkscape, or SVG edit, followed by an optimizing clean-up in svgo or SVGOMG.

1. IMAGES

Properly size images

- Use vector based image formats:
  - There are three primary ways to add an SVG to a web page:
    - Add SVGs using the <img> tag;
    - Add SVGs as CSS background images;
    - Embed SVGs into the page.

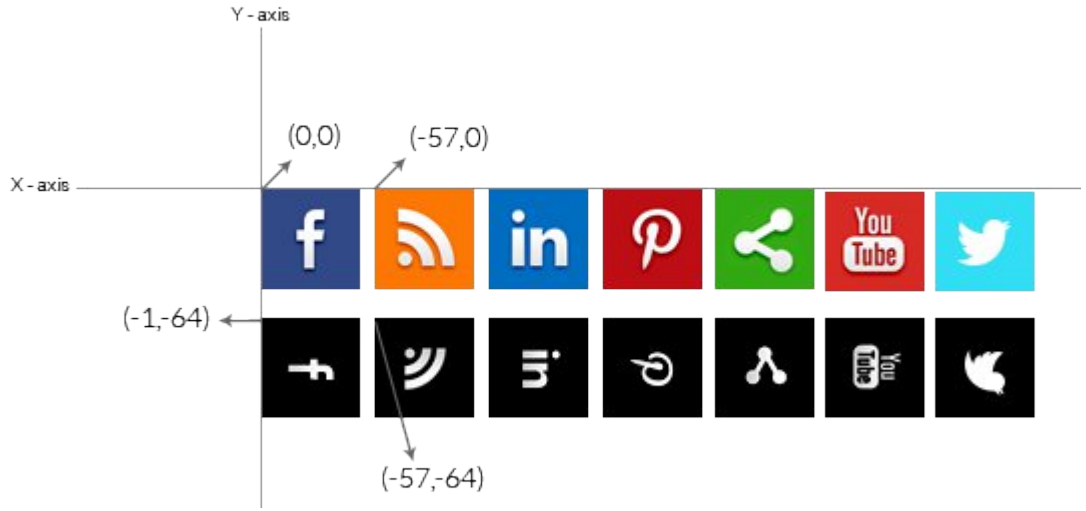
```css
.mysvgbackground {
    background-image: url('image.svg');
}
```

```html
<body>
    <svg class="mysvg" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 800 600">
        <circle cx="400" cy="300" r="50" />
    </svg>
</body>
```

1. IMAGES

Properly size images

- Consider Image Sprites:
  - Often-used images can be packaged into a single sprite file so individual items can be accessed in CSS. This is an old technique, but it continues to offer advantages:
    - A single HTTP request is required instead of several requests (one for each image);
    - A single image will normally result in a smaller file size than the total weight of the individual images;
    - All referenced images appear instantly after the sprite has loaded.



1. IMAGES

Properly size images

- Consider Image Sprites:
  - The following image defines 5 64x64px icons in a single image of 320x64px file:

  - Background position offsets are then defined CSS:

```css
.sprite {
    width: 64px;
    padding: 64px 0 10px 0;
    text-align: center;
    background: url("browser-sprite.png") 0 0 no-repeat;
  }

  .sprite.edge    { background-position: -64px  0; }
  .sprite.firefox { background-position: -128px 0; }
  .sprite.opera   { background-position: -192px 0; }
  .sprite.safari  { background-position: -256px 0; }
```

1. IMAGES

Properly size images

- Consider Image Sprites:
  - Then the individual images can be referenced in HTML using the right CSS classes:

```
<div class="sprite chrome">Chrome</div>
<div class="sprite edge">Edge</div>
<div class="sprite firefox">Firefox</div>
<div class="sprite opera">Opera</div>
<div class="sprite safari">Safari</div>
```

  - Image Sprites can be generated in a graphics package, using tools such as SpriteCow or Instant Sprite, or in your build process.

