



Uvod u JavaScript

JavaScript

- 20 sati predavanja, 40 sati vježbi
- Parcijalni ispit

Cjeline

- Uvod u JavaScript
- Osnove programiranja
- DOM manipulacija
- Napredni JavaScript
- Upravljanje pogreškama
- HTML5 razvojni koncepti

Uvod u JavaScript

Povijest JavaScripta

JavaScript

- Uz HTML i CSS, jedna od temeljnih tehnologija web-a.
- Kreiran 1995. od strane kompanije Netscape kao jezik za browsere koji opisuje ponašanje web stranica.
- Koristi se i naziv ECMAScript, koji predstavlja standard jezika.
 - Zadnji objavljen standard jezika je ES2023

JS

Klasifikacija jezika

- JavaScript != Java
- JavaScript je skriptni programski jezik visoke razine, dobar i za objektno-orientirano i funkcionalno programiranje.
- Skup funkcija koje pruža ovisi o okolini u kojoj se koristi.

Verzije JavaScript jezika

- Od početka jezika u 90-ima JavaScript redovito dobiva nove verzije jezika sa dodanim ili oduzetim dijelovima temeljnih funkcionalnosti. Time se jezik razvija kroz vrijeme, ali i od nas zahtijeva da ga nanovo učimo i nadogradimo svoje znanje.
- Najvažnije verzije koje nas zanimaju su:
 - ES5
 - ES6

ES5

- ES5 je donio mnoge stabilne i zrele funkcionalnosti jezika. Ovo je postalo osnova za razvoj modernih JavaScript aplikacija i omogućilo programerima da pišu pouzdane i održive kodove.
- Nove metode i objekti: ES5 je uveo nove metode i objekte za rad sa nizovima (array) i karakterima (string), kao što su forEach, map, filter, reduce, JSON.parse, JSON.stringify itd.
- Ovo je znatno pojednostavilo i poboljšalo obradu podataka u JavaScriptu.
- Strict mode: ES5 je uveo "strict mode" (strogi režim), koji pomaže u otkrivanju grešaka i ograničava upotrebu nepoželjnih funkcija i osobina jezika, čime se poboljšava sigurnost i performanse koda.

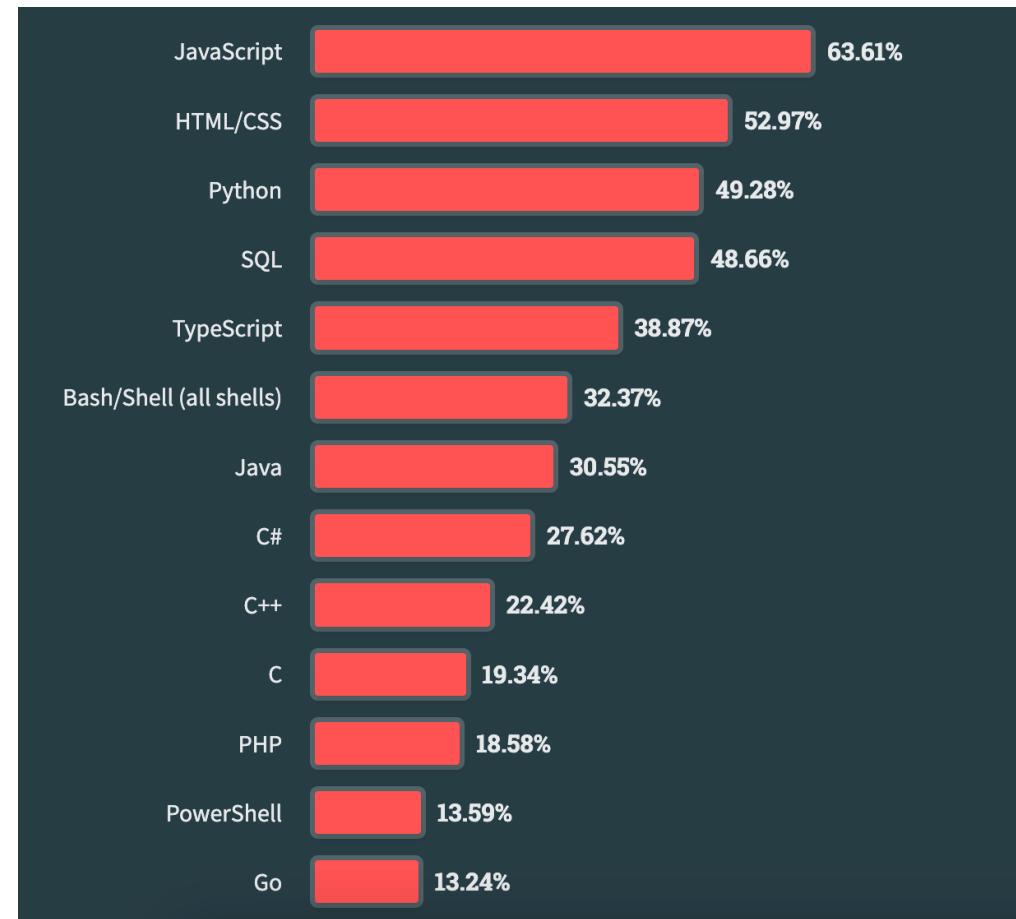
ES6

- Moderni JavaScript: ES6 donosi mnoge moderne funkcionalnosti jezika koje olakšavaju pisanje čitljivijeg i efikasnijeg koda.
- Ovo uključuje deklaraciju promjenljivih varijabli let i const, streličaste funkcije (arrow functions), klase (class), i mnoge druge.
- Novi tipovi podataka: ES6 uvodi nove tipove podataka kao što su Set i Map, koji olakšavaju manipulaciju i upravljanje podacima.
- Moduli: ES6 donosi podršku za modulski sistem, omogućavajući razdvajanje koda u manje module, što olakšava održavanje i ponovnu upotrebu koda.
- Promises: Uvođenjem promisa (obećanja), ES6 poboljšava način na koji se rukuje asinkronim kodom, čineći ga čitljivijim i manje podložnim greškama.

Vrijednost učenja JavaScripta

Široko raširen jezik, ne koristi se više samo u browser okolini.

survey.stackoverflow.co/2023/#technology-most-popular-technologies



Primjeri JavaScripta na web-u

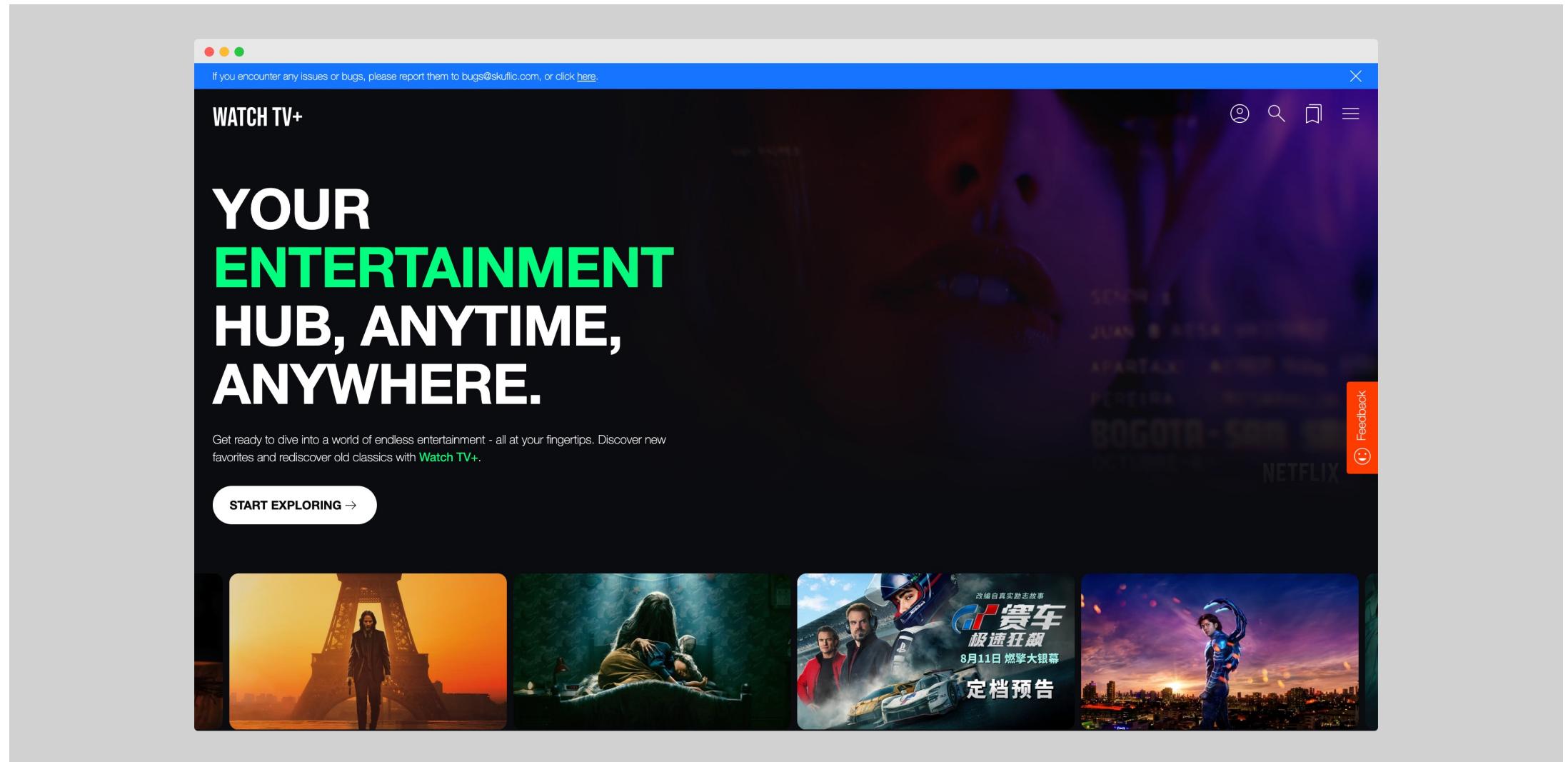
- todo.learn.skuflic.com
- facebook.com
- google.com/maps
- awwwards.com/inspiration/search?text=javascript&type=external_element

Što ako isključimo JavaScript?

smashingmagazine.com/2018/05/using-the-web-with-javascript-turned-off/

Važno je razvijati web stranice imajući na umu osnove web developmenta, s JavaScriptom kao nadogradnjom.





