Element loading approach →	Basic approach: 1. Load all elements directly	Sleep approach: 1. Apply sleep (0.1 seconds) 2. Load a batch of element,	Await JS approach: 1. Await JS and wait for reply 2. Load a batch of element,
BEFORE CONNECTION	• Slow page load, • Large document size	• Does not fix the issue • Adds delay instead	• HANGS THE PAGE!!! • DO NOT USE!!!
Recommended Approach	➔ Show as many elements as possible if time permits, because it is better than after-connection loading ➔ At the very least, populate enough elements to have something on the screen immediately, scrolling otherwise		
AFTER CONNECTION 1. **Show spinner** 2. **Load data** 3. **Remove spinner**	• One single large lag spike. • Spinner stops spinning. • All the data is ready at once.	• Several small lag spikes • Browser chokes more and more as DOM tree grows, but server delay remains constant. • As such, spinner stutters more and more • But this is agnostic to network delay	• Several small lag spikes • Browser chokes more and more as DOM tree grows, server waits for browser. • As such, spinner stuttering remains constant (mostly) • But higher the network delay, slower the loading
Recommended Approach	➔ For small amounts of data, loading at once is better than the overhead with multiple batches	➔ For medium amounts of data, where the DOM tree doesn't grow a lot, sleep in-between batches can suffice ➔ Or, for poor network delay situations	➔ For large amounts of data, where the DOM tree grows so much, handling each message takes more than 0.1 seconds ➔ And, for low network delay situations

Element definition approach →	Basic approach: 1. Load all elements directly	Simple Custom Element: 1. Custom Vue Component	Advanced Custom Element 1. Custom Vue Component 2. Attach IntersectionObserver
PROS	• Maintain access to NiceGUI features (value binding, etc.)	• Reduce sent document size	• Reduce sent document size • Reduce DOM element count
CONS	• Large DOM element count • More browser JS workload (esp. for nested elements)	• Lose access to NiceGUI features (value binding, etc.) • Harder to develop	• Lose access to NiceGUI features (value binding, etc.) • Harder to develop • Must know size beforehand