TIPS FOR REAL-WORLD ALPINEJS

Tackle Alpine FAQs, common issues & web patterns.

Hugo Di Francesco (@hugo__df)

Founder Code with Hugo & AlpineJS Weekly

Slides: codewithhugo.com/alpine-tips

ABOUT

React/Node background I liked how easy it was to get going with Alpine.

Contributed to Alpine: typos, bugfixes, new modifiers, error handling.

Also involved in the community devtools.

CONTENTS

1. WHAT EVEN IS `Proxy {}`?

Demo: https://codepen.io/hugodf/pen/dyvqPbw

```
<div x-data="{ obj: { count: 1 } }">
  <button @click="console.log(obj)">Proxy</button>
</div>
```

On click, console output:

```
Proxy { <target>: {}, <handler>: {} }
```

Proxy: a JavaScript object which enables you to wrap JS objects and intercept operations on it

This is core to Alpine's reactivity, proxies allow Alpine to detect data updates.

HOW DO I USE THE DATA IN THE PROXY?

Demo: https://codepen.io/hugodf/pen/vYxzEEw

```
<div x-data="{ obj: { count: 1 } }">
  <button @click="obj.count++" x-text="obj.count"></button>
</div>
```

The same as you would use the data if it wasn't in a proxy.

HOW DO I PRINT THE DATA IN THE PROXY?

Demos: https://codepen.io/hugodf/pen/yLMxyeo

Print it as a string

```
<button
  @click="console.log(JSON.stringify(obj))"

>
  stringified
</button>
```

Output: `'{"count":1}'` (string)

Unfurl the proxy

```
<button
  @click="console.log(JSON.parse(JSON.stringify(obj)))"
>

  unfurled
</button>
```

Output: `Object { count: 1 }` (JavaScript Object)

DEBUGGING TIP

```
<pre x-text="JSON.stringify(obj, null, 2)"></pre>
```

Don't forget `null, 2` it does the pretty-printing.

Output:

```
{
  "count": 1
}
```

2. FETCH DATA

How do I load data with Alpine?

The `fetch` API is a native way to load data in modern browsers from JavaScript.

I want to load & list books that match "Alpine" from Google Books API

Let me initialise a `books` array

```
<div
  x-data="{
    books: []
  }"
>
</div>
```

On component startup (`x-init`), load data from Google Book Search API & extract the response

```
<div
  x-data="{
    books: []
  }"

  x-init="
    fetch('https://www.googleapis.com/books/v1/volumes?q=Alpin
      .then(res => res.json())
      .then(res => console.log(res))
  "
>
</div>
```

https://codepen.io/hugodf/pen/BaWOrMX

Output:


```
{
  "kind": "books#volumes",
  "totalItems": 1873,
  "items": [
    {
      "kind": "books#volume",
      "id": "i6H0ESHr3n8C",
      "etag": "ct+4Nub4wP4",
      "selfLink": "https://www.googleapis.com/books/v1/volumes/i6H0ESHr3n8C",
      "volumeInfo": {
        "title": "Alpine Climbing",
        "subtitle": "Techniques to Take You Higher",
        "authors": [
          "Mark Houston",
          "Kathy Cosley"
        ],
```

Store the `volumeInfo` of each `items` as `books`.

```
<div
  x-data="{
    books: []
  }"
  x-init="
    fetch('https://www.googleapis.com/books/v1/volumes?q=Alpin
      .then(res => res.json())
      .then(res => {
        books = res.items.map(item => item.volumeInfo)
      })
  "
>
  <div x-text="JSON.stringify(books)"></div>
</div>
```

https://codepen.io/hugodf/pen/QWpVwXQ

Output:

