### Java Script – uvod

* Moze se ubaciti kao eksterni file. Na taj nacin krijemo kod od korisnika.
* <script src="javascript.js"></script>
* Viselinijski komentari u kodu pisu se : */\* komentar \*/*
* Obican komentar : *//Komentar…*
* **${i + 1}** predstavlja redni broj studenta povećan za 1 (jer indeksi kreću od 0), a ta tačka se dodaje nakon rednog broja kako bi označila kraj rednog broja i početak imena.

Provera kojeg je tipa var :

var test= null;

console.log (typeof test); //shows object

ako ne bi bilo NULL onda bi izbacilo na kraju UNDEFINED sto znaci da nije data vrednost varijabli. Ako stavimo null, onda iznacuje OBJECT. Ovo je primer kako proveravamo kog je tipa varijabla.

var first = 1;

console.log( first, "first");

//comparison operators

var x = 5;

var y = 2;

console.log(x == y); //false

console.log(x != y); //true

console.log(x > y); //true

console.log(x < y); //false

console.log(x >= y); //true

console.log(x <= y); //false

//asignment operators

var x = 5;

x += 2; // 7

console.log(x);

x -= 2; // 5

console.log(x);

x \*= 2; // 10

console.log(x);

x /= 2; // 5

console.log(x);

x %= 2; // 1

console.log(x);

**Variables** – Variables are used to store data in memory.

!= - must not be equal

Prvo se deklarise, onda joj se daje vrednost :

var number; // Declaration

number = 5; // Initialization

var number = 5; // Declaration and Initialization

**Primitivni tipovi podataka (vrednosti)**

* **Null**
* **Undefined**
* **Numbers**
* **Strings**
* **Booleah**

**a++** operacija, prvo izbacuje pocetnu vrednost, pa onda uvecava za 1.

**++a** operacija, odmah sabira.

**ASSIGNMENT OPERATORS**

+= a += b a = a + b

-= a -= b a = a - b

\*= a \*= b a = a \* b

/= a /= b a = a / b

%= a %= b a = a % b

**Cas 2**

**Strings** – konkatenacija (povezivanje stringova uz)

Mogu se ispisivati uz dvostruke ili jednostruke navodnike.

* Konkatenacija (“rezultat” **+** **“** zbog razmaka **” +** “drugi rezultat”)
* Interpolacija ( ` $ { rezultat } ` )

var rez1 = `tekst ${prom1} neki tekst ${prom2} još teksta`;  
var rez2 = "tekst " + prom1 + " neki tekst " + prom2 + " još teksta";

const fullName = firstName + " " + lastName;

var rez = prom1.concat(prom2) + "!";

– combining strings known as concatenation.

* console.log (“rezultat”)
* console.log ( ` $ { rezultat } ` )

**NaN** – not a number (matematicka operacija u kojoj nisu obe promenljive bojevi ce izbaciti NaN).

**Proverava da li je broj** console.log (Number.isNaN(b))

**Infinity** –

**LOGICAL OPERATORS**

* AND&& (svi uslovi moraju biti tacni da bi sve bilo true)
* var rez = (statusA === "open" && longitude > 0); // true and true = true
* OR|| (jedan je dovoljan kao tacan da bi sve bilo tacno)
* var rez = (statusA === "open" || latitude > 0);
* NOT! (! Ispred true daje false I obratno)

var rez2 = (statusA !== "open");

console.log("rez", rez2); //false

**FALSE/TRUE**

**FALSY VALUES:**

false

0, -0

"" (empty string)

null

undefined

NaN (invalid number)

**TRUTHY VALUES:**

true

"hello", "0"

25

[ ], [ 1, "2", 3 ] (arrays)

{ }, { a: 42 } (objects)

function foo() { .. } (functions)

everything else :

**IF ELSE**

      var x = 5;

      var a = 10;

      var b = 20;

      var z = ( x > 10 ? a : b);

Ako je vece od 10 bice a, ako nije, bice b.

let pare = prompt ("Koliko imas para?");

let kolikoimaspara = parseInt(pare);

if (kolikoimaspara <=500) {

    console.log ("Kupi nesto");

}

else {

    console.log ("Nema para");

}

Ternarni izraz umesto IF ELSE:

var broj = 5;

var rezultat = (broj > 0) ? "Pozitivan" : "Negativan";

console.log(rezultat);

**Vazno**

parseInt – pretvara string u broj

let unos = prompt ("Unesi broj dana u nedelji");

let dan = parseInt(unos);

let stringBroj = "42";

let broj = parseInt(stringBroj); // broj će biti 42

Zagrade [] – za nizove vrednosti

* Console.log – izbacuje vrednost iz varijable ZNAK na mestu koji je odredjen ostatkom u zagradi.

if (!isNaN(godina)) {

    var znak = ["Rat", "Ox", "Tiger", "Rabbit", "Dragon", "Snake", "Horse", "Goat", "Monkey", "Rooster", "Dog", "Pig"];

    var ostatak = (unos - 4) % 12;

    console.log (znak[ostatak]);}

    else {

        console.log ("Nista");

    }

if ( !**isNaN**(unos) – kako se postavlja uslob za NaN

if (!isNaN(godina))

.toLowerCase()

let plata = 100;

let dohodak = 0;

let iznos = prompt ("Unesi ime pozicije:");

let unos = iznos.toLowerCase();

Primer:

// function rezultat (r,p) {

//     let povrsina = r\*p;

//     console.log (povrsina, 'povrsina');

// }

// rezultat (2, 0.30);

.toString – pretvara u string

let number = 1425;

let numberString = number.toString();

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

    console.log(numberString[i]); }

Zadatak : Ispisati ime kineskog znaka u horoskopu. Postoji formula kako se izracunava i broj znakova je 12 i svaki zauzima svoje mesto.

var unos = prompt ("Unesi godinu rodjenja:");

var godina = parseInt(unos);

if (!isNaN(godina)) {

    var znak = ["Rat", "Ox", "Tiger", "Rabbit", "Dragon", "Snake", "Horse", "Goat", "Monkey", "Rooster", "Dog", "Pig"];

    var znakindex = (unos - 4) % 12;

    console.log (znak[znakindex]);}

    else {

        console.log ("Nista");

    }

**CAS 3**

* **Switch**

Može se koristiti kada imamo više slučajeva koji se mogu odnositi na jednu promenljivu.

