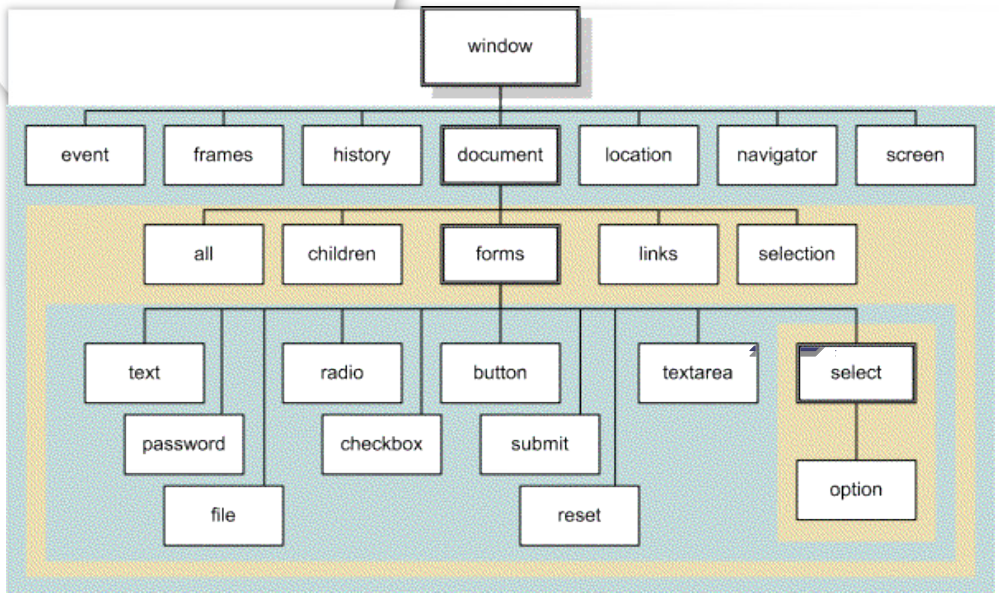
Univerza v Ljubljani
Fakulteta za računalništvo
in informatiko



