```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport"
            content="width=device-width,
             initial-scale=1.0">
   <link rel="stylesheet" type="text/css"
            href="styles/main.css">
   <script src="scripts/main.js"></script>
</head>
<body></body>
</html>

 <b>Bold <strong>text</strong></b>
<i>Italic <em>text</em></i>
<ins>underlined<del>crossed text</del></ins>
<time datetime=
"1969-08-20 20:17">20.srp '69., 20:17</time>
<figure><img src="slika.jpg" height="300px"
 alt="Opis">
<figcaption>Tekst ispod slike</figcaption></figure>
<ul style="list-style-type:square;">
  <li>G</li>
  <li style="list-style-type:circle;">K</li>
</ul>
```

▪ G
○ K

```
<ol type="1" start="2">
  <!-- 1,A,a,I -> nacin numeriranja -->
  <li>Java</li>
  <li>C</li>
</ol>
```

2. Java
3. C

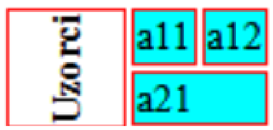
```
&euro;&copy;&gt;&lt;&amp;&quot;
&nbsp;
```

€©><&"

```
<dl>
  <dt>Naslov1</dt><dd>Opis1</dd>
  <dt>Naslov2</dt><dd>Opis2</dd>
</dl>
```

Naslov1
    Opis1
Naslov2
    Opis2

```
<table>
 <colgroup> <!-- uredivanje stila stupaca -->
     <col> <!-- preskoci prvi stupac -->
     <col span="2" style="background-color: cyan;">
    <!-- uredi iduca dva -->
 </colgroup>
 <tr>
   <th rowspan="2" style="transform:
rotate(-90deg)">
Uzorci</th>
     <td>a11</td>
     <td>a12</td>
 </tr>
 <tr>
   <td colspan="2">a21</td>
 </tr>
</table>
```



Uzorci	a11	a12
	a21	

```
<a href="http://www.fer.hr/index.html">fer</a>
 <a href="#b2" title="1. poglavlje">poglavlje</a>
 <h3 id="b2">Skoci na ovo poglavnje</h3>
 <hr /> <!-- horz crta-->
     margin: 1px;
     color: black;
     font-size: 2em;
     text-align: center;
   }
 <code></code>
 <pre>preformat</pre>
 <kbd></kbd>
 <samp></samp>
```

by Aux,Spike,Krampert

```
<form target="_self" action="/processForm.php" method="GET">
  <!-- target: self unutar istog, blank unutar novog -->

  <label>Korisnicko ime: <input type="text"
                                 name="username" value="enter your username"
                                 size="30"></label>
   <br/>
   <label>Lozinka: <input type="password"
                          name="password"
                          maxlength="30" required></label><br/>
   <label>Skriven input <input type="text" hidden readonly></label>

 <fieldset> <!-- radio-button -> izabir jednog -->
     <legend>Uloga</legend>
     <label><input type="radio" name="role"
                          value="admin" checked>Administrator</label><br/>
     <input type="radio" id="user" name="role" value="user">
     <label for="user">Korisnik</label></p>
  </fieldset>

  <fieldset> <!-- checkbox -> bilo koji broj -->
        <legend>Dodatne opcije</legend>
        <label><input type="checkbox"
                   name="stakla" value="da" checked>Stakla</label><br/>
        <label><input type="checkbox"
                   name="felge" value="da">Felge</label>
   </fieldset>
     <select name="padajuca_lista" size="1">
        <!-- multiple za selectanje vise -->
        <option value="opcija" selected>Opcija</option>
        <option value="opcija">Opcija2</option>
     </select><br/>
     <input type="date" name="trip_start"><br/>
     <label>Datoteka:
            <input type="file" name="datoteka"></label><br/>
            <!-- mora biti POST u method -->
     <textarea name="tijelo" rows="10"
                          cols="50">Inicijalan sadrzaj</textarea><br/>
     <input type="submit" value="Submit">
     <input type="reset" value="Odustani">
   </form>
   <!--
     greske kodiranja sadrzaja:
     ponovljeno ime elementa, atribut disabled, atribut nije definiran
     application/x-www-form-urlencoded -> cust=Pero+Peri%C4
     text/plain -> za developere -> cust=Pero address=Ulica
   -->
```