* let unos = prompt ("Unesi broj dana u nedelji");
* let dan = parseInt(unos);
* switch (dan) {
* case 1 :
* console.log ("Monday");
* break;
* case 2 :
* console.log ("Thusday");
* break;
* case 3 :
* console.log ("Wednesday");
* break;
* case 4 :
* console.log ("thursday");
* break;
* case 5 :
* console.log ("Friday");
* break;
* case 6 :
* console.log ("Sathurday");
* break;
* case 7 :
* console.log ("Sunday");
* break;
* default:
* console.log ("Unesi pravilan unos")
* break;
* }

**FUNCTIONS**

function mojafunkcija (x, y) { - TELO FUNKCIJE

    let mnozenje = x \* y;

    let sabiranje = x + y;

    console.log (mnozenje, 'Ovo je mnozenje');

    console.log (sabiranje, 'Ovo je sabiranje');

}

mojafunkcija (2, 2); - POZIVANJE FUNKCIJE

1.zadatak (izracunavanje povrsine kvadrata)

function krug (raduis) {

   let povrsina = Math.PI \* Math.pow (radius, 2);

   return povrsina;

}

 let radius = prompt("Upisi radius");

radius = parseInt(radius);

if (isNaN(radius) || radius <= 0) {

    console.log ("unesi pravilan unos")

} else {

    let kraj = krug(radius);

       console.log (kraj);

}

Matematicke operacije : ( Math.PI ) ( Math.pow ) (Math.floor)

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/PI>

- RETURNING VALUESvraca vrednost. Tj. Drugo varijabli mozemo dodeliti vrednost prve varijable ako je u prvoj varijabli stavljen RETURN. **Pogledati primer sa povrsinom kruga.**

function myFunction(x, y) {

let myVar = x \* y;

console.log(myVar);

return myVar;

}

myFunction(1,3);

myFunction(2,5);

myFunction(100,232131);

let myresult = myFunction(7,8);

**HOW FUNCTIONS WORK**

A sandwich and a math equation

Description automatically generated with medium confidence

MISMATCH – ne podudaraju se parametri

  function calculateLoan   ( amount, months,interest,name ) {

    //lots and lots of code

    console.log(name, amount, months, interest);

  }

calculateLoan(1000, 12, 7, "John Doe");  //correct

calculateLoan(1000, 12, 7, "John Doe", "Something Extra");  //extras are ignored

calculateLoan(1000, 12);   //missing are passed as undefined

DEFAULT PARAMETER :

* https://www.tutorialspoint.com/javascript/javascript\_builtin\_functions.htm

function printValues(name = "Ivan", age, city) {

    console.log(name, age, city);

  }

  printValues("Marija", 24, "Belgrade");

  printValues("Marija ", 24, " Belgrade", 1998);

  printValues("Marija ", 24);

  printValues("Marija ", " Belgrade");

  printValues(1, 2, 3);

  printValues();

ARROW function

var multiply = (x, y) => {

    var tekst = "Ovo je funkcija";

    console.log(tekst)

};

* https://www.webprogramiranje.org/arrow-funkcija/

**RETURN IN FONCTIONS**

**Ako dodajemo return da bi vratili rezultat I ispisali, onda nakon tela funkcije moramo dodati novu varijablu u cijoj vrednosti pozivamo funkciju I onda u console.log dodajemo tu novu varijablju da bi se ona ispisala.**

let sredstva = 500;

function bankomat() {

    let unos = prompt("Unesi željnu sumu novca:");

    let iznos = parseInt(unos);

    if (isNaN(iznos) || iznos <= 0) {

        return "Molimo unesite validan iznos.";

    } else if (iznos > sredstva) {

        return "Nemate dovoljno para na računu.";

    } else {

        return "Biće isplaćeno " + iznos + " evra.";

    }

}

let rezultat = bankomat();

console.log(rezultat);

Takodje moguce je pozvati funkciju u console.log :

function temperatura (celzijus) {

   let farenhajt = celzijus \* 1.8 + 32;

   return (farenhajt)

}

let farenhajt = temperatura(50)

console.log (farenhajt);

….

function calculateAge (birthyear, currentyear) {

       let godina = currentyear - birthyear;

       return godina;

}

console.log (calculateAge (1995, 2024));

**SCOPE – Opseg** odredjuje se ovim zagradama {}

* Scope (opseg ili domen) u JavaScript-u odnosi se na oblast koda gde se promenljive mogu koristiti ili gde im je vrednost vidljiva. Postoje dva osnovna tipa opsega u JavaScript-u: globalni opseg (global scope) i lokalni opseg (local scope).

Globalni opseg (Global Scope): Varijable koje su definisane van svih funkcija imaju globalni opseg. Ove promenljive mogu se koristiti bilo gde u kodu, kako u funkcijama, tako i izvan njih. One su dostupne globalno i nazivaju se globalne promenljive.

let globalnaPromenljiva = "Ja sam globalna!";

function primerFunkcije() {

console.log(globalnaPromenljiva);

}

primerFunkcije(); // Ispisuje: Ja sam globalna!

Globalna moze biti I ako se neka varijabla ne deklarise ali joj se da vrednost u sklopu funkcije. Ona automatski postaje globala.

myFunction();  
  
// code here can use carName  
  
function myFunction() {  
  carName = "Volvo";  
}

Lokalni opseg (Local Scope): Promenljive koje su definisane unutar funkcija imaju lokalni opseg. Ove promenljive su vidljive samo unutar funkcije u kojoj su definisane. Ponekad se lokalno referišu kao promenljive sa funkcionalnim opsegom (function scope).

function primerFunkcije() {

let lokalnaPromenljiva = "Ja sam lokalna!";

console.log(lokalnaPromenljiva);

}

primerFunkcije(); // Ispisuje: Ja sam lokalna!

// console.log(lokalnaPromenljiva); // Ovde bi izazvalo grešku, jer 'lokalnaPromenljiva' nije vidljiva izvan funkcije.

Važno je napomenuti da promenljive koje su definisane u lokalnom opsegu (npr. unutar funkcije) neće biti vidljive izvan tog opsega. Ali se sa istim imenom mogu koristiti u drugim funkcijama. Zato one imaju function scope.

Blokovski opseg (Block Scope): Uvodi se sa ECMAScript 6 (ES6) i predstavlja opseg koji vezuje promenljive unutar blokova koda, kao što su if, for, while, switch, i sl.

if (true) {

let x = 10; // Blokovski opseg

console.log(x); // Ispisuje: 10

}

// console.log(x); // Ovde bi izazvalo grešku, jer 'x' nije vidljiva izvan bloka if.

Let I const su block scope varijablje. Nisu vidljive van bloka. Var je vidljiva !!!!!!!!!!!!!!!!!

Koristite scope kako biste kontrolisali vidljivost i životni vek promenljivih. Ovo pomaže u izbegavanju nesporazuma i konflikata između različitih delova vašeg koda.

