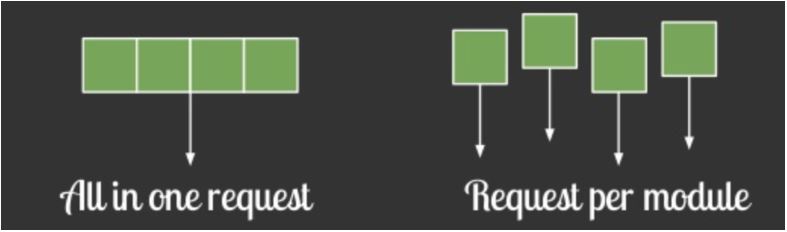
**LAZY LOADING:**

As the name suggests lazy loading is a process of loading parts (chunks) of your application lazily. In other words — loading them only when we really need them. Code splitting is just a process of splitting the app into this lazily loaded chunks.



**LAZY LOAD OFFSCREEN IMAGES:**

Lazy loading is a technique used to prevent or delay the loading of non-critical resources until they are needed. You can use this mechanism for different types of resources, but in the case of images, our goal is to lazily load everything that is not visible to the user within the initial viewport. All other images can be loaded when the user scrolls down the page.

Use the loading="lazy" attribute to load an image lazily. It also works for the <nuxt-img> component.

**<img src="..." loading="lazy">**

**<nuxt-image src="..." loading="lazy" />**

**LAZY LOADING VUEX MODULES:**

Let’s say we have testimonials section on our **Home.vue** where we want to display positive opinions about our services. There’s a lot of them so we don’t want to show them right after user enters our website. It’s much better to display them only if user want it. We can add “Show Testimonials” button that will load and display the testimonials below it after clicked.



To store testimonials data we need one more Vuex module. Let’s call it **testimonials**. The module will be responsible for showing previously added testimonials and adding new one. We don’t need to know about implementation details.

We want **testimonials** module to be downloaded ONLY if user clicks the button since it’s not needed before. Let’s see how we can make use of dynamic module registration and dynamic imports to achieve this functionality. **Testimonials.vue** is a component inside **Home.vue.**

****

Let’s quickly review the code.

When user clicks Show Testimonials button getTestimonials() method is invoked. It is responsible for invoking getTestimonialsModule() which is fetching testimonials.js. Once promise is resolved (which means module is loaded) we are dynamically registering it and dispatching action responsible for fetching testimonials.

Thanks to dynamic imports testimonials.js content is bundled into a separate file that is downloaded only when getTestimonialsModule method is invoked.