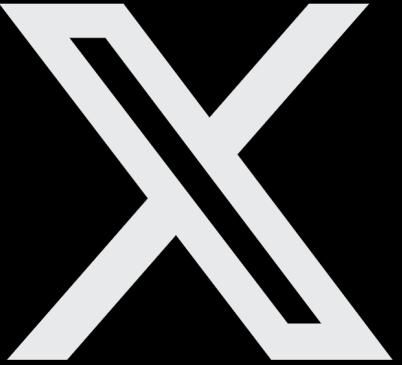


JavaScript is not available.

We've detected that JavaScript is disabled in this browser.

Please enable JavaScript or switch to a supported browser to continue using Watch TV+.

[ENABLE JAVASCRIPT](#)



Happening now

Join today.

 Sign up with Google

 Sign up with Apple

or

Create account

By signing up, you agree to the [Terms of Service](#) and [Privacy Policy](#), including [Cookie Use](#).

Already have an account?

Sign in

[About](#) [Help Center](#) [Terms of Service](#) [Privacy Policy](#) [Cookie Policy](#) [Accessibility](#) [Ads info](#) [Blog](#) [Status](#) [Careers](#) [Brand Resources](#) [Advertising](#) [Marketing](#) [X for Business](#) [Developers](#)
[Directory](#) [Settings](#) © 2023 X Corp.



JavaScript is not available.

We've detected that JavaScript is disabled in this browser. Please enable JavaScript or switch to a supported browser to continue using twitter.com. You can see a list of supported browsers in our Help Center.

[Help Center](#)

[Terms of Service](#) [Privacy Policy](#) [Cookie Policy](#) [Imprint](#) [Ads info](#) © 2023 X Corp.

Uvod u JavaScript

Razvojni alati

Za one koji žele znati više

- developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/Installing_basic_software
- code.visualstudio.com/docs/nodejs/working-with-javascript

Visual Studio Code

- Visual Studio Code uključuje ugrađeni JavaScript IntelliSense, uklanjanje pogrešaka, oblikovanje, navigaciju kroz kod i mnoge druge napredne jezične značajke.
- IntelliSense vam pomaže sa inteligentnim dovršavanjem koda i informacije na hover kako biste brže i ispravnije mogli pisati kôd.

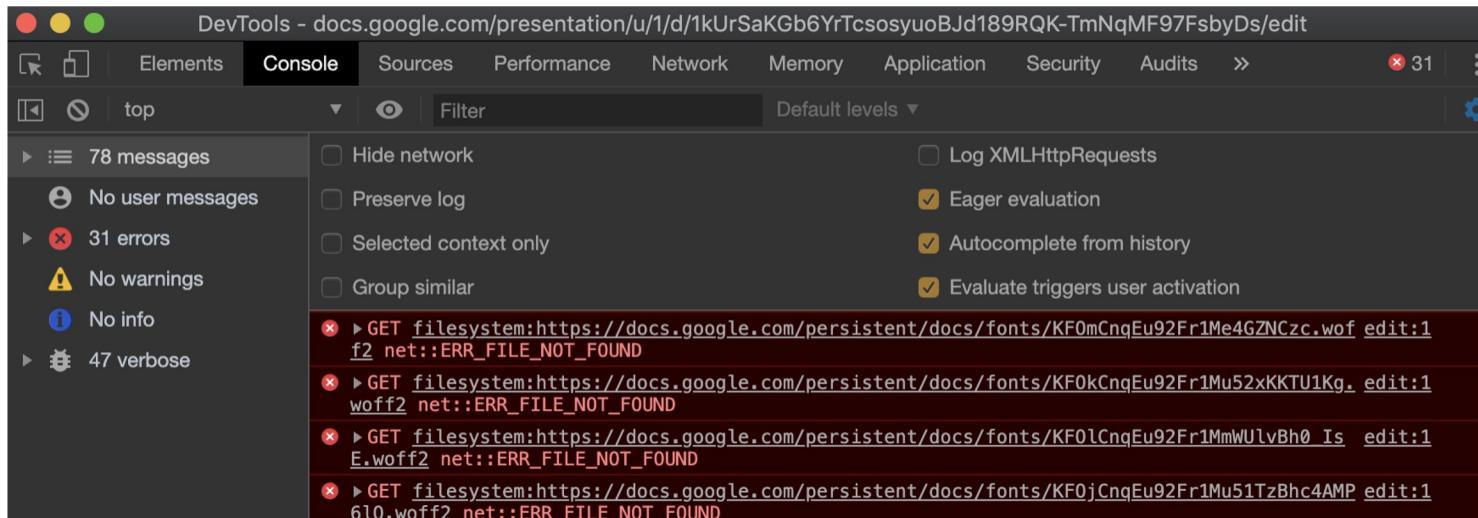
Chrome / Firefox / Edge DevTools

- Chrome / Firefox / Edge DevTools skup je alata za web programere ugrađenih izravno u preglednike i najčešće je korišten web developer alat.

Tab	Uloga
Elements	Pregled HTML-a i CSS-a
Console	Debuggiranje grešaka i JavaScript koda
Sources	Pregled svih izvora stranice i za postavljanje breakpointova
Performance	Praćenje redoslijeda izvođenja i utjecaja na brzinu stranice
Network	Prikaz redoslijeda učitavanja resursa i detalja zahtjeva (request)

Chrome, Firefox i Edge DevTools

- Vjerojatno najvažniji tab za JS developera
- Kroz konzolu pratimo greške koje se javljaju i koje nam ruše aplikaciju
- console.log i debugger; naredbe nam omogućuju praćenje koda i njegov tijek



Firefox, Chrome i Edge

- Mozilla Foundation je sudjelovao u razvijanju JavaScript jezika pa Firefox web preglednik (ali i Chrome i Edge) ima odličnu podršku za testiranje i razvijanje JavaScripta.
 - Isti alati kao Google Chrome
 - Plus WebIDE i Scratchpad
- Njihova stranica je referentna za JavaScript standarde - developer.mozilla.org/en-US/docs/Web/JavaScript

Npm i node paketi

- Node Package Manager
- Otkad je JavaScript razvijen i za poslužiteljsku stranu, postalo je jako popularno razvijati i downloadati pakete razvijene u NodeJS-u. npmjs.com ima na stotine tisuća besplatnih paketa za preuzimanje i korištenje.

Gulp	Automatizacija zadataka kao što su minifikacija
JSHint	Analiza koda i ispravljanje grešaka
lodash	Za rad sa naprednim tipovima podataka
Babel	Za rad sa različitim verzijama JS-a

Ostali alati

Postman	Za razvijanje i testiranje backend API-ja
Sublime text	Alternativni text editor
CircleCI	Za deploy koda
Terminal	Tekstualni pristup operativnom sustavu
GitHub / Bitbucket	Za kontrolu verzija koda
Sourcetree	Vizualni prikaz git protokola

Uvod u JavaScript

JavaScript na web stranici

Klijentski JavaScript

- Prije nego počnemo razgovarati o JavaScriptu, vrijedno je razmisliti o web stranicama i aplikacijama koje prikazujemo u web preglednicima.



Ugradnja JavaScripta u HTML

- Inline, između <script> i </script>
- Iz vanjske datoteke definirane src atributom <script> tag-a
- U atributu alata za obradu HTML događaja, kao što su onclick ili onmouseover
- U URL-u koji koristi poseban javascript: protokol.

```
<script>  
    ...  
</script>
```

```
<script src="scripts.js"></script>
```

Izvođenje JavaScript koda

- Odvija se u 2 faze:
 1. Učitava se sadržaj dokumenta i kôd iz <script> elemenata je pokrenut.
 2. Druga faza se odvija asinkrono i kao reakcija na dogadaje koji se odvijaju unutar web preglednika (klik, loadanje dokumenta, itd.)

Loadanje JS na stranici

- Zadano ponašanje koje određuje izvođenje JavaScript koda na web stranici u prvoj fazi je sinkrono i blokirajuće. Što znači da će dijelovi naše stranice čekati dok se ne učita i izvede sva logika iz naših JS skripti!

Kad HTML parser naiđe na **<script>** element, on mora učitati i pokrenuti skriptu prije nego što moći nastaviti s analizom i prikazom dokumenta.

Sinkrone, asinkrone i odgođene skripte

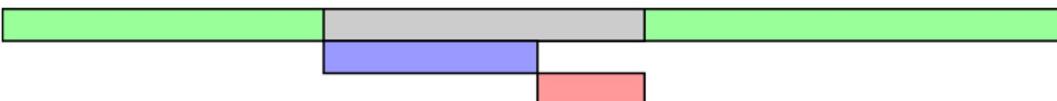
- Različiti tipovi učitavanja:
 - Sinkrono - uključenje JS skripte prije kraja <body> elementa - kad je 99% html-a već učitano
 - Asinkrono - stavljanjem async atributa na <script> element: loadanje skripte ne blokira dokument
 - Odgođeno - stavljanjem defer atributa na <script> element: ni loadanje ni izvođenje JS-a neće blokirati dokument

Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution

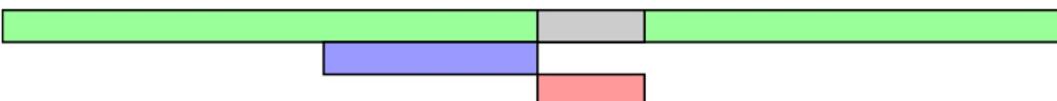
<script>

Let's start by defining what `<script>` without any attributes does. The HTML file will be parsed until the script file is hit, at that point parsing will stop and a request will be made to fetch the file (if it's external). The script will then be executed before parsing is resumed.



<script async>

`async` downloads the file during HTML parsing and will pause the HTML parser to execute it when it has finished downloading.



<script defer>

`defer` downloads the file during HTML parsing and will only execute it after the parser has completed. `defer` scripts are also guaranteed to execute in the order that they appear in the document.



JavaScript na web stranici

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Programeri su inženjeri budućnosti."
- Anonymous



Uvod u JavaScript

Biblioteke i frameworks

Biblioteka (Library)

Skup unaprijed napisanog JavaScript koda koji omogućava lakši razvoj JavaScript aplikacija.

