Spotkanie wtorek 19:00

# Dlaczego nie można użyć funkcji Math.min/max dla obiektu/tablicy bez spread syntax?

  // function which is counting a number of times a specific tag occurs

  function calculateTagsParams(*tags*){

    const obj = *tags*;

    let arr = *Object*.values(obj);

    console.log(arr, 'qqqqqqqq');

    let min = Math.min(...arr);

    let max = Math.max(...arr);

    console.log( `Min value: ${min}, max value: ${max}` );

  }

# Które rozwiązanie najlepsze?

  function calculateTagsParams(*tags*){

    const obj = *tags*;

    let arr = *Object*.values(obj);

    console.log(arr, 'qqqqqqqq');

    let min = Math.min(...arr);

    let max = Math.max(...arr);

    console.log( `Min value: ${min}, max value: ${max}` );

    const params = {'minimum': min, 'maximum': max};

    return params;

    // const params = {max: 0, min: Infinity};

    // for(const tag in tags){

    //   // params.max = Math.max(tags[tag], params.max);

    //   // params.min = Math.min(tags[tag], params.min);

    //   params.max < tags[tag] ? params.max = tags[tag] : params.max;

    //   !params.min < tags[tag] ? params.min = tags[tag] : params.min;

    // }

    // return params;

  }

# Czy takie rozwiązanie też mogłoby być?

  function calculateTagClass(*count*, *params*){

    for(let i = 1; i < optCloudClassCount + 1; i++){

      if (*count* <= i/optCloudClassCount\**params*.maximum) return optCloudClassPrefix + i;

    }

}

Nie mogłoby, ale czy da się to jakoś sensownie przerobić?

  function calculateTagClass(*count*, *params*){

    console.log('calculateTagClass count', *count*);

    console.log('calculateTagClass params', *params*);

    // for(let i = 1; i < optCloudClassCount + 1; i++){

    //   if (count <= i/optCloudClassCount\*params.maximum) return optCloudClassPrefix + i;

    // }

    const normalizedCount = *count* - *params*.minimum;

    const normalizedMax = *params*.maximum - *params*.minimum;

    const percentage = normalizedCount / normalizedMax;

    const classNumber = Math.floor(percentage \* (optCloudClassCount - 1) + 1);

    return optCloudClassPrefix + classNumber;

  }

# Add default active class to the link

titleLink.innerHTML = html;

