Animation Trigger Script: A Friendly Guide for Designers

In a Nutshell, What Does This Script Do?

Imagine adding interactive animations to your website without delving into complex JavaScript code. The **Animation Trigger Script** makes this possible. It's a tool that allows you to control how elements on your webpage change their appearance or behavior when users interact with them—like clicking a button, hovering over an image, or scrolling down the page.

State Applications

A "state" refers to a specific class applied to an element, configured using a simple data-attribute. By defining different states, you can control how an element looks or behaves at different times. For example, a button might have a default state, a “big” state, and a “small” state, each looking slightly different. The element can then cycle through these states based on various kinds of interactions, and then dynamic changes and animations can be applied using CSS.

User Story

Consider a designer named Alex who wants to create an engaging landing page. Alex wants images to fade in as users scroll, buttons to change color when clicked, and sections to expand when hovered over—all without writing any JavaScript code or relying on bulky page-builders or plugins. With the Animation Trigger Script, Alex can achieve all this by simply adding some classes and data attributes to the HTML elements.

Separation of Concerns

The beauty of this script lies in the clear separation between how things work and how they look:

- **JavaScript Controls How It Works**: The script handles the logic behind triggering animations and changing states based on user interactions.
- **CSS Controls How It Looks**: You define the visual styles for each state in your CSS files, giving you full creative control over the design.

This separation allows designers to focus on the visual aspects while relying on the script to manage the interactive behavior.

Table of Contents

---

1. Getting Started

1.1 Script Installation

To begin using the Animation Trigger Script, you need to include it in your HTML file. Here's how you can do it:

1. **Download the Script**: Obtain the `animation-trigger-script.js` file from the source.
    - I have not yet figured out how to use Github, so for now you can download the script here:
    -

2. **Include the Script in Your HTML**: Place the following line before the closing `</body>` tag in your HTML document:

```
Unset
<script src="path/to/animation-trigger-script.js"></script>
```

   Remember to replace `"path/to/"` with the actual path to the script file in your project.

3. **Prepare Your HTML Elements**: Add the necessary classes and data attributes to the elements you want to animate (we'll cover this in detail later).

---

2. Required and Optional Classes

2.1 What Are the Required Classes and What Do They Do?

To make elements interactive using the Animation Trigger Script, certain classes need to be added to your HTML elements.

`.animation-trigger`

- **Required**: Yes
- **Purpose**: This class tells the script that this element is an animation trigger. The script will look for this class to apply the defined behaviors.

`.animation-trigger-parent`

- **Required**: No (Optional)

- **Purpose**: Use this class when you want to pass trigger configurations from a parent element to one or more child elements. This is helpful when you have multiple elements that should share the same behavior.

2.2 How They Work (Technical)

- **`.animation-trigger` Elements**: The script scans the document for elements with this class and reads their data attributes to determine how they should behave when triggered.
- **`.animation-trigger-parent` Elements**: When this class is used, the script looks for child elements specified by the `data-child-target` attribute and applies the parent's configurations to those child elements.

2.3 How and When to Use Them

- **Use `.animation-trigger`**: Whenever you want an element to change states based on user interaction (like clicks or hovers), you add this class to it.
- **Use `.animation-trigger-parent`**: When you have multiple child elements that should behave similarly, and you want to avoid repeating the same configurations for each one.

Can They Be Used Simultaneously?

Yes! An element can have both `.animation-trigger` and `.animation-trigger-parent` classes. This means the element itself is an animation trigger, and it also passes its configurations to specified child elements.

2.4 Usage Examples

Single Trigger Element:

```
Unset
<div class="animation-trigger"
      data-trigger-click="#myButton"
      data-states="state1,state2"
      >

  Clickable Element

</div>
```

Parent Trigger with Child Elements:

```
Unset
<div class="animation-trigger-parent"
       data-child-target=".child-element"
       [Additional Attributes]
       >

       <div class="child-element">Child Element 1</div>

</div>

  <div class="child-element">Child Element 2</div>
```

---

3. Data Attributes

3.1 How to Use Attributes

Data attributes are custom attributes added to HTML elements that start with `data-`. They store additional information that can be accessed by scripts. In the context of the Animation Trigger Script, data attributes define the behavior of your animation triggers.

3.2 States

What Are "States"?