Ima više kategorija biblioteka, koje se bave:

- DOM manipulacijom
- Grafikom na web-u
- User Interface-om
- Testiranjem koda
- AJAX (komunikacija sa serverom)

jQuery

- jQuery je JavaScript biblioteka dizajnirana za pojednostavljenje manipulacije HTML-om, kao i za rukovanje događajima, CSS animaciju i Ajax.
- Jedna od povjesno najpopularnijih JS biblioteka! jQuery koristi čak 97,5% svih web stranica čiju JavaScript biblioteku poznajemo, tj. 74,2% svih web stranica.
- Uz temeljnu jQuery biblioteku postoje i mnoge biblioteke razvijene na temelju jQuery-ja:
creativebloq.com/jquery/top-jquery-plugins-6133175

jQuery

```
document.addEventListener("ready", (function(){
    document.getElementsByTagName("button")[0].addEventListener("click", (function(){
        document.querySelector("p").hide();
    }));
});
```

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

JS Frameworks (okviri)

- JavaScript framework je aplikacijska apstrakcija napisana u JavaScriptu.
- Razlikuje se od JavaScript knjižnice (library) u načinu na koji ga koristimo: biblioteka nudi funkcije koje bi mogle biti pozvane kad nam zatrebaju i pomažu nam u kodiranju, dok okvir (framework) definira cjelokupni dizajn i arhitekturu aplikacije.
 - Angular, teži za naučiti
 - React (postoje dvojbe između njegove definicije)
 - Vue.js (sličan Reactu, lakši za naučiti)

MVC arhitektura

- Model, View, Controller
 - Model je odgovoran za održavanje podataka o aplikaciji i poslovne logike
 - View je korisničko sučelje aplikacije koje prikazuje podatke
 - Controller obrađuje zahtjeve korisnika i poziva odgovarajući View s podacima Modela
- U tom smislu, React je View dio frameworka, i pomaže nam u kreiranju interaktivnih korisničkih sučelja.



Osnove programiranja

Osnove programiranja

- 5 sati predavanja
- 15 sati vježbi

Cjeline

- Leksička struktura jezika
- Deklaracije
- **Vježba 1**
- Tipovi podataka
- Rad sa tekstom
- Operatori
- Izjave
- **Vježba 2**
- Nizovi
- Funkcije
- Objekti
- **Vježba 3**
- Standardni ugrađeni objekti
- Uvod u napredne koncepte

Osnove programiranja

Leksička struktura jezika

Leksička struktura

Leksička struktura je skup elementarnih pravila koja određuju kako pišete programe na tom jeziku.

- Određuje:
 - Ulogu praznina u jeziku
 - Kako izgledaju nazivi varijabli
 - Kako pišemo komentare
 - Kako se jedna programska izjava odvaja od sljedeće

Elementi sintakse

Vrijednosti – literali (npr. 674, 3.14, true ili 'x') i varijable

Operatori - aritmetički, operator dodjele vrijednosti ...

Izrazi - kombinacija vrijednosti i operatora koje se izračunavaju

Ključne riječi - npr. var, let, const, return, function, new itd.

Komentari - za objašnjenje koda

Razmaci i indentacija - za lakše čitanje koda

Zagrade

Izjava

Instrukcija koju računalo izvodi. Naš računalni program je kolekcija izjava.

— Izjave se sastoje od:

- Vrijednosti
- Operatera
- Izraza
- Ključnih riječi
- Komentara

```
var a, b, c;
```

```
a = 5;  
b = 6;  
c = a + b;
```

IZJAVA

PROGRAM

Zagrade

- Zagrade određuju blok izjavu, koja grupira nula ili više izjava.
- Vitičaste zagrade uvijek se trebaju koristiti, čak i u slučajevima kada nisu obavezni, to čini naš kod dosljednim i lakšim za ažuriranje!

```
if (x)
  console.log('Hello');

if (x) {
  console.log('Classroom');
}

if (x) {
  console.log('How are you ?');
  // Neka druga izjava
}
```

Indentacija

- JavaScript ignorira indentaciju, ali radi lakšeg čitanja kôda ju koristimo, i ona bi trebala biti konzistentna.

Postavite pravilo indentacije
u vašem tekstu editoru!
(VS Code = auto format)

```
if ('this_is' == /an_example/) {  
    of_beautifier();  
} else {  
    var a === b ? (c % d) : e[f];  
}  
  
if ('this_is' == /an_example/) {  
    of_beautifier();  
} else {  
    var a === b ? (c % d) : e[f];  
}
```

Točka-zarez

- JavaScript upotrebljava točku-zarez za odvajanje izjava, ali ona je neobavezna u slučaju da su izjave u zasebnim linijama.
- Mi ćemo uvijek upotrebljavati točku-zarez na kraju naših izjava, radi konzistencije i da izbjegnemo krivu interpretaciju našeg koda.



,

Leksička struktura jezika

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Programeri su inženjeri budućnosti."
- Anonymous



Osnove programiranja

Deklaracije

Vrijednosti

- **Literali** su nepromjenjive vrijednosti, poput brojeva i teksta.
 - **Varijable** su promjenjive, i koriste se za spremanje podataka:
 - Koriste ključnu riječ var za deklaraciju
 - Koriste znak jednakosti = za dodjeljivanje vrijednosti
 - Puni izraz za varijablu name
- var name = 'Nino';

Deklaracija

- Prije upotrebe varijable u programu JavaScript, trebali biste je kreirati. Kreacija varijable se službeno zove deklaracija. Dodjeljivanje vrijednosti se zove inicijalizacija.
- Varijable su deklarirane s ključnom riječi var.

```
var i;  
var a = 1;
```

```
var name;  
var name = 'Nino';
```

Deklaracija

```
// Deklaracija zasebnih varijabli  
var name;  
var age;  
  
// U istom retku  
var name, age;  
  
// Deklaracija i inicializacija  
var i = "Nino";  
var age = 26;
```

Koja je vrijednost
varijabli name i
age nakon
deklaracije?

Imenovanje

Veliko i malo slovo

- JavaScript je jezik koji razlikuje velika i mala slova. To znači da jezične ključne riječi, varijable, imena funkcija i drugi identifikatori moraju uvijek biti upisani s dosljednim velikim slovima.
- Na primjer: `mojaVarijabla` je različita od `mojavarijabla`

Imenovanje

Identifikatori

- Identifikator je jednostavno ime za imenovanje varijabli i funkcija i za pružanje oznaka za određene petlje u JavaScript kodu. Ima par pravila kod imenovanja:
 - Mora početi sa **slovom**, **podvlakom** (_) ili **znakom dolara** (\$)
 - Sljedeći znakovi mogu biti slova, znamenke, podvlake ili \$

Imenovanje

Rezervirane riječi

- JavaScript zadržava određeni broj identifikatora kao ključne riječi samog jezika.
- break, delete, case, do, if, switch, var, catch, else, in, this, void, continue, new, null, for, with, default, NaN, Number, String ...
- Ove se riječi **ne smiju** koristiti kao identifikatori!
 - Cijeli popis dostupan je na w3schools.com/js/js_reserved.asp

Komentari

- Komentari se zanemaruju prilikom evaluacije koda.
- JS podržava dva tipa komentara:
 - Sa `//` znakovima - može se koristiti samo u jednom redu
 - Između znakova `/*` i `*/` - može se koristiti u više redova

```
// Komentar u jednom retku  
/* Još jedan komentar */  
  
/* Komentar u više redova  
 * Ekstra zvjezdice u svakom  
retku  
 * se koriste u vizualne svrhe,  
 * ali su neobavezne.  
 */
```

Komentari

JSDoc

- Varijanta JavaDoc standarda komentiranja koji služi objašnjenju API-ja funkcije ili koda koji stvaraju.
- Kada se koristi JSDoc, razni alati za generiranje dokumentacije (kao što su JSDoc generatori) mogu automatski izvući ove komentare i generirati čitljivu dokumentaciju u različitim formatima (npr. HTML ili Markdown) koja pomaže drugim programerima da bolje razumiju vaš kod i kako ga koristiti.

```
/**  
 * Funkcija koja zbraja dva broja.  
 *  
 * @param {number} a - Prvi broj.  
 * @param {number} b - Drugi broj.  
 * @returns {number} - Rezultat zbrajanja.  
 */  
function addNumbers(a, b) {  
    return a + b;  
}  
  
const result = addNumbers(5, 3);  
console.log(result); // Ispis: 8
```

Deklaracije

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 20 minuta

"Programiranje je izazov koji nikada ne prestaje. Uvijek postoji nešto novo za naučiti."
- Anonymous



Leksička struktura

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

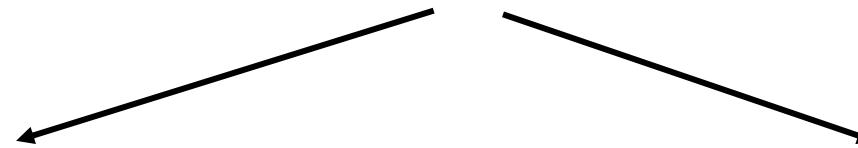
"Kroz programiranje, možemo graditi mostove između ljudi i tehnologije."
- Anonymous



Osnove programiranja

Tipovi podataka

Tipovi podataka u JavaScriptu



Primitivni
Brojevi
Tekst (String)
Boolean vrijednosti
undefined

Objektni
Objekti
Nizovi
Funkcije
null

Promjenjivi i nepromjenjivi tipovi

- Također se mogu kategorizirati kao promjenjivi i nepromjenjivi tipovi
- Nepromjenjivi tipovi podatka su oni čije se stanje ne može promijeniti nakon što se podatak stvori
 - Primitivni tipovi podataka (i null) su nepromjenjivi
 - Objektni tipovi podataka su promjenjivi

```
// STRING
let str = 'Hello';
str + 'World'; // This creates a new string, it doesn't modify the original console.
console.log(str); // Output: 'Hello World'

// BOOLEAN
let isTrue = true;
isTrue = false; // You're actually assigning a new value, not changing the original
console.log(isTrue); // Output: false

// NUMBER
let num = 5;
let newNum = num + 2; // Creating a new number by adding 2
console.log(newNum); // Output: 7
```

Brojevi

- Integer je cijelobrojna vrijednost (bez decimalnih vrijednosti).
- Float je broj sa decimalnom vrijednošću, obično zauzima više memoriskog prostora.
- Svi brojevi u JavaScriptu su floating point tip broja, i spremaju se u 64-bitnom zapisu!