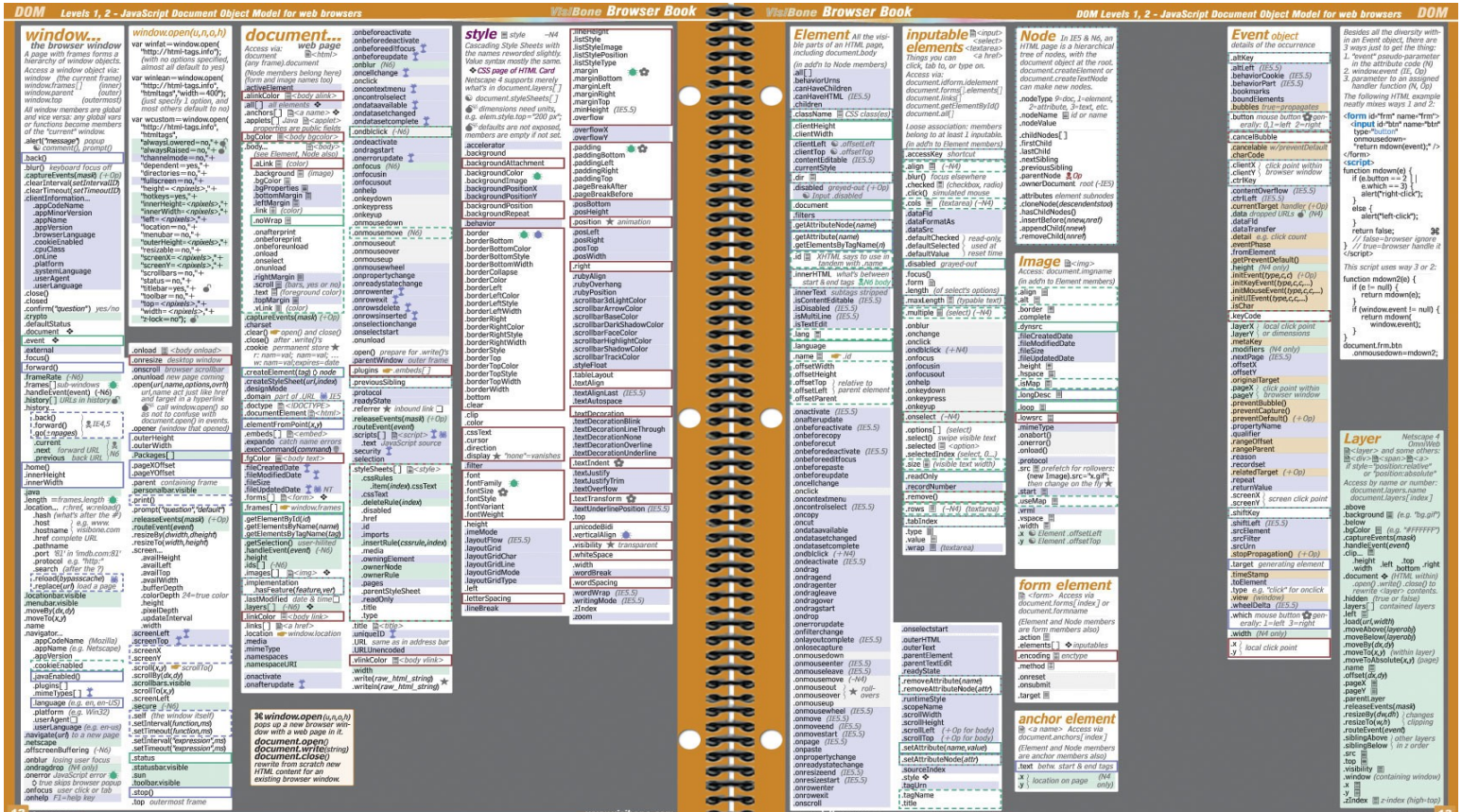
JavaScript in DOM

JavaScript

- JavaScript kot programski jezik
  - sintaksa, funkcije, operatorji, stavki, objekti
  - polja, regularni izrazi
  - integracija z brskalnikom
- komunikacija z vsebino dokumenta (DOM)
  - objekti brskalnika, naslavljanje
  - dogodki, registracija dogodkov
  - programsko spreminjanje dokumenta
- dinamične spletne strani
  - premikanje elementov, vidnost, barve, pisave, prekrivanje
  - animacija

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Dokumentni objektni model (DOM)



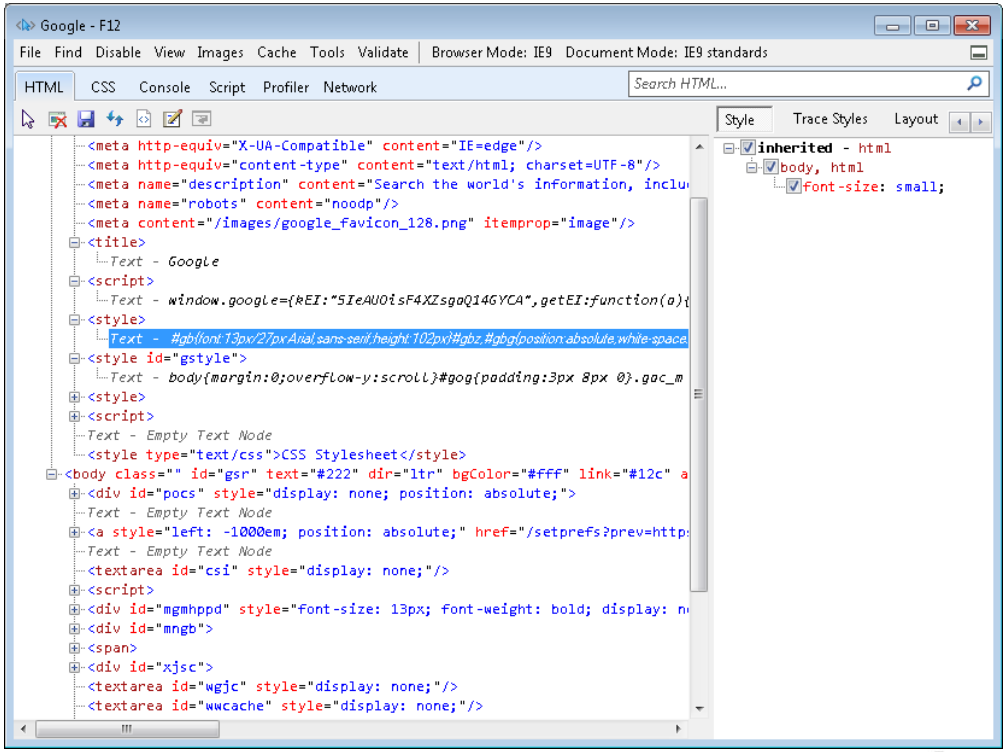
Dokumentni objektni model (DOM)

