UNIVERZITET U NOVOM SADU, FAKULTET TEHNIČKIH NAUKA
DEPARTMAN ZA GRAFIČKO INŽENJERSTVO I DIZAJN

WEB DIZAJN

predavanje XI – vektorska grafika (canvas, SVG, Google map, ikonice), microdata, API, dobra praksa – pristupačnost

<canvas>

- element **<canvas>** predstavlja kontejner (prazno "platno") na veb stranici **za crtanje grafike** („on the fly") uz pomoć **JS**
- <canvas> nema mogućnosti generisanja grafike sam po sebi, već je samo „placeholder" za grafiku koja se zadaje skriptom
- <canvas> podržavaju najnovije verzije svih čitača

mogućnosti kanvasa

- crtanje (ispisivanje) **teksta**
- crtanje **grafika**
- **animiranje** objekata kanvasa
- **interaktivnost** kanvasa

korak 1: povezivanje kanvasa sa JS

- **id** vrednost se koristi za **povezivanje** kanvasa sa **JS**
- **primer:**
- definisanje kanvasa određene visine i širine i oivičenog:

```
<canvas id=„crtez" width="200" height ="200"
style="border:1px solid #000000;"></canvas>
```

- definisanje JS promenljive od kanvasa pomoću HTML DOM metoda **getElementById():**

```
var c = document.getElementById(„crtez");
```

korak 2: kreiranje objekta za crtanje

- metod getContext() vraća ugrađen HTML objekat za crtanje grafike po kanvasu:

```
var ctx = c.getContext("2d");
```

korak 3: crtanje po kanvasu - svojstva

- crtanje po kanvasu omogućavaju **svojstva i metode** getContext() objekta
- **svojsta** getContext() objekta u vezi sa bojom, stilom i senkama
  - **fillStyle** – definiše stil, gradijent ili patern popune
  - **strokeStyle** – definiše stil, gradijent ili patern okvirne linije
  - **shadowColor**, **shadowBlur**, **shadowOffsetX**, **shadowOffsetY** – definišu, respektivno, boju, stepen „zamućenja“, horizontalni i vertikalni pomeraj senke
- pored ovih, postoje i svojsvta za definisanje teksta, slika itd.

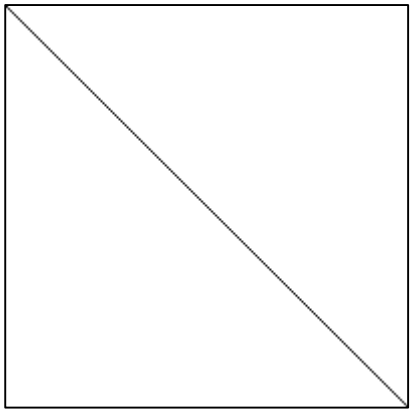
korak 3: crtanje po kanvasu

- metode za crtanje **putanje**:
  - **fill()** – popuna putanje
  - **stroke()** – iscrtava definisanu putanju
  - **beginPath()** – započinje putanju ili resetuje trenutnu putanjzu
  - **closePath()** – zatvara putanju od trenutne tačke do početne
  - **clip()** – isecanje regiona kanvasa
  - **moveTo()** – pomera putanju na određenu tačku bez kreiranja linije
  - **lineTo()** – dodaje novu tačku i kreira liniju do te tačke od prethodne
  - **arc()** i **arcTo()** – kreira, respektivno, kružnicu i luk između 2 tangente
  - **bezierCurveTo(), quadraticCurveTo()** – kreira, respektivno, kubične i kvadratične Bezijerove krive

primer JS za iscrtavanje linije

- **moveTo(x,y)** – početna tačka
- **lineTo(x,y)** – krajnja tačka

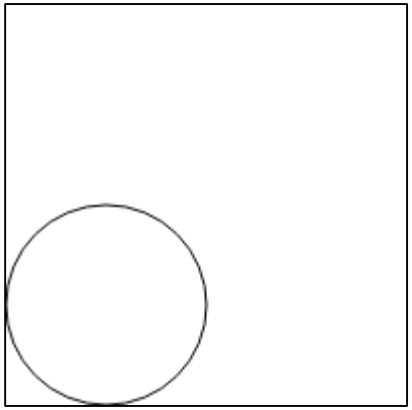
```
<script>
var c = document.getElementById(„crtez");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,200);
ctx.stroke();
</script>
```



primer JS za iscrtavanje kružnice

- **arc(x,y,r,startangle,endangle)**

```
<script>
var c = document.getElementById(„crtez");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(50,150,50,0,2*Math.PI);
ctx.stroke();
</script>
```



primer JS za ispisivanje teksta

- **fillText()** – crta „popunjen“ (obojen) tekst
- **strokeText()** – crta tekst bez popune sa okvirnom linijom

```
<script>
var c = document.getElementById(„crtez");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.strokeText("Hello World",15,100);
</script>
```