**ARRAY – NIZOVI**

**let niz = new Array(duzina)** – ovako se pise nova varijabla ako zelimo da uvedemo novi niz u program. To se radi kada mu rezervisemo mesto ali mu nisu date vrednosti.

let nizSaElementima = new Array(5)

**ili**

let prazanNiz = [];

* Pisu se u [ ] zagradama. Uglavnom su isti tipovi vrednosti u njima.
* let days = ['Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun'];
* ovde vidimo kako se ispisuje neka vrednost iz niza.

    var znak = ["Rat", "Ox", "Tiger", "Rabbit", "Dragon", "Snake", "Horse", "Goat", "Monkey", "Rooster", "Dog", "Pig"];

    var ostatak = (unos - 4) % 12;

    console.log (znak[ostatak]);}

* .length – sluzi za proveru duzine niza

 const days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    console.log (" prikazi : ", days[2]);

    console.log (" duzina niza je: ", days.length);

* Zamena vrednosti u nizu:

    const days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    days [2] = "Neki drugi dan:";

    console.log (days[2]);

* Zamena vrednosti na poslednjoj poziciji:
* days[days.length - 1] = 'Sunday';
* Dodavanje na kraju niza - .length

        const days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

        days[days.length] = 'No more days in the week'

        console.log (days);

* Dodavanje na kraju - .push

    let days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    days.push("DODATAK");

    console.log(days);

* Dodavanje na pocetku : .unshift

    let days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    days.unshift('DODATAK');

    console.log (days);

* Skidanje sa pocetka .shift

   let days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    days.shift();

    console.log (days);

* Skidanje sa kraja .pop

    let days = [ 'Mon', 'Tues', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun' ];

    days.pop ();

    console.log (days);

**Petlje –**

* **For Of**
* // Definiramo niz objekata
* let ljudi = [
* { ime: 'Ana', godine: 30 },
* { ime: 'Marko', godine: 25 },
* { ime: 'Petra', godine: 35 },
* { ime: 'Ivan', godine: 40 }
* ];
* // Prikazujemo osobe starije od 30 godina
* for (let osoba of ljudi) {
* if (osoba.godine > 30) {
* console.log(osoba.ime + ' ima više od 30 godina.');
* }
* }

**DOM:**

<html>

  <head>

    <title>My cool page</title>

  </head>

  <body>

    <div id="main" class="myDiv">

      <h1 id="myTitle">Yea! Cool page</h1>

      <p title="Short explanation" align="center" class="myParagraph">

        This page is really cool. Please don't change anything!

      </p>

    </div>

    <div class="myDiv">

      <p align="center" class="myParagraph second">

        No really, trust me dude!

      </p>

      <text align="center">

        See? It has random text!

      </text>

    </div>

    <script src="dom.js"></script>

  </body>

</html>

* document.getElementById(“MyTitle”)
* let pera = document.getElementById("myTitle");
* console.log(pera);
* console.log(pera.textContent);
* document.getElementByClassName(“myParagraph”)

let paragraphs = document.getElementsByClassName("myParagraph");

let secondParagraph = document.getElementsByClassName("second");

console.log(paragraphs); // Will give us an array with 2 paragraphs

console.log(paragraphs[0]); // Will give us the first paragraph

console.log(secondParagraph); // Will give us an array with 1 paragraph

console.log(secondParagraph[0]); // Will give us the only paragraph

* document.getElementByTagName("p");
* let pera = document.getElementsByTagName("p");
* console.log (pera);
* console.log (pera.textContent);
* document.querySelectorAll !!!!!!!!
* let pera1 = document.querySelectorAll("p");
* let pera2 = document.querySelector(".myParagraph");
* let pera3 = document.querySelector("#myTitle");
* .elementSibling, .nextElementSibling, .previousElementSibling.

let paragraph = document.getElementsByClassName("myParagraph")[0];

let sibling = paragraph.nextElementSibling; // We add the next sibling element here

console.log(paragraph); // The first paragraph

console.log(sibling); // The second paragraph

console.log(sibling.previousElementSibling); // Will log the first paragraph again

* .parentElement
* let paragraph = document.getElementsByClassName("myParagraph")[0];
* let parentDiv = paragraph.parentElement; // We take the parent of the paragraph
* console.log(paragraph); // The first paragraph
* console.log(parentDiv);
* .children, .firstElementChild, .lastElementChild.

let myDiv = document.getElementById("main");

let divChildren = myDiv.children; // All children of myDiv

let divFirstCh = myDiv.firstElementChild; // Get first child of myDiv

let divLastCh = myDiv.lastElementChild; // Get last child of myDiv

console.log(divChildren); // Array of all children of myDiv

console.log(divFirstCh); // The first child element of myDiv

console.log(divLastCh); // The last child element of myDiv

.textContent (with white spaces) If used on parent, it will take all child elements without tags

<p id="demo" onclick="myFunction()">Click me to change my textual content.</p>

<script>

function myFunction() {

  document.getElementById("demo").textContent = "I have changed!";

}

- .InnerText (gets or changes only the text without white spaces)

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

 console.log(header.innerText); // Checking what the text is

 header.innerText = ""; // Changing the text to nothing

 console.log(header.innerText);

 header.innerText = "New TEXT!"; - menja se tekst sa =

 console.log(header.innerText);

 header.innerText += " Ultra new text"; - dodaje se tekst sa +=

 console.log(header.innerText);

* I ovako se moze promeniti tekst
* header3last.innerText = "Text added from JS";
* header1last.innerText = "Text added from JS also";
* .innerHTML += `<p>…..</p>` (to se koristi samo na DOM elementima, da se direktno na njih lepi. Ako nije DOM element, pristupa se na drugi nacin:)

let myDiv = document.getElementById("main");

console.log(myDiv);

// Adding new element in the div

myDiv.innerHTML += `<p class="new">

Paragraph generated from JavaScript

</p>`;

console.log(myDiv);

noviProzor.document.body.innerHTML ????