- DOM - vmesnik med dokumenti in programom v jeziku JavaScript
- so brskalnikov "pogled" na dokument
- verzije:
  - DOM 0 (delno dokumentiran v HTML 4 specifikaciji)
  - DOM 1 (1998, usmerjen na XHTML)
  - DOM 2 (2000, specificira model slogovnih predlog, kako se jih manipulira)
  - DOM 3 (2004, vsebinski modeli za XML, prehod preko drevesa, le delno podprt v današnjih brskalnikih)

	Trident	Tasman	Gecko	WebKit	KHTML	Presto
DOM1	6.0	Yes	1.0	85	Yes	1.0
DOM2	Mostly	Partial	Mostly	Partial	Mostly	Mostly
DOM3	No	No	Partial	Partial	Partial	Partial

Dokumentni objektni model (DOM)

- drevesna struktura
  - `Window` – predstavlja okno v brskalniku, ki prikazuje dokument
  - `Window.document` – predstavlja prikazani dokument HTML
    - forms (polje obrazcev)
    - anchors
    - links
    - images
  - `Window.navigator` – predstavlja uporabnikov brskalnik
    - appName
    - appVersion



Dostop do elementov

1. **z uporabo polj objektov** (preko *forms* in *elements* polj, originalni DOM 0 način)

```
<form action = ""> <input type = "button" name = “turnItOn" /> </form>
```

=> document.forms[0].element[0]

PROBLEM: naslov DOM je določen s položajem elementa v dokumentu

2. **z uporabo imen elementov**

```
<form name = "myForm" action = ""> <input type = "button" name = "pushMe" />
</form>
```

=> document.myForm.pushMe

PROBLEM: XHTML 1.1 prepoveduje uporabo parametra *name* v elementu *form*

PROBLEM: HTML 5 odpravlja uporabo parametra *name* v nekaterih elementih

3. **z uporabo oznak ID za posamezen element**

```
<form action = ""> <input type = "button" id = "pushMe" /> </form>
```

=> document.getElementById("pushMe")

POMANJKLJIVOST: težavno preiskovanja polj stikal (za izključujočo izbiro)

Izbirna in potrditvena polja

- vendar: množica stikal (in množica stikal za izključujočo izbiro) ima isto ime!
- množica takih stikal, ki sodijo skupaj, je v DOM predstavljena s poljem
- to polje ima isto ime kot pripadajoče polje stikal in je lastnost tega obrazca

```
<form id = "obrazec">
    <input type = "checkbox"  name = "dnevi" value = "ponedeljek" />
        ...
    <input type = "checkbox"  name = "dnevi" value = "petek" />
</form>
// preštejemo število izbranih možnosti
var numChecked = 0;
var dom = document.getElementById("obrazec");
for (index = 0; index < dom.dnevi.length; index++)
if (dom.dnevi[index].checked) numChecked++;
```

Dogodki (events)

- **dogodki** so obvestila, da se je zgodila specifična aktivnost (nalaganje dokumenta, klik miške, pritisk tipke na tipkovnici, ...); je objekt, ki je implicitno kreiran kot odziv na nek dogodek
- **rokovalnik dogodka** (*event handler*) je del kode, ki se izvede kot reakcija na dogodek; pogosto je to pregledovanje za napake in pomanjkljive vnose; poleg dogodka je pomembna informacija tudi element, na katerem se je zgodil dogodek
- **registracija** je proces povezovanja rokovalnika dogodka z dogodkom
- različni elementi imajo različne dogodke
  - <a> ima onblur, onclick, onfocus, ...
  - <input> ima onblur, onchange, onclick, onselect,
  - ...