primer JS za učitavanje slike – način I

- **drawImage()** metod

```
<img id="slika" src="img_the_scream.jpg" width="220" height="277">
<canvas id="crtez" width="300" height="300"
style="border:1px solid #d3d3d3;" onclick="myCanvas()"></canvas>
<script>
function myCanvas() {
    var c = document.getElementById("crtez");
    var ctx = c.getContext("2d");
    var img = document.getElementById("slika");
    ctx.drawImage(img,10,10);
}
</script>
```

primer JS za učitavanje slike – način II

- slika mora da se **prvo učita** pre poziva drawImage() metoda

```
<script>
function myCanvas() {
   var c = document.getElementById("myCanvas");
   var ctx = c.getContext("2d");
   slika = new Image();
   slika.src = "img_the_scream.jpg" ;
   slika.onload = function(){
      ctx.drawImage(slika, 10, 10);    }
}
</script>
```

primer JS za iscrtavanje pravougaonika

- **fillRect()** – popunjen**, strokeRect()** – bez popune
- **fillRect(x, y, width, height)** – gde su x i y koordinate gornjeg levog temena

```
<script>
var canvas = document.getElementById("myCanvas");
var ctx = canvas.getContext("2d");
ctx.fillRect(0,0,100,50);
ctx.fillStyle = "#FF0000";
</script>
```

SVG (Scalable Vector Graphics)

- element **<svg>** se koristi za definisanje **vektorsk**e grafike
- vektorska grafika se definiše u **XML** formatu
- SVG grafika **ne gubi na kvalitetu** sa skaliranjem (zumiranjem)
- **svaki element** i **atribut** unutar <svg> se može **animirati**

SVG primer

```
<svg width="100" height="100">
   <circle cx="50" cy="50" r="40" stroke="red"
stroke-width="10" fill="yellow" />
   Sorry, your browser does not support inline SVG.
</svg>
```

** Pošto je SVG pisan u XML formatu,
svi elementi moraju biti pravilno zatvoreni.

prednost SVG u odnosu na druge formate grafike (JPEG i GIF)

- SVG slike mogu biti kreirane i editovane u bilo kojem **tekst editoru**
- SVG slike se mogu **pretraživati**, **indeksirati**, **povezati sa skriptom** i **kompesovati**
- SVG slike su **skalabilne** i mogu se zumirati bez degradacije kvaliteta
- SVG slike mogu biti odštampane u visokom kvalitetu na pri bilo kojoj rezoluciji
- SVG je **otvoren standard** (**XML**) i kompatibian je sa drugim W3C standardima (DOM i XSL) (za razliku od Flash formata)

alati za kreiranje SVG

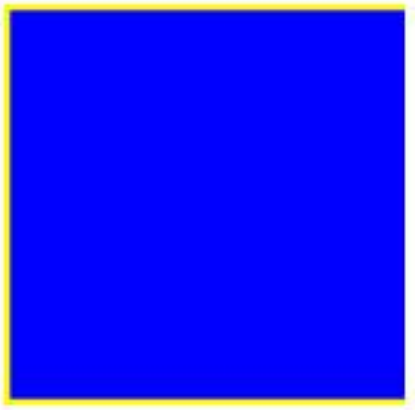
- **tekst editor**: npr. Notepad++
- **grafički editor:** Inkscape (besplatan), Adobe Illustrator (komercijalni)…

SVG elementi za predefinisane oblike (shapes)

- Rectangle <rect>
- Circle <circle>
- Ellipse <ellipse>
- Line <line>
- Polyline <polyline>
- Polygon <polygon>
- Path <path>

Rectangle <rect>

<rect **x**="50" **y**="20“ **width**="100" **height**="100„ **fill**="blue" **stroke-width**="3" **stroke**="yellow" **stroke-opacity**="0.9">



*kraće pisanje (grupisano sa style):

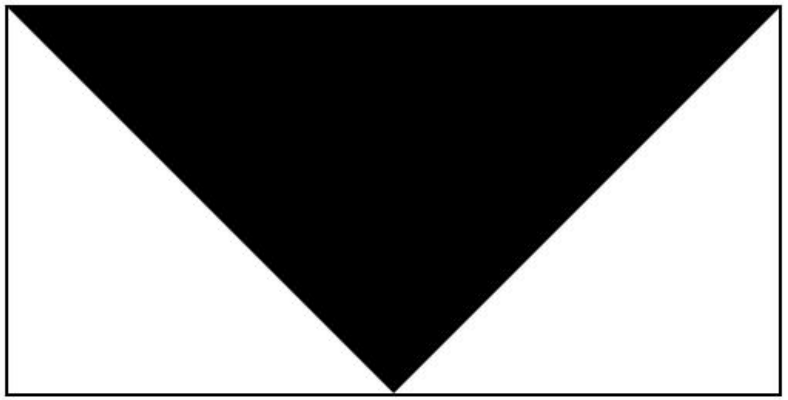
<rect **x**="50" **y**="20“ **width**="100" **height**="100" **style**="**fill:**blue;**stroke-width**:3;**stroke**:yellow; **stroke-opacity:**0.9">