Tekst

- Tekst se u JavaScriptu sprema u string tipu podatka.
- Prepoznajemo ga po navodnicima.

```
var x = 'Lorem ipsum';
var y = "Lorem ipsum dolor sit amet";
```

Boolean tip podatka

- Predstavlja istinu ili neistinu, uključeno ili isključeno, da ili ne
- Postoje samo dvije moguće vrijednosti ove vrste: true ili false
- **Truthy** i **falsey** vrijednosti su one koje se konvertiraju u true i false kada izraz očekuje Boolean vrijednost. Ova kategorizacija pomaže u donošenju odluka o tome kako JavaScript interpretira različite vrijednosti u kontekstu logičkih operacija, naredbi if, while, for, i drugih mesta gdje se očekuje boolean vrijednost.

```
var x = Boolean(null);      // x = false  
var x = Boolean({});       // x = true
```

null i undefined

- **null** je ključna riječ koja se obično koristi da naznači odsutnost neke vrijednosti.
- **undefined** je vrijednost varijabli koje nisu inicijalizirane i vrijednost koju dobivate kada pokušavate dohvatiti vrijednost objekta ili elementa niza koji ne postoji. Također ćete dobiti undefined za izvršenje funkcije koja ne vraća nikakvu vrijednost.

typeof operator

- Možete koristiti operator typeof da biste pronašli vrstu podataka.

```
typeof "John"           // "string"
typeof 3.14             // "number"
typeof NaN              // "number"
typeof false            // "boolean"
typeof [1,2,3,4]        // "object"
typeof {name:'John', age:34} // "object"
```

Automatska konverzija podataka

- JavaScript obavlja mnoge konverzije tipa podatka automatski ovisno o kontekstu u kojemu se podatak koristi, te operatoru.

```
> '5' - 3
< 2
> '5' + 3
< "53"
> '5' - '4'
< 1
> '5' + + '5'
< "55"
> 'foo' + + 'bar'
< "fooNaN"
> '5' + - '2'
< "5-2"
> '5' + - + - + - - + + - + - + - - - '2'
< "5-2"
> x = 3
< 3
> '5' + x - x
< 50
> '5' - x + x
< 5
```

Tipovi podataka

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 20 minuta

"Svaka linija koda koju napišete može promijeniti svijet."
- Steve Jobs



Osnove programiranja
Rad sa tekstom

String

String je niz znakova (0 ili više) **okružen navodnicima**.

- Navodnici mogu biti jednostruki ili dvostruki.
- Brojanje pozicije znaka u stringu počinje od nule!

```
var myString = "";           // prazan string  
myString = "Nino Škuflić";  
myString = " Plava papiga"; // string sa prazninama na početku
```

Konkatenacija

- Operacija spajanja dva ili više stringova
- U JavaScriptu se koristi operator +

```
x = "Hello, " + "World";  
x = "Hello, " + 5;
```

```
var x = "Ovo je vrlo dugi string" +  
        "koji je u dva reda " +  
        "jer se inače ne može čitati.";
```

```
var x = "Ovo je vrlo dugi string \  
        "koji je u dva reda \  
        "jer se inače ne može čitati.";
```

Posebni znakovi

- Neki znakovi unutar stringa se moraju "izbjegći" kako bi se prikazali
- To radimo sa znakom \ (backslash)

```
var x = "Ovo je moj \"string\"";
```

\'	Jednostruki navodnici
\"	Dvostruki navodnici
\\"\\	Backslash
\n	Nova linija
\r	Carriage Return (stari način korištenja)

Metode

length	Duljina stringa
charAt	Vraća znak na poziciji
trim	Miče praznine na početku i kraju stringa
indexOf	Vraća poziciju stringa unutar stringa
split	Pretvara string u niz - <i>više o tome kasnije!</i>
replace	Vraća novi string sa zamijenjenim izrazom
toUpperCase	Pretvara stringu u sva velika slova

OPREZ

Neke metode mijenjaju izvorni string, a neke vraćaju novi string.

Rad sa tekstom

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Najbolji način da naučiš programirati je da pišeš kod i učiš iz vlastitih pogrešaka."
- Anonymous



Osnove programiranja

Operatori

Tipovi operatora

Po tipu operacije

Logički	&& !
Arimetički	+ - * / % ** ++ --
Operatori dodjeljivanja	= += -= *= /= %= **=
Relacijski	== === != !== > < >= <= ?
Bitovni (bitwise)	& ~ ^ << >> >>>
Operator konkatenacije	+
Ostali	delete, typeof, void, instanceof, in

Tipovi operatora

Po broju operanada

- Unarni
- Binarni
- Ternarni - kombinira 3 operacije u jednu

```
var x += 1;           // Unarani operator  
var x = y + 1;       // Binarni operator  
var x = y ? y : x + 1; // Ternarni operator
```

Unarni

```
// Unarni plus (+) i minus (-)
// Koristi se za promjenu predznaka broja ili za eksplisitnu konverziju vrijednosti u broj.
let broj = 5;
let negativno = -broj; // Promjena predznaka
let tekst = "10";
let brojIzTeksta = +tekst; // Konverzija u broj

// Unarni negacija (!)
// Koristi se za negiranje logičke vrijednosti. Ako je operand falsy, rezultat će biti true, inače će biti false.
let istina = true;
let laz = !istina; // Negacija, rezultat je false

// Inkrement (++) i dekrement (--)
// Koriste se za povećanje ili smanjenje vrijednosti varijable za 1.
let broj2 = 10;
broj2++; // Inkrement, broj postaje 11
broj2--; // Dekrement, broj postaje 10
```

Aritmetički operatori

-	oduzimanje	$x = x - 1$	$x -= 1$
+	zbrajanje	$x = x + 1$	$x += 1$
*	množenje	$x = x * 2$	$x *= y$
/	podjela	$x = x / 2$	$x /= 2$
%	modulo (ostatak dijeljenja)	$x = x \% 2$	$x \%= 2$
**	potenciranje (ES6)	$x = x ** 2$	$x **= 2$
--	dekrement	$x--$ ili $--x$	Različito ponašanje ovisno o kontekstu
++	inkrement	$x++$ ili $++x$	

Relacijski operatori

- Testiraju relativni poredak (brojčani ili abecedni) njihova dva operanda:

<code>==</code>	jednaka vrijednost
<code>===</code>	jednaka vrijednost i jednaki tip podatka
<code>!=</code>	Nije jednako
<code>!=</code>	nije jednaka vrijednost ili nije jednaki tip podatka
<code>></code>	više od
<code><</code>	manje od
<code>>=</code>	više ili jednako
<code><=</code>	manje ili jednako

Logički izrazi

&&	i
	ili
!	ne

Operatori

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Nema većeg zadovoljstva od rada na problemu i gledanja kako vaš kod donosi rješenje."
- David Heinemeier Hansson

Osnove programiranja

Izjave

Vrste izjava

- Uvjeti izvršavaju ili preskaču druge izjave ovisno o vrijednosti određenog izraza, tj. granaju kod na više mogućnosti.
- Petlje repetitivno izvršavaju dijelove koda.
- Izjave skokova (poput break, return, throw) naređuju interpretérou da skoči na neki drugi dio programa.

Uvjeti

- Ako je izraz truthy, onda se izvrši izjava/e unutar zagrada. Inače se izjava preskače.
- **Else** nam daje alternativu za slučaj da je izraz falsey.
- Možemo koristiti i **else if** ako želimo postaviti dodatan uvjet.

```
// If - else
if (izraz) {
    // Izvrši kod u bloku #1
} else {
    // Izvrši kod u bloku #2
}
```

Uvjeti

If, else if

```
// Jednostavan if
if (izraz) {
    // Izvrši kod u bloku #1
}
```

```
// If - else if
if (izraz) {
    // Izvrši kod u bloku #1
} else if (izraz2) {
    // Izvrši kod u bloku
#2
}
```

Uvjeti

switch

- Switch izjava je bolja alternativa kompliciranim if izjavama.

```
switch(n) {  
    case 1: // Execute code block #1. break;  
    case 2: // Execute code block #2. break;  
    default: // Execute code block #3. break;  
}
```

Smjer provjere

Petlje

while, do ... while

```
var count = 0;  
while (count < 10) {  
    console.log(count);  
    count++;  
}
```

```
var count = 0;  
do {  
    console.log(count);  
    count++;  
} while (count < 10);
```

Petlje

for, for ... in

```
for (var count = 0; count < 10; count++) {  
    console.log(count);  
}
```

- For ... in petlja se koristi sa objektnim tipovima podataka

```
for (var i = 0; i < a.length; i++) {  
    console.log(a[i]);  
}
```

Izjava

break, continue

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { continue; }  
    text += "The number is " + i + "<br>"  
}
```

— **continue** izjava preskače obradu trenutnog člana petlje i nastavlja sa sljedećim

```
for (i = 0; i < 10; i++) {  
    if (i === 3) { break; }  
    text += "The number is " + i + "<br>"  
}
```

— **break** izjava prekida petlju i izlazi iz nje

Izjave

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 20 minuta

"Programiranje je jezik budućnosti koji svi trebamo naučiti i razumjeti."
- Mark Zuckerberg



Osnove programiranja – 1. dio

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

"Svaki programer je umjetnik čija djela žive na milijunima ekrana."
- Anonymous



Osnove programiranja

Nizovi

Niz (array)

- Sortirna zbirka vrijednosti
- Svaka pojedina vrijednost u nizu se naziva elementom niza. Može biti bilo kojeg JS tipa podatka, a sam niz je tipa objekt. Niz prepoznajemo po uglatim zagradama ([]).

```
var name1 = 'John';
```

```
var name2 = 'Jane';
```

```
var name3 = 'Mike';
```

```
var names = ['John', 'Jane', 'Mike'];
```

```
console.log(names);
```

```
console.log(typeof(names));
```

Niz

Kreiranje niza

literal

```
var emptyArray = [];
var weirdArray = [1,,3];
var names = ['John', 'Jane', 'Mike'];
```

new Array()

```
var emptyArray = new Array();
var arrayLength5 = new Array(5);
var names = new Array('John', 'Jane', 'Mike');
```

Elementi niza

Postavljanje i dohvaćanje

- Za razliku od klasičnih objekata, elementi u nizu se dohvaćaju brojkom (ne-negativnim intergerom), tj. **indexom** elementa. Indexi elemenata počinju od **nule**.
- Nizovi se mogu proširivati i smanjivati po volji.