- Delete text from an paragraph

<p id="demo">Click the button to delete my HTML content (innerHTML).</p>

<button onclick="myFunction()">Try it</button>

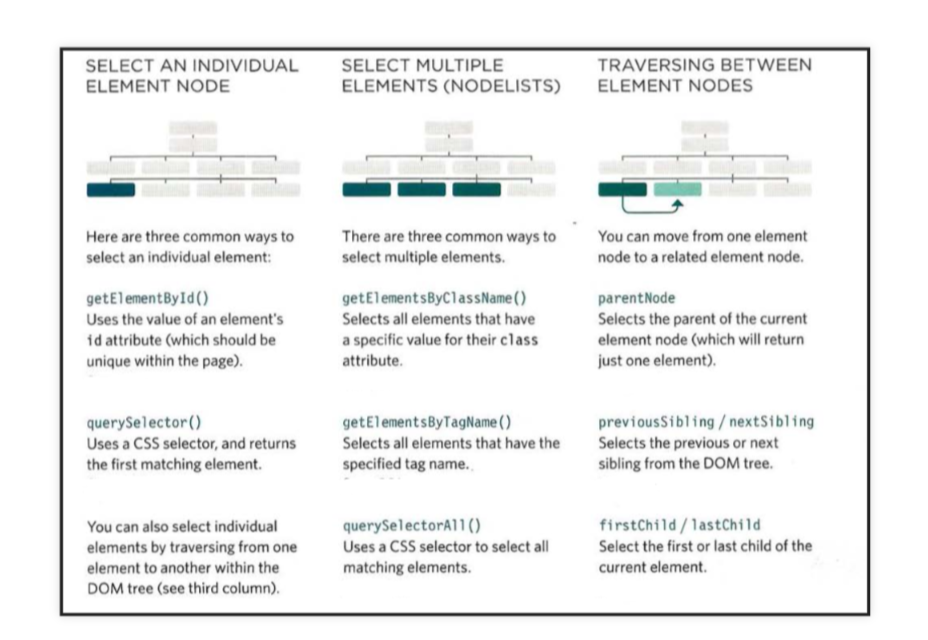
<script>

function myFunction() {

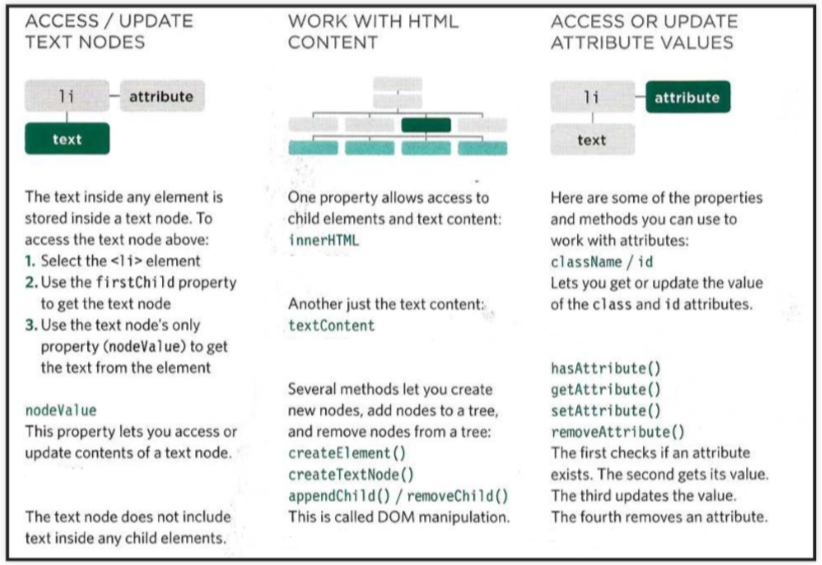
  document.getElementById("demo").innerHTML = "";

}

</script>



* IZUCITI !!!!



<div id="first" class="aDiv">

  <h1 id="myTitle">Yea! Cool page</h1>

  <p class="paragraph">

    This is an exercise. It's pretty easy

  </p>

</div>

<div class="anotherDiv">

  <p class="paragraph second">

    No really, It's easy!

  </p>

  <text>

    It's about selecting elements and

  </text>

</div>

<div>

  <h1>This should be changed</h1>

  <h3>And also this!</h3>

</div>

**MANIPULATE THE DOM**

* Select the first div
* Select all paragraphs - moze biti i: **document.querySelectorAll(“p”)**
* Select the last div - moze biti i: **document.querySelector("div:last-of-type")**

Moze I ovako 3. zadatak:  **let sviDivovi = document.querySelectorAll("div");**

**let poslednjiDiv = sviDivovi[sviDivovi.length - 1];)**

* Select the header 3 in the last div
* Select the header 1 in the last div
* Get the text from the first paragraph in the second div
* Add the word "text" to the text element in the second div
* Change the text of the header 1 in the last div
* Change the text of the header 3 in the last div

let firstDiv = document.querySelector("div");

let allParagraphs = document.getElementsByTagName("p");

let lastDiv = document.getElementsByTagName("body")[0].lastElementChild;

let header3last = lastDiv.getElementsByTagName("h3")[0];

let header1last = header3last.previousElementSibling;

let paragraphSecondText = document.getElementsByClassName("second")[0].innerText;;

document.querySelector("text").innerText += " text!";

header3last.innerText = "Text added from JS";

header1last.innerText = "Text added from JS also";

**EVENTS**

* Mogu se uvoditi u HTML dokumentu
* function countRabbits() {
* for(let i=1; i<=3; i++) {
* alert("Rabbit number " + i);
* }
* }
* <input type="button" onclick="countRabbits()" value="Count rabbits!"></input>
* Moze posebno u javascript-u.

var elem = document.getElementByid("btnHello");

elem.onclick = function() {

        alert("Hello World");

    };

<input type="button" id="btnHello"  value="Say Hello">

* EVENTS LISTENERS
* elem.addEventListener("blur", greetingFunc);
* function greetingFunc(){
* var greeting = `Hello ${elem.value}`;
* alert(greeting);
* }
* elem.addEventListener("blur", greetingFunc);

var elem = document.getElementById("redDiv");

function setColorToDiv(event){

    event.target.style.backgroundColor = "red";

}

elem.addEventListener("mousemove",setColorToDiv);

elem.removeEventListener("mousemove", setColorToDiv);

Value from input

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

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

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

 // gets the value

 console.log(`Somebody entered the name: ${input.value}`);

 // changes the value to empty string

 input.value = "";

})

* If you need to pass arguments to a function that is called by an event handler or listener, you wrap the function call in an anonymous function.

Pisemo samo function() ako pozivamo ono sto neko treba da unese, a ne postoji u kodu.

**- Izmena stila paragrafa na drugi nacin**

let paragraph = document.getElementById("myParagraph");

let btn = document.getElementById("myBtn");

function changeElement(element){

    element.style.color = "red";

    element.style.backgroundColor = "blue";

    element.style.fontSize = "34px";

}

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

    changeElement(paragraph);

});

.join(' ') ; – koristi se za ispisivanje niza kao stringa.

window.open(‘’) – za otvaranje novog prozora (window.open(url, target, options))

document.write() – za ispisivanje. !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

document.body.innerHTML += `<p>…..</p>` (to se koristi samo na DOM elementima, da se direktno na njih lepi. Ako nije DOM element, pristupa se na drugi nacin:). Moze biti **myDiv.innerHTML += <p>…</p>**

noviProzor.document.getElementById('mojDiv').innerHTML +=

let vezba = [ime1, prezime1, sifra1, email1];

// Kreirajte string od niza

let stringNiza = vezba.join(', ');