Event	Tag Attribute
blur	onblur
change	onchange
click	onclick
dblclick	ondblclick
focus	onfocus
keydown	onkeydown
keypress	onkeypress
keyup	onkeyup
load	onload
mousedown	onmousedown
mousemove	onmousemove
mouseout	onmouseout
mouseover	onmouseover
mouseup	onmouseup
reset	onreset
select	onselect
submit	onsubmit
unload	onunload

Registracija dogodkov

DOM 0 - Stari način:

- atributu značke priredimo skripto, ki naj se izvede ob dogodku

```
<input type="radio" id="172" onclick = "alert('Izbira: 172');">
```

- atributu značke priredimo ime funkcije, ki naj se izvede ob dogodku

```
<input type="radio" id="172" onclick = "izbira(172)">
```

- ime rokovalnika dogodka priredimo lastnosti objekta, ki predstavlja element HTML

```
var dom = document.getElementById("myForm");
dom.elements[2].onclick = izbiraN;
```

oziroma krajše:

```
document.getElementById("175").onclick=izbiraN;
```

- pri tem načinu ni podajanja parametrov rokovalniku dogodkov, tako da moramo to rešiti v kodi

Registracija dogodkov

```
<!DOCTYPE HTML>
<html> <head> <title>Dogodki z JavaScript</title> <meta charset="utf-8"/>
  <script type="text/javascript">
    function izbira(val){
      if (typeof(val)=="number"){ alert('Izbral si element: '+val); }
      else {alert('Izbral si element: ' + getChecked()); } }
    function izbiraN(){ alert('Izbrani element: ' + getChecked()); }
    function getChecked(){
      var dom=document.getElementById("myForm");
      for (var i=0;i<dom.rb1.length;i++) if (dom.rb1[i].checked) return (dom.rb1[i].id); }
  </script> </head>
  <body>  <form id="myForm" >
      172<input type="radio" id="172" name="rb" onclick="alert('Izbira: 172')"/> <br />
      173<input type="radio" id="173" name="rb1" onclick="izbira(173)" />
      174<input type="radio" id="174" name="rb1" />
      175<input type="radio" id="175" name="rb1" />
      176<input type="radio" id="176" name="rb1" />
    </form>
    <script type="text/javascript">
      var dom=document.getElementById("myForm");
      dom.elements[2].onclick=izbiraN;
      document.getElementById("175").onclick=izbiraN;
      document.getElementById("176").onclick=izbira;
    </script>
</body> </html>
```

Registracija dogodkov

DOM 2 - Novi način

- dodajanje rokovalnika ima naslednjo obliko:

```
document.elementName.addEventListener("change",
                    chkName,
                    enabled);
```

  - “change” - ime dogodka
  - chkName - ime rokovalnika
  - enabled - Boolean: omogočen rokovalnik med prvo fazo propagacije (capturing)

- rokovalnik lahko tudi odstranimo s klicem metode:
  - `removeEventListener`
  - isti parametri kot `addEventListener`