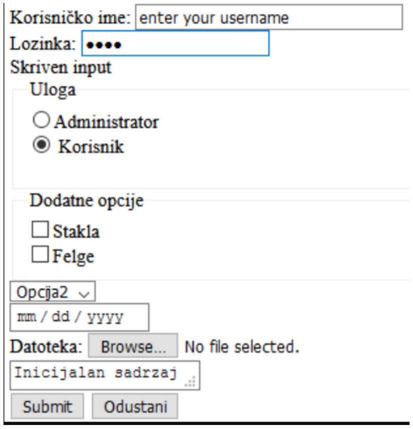
```
<ul class="flex-container">
   <li class="item">1</li>
   <li class="item">3</li>
</ul>
```

```
.flex-container {
     display: flex;
     flex-flow: column wrap;
    /*   nowrap (stisni sve elemente ako se smanji prostor)
       wrap   (kreni slagat elemente po redu odozgora prema dolje)
       wrap-reverse(odozdo prema gore ih slazi po redu)
         justify-content:flex-start;
    /*
         justify-content:flex-start(lijevo),flex-end(desno)
         center,space-betweeen,space-around,space-evenly po x-osi
         align-items: flex-start;
          align-items:flex-start(gore),flex-end(dolje),center(centar)
           /* align-content: space-around;
             align-content: flex-start(gore),
                flex-end(dolje),center,space-between,
                     space-around,space-evenly
                        skupi sve elemente i mice ih po y osi
```

```
<div class="grid-container">
  <div class="grid-item">1</div>
  <div class="grid-item">2</div>
  <div class="grid-item spoji">\</div>
  <div class="grid-item spoji ">/</div>
</div>

.grid-container{
   grid-template-columns: 10fr 5fr;
   justify-items: end;
   /* lijevo,centar,desno -> start,center,end */
   align-items: end;
   /* gore,sredina,dolje -> start,center,end,stretch */ }

.spoji{
     grid-column-start: 1;
     grid-column-end: 3;
     grid-row-start: 2;
     grid-row-end: 3;
   }
```



Korisničko ime: enter your username
Lozinka: ••••
Skriven input
Uloga
○ Administrator
● Korisnik
Dodatne opcije
□ Stakla
□ Felge
Opcija2
mm / dd / yyyy
Datoteka: Browse... No file selected.
Inicijalan sadrzaj
Submit Odustani

Jednostavni:
* => sve
h1, li => svi h1 i li
li.c1 => svi li s klasom c1
Atributni:
li[id="z2"] => svi li s id=z2

Kombinirani:
div span => svi span unutar div
div > span => neposredna djeca span roditelja div
div+span => svaki span koji ima zajednickog roditelja s nekim div
div~span => isto kao +, ali odabire samo nakon diva

Pseudoklasa:
div:hover => na hover miša
li:first-child => li koji je prvo dijete roditelja
input:required => svi required inputi

Pseudoelementi:
p::before, p::after {content: "\"";} => ubaci " prije i poslije paragrafa
p::first-letter
p::first-line
p::selection => dio elementa koji je odabrao korisnik
inline => 1000
#id => 100
.class, :pseudo-class, [attribute] => 10
<TAG>, ::pseudo-element => 1
<body>, * => 0
border: inherit => uzmi od roditelja
border: initial => iskljuci sve, uzmi od browsera
border: unset => inherit ako ima matching value, initial inace

neki css elementi:
font-family, font-weight:bold, font-size, font-style:italic, text-decoration:underline/none
text-align, text-indent, letter-spacing, line-height
background-image: url('./images/x.jpg')
background-repeat: no-repeat, repeat
box-sizing: border-box
display: block/inline/inline-block
padding, border, margin
margin:auto => centriranje
max-width, width
position: relative/absolute/fixed