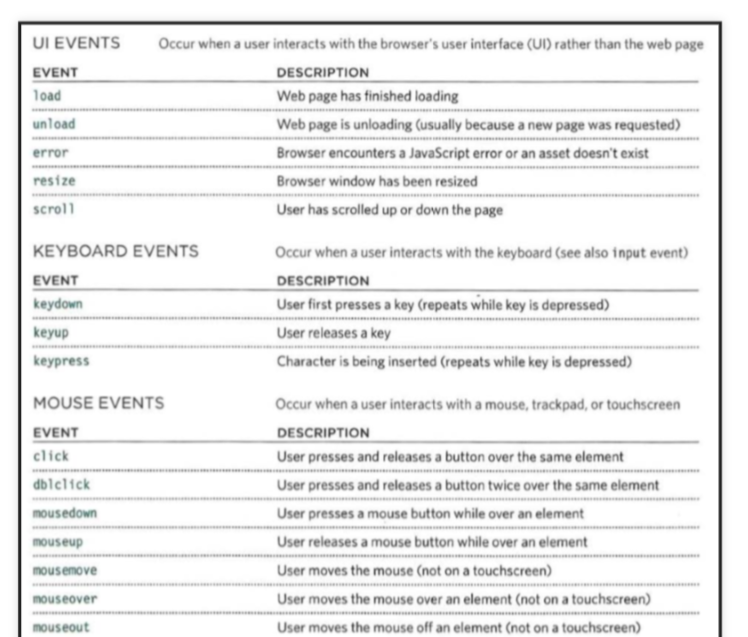
// Otvaranje novog prozora i postavljanje HTML sadržaja na string niza

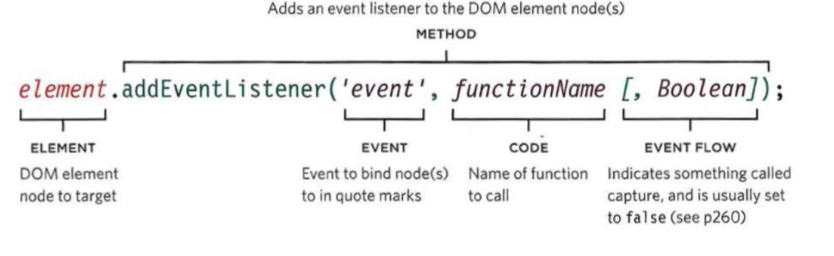
let noviProzor = window.open('');

noviProzor.document.write(`<p>${stringNiza}</p>`);

}

dugme.onclick = ispis;





* Menjanje CSS a kroz DOM
* let element = document.getElementById("myElement");
* element.style.color = "red"; // Changes text color to red
* element.style.display = "none"; // Removes the element
* element.style.backgroundColor = "blue"; // Changes background color to blue

Neki event - i za HTML

onchange    An HTML element has been changed

onclick The user clicks an HTML element

onmouseover The user moves the mouse over an HTML element

onmouseout  The user moves the mouse away from an HTML element

onkeydown   The user pushes a keyboard key

onload  The browser has finished loading the page

Promena boje nekog header-a

var heading = document.getElementById("heading");

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

    this.style.color = "red";

});

Dodavanje pogledati - querySelectorAll gde uzima paragraf sa odredjenom klasom I dodaje u P sadrzaj prvog P sa klasom intro.

 <p>Finding HTML Elements by Query Selector</p>

<p class="intro">Hello World!.</p>

<p class="intro">This example demonstrates the <b>querySelectorAll</b> method.</p>

<p id="demo"></p>

<script>

const x = document.querySelectorAll("p.intro");

document.getElementById("demo").innerHTML =

'The first paragraph (index 0) with class="intro" is: ' + x[0].innerHTML;

</script>

Pristupanje formi

 <form id="frm1" action="/action\_page.php">

    First name: <input type="text" name="fname" value="Donald"><br>

    Last name: <input type="text" name="lname" value="Duck"><br><br>

    <input type="submit" value="Submit">

  </form>

  <p>These are the values of each element in the form:</p>

  <p id="demo"></p>

  <script>

  const x = document.forms["frm1"];

  let text = "";

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

    text += x.elements[i].value + "<br>";

  }

  document.getElementById("demo").innerHTML = text;

  </script>

Dodavanje datuma u neki paragraf

 <p id="demo"></p>

 <script>

 document.getElementById("demo").innerHTML = "Date : " + Date();

 </script>

Validacija numerickog prikaza

<input id="numb">

<button type="button" onclick="myFunction()">Submit</button>

<p id="demo"></p>

<script>

function myFunction() {

  // Get the value of the input field with id="numb"

  let x = document.getElementById("numb").value;

  // If x is Not a Number or less than one or greater than 10

  let text;

  if (isNaN(x) || x < 1 || x > 10) {

    text = "Input not valid";

  } else {

    text = "Input OK";

  }

  document.getElementById("demo").innerHTML = text;

}

</script>

Kako se na unos inputa menjaju slova

 <html>

    <body>

    <h1>JavaScript HTML Events</h1>

    <h2>The oninput Attribute</h2>

    Enter your name: <input type="text" id="fname" oninput="upperCase()">

    <p>When you write in the input field, a function is triggered to transform the input to upper case.</p>

    <script>

    function upperCase() {

      const x = document.getElementById("fname");

      x.value = x.value.toUpperCase();

    }

    </script>

    </body>

    </html>

Ako se umesto **oninput** stavi **onchange**, slova ce se povecati kada se zavrsi sa upisom u input.

Kreiranje novog P I ubacivanje njega u postojeci DIV !!!

    <!DOCTYPE html>

<html>

<body>

<h2>JavaScript HTML DOM</h2>

<p>Add a new HTML Element.</p>

<div id="div1">

<p id="p1">This is a paragraph.</p>

<p id="p2">This is another paragraph.</p>

</div>

<script>

const para = document.createElement("p");

const node = document.createTextNode("This is new.");

para.appendChild(node);

const element = document.getElementById("div1");

element.appendChild(para);

</script>

</body>

</html>

Brisanje, uklanjanje P na klik

 <!DOCTYPE html>

 <html>

 <body>

 <h2>JavaScript HTML DOM</h2>

 <h3>Remove an HTML Element.</h3>

 <div>

 <p id="p1">This is a paragraph.</p>

 <p id="p2">This is another paragraph.</p>

 </div>

 <button onclick="myFunction()">Remove Element</button>

 <script>

 function myFunction() {

 document.getElementById("p1").remove();

 }

 </script>

 </body>

 </html>

- <https://www.w3schools.com/js/js_htmldom_nodelist.asp>

Kako se pravi dynamic table .insertRow() .insertCell()

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);}

**Object**

const person = {

    firstName : "John",

    lastName  : "Doe",

    age     : 50,

    eyeColor  : "blue"

  };

Let x = person; - objekat se moze smestiti u jednu varijablu

Delete person.age; - brisanje propertia

Loop FOR IN

<p id="demo"></p>

const person = {

  fname:"John",

  lname:"Doe",

  age:25

};

let txt = "";

for (let x in person) {

  txt += person[x] + " ";

}

document.getElementById("demo").innerHTML = txt;

Ovde se mora koristiti person[x], sa zagradama, jer je x varijabla.