Registracija dogodkov

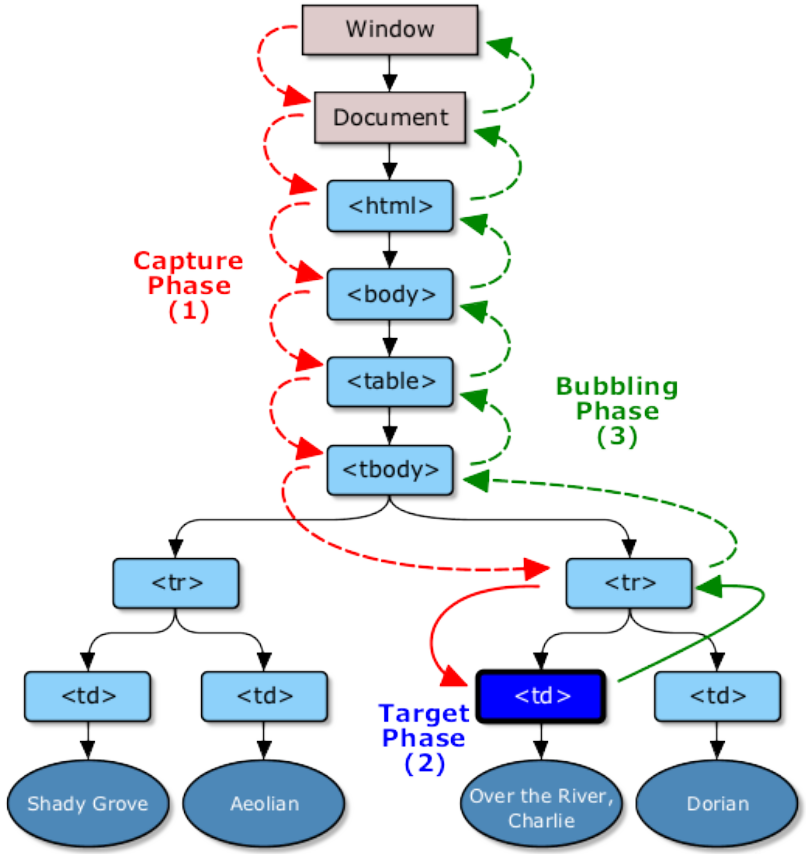
```
<!DOCTYPE HTML>
<html> <head> <title>Dogodki z JavaScript</title> <meta charset="utf-8"/>
  <script type="text/javascript">
    function izbira(val){
      if (typeof(val)=="number") {alert('Izbral si element: '+val); }
      else { alert('Izbral si element: ' + getChecked()); } }
    function izbiraN() {alert('Izbrani element: ' + getChecked()); }
    function getChecked(){
      var dom=document.getElementById("myForm");
      for (var i=0;i<dom.rb1.length;i++) if (dom.rb1[i].checked) return (dom.rb1[i].id); }
  </script> </head>
  <body> <form id="myForm" >
    173<input type="radio" id="173" name="rb1" />
    174<input type="radio" id="174" name="rb1" />
    175<input type="radio" id="175" name="rb1" />
    176<input type="radio" id="176" name="rb1" />
  </form>
  <script type="text/javascript">
    var dom=document.getElementById("myForm");
    dom.elements[0].addEventListener("change",izbira, true);
    dom.elements[1].addEventListener("change",izbiraN, false);
    document.getElementById("175").addEventListener("click", izbiraN, true);
    document.getElementById("176").addEventListener("click", izbira, false);
  </script> </body> </html>
```

Izvedba dogodkov

1. **faza zajemanja** (*capturing phase*): aktivacija izbranih (omogočenih) rokovalnikov za dogodek v smeri korena dokumenta → ciljno vozlišče

2. **izvedba dogodka** v ciljnem vozlišču (*target node phase*): izvedejo se vsi rokovalnik za dogodek, registrirani v ciljnem vozlišču,

3. **mehurčkasta faza** (*bubbling phase*): vračanje proti korenu, izvedba registriranih rokovalnikov za dogodek v smeri ciljno vozlišče → koren dokumenta



Rokovalniki dogodkov

- rokovalnik za dogodek dodamo s klicem
  `element.addEventListener("dogodek", rokovalnik, useCapture)`

- metoda **`preventDefault()`** prekine izvedbo privzete akcije dogodka (oddaja obrazca, preusmeritev povezave, izbira gumba, ...)

- metoda **`stopPropagation()`** prekine razširjanje dogodka po hierarhiji

- rokovalnik za dogodek odstranimo s klicem
  `element.removeEventListener("dogodek", rokovalnik, useCapture)`