SVG putanja <path>

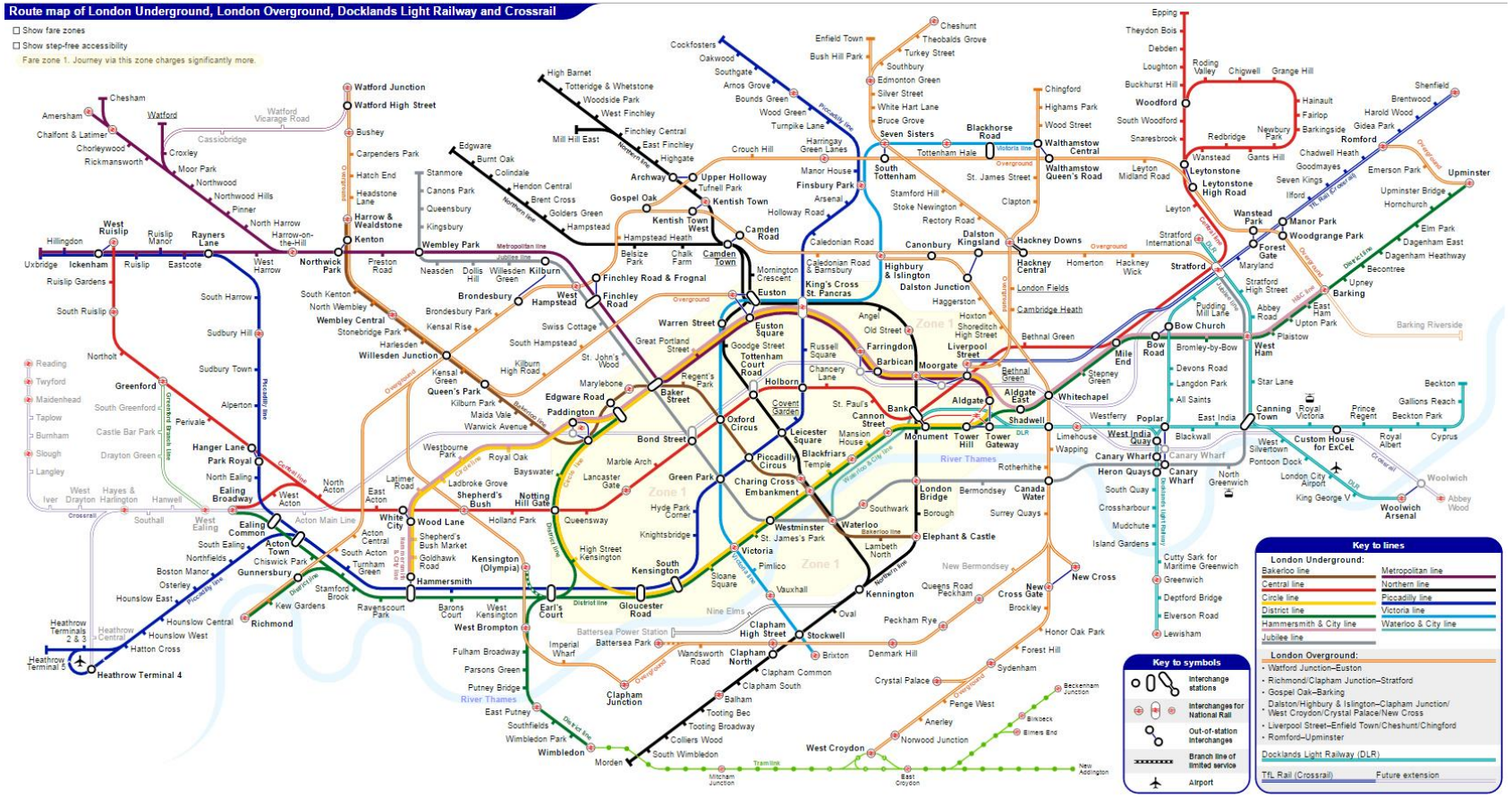
- komande za kreiranje putanje:
  - M = moveto
  - L = lineto
  - H = horizontal lineto
  - V = vertical lineto
  - C = curveto
  - S = smooth curveto
  - Q = quadratic Bézier curve
  - T = smooth quadratic Bézier curveto
  - A = elliptical Arc
  - Z = closepath
- ako su komande napisane **velikim slovima** to podrazumeva **apsolutno** pozicioniranje tačaka, a **mala slova** komandi označavaju **relativno** pozicioniranje tački

