Take key from object with for (let key in object)!!!!!!!

let object = {name: 'Ivan', age: 13, city: 'Varna'}

for (let key in object) {

    console.log(key)

}

Съкратен синаксис на if statement

function solve(num1, num2, num3) {

    let sum = (num1 + num2) / num3

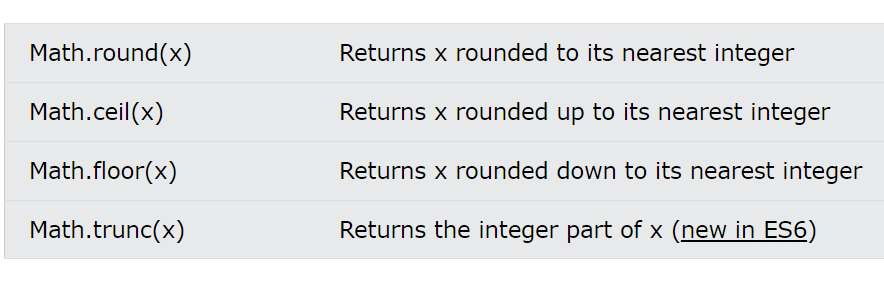
res = ''

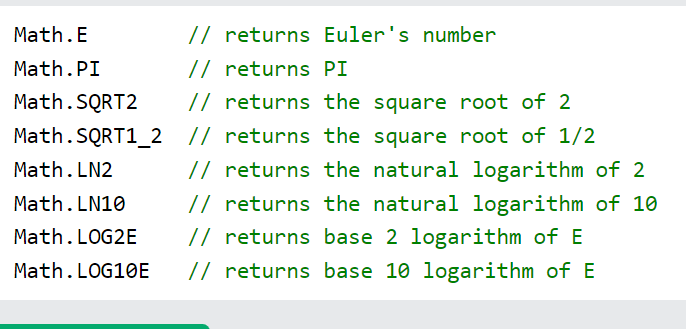
    sum % 2 === 0 ? sum += ' - Integer': sum += ' - Float'

    console.log(res) ако това е вярно върни това ако не върни това

}

Math Library





Array elements

let numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

let firstElement = numbers[0]

let lastElement = numbers[numbers.length - 1]

// destructuring of elements

let nums = [1, 2, 3]

let [first, second , third] = nums

console.log(`first = ${first} , second = ${second} , third = ${third}`)

output:

first = 1 , second = 2 , third = 3

spred operator . . . args

let nestedArrey = [1, 2, 3, ... [4, 5, 6]]

output:

[1, 2, 3, 4, 5, 6]

console.table(nestedArrey)