Rokovalniki dogodkov

```
<!DOCTYPE HTML >
<html>
  <head> <title>Registracija dogodkov</title> <meta charset="utf-8" />
  <script type = "text/javascript">
    function prvaFunkcija(){
      var ime = prompt("Vpisi svoje ime", "Danko Bananko");
      var pat=new RegExp("^([A-Z][a-z]*) +([A-Z][a-z]*) *$");
      return(pat.test(ime)); }
    function drugaFunkcija() { return (prvaFunkcija()); }
    function tretjaFunkcija(event){ if (!prvaFunkcija()) event.preventDefault(); }
  </script> </head>
  <body> <h1>Registracija rokovalnikov funkcij</h1>
  <form action="cetrtiJS.html" onsubmit="return drugaFunkcija();">
    <input type="submit" value="Pritisni me - DOM 0 - 1" /> </form>
  <form id="obrazec0" action="cetrtiJS.html">
    <input type="submit" value="Pritisni me - DOM 0 - 2" /> </form>
  <form id="obrazec1" action="cetrtiJS.html" >
    <input type="submit" value="Pritisni me - DOM 2" /> </form>
  <script>
  document.getElementById("obrazec0").onsubmit=prvaFunkcija;
  document.getElementById("obrazec1").addEventListener("submit", tretjaFunkcija, false);
  </script>
  </body> </html>
```

Izvedba dogodkov

- krmilniki dogodkov prejmejo objekt s parametri dogodka (tip `Event`):
  `function rokovalnikKlik(e)`
  - `e.altKey, e.ctrlKey, e.shiftKey` – true, če je bil pritisnjen ustrezen gumb
  - `e.clientX, e.clientY, e.screenX, e.screenY` – vrednosti koordinate X oziroma Y v koordinatnem sistemu brskalnika oziroma zaslona
  - `e.target, e.currentTarget` – element DOM, ki je sprejel dogodek
  - `e.type` - tip sproženega dogodka
  - `e.target.innerHTML` – vrne/nastavi vsebino elementa HTML
  - `e.target.getAttribute(atr)` – vrne vrednost atributa atr
  - e.target.setAttribute(atr,vrednost) – nastavi vrednost atributa atr
  - e.target.tagName – vrne ime elementa npr. href
  - …

Upravljanje DOM 2

- metodi `getAttribute` in `setAttribute` spreminjata objekte
- objekt `currentTarget` je referenca na objekt, katerega rokovalnik se izvaja

- metode za navigacijo (DOM Tree Traversal):
  - `parentNode`
  - `previousSibling`
  - `nextSibling`
  - `firstChild,lastChild`
  - `childNodes`

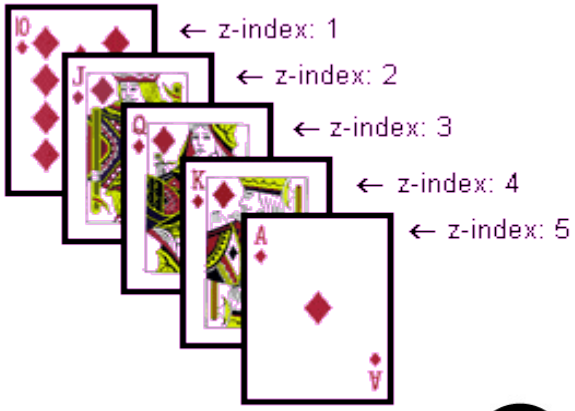
- metode za spreminjanje strukture elementov/strani:
  - `insertBefore`
  - `replaceChild`
  - `removeChild`
  - `appendChild`

JavaScript

- JavaScript kot programski jezik
  - sintaksa, funkcije, operatorji, stavki, objekti
  - polja, regularni izrazi
  - integracija z brskalnikom
- komunikacija z vsebino dokumenta (DOM)
  - objekti brskalnika, naslavljanje
  - dogodki, registracija dogodkov
  - programsko spreminjanje dokumenta
- dinamične spletne strani
  - premikanje elementov, vidnost, barve, pisave, prekrivanje
  - animacija
  - zunanji API