SVG putanja <path>

```
<svg height="100" width="200" style="border:1px solid black">
  <path d="M0 0 L200 0 L100 100 Z" />
  Sorry, your browser does not support inline SVG.
</svg>
```



kompleksniji SVG



kompleksniji SVG

linija.html | London_Underground_Overground_DLR_Crossrail_map.svg

```
<use xlink:href="#st" id="stbakerloo" class="sbakerloo"/>
<use xlink:href="#st" id="stcentral" class="scentral"/>
<g id="stcircle">
    <use xlink:href="#stb" class="meb bcircle"/>
    <use xlink:href="#st" class="scircle"/>
</g>
<use xlink:href="#st" id="stdistrict" class="sdistrict"/>
<g id="sthnc">
    <use xlink:href="#stb" class="meb bhnc"/>
<use xlink:href="#st" class="shnc"/>
</g>
<use xlink:href="#st" id="stjubilee" class="sjubilee"/>
<use xlink:href="#st" id="stmetropolitan" class="smetropolitan"/>
<use xlink:href="#st" id="stnorthern" class="snorthern"/>
<use xlink:href="#st" id="stpiccadilly" class="spiccadilly"/>
<use xlink:href="#st" id="stvictoria" class="svictoria"/>
<use xlink:href="#st" id="stdlr" class="sdlr"/>
<use xlink:href="#st" id="stog" class="sog"/>
<use xlink:href="#st" id="stcrossrail" class="stflrail"/>
<use xlink:href="#st" id="stgreenford" class="sgreenford"/>
<path id="stx" style="fill:none;stroke-width:5" d="M 8,0 L 2,0"/>
<path id="stxm" style="fill:none;stroke:#fff;stroke-width:3.5" d="M 7.25,0 L 0,0"/>
<path id="stxm2" style="fill:none;stroke:#fff;stroke-width:1.5" d="M 6.25,0 L 0,0"/>
<g id="stmetropolitanx">
    <use xlink:href="#stx" class="smetropolitan"/>
    <use xlink:href="#stxm"/>
</g>
<g id="stnorthernx">
    <use xlink:href="#stx" class="snorthern"/>
    <use xlink:href="#stxm"/>
</g>
<g id="stogx">
    <use xlink:href="#stx" class="sog"/>
    <use xlink:href="#stxm"/>
</g>
<g id="stogx2">
    <use xlink:href="#stx" class="sog"/>
    <use xlink:href="#stxm2"/>
</g>
<g id="stcrossrailx">
    <use xlink:href="#stx" class="scrossrail"/>
    <use xlink:href="#stxm"/>
</g>
<g id="stgreenfordx">
    <use xlink:href="#stx" class="sgreenford"/>
    <use xlink:href="#stxm"/>
</g>
<circle id="sttl" cx="0" cy="0" r="4" class="ftl"/>
<g id="sttlnr">
    <circle cx="0" cy="0" r="5.5" style="fill:#fff;stroke:#000;stroke-width:0.75"/>
    <path style="fill:none;stroke:#ef2721;stroke-width:1" d="
    M -5,-1.25 H 5
    M -5,1.25 H 5
```

ikonice

- za dodavanje ikonica na veb stranici se koriste inline elementi (najčešće **** i **<i>)** kojima se dodeljuje klasa koja određuje **klasu ikonica**
- najčešće korišćene biblioteke ikonica su:
  1. **Font Awesome Icons**
  2. **Bootstrap Icons**
  3. **Google Icons**

Font Awesome Icons

- za korišćenje **Font Awesome** ikonica potrebno je povezati se u <head> sekciji sa **Font Awesome bibliotekom**:

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

- primer:

```
<i class="fa fa-cloud"></i>
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>
```

Font Awesome Icons

- ikonice predstavljaju **skalabilne vektore** koji se mogu menjati **CSS-**om (boja, veličina, senka itd.)
- primer:

```
<i class="fa fa-cloud" style="font-size:24px;"></i>
<i class="fa fa-cloud" style="font-size:36px;"></i>
<i class="fa fa-cloud" style="font-size:48px;color:red;"></i>
<i class="fa fa-cloud" style="font-size:60px;
color:lightblue;text-shadow:2px 2px 4px #000000;"></i>
```

Font Awesome Icons

- ikonice društvenih mreža:

```
<a href="#"><i class="fa fa-facebook-official"></i></a>
<a href="#"><i class="fa fa-pinterest-p"></i></a>
<a href="#"><i class="fa fa-twitter"></i></a>
<a href="#"><i class="fa fa-flickr"></i></a>
<a href="#"><i class="fa fa-linkedin"></i></a>
```

image sprite

- image sprite (sprajt) – kolekcija slika sačuvanih kao jedna slika
- **zašto se koristi?**
- veb strana koja ima veliki broj slika može zahtevati duže vreme za učitavanje i generiše više serverskih zahteva >> upotreba sprajta redukuje broj serverskih zahteva i čuva propusni opseg (bandwidth)
- primer: 

image filter

- primer:

```
img {
filter: blur(5px); filter: grayscale(100%);
}
img:hover {
filter: none;
}
```