┌─────────┬────────┐

│ (index) │ Values │

├─────────┼────────┤ printing like table

│    0    │   1    │

│    1    │   2    │

│    2    │   3    │

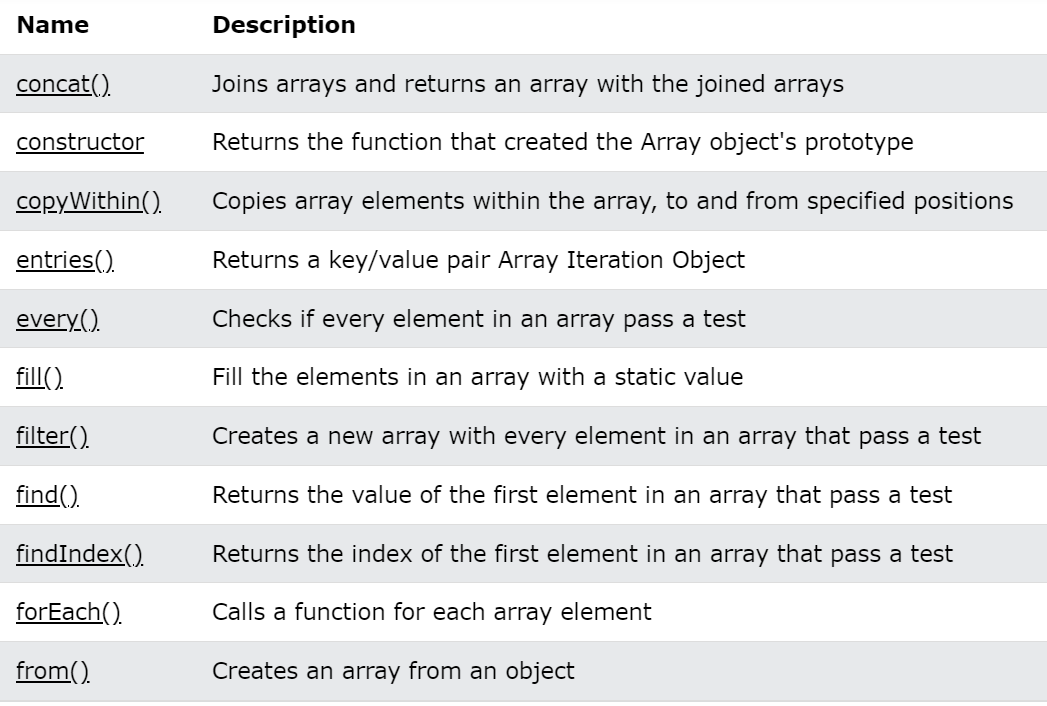
│    3    │   4    │

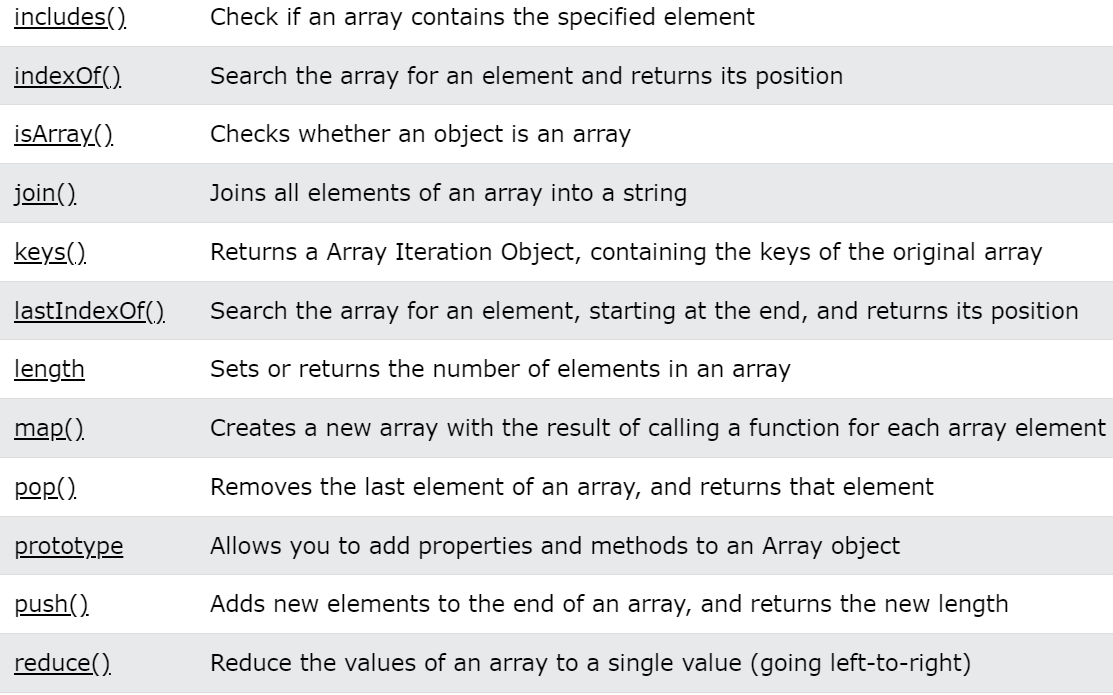
│    4    │   5    │

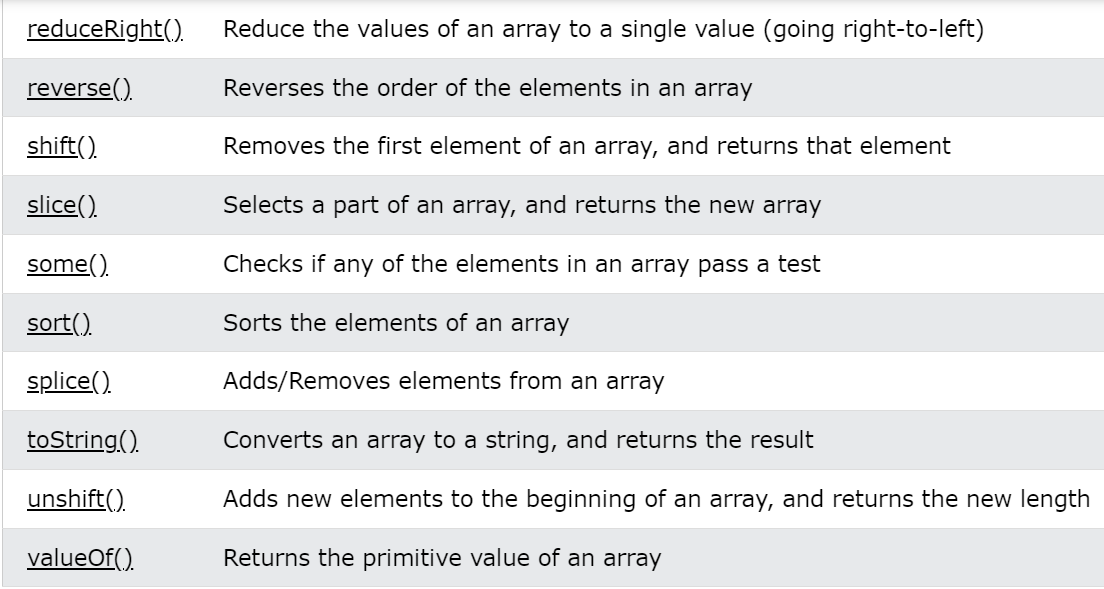
│    5    │   6    │

└─────────┴────────┘

Array methods







Sum all elements in array, check all element in array is the same

function solve(num) {

    let array1 = num.toString().split('')

    const isTrue = array1.every(x => x === array1[0]) this is like all() in Python

    // const initialValue = 0;

    let sum  = array1.reduce((x,y) => Number(x) + Number(y) //, initialValue 0)

    console.log(isTrue) this is like sum() in Python

    console.log(sum)

}

solve(2222222)

solve(1234)

count element in array

let count = newArray.filter(x => x === searchNum).length

console.log(`Number ${searchNum} occurs ${count} times.`)

Array sort methods

let number = [31, 2 , 432, 32, 5 , -1]

let names = ['Kiro', 'Pesho', 'Ivan', 'Sasho']

// ascending sort number

let sortedAscNum = [...number].sort((a, b) => a - b)

console.log(sortedAscNum)

// descending sort number

let sortedDescNum = [...number].sort((a, b) => b - a)

console.log(sortedDescNum)

// ascending sort string

let sortedAscStr = [...names].sort((a, b) => a.localeCompare(b))

console.log(sortedAscStr)

// descending sort string

let sortedDescStr = [...names].sort((a, b) => b.localeCompare(a))

console.log(sortedDescStr)

output :

[-1, 2, 5, 31, 32, 432]

[432, 32, 31, 5, 2, -1]