```
var names = ['Marko', 'Ivana', 'Sanja'];
var name1 = names[0];

names[3] = 'Patrik';

console.log(names); // [ 'Marko', 'Ivana', 'Sanja', 'Patrik' ]
```

Metode nad nizom

push	Dodaje element na kraj niza
pop	Miče zadnji element niza
join	Pretvara sve elemente niza u konkatenirani string
reverse	Obrne redoslijed elemenata u nizu
sort	Sortira elemente niza (default: po abecedi)
forEach (ES5)	Iterira kroz niz, pozivajući funkciju koju odredite za svaki element
map (ES5)	Prosljeđuje svaki element niza funkciji i vraća niz koji sadrži vrijednosti vraćene tom funkcijom

Multidimenzionalni nizovi

- Za stvaranje dvo-dimenzionalnih matrica, u JavaScriptu jednostavno koristimo niz nizova.

```
var matrix = [[1, 2, 3], ['John', 'Jane', 'Mike']]  
  
matrix[0][1];  
matrix[2][2];  
matrix[0][4] = 5;  
matrix[1] = [];
```

Koje su vrijednosti ovih izjava?
Koja je duljina matrix varijable nakon zadnje izjave?

Iteracije nad nizom

for, forEach

```
var names = ['John', 'Jane', 'Mike'];

for (var i = 0; i < names.length; ++i) {
    console.log(names[i]);
}

names.forEach(function(value){
    console.log(value);
});
```

Zadaća: Proučite i for/of i for/in metode iteracija nad nizom, te njihove prednosti/nedostatke.

Nizovi

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Svako veliko programiranje počinje s kritičnim razmišljanjem i planiranjem."
- Anonymous



Osnove programiranja

Funkcije

Funkcija

Funkcija je blok JavaScripta koda koji izvršava neki zadatak.

- U drugim jezicima funkcija se naziva procedura ili subrutina.
- Funkcije koristimo jer njezin kôd možete ponovo upotrijebiti: definirajte ga jednom i koristite ga više puta. Možete koristiti isti kôd više puta s različitim argumentima za dobivanje različitih rezultata.

Definicija i pozivanje funkcije

```
function imeFunkcije() {  
    // Do something  
}  
  
// Other JS code  
  
imeFunkcije();
```

Definicija i pozivanje funkcije

DEFINICIJA

```
function imeFunkcije() {  
    // Do something  
}
```

POZIVANJE

```
imeFunkcije();
```

“Return” ključna riječ

- Svaka funkcija vraća vrijednost `undefined`, osim ako upotrijebimo ključnu riječ `return`. Vraća izračunatu vrijednost iz funkcije.
- U slučaju izjave `return` prije kraja funkcije, ostatak funkcije je ignoriran.

```
function imeFunkcije() {  
    var value = 1;  
    return value;  
}  
  
console.log(imeFunkcije());
```

```
function imeFunkcije() {  
    var value = 1;  
    return value;  
    value += 1;  
}  
  
console.log(imeFunkcije());
```

Što se ispisuje u konzoli u ova dva slučaja?

Parametri i argumenti

- Definicija funkcije može (ali ne mora) sadržavati popis identifikatora, poznatih kao parametri, koji djeluju kao lokalne varijable za tijelo funkcije.

PARAMETRI

```
function udaljenostKoordinata(x1, y1, x2, y2) {  
    var dx = x2 - x1;  
    var dy = y2 - y1;  
    console.log(Math.sqrt(dx*dx + dy*dy));  
}
```

ARGUMENTI

```
udaljenostKoordinata(1,1,3,4);
```

Parametri

- JavaScript ne zahtjeva tip podataka u parametrima koje dobiva u funkciji, kao ni točan broj parametara. U funkciji možemo računati na te optionalne parametre.

```
function myFunc(a, b) {          function myFunc(a, b) {          function myFunc(a, b = 1) {  
    if (!b) {                    b = b || 1;            return a + b;  
        b = 1;                  return a + b;        }  
    }                            }  
    return a + b;                myFunc(1,1);  
}  
  
myFunc(1);  
myFunc(1,1);  
myFunc(1,1);
```

Samo ES6

Funkcija kao vrijednost

Funkcija se u JavaScriptu može koristiti kao bilo koja druga varijabla!

```
var addOne = function(value) {  
    return value + 1;  
};
```

```
var myVar = addOne(1);
```

Funkcije

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 15 minuta

"Kodiranje može biti kreativno i čarobno."
- Charles Petzold

Osnove programiranja

Objekti

Objekt

- Zbirka svojstava od kojih svako ima ime (ključ) i vrijednost. Objekt se prepoznaće po vitičastim zagradama.

```
var dog = {  
    'Golden retriever',  
    bark: function() {  
        return 'Wuf Wuf';  
    },  
};
```

Kreiranje objekta

- Možemo kreirati objekt pomoću:
 - Literala
 - Operatora new - obradit ćemo kasnije!
 - Object.create() funkcije - obradit ćemo kasnije!

Kreiranje objekta

- Možemo kreirati objekt pomoću:
 - Literala

```
var cat = {};
var dog = {
    breed: 'Golden retriever',
    bark: function() {
        return 'Wuf Wuf';
    },
    "has toy": false,
};
```

Svojstva objekta (properties)

- Svojstva objekta imaju ime i vrijednost, the opisne attribute.
- Vrijednost svojstva može biti bilo koja JS izjava, ili get/set funkcija (ES5).

```
var dog = {  
    breed: 'Golden retriever',  
    bark: function() {  
        return 'Wuf Wuf';  
    },  
}
```

Svojstva objekta, sa opisnim atributima:

- Writable
- Enumerable
- Configurable

Svojstva objekta (properties)

Dohvaćanje i postavljanje

- Svojstva možemo postaviti i dohvatiti sa točkom (.) ili zagradama ([]).

```
var dog = {};  
dog.breed = 'Golden retriever';  
dog['other breed'] = 'Chihuahua' ;————
```

```
var dog = {  
    breed: 'Golden retriever',  
    'other breed': 'Chihuahua',  
};
```

```
var firstBreed = dog.breed;  
var secondBreed = dog['second breed'];
```

Svojstva objekta (properties)

Micanje svojstva

- Svojstva mičemo sa ključnom riječi `delete`. U slučaju da je micanje uspjelo ili objekt niti nema to svojstvo, izjava vraća vrijednost `true`.

```
delete dog.breed; // dog objekt više nema svojstvo breed
```

```
var hasDeleted = delete dog['toy'];
console.log(hasDeleted);
```

Probajte pristupiti, postaviti i
micati par svojstava u svom
vlastitom objekt literalu!

Metode nad objektom

propertyIsEnumerable	Provjerava da li je svojstvo objekta brojivo (i nije nasljeđeno)
toString / toLocaleString	Pretvara objekt u (lokalizirani) string
valueOf	Pretvara objekt u primitivnu vrijednost
toJSON / JSON.stringify(object)	Pretvara objekt u serijalizirani JSON objekt (string)

JSON format

- JavaScript Object Notation, podskup JavaScript sintakse, vrlo popularan format za prijenos podataka!
- JSON sintaksa ima sljedeća pravila:
 1. Podaci su u parovima ime / vrijednost (key, value)
 2. Ime vrijednosti mora biti string
 3. Stringovi se pišu isključivo sa dvostrukim navodnicima
 4. Vrijednost JSON svojstva ne može biti funkcija (bit će zanemarena u pretvorbi)
 5. Popularan izbor za REST API (više o tome kasnije)

JSON format - primjer

```
// JavaScript objekt          // Pretvoreni JSON objekt
var person = {                  {
    name: 'John',             "name": "John",
    age: 31,                  "age": 31,
    city: 'New York',         "city": "New York"
    speak: function(word) {   }
        return word;
    }
};
```

Objekti

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

"Programiranje je umjetnost stvaranja nečega iz ničega."

- John McAfee

Osnove programiranja

Standardni ugrađeni objekti

Tipovi objekata po podrijetlu

- Nativni objekt je objekt ili klasa objekata definirana u ECMAScript specifikaciji (Array, Function, Date ...)
- Host objekt je objekt definiran okruženjem domaćina, u našem slučaju web preglednika (npr. HTMLElement objekt)
- Korisnički definiran objekt je svaki objekt kreiran izvršenjem JavaScript koda.

Globalni objekti

Globalni objekt

- Samom globalnom objektu može se pristupiti pomoću this operatora u globalnom području (ali samo ako se ne koristi strogi način rada ECMAScript 5; u tom slučaju vraća se nedefiniran). U stvari, globalni se opseg sastoji od svojstava globalnog objekta, uključujući i naslijedena svojstva, ako ih ima.

Globalni objekti

- Odnose se na objekte globalnog opsega.
- Izraz "globalni objekti" (ili standardni ugrađeni objekti) ovdje se ne smije miješati s globalnim objektom.

Date objekt

- Core JavaScript uključuje konstruktor Date () za kreiranje objekata koji predstavljaju datume i vremena. Ovi objekti Date imaju metode koje pružaju API za jednostavno izračunavanje datuma. Objekti datuma nisu osnovna vrsta kao što su brojevi.

```
var d = new Date();
var d = new Date(2023, 11, 24, 10, 33, 30, 0);
var d = new Date("October 13, 2023 11:13:00");
var d = new Date(-100000000000);
```

Math objekt

- Pored ovih osnovnih aritmetičkih operatora, JavaScript podržava složenije matematičke operacije kroz skup funkcija i konstanti definiranih kao svojstva objekta Math.

```
Math.PI;  
Math.round(4.4); // 4  
Math.sqrt(64);  
Math.ceil(4.4);  
Math.min(0, 150, 30, 20, -8, -200);
```

RegEx

- Regular Expressions
- Redovni izrazi obrasci su koji se koriste za podudaranje kombinacija znakova u tekstu. U JavaScript-u su i redoviti izrazi objekti. Ti se obrasci koriste s exec i testnim metodama RegExp-a te sa metodama match, matchAll, substitute, search i split globalnog String objekta.