Petlja for...in se koristi za iteriranje kroz enumerabilna svojstva objekta.

const obj = { a: 1, b: 2, c: 3 };

for (let key in obj) {

  console.log(key + ": " + obj[key]);

}

Izbacice a: 1

Nasted objects

const myObj = {

    name: "John",

    age: 30,

    cars: {

    car1: "Ford",

    car2: "BMW",

    car3: "Fiat"

    }

  }

myObj.cars.car2;

Nasted arrays and objects !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

<p id="demo"></p>

let x = "";

const myObj = {

  name: "John",

  age: 30,

  cars: [

    {name:"Ford", models:["Fiesta", "Focus", "Mustang"]},

    {name:"BMW", models:["320", "X3", "X5"]},

    {name:"Fiat", models:["500", "Panda"]}

  ]

}

for (let i in myObj.cars) {

  x += "<h2>" + myObj.cars[i].name + "</h2>";

  for (let j in myObj.cars[i].models) {

    x += myObj.cars[i].models[j] + "<br>";

  }

}

document.getElementById("demo").innerHTML = x;

Object.value da se od vrednosti propertija objekta napravi niz

<p id="demo"></p>

const person = {

  name: "John",

  age: 30,

  city: "New York"

};

document.getElementById("demo").innerHTML = Object.values(person);

Get

const person = {

    firstName: "John",

    lastName: "Doe",

    language: "en",

    get lang() {

      return this.language;

    }

  };

    document.getElementById("demo").innerHTML = person.lang;

Set

const person = {

    firstName: "John",

    lastName: "Doe",

    language: "",

    set lang(lang) {

      this.language = lang.toUpperCase();;

    }

  };

  person.lang = "en";

  document.getElementById("demo").innerHTML = person.language;

https://www.w3schools.com/js/js\_object\_accessors.asp

Object Constructor - kada imamo vise slicnih objekata.

<p id="demo"></p>

function Person(first, last, age, eye) {

  this.firstName = first;

  this.lastName = last;

  this.age = age;

  this.eyeColor = eye;

}

const myFather = new Person("John", "Doe", 50, "blue");

const myMother = new Person("Sally", "Rally", 48, "green");

- ovde dodajemo metodu u objekat

myFather.name = function() {

  return this.firstName + " " + this.lastName;

};

document.getElementById("demo").innerHTML =

"My father is " + myFather.name();

Promena vrednosti

function Person(firstName,lastName,age,eyeColor) {

    this.firstName = firstName;

    this.lastName = lastName;

    this.age = age;

    this.eyeColor = eyeColor;

    this.changeName = function (name) {

      this.lastName = name;

    }

  }

  // Create a Person object

  const myMother = new Person("Sally","Rally",48,"green");

  // Change last name

  myMother.changeName("Doe");

document.getElementById("demo").innerHTML =

"My mother's last name is " + myMother.lastName;

Dodavanje propertija na objekat. Na konstruktor ne moze biti dodat, osim u konstruktor funkciji.

function Person(first, last, age, eye) {

  this.firstName = first;

  this.lastName = last;

  this.age = age;

  this.eyeColor = eye;

}

const myFather = new Person("John", "Doe", 50, "blue");

const myMother = new Person("Sally", "Rally", 48, "green");

myFather.nationality = "English"; - dodavanje na objekat

document.getElementById("demo").innerHTML =

"The nationality of my father is " + myFather.nationality;

Dodavanje metode na objekat I konstruktor

<p id="demo"></p>

function Person(first, last, age, eye) {

  this.firstName = first;

  this.lastName = last;

  this.age = age;

  this.eyeColor = eye;

  this.name = function() {

    return this.firstName + " " + this.lastName - dodavanje na konstruktor

  };

}

const myFather = new Person("John", "Doe", 50, "blue");

myFather.name = function() {

    return this.firstName + " " + this.lastName - dodavanje na objekat

  };

document.getElementById("demo").innerHTML =

"My father is " + myFather.name();

Bulit-in java script constructor

let x1 = "";      // string

let x2 = 0;       // number

let x3 = false;   // boolean

const x4 = {};    // Object object

const x5 = [];    // Array object

const x6 = /()/;  // RegExp object

const x7 = function(){};  // function

// Display the type of all

document.getElementById("demo").innerHTML =

"x1: " + typeof x1 + "<br>" +

"x2: " + typeof x2 + "<br>" +

"x3: " + typeof x3 + "<br>" +

"x4: " + typeof x4 + "<br>" +

"x5: " + typeof x5 + "<br>" +

"x6: " + typeof x6 + "<br>" +

"x7: " + typeof x7 + "<br>";

https://www.w3schools.com/js/js\_object\_maps.asp

Create a student registry form. The form should have: First Name / Last Name / Age

The form should have a save button which will create a student object and add it to an array (students). Log the array after every save to see the results in the console.

<body>

    <h2>Student Registry Form</h2>

    <form id="studentForm">

      <label for="firstName">First Name:</label>

      <input type="text" id="firstName" required>

      <label for="lastName">Last Name:</label>

      <input type="text" id="lastName" required>

      <label for="age">Age:</label>

      <input type="number" id="age" required>

       <button type="submit">Save</button>

    </form>

    <script src="homework7.js"></script>

</body>

- myFunction.preventDefault();

let students = [];

function Student (firstName, lastName, age){

    this.firstName = firstName;

    this.lastName = lastName;

    this.age = age;

}

document.getElementById("studentForm").addEventListener("submit", function(myFunction){

myFunction.preventDefault();

    let firstName = document.getElementById("firstName").value;

   let lastName = document.getElementById("lastName").value;

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

   let student = new Student (firstName, lastName, age);

   students.push(student);

   console.log(students);

   document.getElementById("studentForm").reset();

})

- myFunction.preventDefault();

- mora da se ubaci da bi se sprecio podrazumevani tok forme, a to je da se ona automatski salje na server. To se ubacuje da bi mogli da manipulisemo formom I da unosimo vise puta unose.

document.getElementById("studentForm").reset();

- da se resetuje forma

Kada u HTML-u imamo onclick= function(), to znaci da u JS kodu moramo imatu glavnu funkciju koju pozivamo. Jer onclick zahteva tako da bi se izvrsio.   
Ako u JS kodu pisemo add.EventListener, to znaci da u JS kodu ne moramo imati glavnu funkciju.

Ovde je add.EventListener

    <input type="text" id="name"><br>

    <input type="text" id="kind"><br>

    <input type="text" id="message"><br>

    <button id="speakButton">Speak</button>

    <script>

        document.getElementById("speakButton").addEventListener("click", function(){

            let name = document.getElementById("name").value;

            let kind = document.getElementById("kind").value;

            let message = document.getElementById("message").value;

            let Animal = {

                name: name,

                kind: kind,

                speak: function(){

                    console.log(`${this.name} says: '${message}'`)

                }

            }

            Animal.speak();

        });

    </script>

Ovde je onclick

 <input type="text" id="name"><br>

    <input type="text" id="kind"><br>

    <input type="text" id="message"><br>

    <button onclick="speak()">Speak</button>

    <script>

        function speak(){

            let name = document.getElementById("name").value;

            let kind = document.getElementById("kind").value;

            let message = document.getElementById("message").value;

            let Animal = {

                name: name,

                kind: kind,

                speak: function(){

                    console.log(`${this.name} says: '${message}'`)

                }

            }

            Animal.speak();

        }

    </script>

**classList** je svojstvo DOM elementa koje pruža pristup listi klasa tog elementa. Ovo svojstvo omogućava manipulaciju klasama elementa, kao što su dodavanje, uklanjanje i provera prisustva određene klase.

