Alpine.js

The easy way to add interactivity to your Umbraco site

Søren Kottal

- Tech Lead at Ecreo
- Umbraco MVP since 2018
- Maintainer/author of Doc Type Grid Editor, Full Text Search, Matryoshka and others...
- Tinkerer by heart

Ecreo loves Umbraco

> Simple things should be simple, complex things should be possible

Alan Kay

Building CMS driven websites should be simple

- Routing
- Templating
- Extensibility

A brief history of interactivity on websites

- Flash
- DHTML
- The early frameworks

Then jQuery happened

```
$("#my-div").addInteractivity(); // yay
```

```
// handles how the fixed navigation reacts
var position = 0;
$(window).scroll(function (e) {
    var $element = $('.header');
    var scrollTop = $(this).scrollTop();

    if (scrollTop <= 1) {
        $element.removeClass('is-hidden').removeClass('is-scrolling').removeClass('is-scroll-up');
        if ($('.hero').length == 0) {
            $element.addClass('is-scrolling').addClass('is-scroll-up');
        }
    } else if (scrollTop < position) {
        $element.removeClass('is-hidden');
        $element.addClass('is-scroll-up');
    } else if (scrollTop > position) {
        $element.addClass('is-scrolling');
        $element.removeClass('is-scroll-up');
        if (scrollTop + $(window).height() >= $(document).height() - $element.height()) {
            $element.removeClass('is-hidden');
            $element.addClass('is-scroll-up');
        } else if (Math.abs($element.position().top) < $element.height()) {
            $element.addClass('is-hidden');
            $('.header__basket').removeClass('open');
        }
    }
    position = scrollTop;
});
```

```
$(document).ready(function () {

    if (location.hash != "") {
        scrollToElement($(location.hash));
    }

    $("[data-ga-pageview]").on("click", function () {
        if (ga) {
            ga("send", "pageview", $(this).attr("data-ga-pageview"));
            ga("sub.send", "pageview", $(this).attr("data-ga-pageview"));
        }
    });

    $(".checkout__form").each(function () {

        var checkout = $(this);

        checkout.find("[name=Type]").on("change", function () {
            var selected = checkout.find("[name=Type]:checked").val();

            if (selected === "Privat") {
                checkout.find("[name=Organization]").closest(".control-group").hide();
                checkout.find("[name=Address1]").attr("placeholder", "Din adresse");
            }
            else if (selected === "Skole") {
                checkout.find("[name=Organization]").closest(".control-group").show();
                checkout.find("[name=Address1]").attr("placeholder", "Skolens adresse");
            }

            if (checkout.find("[name=PaymentMethod][value=Faktura]").is(":checked") && selected === "Skole") {
                checkout.find("#ean-option-message").show();
            }
            else {
                checkout.find("#ean-option-message").hide();
            }

        });

        checkout.find("[name=PaymentMethod]").on("change", function () {


            if (checkout.find("[name=PaymentMethod][value=Faktura]").is(":checked") && checkout.find("[name=Type]:checked").val() === "Skole") {
                checkout.find("#ean-option-message").show();
            }
            else {
                checkout.find("#ean-option-message").hide();
            }
        });

    });
```

```
    // card__item__info
    $('.cardlist__item').on('click', '.cardlist__item__info__btn', function () {
        $(this).parent().find('.cardlist__item__info').toggleClass('open');
        $(this).find('.icon').toggleClass('closed');
        $(this).find('.close').toggleClass('open');
    });
    if ($('#type-person').prop('checked')) {
        $('label[for="type-person"]').addClass('active');
        $('#ean-option').hide();
    }

    if ($('#type-school').prop('checked')) {
        $('label[for="type-school"]').addClass('active');
        $('#ean-option').show();
        $('#ean-option .ean').show();
    }

    if ($('#payment-invoice').prop('checked')) {
        $('label[for="payment-invoice"]').addClass('active');
    }

    if ($('#payment-ean').prop('checked')) {
        $('label[for="payment-ean"]').addClass('active');
    }

    if ($('#AcceptTerms').prop('checked')) {
        $('label[for="AcceptTerms"]').addClass('active');
    }

    $('#type-school').change(function () {
        if (this.checked) {
            $('label[for="type-person"]').removeClass('active');
            $('label[for="type-school"]').addClass('active');
            $('#ean-option').show();
        }
    });
    $('#type-person').change(function () {
        if (this.checked) {
            $('label[for="type-person"]').addClass('active');
            $('label[for="type-school"]').removeClass('active');
            $('#ean-option').hide();
        }
    });

    if ($('#SameDelivery').prop('checked')) {
        $('label[for="SameDelivery"]').addClass('active');
        $('#delivery-address').show();
    }

    $('#SameDelivery').change(function () {
        if (this.checked != true) {
            $('label[for="SameDelivery"]').removeClass('active');
            $('#delivery-address').hide();
        }
        else {
            $('label[for="SameDelivery"]').addClass('active');
            $('#delivery-address').show();
        }
    });

    $('#payment-invoice').change(function () {
        if (this.checked) {
            $('label[for="payment-ean"]').removeClass('active');
            $('label[for="payment-invoice"]').addClass('active');
        }
    });
    $('#payment-ean').change(function () {
        if (this.checked) {
            $('label[for="payment-ean"]').addClass('active');
            $('label[for="payment-invoice"]').removeClass('active');
        }
    });

    $('#AcceptTerms').change(function () {
        if (this.checked) {
            $('label[for="AcceptTerms"]').addClass('active');
        }
        else {
            $('label[for="AcceptTerms"]').removeClass('active');
        }
    });

    // checkout ean
    $('input[name="PaymentMethod"]').on('change', function () {
        var radio = $(this);
        if (radio.val() == 'EAN') {
            $('#ean-option .ean').show();
        }
        else {
            $('#ean-option .ean').hide();
        }
    });


    //$.validator.unobtrusive.adapters.addBool("mustbetrue", "required");

});
```

