JavaScript

Lazy Loading Images und der Viewport - Vanilla JS

Lazy Loading nativ, per Observer oder Event gesteuert abhängig von den Fähigkeiten des Browsers den aktuellen Viewport zu ermitteln.

Inhalte nur zu laden, wenn sie wirklich benötigt werden, sorgt nicht nur dafür, dass die Seite schneller das Dom-Ready erreicht, sondern erspart unter Umständen jede Menge Datentransfer, was nicht nur Google erfreut, sondern auch den Betrachter der Website.

Diese Implementierung bringt eine Filterung aller Bilder auf der Seite mit sich, die im data-src-Attribut einen Inhalt entalten, dieser Wert braucht nur in das src-Attribut verschoben werden, sobald das Bild im sichtbaren Bereich auftaucht und auch wirklich sichtbar sein soll.

Einzige Voraussetzung `<img data-src="..." loading="lazy" />`

Diese Implementierung unterstützt die native Variante **loading="lazy"** als img Attribute bei einem aktuellen Browser (ab Opera 64, Firefox 75, Chrome 76, Edge 79, Andoid Browser 80, Chrome for Android 80), aber auch den **IntersectionObserver** bei den etwas äteren (ab Opera 45, Firefox 55, Chrome 58, Edge 16, Safari 12.2, Andoid Browser 80, Chrome for Android 80, ...), sowie eine Variante über **EventLisener** für die ganz alten Browser wie z.B. den Internet Explorer, dadurch entsteht auch die breiteste Abdeckung für Lazy Loading. Zusätzlich wurde das Laden der Bilder vor dem Ausdrucken mit eingebaut, welches bisher von keinem Browser als Standard zur Verfügung gestellt wird.

viewport.js JavaScript (7,23 kByte) 18.12.2020 14:28

```
// coding: utf-8
/** Created by: Udo Schmal | https://www.gocher.me/ */
(function() {
  'use strict';
  // lazy loading image / iframe
  function loadSrc(el) {
    if (el.getAttribute('data-src')) {
      el.setAttribute('src', el.getAttribute('data-src'));
      el.removeAttribute('data-src');
    }
    if (el.getAttribute('data-srcset')) {
      el.setAttribute('srcset', el.getAttribute('data-srcset'));
      el.removeAttribute('data-srcset');
    }
    if (el.getAttribute('data-sizes')) {
      el.setAttribute('sizes', el.getAttribute('data-sizes'));
      el.removeAttribute('data-sizes');
    }
    return true;
  }
  // lazy preloading video
  function preloadVideo(el) {
    if (el.getAttribute('data-autoplay')) {
      el.setAttribute('preload', 'auto');
      el.setAttribute('autoplay', 'autoplay');
```

```
      el.autoplay = true;
      el.play();
      el.removeAttribute('data-autoplay');
    }
  }
  // Viewport prototype to trigger event if element moves into viewport the
fist time
  function Viewport(els, callback) {
    this.els = els;
    this.callback = callback;
    this.active = false;
    var self = this;
    var hidden = [];
    function alreadyObserved(el) {
      for(var i=0; i<hidden.length; i++) {
        if (el === hidden[i]) {
          return true;
        }
      }
      return false;
    }
    for (var i=0; i < els.length; i++) {
      var el = els[i];
      if (!el.parentNode.style.position) {
        // parent must have position attribute for getBoundingClientRect
        el.parentNode.style.position = 'relative';
        while(el && el.tagName.toLowerCase() !== 'body') {
          if (window.getComputedStyle(el).display === "none") {
            if (!alreadyObserved(el)) {
              hidden.push(el);
              var observer = new MutationObserver(function(mutations) {
                if (mutations[0].target.style.display !== 'none') {
                  self.handleEvent();
                }
              });
              observer.observe(el, { attributes: true });
              break;
            }
          }
          el = el.parentNode;
        }
      }
    }
    this.handleEvent.bind(this);
    // is already in viewport after dom ready
    function ready(f){
      /complete|loaded/i.test(document.readyState) ? f() :
setTimeout(function(){ready(f);},9);
    }
    ready(function(){self.handleEvent();});
```

```
    // add event listener to scroll event to check visibility change
    document.addEventListener('scroll', this, true);
    window.addEventListener('resize', this, true);
    window.addEventListener('orientationchange', this, true);

  }

  Viewport.prototype = {
    // check if element is visible
    isVisible: function (el) {
      var style = window.getComputedStyle(el);
      return !!(el.offsetWidth || el.offsetHeight ||
el.getClientRects().length);
    },
    // check if element is in viewport
    isInViewport: function (el) {
      var bounding = el.getBoundingClientRect();
      return (
        bounding.bottom >= 0 &&
        bounding.right >= 0 &&
        bounding.top <= (window.innerHeight ||
document.documentElement.clientHeight) &&
        bounding.left <= (window.innerWidth ||
document.documentElement.clientWidth)
      );
    },
    // handle the visibility check and the resulting action
    handleEvent: function () {
      if (this.active === false) {
        this.active = true;
        for (var i=0; i < this.els.length; i++) {
          if (this.isInViewport(this.els[i]) && this.isVisible(this.els[i]))
{
            this.callback(this.els[i]);
          }
        }
        this.active = false;
      }
    }
  };

  const type = { event: 0, intersection: 1, loading: 2};
  var support;
  // if Google Chrome for Android "reduce data usage" / "lite mode" is
active
  // var dataSave = !('connection' in navigator) ||
(navigator.connection.saveData);
  // if Google Chrome chrome://flags/#enable-lazy-image-loading is enabled
  if ("loading" in HTMLImageElement.prototype) {
    console.log('native lazy loading');
```

```
    /* Opera 64, Firefox 75, Chrome 76, Edge 79, Andoid Browser 80, Chrome
for Android 80 */
    support = type.loading;
  } else if  ("IntersectionObserver" in window) {
    console.log('IntersectionObserver lazy loading');
    /* Opera 45, Firefox 55, Chrome 58, Edge 16, Safari 12.2, Andoid Browser
80, Chrome for Android 80, ... */
    support = type.intersection;
  } else {
    console.log('Event based lazy loading');
    /* Internet Explorer and other old browsers */
    support = type.event;
  }

  // activate viewport for image loading
  var images = document.querySelectorAll("img[data-src]");
  if (images.length > 0) {

    // load all images before print
    window.addEventListener("beforeprint", function(event) {
      for (var i=0; i < images.length; i++) {
        if (support == type.loading) {
          images[i].setAttribute('loading', 'eager');
        } else {
          loadSrc(images[i]);
        }
      }
      function sleep(milliseconds) {
      const date = Date.now();
        let currentDate = null;
        do {
          currentDate = Date.now();
        } while (currentDate - date < milliseconds);
      }
      sleep(1000);
    });

    // use native lazy loading
    if (support == type.loading) {
      // set attribute loading to lazy and move attribute value from data-
src to src
      for (var i=0; i < images.length; i++) {
        loadSrc(images[i]);
      }
    } else if (support == type.intersection) {
      // create intersection observer
      let lazyobserver = new IntersectionObserver(function(entries,
observer) {
        for (var i=0; i < entries.length; i++) {
          if (entries[i].isIntersecting) {
```