States represent different appearances or conditions of an element and are dynamically applied as CSS classes to elements. By defining states with the `data-states` attribute and creating corresponding CSS rules, you can specify how an element should look or behave at different times.

How They Work

- **State Classes**: Each state is represented by a CSS class. These classes are added or removed from the element based on triggers.
- **State Transitions**: When a trigger occurs, the script changes the state by updating the class on the element.

How to Define and Use States

Use the `data-states` attribute to list all the states for an element, separated by commas.

```
Unset
<div class="animation-trigger"
      data-states="state1,state2,state3"
      >

  Content

</div>
```

`data-initial-state` Attribute

This optional attribute defines which state the element should be in when the page loads. If not specified, the element starts with the first state in the `data-states` list.

```
Unset

<div class="animation-trigger"
      data-states="state1,state2"
      data-initial-state="state1"
      >

  Content

</div>
```

Theoretical Usage Examples

- **Button States**: A button could have states like `default`, `hovered`, `clicked` to change its appearance on user interaction.
- **Section Visibility**: A section might have states `hidden`, `visible` to control its display.

3.3 Triggers

What Are "Triggers"?

Triggers are events or conditions that cause the state of an element to change. They define when and how an element should respond to user interactions or other actions.

How They Work

When a trigger event occurs (like a click or hover), the script updates the element's state according to the defined behavior.\

Available Triggers

Click Trigger

Explanation: Changes the element's state when a specified element is clicked.

Configuration Options:
Use `data-trigger-click` to specify the selector(s) of the element(s) that, when clicked, will trigger the state change.

Requirements:
The selector(s) in `data-trigger-click` must match one or more elements in the DOM. Trigger element can be a separate element or itself.

Example Separate Trigger:

```
Unset
<button id="myButton">Click Me</button>

<div class="animation-trigger"
       data-trigger-click="#myButton"
       data-states="state1,state2"
       >

  Content

</div>
```

Example Self Trigger:

```
Unset
<div class="animation-trigger" id="myButton"
       data-trigger-click="#myButton"
       data-states="state1,state2"
       >
```

```
  Content

</div>
```

Hover Trigger

Explanation:
Changes the element's state when the user hovers over a specified element.

Configuration Options:
Use `data-trigger-hover` to specify the selector(s) of the element(s) that, when hovered over, will trigger the state change.

Unique Attribute Options:

- `data-hover-event="enter"`: Trigger when the mouse enters the element.
- `data-hover-event="leave"`: Trigger when the mouse leaves the element.
- `data-hover-event="hold"`: Trigger immediately upon hovering and continue advancing states when used in conjunction with with a `data-hover-time` trigger.

Strict Requirements:
The selector(s) in `data-trigger-hover` must match one or more elements in the DOM. Trigger element can be a separate element or itself.

Example Separate Trigger:

```
Unset
<!-- Hover Area -->

<div class="hover-area">Hover Over Me</div>

<!-- Animated Element -->

<div class="animation-trigger"
       data-trigger-hover=".hover-area"
       data-hover-event="enter" data-states="state1,state2">

  Content

</div>
```

Example Self Trigger:

```
Unset
<!-- Hover Area -->

<div class="hover-area">Hover Over Me</div>

<!-- Animated Element -->

<div class="animation-trigger" data-trigger-hover=".hover-area"
data-hover-event="enter" data-states="state1,state2">

  Content

</div>
```

Time Triggers

Explanation:
Changes the element's state based on time intervals.

Configuration Options:

- **Duration (Plays Once)**: `data-trigger-time="delay:5s"` triggers once after 5 seconds.
- **Loop (Repeats)**: `data-trigger-time="loop:3s"` triggers every 3 seconds indefinitely.
- **Interval (Sequence Once)**: `data-trigger-time="interval:1s,2s,3s"` triggers at 1s, 2s, and 3s, then stops.
- **Loop Interval (Loops a Sequence)**: `data-trigger-time="loop interval:1s,2s"` loops through the intervals repeatedly.

Integration with Hover Events:

- **Hold Event with Time Trigger**: When using `data-hover-event="hold"`, the time trigger (`data-trigger-time`) becomes active only while hovering over the specified element.
- **Immediate State Change on Hover**: Using `data-hover-event="hold"` causes the state to change immediately upon hovering.