- vrste filtera: **none| blur() | brightness() | contrast() | drop-shadow() | grayscale() | hue-rotate() | invert() | opacity() | saturate() | sepia() | url()**

image filter

Google Maps API

- API (**A**pplication **P**rogramming **I**nterface) predstavlja **set metoda i alata** koji se mogu koristiti **za kreiranje softverskih aplikacija**
- Google Maps API dozvoljava kreiranje mape na sajtu

Google Map API - primer

```
<div id="map" style="width:100%;height:500px"></div>
<script>
function myMap() {
  var mapCanvas = document.getElementById("map");
  var mapOptions = {
    center: new google.maps.LatLng(45.25, 19.83),
    zoom: 10 }
  var map = new google.maps.Map(mapCanvas, mapOptions);
}
</script>
<script src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBu-916DdpKAjTmJNIgngS6HL_kDIKU0aU&callback=myMap"></script>
```

korak 1: veza sa Google Maps API

- Google Maps API je **JS biblioteka** koja se povezuje sa veb stranicom pomoću sledećeg <script> taga:

  <script src="**https://maps.googleapis.com/maps/api/js?callback=myMap**"></script>

- parametar **callback** definiše funkciju koja će se pozvati (u ovom slučaju **myMap**) kada je API spreman

korak 2: kreiranje kontejnera mape

- Google mapa se smešta u predviđeni HTML element i nasleđuje automatski njegovu veličinu:

```
<div id="map" style="width:100%;height:500px"></div>
```

korak 3: crtanje mape tj. definicija funkcije (primer - Novi Sad)

```
<script>
function myMap() {
  var mapCanvas = document.getElementById("map");
  var mapOptions = {
    center: new google.maps.LatLng(45.25, 19.83),
    zoom: 10
  }
var map = new google.maps.Map(mapCanvas, mapOptions);
}
</script>
```

korak 3: crtanje mape tj. definicija funkcije (primer - Novi Sad)

- **myMap** funkcija **inicijalizuje** i **prikazuje** mapu
- svojstvo **center** definiše centar mape pomoću **LatLng** objekta koje se prosleđuju koordinate u sledećem redosledu: latituda, longituda.
- svojstvo **zoom** definiše zoom nivo mape – veći zoom nivo podrazumeva veću rezoluciju (manju razmeru) i obrnuto
- kod **new google.maps.Map()** kreira novi Google Maps objekat

vrste mapa (primer - Novi Sad)

1. ROADMAP, 2. SATELLITE, 3. HYBRID, 4. TERRAIN

vrste mape (primer - Novi Sad)

```
<div id="googleMap1"
style="width:400px;height:300px;"></div>
<script>
function myMap() {
var mapOptions1 = {
    center: new google.maps.LatLng(45.25, 19.83),
    zoom:11,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  };
 var map1 = new
google.maps.Map(document.getElementById("googleMa
p1"),mapOptions1);
```

HTML APIs

- **Geolocation**
- **Drag/Drop**
- **Local Storage**
- **App Cashe**
- **Web workers**
- **SSE**

geolocation API

- **geolocation API** se koristi za dobijanje podataka o geografskoj poziciji korisnika
- s obzirom da se radi o osetljivoj informaciji, ovaj podatak je dostupan samo ako korisnik to odobri
- najtačniji podaci se dobijaju kod uređaja koji imaju **GPS**
- ne podržavaju svi čitači ovaj API (npr. od Chrome 50 verzija geolokacija je dostupna samo uz HTTPS protokol)

geolocation API

```
<p id="demo"></p>
<script>
var x = document.getElementById("demo");
function getLocation() {
    if (navigator.geolocation) {
        navigator.geolocation.getCurrentPosition(showPosition);  }}
function showPosition(position) {
    x.innerHTML = "Latitude: " + position.coords.latitude +
    "Longitude: " + position.coords.longitude;}
</script>
```

- Rezultat (FTN):

Latitude: 45.2671352 Longitude: 19.833549599999998

Drag/Drop

- Drag/Drog API omoguća prevlačenje elemenata na drugu lokaciju
- u HTML5 standardu **svaki element** se pože prevući (postati draggable): <img **draggable="true">**

Local Storage

- Local Storage API omogućava veb aplikacijama da skladište podatke lokalno u okviru čitača korisnika
- pre HTML5 standarda, aplikacije su skladištile podatke u „kolačićima“ (cookies) koji su se uključivali u svaki serverski zahtev
- u odnosu na „kolačiće“, lokalno skladištenje je **sigurnije** (informacije se nikad ne šalju serveru) i može da **čuva veću količinu** podataka lokalno (preko 5MB) bez narušavanja izvršavanja (performantosti) veb aplikacije

Local Storage

- objekti za lokalno čuvanje podataka

1. window.**localStorage** – čuva podatke bez roka isteka
2. window.**sessionStorage** - čuva podatke za jednu sesiju (podaci se izgube kada se tab čitača zatvori)