[{"title":"Alpine Climbing","subtitle":"Techniques to Take You Higher","authors":["Mark Houston","Kathy Cosley"],"publisher":"The Mountaineers
Books","publishedDate":"2004","description":"* For climbers who know the basics and are ready to venture at higher altitudes* Written by longtime guides and climbing instructors certified by the American Mountain Guide Association (AMGA)* Teaches situational thinking and learning as well as techniqueThis intermediate-level guide addresses tools, skills, and techniques used in alpine terrain including rock, snow, ice, and glaciers at moderate altitude - approximately 5000 meters (16,000 feet) and lower. The technical protection systems are covered, of course. But 30 years of alpine climbing experience has convinced the authors that mastery - and safety - lie in the far more difficult task of knowing exactly which techniques to use, where and when. Therefore, they teach step-by-step decision-making skills, providing scenarios, checklists, and self-posed questions to inform the decision process. Alpine Climbing assumes some prior knowledge, primarily in rock climbing skills and techniques. Basic knots, belaying,rappelling, building rock anchors, leading, placing rock protection, and movement skills on rock: variations of these skills that are of particular value in the alpine environment are addressed in this book.","industryIdentifiers":[{"type":"ISBN_10","identifier":"0898867495"},
{"type":"ISBN_13","identifier":"9780898867497"}],"readingModes":{"text":true,"image":false},"pageCount":325,"printType":"BOOK","categories":["Sports &
Recreation"],"averageRating":3.5,"ratingsCount":3,"maturityRating":"NOT_MATURE","allowAnonLogging":true,"contentVersion":"2.1.4.0.preview.2","panelizationSummary":
{"containsEpubBubbles":false,"containsImageBubbles":false},"imageLinks":{"smallThumbnail":"http://books.google.com/books/content?id=i6H0ESHr3n8C&printsec=frontcover&img=1&
zoom=5&edge=curl&source=gbs_api","thumbnail":"http://books.google.com/books/content?id=i6H0ESHr3n8C&printsec=frontcover&img=1&zoom=1&edge=curl&

We can clean up the output with `x-for` + `x-text` instead of dumping the data.

```
<ul>
  <template x-for="book in books">
    <li x-text="book.title"></li>
  </template>
</ul>
```

https://codepen.io/hugodf/pen/YzZOKgE

Output:

- Alpine Cooking
- Alpine Climbing
- Tectonic Aspects of the Alpine-Dinaride-Carpathian System
- Structure and Function of an Alpine Ecosystem
- The Amazing Tale of Mr. Herbert and His Fabulous Alpine Cowboys Baseball Club
- Alpine & Renault
- Alpine Caving Techniques
- Alpine Renault
- The Alpine Glee Singer
- Alpine Winter in Its Medical Aspects

HOW ABOUT A LOADING STATE?

Pattern:

1. action causes a data load
2. set loading = true & start the data load
3. receive/process the data
4. loading = false, data = newData

```
<div
  x-data="{
    books: [],
    isLoading: false
  }"
  x-init="
    isLoading = true;
    fetch('https://www.googleapis.com/books/v1/volumes?q=Alpin
      .then(res => res.json())
      .then(res => {
        isLoading = false;
        books = res.items.map(item => item.volumeInfo)
      })
  "
```

```
<div x-show="isLoading">Loading...</div>
<ul>
  <template x-for="book in books">
    <li x-text="book.title"></li>

  </template>
</ul>
```

https://codepen.io/hugodf/pen/dyvqPxY

Promises/data fetching in JavaScript can easily fill a whole other talk.

See codewithhugo.com/async-js for a deeper look at topics such as fetching in parallel & delaying execution of a Promise.

3. SEND AND HANDLE EVENTS

One of the other key Alpine features: the ability to send and receive events using `x-on` + `$dispatch`.

ALPINE -> ALPINE EVENTS

```
$dispatch('event-name', 'event-data')
```

Creates and sends an "event-name" event with "event-data" as the "detail".

The 2nd parameter ("detail") doesn't need to be a string, it can be an object too.

```
$dispatch('event-name', { count: 1 })
```

Receiving events using `x-on`.

```
<div
  x-on:peer-message.window="msg = $event.detail"
  x-data="{ msg: '' }"
>
  <div x-text="msg"></div>
</div>

<button
  x-data
  @click="$dispatch('peer-message', 'from-peer')"
>
  Send peer message
</button>
```

https://codepen.io/hugodf/pen/NWpLPXj

WHEN TO USE `.window`?