And we moved on to MVC like frameworks

- AngularJS
- Vue
- React

NPM became a thing

- Grunt
- Gulp
- Webpack

"Did you npm install?"

CSS were no longer just CSS

- Less
- Sass (SCSS)
- PostCSS

Difficulties with teams and "build-and-forget"-type projects

- Coding styles
- Approaches
- Documentation

"We are really busy, can you add these features, to this stale site, built like no other sites in our portfolio?"

The revelation

- Best practices you read about is mostly someone elses practises
- CSS Utility Classes and "Separation of Concerns"

Utility first CSS

- TailwindCSS
- Configure and consume
- Well documented coding style and approach
- Optimized build

Back to Javascript

- We had already ditched jQuery
- Vue and friends felt too big for simple things

Problems with the big boys

- Vue single file components is great
- But you have to build them to use them, and have boilerplate code for attaching Vue to your DOM
- And they add your markup and styles to your JS bundle

Core Web Vitals

- Speed
- Interactivity
- Visual stability

Core Web Vitals

- Largest Contentful Paint (LCP)
- First Input Delay (FID)
- Cumulative Layout Shift (CLS)

Javascript is hurting your Core Web Vitals

- Every kB of Javascript must add value
- Not just "what can each framework do"
- But "what can each framework do, with the smallest payload"

The weigh in

Framework	Version	Filesize
Alpine.js	`3.5.0`	13.8 kB
Vue.js	`2.6.14`	35.4 kB
React + React DOM	`16.7.0`	38.4 kB
AngularJS	`1.8.2`	55.0 kB
jQuery	`3.6.0`	86.4 kB

Modern frameworks are nice

- Declarative syntax reduces complexity
- `v-if`, `v-text`, `v-bind`, `v-on` etc.
- Less DOM traversing

What is Alpine.js anyway?

> “Alpine.js offers you the reactive and declarative nature of big frameworks like Vue or React at a much lower cost. You get to keep your DOM, and sprinkle in behavior as you see fit.”

Introducing Alpine.js @ Smashing Magazine

Alpine.js works like other frameworks

- `x-if`, `x-text`, `x-bind`, `x-on` etc.
- Works by adding a script tag
- Sprinkle your interactivity where you need it
- Refactor to components when too complex

Alpine.js works like other frameworks

- `x-data` initializes your component
- All in your markup, no querying for a node to attach "the app"

TailwindCSS for Javascript

- Where TailwindCSS gives us tools to write less CSS, while still being able to make our own bespoke designs, Alpine.js gives us tools to write less Javascript, and makes it easier to sprinkle interactivity where needed.
- Less code - less bugs
- Same approach across projects and teams, documentation out of the box

Familiar syntax if you are used to popular frameworks

`v-if` `v-bind` `v-on` `v-text` `v-html` `v-model` `v-show`
`v-transition` `v-for` `v-ref` `v-cloak`

Familiar syntax if you are used to popular frameworks

`x-if` `x-bind` `x-on` `x-text` `x-html` `x-model` `x-show`
`x-transition` `x-for` `x-ref` `x-cloak`

And then some

`x-init` `x-effect` `x-ignore` `x-intersect` 📦 `x-trap` 📦 `x-collapse` 📦

📦 = from plugins

Magic properties

`$el` `$refs` `$store` `$watch` `$dispatch` `$nextTick` `$root` `$persist` 📦

📦 = from plugins

Where and when to use

- CMS driven sites (Umbraco, Wordpress, Drupal etc.)
- Simple static site generators (11ty, Hugo etc.)
- When doing proof of concepts in Codepen