['ivan', 'Kiro', 'Pesho', 'Sasho']

['Sasho', 'Pesho', 'Kiro', 'ivan']

Sort by two criteria

function solve(arr) {

   let sorted = arr.sort((a, b) => {

    if (a.length !== b.length) {

        return a.length - b.length сортираме по дължина и после по азбучен ред

    }

    return a.localeCompare(b)

   })

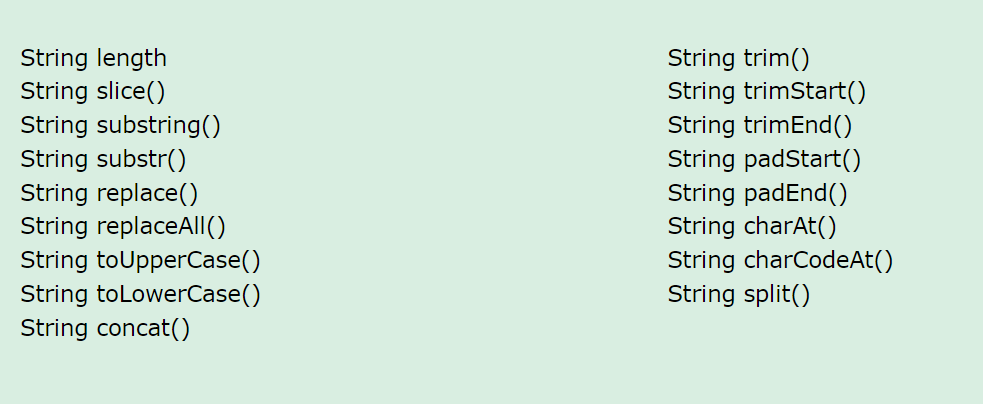
   console.log(sorted)

}

solve(['alpha', 'beta', 'gamma'])

output: ['beta', 'alpha', 'gamma']

string methods



Char in ascii table

function solve(char1, char2) {

   let start = char1.charCodeAt(0) от символ към цифра

   let end = char2.charCodeAt(0)

   let character = []

   for (let index = start + 1; index < end; index++) {

        character.push(String.fromCharCode(index)) от цифра към символ

   }

   return character.join(' ')

}

console.log(solve('#',':'))

if element is not in arrey the index is -1

let array = [1, 2, 3, 4]

let num = 5

let result = array.indexOf(num)

console.log(result)

output = -1

make unique array

function solve(arr) {

    let uniqueArr = arr.filter(function (element, index, self) {

        return self.indexOf(element) === index;

    });

    console.log(uniqueArr.join());

}

solve([7, 8, 9, 7, 2, 3, 4, 1, 2])

output : 7 8 9 2 3 4 1

short syntax for object whit same key , value

function solve(firstName, lastName, age) {

    // person = {}

    // person['firstName'] = firstName

    // person['lastName'] = lastName

    // person['age'] = age

    person = { person = {

       firstName, same firstName: firstName,

       lastName, lastName: lastName,

       age age: age

    }

    return person

}

Output : {firstName: 'Peter', lastName: 'Pan', age: '20'}

Copy Object and Array

let person = {

    name: 'Iva',

    age: 15,

    isStudent: true

}

console.log(`${person['name']} is ${person.age} age old and is student ${person['isStudent']}`)

console.log('-------------------------------------------')

let copyPerson = Object.assign({}, person) copy Object

console.log(person)

console.log(copyPerson)

console.log('-------------------------------------------')

let array = [1, 2, 3, 4]

let copyArray = array.slice() copy Array

console.log(array)

console.log(copyArray)

output:

Iva is 15 age old and is student true

-------------------------------------------

{name: 'Iva', age: 15, isStudent: true}

{name: 'Iva', age: 15, isStudent: true}

-------------------------------------------

(4) [1, 2, 3, 4]

(4) [1, 2, 3, 4]

Check is the key exist in Object

let person = {

    firstName: 'iva',

    lastName: 'Ivanova', използваме тази функция

    age: 30, hasOwnProperty

}

console.log(person.hasOwnProperty('firstName'))

Math.min and Math.max

function solve(firstNum, secondNum, thirdNum) {

    let min = Math.min(firstNum, secondNum, thirdNum)

    return min

}

console.log(solve(2,5,3))

output: 2

function solve(firstNum, secondNum, thirdNum) {

    let max = Math.max(firstNum, secondNum, thirdNum)

    return max

}

console.log(solve(2,5,3))

output: 5

Arrow function more one row

let solve = (firstNum, secondNum, thirdNum) => { curly brackets

    let sum = (a, b) => a + b;

    let subtract = (mySum, c) => mySum - c;

    return subtract(sum(firstNum, secondNum), thirdNum);

}

console.log(solve(2,5,3))

Make matrix with new Array and fill() function

function solve(n) {

    let result = new Array(n).fill(new Array(n).fill(n).join(' '))

    console.log(result.join('\n'))

} output: 4 4 4 4

4 4 4 4

solve(4) 4 4 4 4

4 4 4 4

Sorted object by key or by value

  Object

    .keys(tracker) this is sort by key:

    .sort((a, b) => a - b)

    .forEach(key => console.log(`${key} - ${tracker[key]}`))