Strict Requirements:
Time values must be provided in seconds (`s`) or milliseconds (`ms`). (e.g., 5s, 5000ms, etc)

Basic Example:

```
Unset
<div class="animation-trigger"
      data-trigger-time="loop:3s"
      data-states="state1,state2">

  Content

</div>
```

Example with Hover:

```
Unset
<div class="animation-trigger hover-area"
       data-trigger-hover=".hover-area"
       data-hover-event="hold"
       data-trigger-time="loop:3s"
       data-states="state1,state2,state3"
       data-initial-state="state1"
       >

  Hover and Watch Me Change

</div>
```

Scroll-Based Triggers

Explanation:
Changes the element's state based on its position within the viewport as the user scrolls.

Defining Ranges:
Two options exist to define ranges. Points and Ranges, both share the same behavior and the only difference is in your preference for how ranges are defined.

- **Points**: Use `data-trigger-points="0.25,0.75"` to define a range between 25% and 75% of the viewport height. When ranges are defined with points, the first and last values will be used to open and close the first and last ranges. Points will be strung together to form a series of contiguous ranges.
- **Ranges**: Use `data-trigger-ranges="0-0.5,0.5-1"` to directly define ranges from 0% to 50% and 50% to 100% of the viewport height. Defining ranges directly allows for non-contiguous ranges.

Configuration Options:

- **State Behavior with One Range**: The element switches between two states before and after the range.
- **State Behavior with Multiple Ranges**: The element can switch between multiple states as it enters different ranges.

Scroll Animate Attribute:

- **Purpose**: When set to `true`, it updates a CSS variable `--scroll-progress` a decimal point between 0 and 1, that represents the element's progress through the range. This variable can be used to make on-the-fly animation calculations.
- **Usage**: `data-scroll-animate="true"`

```
Unset
#element{
    transform: translateX(calc(var(--scroll-progress) * -50%));
    transition: transform 50ms ease, background-color 2s ease, color 2s ease;
}
```

Advancement Behavior:

- **Aligned**: With scroll-based triggers, the default advancement behavior is `aligned`, meaning states align directly with the ranges or points.
- **Other Advancements:** Scroll based triggers also work with `advance`, `advance-reset` and `toggle-initial` advancement configurations. In this case entering any new range will act like a trigger.

Viewport Alignment (Optional):

- **Purpose**: `data-viewport-align` determines which part of the element is used to calculate its position within the viewport.
- **Options**: `top`, `middle` (default), `bottom`.

Examples:

```
Unset
<div class="animation-trigger"
     data-trigger-ranges="0-0.5,0.5-1"
     data-states="state1,state2"
     data-scroll-animate="true"
     >

  Scroll to Animate Me
```

```
</div>
```

Cascade Triggers

Explanation:
Changes the element's state when another element changes its state.

Configuration Options:

- Use `data-trigger-cascade` to specify the selector(s) of the element(s) that, when their state changes, will trigger the state change.
- Accepts complex CSS selectors — e.g., .child-test-3:nth-of-type(2)

Strict Requirements:

- The specified elements must themselves be animation triggers that change state.

Examples:

```
Unset
<!-- Primary Trigger Element -->

<button id="primaryButton" class="animation-trigger"
       data-states="state1,state2"
       data-trigger-click="#someElement"
       [Unique Configuration]
       >

  Trigger

</button>

<!-- Element That Cascades -->

<div class="animation-trigger"
       data-trigger-cascade="#primaryButton"
       data-states="stateA,stateB"
       [Unique Configuration]
       >

  Cascading Trigger
```

```
</div>
```

Child Target

Explanation:

- Works only with the `.animation-trigger-parent` class.
- Passes configurations from the parent to child elements.
- Accepts complex CSS selectors — e.g., .child-test-3:nth-of-type(2)

Configuration:
Use `data-child-target` to specify child elements that should inherit the parent's trigger configurations.

Strict Requirements:
Must be used on an element with the `.animation-trigger-parent` class.

Example:

```
Unset
<!-- Parent Element -->

<div class="animation-trigger-parent"
       data-child-target=".child-element"
       data-trigger-hover=".hover-area"
       [Configuration]
       >

</div>

<!-- Child Elements -->

<div class="child-element">Child 1</div>
<div class="child-element">Child 2</div>
```

Active Space

The Active Space attribute is useful when you want to define a specific range where a trigger is active, or want to disable a trigger when it is sufficiently off screen.