When the element dispatching the event is not a child/descendant of the one that should receive it.

EXAMPLE OF WHEN `.window` IS NOT NECESSARY

```
<div
  x-on:child-message="msg = $event.detail"
  x-data="{ msg: '' }"
>
  <div x-text="msg"></div>
  <button
    x-data
    @click="$dispatch('child-message', 'from-child')"
  >
    Send message to parent
  </button>
</div>
```

https://codepen.io/hugodf/pen/NWpLPXj

The `button` (element that dispatches "child-message") is a descendant/child of the `div` with `x-on:child-message`.

The name for this is "event bubbling"

Bubbling: browser goes from the element on which an event was triggered up its ancestors in the DOM triggering the relevant event handler(s).

If the element with the listener (`x-on`) is not an ancestor of the dispatching element, the event won't bubble up to it.

JAVASCRIPT -> ALPINE EVENTS

How is `$dispatch` implemented? (See the source)

```
el.dispatchEvent(new CustomEvent(event, {
  detail,
  bubbles: true,
}))
```

We can do the same in our own JavaScript

```
<button id="trigger-event">Trigger from JS</button>
<script>
  const btnTrigger = document.querySelector('#trigger-event')
  btnTrigger.addEventListener('click', () => {
    btnTrigger.dispatchEvent(
      new CustomEvent('peer-message', {
        detail: 'from-js-peer',
        bubbles: true
      })
    )
  })
</script>
```

https://codepen.io/hugodf/pen/NWpLPXj

ALPINE -> JAVASCRIPT EVENTS

We can use `document.addEventListener` and read from the `event.detail` (same as when using `x-on`).

```
<div id="listen">Listen target from JS</div>
<script>
  const listenTarget = document.querySelector('#listen')
  document.addEventListener('peer-message', (event) => {
    listenTarget.innerText = event.detail
  })
</script>
```

https://codepen.io/hugodf/pen/NWpLPXj

EVENT NAME X-ON QUIRKS

HTML is case-insensitive and `x-on:event-name` is a HTML attribute.

To avoid surprises, use dash-cased or namespaced event names, eg. `hello-world` or `hello:world` instead of `helloWorld`.

4. `x-show` VS `x-if`

What can you do with `x-show` that you can't with `x-if` and vice versa?

CONCEPT: SHORT-CIRCUIT/GUARD

Example:

```
function short(maybeGoodData) {
  if (!maybeGoodData) return [];
  return maybeGoodData.map(...)
}
```

If `maybeGoodData` is `null` we won't get a "Cannot read property 'map' of null" because the `.map` branch doesn't get run.

The "if" is sometimes called a guard or guard clause & the whole pattern is sometimes called "short-circuiting return", it avoids running code that could cause errors.

`x-if` doesn't evaluate anything inside the template if `x-if` evaluates to `false`.

Same as the guard in our `short` function it helps us skip evaluations that could be dangerous.

`x-show` keeps evaluating everything (x-show only toggles `display: none`).

```
<div x-data="{ obj: null }">
  <template x-if="obj">
    <div x-text="obj.value"></div>
  </template>
  <span x-text="'not crashin'">crashed</span>
</div>

<div x-data="{ obj: null }">
  <div x-show="obj">
    <div x-text="obj.value"></div>

  </div>
  <span x-text="'not crashin'">crashed</span>
</div>
```

https://codepen.io/hugodf/pen/vYxzOze

`x-if` doesn't crash

`x-show` does due to `obj.value`

PERFORMANCE

Depending on what you're doing you might find that `x-show` or `x-if` yields better performance.

`x-show` toggles a single `style` property.

`x-if` inserts/removes DOM Nodes.

If you're having performance issues with one, try the other.

USEFUL LINKS

- "slides": codewithhugo.com/alpine-tips
- all the demos: codepen.io/collection/ZMMNgj
- Async JS: codewithhugo.com/async-js