Dinamični HTML dokumenti

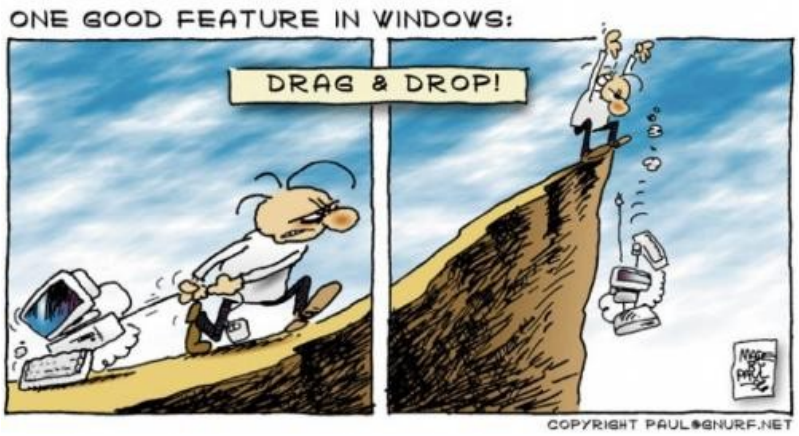
- dinamičnost dodamo s spreminjanjem **lastnosti**, **stila** in **vsebine** značk po tem, ko brskalnik dokument prikaže na zaslonu
  - premikanje elementov
    - sprememba atributov `left` in `top` absolutnega ali relativnega položaja
  - skrivanje/prikazovanje elementov
    - lastnost `visibility = {visible | hidden}`
  - barve in pisave
    - lastnosti `color` in `bgColor`, `font`, `fontSize`, `fontStyle` itd.
  - dinamično prikazovanje vsebine
  - prekrivanje elementov
    - atribut `z-index`
  - interakcija z uporabnikom (miška)
    - lastnosti `event.clientX`, `event.clientY`, `event.screenX`, `event.screenY`

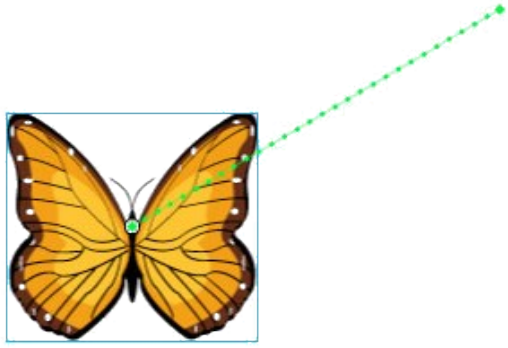


Animacija in premikanje

- **vtis animacije se doseže z zaporednimi majhnimi premiki**

- **gibanje elementov**
  - `setTimeout(funkcija, milisekunde)`
  - `setInterval(funkcija, milisekunde), clearInterval`

- 



Animacija in premikanje

```
<!DOCTYPE HTML >
  <html><head> <title>Test animacije</title> <meta charset="utf-8" />
  <script type="text/javascript">
    var animiraj, dir=1, slika = null;
    function moveRight(){ var newS=parseInt(slika.style.left)+1;
      if (newS+slika.clientWidth>parseInt(window.innerWidth)) newS=0;
      slika.style.left = newS+'px';
      animiraj = setTimeout(moveRight, 10); }
    function moveBoth(){
      var newS=parseInt(slika.style.left)+dir;
      if (newS+slika.clientWidth>parseInt(window.innerWidth)) dir=-1;
      if (newS<0) dir=1; slika.style.left = newS+'px';
      animiraj = setTimeout(moveBoth, 10); }
    function stop(){ clearTimeout(animiraj); }
  </script> </head>
  <body onload="slika=document.getElementById('nasmeh');">
    <img id="nasmeh" src="smilie2.jpg" style="position:absolute; top: 100px; left: 0px;"/>
    <p> Use animation buttons</p>
    <input type="button" value="Start" onclick="moveRight()" />
    <input type="button" value="Stop" onclick="stop()" />
    <input type="button" value="Start" onclick="moveBoth()" />
    <input type="button" value="Stop" onclick="stop()" />
  </body> </html>
```

Dostop do zunanjih API-jev

- številne spletne strani uporabljajo storitve, ki jih zagotavljajo druga podjetja
- do storitev se dostopa preko objavljenih API-jev
- najpopularnejše storitve
  - Google Maps- omogoča vgradnjo zemljevidov v spletno stran
  - Google Maps Geocoding- vrača koordinate za podano informacijo (naslov, mesto, pošta,...)
  - YouTube- omogoča zagotavljanje multimedijskih vsebin
  - IBM Watson- IBM-ova storitev umetne inteligence
  - FullContact- zagotavljanje informacije za osebe za elektronskim naslovom (ime, starost, ...)
  - Twitter- vključevanje Twitter-ja v spletno aplikacijo
  - Facebook- vključevanje Facebook-a v spletno aplikacijo
- glede na tip aplikacije in potrebe je mogoče vključiti različne storitve