Explanation and Configuration

- Defines a range within which triggers are active, preventing elements from changing state when they outside the defined range.
- Use `data-active-space="0.1,0.9"` to set the active range from 10% to 90% of the viewport height.

Strict Requirements

- The first and last value will be used as the start and end of the range.
- If `data-active-space` is not defined, the default active space is `"-1, 2"`.
- Setting `data-active-space="full"` disables the feature, allowing triggers to work regardless of the element's position.

Examples:

Unset

```
<div class="animation-trigger"
       data-trigger-time="loop:3s"
       data-states="state1,state2"
       data-active-space="0,1"
       >

  Content

</div>
```

Debounce

The debounce attribute can be used to optimize a page with many triggering elements. By default it is set to 10ms.

Explanation and Configuration

- Prevents excessive triggering by adding a delay between trigger activations.
- Use `data-debounce="500"` to set a debounce delay of 500 milliseconds.

Strict Requirements

The value should be a number representing milliseconds.

Examples

```
Unset
<div class="animation-trigger"
       data-trigger-hover=".hover-area"
       data-debounce="200"
       data-states="state1,state2"
       >

  Content

</div>
```

3.4 Advancement Behaviors

What Are "Advancement Behaviors"?

Advancement behaviors determine how the element's state changes when triggered. They define the order and conditions under which states transition.

How Do Advancement Behaviors Work?

You specify the behavior using the `data-advancement` attribute. This tells the script how to move from one state to another when a trigger occurs.

Available Advancement Behaviors

---

1. Advance

Explanation:
Moves to the next state in the `data-states` list each time the trigger occurs. Loops back to the first state at the end.

Example:

```
Unset
<div class="animation-trigger"
       data-trigger-click="#myButton"
       data-advancement="advance"
       data-states="state1,state2,state3"
       >
```

```
  Content

</div>
```

Result:
State1 > State2 > State3 > State4

---

2. Advance-Reset

Explanation:
Advances through the states alternating between the initial state and the next state.

Usage Example:

```
Unset
<div class="animation-trigger"
      data-trigger-click="#myButton"
      data-advancement="advance-reset"
      data-states="state1,state2,state3,state4"
      >

  Content

</div>
```

Result:
State1 > State2 > State1 > State3 > State1 > State4

---

3. Toggle-Initial

Explanation:
Toggles between the initial state and the next state each time the trigger occurs.

Usage Example:

```
Unset
<div class="animation-trigger"
      data-trigger-hover=".hover-area"
      data-advancement="toggle-initial"
      data-states="state1,state2,state3,state4"
      data-initial-state="state3">

  Content

</div>
```

Result:
State3 > State4 > State3 > State4

---

4. Aligned

The aligned advancement method only works with scroll based triggers.

Explanation:
Aligns the states directly with defined ranges or points in scroll-based triggers.

Default Behavior with Scroll Triggers:
When using `data-trigger-points` or `data-trigger-ranges` without specifying `data-advancement`, the default is `aligned`.

Handling Mismatched Ranges and States:
- **More States than Ranges**: The extra states are ignored.
- **More Ranges than States**: The states loop to match the number of ranges.

Example:

```
Unset
<div class="animation-trigger"
      data-trigger-ranges="0-0.25,0.25-0.75,0.75-1"
      data-advancement="aligned"
      data-states="state1,state2,state3">

  Content

</div>
```

Result:
- **Before the first range:** state1
- **In the range 0-0.25:** state1

- **In the range 0.25-0.75:** state2
- **In the range 0.75-1:** state3
- **After the last range:** state3

---

4. Detailed Configuration Examples

Example 1: Simple Click Trigger

Goal: Change an element's state when a button is clicked or when another element is hovered over.

Configuration:

```
Unset
<div class="animation-trigger" id="clickButton"
       data-trigger-click="#clickButton"
       data-trigger-hover=".hover-area"
       data-states="red,blue"
       >

Content

</div>
```

CSS:

```
Unset
#clickButton .red {
       background-color:red;
}

#clickButton .blue {
       background-color:blue;
}
```

Explanation:

- Clicking on the #clickButton element switches the elements background color to blue. Clicking again switches the color back to red.

Example 2: Simple Time Triggers

Goal: Animate an element through the rainbow over 10 seconds.