```
var re = new RegExp('ab+c');  
var re = /ab+c/;
```

```
// Primjer RegExp-a za traženje email adrese
const emailUzorak = /\bIA-Za-z0-9._%+-1+@LA-Za-z0-9.-1+\.\ LA-Za-z]{2, \b/;

// Tekst u kojem cemo traziti email adrese
const tekst = 'Evo nekoliko email adresa: john.doe@microsoft.com, alice_smith123@google.com i info@apple.net.';

// Koristimo metodu match() za pronađenje email adresa
const emailAdrese = tekst.match(emailUzorak);

// Ispisujemo pronađene email adrese
console.log(emailAdrese);
```

Osnove programiranja – 2. dio

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

"Programiranje je alat koji omogućuje ljudima da izraze svoje najkreativnije ideje."
- Drew Houston





DOM Manipulacija

DOM Manipulacija

- Ključne riječi
 - HTML DOM, Window objekt, getElementsByClassName, innerHTML

Cjeline

- Selektiranje DOM elemenata
- Mijenjanje DOM strukture
- Mijenjanje sadržaja i atributa
- Mijenjanje stila
- Događaji
- Vježba

DOM manipulacija

Browser okolina

Web API-ji

- Web nudi mnoštvo standardiziranih API-ja i sučelja (tipova objekata) koje biste mogli koristiti tijekom razvoja web aplikacije ili web mjesta.
- API
 - File System, Storage, DOM, Web Animations, Canvas, Ambient Light Events, WebGL, Touch Events, Fetch
...
- Sučelja
 - Navigator, Node, Animation, Performance, Bluethooth, CSS, Position, Cache, Comment, CustomEvent, SVGELEMENT, History, ...

HTML DOM

Platforma i jezično neutralno sučelje koje omogućuje programima i skriptama dinamički pristup i ažuriranje sadržaja, strukture i stila dokumenta.

Definira:

- HTML elemente kao objekte
- Svojstva svih HTML elemenata
- Metode pristupa svim HTML elementima
- Događaje za sve HTML elemente

JavaScript

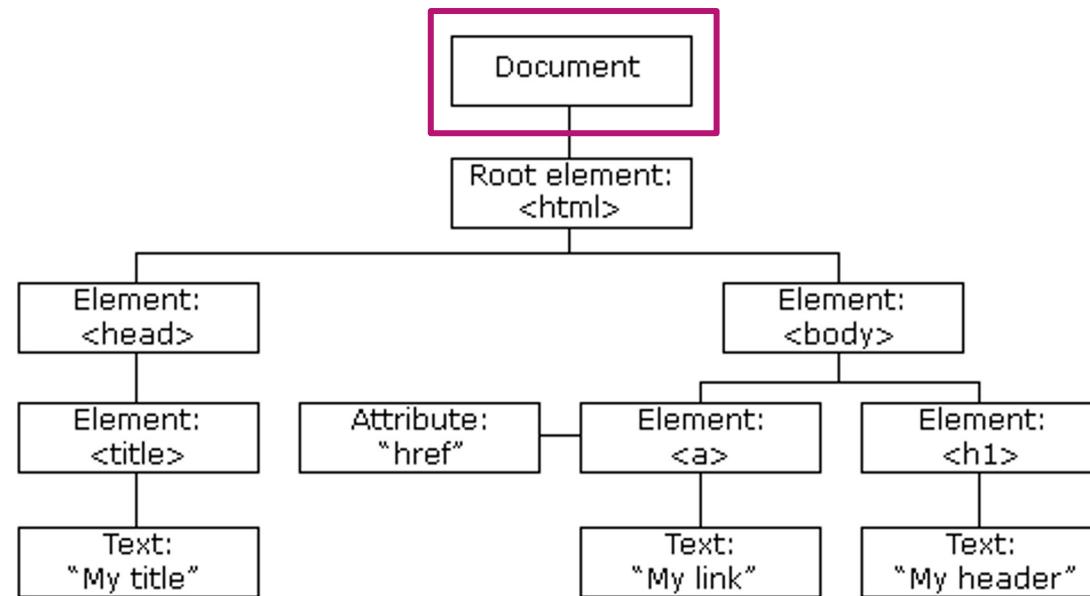
- HTML DOM-u može se pristupiti s JavaScript-om.
- Elementi DOM-a su objekti, te imaju svojstva i metode.

Zapamtite!

~~JavaScript sloj, zadužen za ponašanje stranice, trebao bi biti nonametljiv, što znači da ne bi trebao učiniti stranicu neupotrebljivom u nepodržanim preglednicima i ne bi trebao biti uvjet da možemo pristupiti sadržaju stranice.~~

Document objekt

- Objekt dokumenta predstavlja našu web stranicu.
- Ako želite pristupiti bilo kojem elementu HTML stranice, uvijek započinjemo s pristupom objektu dokumenta.



BOM i Window objekt

- Browser Object Model omogućava JavaScriptu razgovor s web preglednikom.
- Objekt preko kojeg to ostvarujemo je globalni window objekt.
- *Window* ima sljedeća svojstva, među ostalima:

document	Isti objekt kao u HTML DOM-u
innerHeight & innerWidth	Veličine trenutnog prozora
screen	Objekt koji drži informacije o korisničkom zaslonu
location	Objekt koji sadrži metode za manipulaciju url-a



Browser okolina

Pratite upute u Instructions.md
Trajanje vježbe: 20 minuta

Programiranje nije o znanju, već o rješavanju problema.
- Kathy Sierra

DOM manipulacija

Mijenjanje DOM strukture

Primjer DOM-a

- Na ovom satu koristit ćemo sljedeću DOM strukturu kao početnu točku:

```
<!DOCTYPE HTML>
<html>

    <head>
        <title>Karlovac</title>
    </head>

    <body>
        <div class='description'>
            Grad na <span>4 rijeke</span>
        </div>

        <footer id='footer'>
            <ul>
                <li></li>
                <li></li>
            </ul>
        </footer>

    </body>

</html>
```

Selektiranje DOM elementa

- Za mijenjanje DOM-a ili elementa u DOM-u prvo ga moramo ‘pronaći’ u dokument strukturi i selektirati.

DOM definira brojne načine odabira elemenata, koristeći:

- ID elementa
- tag elementa (tip)
- CSS klasu ili drugi selektor
- atribut imena

Selektiranje

```
document.getElementById(footer);
```

```
> document.getElementById("footer");
<-- ► <footer id="footer">...</footer>
```

```
<!DOCTYPE HTML>
<html>

<head>
    <title>Karlovac</title>
</head>

<body>
    <div class='description'>
        Grad na <span>4 rijeke</span>
    </div>

    <footer id='footer'>
        <ul>
            <li></li>
            <li></li>
        </ul>
    </footer>

</body>

</html>
```

Selektiranje

```
document.getElementsByType(li);
```

U slučaju da postoji samo jedan element tog tipa, vraća se HTML tog elementa.

U slučaju da ih ima više, vraća se niz elemenata pod nazivom NodeList.

```
<!DOCTYPE HTML>
<html>

<head>
    <title>Karlovac</title>
</head>

<body>
    <div class='description'>
        Grad na <span>4 rijeke</span>
    </div>

    <footer id='footer'>
        <ul>
            <li></li>
            <li></li>
        </ul>
    </footer>

</body>

</html>
```

NodeList

```
var paragraphs = document.querySelectorAll('p'); // Dobivanje NodeList-a svih <p>  
elemenata  
  
for (var i = 0; i < paragraphs.length; i++) {  
    console.log(paragraphs[i].textContent); // Prikazivanje teksta svakog <p> elementa  
}
```

Selektiranje

```
document  
  .getElementsByClassName(description);  
  
document.querySelector(.description);  
  
document  
  .querySelectorAll(.description);
```

```
<!DOCTYPE HTML>  
<html>  
  
<head>  
  <title>Karlovac</title>  
</head>  
  
<body>  
  <div class='description'>  
    Grad na <span>4 rijeke</span>  
  </div>  
  
  <footer id='footer'>  
    <ul>  
      <li></li>  
      <li></li>  
    </ul>  
  </footer>  
  
</body>  
  
</html>
```

Kreiranje i dodavanje elementa

```
var f = document.querySelector('#footer');
var d = document.createElement(div);
f.appendChild(d);
```

```
<!DOCTYPE HTML>
<html>

<head>
  <title>Karlovac</title>
</head>

<body>
  <div class='description'>
    Grad na <span>4 rijeke</span>
  </div>

  <footer id='footer'>
    <ul>
      <li></li>
      <li></li>
    </ul>
    <div></div>
  </footer>

</body>

</html>
```

Micanje elemenata

```
var f = document.querySelector('#footer');
f.parentNode.removeChild(f);
```

```
<!DOCTYPE HTML>
<html>

<head>
    <title>Karlovac</title>
</head>

<body>
    <div class='description'>
        Grad na <span>4 rijeke</span>
    </div>

    <div id='content'>
        <ul>
            <li></li>
            <li></li>
        </ul>
        <div></div>
    </div>
</body>

</html>
```



Mijenjanje DOM strukture

Pratite upute u Instructions.md
Trajanje vježbe: 25 minuta

DOM manipulacija

Mijenjanje sadržaja i atributa

Sadržaj

Koji je sadržaj div.description-a?

- a) Grad na 4 rijeke
- b) Grad na 4 rijeke
- c) Tekstualni i element node-ovi

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Karlovac</title>
</head>
<body>
    <div class="description">
        Grad na <span>4 rijeke</span>
    </div>
    <footer id="footer">
        <ul>
            <li></li>
            <li></li>
        </ul>
    </footer>
</body>
</html>
```

Sadržaj

Koji je sadržaj div.description-a?

- a) Grad na 4 rijeke
- b) Grad na 4 rijeke
- c) Tekstualni i element node-ovi

```
var content = description.innerHTML;  
description.innerHTML =  
    'Ima <span>4 rijeke</span>' ;
```

Sadržaj

Koji je sadržaj div.description-a?