How to use

```
<script src="//unpkg.com/alpinejs" defer></script>
```

Simple stuff should be simple

```
<div x-data="{ open: false }">
  <button x-on:click="open = !open">👇</button>
</div>
```

Complex stuff should be possible

```
<div x-data="dropdown">
  <button x-on:click="toggle">👇</button>
</div>
```

```
document.addEventListener("alpine:init", () => {
  Alpine.data("dropdown", () => ({
    open: false,

    toggle() {
      this.open = !this.open;
    },
  }));
});
```

A typical vanilla JS component - markup

```
<div class="hamburger">
  <button class="hamburger__toggle">🍔</button>
  <div class="hamburger__content">🍞🍅🥩🧀🍀</div>
</div>
```

A typical vanilla JS component - css

```css
.hamburger__content {
  display: none;
}

.hamburger[aria-expanded] > .hamburger__content {
  display: block;
}
```

A typical vanilla JS component - script

```
const hamburger = {
  init: function () {
    let hamburgerElm = document.querySelector(".hamburger");
    let toggleElm = hamburgerElm?.querySelector(".hamburger__toggle");

    toggleElm?.addEventListener("click", () => {
      hamburgerElm.ariaExpanded = !hamburgerElm.ariaExpanded;
      if (hamburgerElm.ariaExpanded) {
        window.addEventListener(
          "keydown",
          (event) => {
            if (event.code === "Escape") {
              hamburgerElm.ariaExpanded = false;
            }
          },
          { once: true }
        );
      }
    });
  },
};

export default hamburger.init();
```

A typical vanilla JS component - index.js

```
import "./components/hamburger";
```

The same in Alpine.js

```
<div
  class="hamburger"
  x-data="{ open: false }"
  x-on:keydown.escape.window="open = false"
  x-bind:aria-expanded="open"
>
  <button class="hamburger__toggle" x-on:click="open = !open">🍔</button>

  <div class="hamburger__content" x-show="open">🍞🍅🥩🧀🍀</div>
</div>
```

Demo time

Updating values

Using values across components

Persisting values to localStorage

Plugin provided - adds just 369 bytes of javascript to your bundle.

Easy transitions

Combine `x-transition` with `x-show` and get transitions without breaking a sweat

Easy transitions

Triggering animation with `x-intersect`

`x-intersect` allows you to easily set up Intersection Observers on your elements.

- Lazy loading images
- Infinite scrolling
- Tracking how much content has been seen by the user
- Triggering animations
- 398 bytes of javascript

Triggering animation with `x-intersect`

```
<div
  x-data="{ isIn: false }"
  x-intersect:enter.half="isIn = true"
  x-bind:class="{'opacity-0 transform translate-x-1/2' : !isIn }"
  class="transition"
>
  ...
</div>
```

Triggering animation with `x-intersect`

Intersection Observer

```
let options = {
  root: document.querySelector('#scrollArea'),
  rootMargin: '0px',
  threshold: 1.0
}

let callback = (entries, observer) => {
  entries.forEach(entry => {
    // Each entry describes an intersection change for one observed
    // target element:
    //   entry.boundingClientRect
    //   entry.intersectionRatio
    //   entry.intersectionRect
    //   entry.isIntersecting
    //   entry.rootBounds
    //   entry.target
    //   entry.time
  });
};

let observer = new IntersectionObserver(callback, options);

let target = document.querySelector('#listItem');
observer.observe(target);
```

Easy keyboard handling

- Focus styles are crucial for keyboard users, but often redundant for pointer users
- AlpineJS makes it easy to detect keyboard users

Easy keyboard handling

```
<div
  x-data
  x-on:keydown.tab.window="$store.usingKeyboard = true"
  x-on:mousedown.window="$store.usingKeyboard = false"
  x-on:touchend.window="$store.usingKeyboard = false"
></div>

...

<a
  x-bind:class="{
    'focus:outline-none' : !$store.usingKeyboard,
    'focus:ring-2' : $store.usingKeyboard
   }"
  >Link</a
>
```

Easy keyboard handling

```
<div
  x-data
  x-on:keydown.tab.window="$store.usingKeyboard = true"
  x-on:mousedown.window="$store.usingKeyboard = false"
  x-on:touchend.window="$store.usingKeyboard = false"
></div>

...

<body
  x-bind:class="{
    'not-using-keyboard' : !$store.usingKeyboard,
    'using-keyboard' : $store.usingKeyboard
   }"
></body>
```

Preventing scroll when modal is open

```
<div x-data="
  {
    open: false,
    toggle() {
      this.open = !this.open
      if (this.open) {
        document.body.classList.add('overflow-y-hidden')
      }
      else {
        document.body.classList.remove('overflow-y-hidden')
      }
    }
  }">
  <button x-on:click="open != open">
</div>
```