Configuration:

```
Unset
<div class="animation-trigger"
       data-trigger-time="loop:2s"
       data-states="red,blue,yellow,green,purple"
       >

Content

</div>
```

CSS:

```
Unset
.red {
       background-color:red;
       transition: color 2s ease;
       }

.blue {
       background-color:blue;
       transition: color 2s ease;
}

.yellow {
       background-color:yellow;
       transition: color 2s ease;
}

.green {
       background-color:green;
       transition: color 2s ease;
}

.purple {
       background-color:purple;
       transition: color 2s ease;
}
```

Explanation:
The background of the element changes color every 2 seconds until it cycles back around after 10 seconds.

Example 3: Scroll Range and Cascade Trigger

Goal: Fade a button in from the left as the user scrolls, and increase its size when the user hovers over it.

Configuration:

```
Unset
<div class="animation-trigger" id="sliding-element"
     data-trigger-ranges="0-0.5"
     data-scroll-animate="true"
     data-states="left,right"
     data-initial-state="small"
     data-trigger-cascade="#expandButton"
     >

<button id="expandButton"
       data-trigger-hover="#expandButton"
       data-states="small,large"
       data-initial-state="small"
       >

       Hover

       </button>

</div>
```

CSS:

```
Unset
.left {
       transform:translateX(calc((1 - var(--scroll-progress)) * -50vw));
       opacity:var(--scroll-progress);
       transition: transform 10ms, opacity 10ms;
}

.right {
    opacity:1;
}

.large {
	scale:1.25;
	transition: scale 250ms ease;
}

.small {
	scale:1;
	transition: scale 250ms ease;

}
```

Explanation:

As the user scrolls, the `#sliding-element` will fade in move in from the left. When the user hovers over the button, it will gently scale to 1.25x it's size.

Example 4: Hover Trigger with Time-Based Advancement

Goal: Change the background color of an element every two seconds as long as the user is hovering over the element.

Configuration:

```
Unset
<div class="hover-area animation-trigger"
	data-trigger-hover=".hover-area"
	data-hover-event="hold"
	data-trigger-time="loop:2s"
	data-states="state1,state2,state3"
	data-initial-state="state1"
	>

Hover and Hold Over Me

</div>
```

CSS:

```
Unset
.state1 {
       background-color:red;
       transition: background-color 2s ease;
}

.state2 {
       background-color:blue;
       transition: background-color 2s ease;
}

.state3 {
       background-color:green;
       transition: background-color 2s ease;
}
```

Explanation:

The element will start with a red background color. When the user hovers over the element, it will change to blue, and then continue cycling through the colors as long as the user continues to hover over the element.

Example 5: Click Trigger with Delayed Cascades

Goal: Create a waving color change effect using a click trigger and delayed cascade triggers.

Configuration:

```
Unset
<div class="box animation-trigger box" id="click"
       data-trigger-click="#click"
       data-states="red,blue,green"
       data-initial-state="red"
       data-advancement="advance"
       >

       </div>

<div class="animation-trigger box" id="cascade1"
       data-trigger-cascade="#click"
       data-delay="500ms"
       data-states="red,blue,green"
       data-initial-state="red"
```

```
        data-advancement="advance"
        >

        </div>

<div class="animation-trigger box" id="cascade2"
        data-trigger-cascade="#cascade1"
        data-delay="500ms"
        data-states="red,blue,green"
        data-initial-state="red"
        data-advancement="advance"
        >

        </div>
```

CSS:

```
Unset
.box {
        width:100px;
        height:100px;
        }

.red {
        background-color:red;
        }

.blue {
        background-color:blue;
        }

.green {
        background-color:green;
        }
```

Explanation:

When the user clicks the `#click` element, the color will change from red to blue, which will trigger the same change in `#cascade1` 500ms later, followed by the same change in `#cascade2` 500ms after that.

Example 6: Use animation-trigger-parent To Configure a Different Element

Goal: Use a parent element to configure a different element.

Configuration:

Unset

```
<span class="animation-trigger-parent hide"
       data-child-target=".target"
       data-trigger-time="loop:2s"
       states="red,blue,green"
       data-advancement="advance-reset"
       >
       </span>


<span class="target"></span>
```

CSS:

Unset

```
.hide{
       visibility:none;
       }

.red {
       background-color:red;
       }

.blue {
       background-color:blue;
       }

.green {
       background-color:green;
       }
```

Explanation:

The configuration is applied to a hidden span element with the `animation-trigger-parent` class. This configuration is passed onto the `.target` element and its background will change from red to blue to green every 2 seconds.

