JavaScript

Focus (Point) Vanilla JS

Zentrierung Bild im Rahmen leicht gemacht

Ein Bild im Rahmen zu zentrieren oder viel mehr so zu positionieren das der wichtigste Bestandteil im sichtbaren Bereich liegt, dafür braucht man nicht viel, ein bisschen Logik in JavasScript, ein DIV als Rahmen und ein Bild und schon läuft es.

Diese Implementierung bringt eine Filterung aller Bilder auf der Seite mit sich die im class-Attribut focus haben, für diese wird ein EventListener auf das load-Event eingerichtet, so dass auch ein lazy loading berücksichtigt ist. Ist das Bild geladen wird es initial fokussiert und ein EventListner am Window Resize sorgt dafür das bei Änderung auf edwaige Einflüsse auf die Rahmen-Abmessungen sofort reagiert wird.

Einzige Voraussetzung: `<img class="focus" src="..." />`, das src-Attribute kann natürlich auch durch ein data-src über meine Viewport (lazy loading) Variante befüllt werden, ohne data-focus-x und data-focus-y Attribute wird das Bild zentriert, bei Angabe von zumindest einem der beiden Werte wird eine Fokussierung durchgeführt. Der Ursprung ist die linke obere Ecke des Bildes und die Angaben gehen prozentual von 0 .. 100, also `<img class="focus" data-focus-x="75" data-focus-y="25" />` wäre dann rechts oben.
Kleines zusätzliches Feature: steht in dem img-Tag ein Attribute data-focus mit dem Wert contain `<img data-focus="contain" />` wird das Komplette Bild Zentriert dargestellt, der Wert cover `<img data-focus="cover" />` entspricht dem Standard muss allerdings nicht gesetzt werden.

focus.js JavaScript (5,47 kByte) 13.12.2021 01:40

```
// coding: utf-8
/*! Created by: Udo Schmal | https://www.gocher.me/ */
(function() {
  'use strict';
  // initialize focus object
  function Focus() {
    var self = this;
    // array for obsered elements image and wrapper
    var observables = [];
    // get item by target
    function getObservable(target) {
      for (let i=0; i < observables.length; i++) {
        if (observables[i].target == target) {
          return observables[i];
        }
      }
    }
    // options for MutationObserver
    var options = {
      attributes: true,
      attributeFilter: ['data-focus', 'data-focus-x', 'data-focus-y']
    };
    // ResizeObserver to observe image wrapper size changes
    var resizeObserver = new ResizeObserver(function (entries, observer) {
      for (let i=0; i < entries.length; i++) {
```

```
        self.handleEvent(getObservable(entries[i].target));
      }
    });
    // IntersectionObserver to observe if image is in viewport
    var intersectionObserver = new IntersectionObserver(function (entries,
observer) {
      for (let i=0; i < entries.length; i++) {
        let observable = getObservable(entries[i].target);
        if (entries[i].isIntersecting) {
          // images comes into viewport
          if (!observable.viewport) {
            // activate ResizeObserver
            resizeObserver.observe(entries[i].target);
            // activate MutationObserver
            observable.mutationObserver = new MutationObserver(function () {
              self.handleEvent(observable);
            });
            observable.mutationObserver.observe(observable.img, options);
            // set viewport flag
            observable.viewport = true;
          }
        } else {
          // images goes outoff viewport
          if (observable.viewport) {
            // deactivate ResizeObserver
            resizeObserver.unobserve(entries[i].target);
            // deactivate MutationObserver
            observable.mutationObserver.disconnect();
            // unset viewport flag
            observable.viewport = false;
          }
        }
      }
    }, {});
    this.observe = function (img) {
      console.log('use focus.js for ' + img.src);
      let item = { 'img': img, 'target': img.parentNode, 'viewport': false,
'mutationObserver': null };
        // find frame if it's not the parent
      while ((item.target.tagName.toLowerCase() !== 'body') &&
        ((item.target.offsetWidth === 0) || (item.target.offsetHeight ===
0))) {
        item.target = item.target.parentNode;
      }
      if (item.target.tagName.toLowerCase() !== 'body') {
        // frame must have position attribute for focused content
        let properties = window.getComputedStyle(item.target);
        if (!properties.getPropertyValue('position')) {
          item.target.style.position = 'relative';
        }
```

```
        // image must have a absolute position
        img.style.position = 'absolute';
        img.style.maxWidth = 'none';
        observables.push(item);
        intersectionObserver.observe(item.target);
      }
    };
    this.handleEvent = function (item) {
      // get dependencies from attributes
      let img = item.img, target = item.target,
        width = target.offsetWidth, // frame width
        height = target.offsetHeight, // frame height
        type = img.getAttribute('data-focus'), // full centered image else
cover
        focus_x = .50, // center
        focus_y = .50; // center
      if (type !== 'contain') {
        // overwrite focus x coortinate else center
        if (img.getAttribute('data-focus-x')) {
          focus_x = parseFloat(img.getAttribute('data-focus-x'), 10) / 100;
        }
        // overwrite focus y coordinate else center
        if (img.getAttribute('data-focus-y')) {
          focus_y = parseFloat(img.getAttribute('data-focus-y'), 10) / 100;
        }
      }
      // concept: depending on the aspect ratio and the type of filling of
the frame
      // one side adapts to the frame and the other side is cut off or has a
stub
      let scaled, value;
      if ((type !== 'contain')  === ((img.naturalWidth / img.naturalHeight)
>= (width / height))) {
        img.style.width = 'auto';
        img.style.height = '100%';
        img.style.top = '0';
        scaled = img.naturalWidth * (height / img.naturalHeight);
        value = - Math.floor((scaled * focus_x) - (width / 2));
        if (type !== 'contain')  {
          if (value > 0) {
            value = 0;
          } else if (value < width - scaled) {
            value = Math.floor(width - scaled);
          }
        }
        img.style.left = value + 'px';
      } else {
        img.style.width = '100%';
        img.style.height = 'auto';
        img.style.left = '0';
```

```
        scaled = img.naturalHeight * (width / img.naturalWidth);
        value = - Math.floor((scaled * focus_y) - (height / 2));
        if (type !== 'contain')  {
          if (value > 0) {
            value = 0;
          } else if (value < (height - scaled)) {
            value = Math.floor(height - scaled);
          }
        }
        img.style.top = value + 'px';
      }
    }
  }

  // get all images on page that class list contains "focus"
  var els = document.querySelectorAll('img[class~="focus"]');
  if (els.length > 0) {
    let focus = new Focus();
    // this don't breaks viewport (lazy loading)
    for (var i=0; i < els.length; i++) {
      if (!els[i].complete || (els[i].naturalWidth === 0)) {
        els[i].addEventListener("load", function (e) { focus.observe(this);
}, false);
      } else {
        focus.observe(els[i]);
      }
    }
  }

})();
```

width 400 px
height 200 px
type cover
focus-x 50 %
focus-y 50 %

Autor: Udo Schmal, veröffentlicht: 12.02.2020, letzte Änderung: 26.07.2024

© Copyright 2024 Udo Schmal