1. IMAGES

Properly size images

EXAMPLE 3

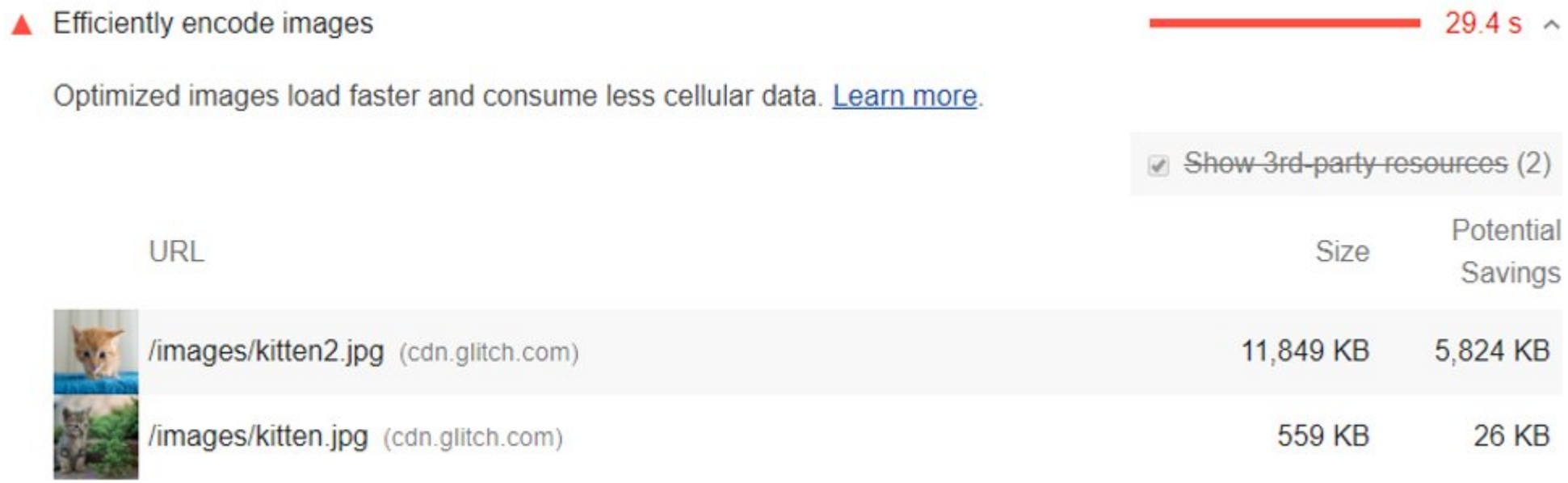
1. IMAGES

- Main Actions:
  - Properly size images;
  - **Efficiently encode images;**
  - Serve images in next-gen formats;
  - Defer off-screen images.

1. IMAGES

Efficiently encode images

- The opportunities section of your lighthouse report list all unoptimized images, with potential savings;
- Optimize these images so that the page loads faster and consumes less data.



1. IMAGES

Efficiently encode images

- How lighthouse flags images as optimizable?
  - Lighthouse collects all the JPEG or BMP images on the page, sets each image's compression level to 85, and then compares the original version with the compression version;
  - If the potential savings are 4KB or greater, lighthouse flags the image as optimizable.

1. IMAGES

Efficiently encode images

- Optimize images using GUI tools:
  - An approach is to run your images through an optimizer installed in your computer and run as GUI;
  - **ImageOtpim:** you drag and drop images into its UI, and then it automatically compresses the images without compromising quality noticeably;
  - **Squoosh:** is another option. Is maintained by the Google Web DevRel team.

1. IMAGES

Efficiently encode images

- Compress images:
  - Uncompressed images bloat your pages with unnecessary bytes.

20 KB

12 KB

1. IMAGES

Efficiently encode images

- Compress images:
  - Imagemin: is an excellent choice for image compression because it supports a wide variety of image formats and is easily integrated with build scripts and build tools. Is available as both a CLI and an npm module;
  - TinyPNG.

1. IMAGES

- Main Actions:
  - Properly size images;
  - Efficiently encode images;
  - **Serve images in next-gen formats;**
  - Defer off-screen images.

1. IMAGES

Serve images in next-gen formats

- The opportunities section of your lighthouse reports lists all images in older image formats, showing potential savings gained by serving WebP versions of those images.

1. IMAGES

Serve images in next-gen formats

- Choosing the correct format will radically reduce image file sizes. In general:
  - The JPG/JPEG format is best for photographs with intricate details;
  - The PNG format is best for logos, diagrams, and charts with solid block of color. The 8-bit 256-color format will normally result in smaller files if you don’t require 24-bit true-color or alpha transparency.

1. IMAGES

Serve images in next-gen formats

- You should also consider:
  - SVG define lines, paths, and shapes in XML rather then individual pixels. They are best suitable for logos and diagrams, since they can be scaled without losing quality;
  - GIF can be animated and sometimes result in smaller files than similar 8-bit PNGs;
  - WebP format can compress any type of image, but is not supported in IE, Safari and older browsers.

1. IMAGES

Serve images in next-gen formats

- Using WebP images:
  - WebP images are smaller than JPEG and PNG in about 25% to 35% in the file size;
  - This decreases page sizes and improves performance;
  - Examples:
    - Youtube found that switching to WebP thumbnails resulted in 10% faster page loads;
    - Facebook experienced a 25% to 35% file size savings for JPEGs and an 80% file size savings for PNGs when they switched to using WebP.

1. IMAGES

Serve images in next-gen formats

- Why serve images in WebP format?
  - JPEG2000, JPEG XR, and WebP are image formats that have superior compression and quality characteristics compared to their older JPEG and PNG counterparts;
  - Encoding your images in these formats rather than JPEG or PNG means that they will load faster and consumes less cellular data;
  - WebP is supported in Chrome and Opera and provides better lossy and lossless compression for images on the web.

1. IMAGES

Serve images in next-gen formats

- Using WebP images:
  - WebP is an excellent replacement for JPEG, PNG and GIF images;
  - In addiction, WebP offers both lossless and lossy compression:
    - In lossless compression no data is lost;
    - Lossy compression reduces file size, but at the expense of possibly reducing image quality.


JPEG file size: 251.03 KB


WebP file size: 172.82 KB

1. IMAGES

Serve images in next-gen formats

- How lighthouse calculates the potential savings?
  - Lighthouse collects each BMP, JPEG or PNG image on the page, and then converts each one of them to WebP, reporting the potential savings based on the conversation figures;
- Browser compatibility:
  - Once WebP is not supported by all browsers you need to serve a fallback PNG or JPEG image.

```
<picture>
    <source type="image/webp" srcset="flower.webp">
    <source type="image/jpeg" srcset="flower.jpg">
    <img src="flower.jpg" alt="">
</picture>
```

1. IMAGES

Serve images in next-gen formats

WebP image format - UNOFF

Image format (based on the VP8 video format) that supports lossy and lossless compression, as well as animation and alpha transparency. WebP generally has better compression than JPEG, PNG and GIF and is designed to supersede them. AVIF and JPEG XL are designed to supersede WebP.

Usage: Global — % of all users: 90.11% + 3.22% = 93.33%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	Safari on iOS *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
			4-8		10.1											
			1 9-22		1 11.5			2.1-3								
	12-17	2-64	2 23-31		2 12.1-18			1 4-4.1								
6-10	18-88	65-85	32-88	3.1-13.1	19-72	3.2-13.7		4.2-4.4.4	12-12.1				4-12.0			
11	89	86	89	3 14	73	14.5	all	89	62	89	86	12.12	13.0	10.4	7.12	2.5
		87-88	90-92	3 TP												

1. IMAGES

Serve images in next-gen formats

- Convert to WebP format tools:
  - Generic tools such as Photoshop or Gimp;
  - Cwebp command-line tool;
  - Imagemin WebP plugin (npm).