Local Storage

- localStorage.**setItem**(ime, vrednost);
- localStorage.**getItem**(ime);
- primer:

```
<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
    // Store
localStorage.setItem("lastname", "Petrovic");
    // Retrieve
 document.getElementById("result").innerHTML =
localStorage.getItem("lastname");
}
</script>
```

Application Cache

- Application Cache API – omogućava keširanje veb aplikacije i pristup aplikaciji bez internet konekcije
- **prednosti:**

1. **offline browsing**
2. **brzina** (keširani resursi se učitavaju brže)
3. **smanjuje opterećenje servera**

Web Worker

- Web Worker – omogućava izvršavanje JS skripti u pozadini bez uticaja na performantnost stranice
- **prednost:** bez Web Workera, kada traje izvršavanje skripti na veb stranici, strana je „nedostupna“ (ne mogu se elementi strane selektovati, kliknuti...) dok se skripta ne završi

SSE (Server-Sent Events)

- Server-Sent Events – omogućava da veb stranica dobije **automatski** ažuriranja (update) od servera (one-way messaging)
  - ranije je ovo bilo moguće ali ne automatski već je veb stranica trebala da „pita“ da li postoje neka ažuriranja
- **primer upotrebe:** Facebook/Twitter updates, ažuriranja cena akcija/deonica/deviznih kursvea, nove vesti (news feeds), sportski rezulrati

microdata

- dodavanje mikropodataka na veb stranice pomaže **pretraživačima** da bolje „razumeju“ sadržaj stranica odnosno glavna uloga mikropodataka je SEO (Search Engine Optimization)
- mikropodaci nisu vidljivi krajnjim korisnicima- oni nose čistu semantičku informaciju
- popularne vrste mikropodataka su događaji, profili korisnika, opisi organizacije, opisi proizvoda, geolokacija, detalji recepta itd.

primer mikropodataka - HTML

```
<div itemscope itemtype="http://schema.org/Person">
   Moje ime je <span itemprop="name">Petar Petorović</span> i ja sam trenutno zaposlen kao <span itemprop="jobTitle"> front-end veb dizajner </span>u kompaniji <a href="http://www.grid.uns.ac.rs" itemprop="affiliation">GRID</a>.
Moj email je: <span itemprop="email">ppetrovic@gmail.com</span>.
 Mesto prebivanja je <span itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
    <span itemprop="addressLocality">Novi Sad</span>,
    <span itemprop="addressRegion">Vojvodina</span>.
</div>
```