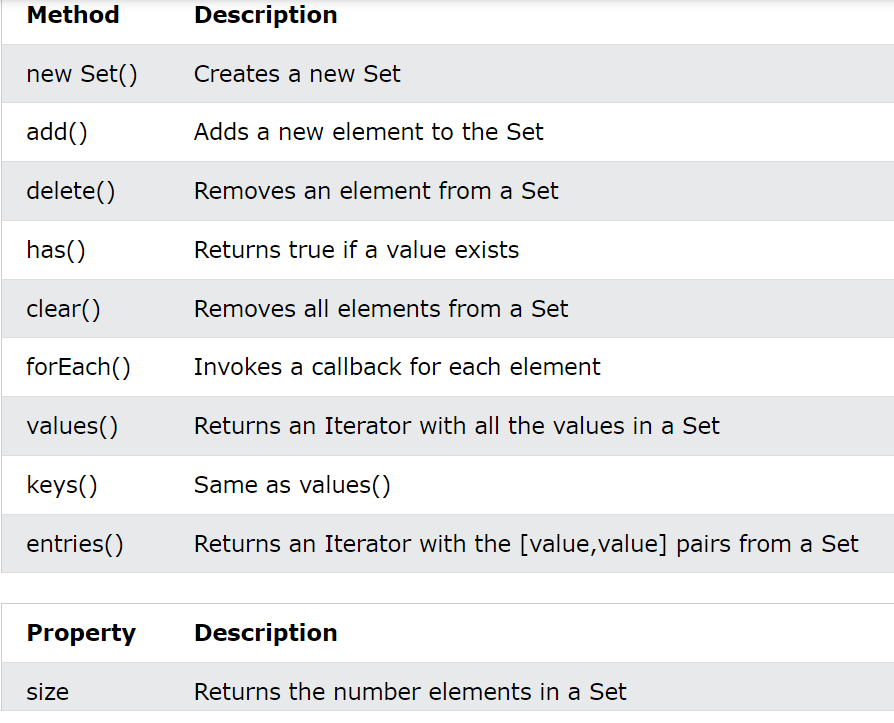
Object this is sort by value:

    .keys(tracker)

    .sort((a, b) => tracker[a] - tracker[b])

    .forEach(key => console.log(`${key} - ${tracker[key]}`))

Set and sets method



Take key value from object

let person = {

    name: 'John',

    age: 34,

    city: 'London'

};

let keyValues = Object.entries(person);

for (let [key, value] of keyValues) {

    console.log(`key: ${key} -> value: ${value}`)

}

Output :

key: name -> value: John

key: age -> value: 34

key: city -> value: London

take only key of object with for in

let person = {

    name: 'John',

    age: 34,

    city: 'London'

};

for (let key in person) {

    console.log(key)

}

Output:

name

age

city

присвояване на стойност на две променливи едновременно

function solve(array) {

    let phoneBook = {};

    array.forEach(info => {

        let [name, number] = info.split(' ');

        phoneBook[name] = number

    })

    Object

    .keys(phoneBook)

    .forEach( key => {

        console.log(`${key} -> ${phoneBook[key]}`)

    });

}

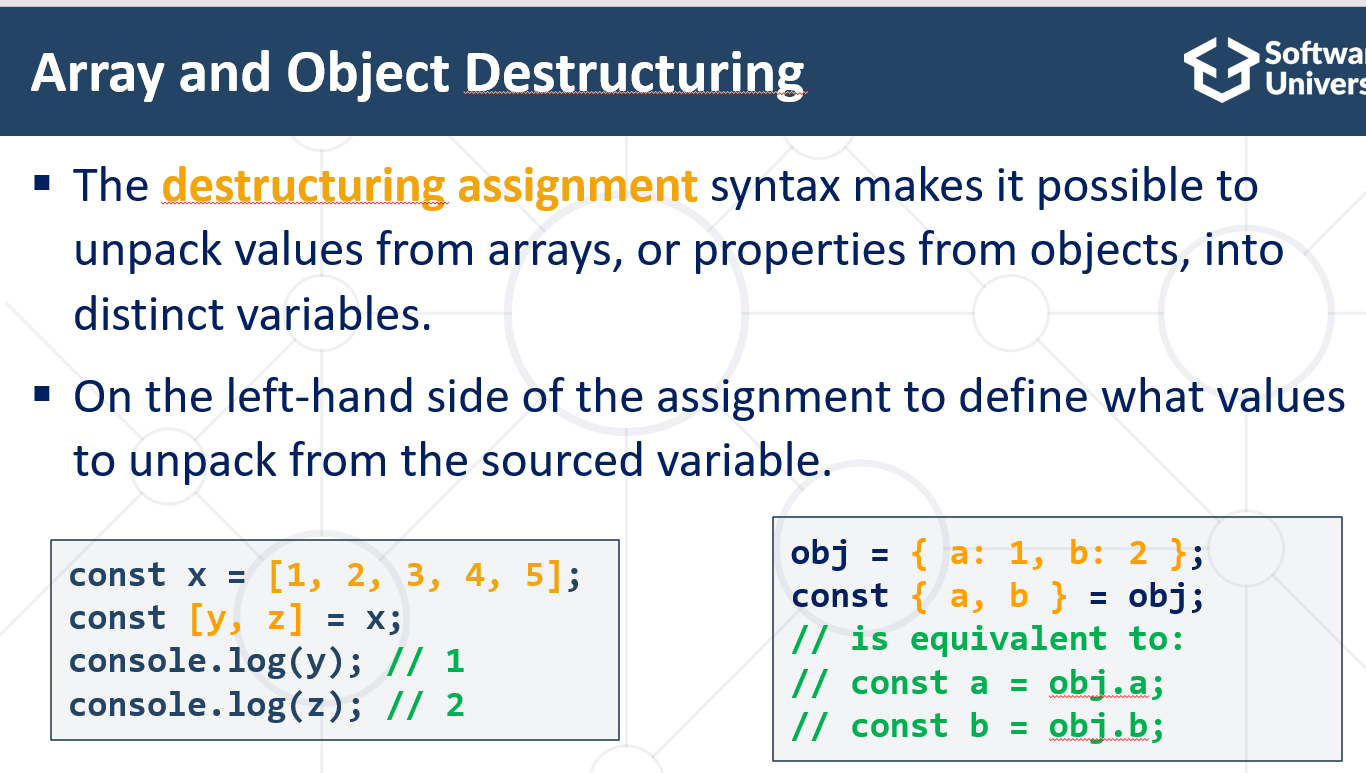
solve(['Tim 0834212554',

    'Peter 0877547887',

    'Bill 0896543112',

    'Tim 0876566344']