1. IMAGES

Serve images in next-gen formats

EXAMPLE 4

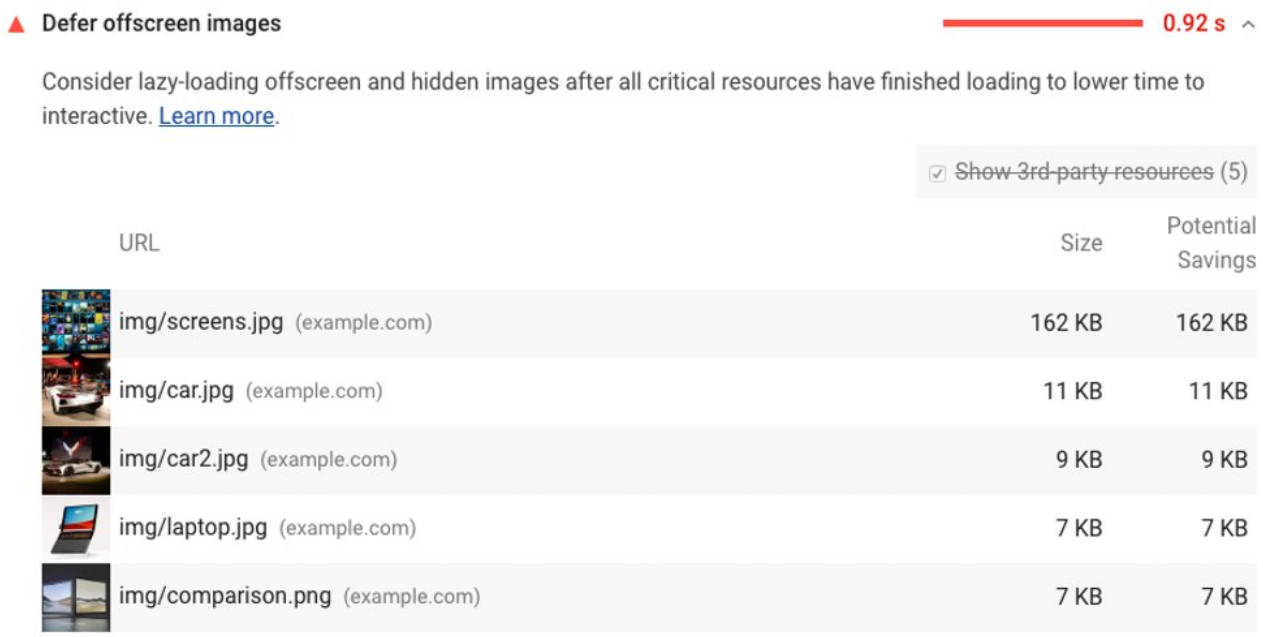
1. IMAGES

- Main Actions:
  - Properly size images;
  - Efficiently encode images;
  - Serve images in next-gen formats;
  - **Defer off-screen images.**

1. IMAGES

Defer off-screen images

- The lighthouse opportunities section list all off-screen or hidden images in your page along with the potential savings;
- Consider lazy-loading these images after all critical resources have finished loading to lower the TTI.



▲ Defer offscreen images — 0.92 s

Consider lazy-loading offscreen and hidden images after all critical resources have finished loading to lower time to interactive. Learn more.

☑ ~~Show 3rd-party resources~~ (5)

URL	Size	Potential Savings
img/screens.jpg (example.com)	162 KB	162 KB
img/car.jpg (example.com)	11 KB	11 KB
img/car2.jpg (example.com)	9 KB	9 KB
img/laptop.jpg (example.com)	7 KB	7 KB
img/comparison.png (example.com)	7 KB	7 KB

1. IMAGES

Defer off-screen images

- Lazy-loading images:
  - Browser-level support for lazy-loading images is now supported on the web;
  - Supported by Chromium-powered browsers (Chrome, Edge, Opera) and Firefox;
  - According to HTTP Archive, images are the most requested asset type for most websites and usually take up more bandwidth than any other asset;
  - At the 90th percentile, sites send about 4.7MB of images on desktop and mobile;
  - Before lazy-loading support, there has two ways to defer the loading of off-screen images:
    - Using the Intersection Observer API;
    - Using scroll, resize and orientation change event handlers.

1. IMAGES

Defer off-screen images

- Lazy-loading images:
  - With lazy-loading supported directly by the browser, there is no need for that;
  - Browser-level lazy-loading also ensures that deferred loading of images still works even if JavaScript is disabled on the client.

1. IMAGES

Defer off-screen images

- Lazy-loading images:
  - **Loading Attribute:**
    - Chrome already loads images at different priorities depending on where they are located with respect to the device viewport;
    - Images bellow the viewport are loaded with a lower priority, but they are still fetched as soon as possible;
    - In Chrome 76+, you can use the loading attribute to completely defer the loading of off-screen images that can be reached by scrolling.

```
<img src="image.png" loading="lazy" alt="…" width="200" height="200">
```

1. IMAGES

Defer off-screen images

- Lazy-loading images:
  - **Loading Attribute Values:**
    - **auto:** default lazy-loading behaviour of the browser, which is the same as not including the attribute;
    - **lazy:** defer loading of the resource until it reaches a calculated distance from the viewport;
    - **eager:** load the resource immediately, regardless of where it is located on the page.