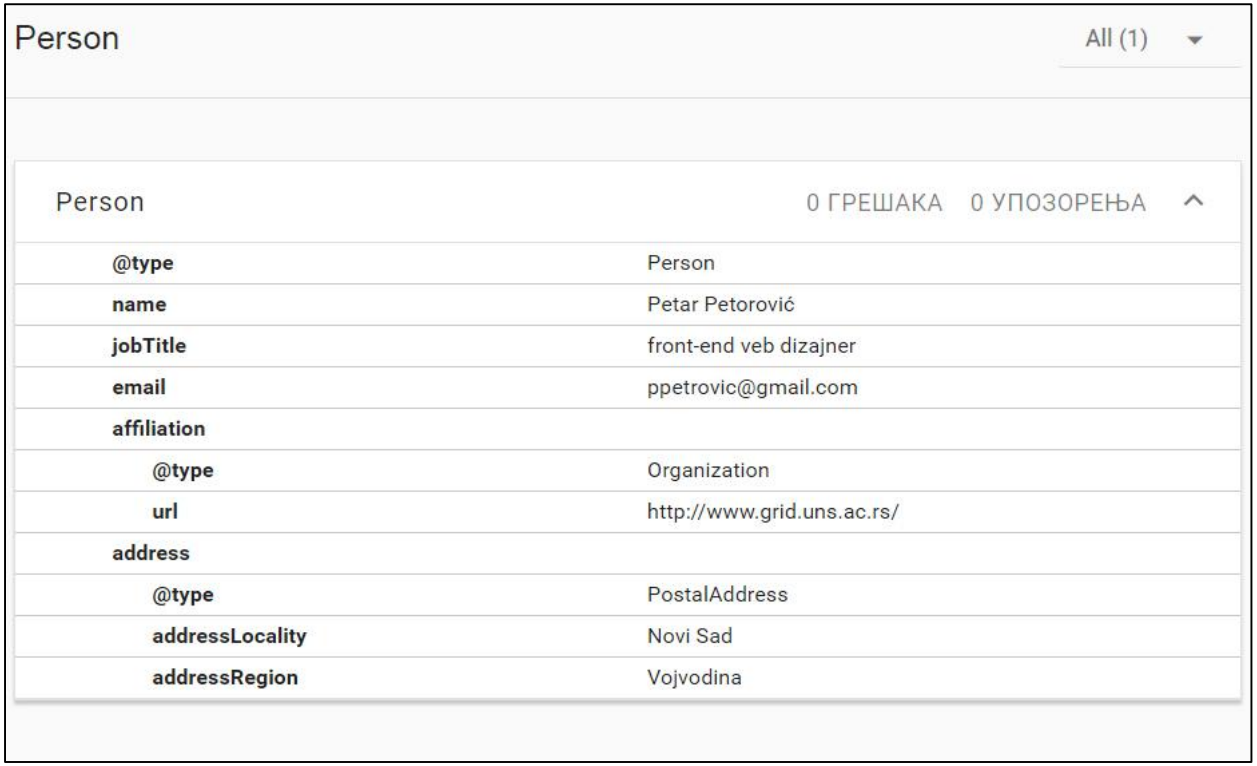
primer mikropodataka - HTML

- izgled u čitaču:

  Moje ime je Petar Petorović i ja sam trenutno zaposlen kao front-end veb dizajner u kompaniji GRID. Moj email je: ppetrovic@gmail.com. Mesto prebivanja je Novi Sad, Vojvodina.

primer mikropodataka - HTML

- kako vidi stranicu Google - https://search.google.com/structured-data/testing-tool/u/0/

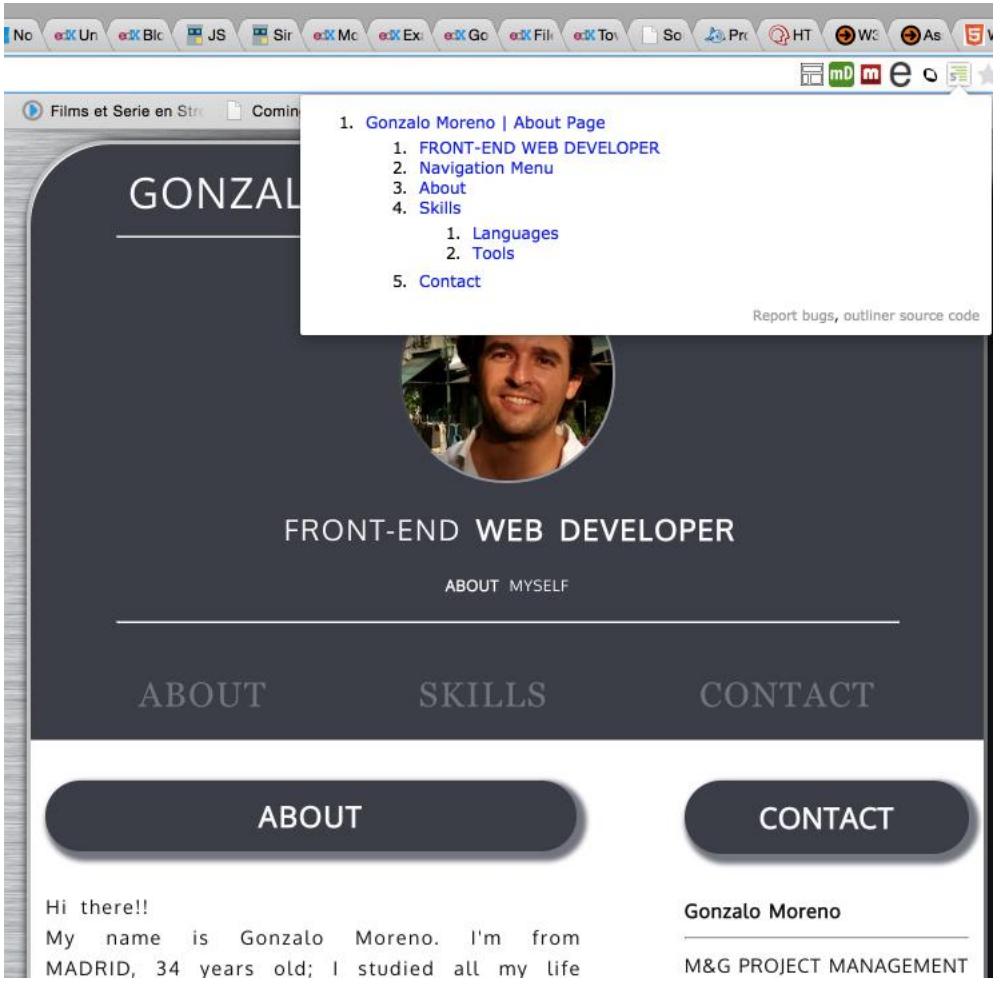


mogućnosti mikropodataka

- mikropodaci mogu biti procesirani, organizovani ili prezentovani u određenom kontekstu
- čitač ili ekstenzija čitača može interpretirati podatke npr. adresu i predložiti da se pošalje aplikaciji mape ili recimo za mikropodatke događaja (event) čitač može otvoriti popup prozor sa aplikacijom kalendara itd.
- određeni JS kod može pristupiti ovim podacima