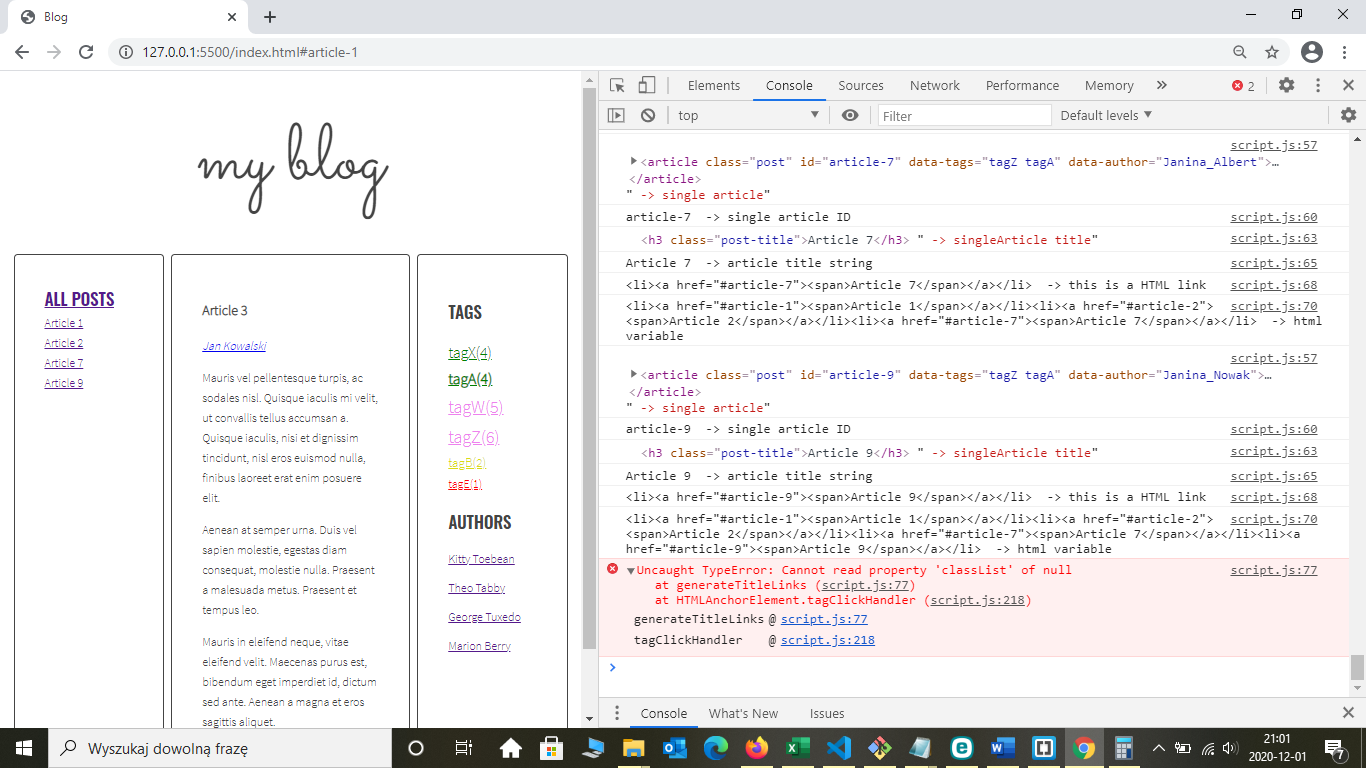
    // Add default active class to the link

    // const defArticleId = document.querySelector('article.active').getAttribute('id');

    // const defArticleLink = document.querySelector(`a[href\*=${defArticleId}]`);

    // defArticleLink.classList.add('active');

    // Articles' title links from a sidebar



Podczas używania filtrowania listy artykułów za pomocą tag cloud wyskakuje błąd, pojawiał się on w momencie w którym w wyniku filtrowania artykuł, który był aktualnie z klasą activ nie wyświetlał się w spisie linków.

# Dodatkowa opcja

* Powrót do widoku wszystkich linków po filtrowaniu.

  function allArticleLinksVisible() {

    console.log('function allArticleLinksVisible is working');

    const abc = document.getElementById('article-caller');

    abc.addEventListener('click', function(){

      generateTitleLinks('');

    });

  }

# Czemu template wstawiamy przed handlebars?

    <script *id*="template-article-link" *type*="text/x-handlebars-template">

      <li><a *href*="#{{ id }}"><span>{{ title }}</span></a></li>

    </script>

    <script *src*="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/4.1.0/handlebars.min.js"></script>

    <script *src*="js/script.js"></script>

# Problem z plikiem .eslintrc.json

[Info  - 12:08:42] .eslintrc.json:  Configuration for rule "globals" is invalid:    Severity should be one of the following: 0 = off, 1 = warn, 2 = error (you passed '{ Handlebars: false }').

# Jak działa poniższy przykład

function foo(cb, text) {

cb(text);

}

function bar(textOne, textTwo) {

console.log(textOne, textTwo);

}

foo(function(txt) { bar(txt, 'World') }, 'Hello');

To znacznie trudniejszy przykład, spróbuj jednak go rozwikłać.

Kod jest poprawny. Konsola wyświetli napis Hello World. Napis Hello pochodzi z argumentu text, który bar dostaje po wywołaniu cb (czyli funkcji function(txt) { bar(txt, 'World') }). World jest za to dostarczany do bar bezpośrednio.

# Działanie funkcji

Handlebars.registerHelper('ifEquals', function(*arg1*, *arg2*, *options*) {

  return (*arg1* == *arg2*) ? *options*.fn(*this*) : *options*.inverse(*this*);

})

# Kolejność wyświetlania własności obiektu za pomocą conslole.log