1. IMAGES

Defer off-screen images

- Lazy-loading images:
  - **Dimension Attributes:**
    - Images should include dimensions attributes;
    - While the browser loads an image, it does not immediately know the image's dimension, unless these are explicitly specified;
    - To enable the browser to reserve the sufficient space on a page for images, it is recommended that all <img> tags includes both with and height attributes;
    - Without dimensions specified, layout shifts can occur.

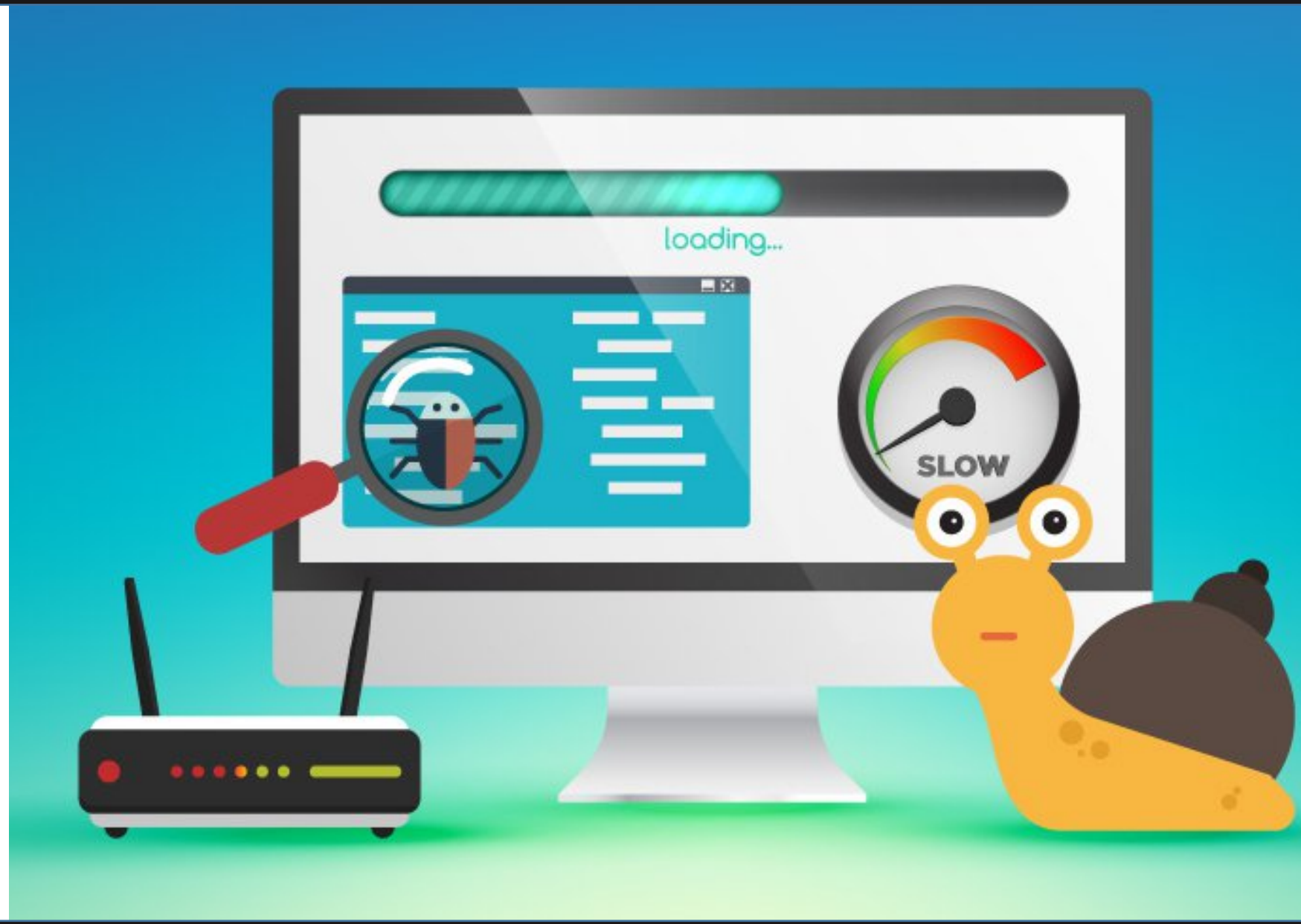
```
<img src="image.png" loading="lazy" alt="…" width="200" height="200">

<img src="image.png" loading="lazy" alt="…" style="height:200px; width:200px;">
```

1. IMAGES

Defer off-screen images

EXAMPLE 5

VIDEO



2. VIDEO

- Video can offer an engaging experience, although they have a higher bandwidth and performance cost than any other web asset;
- For the average website, 25% of the bandwidth comes from video;
- Optimizing video has the potential for very large bandwidth savings;
- Some recommendations:
  - Do you need to play the video to all visitors? Remove media assets when it is possible;
  - Ensure the video is as short as possible;
  - Transcode the video into multiple formats using the minimum dimensions with optimal compression. Many video CDNs and services will handle it for you;
  - Only play the video on demand - not automatically when user accesses the website.

2. VIDEO

- Main Actions:
  - **Compress videos;**
  - Optimize <source> order;
  - Video autoplay;
  - Remove audio from muted hero videos;
  - Video preload;
  - Play audio and video on demand;
  - Use video formats for animated content.

2. VIDEO

Compress videos

- Most video compression software compares adjacent frames within a video with the intent of removing detail that is identical in both frames;
- Compress the video and export to multiple video formats: WebM, MPEG-4/H.246, and Ogg/Theora;
- Suggestions:
  - Reduce file size in your video editing software;
  - Use online tools such as FFmpeg that encode, decode, convert, and perform other optimization functions.

