Alpine.js

Reactive behavior written right in your HTML.

01 Install & start

```
<script defer src="https://cdn.jsdelivr.net/
  npm/alpinejs@3.x.x/dist/cdn.min.js">
</script>
```

```
// npm install alpinejs
import Alpine from 'alpinejs'
Alpine.start()
```

Keep the src URL on one line. defer is required.

02 x-data: component state

```
<div x-data="{ open: false, count: 0 }">
  <button @click="open = !open">Nav</button>
  <button @click="count++">Add</button>
  <p x-show="open" x-text="count"></p>
</div>
```

x-data starts a component; children read and change its data, and the DOM follows.

03 Component functions

```
<div x-data="contactForm()">
  <button @click="submit()">Send</button>
</div>
```

```
function contactForm() {
  return {
    email: '',
    submit() { alert(this.email) },
  }
}
```

A function keeps bigger logic reusable. Methods use this.

04 x-text & x-html

```
<div x-data="{ name: 'John', count: 5,
  post: '<h2>Title</h2><p>Text</p>' }">
  <p x-text="name + ' Doe'"></p>
  <p x-text="`Count: ${count}`"></p>
  <p x-text="name.toUpperCase()"></p>
  <div x-html="post"></div>
</div>
```

x-text takes any expression and sets text content, so HTML shows as plain text. x-html sets innerHTML: trusted content only, never user input (XSS).

05 x-show: toggle visibility

```
<div x-data="{ open: false, role: 'dev' }">
  <button @click="open = !open">Nav</button>
  <nav x-show="open" x-transition>...</nav>
  <p x-show="role === 'dev'">Dev tools</p>
</div>
```

x-show toggles CSS display, so the element stays in the DOM. Use it for content that toggles often.

06 x-on: events

```
<div x-data="{ msg: '', hover: false }">
  <button x-on:click="msg = ''">X</button>
  <button @click="msg = ''">X</button>
  <input @input="msg = $event.target.value">
  <div @mouseenter="hover = true"
       @mouseleave="hover = false"></div>
  <form @submit.prevent="save">...</form>
</div>
```

@ is shorthand for x-on:. $event is the native event. A method name with no () is called with the event.

07 Event & key modifiers

.prevent / .stop	Cancel default / bubbling
.once	Runs the first time only
.outside	Clicks outside the element
.debounce	Waits until events pause
.throttle	At most once per interval
.enter / .escape	Key filters for key events

Example: @click.outside="open = false".

08 x-model: form inputs

```
<div x-data="{ name: '', agree: false,
  color: 'blue', tags: [] }">
  <input type="text" x-model="name">
  <input type="checkbox" x-model="agree">
  <select x-model="tags" multiple>
    <option value="js">JavaScript</option>
  </select>
</div>
```

A checkbox binds a boolean, a multiple select an array. Modifiers: .lazy (on change), .number, .debounce.

09 x-bind: attributes

```
<div x-data="{ id: 7, busy: false }">
  <a x-bind:href="'/users/' + id">User</a>
  <a :href="`/users/${id}`">User</a>
  <img :src="`/avatars/${id}.jpg`">
  <button :disabled="busy">Save</button>
</div>
```

: is shorthand for x-bind:. Any attribute can be bound, and the value is a JavaScript expression.

Alpine.js

Conditionals, lists, transitions and magics.

10 Class & style binding

```
<div x-data="{ on: true, color: 'red' }">
  <p :class="{ 'active': on,
    'text-red': color === 'red' }">Hi</p>
  <p :style="`color: ${color}`">Hi</p>
  <button x-bind="{ class: 'btn',
    'data-id': 7 }">Save</button>
</div>
```

Object syntax adds each class whose value is true. x-bind with no attribute spreads an object of attributes.

11 x-if: add & remove

```
<div x-data="{ loggedIn: false }">
  <template x-if="loggedIn">
    <div>Welcome back!</div>
  </template>
</div>
```

Needs a <template> with one root element. x-if adds and removes the element: use it for heavy, rarely shown parts and x-show for frequent toggles.

12 x-for: lists & keys

```
<ul x-data="{ users: [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' } ] }">
  <template x-for="user in users"
            :key="user.id">
    <li x-text="user.name"></li>
  </template>
</ul>
```

Also a <template> with one root element. Give each item a unique :key. The list updates when the array changes.

13 Index, ranges & nesting

```
<template x-for="(fruit, i) in fruits">
  <li x-text="`${i + 1}. ${fruit}`"></li>
</template>
<template x-for="i in 5">
  <span x-text="i"></span>
</template>
```

The index starts at 0, while i in 5 counts 1 to 5. Loops can nest inside the root element.

14 x-ref & $refs

```
<div x-data>
  <input x-ref="email" type="email">
  <button @click="$refs.email.focus()">
    Focus
  </button>
</div>
```

x-ref names an element and $refs reaches it, no querySelector needed. Methods use this.$refs.

15 x-transition

`x-transition`	Fade and scale, the default
`.opacity / .scale`	Fade only / scale only
`.duration.500ms`	Set the duration

```
<div x-show="open"
  x-transition:enter="transition"
  x-transition:enter-start="opacity-0"
  x-transition:enter-end="opacity-100">
```

Add :leave, :leave-start and :leave-end the same way. Works with x-show, not x-if.

16 Magic properties

`$el`	The current element
`$root`	The component root element
`$refs`	Elements named with x-ref
`$event`	The event, inside x-on
`$store`	Global shared state

```
<button @click="$el.remove()">X</button>
```

$el is the element the expression is on, not the root.

17 $watch, $dispatch & $nextTick

```
<div x-data="{ count: 0 }"
  x-init="$watch('count', value =>
    console.log('count is', value))"
  @notify="alert($event.detail.message)">
  <button @click="$dispatch('notify',
    { message: 'Hello' })">Notify</button>
  <button @click="count++; $nextTick(() =>
    console.log('DOM updated'))">+1</button>
</div>
```

x-init runs when the component starts. $dispatch fires a bubbling event; listeners read $event.detail.

18 x-cloak

```
[x-cloak] { display: none !important; }
```

```
<div x-data="{ message: 'Hello' }" x-cloak>
  <p x-text="message"></p>
</div>
```

Hides markup until Alpine starts and removes x-cloak.