5. Tips & Best Practices

Consistent Naming: Use clear and consistent names for IDs, classes, and states to keep your code readable.

Active Space Awareness: Remember that elements using triggers will not work if the triggering element is outside the active space. Adjust `data-active-space` accordingly, or set to “full” to disable the active space.

Initial State in HTML: For smoother transitions, you might want to hard-code the initial state class directly in the element's `class` attribute to prevent any initial animation.

Use States Wisely: Define only the states you need to keep your CSS and HTML clean. In complex set ups, consider using `animation-trigger-parent` to define your configurations using hidden spans all in one place, rather than in each element.

Viewport Alignment: Choose the appropriate `data-viewport-align` based on your animation needs. The default is `middle`, but you can use `top` or `bottom` as needed.

Debounce Usage: Use `data-debounce` to prevent performance issues caused by rapid triggering. Under basic circumstances this can generally be ignored.

Using `data-delay`: Introduce delays before executing trigger actions for more controlled animations.

Using `data-hover-event="hold"`: Pair this with `data-trigger-time` to advance states while hovering.

Scroll Animations: Enable `data-scroll-animate` to create animations that respond to scroll position.

---

6. Troubleshooting

No State Change Occurs:

- Ensure that the selectors in trigger attributes match existing elements.
- Verify that `data-states` includes the states you intend to use.
- Check if the element is properly aligned based on `data-viewport-align`.
- Confirm that `data-advancement` is set correctly, especially when using range-based triggers.
- Ensure the element is within the defined `data-active-space` range.

Unexpected Behavior:

- Check for typos in data attribute names and values.

- Ensure that the number of states matches the number of trigger points or ranges where applicable.
- Remember that `data-scroll-animate` affects scroll animations independently.

Hover 'Hold' Issues:

- Ensure that `data-trigger-time` is specified when using `data-hover-event="hold"`.
- Verify that the hover target matches the selector in `data-trigger-hover`.
- If the state doesn't change on hover, check that the `data-advancement` method is correctly set.

Trigger Delay Not Working:

- Ensure that the `data-delay` attribute is correctly formatted with appropriate time units (`s` for seconds, `ms` for milliseconds).
- Verify that the element has the appropriate trigger attributes alongside `data-delay`.

Performance Concerns:

- Use `data-active-space` to limit actions on elements based on position.
- Avoid unnecessary triggers on elements that are not visible.

"Undefined" class applied to range based triggers:

- There is a mismatch between the number of states and the number of ranges.

Range based animations appear jittery or delayed:

- Set the transition speed in your CSS to a very small number, such as 10ms.
- E.g., transition: `transition: transform 10ms`

---

7. Appendices

7.1 Quick Reference Summary Table

Attribute	**Description**	**Example**
`data-trigger-click`	Selector(s) for click triggers	`data-trigger-click="#myButton"`
`data-trigger-hover`	Selector(s) for hover triggers	`data-trigger-hover=".hover-area"`

Attribute	Description	Example
`data-trigger-time`	Time-based triggers	`data-trigger-time="loop:3s"`
`data-trigger-points`	Scroll trigger points	`data-trigger-points="0.25,0.75"`
`data-trigger-ranges`	Scroll trigger ranges	`data-trigger-ranges="0-0.5,0.5-1"`
`data-trigger-cascade`	Selector(s) for cascade triggers	`data-trigger-cascade="#triggerElement"`
`data-child-target`	Selector(s) of child elements to inherit parent configs	`data-child-target=".child-element"`
`data-active-space`	Defines active trigger range in viewport fractions	`data-active-space="0,1"`
`data-debounce`	Sets debounce delay in milliseconds	`data-debounce="500"`
`data-advancement`	Defines state advancement behavior	`data-advancement="advance"`
`data-states`	Comma-separated list of state classes	`data-states="state1,state2,state3"`
`data-initial-state`	Sets the initial state of the element	`data-initial-state="state1"`
`data-hover-event`	Specifies hover events (`enter`, `leave`, `hold`)	`data-hover-event="hold"`
`data-scroll-animate`	Enables scroll animation (`true` or omitted)	`data-scroll-animate="true"`
`data-delay`	Introduces delay before executing trigger action	`data-delay="1s"`
`data-viewport-align`	Determines element's position calculation reference	`data-viewport-align="bottom"`

Good luck and happy animating!