- a) Grad na 4 rijeke
- b) Grad na 4 rijeke
- c) Tekstualni i element node-ovi

```
var content = description.textContent;  
description.textContent = 'Ima 4 rijeke';  
description.innerText = 'Ima 4 rijeke';
```

Atributi

- Vrijednosti atributa HTML elemenata dostupne su kao svojstva HTMLElement objekata koji predstavljaju te elemente.
- var image = document.querySelector(image);
- image.src = ipsum.png;

```
<!DOCTYPE HTML>
<html>
<head>
  <title>Karlovac</title>
</head>
<body>
  <image src="lorem.png" width="30"/>
  <div class="description">
    Grad na <span>4 rijeke</span>
  </div>
  <footer id="footer">
    <ul>
      <li></li>
      <li></li>
    </ul>
  </footer>
</body>
</html>
```

Ne-HTML i data Atributi

```
var image = document.querySelector(image);
image.getAttribute(width);
image.setAttribute(height) = 40;
image.removeAttribute(height);

var ul = document.querySelector(ul);
ul.dataset.index = 2;
```

```
<!DOCTYPE HTML>
<html>
<head>
    <title>Karlovac</title>
</head>
<body>
    
    <div class="description">
        Grad na <span>4 rijeke</span>
    </div>
    <footer id="footer">
        <ul data-index="1">
            <li></li>
            <li></li>
        </ul>
    </footer>
</body>
</html>
```

Dataset

```
<div id='myElement' data-name='John' data-age='30' data-city='New York'>Hello, World!</div>

const element = document.getElementById('myElement');
const elName = element.dataset.name;
const age = element.dataset.age;
const city = element.dataset.city;

console.log(elName); // Ispisat ce "John"
console.log(age); // Ispisat ce "30"
console.log(city); // Ispisat ce "New York"
```

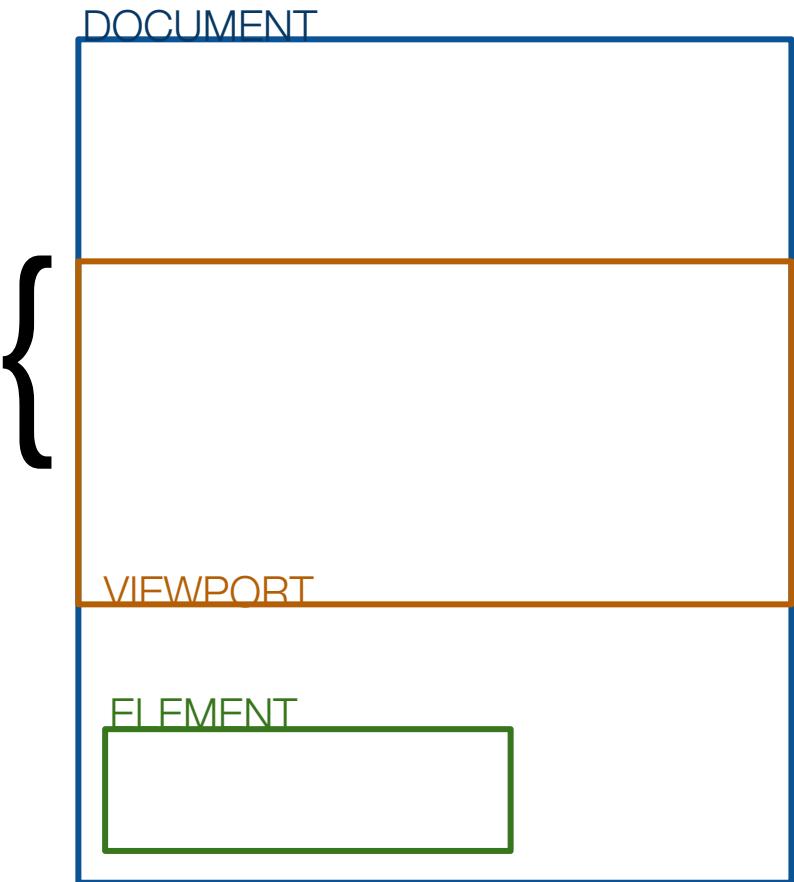
Geometrija elemenata

- Svaki HTML element ima širinu i visinu te poziciju izraženu u pikselima.

Koordinate x i y mogu biti u odnosu na:

- gornji lijevi kut dokumenta
- gornji lijevi kut viewporta
- Atributi elementa kojima možemo pristupiti:
pageOffsetX, pageOffsetY,
 - scrollLeft, scrollTop

scroll offset





Mijenjanje sadržaja i atributa

Pratite upute u Instructions.md
Trajanje vježbe: 30 minuta

DOM manipulacija

Mijenjanje stilizacije

Inline stilovi

- Atribut stila HTML elementa je njegov inline stil i poništava sve specifikacije stila u prije definiranom stylesheetu. Svojstvo stila ima najviši prioritet u CSS kaskadi kao inline deklaracija stila postavljena putem atributa stila.

```
document.getElementById(id).style.property = new style;  
element.style.position = relative;
```

CSSStyleDeclaration

- Svojstvo stila se razlikuje od drugih atributa HTML elementa po tome što njegova vrijednost nije niz, već CSSStyleDeclaration objekt, i ima posebna pravila imenovanja CSS svojstava.

element.style.**font-family** = Arial; ✗

element.style.**fontFamily** = Arial; ✓

element.style.**float** = left; ✗

element.style.**cssFloat** = left; ✓

Izračunati stilovi

- Teže je doći do već predefiniranih vrijednosti stilova primijenjenih na elementu, ali možemo koristiti window metodu getComputedStyle().
- Ta nam metoda vraća stil koji je na kraju primijenjen na elementu i read-only je.

```
var element = document.getElementById(id);  
var styles = window.getComputedStyle(element, null);  
var style = getPropertyValue(styles, font-size);
```

Mijenjanje CSS klase

- Ponekad želimo umjesto inline stila promijeniti CSS klasu elementa.
- Za to koristimo className i classList svojstva elemenata.

```
var element = document.getElementById(id);
element.className = containerElement containerElement--left;

console.log(element.classList);
element.classList.add(hide);
element.classList.remove(hide);
```

Animiranje CSS-a

- Zbog svoje dinamičke i interaktivne prirode, JavaScript se često koristi za animiranje CSS svojstava.

```
var elem = document.getElementById(cube);
var pos = 0;
var id = setInterval(frame, 10);

function frame() {
    pos++;
    elem.style.left = pos + 'px';
}
```

Mijenjanje stilizacije

Pratite upute u Instructions.md
Trajanje vježbe: 25 minuta



DOM manipulacija

Dogadaji

Događaj

- Kad god se nešto dogodi na našoj web stranici, možemo koristiti JavaScript u reakciji na taj događaj.

Dogadaj

Pojmovi

event type	Tip događaja
event target	Objekt nad kojim se događaj dogodio ili s kojim je događaj povezan
event handler / listener	Funkcija koja reagira na događaj
event object	Objekt koji sadrži detalje o događaju (npr. type i target)
event propagation	Proces koji browser koristi da odredi event target
event capturing	Akcija koja se može poduzeti prije nego događaj dođe do mete

Tipovi događaja

Form	blur, change, focus, input, invalid, reset, submit, ...
Window	load, offline, line, popstate, resize, storage, ...
Mouse	click, mousedown, mouseup, mousewheel, wheel, ...
Key	keydown, keypress, keyup
HTML5	drag, dragstart, scroll, play, pause, progress, ...
Mobile	touchstart, touchend

addEventListener()

Ime event handlera ima prefix “on-”

- 3 su načina postavljanja:
 1. inline
 2. kao svojstvo na elementu
 3. addEventListener metoda

```
<input type=button onclick=blur(); value=Click />
```

addEventListener()

Ime event handlera ima prefix “on-”

- 3 su načina postavljanja:
 1. inline
 2. kao svojstvo na elementu
 3. addEventListener metoda

```
<input type=button value=Click />  
  
var button = document.querySelector('button');  
button.onclick = function() {  
    ...  
}
```

addEventListener()

- 3 su načina postavljanja:
 1. inline
 2. kao svojstvo na elementu
 3. addEventListener metoda

```
<input type=button value=Click />
```

```
var button = document.querySelector(button);  
function handleBlur() { ... }
```

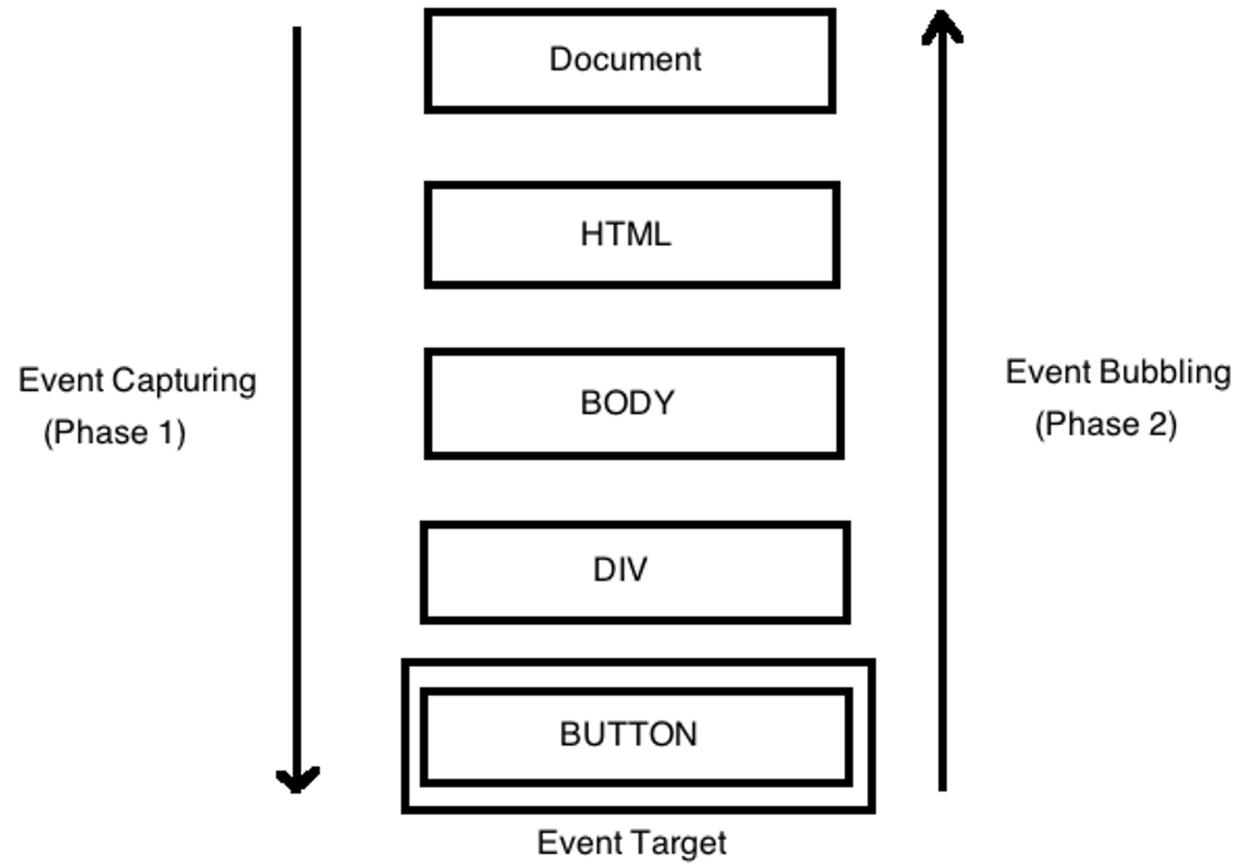
```
button.addEventListener(click, handleBlur);  
button.removeEventListener(click, handleBlur);
```

