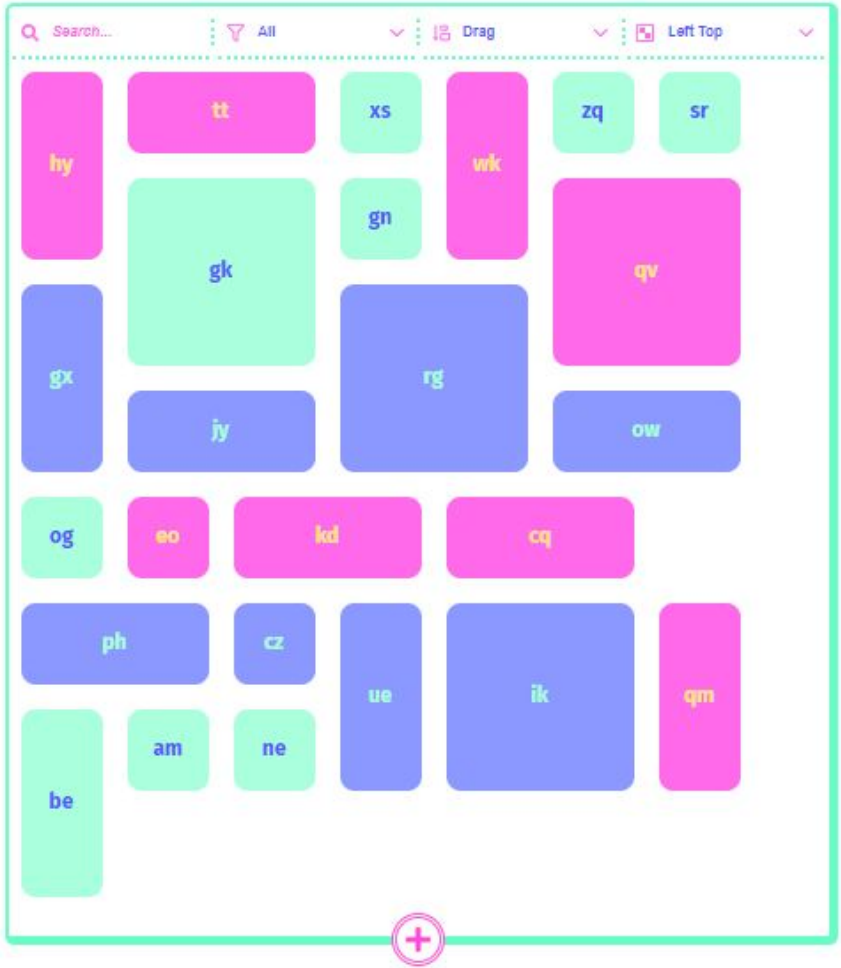
jQuery Plugin

By Farnaz Labbaf

Muuri is a powerful jQuery plugin that allows you to create dynamic and responsive grid layouts with ease. With Muuri, you can effortlessly arrange and organize grid items in a visually appealing manner.

The plugin provides a range of customization options, such as drag and drop support for item reordering, sorting based on custom data attributes, and filtering to display specific items.

Muuri also offers smooth layout animations, making transitions and simplifies the process of creating flexible and interactive grid layouts, enhancing the user experience.



Getting Started

1. Download the plugin using this page:

   https://docs.muuri.dev/getting-started.html

   Or simply link directly:

   <script src="https://cdn.jsdelivr.net/npm/muuri@0.9.5/dist/muuri.min.js"></script>

Get Web Animation Polyfill (if needed)

2. If a browser does not support Web Animations you need to use a polyfill.

   Install it from this link:

   https://docs.muuri.dev/getting-started.html#_2-get-web-animations-polyfill-if-needed

   Or download:

   Web-animations.min.js

   Or link directly:

   <script src="https://cdn.jsdelivr.net/npm/web-animations-js@2.3.2/web-animations.min.js"></script>

Add MarkUp

3. Each grid needs two containers (div)

   The outer div is used for positioning the item and the inner div is used for animating the item's visibility.

```
<div class="grid">
  <div class="item">
    <div class="item-content">
      <!-- Safe zone, enter your custom markup -->
      This can be anything.
      <!-- Safe zone ends -->
    </div>
  </div>

  <div class="item">
    <div class="item-content">
      <!-- Safe zone, enter your custom markup -->
      <div class="my-custom-content">Yippee!</div>
      <!-- Safe zone ends -->
    </div>
  </div>
</div>
```

Add Style

4. The grid element must be "positioned" property and set to *relative*, *absolute* or *fixed*.

   Muuri automatically resizes the grid element's width/height depending on the area the items cover and the layout algorithm configuration.

   Items must have their CSS position set to *absolute*.

   Items must not have any CSS transitions or animations applied to them, because they might conflict with Muuri's internal animation engine. However, the grid element can have transitions applied to it if you want it to animate when it's size changes after the layout operation.

   You can control the gaps between the items by giving some margin to the item elements.One last thing.

   Never ever set `overflow: auto;` or `overflow: scroll;` to the grid element. Muuri's calculation logic does not account for that and you *will* see some item jumps when dragging starts. Always use a wrapper element for the grid element where you set the `auto`/`scroll` overflow values.

```
.grid {
  position: relative;
}
.item {
  display: block;
  position: absolute;
  width: 100px;
  height: 100px;
  margin: 5px;
  z-index: 1;
  background: #000;
  color: #fff;
}
.item.muuri-item-dragging {
  z-index: 3;
}
.item.muuri-item-releasing {
  z-index: 2;
}
.item.muuri-item-hidden {
  z-index: 0;
}
.item-content {
  position: relative;
  width: 100%;
  height: 100%;
}
```

Final step

2. The last step is to define the grid element so Muuri can fetch the element for you.

```
var grid = new Muuri('.grid');
```

Youtube tutorial

You can find all needed coding here

Explore fun examples