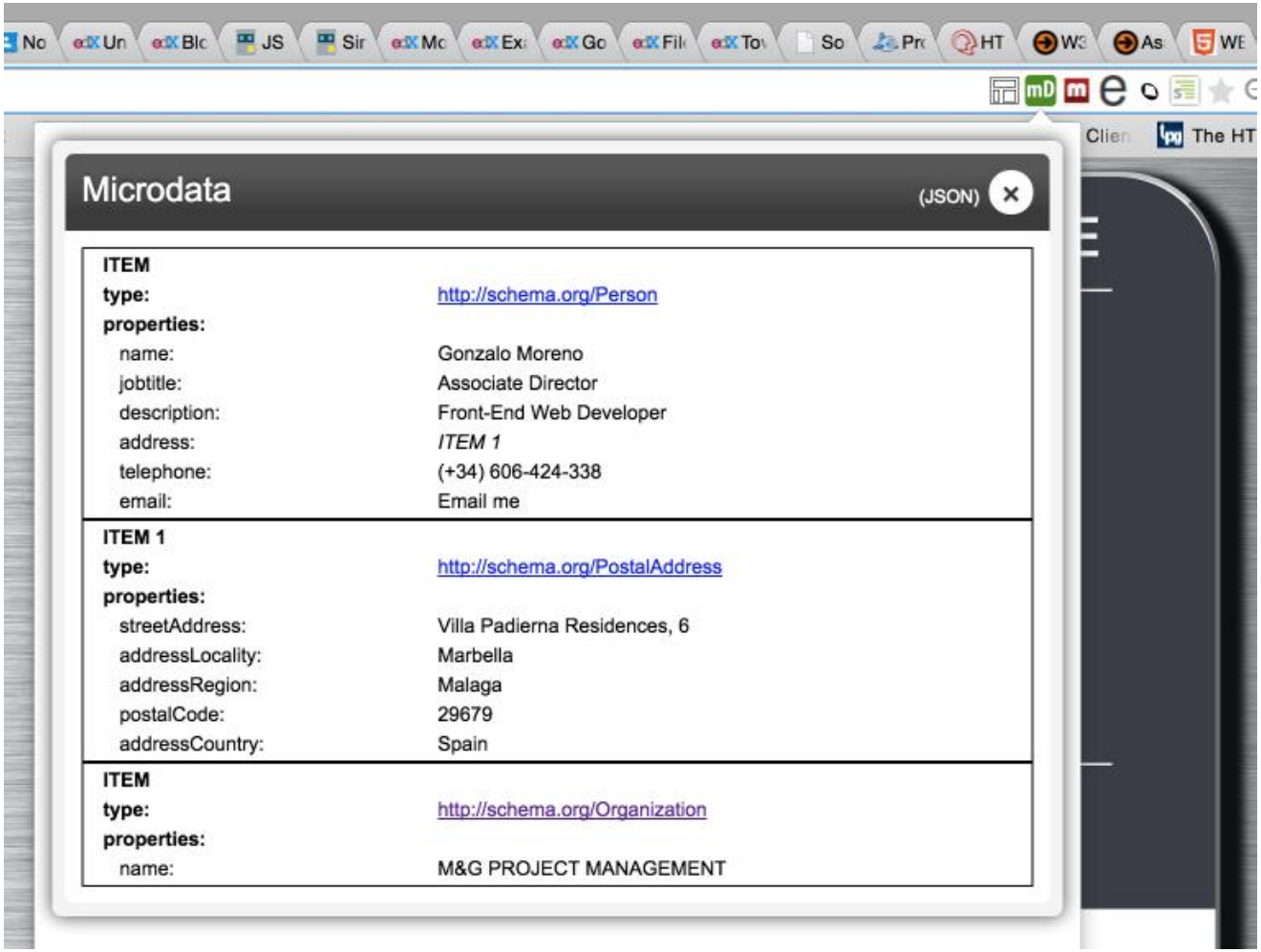
primer dobro optimizovane stranice

- struktura:



primer dobro optimizovane stranice

- microdata:



validatori koda

- **W3C VALIDATOR** proverava validnost (ispravnost) HTML (kao i XML) sintakse
- **CSS VALIDATOR** proverava Cascading Style Sheets (CSS) i (X)HTML dokumente koji koriste CSS stilove
- **UNICORN** je W3C objedinjeni validator koji pomaže veb programerima da se popravi kvalitet veb stranica izvođenje raznih provera i testova (sakuplja rezultate popularnih HTML i CSS validatora i ostalih servisa poput Internationalization ili RSS fida)
- **LINK CHECKER** provera hiperlinkove i referencirane objekte na pojedinačnoj veb stranici, CSS eksternom stilu ili kompletnom sajtu

pristupačnost

- atributi **lang** za definisanje jezika, **alt** za alternativni tekst (slike, video)
- vizuelna i strukturna hijerarhija pomoću naslova (h1 do h6)
- mogućnost menjanja veličine fonta bez narušavanja izgleda