- 1. da se ispise ime svih studenata.

function Student(firstName, lastName) {

//   undefined

  firstName ? firstName : "---";

  this.firstName = firstName || "---";

  this.lastName = lastName || "---";

  this.getFullName = function () {

    console.log(`${this.firstName} ${this.lastName}`);

  };

}

const students = [new Student("Aida", "P"), new Student("Miroslav", "P")];

for (let element of students) {

  element.getFullName();

}

students.forEach(function (el) {

  console.log(el);

});

1. Create three inputs for numbers. Print the average of the three numbers in an h1 element If the average is larger or the same as 10 the result should be in green.If the average is smaller than 10 the result should be red.

    <div>

        <input type="text" id="put1">

         <input type="text" id="put2">

        <input type="text" id="put3">

        <button id="btn">Button</button>

    </div>

<div id="div2"> </div>

$(document).ready(function(){

    let button = $("#btn");

let div2 = $("#div2");

    button.click(function(){

    let num1 = parseFloat($("#put1").val());

let num2 = parseFloat($("#put2").val());

    let num3 = parseFloat($("#put3").val());

    let average = (num1 + num2 + num3) / 3;

    if(average >= 10){

        div2.css("color", "green");

    } else {

        div2.css("color", "red");

    }

    div2.html(`<h1> ${average.toFixed(2)} </h1>`);

});

});

Ovo je drugi nacin

let input = $(".input-form");

let result = $("#result")

function calculateAverage(num1, num2, num3) {

  let sum = num1 + num2 + num3;

  return Math.floor(sum / 3);

}

$("button").click(function () {

  let numOne = parseInt(input[0].value);

  let numTwo = parseInt(input[1].value);

  let numThree = parseInt(input[2].value);

  let average = calculateAverage(numOne, numTwo, numThree);

- ovde se pozivanjem funkcije daje prosek, ne moze se uzeti sum jer se ta varijabla

nalazi u drugoj funkciji, lokalnog opsega.

  if (average > 10) {

    result.text(average).css("color", "yellow");

  } else {

    result.text(average).css("color", "red");

  }

})

Ovde vidimo da mozemo uvek napraviti posebno funkciju za izracunavanje nekog rezultata I onda tu funkciju ubaciti u varijablu u daljem kodu ako nam je potrebna kroz pozivanje funkcije.

1. Create a header generator. Create two inputs, one for text and one for color. Create a button that says: generate h1. Create an h3 element for messages. When the button is clicked create a new header below the inputs and button. The new header should have the text and color from the inputs

If the person enters an invalid color or an empty text show an error message to the message element

You must use JQuery to complete the task.

$(document).ready(function(){

    let button = $("#btn");

let h3 = $("#h3");

button.click(function(){

        let text = $("#text").val();

        let color = $("#color").val();

        let isValid = true;

        if (!text || !color || !isNaN(color) || !isNaN(text)){

            h3.text("input invalid");

            isValid = false;

        }

        if (isValid){

        let newH1 = $("<h1>").text(text).css("color", color);

        button.after(newH1);

}

})

})

- Kreiranje tabele

function createTable() {

    let rows = parseInt(document.getElementById("rows").value);

    let cols = parseInt(document.getElementById("cols").value);

    let table = document.createElement("table");

    for (let i = 1; i <= rows; i++) {

      let row = table.insertRow();

      for (let j = 1; j <= cols; j++) {

        let cell = row.insertCell();

        cell.textContent = "Row-" + i + " Column-" + j;

      }

    }

    let tableContainer = document.getElementById("tableContainer");

    tableContainer.innerHTML = "";

    tableContainer.appendChild(table);}

- moze I ovako

function tabela(){

    let rows = parseInt(document.getElementById("rows").value);

let columns = parseInt(document.getElementById("columns").value);

let tabelaHTML = "<table>";

    for (let i = 1; i <= rows; i++){

        tabelaHTML += "<tr>";

        for (let j = 1; j <= columns; j++){

        tabelaHTML += "<td>Row-" + i + "Column-" + j + "</td>";

        }

        tabelaHTML += "</tr>";

    }

    tabelaHTML += "</table>";

    document.getElementById("divid").innerHTML = tabelaHTML;}

1. - Sledeci zadaci se nalaze na VIsual class3 - JS advanced.

- Create a button When the button is clicked, call the StarWars api for the first person. Print the person name in an **h1** tag. Print the person stats in a **table**: Height, Weight, Eye color, Hair color. **URL:** <https://swapi.dev/api/people/1>  
**NOTE:** JQuery will autmatically parse this call (js will not).