)



Sorted Object by key and value and return new Object

let person = {

    c: ['x', 'z', 'c'],

    b: ['z', 'b', 'p'],

    a: ['a', 'z', 'b'],

}

let sortedPerson = Object.fromEntries(

Object.entries(person)

.sort((a, b) => a[0].localeCompare(b[0]))

.map(element => {

    return [element[0], element[1]

    .sort((a, b) => a.localeCompare(b))]

}))

Output:

{a: ['a','b','z'],

b: ['b','p','z'],

c: ['c','x','z'],

}

DOM Methods

* **document.getElementById(id):** Returns the element with the specified ID.
* **document.getElementsByClassName(className)**: Returns an array-like object of all elements with the specified class name.
* **document.getElementsByTagName(tagName)**: Returns an array-like object of all elements with the specified tag name.
* **document.querySelector(selector)**: Returns the first element that matches the specified CSS selector.
* **document.querySelectorAll(selector)**: Returns a NodeList of all elements that match the specified CSS selector.
* **element.setAttribute(attribute, value)**: Sets the value of the specified attribute on the specified element.
* **element.getAttribute(attribute)**: Returns the value of the specified attribute on the specified element.
* **element.appendChild(node)**: Appends a new child node to the specified element.
* **element.removeChild(node)**: Removes a child node from the specified element.
* **element.classList.add(className)**: Adds a class name to the specified element's list of classes.
* **element.classList.remove(className)**: Removes a class name from the specified element's list of classes.
* **element.classList.toggle(className)**: Toggles a class name on the specified element's list of classes.
* **element.innerHTML**: Gets or sets the HTML content of the specified element.
* **element.innerText**: Gets or sets the text content of the specified element.
* **window.alert(message)**: Displays an alert box with the specified message.
* **window.confirm(message)**: Displays a dialog box with a message and two buttons: OK and Cancel.
* **window.prompt(message, default)**: Displays a dialog box that prompts the user for input.

Element.remove()

<div id="div-01">Here is div-01</div>

<div id="div-02">Here is div-02</div>

<div id="div-03">Here is div-03</div>

const element = document.getElementById("div-02");

element.remove(); // Removes the div with the 'div-02' id

select element by name

let email = document.querySelector('input[name="email"]').value;

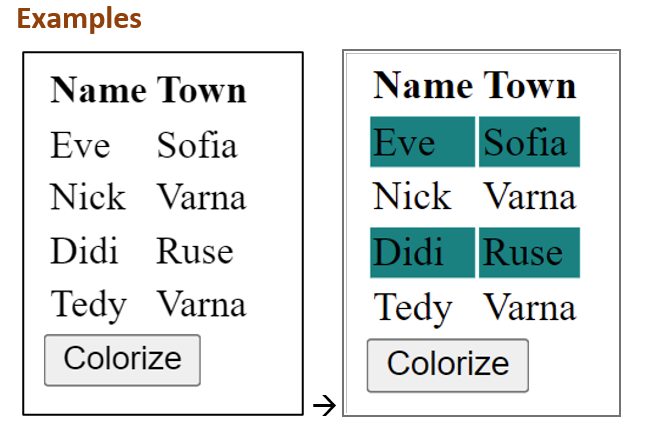
nth-of-type

function colorize() {

   let colorizeRows = Array.from(document.querySelectorAll('tr:nth-of-type(2n)'));

   colorizeRows.forEach(td => td.style.backgroundColor = 'teal');

}



Get element width and mouse move coords.

const gradient = document.getElementById('gradient');

gradient.addEventListener('mousemove', gradMove);

function gradMove(ev) {

        console.log(ev.offsetX, ev.offsetY, ev.target.offsetWidth);

    }

Check if radio buton is checked

function showOrHide(ev) {

        const profile = ev.target.parentElement

        const isActive = profile.querySelector('input[type="radio"][value="unlock"]').checked