2. VIDEO

- Main Actions:
  - Compress videos;
  - **Optimize <source> order;**
  - Video autoplay;
  - Remove audio from muted hero videos;
  - Video preload;
  - Play audio and video on demand;
  - Use video formats for animated content.

2. VIDEO

Optimize <source> order

- Order video source from smallest to largest;

```
<video width="400" height="300" controls="controls">
    <!-- WebM: 10 MB -->
    <source src="video.webm" type="video/webm" />
    <!-- MPEG-4/H.264: 12 MB -->
    <source src="video.mp4" type="video/mp4" />
    <!-- Ogg/Theora: 13 MB -->
    <source src="video.ogv" type="video/ogv" />
</video>
```

- The browser downloads the first formats it understands;
- The goal is to offer smaller versions ahead of larger versions;
- With the smallest version, make sure that the most compressed video stills look good.

2. VIDEO

- Main Actions:
  - Compress videos;
  - Optimize <source> order;
  - **Video autoplay;**
  - Remove audio from muted hero videos;
  - Video preload;
  - Play audio and video on demand;
  - Use video formats for animated content.

2. VIDEO

Video autoplay

- To ensure that a looping background video autoplays, you must add several attributes to the <video> tag: autoplay, muted, and playsinline;

```
<video autoplay="" loop="" muted="true" playsinline="" src="backgroundvideo.mp4">
```

- While the loop and autoplay make sense for a looping and autoplaying video, the muted attribute is required for autoplay in mobile devices;
- Playsinline is required for mobile Safari browser, allowing videos to play without forcing fullscreen mode.

2. VIDEO

- Main Actions:
  - Compress videos;
  - Optimize <source> order;
  - Video autoplay;
  - **Remove audio from muted hero videos;**
  - Video preload;
  - Play audio and video on demand;
  - Use video formats for animated content.

2. VIDEO

Remove audio from muted hero videos

- For hero-video or other video without audio removing audio can saves 20% of the bandwidth;

```
<video autoplay="" loop="" muted="true" playsinline="" id="hero-video">
    <source src="banner_video.webm" type='video/webm; codecs="vp8, vorbis"'>
    <source src="web_banner.mp4" type="video/mp4">
</video>
```

- Characteristics:
  - This is common to conference websites and corporate web pages;
  - It includes a video that is autoplaying, looping and muted;
  - There are no controls, so there is no other way to hear audio;
  - The audio is often empty, but still present, and will use bandwidth;
  - There is no reason to serve audio with video that is always muted.

2. VIDEO

- Main Actions:
  - Compress videos;
  - Optimize <source> order;
  - Video autoplay;
  - Remove audio from muted hero videos;
  - **Video preload;**
  - Play audio and video on demand;
  - Use video formats for animated content.

2. VIDEO

Video preload

- Control how much of a video file downloads with page load;
- You can save data by deferring download for less popular videos;
- The preload attribute has three available options:
  - metadata (default): may download up to 3% of the videos on page load. This is a useful option for some small or moderately size files;
  - auto: tells the browser to automatically download the entire video. Do this only when playback is very likely, otherwise it wastes a lot of bandwidth;
  - none: results in none of the video being downloaded until playback. It delays startup but offers significant data savings for videos with a low probability of playback.

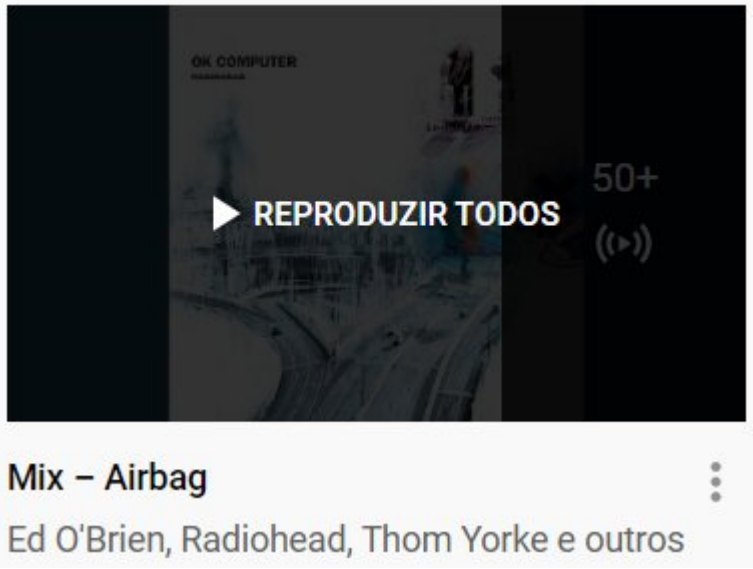
2. VIDEO

- Main Actions:
  - Compress videos;
  - Optimize <source> order;
  - Video autoplay;
  - Remove audio from muted hero videos;
  - Video preload;
  - **Play audio and video on demand;**
  - Use video formats for animated content.

2. VIDEO

Play audio and video on demand

- Auto-playing media saps bandwidth, degrades performance, and is unlikely to be appreciated by users;
- Modern browsers will also block or silence auto-playing by default;
- In most cases, it is preferable to show a thumbnail image - perhaps with a play icon overlay - which the user can click to starts the media.