- Create a button When the button is clicked, get the data from a given url with an AJAX call. Print the name of the academy in an **h1** tag. Print all student names in an **unordered list**. **URL:** <https://raw.githubusercontent.com/Drakso/AJS2019/master/Class1/students.json>  
**NOTE:** You need to parse this data before using it.

- Create a button with text Get dog image When the button is clicked, call the Dog API to get random dog image. Create image and set its source as the received one from the API. Display image in the body of page. Remember to add some height and width to that image. **NOTE:** Each time button is clicked there should be created and shown new image for dog, so we got many dog images. **URL:** <https://dog.ceo/api/breeds/image/random>

1. Create 3 inputs: Color, FontSize, Text. Create a button for generating titles. When the button is clicked generate a new h1 element with the color, font size, and text from the inputs.

const generateBtn = document.getElementById("generateBtn");

const colorInput = document.getElementById("colorInput");

const fontSizeInput = document.getElementById("fontSizeInput");

const itemsInput = document.getElementById("itemsInput");

function onGenerate() {

  const list = document.createElement("ul");

  const itemsArray = itemsInput.value.split(","); // ['aa', 'bbb', 'ccc'];

  const color = colorInput.value;

  const fs = fontSizeInput.value;

  let isFormValid = true;

  if (!colorInput.value) {

    console.warn("Color is required");

    isFormValid = false;

  }

  if (!fontSizeInput.value || isNaN(fontSizeInput.value)) {

    console.warn("font size is required and must be a number");

    isFormValid = false;

  }

  if (!itemsInput.value) {

    console.warn("items are required");

    isFormValid = false;

  }

  if (isFormValid) {

    itemsArray.forEach(function (element) {

      const li = document.createElement("li");

      li.style.color = color;

      li.style.fontSize = `${fs}px`;

      li.innerText = element;

      list.appendChild(li);

});

document.body.appendChild(list);

    colorInput.value = "";

    fontSizeInput.value = "";

    itemsInput.value = "";

  }

}

// function onGenerate2() {

//   let list = '<ul>';

//   const itemsArray = itemsInput.value.split(',');// ['aa', 'bbb', 'ccc'];

//   console.log(itemsArray);

//   itemsArray.forEach(function(element){

//     list += `<li>${element}</li>`;

//   });

//   list += '</ul>';

//   console.log(list);

//   document.body.innerHTML += list;

// }

// generateBtn.addEventListener("click", onGenerate);

1. Create an array of 5 names. Create an HTML page with: A header, An empty unordered list, A button. When the button is clicked it should fill in the empty unordered list with the names of the array.

let list = document.getElementById("list");

let niz = ["Milica", "Sava", "Djordje", "Zarko", "Sava"];

function newList(){

    for (let ime of niz){

        list.innerHTML += `<li> ${ime}  </li>`

    }

}

const names = ["Alice", "Bob", "Charlie", "David", "Eve"];

function fillList() {

  let list = document.getElementById("name-list");

  for (let i = 0; i < names.length; i++) {

    let item = document.createElement("li");

    item.textContent = names[i];

    list.appendChild(item);

  }

}

6 Create 2 variables with arrow functions. 1. First arrow function will accept two parameters, one for element and one for color. The function should change the given element text color with the color given from the second color parameter. If no parameter is passed for color, the default value is **black**. 2. Second arrow function will accept two parameters, one for element and one for textSize. The function should change the given element text size to the number given from the second textSize parameter. If no parameter is passed for textSize, the default value is 24. Create an HTML document with two inputs, a button and an h1 header with some text. The first input should be for text size and the second for color. When the button is clicked the h1 header should change according to the input values ( change size as the first input value and color as the second input value ). Use the functions that we declared earlier and use arrow function for the event listener of the button.

 const header = document.getElementById("h1");

const textSizeInp = document.getElementById("textSize");

const textColorInp = document.getElementById("color");

const genTextColor = (element, color) => {

  element.style.color = color ? color : "black";

};

const genTextSize = (element, textSize) => {

  element.style.fontSize = textSize ? textSize + "px" : "24 px";

};

function handleStyle() {                          //  onclick u HTML- u se desava ova funkcija.

  genTextColor(header, textColorInp.value);

  genTextSize(header, textSizeInp.value);

1. Create an HTML page. On every refresh the page should pick a random color and change the background of the page. The RGB values of the color should be shown in the center of the page on every restart.

function getRandomNumber() {

  return Math.floor(Math.random() \* 256);

}

function getColor() {

  return `rgb(${getRandomNumber()}, ${getRandomNumber()},  ${getRandomNumber()})`;