Dostop do Google Maps API-ja

- najpopularnejša storitev je Google Maps, dostop preko javnega API-ja
- vključuje se ga v spletne strani preko JavaScript klicev storitvi Google Maps
- omogoča različne poglede (satelit, teren, ulični pogled)
- od 22. junija 2016 dalje je za dostop do Google Maps API-ja potreben ključ
  - dostop brez ključa ni več podprt
  - dovoljenih je 25.000 nalaganj zemljevidov na dan na ključ je dovoljenih
  - zgornja omejitev velja za vse strani, kreirane po 22. 6. 2016, za prej kreirane to ne velja
  - zgornja omejitev velja tudi za vse strani, ki gostujejo na `localhost (127.0.0.1)`
- zagotovitev ključa
  - odpri Google API konzolo in ustvari ali izberi projekt
  - omogoči API in povezane storitve
  - na strani za poverilnice pridobi ključ in ga **zaščiti**, da se izogneš kraji kvot

Dostop do Google Maps API-ja

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>GoogleMaps spletna stran</title>
  <meta charset="utf-8">
  <style> #zemljevid { width: 500px; height: 500px; background-color: #ccc; }</style>
 </head>
 <body>
  <h1>This is Google Maps test</h1>
  <div id="zemljevid"> </div>
 <script>
  function initMap(){
    var zDiv = document.getElementById("zemljevid");
    var zem = new google.maps.Map(zDiv,
                {center: {lat: 46.050226, lng:14.469099}, zoom: 16,
                 mapTypeId: google.maps.MapTypeId.HYBRID});
  }
 </script>
 <script async defer type="text/javascript"
  src="https://maps.googleapis.com/maps/api/js?key=<KEY>&callback=initMap">
 </script>
 </body> </html>
```

Dostop do naprav z JavaScript

- JavaScript dovoljuje dostopanje do različnih naprav na računalniku
- multimedijski tok podatkov se lahko pridobi preko ustreznega API-ja: `navigator.getUserMedia`, ki je lahko neodvisen od brskalnika
- lahko se zajameta avdio in video tokova podatkov, če sta na voljo, potrebno je specificirati, kaj želimo zajeti
- avdio tok se lahko preusmeri na zvočnike (omogoči mikrofon in zvočnike)
- video tok se lahko preusmeri na video element dokumenta
- individualne slike se lahko zajamejo v video elementu
- zajete slike se lahko prikažejo v `canvas`-u
- obstaja Camera API (nestandarden), ki omogoča zajemanje slik, obstaja mobilna podpora, ne pa tudi za namizne brskalnike

Dostop do naprav z JavaScript

```
<!DOCTYPE html>
<html lang="en" xmlns:fb="http://ogp.me/ns/fb#">
  <head>  <meta charset="utf-8" />
    <title>Kamera s HTML5</title>
    <script type="text/javascript">
      function openStream(stream){
        video.src=window.URL.createObjectURL(stream);
        video.play();};
      function errorFunction(error){
        console.log("Error: ", error);}
      function paintCapture() {
        context.drawImage(video, 0, 0, 640, 480);}
      function doVideo() {
        canvas = document.getElementById("slika"),
        context = canvas.getContext("2d"),
        video = document.getElementById("video"),
        videoObj = { "video": true, "audio": true  };
        navigator.getUserMedia =
          navigator.getUserMedia || navigator.webkitGetUserMedia ||
          navigator.mozGetUserMedia || navigator.msGetUserMedia;
        if(navigator.getUserMedia) {
          navigator.getUserMedia(videoObj, openStream, errorFunction );
        }
      }
    </script> </head>
```

```
<body>
  <video id="video" width="800"
     height="600" autoplay></video>
  <p />
  <button id="slikaj">Posnemi sliko</button>
  <p />
  <canvas id="slika" width="800"
     height="600"></canvas>
  <p />
 <script>
  window.addEventListener
   ("DOMContentLoaded",doVideo,false);
  document.getElementById
     ("slikaj").addEventListener
       ("click", paintCapture,false));
 </script>
</body>
</html>
```