Preventing scroll when modal is open

```
<body x-data x-class="{ 'overflow-y-hidden' : $store.disableBodyScroll }">

<div x-data="
  {
    open: false,
    toggle() {
      this.open = !this.open
      this.$store.disableBodyScroll = this.open
    }
  }">
  <button x-on:click="open != open">
</div>
```

Preventing scroll when modal is open

```
<body>

<div x-data="
  {
    open: false,
    toggle() {
      this.open = !this.open
    }
  }">
  <button x-on:click="open != open">
  <div x-show="open" x-trap.noscroll>
</div>
```

Initializing JS libraries

- Cleave.js is a library for formatting `<input>` content while typing

```
var cleave = new Cleave(".input-element", {
  numeral: true,
});
```

Cleave.js with Alpine

```
<input x-init="new Cleave($el, { numeral: true })" />
```

Umbraco and Alpine

- Razor views and declarative javascript directives

Easy filtering

```
<div x-data="{ filter: '' }">
  <input type="text" x-model="filter" />

  @foreach (var person in Model.Persons) {
  <div x-show="'@person.Name'.indexOf(filter) > -1">@person.Name</div>
  }
</div>
```

Easy filtering

Less easy filtering

```
<div x-data="{
      search() {
        fetch(this.$refs.searchForm.action, {
                  body: new FormData(this.$refs.searchForm),
                  method: this.$refs.searchForm.method
              }).then(response => {
                  let parser = new DOMParser();
                  let doc = parser.parseFromString(response.content, 'text/html')
                  let persons = doc.querySelector('[x-ref=persons]')
                  this.$refs.persons.innerHTML = persons.innerHTML
              });
      }
}">
  <form
    action="@Model.Url()"
    method="get"
    x-ref="searchForm"
    x-on:submit.prevent="search"
  >
    <input type="text" name="filter" x-on:keyup.debounce="$dispatch('submit')" />
  </form>

  <div x-ref="persons">
    @foreach (var person in Model.Persons.Where(x => x.Name.InvariantContains(Request["filter"]))) {
      <div>@person.Name</div>
    }
  </div>
</div>
```

Making a tabbed interface

Ingredients:

- Nested Content
- Tab element type containing a title and some content
- Razor view for rendering the tabbed interface

```
<div x-data="{ activeTab: '@Model.Tabs.FirstOrDefault().Key' }">
  <ul>
    @foreach (var tab in Model.Tabs) {
    <li><button x-on:click="activeTab = '@tab.Key'">@tab.Title</button></li>
    }
  </ul>

  @foreach (var tab in Model.Tabs) {
  <template x-if="activeTab == '@tab.Key'">
    <div>@tab.TabContent</div>
  </template>
  }
</div>
```

Making a tabbed interface

Ajax submitting Umbraco Forms

Swap

```
@using (Html.BeginUmbracoForm<Umbraco.Forms.Web.Controllers.UmbracoFormsController>("HandleForm"))
```

with

```
@{
    var htmlAttrs = new Dictionary<string, object>();
    htmlAttrs.Add("x-data", "asyncUmbracoForm"); // our Alpine component
    htmlAttrs.Add("x-on:submit", "submit"); // the submit method
}
@using (Html.BeginUmbracoForm<Umbraco.Forms.Web.Controllers.UmbracoFormsController>("HandleForm", new { }, htmlAttrs))
```

```
Alpine.data("asyncUmbracoFrom", () => ({
  submitting: false,
  submit(event) {
    this.submitting = $(this.$root).valid(); // jQuery validate 👷
    if (!this.submitting) {
      event.preventDefault();
    } else {
      event.preventDefault();
      fetch(this.$root.action, {
        body: new FormData(this.$root),
        method: this.$root.method,
      }).then((response) => {
        if (response.redirected) {
          window.location.href = response.url;
        } else if (response.headers.get("UmbracoFormsSubmitted")) {
          this.submitting = false; // TODO: Somehow tell the user, the form has submitted
        } else {
          alert("Oh no, it didn't work");
        }
      });
    }
  },
}));
```

Why is this nice?

- We let Forms do what Forms does best
- Handling fields, conditions, validation and serverside stuff
- We orchestrate behaviour using Alpine, piggybacking onto existing stuff.

What do I love the most of Alpine.js

- Transitions
- Event handling
- State management

What are the alternatives?

- The usual suspects - Vue, React, Angular, Svelte
- HTMX
- Petite Vue

Petite Vue vs Alpine

- If you love Vue and don't want to let it go
- No transitions
- No plugins (x-intersect, x-persist etc.)
- Half the size
- Great potential

Summing up

- Easy to get going
- Sprinkle in interactivity when you need it
- Stay out of your way when you don't
- Mature with a growing community

```
←script src="//unpkg.com/alpinejs" defer→
```

Questions?

Twitter @skttl

Catch me later 🍻