}

7.Create an html page with a table and a button. When the button is clicked show results for the first 10 planets from the Star Wars api. The information in the table are: Planet Name, Population, Climate, Gravity. There should be a function that makes the call to the api for the planets ( should have URL for a parameter ) There should be a function that prints planets in to the table \*\*API URL: \*\* <https://swapi.dev/api/planets/?page=1>  
- After the user clicks the button to get the first 10 planets, a button should be shown below the table that says: NEXT 10. When the button is clicked you should make another call and get the next 10 planets and change the table contents with information about the next 10 planets. After the next 10 planets are shown the button NEXT 10 should disapear and a new button PREVIOUS 10 should appear. The previous button should return the first 10 planets in the table and hide the PREVIOUS 10 button and show the NEXT 10 button.\*\*API URL: \*\* <https://swapi.dev/api/planets/?page=2>

const button = document.getElementById("btn");

const mainTable = document.getElementById("mainTable");

let page = 1;

button.addEventListener("click", function(){

starWars();

    const newButton = document.createElement("button");

    newButton.innerText = "Next 10";

newButton.addEventListener("click", starWars2)

document.body.appendChild(newButton);

});

function starWars() {  
    fetch("https://swapi.dev/api/planets/?page=1")  
    .then(function(response){

        return response.json();

    })

    .then(function(data){

        const planets = data.results;

        contentTable (planets);

})

}

function starWars2() {

    page++

    console.log("page", page);

    fetch("https://swapi.dev/api/planets/?page=" + page)

    .then(function(response){

         return response.json()

    })

    .then(function(data){

        const otherPlanets = data.results;

        contentTable(otherPlanets);

    })

}

function contentTable (planets){

    mainTable.innerHTML = "";

        let arrayNames = ["Planet Name", "Population", "Climate", "Gravity"];

const headerTr = document.createElement("tr");

    arrayNames.forEach(function(names){

        const cell = document.createElement("th");

        cell.innerText = names;

        headerTr.appendChild(cell);

        });

        mainTable.appendChild(headerTr);

const arrayValues = ["name", "population", "climate", "gravity"];

    planets.forEach(function(planet){

        const headerValues = document.createElement("tr");

        arrayValues.forEach(function(values){

         const cell = document.createElement("td");

         cell.innerText = planet[values];

         headerValues.appendChild(cell);

});

mainTable.appendChild(headerValues);

    });

};

1. There is a JSON file with students. Make a call to the file and get the following data from it:

* All students with an average grade higher than 3
* All female student names with an average grade of 5
* All male student full names who live in Skopje and are over 18 years old
* The average grades of all female students over the age of 24
* All male students with a name starting with B and average grade over 2

 const students = () => {

    fetch("https://raw.githubusercontent.com/sedc-codecademy/skwd9-04-ajs/main/Samples/students\_v2.json")

    .then(response => response.json())

    .then(students => {

            const ex1 = students.filter(grades => grades.averageGrade > 3);

                        console.log(ex1);

            const ex2 = students.filter(nameAndGrade => nameAndGrade.gender === "Female" && nameAndGrade.averageGrade === 5);

                        console.log(ex2);

            const ex3 = students

                        .filter(male => male.age > 18 && male.gender === "Male" && male.city === "Skopje")

                        .map(male => `${male.firstName} ${male.lastName}`);

                        console.log(ex3);

            const ex4 = students

                        .filter(female => female.age > 24 && female.gender === "Female")

                        .map(female => female.averageGrade);

                        console.log(ex4);

           const ex5 = students.filter (average => average.gender === "Male" && average.averageGrade > 2 &&

                       average.firstName.startsWith("B") || average.lastName.startsWith("B"))

                       console.log(ex5);

    })}

    students();

1. Create a movie renting app. There should be an array of movie names. There should be an input and a search button. When the person enters a name of a movie it should search the array. If the name exists it should show an H1 element that says: "The movie can be rented" in green text. If the name does not exist it should show an H1 element that says: "The movie can't be rented" in red text. The input should not be case sensitive ( it should find the movie regardless of capital or small letters ).

let movies = ["Lord of the rings", "Harry Poter", "Joker", "Besa", "The Godfather"];

let header = document.getElementById("result");

let input = document.getElementById("searchInput");

let button = document.getElementById("searchBtn");

function rentingApp(){

     header.innerText="";

    if(!input.value){

        header.innerText = "You must enter a film";

        header.style.color = "black";

    } else {

        let result = findMovie();

        if(!result){

            header.innerText = "The movie was not found";

            header.style.color = "red";