2. VIDEO

Play audio and video on demand

- Both the <video> and <audio> tags support this feature with the attributes:
  - **autoplay=”false”:** to stop auto-playing;
  - **preload=”none”:** to prevent media preloading or **preload=”metadata”** to fetch meta data such as the video duration;
  - **poster=”image.png”:** to show a thumbnail image;
  - **controls="true”:** to enable native playback controls.

```
<video controls="true" autoplay="false" preload="metadata" poster="videothumb.jpg">
    <source src="video.mp4" type="video/mp4">
    <source src="video.webm" type="video/webm">
</video>
```

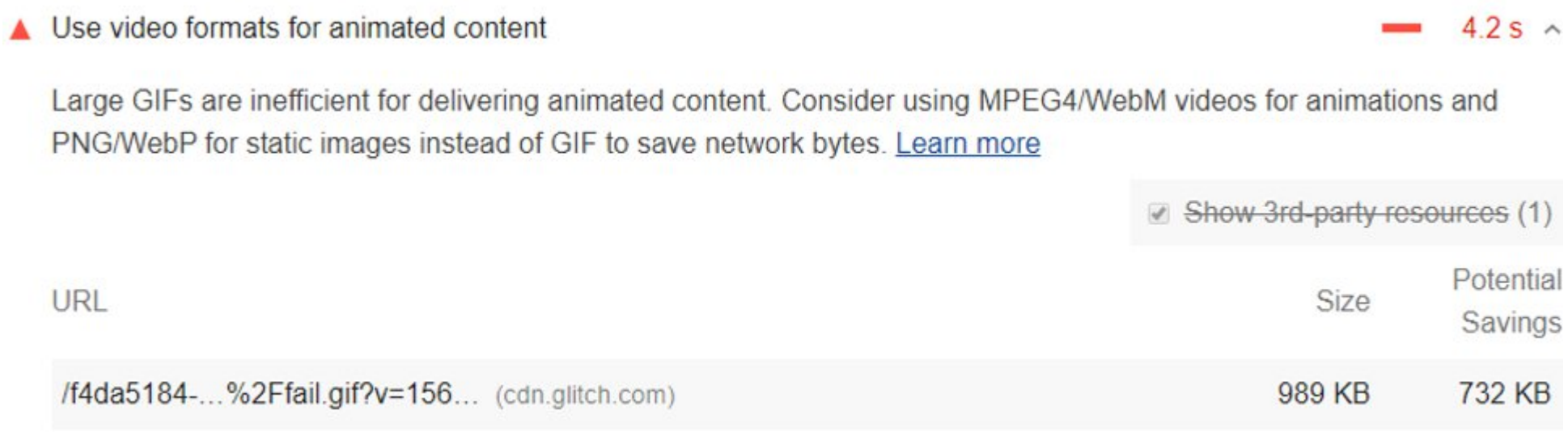
2. VIDEO

- Main Actions:
  - Compress videos;
  - Optimize <source> order;
  - Video autoplay;
  - Remove audio from muted hero videos;
  - Video preload;
  - Play audio and video on demand;
  - **Use video formats for animated content.**

2. VIDEO

Use video formats for animated content

- The opportunities section of lighthouse report lists all animated GIFs, along with estimated savings achieved by converting these GIFs into videos.