        const hideDiv = Array

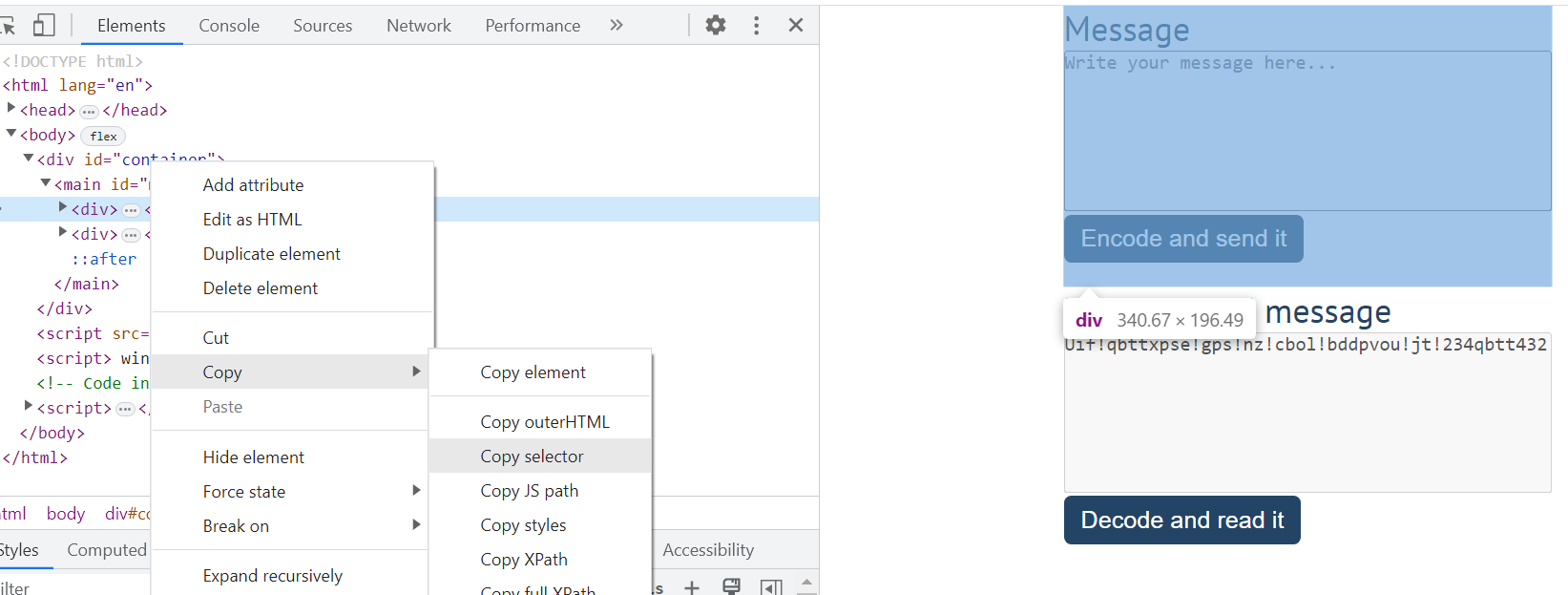
        .from(profile.querySelectorAll('div'))

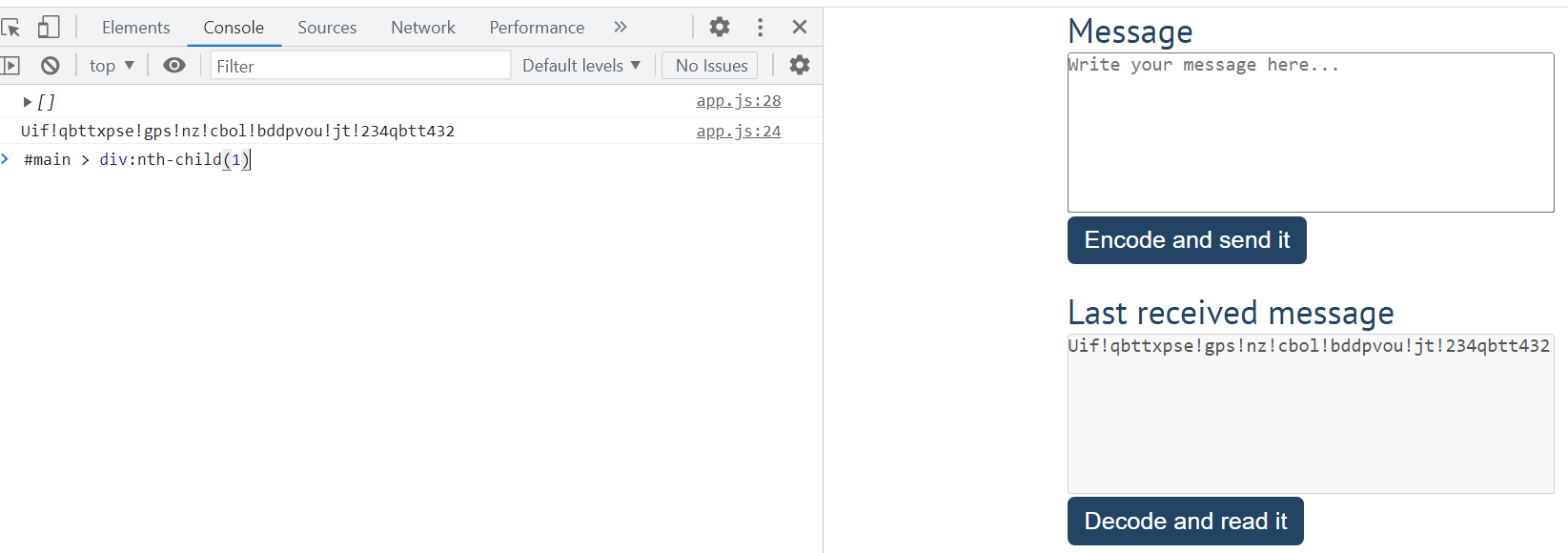
        .find(d => d.id.includes('HiddenFields'))

    }



Copy selector from browser





#main > div:nth-child(1)

This selector is copied from browser

DOM events

<https://www.w3schools.com/jsref/dom_obj_event.asp>

curentTarget vs target

const anchor = document.getElementsByTagName('a')[0];

document.querySelector('main').addEventListener('click', removeRow);

function removeRow(ev) {

   console.log(ev.target.tagName) curentTarget is the listener

    if (ev.target.tagName === 'A') {

     ev.target.parentNode.remove(); target is object who

    } treggered event

 }

Check is the checkbox is checked

function buy(e) {

    const allCheckbox = Array

      .from(document.querySelectorAll('input[type=checkbox]:checked'))

      // .filter(inp => inp.checked);

Функция на която единия от аргументите не е подаден

function test(param1, param2, param3) {

    if (param3) {

        return param1 + param2 + param3

    }

    return param3 функция на която единия от аргументите не е

} подаден не гърми а връща undefined което е

добре и можем да го проверим с if и да

console.log(test(1, 2, 3)) използваме това по нататък в кода

console.log(test(1, 2))

output:

6

undefined

Set create element and set atribute and append in html

let input1 = document.createElement('input');

    let input2 = document.createElement('input');

    let input3 = document.createElement('input');

let input4 = document.createElement('input');

    input1.setAttribute('type', 'radio');

    input1.setAttribute('class', 'input-1');

    let propertyType = 'type';

    input2[propertyType] = 'checkbox';

    input2.setAttribute('class', 'input-2');

    let props = {

        type: 'button' взимаме атрибута от ключа на обекта с for

    }; и го сетваме, ключа е type в случая

    for (let property in props) {

        input3[property] = props[property]; това е типа на атрибута в случая

    }; 'button'

    input3.setAttribute('class', 'input-3')

    input3.value = 'input-like-button'

можем да сетваме със съкратен синтаксис

input4.type = 'date'; type и value

    input4.setAttribute('class', 'input-4')

с дългия синтаксис сетваме class

  body.appendChild(input1); четири различни начина за сетване на атрибут

    body.appendChild(input2); третия е особенно полезен !!!!!!!!!!

    body.appendChild(input3);

    body.appendChild(input4);

button enabled and disabled

function deleteData(ev) {

        preview.removeChild(li)

        btnPublish.disabled = false

      }

classList methods (contains, add, remove)

for (let row of allRows) {

         if (row.classList.contains('select')) {

            row.classList.remove('select')

         }

         let trimmedText = row.textContent.trim()

         if (trimmedText.includes(searchText)) {

            row.classList.add('select')

         }

Exec function and regex

function extract(content) {

    let paragraph = document.getElementById(content).textContent;

    let pattern = /\(([^)]+)\)/g;

    let result = [];

    let match = pattern.exec(paragraph);

    while(match) {

      result.push(match[1]);

      match = pattern.exec(paragraph);

    }

    console.log(result.join('; '))

    return result.join('; ')

  }

Check is the option is selected

 function onClick(ev) {

     let options = Array.from(select.querySelectorAll('option'))

      if (options[1].selected){

        result.value = (num.value >>> 0).toString(2);

      }

      else if (options[2].selected) { convert to binary

        result.value = (num.value >>> 0).toString(16).toUpperCase();

      }

      else {

        result.value = '' convert to hexadecimal

      }

   }

preventDefault()

 window.addEventListener('load', solve);

function addSong(ev) { горната команда презарежда страницата , а

        ev.preventDefault() prventDefault() спира това поведение

    }

AJAX and fetch

function loadCommits() {

  const user = document.getElementById('username');

  const repo = document.getElementById('repo');

  const commits = document.getElementById('commits')

  const userValue = user.value

  const repoValue = repo.value

  const BASE\_URL = 'https://api.github.com/repos'

  fetch(`${BASE\_URL}/${userValue}/${repoValue}/commits`)

    .then((res) => {

      if (res.ok === false) {

        throw new Error(`${res.status} (Not Found)`)

      }

      return res.json()

    })

    .then(handleResponse) fetch block

    .catch((err) => {

      let li = document.createElement('li')

      li.textContent = err

      commits.appendChild(li)

    })

  function handleResponse(data) {

    data.forEach(obj => {

      let text = `${obj.commit.author.name}: ${obj.commit.message}`

      let li = document.createElement('li')

      li.textContent = text

      commits.appendChild(li)

    });

  }

}

Check is one element contain in another and remove it

let currentForecastDiv = document.querySelector('#current > div.forecast')

        if (current.contains(currentForecastDiv)) {

            current.removeChild(currentForecastDiv)

        }

CRUD operation with Fetch API

 await fetch(BASE\_URL, {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json'

            },

            body: JSON.stringify(messageObj)

        })

 await fetch(`${BASE\_URL}/${id}`, {

      method: 'PUT',

      headers: {

        'Content-Type': 'application/json'

      },

      body: JSON.stringify({

        author: author.value,

        title: title.value

      })

async function deletePerson(ev) {

       if (ev.target.tagName === 'BUTTON') {

         let id = ev.target.classList[0]

         await fetch(`${BASE\_URL}/${id}`,

         {

            method: 'DELETE'

         })

         loadPhones()

       }

    }

entries function and forEach with two parameters

Object масива е поставен в скоби

            .entries(data)

            .forEach(([key, value]) => {

                let option = document.createElement('option')

                option.value = key

                option.textContent = value.title

                select.appendChild(option)