Važniji globalni atributi:id,class,lang,title,style

```
var x = 123e-5; let y = "string";
const z = 'string'; // const
//operatori dodaj Aux
let exp = 2**3 // 2^3 = 8
// strings
let s = "he said: \"xd\"";
let len = s.length;
let index = s.indexOf("xd");  // .lastIndexOf()
let newString = s.slice(1,3);
let numToString = (123).toString();
let stringToNum = Number(numToString);
// functions
function f1(x="default value") {
        console.log(x);
}
let f2 = function(x) {console.log(x);}
let f3 = (x) => console.log(x);
// arrays
let arr = [1,2,3,5,6,1,7,8];
let last = arr.pop(); // makne i returna zadnji
arr.push(4);  // append na kraj
let first = arr.shift(); // makne i returna prvi
arr.unshift(5); // ubaci na pocetak
arr.splice(1,2); // brise 2 elementa od indeksa 1
let arr2 = [6, 7];
let arr3 = arr.concat(arr); // spoji
arr4 = arr3.slice(1,3);
console.log(arr4.includes(6));
arr.sort( function(a,b){return b - a} );  // sort sa komparatorom (desc)
arr.reverse();
arrEven = arr.filter( (x) => x%2==0);
for (let element of arr) {console.log(element);} // elementi
for (let index in arr) {console.log(index);}    // indeksi
// classes, objects
class Person {
    lastName = "Doe";
    age = 50;
    constructor(firstNameValue) {this.firstName = firstNameValue;}
    get lastName() {return this.lastName;}
    set lastName(newLastName) {this.lastName = newLastName}
}
let person1 = new Person("John");
person1.lastName = "Williams";
person1.firstName = "Jake";
for (let value of Object.values(person1)) {console.log(value)}
// try-catch
try {
        throw "error";
} catch(err) {
        console.log(err);
} finally {
        var x = 2;
}
```

```
setTimeout( () => console.log(3) , 3000 ); // nakon 3s
let promise = new Promise( (resolve, reject) => {
        setTimeout( () => {
                console.log("nakon 3 sekunde...");
                if (false) {
                        resolve("dobro izvrsen");
                } else {
                        reject("lose izvrsen");
                        // moze i npr reject(new Error("lose izvrsen")) }
        },
        3000);
});
promise.then(
        function(result) {console.log(result);},
                // ako resolve, onda se ovo izvrsava, result = "dobro izvrsen"
        function(error) {console.log(error);}
                // ako reject, onda se ovo izvrsava, result = "lose izvrsen"
);
promise.catch(
        function(error) {console.log(error);}   // samo ako reject
)
promise.catch(
        function(error){console.log(error);}
).then(
        function(result){console.log("Resolve:"+result)},
        function(result){console.log("Reject:"+result)}
);      // catch ce uhvatit error, u then se ce pozvat prva funkcija s result = undefined
let promise2 = fetch("https://web1lab2.azurewebsites.net/products?categoryId=1");
promise2.then(      // obraduje se promise od fetcha
        function(response) {
                if (!response.ok) { throw new Error("Cannot load"); }
                else { return response.json(); } // novo obecanje reponse.json()
        },
        function(error) { throw error; }
).then(             // obraduje se promise od response.json
        function(response) { console.log("Loaded JSON"); }
).catch(            // catch hvata gresku u bilo kojem promiseu
        function(error) { console.log(error); }
)
async function LoadJSON() {     // funkcija se izvodi asinkrono
    let promise = await fetch("https://web1lab2.azurewebsites.net/products?categoryId=1");
       // unutar funkcije, await se izvodi sinkrono (ostatak funkcije ceka)
    if (!promise.ok) { throw new Error ("Cannot load"); }
    else { var jsonContents = await promise.json(); }
        console.log(jsonContents);
}
LoadJSON().catch(
        (error) => {console.log(error);}
)
```

```
let html = document.documentElement;
let body = document.body;
let head = document.head;
let bodyChildrenElements = body.children; // samo html elementi
let bodyChildren = body.childNodes;
 // sadrzi i tekst, npr "aa bb" je jedan clan, te svaki \n
let h1 = body.firstElementChild;
   // samo html elementi, .firstChild za sve
let script = body.lastElementChild;
  // samo html elementi, .lastChild za sve
let p = h1.nextElementSibling;
h1 = p.previousElementSibling;
body = h1.parentElement;
p = document.getElementById("i1");
h1 = document.querySelector(".c2"); // css selektori
p_h1 = document.querySelectorAll(".c2");
p.style.backgroundColor = "Red";
p.innerHTML = "promjena teksta";
```

```
<body>
   aa
   bb
   <h1 class="c2"> Naslov </h1>
   <p id="i1" class="c2"> Paragraf </p>
</body>
```

```
let newP = document.createElement("p");
newP.innerHTML = "novi paragraf";
body.appendChild(newP);
alert("hey");
let answer = prompt("name?", "default");
let response = confirm("cookies?");
// JSON
let person = {firstName:"John"
          , lastName:"Doe", age:50, eyeColor:"blue"};
let personJSON = JSON.stringify(person);
let personFromJSON = JSON.parse(personJSON);
```

```
// local storage
let ls = window.localStorage;
ls.setItem("item", 2);
let keyAtIndex0 = ls.key(0);
let length = ls.length;
let item = ls.getItem("item");
ls.removeItem("item");
ls.clear();
```