Evo nekoliko često korišćenih metoda classList:

add(className): Dodaje navedenu klasu elementu.

remove(className): Uklanja navedenu klasu iz elementa.

toggle(className): Ako klasa postoji, uklanja je; ako ne postoji, dodaje je.

contains(className): Proverava da li element sadrži navedenu klasu i vraća true ili false.

element.classList.add("new-class");

Prompt se moze uvesti na dva nacina

1. Nacin:

let Animal = {

     name: "",

     kind: "",

     message: "",

    speak: function(){

   console.log (`${this.name} says: '${this.message}'`)

    }

}

Animal.name = prompt("Unesi ime zivotinje:");

Animal.kind = prompt("Unesi vrstu zivotinje:");

Animal.message = prompt("Kojim zvukom se oglasava ?");

Animal.speak();

1. Nacin:

let book = {

    title: prompt("Unesite naslov knjige:"),

    author: prompt("Unesite ime autora:"),

    readingStatus: confirm("Da li ste već pročitali knjigu?"),

info: function() {

        if (this.readingStatus) {

            return `Već ste pročitali '${this.title}' od strane ${this.author}.`;

        } else {

            return `Još uvek treba da pročitate '${this.title}' od strane ${this.author}.`;

        }

    }

};

console.log(book.info());

Postavljanje uslova: (ako nije true ili ako nije unet, izbacice tackice) Ternarni operator

function Student (firstName, lastName, age, email){

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

    this.lastName = lastName || "...";

}

  firstName ? firstName : "---";

Do While

let broj = 0; // undefined

do {

  broj = Number(prompt("Enter a number")); //9

} while (broj !== 0);

let i = 0;

do {

  console.log("Vrednost i:", i);

  i++;

} while (i < 5);

Niz od dva objekta: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

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();

}

for.Each

let numbers = [1, 2, 3, 4, 5];

numbers.forEach(function(number, index) {

  console.log("Element na indeksu " + index + " je " + number);

});

let test = [

    true,

    12,

    13,

"Bob",

]

  const onlyStrings = [];

/  test.forEach(function (element) {

    if (typeof element === "string") {

      onlyStrings.push(element);

    }

  });

  console.log(onlyStrings);

PRIMER KAKO SE MENJA BOJA

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Change Color on Click</title>

<style>

.student {

cursor: pointer;

 }

</style>

</head>

<body>

<h2>Students</h2>

<ul id="studentList">

</ul>

<script>

class Student {

constructor(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

this.color = 'black'; // Početna boja

}

}

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

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

students.forEach((student, index) => {

const listItem = document.createElement('li');

listItem.textContent = `${student.firstName} ${student.lastName}`;

listItem.classList.add('student');

listItem.style.color = student.color; // Postavljamo početnu boju

listItem.addEventListener('click', () => {

// Menjamo boju kada se klikne

student.color = student.color === 'black' ? 'red' : 'black';

listItem.style.color = student.color;

});

studentList.appendChild(listItem);

});

</script>

</body>

</html>

**Librairies - JQuery -** <https://www.w3schools.com/jquery/jquery_animate.asp>

- koriste se da bi se pojednostavio kod. Zapravo je to vec spreman kod samo se importuje. Koriste se kada treba ubaciti neke velike sadrzaje kao npr. kalendari, tebele…

Moze se koristiti CDN (content delivery network) - na serveru neke biblioteke. Ili koristimo lokalne.

JQuery **$(selector).action()**

- najkoriscenija biblioteka. Prepoznatiljiva po znaku dolara $. Ima dosta ugradjenih funkcija.

Benefiti: laksa DOM manipulacija, lakse funkcionalnosti AJAX poziva.

    <div class="wrapper">

        <h1 id="mainTitle">Hey this is my page</h1>

        <div class="innerWrapper">

          <h3 id="firstTitle">This is some subheader</h3>

          <p>Hello!</p>

          <p>Yes it is a sub header!</p>

        </div>

        <div class="innerWrapper">

          <h3 id="SecondTitle">This is some other subheader</h3>

          <p>Hello again!</p>

          <p>Yes it is another sub header!</p>

        </div>

        <button>It does nothing :</button>

        <input type="text" value="Cool">

      </div>

let allElements = $("\*");

let byIdElement = $("#mainTitle");

let byClassElements = $(".innerWrapper");

let byTagNameElements = $("p");

let wrapperAndInnerWrapper = $(".wrapper,.innerWrapper");

let firstParagraph = $("p:first");

let lastParagraph = $("p:last");

let firstChildHeaders = $("h3:first-child");

let lastChildParagraphs = $("p:last-child");

let secondChildParagraphs = $("p:nth-child(2)");

console.log(firstParagraph.text());

// selecting

allElements.first();

allElements.last();

allElements.find(".wrapper").find("p").last();

allElements[0];

// Get all DOM elements from a JQuery array

byClassElements.get();

// Get next or previous element

firstParagraph.next();

firstParagraph.prev();

- primer kako se moze naci neka claka I poslednji p element u tom divu sa klasom wrapper.

allElements.find(".wrapper").find("p").last();

.find - ugradjena funkcija

$(“.clasaDiva”).find(“p”).last();

.get - pravi niz od selektovanog elementa. U ovom slucaju, od svih elemenata koji imaju klasu innerWrapper, napravice niz I sadrzati ga u varijabli. Nakon toga mozemo normalno manipulisati nizom.

let byClassElements = $(".innerWrapper").get();

console.log(byClassElements);

- promena teksta u svim elementima sa klasom innerWrapper

$(document).ready(function() {

    // Selektujemo sve elemente sa klasom .innerWrapper i koristimo .get() metodu

    var divElements = $(".innerWrapper").get();

    // Iteriramo kroz dobijeni niz elemenata i menjamo tekst svakog div-a

    for (var i = 0; i < divElements.length; i++) {

        divElements[i].textContent = "Div broj " + (i + 1);

    }

});

- ovde ce se napraviti niz od svih p elemenata

var paragraphElements = byClassElements.find("p").get();

.val() ; .html() ; .hide() ; .show() ; .text(); .css();

$("input").first().val();// get value from input

secondChildParagraphs.hide();// hide elements

secondChildParagraphs.show();// show elements

byClassElements.first().html();// get innerHTML

byClassElements.first().html(`<h1> This is a new h1! </h1>`);// set innerHTML

byTagNameElements.last().text();// get text

byTagNameElements.last().text("This is a new text");// set text

byTagNameElements.last().css("color");// getting CSS

byTagNameElements.last().css("color", "red");// poslednji ce imati boju red !!!!!!!!