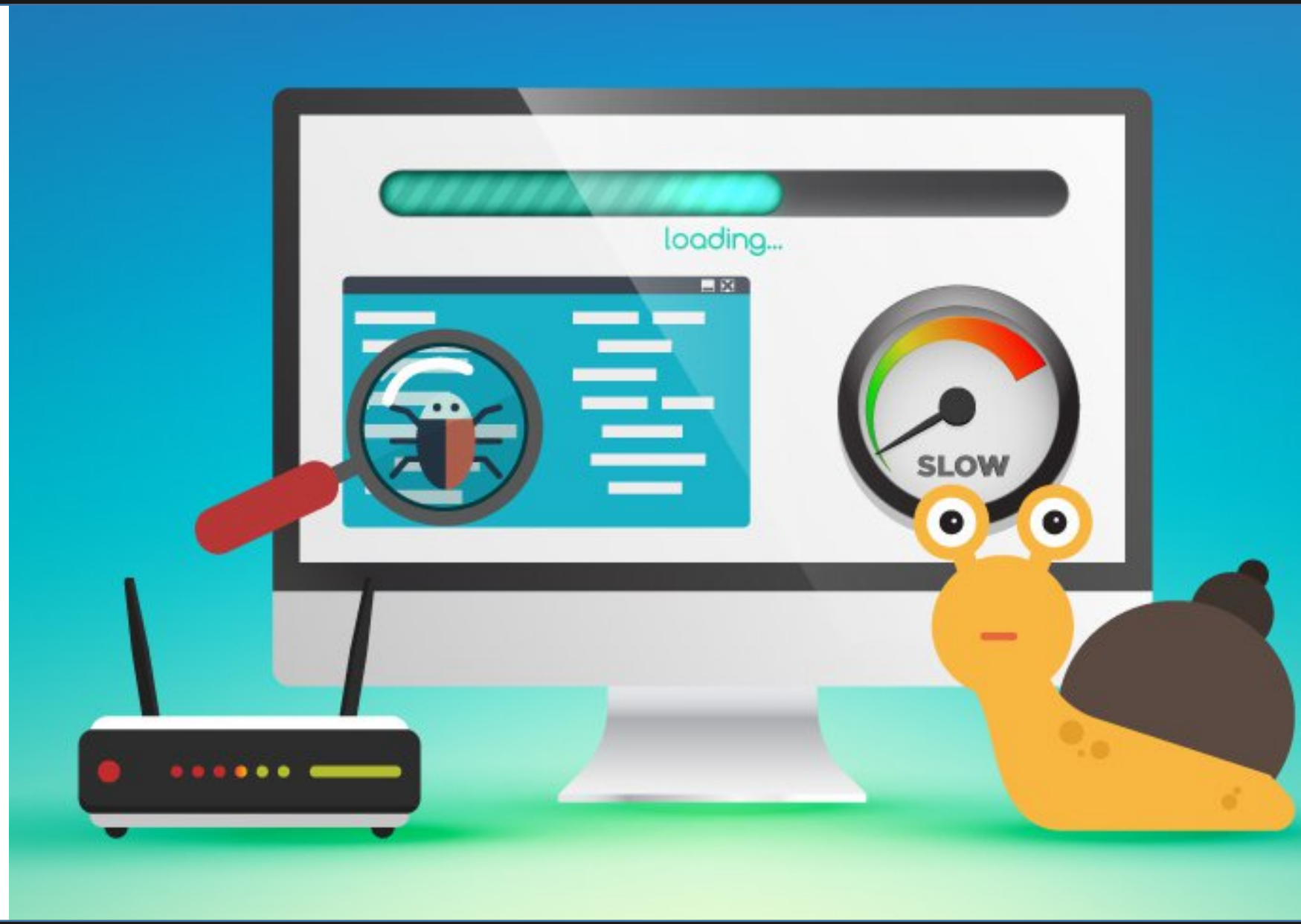


2. VIDEO

Use video formats for animated content

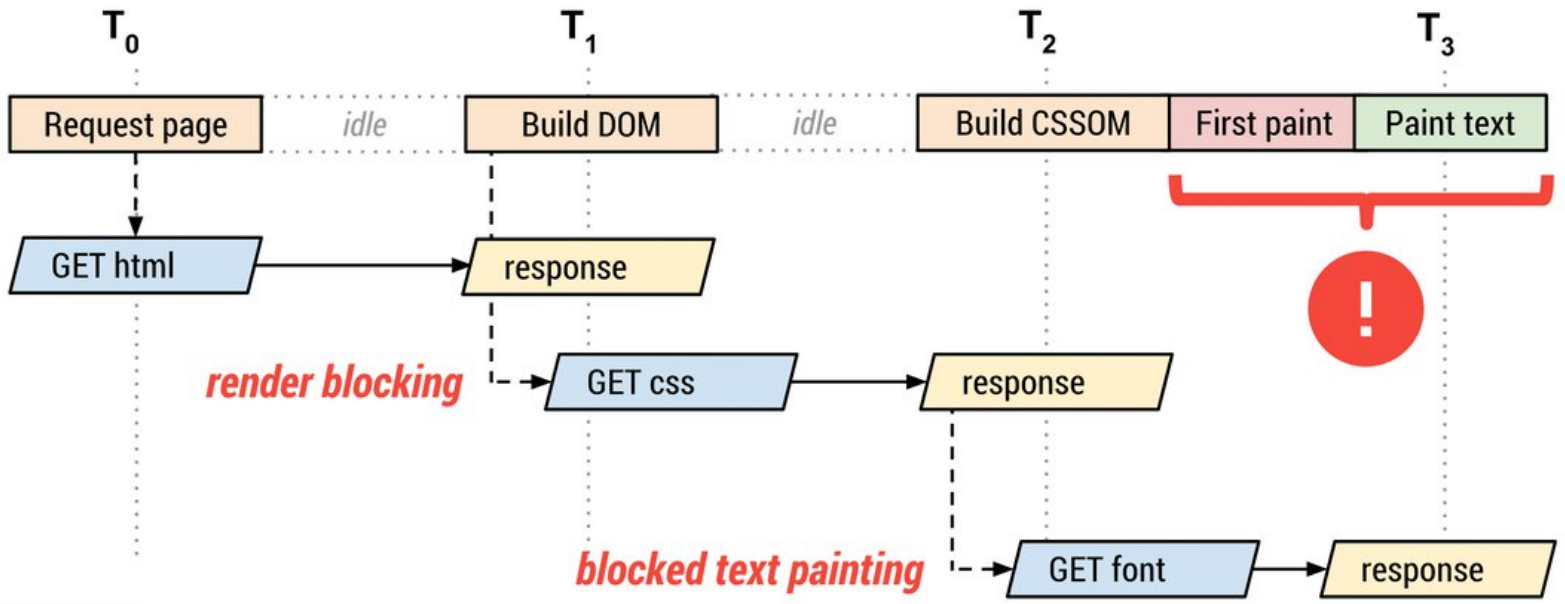
- Large GIFs are inefficient for delivery animated content;
- By converting large GIFs to videos, you can save bandwidth;
- Consider:
  - Using MPEG4/WebM videos for animations;
  - Using PNG/WebP for static images instead of GIFs to save bandwidth.

FONTS



3. FONTS

- It is possible to add dozens of fonts to a page, but that does not mean you should;
- A custom font topically requires a few hundreds KB of data. The more you add, the larger the page weight, and lowers the page performance;
- Fonts are often large files that take awhile to load.



3. FONTS

1. The browser requests the HTML document;
2. The browser begins parsing the HTML response and constructing the DOM;
3. The browser discovers CSS, JS, and other resources and dispatch requests;
4. The browser constructs the CSSDOM after all of the CSS content is received and combines it with the DOM tree to construct the render tree;
    1. Font requests are dispatched after the render tree indicates which font variants are needed to render the specified text on the page.
5. The browser performs layout and paints content to the screen.

3. FONTS

- This situation creates the “blank text problem” where the browser might render page layout but omits any text;
- By preloading WebFonts and using font-display to control how browsers behave with unavailable fonts, you can prevent blank pages and layout shifts due to font loading;
- Using an OS font provides a noticeable performance boost;
- There is no downloading delay or flash of unstyled or invisible text;
- Each platform supplies different default fonts, but fallbacks can be specified as well as the generic font family names of serif, sans-serif, monospace, cursive, fantasy, and system-ui.

```
body {
    font-family: Arial, Helvetica, sans-serif;
}
```

3. FONTS

- Alternatively, the CSS @font-face local() function can be used to locate a font on the user's system first, but load from an URL when it cannot be founded;

```
@font-face {
    font-family: MyHelvetica;
    src: local("Helvetica Neue"),
         local("HelveticaNeue"),
         url("/fonts/Helvetica-webfont.woff2") format("woff2"),
         url("/fonts/Helvetica-webfont.woff") format("woff");
  }

  body {
    font-family: 'MyHelvetica', sans-serif;
  }
```

- An OS font should be the first choice if closely matching branding requirements.