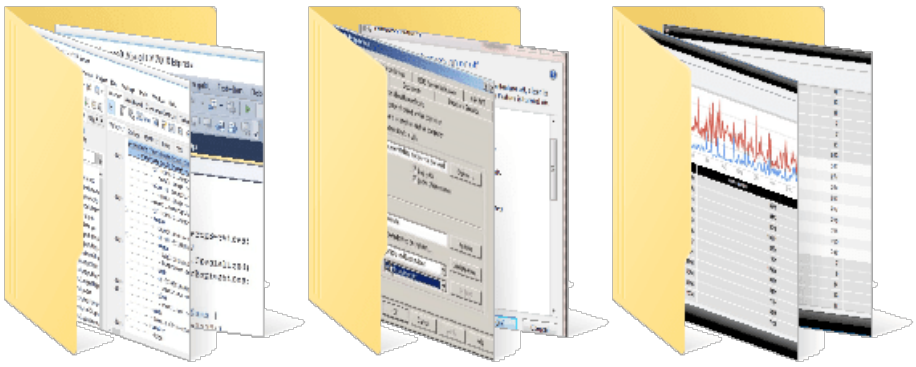
```
            loadSrc(entries[i].target);
            lazyobserver.unobserve(entries[i].target);
          }
        }
      },
      {
        threshold: [0.1],
        // Set a minimum delay between notifications
        delay: 100
      });
      // start observing
      for (var i=0; i < images.length; i++) {
        lazyobserver.observe(images[i]);
      }
    } else /* if (support == type.event) */ {
      // use eventListeners
      new Viewport(images, loadSrc);
    }
  }

  // activate viewport for iframe loading
  var iframes = document.querySelectorAll("iframe[data-src]");
  if (iframes.length > 0) {
    if (support == type.loading) {
      // set attribute loading to lazy and move attribute value from data-
src to src
      for (var i=0; i < iframes.length; i++) {
        loadSrc(iframes[i]);
      }
    } else if (support == type.intersection) {
      // create intersection observer
      let lazyiframeobserver = new IntersectionObserver(function(entries,
observer) {
        for (var i=0; i < entries.length; i++) {
          if (entries[i].isIntersecting) {
            loadSrc(entries[i].target);
            lazyiframeobserver.unobserve(entries[i].target);
          }
        }
      },
      {
        threshold: [0.1],
        // Set a minimum delay between notifications
        delay: 100
      });
      // start observing
      for (var i=0; i < iframes.length; i++) {
        lazyiframeobserver.observe(iframes[i]);
      }
    } else {
```

```
      new Viewport(iframes, loadSrc);
    }
  }

  // activate viewport for video preloading
  var videos = document.querySelectorAll("video[data-autoplay]");
  if (videos.length > 0) {
    new Viewport(videos, preloadVideo);
  }
})();
```

Images



Videos

IFrame (GoogleMaps) external

IFrame (GoogleMaps)

YouTube external

YouTube (autostart)

Vimeo external

Vimeo popup

X (Twitter)

Tweets by XDevelopers (https://twitter.com/XDevelopers?ref_src=twsrc%5Etfw)

Autor: Udo Schmal, veröffentlicht: 12.02.2020, letzte Änderung: 29.08.2024