$("h1").first().after("<p>new p tag after the h1</p>");

$("h1").first().before("<p>new p tag before the h1</p>");

- dodaje element nakon elementa

 $("h1").first().after("<h2> Novi header </h2>");

- dodavanje novog element-a u html

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

addEventListener - u JQuery- u.

let button = $("button").first();

// simple on event click

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

  console.log("Hello");

})

// Even simpler click event

button.click(function(e){

  console.log("HELLO AGAIN");})

**AJAX AND JSON**

We request things from the internet using HTTP calls. To do that dynamically we use AJAX (Asynchronous JavaScript And XML). With this technology we request things from the internet. If we are allowed, we get a response in a package ( the things we requested ). The package that we requested is in a format called JSON.

JSON file

const astrosList = document.getElementById("astros");

    fetch("http://api.open-notify.org/astros.json")

       .then(function(response){

        return response.json();

       })

       .then(function(spaceData){

        console.log(spaceData);

        spaceData.people.forEach((astro) => {

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

            li.innerText = astro.name;

            astrosList.appendChild(li);

        });

        })

        .catch(function(){

            console.error("Route has failed");

        });

Konvertovanje json u objekat I objekat u json string:

const response = `{

    "trainer": "Trainer name",

    "assistant": "Assistant name",

    "students": [

      "Bob",

      "Samantha",

      "Chris",

      "Jill",

      "Greg"

    ],

    "academy": "Code"

  }`;

const jsObject = JSON.parse(response); - pretvaranje json-a u objekat

console.log(jsObject);

let someObject = {

  trainer: "Trainer name",

  assistant: "Assistant name",

  students: ["Bob", "Samantha", "Chris", "Jill", "Greg"],

  academy: "Code",

};

const toJson = JSON.stringify(someObject); pretvaranje JS objekta u JSON format

console.log(toJson);

Asinhrone funkcije

1. Prvo se vidi kako se u trecoj asinhronoj pozivaju prethodne dve.

const groceriesCount = 4;

const canClean = true;

function getGroceries() {

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

    console.log("Getting groceries...");

    setTimeout(() => {

      if (groceriesCount > 0) {

        resolve(["meat", "paprika", "carrot", "peper"]);

      } else {

        reject("No enough groceries!");

      }

    }, 2000);

  });

}

function cleanGroceries() {

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

console.log("Cleaning groceries...");

    setTimeout(() => {

      if (canClean) {

        resolve("Cleaned!!!");

      } else {

        reject("Groceries cannot be cleaned!");

      }

    }, 1000);

  });

}

function makeDinner() {

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

    getGroceries()

      .then((response) => {

        console.log("response 1", response);

        cleanGroceries()

          .then((res) => {

            console.log("response 2", res);

            resolve('Dinner is ready! Enjoy it!!!');

          })

          .catch((err) => {

            console.error("error 2", err);

            reject(err);

          });

      })

      .catch((error) => {

        console.error("error 1", error);

        reject(error);

      });

  });

}

makeDinner().then((res) => {

  console.log(res);

}).catch((error) => {

  console.error(error);

});

1. Ovde se vidi opcija finally koja se ubacuje kada ce se funkcija izvrsiti bez obzira da li je rezultat uslova tacan ili nije tacan.

const memoryFull = false;

function takePhoto() {

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

        console.log("taking photo");

        if(memoryFull) {

            reject("Memory full");

        } else {

            resolve("Photo added");

        }

    })

}

function editPhoto() {

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

        console.log("Editing photo:");

        const button = confirm("Do you want to edit this photo?");

        if(button) {

            resolve("Photo edited");

        } else {

            reject ("Photo won't be edited.");

        }

    })

}

function postingPhoto () {

    takePhoto()

     .then((response) => {

        console.log(response);

            editPhoto()

            .finally(() => {

                console.log("Photo finally added to Instagram");

            })

        })

        .catch((error) => {

            console.error(error);

    })

}

postingPhoto();

1. Laksi nacin da se uradi treca sabirna funkcija 3. primer

const isSelected = false;

function selectItemsFromMenu() {

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

    setTimeout(() => {

      if (isSelected) {

        resolve("Items selected!");

      } else {

        reject("Items not selected");

      }

    }, 2000);

  });

}

function confirmOrder () {

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

    const answer = confirm("Do you want to confirm order ?");

    if(answer) {

      resolve('Order confirmed!');

    } else {

      reject('Order not confirmed!');

    }

  })

}

function orderFood() {

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

        confirmOrder().then((res) => {

            console.log(res);

        })

        .catch((error) => {

            console.error("Order refused");

        });

    })  .catch((error) => {

        console.error("Items not selected");

    })

}

orderFood();

 jednostavniji nacin da se uradi ova treca funkcija :

async function orderFood2() {

try {

        await selectItemsFromMenu();

        await confirmOrder ();

        console.log ("%cOrder is ready!", "color:green; font-size:16px");

    }

    catch(error) {

        console.error("error", error);

    }

}

orderFood2()

Try - Catch blok - omogucava da se kod nastavi izvrsavati iako ova funkcija ne radi.

function mojaFunkcija() {

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

    reject("Neka error poruka");      // Asinhrona funkcija ima reject za error.

  });

}

mojaFunkcija().catch((err) => {

  console.log(err);

});

function mojaFunkcija2() {

  throw new Error("Ovo je novi error");// Sinhrona funkcija nema reject pa se error moze hendlovati na ovaj nacin, try/catch blok

  console.log(student.firstName);

}

// mojaFunkcija2();

try {                       // Sinhrona funkcija nema reject pa se error moze hendlovati na ovaj nacin i try, catch blok.

  mojaFunkcija2();

} catch (error) {

  console.log("Ovo ne radi !!!");

}

console.log('ovo opet se izvrsava');

- Dole objasnjenje ove funkcije u zagradama. -  **IIFE (Immediately Invoked Function Expression)**

(function onPageLoad () {

    addNavigationClickEvents();

    handleSearch();

})();

Ova funkcija je okružena IIFE (Immediately Invoked Function Expression) zagradama, što znači da će se odmah izvršiti kada se skripta učita, bez potrebe da se eksplicitno poziva negdje drugdje.

U ovom slučaju, IIFE je korišten kako bi se stvorio opseg izvršenja (scope) za funkciju **onPageLoad** i osiguralo da neka druga imena funkcija ili varijable unutar te funkcije ne budu globalne. Također, koristi se kako bi se izbjegla mogućnost kolizije imena sa drugim skriptama.

Osim toga, koristi se IIFE kako bi se naglasilo da je funkcija dio procesa inicijalizacije stranice i da će se izvršiti odmah nakon što se skripta učita. Ovo je čest pristup za izvršavanje inicijalizacijskog koda na stranici.

* Nauciti - <https://www.webprogramiranje.org/arrow-funkcija/> !!!!!!
* .elementSibling, .nextElementSibling, .previousElementSibling.