3. FONTS

Embed Web Fonts with <link>

- Many designers will be horrified by using OS fonts, so web font use is inevitable;
- The most popular option is to use a repository that serves fonts from a CDN:
  - Google Fonts;
  - Font Library;
  - Adobe Edge.
- When possible, load fonts using a <link> in your HTML <head>.

```
<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">
```

3. FONTS

Limit font styles and text

- Only request the fonts, weights, and styles you require - and definitely remove any fonts you are not using;

https://fonts.googleapis.com/css?family=Inconsolata:500,700

https://fonts.googleapis.com/css?family=Roboto:bolditalic

https://fonts.googleapis.com/css?family=Inconsolata:500,700%7CRoboto:bolditalic

- In some cases, you may only need specific characters - perhaps for a regularly used title or logo;

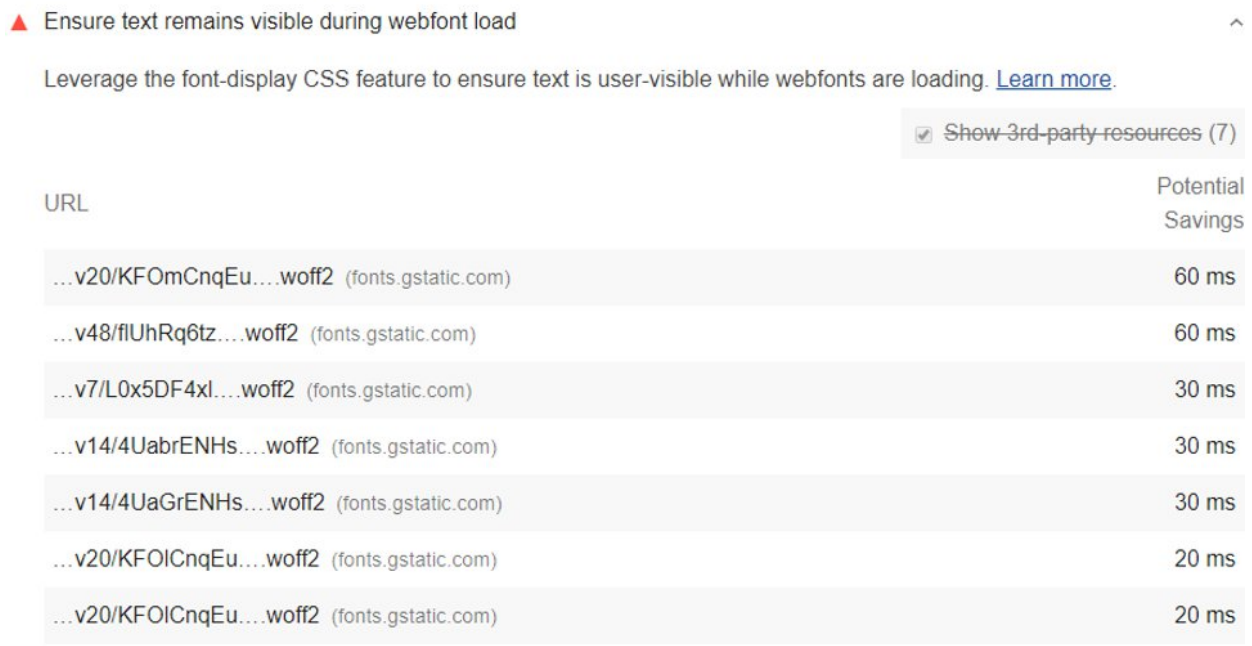
https://fonts.googleapis.com/css?family=Inconsolata&text=Helo

- Finally, you could benefit from hosting the fonts locally or using more popular fonts that have a higher chance of being pre-cached in the user's browser.

3. FONTS

Ensure text remains visible during web font loads

- Fonts are often large files that take awhile to load;
- Some browsers hide text until the font loads, causing a flash of invisible text (FOIT);
- Lighthouse flags any font URLs that might flash invisible text.



3. FONTS

Use a good font-loading strategy

- A web font can take several seconds to download. The browser will choose one or two options:
  - Show a flash of unstyled text (FOUT). The first available font fallback is used immediately. It is replaced by the web font once it is loaded. This process is used by IE; Edge 18 and bellow, and older editions of Firefox and Opera;
  - Show a flash of invisible text (FOIT). No text is displayed until the web font has loaded. This process is used in all modern browsers, which typically wait three seconds before reverting to a fallback.

3. FONTS

Use a good font-loading strategy

- The CSS font-display property allows you to define the font-handling process:
  - **auto:** the browser's default behaviour (usually FOIT);
  - **block:** effectively FOIT which means text will be invisible until web font load. If it fails it will use the fallback font after 3 seconds;
  - **swap:** effectively FOUT. Browser will use fallback font until web font has been loaded;
  - **fallback:** a compromises between FOIT and FOUT. Text is invisible for a short period (usually 100ms) then the first fallback is used until the web font is available;
  - **optional:** the same as fallback, except no font swapping occurs.

3. FONTS

Use a good font-loading strategy

- The easiest way to avoid showing invisible text while custom fonts load is to temporarily show a system font:

```
@font-face {
    font-family: 'mytypeface';
    src: url('mytypeface-webfont.woff2') format('woff2'),
         url('mytypeface-webfont.woff') format('woff');
    font-weight: 500;
    font-style: normal;
    font-display: swap;
}
```

https://fonts.googleapis.com/css?family=Inconsolata:500,700&display=swap