        } else {

            header.innerText = "The movie can be rented";

            header.style.color = "green";

        }

    }

    input.value = "";

}

function findMovie(){

    let flag = false;

    for(movie of movies){

    if(input.value.toLowerCase() === movie.toLowerCase()){

        flag = true;

        break;

    }

}

return flag;

}

button.addEventListener("click", rentingApp);

1. Create a button When the button is clicked, call the StarWars api for the first person. Print the person name in an **h1** tag. Print the person stats in a **table**:

* Height
* Weight
* Eye color
* Hair color

**URL:** <https://swapi.dev/api/people/1>  
**NOTE:** JQuery will autmatically parse this call (js will not).

const sendRequestiButton = document.getElementById("sendRequest");

const personNameHeading = document.getElementById("personNameHeading");

function setHeading(value){

    personNameHeading.innerText = value;

}

function generateTable (data) {

const table = document.createElement("table");

    const headingsTr = document.createElement("tr");

//  ... da li sve atribute sa api a mozemo pretvoriti u niz??

const headings = ["height", "weight", "Eye color", "Hair color"];

    headings.forEach(heading => {

         const cell = document.createElement("th");

         cell.innerText = heading;

         headingsTr.appendChild(cell);

});

table.appendChild(headingsTr);

    const propertyNames = [ "height", "mass", "eye\_color", "hair\_color"];

const contentTr = document.createElement("tr");

    propertyNames.forEach(property => {

        console.log(property, data[property]);

        const cell = document.createElement("td");

        cell.innerText = data[property];

        contentTr.appendChild(cell);

});

table.appendChild(contentTr);

document.body.appendChild(table);

}

function onSendRequestClicked(){

    fetch("https://swapi.dev/api/people/1")

    .then(function(response){

        return response.json();

    })

.then(function(data){

        console.log(data);

        setHeading(data.name);

        generateTable(data);

    })

    .catch(function(){

        console.error("Star Wars API has failed");

    })

}

sendRequestiButton.addEventListener("click", onSendRequestClicked);

1. Create a button with text Get dog image When the button is clicked, call the Dog API to get random dog image. Create image and set its source as the received one from the API. Display image in the body of page. Remember to add some height and width to that image.

**NOTE:** Each time button is clicked there should be created and shown new image for dog, so we got many dog images. **URL:** <https://dog.ceo/api/breeds/image/random>

function dogImage (){

    fetch("https://dog.ceo/api/breeds/image/random")

    .then(function(response){

        return response.json();

    })

    .then(function(data){

        const imageUrl = data.message;

        createImage(imageUrl);

    })

    .catch(function(error){

        console.error(error);

    })

}

function createImage(imageUrl){

    const container = document.getElementById("container");

container.innerHTML = "";

    const image = document.createElement("img");

    image.src = imageUrl;

    image.style.width = "300px";

    image.style.height = "auto";

    container.appendChild(image);

}

document.getElementById("sendRequest").addEventListener("click", dogImage);

1. Create constructor function Book. The structure should be:
   * title - string
   * author - string
   * year - store it as a Number, even someone proceedes string here
   * ratings - array of ratings, not settable.
   * addRating - method which receives a rating object as a param. Use object destructuring to extract only grade, user and comment from it.
     + if user not defined set it as unknown
     + if comment not set, set it as ''
   * getAverageRating - method which calculates average rating for a book and returns it
   * displayDetails - method which displays details about book. Display details as:
     + title - author
     + Year: {year}
     + If there are ratings, display average rating as Average rating: {average}. If no rattings added, display message Book doesn't have ratings yet. Be the first one to add rating!.
   * displayRatings - method which displays each rating for book if there are ratings set.

function Book(title, author, year) {

    this.title =  title || '---';

    this.author = author || '---';

    this.year = Number(year);

    this.ratings = [];

    this.addRating = (rating) => {

      const smallRating = ({ user = "unknown", grade, comment = "" } = rating);    //.....    mora da se stave zagrade() da bi se posle pusalo i da bi to bilo kao celina, gradi novi objekat

      this.ratings.push(smallRating);

    };

    this.getAverageRating = () => {

      let sum = 0;

      this.ratings.forEach((rating) => {

        sum += rating.grade;

      });

      let average = sum / this.ratings.length;

      return average;

    };

    this.displayDetails = () => {

      console.log(`${this.title} - ${this.author} Year: ${this.year}`);

      if (this.ratings.length === 0) {

        console.log(

          `Book doesn't have ratings yet. Be the first one to add rating!`

        );

      } else {

        console.log(`Average rating: ${this.getAverageRating()}`);