                content.push({

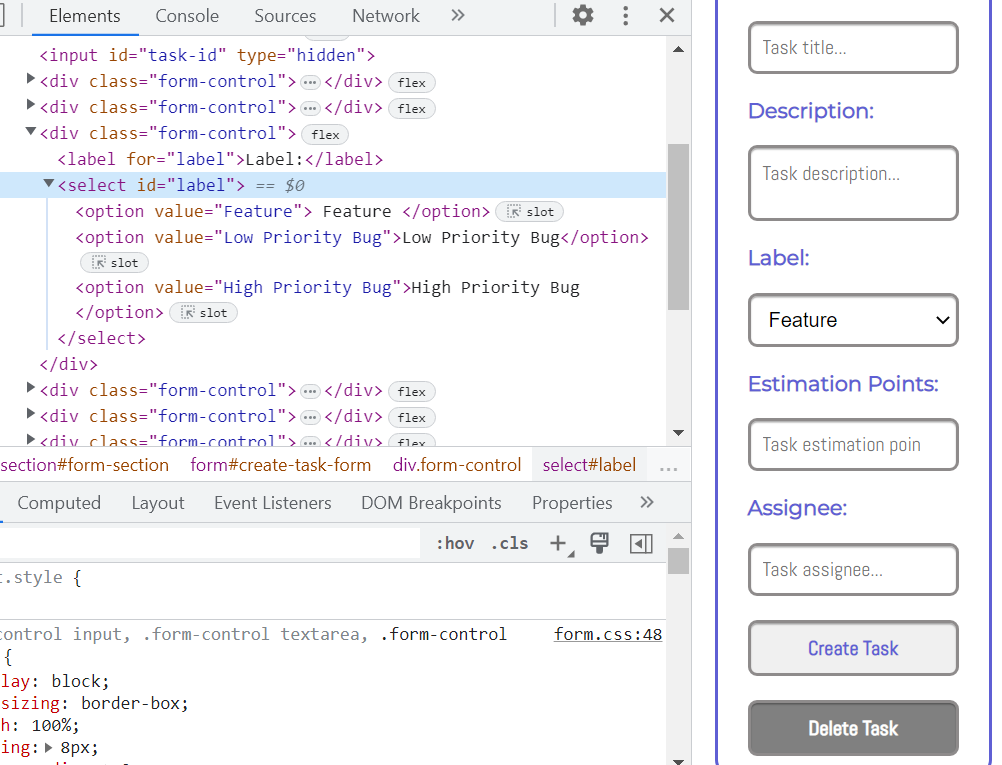
                    id: key,

                    body: value.body

                })

            })

Set value to select element in DOM



const labelEl = document.querySelector('#label') ако искаме да върнем данните във

labelEl.value = '' формата ,задаваме инфото във

labelEl.value = 'Feature' велюто на <select> елемента, a не

в отделните <option> елементи

Create elements Factory

 function createElement(tag, text, parent, \_id, \_class, attributes) {

        let element = document.createElement(tag)

        if (text) {

            if (tag === 'input' || tag === 'textarea') {

                element.value = text

            }

            else {

                element.textContent = text

            }

        }

        if (parent) {

            parent.appendChild(element)

        }

        if (\_id) {

            element.id = \_id

        }

        if (\_class) {

            element.classList.add(...\_class)

        }

        if (attributes) {

            for (key in attributes) {

                element.setAttribute(key, attributes[key])

            }

        }

        return element

    }

Check is input fields is empty

let isFull = Object.values(allInputs).every(el => el.value !== '')

       if (!isFull) {

        return

       }

Event.prevent default

 async function loadTasks(ev) {

        if (ev) {

            ev.preventDefault()

        }

        ev.preventDefault()

        todoList.innerHTML = ''

        const res = await fetch(BASE\_URL)

        const data = await res.json()

        Object.values(data).forEach(el => {

            todoList.innerHTML +=

                `<li id=${el.\_id}>

            <span>${el.name}</span>

            <button>Remove</button>

            <button>Edit</button>

        </li>`

        }) когато работим с localhost ,формата изпраща

    } заявка и сървара се мъчи да я намери и влиза

В безкраен цикъл затова слагаме prevent.Default

    async function addTask(ev) {

        ev.preventDefault()

        await fetch(BASE\_URL, {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json'

            },

            body: JSON.stringify({

               name: input.value

            })

        })

        input.value = ''

    }

<form method="POST">

      <h2>Add Item</h2>

      <input  type="text" name="title" id="title" placeholder="Title" />

      <button id="add-button" type="submit">Add</button>

      <button id="load-button" type="submit">Load All</button>

</form>

Select en option element with selected = true

 let options = Array.from(gender.children)

      const maleOption = options[0]

      const femaleOption = options[1]

      genderValue == "Male"

        ? (maleOption.selected = true)

        : (femaleOption.selected = true);

location.reload() - function refresh the page

find function

   function returnInfoToForm(ev) {

        let id = ev.target.parentNode.parentNode.id

        const obj = allProductInfo.find((el) => el.\_id === id)

        currentId = obj.\_id

        for(let key in allInputs) {

            allInputs[key].value = obj[key]

        }

        btnUpdate.disabled = false

        btnAdd.disabled = true

    }

Reset all input fields

// titleEl.value = '' <input id="title">

        // descriptionEl.value = '' <input id="description">

        // labelEl.value = '' <select id="label">

        // estimationPointsEl.value = '' <input id="points">

        // assigneeEl.value = '' <input id="assignee">

        form.reset()

Когато трябва да върнем данните в инпута но нямаме id нестваме двете функции в главната

function solve() {

  const input = {

    title: document.querySelector('#post-title'),

    category: document.querySelector('#post-category'),

    content: document.querySelector('#post-content')

  }

  const btnPublish = document.querySelector('#publish-btn')

  btnPublish.addEventListener('click', publish)

  const reviewList = document.querySelector('#review-list')

  const publishedList = document.querySelector('#published-list')

  const btnClear = document.querySelector('#clear-btn')

  btnClear.addEventListener('click', clearInfo)

  function publish(ev) {

    ev.preventDefault()

    let title = input.title.value

    let category = input.category.value

    let content = input.content.value

    if (title == '' || category == '' || content == '') {

        return

    }

    const li = document.createElement('li')

    li.className = 'rpost'

    li.innerHTML = `<article>

    <h4>${title}</h4>

    <p>Category: ${category}</p>

    <p>Content: ${content}</p>

    </article>

    <button class="action-btn edit">Edit</button>

    <button class="action-btn approve">Approve</button>`

    const editBtn = li.querySelector('.edit')

    editBtn.addEventListener('click', edit)

    const approveBtn = li.querySelector('.approve')

    approveBtn.addEventListener('click', approve)

    reviewList.appendChild(li)

    input.title.value = ''

    input.category.value = ''

    input.content.value = ''

    function edit() {

      input.title.value = title

      input.category.value = category

      input.content.value = content

      li.remove()

    }

    function approve() {

      lists.published.appendChild(li)

      editBtn.remove()

      approveBtn.remove()

    }

}