      }

    };

    this.displayRatings = () => {

      this.ratings.forEach((rating) => {

        console.log(rating)

      })

    }

  }

const rating1 = {

    grade: 5,

    user: "john",

    comment:

      "I love it! It was exactly what I was looking for! Now my Harry Potter collection is more complete then ever before",

    title: "Awesome!",

    id: 12,

    userId: 1,

    userEmail: "john@john.com",

    isVerified: true,

  };

  const rating2 = {

    grade: 4,

    user: "ana",

    comment:

      "This (the first in the series), or any other Harry Potter book for that matter, needs no introduction. Once you've followed Harry to Hogworts and back during his first term, you'll be coming back for more time and time again!",

    title: "A great escape!",

    id: 10,

    userId: 2,

    userEmail: "ana@ana.com",

    isVerified: true,

  };

  const rating3 = {

    grade: 3,

    user: "tom",

    comment: "Pages missing from my copy of the book.",

    title: "Missing pages",

    id: 11,

    userId: 3,

    userEmail: "tom@tom.com",

    isVerified: true,

  };

  const book = new Book(

    "Harry Potter and the Philosopher's Stone",

    "J. K. Rowling",

    "1997"

  );

  book.addRating(rating1);

  book.addRating(rating2);

  book.addRating(rating3);

  book.displayDetails();

  book.displayRatings();

  console.log("\n");

  const book2 = new Book("Atomic Habits", "James Clear", "2018");

  book2.displayDetails();

1. After user clicks on Get Fact button, there should be loaded fact in fact-text text, and image in fact-img.

API for fetching the fact text is: https://meowfacts.herokuapp.com/ API for fetching the image is: https://random.dog/woof.json

NOTE: Only if both - fact text and image are successfully fetched, we display both of them. If for ex. fetcing image fails, neither of them should be shown on the web page.

Use object destructuring and asynchronous functions where possible

Najbolje ovako:

const btn = document.getElementById("factButton");

const text = document.getElementById("fact-text");

const img = document.getElementById("fact-img");

btn.addEventListener("click", () => {

    fetch("https://meowfacts.herokuapp.com/")

    .then((response) => response.json())

    .then((data) => {

        text.innerText = data.data[0];

        fetch("https://random.dog/woof.json")

        .then((response) => response.json())

        .then((data) => {

            const url = data.url;

            img.src = url;

        })

        .catch((error) => console.error(error));

    })

    .catch((error) => console.error(error));

})

Moze I ovako postupno

button = document.getElementById("factButton");

areaForAdding = document.getElementById("fact-text");

img = document.getElementById("fact-img");

const textApi = () => {

    return new Promise((resolve, reject) => {

        fetch("https://meowfacts.herokuapp.com/")

        .then((response) => response.json())

        .then((data) => {

            areaForAdding.innerText = data.data[0];

            resolve(true);

        }) .catch((error) => {

            reject(error);

        })

    })

}

const picApi = () => {

    return new Promise((resolve, reject) => {

        fetch("https://random.dog/woof.json")

        .then((response) => response.json())

        .then((data) => {

            const url = data.url;

            img.src = url;

            resolve(true);

        }) .catch((error) => {

            reject(error);

        })

    })

}

const fetcingImage = () => {

    return new Promise((resolve, reject) => {

        textApi().then(() =>{

            picApi().then(() => {

                resolve("Good job");

            }).catch(() =>{

                reject("Bad try");

            })

         }).catch(() => {

            reject("Bad try");

     })

  })

}

button.addEventListener("click", () => {

    fetcingImage().then((response) => {

        console.log(response);

    }).catch((error) => {

        console.error(error);

    });

});

Moze I ovako, citkije:

const btn = document.getElementById("factButton");

const text = document.getElementById("fact-text");

const img = document.getElementById("fact-img");

const myFetch = (url) => fetch(url).then((result) => result.json())

const fetchFact = () => myFetch("https://meowfacts.herokuapp.com/");

const fetchImage = () => myFetch("https://random.dog/woof.json");

const getFact = () => {

  return new Promise((resolve, reject) => {

    fetchFact()

      .then((response) => {

        const fact = response.data[0];

        resolve(fact);// 'Neki text'

      })

      .catch((error) => {

        reject(error);

      });

  });

};

btn.addEventListener("click", () => {

  getFact()

    .then((fact) => {// 'Neki text'

      fetchImage()

        .then((imageObject) => {

          const { url } = imageObject;

          text.innerText = fact;

          img.src = url;

        })

        .catch((err) => console.error(err));

    })

    .catch((err) => console.error(err));

});