Handler funkcija

- Funkcija koja reagira na događaj uvijek dobiva objekt događaja kao parametar.
- Također možemo i otkazati defaultnu reakciju na taj događaj, te spriječiti propagaciju događaja na druge elemente.

```
function handleBlur(event) {  
    var target = event.target;  
  
    if (target == element) {  
        event.preventDefault();  
        event.stopPropagation();  
    }  
    return false;  
}
```

Bubbling



```
<div id='parent'>
<button id='child'>Click Me</button>
</div>

const parent = document - getElementById('parent');
const child = document.getElementById('child');

parent.addEventListener('click', () => {
    console.log('Parent clicked');
});

child.addEventListener('click', () => {
    console.log('Child clicked');
});

child.addEventListener('click', (event) => {
    console.log('Child clicked');
    event.stopPropagation(); // Spriječava daljnje bubrenje dogadaja prema
nadredenim elementima
});
```



Napredni JavaScript (napredni koncepti)

Napredni JavaScript

Ključne riječi

closures, scope, “use strict”, OOP, patterns, ES6, API

Cjeline

- Uvod u napredne koncepte
- Funkcije - napredni koncepti
- **Vježba 1**
- JavaScript OOP - 1.dio
- JavaScript OOP - 2.dio
- **Vježba 2**
- Obrasci programskog koda
- ES6 standard - 1.dio
- ES6 standard - 2.dio
- Rad s API-jima
- **Vježba 3**
- Animacije u JavaScriptu

Osnove programiranja

Uvod u napredne koncepte

Eksplicitna konverzija podataka

Par je metoda koje možemo koristiti kad želimo konvertirati vrijednost:

1. Upotrijebiti Number, String, Boolean i Object funkcije
2. `toString` metoda za konverziju bilo koje vrijednosti u string
3. `parseInt` i `parseFloat` parsiraju vrijednost i izvlače broj

Eksplicitna konverzija podataka

```
var tekst = '26 godina';
var broj = parseInt(tekst);

console.log(broj); // Ispisat će 26 jer je '26' pretvoreno u broj, a 'godine' je ignorirano.
```

Hoisting (izvlačenje)

Hoisting je premještanje svih deklaracija na vrh trenutnog opsega (na vrh trenutne skripte ili trenutne funkcije). **Varijable je uvijek dobro deklarirati na početku funkcije!**

```
console.log(x); // Ispisat će 'undefined', ali neće izbaciti gresku
var x = 10;

sayHello(); // Poziv funkcije prije deklaracije

function sayHello() {
  console.log('Pozdrav!');
}
```

Hoisting

- Hoisting u JavaScriptu **nije namjerno dizajniran koncept** za programiranje, već je to više ponašanje jezika koje se javlja tijekom faze kompilacije.
- Lako hoisting može izgledati zbumujuće, nije ga uvijek potrebno koristiti i može dovesti do nesporazuma i grešaka u kodu ako se ne koristi pažljivo.
- Unatoč ovim situacijama u kojima hoisting može biti koristan, **preporučuje se pridržavanje najboljih praksi u kodiranju kako biste izbjegli nesporazume i greške.** To uključuje eksplisitno deklariranje varijabli na vrhu opsega, deklariranje funkcija prije njihovog korištenja i općenito pisanje čitljivog koda koji je lako razumjeti i održavati.

Objekt

- Osnovni oblik podataka JavaScript-a, zbirka svojstava od kojih svako ima ime (ključ) i vrijednost.

```
{ name: 'Golden retriever' }  
function bark() {}  
['John', 'Jane', 'Joe']
```

I objekti i funkcije i nizovi su objektni tip podatka i mogu imati metode i svojstva!

Opseg (scope)

- JavaScript koristi leksički opseg implementiran funkcijama.
- Postoje 2 tipa dosega opsega u Javascriptu: lokalni i globalni.

Opseg određuje dostupnost varijabli, objekata i funkcija iz različitih dijelova koda.

```
var x = 2;  
  
function() {  
    var x = 1;  
}  
  
console.log(x);
```

Globalne i lokalne varijable

```
var x = 2;  
  
function check() {  
    x = 1;  
}  
  
check();  
console.log(x);
```

- U slučaju da je varijabla deklarirana unutar funkcije, ona je lokalna, osim ako se izostavi ključna riječ var (tada je globalna).
- U slučaju da je varijabla deklarirana van funkcije, ona je globalna.

Koju će vrijednost ispisati
console.log?

Globalne i lokalne varijable

```
// Globalna varijabla
var globalnaVarijabla = 'Ovo je globalna varijabla';

function primjerFunkcije() {
    // Lokalna varijabla
    var lokalnaVarijabla = 'Ovo je lokalna varijabla';

    console.log(globalnaVarijabla); // Pristup globalnoj varijabli
    console.log(lokalnaVarijabla); // Pristup lokalnoj varijabli
}

console.log(globalnaVarijabla); // Pristup globalnoj varijabli izvan funkcije
//console.log(lokalnaVarijabla); // Ovako pristup lokalnoj varijabli izvan funkcije će
izazvati grešku
```

Namespace

- Problem s globalnim varijablama je taj što ih dijele svi kodovi u vašoj JavaScript aplikaciji ili web stranici.
- Oni žive u istom **globalnom prostoru** imena i uvijek postoji mogućnost sudara imenovanja - kada dva odvojena dijela aplikacije definiraju globalne varijable s istim nazivom, ali s različitim svrhama.

Ključna riječ „this”

- Ključna riječ this odnosi se na objekt konteksta izvođenja!
- Van objekta ili u funkciji se odnosi na globalni objekt
- U metodi objekta se to odnosi na sam objekt
 - ... o drugim kontekstima pričat ćemo na Naprednom JavaScript dijelu.

```
var person = {  
    name: 'John',  
    speak: function(word) {  
        return this.name;  
    }  
};
```

Ključna riječ “this” u funkcijama

- U običnim funkcijama, "this" se obično odnosi na globalni objekt, osim ako je funkcija pozvana kao metoda objekta, tada se "this" odnosi na taj objekt.

```
var obj = {  
  name: 'Objekt',  
  logName: function () {  
    console.log(this.name); // "this" se odnosi na objekt 'obj'  
  }  
};  
  
obj.logName(); // Ispisuje 'Objekt'
```

Ključna riječ this u konstruktorima

- U konstruktorima "this" se odnosi na novi objekt koji se stvara pomoću konstruktora.

```
function Osoba(ime) {  
    this.ime = ime;  
}  
  
var osoba1 = new Osoba('John');  
console.log(osoba1.ime); // 'John'
```

Ključna riječ `this` u arrow funkcijama

- Arrow funkcije (streličaste funkcije) imaju posebno ponašanje za "this". "this" u arrow funkciji zadržava vrijednost "this" izvan funkcije, često referencirajući kontekst roditeljske funkcije.

```
var obj = {
  name: 'Objekt',
  regularFunction: function () {
    console.log(this.name); // 'this' se odnosi na objekt 'obj'
  },
  arrowFunction: () => {
    console.log(this.name); // 'this' zadržava kontekst izvan objekta, može biti globalni objekt
  }
};

obj.regularFunction(); // Ispisuje 'Objekt'
obj.arrowFunction(); // Ispisuje undefined ili globalni objekt, ovisno o kontekstu
```

(Napredne) Funkcije

Pratite upute u Instructions.md
Trajanje vježbe: 30 minuta

"Sposobnost razmišljanja o onome što želite u JavaScriptu i prevodenja toga u kod je ključna vještina."

- Chris Pine



Napredni JavaScript

Funkcije - napredni koncepti

Načini korištenja funkcije

- U JavaScriptu funkcije su objekti prve klase, što znači se se tretiraju kao bilo koja druga varijabla i da mogu imati vlastita svojstva i metode.
- Funkcije se mogu:
 - Dodijeliti varijabli
 - Prosljediti drugoj funkciji kao argument
 - Vratiti iz druge funkcije (koristeći ključnu riječ return)

Funkcija koja vraća
drugu funkciju zove
se Higher-Order
Function.

Načini korištenja funkcije

Kao varijabla

```
function vratiNesto(x) {  
    return x;  
}
```

```
var y = vratiNesto();
```

```
function vratiNesto(x) {  
    return x;  
}
```

```
var y = vratiNesto;  
y("Hello world");
```

Načini korištenja funkcije

Kao argument druge funkcije

```
function multiplyByTwo(x) {  
    return x * 2;  
}  
  
function operateOnNumber(number, operation) {  
    return operation(number);  
}  
  
const number = 5;  
const doubleNumber = operateOnNumber(number, multiplyByTwo);  
  
console.log(doubleNumber) // Rezultat: 10
```

Načini korištenja funkcije

Kao vraćena vrijednost

```
function createMultiplier(factor) {  
    // Funkcija createMultiplier vraca drugu funkciju.  
    return function (x) {  
        return x * factor;  
    }  
  
const double = createMultiplier(2); // Stvaranje funkcije za dvostruko množenje  
const triple = createMultiplier(3); // Stvaranje funkcije za trostruko množenje  
  
console.log(double(5)); // Rezultat: 10 (5 * 2)  
console.log(triple(5)); // Rezultat: 15 (5 * 3)
```

Funkcija koja prima funkciju kao argument

```
// Funkcija koja prima dvije brojčane vrijednosti i funkciju za zbrajanje ili oduzimanje
function operation(a, b, operation) {
    return operation(a, b); // Poziv funkcije za zbrajanje ili oduzimanje
}

// Funkcija za zbrajanje
function add(x, y) {
    return x + y;
}

// Funkcija za oduzimanje
function subtract(x, y) {
    return x - y;
}

// Primjeri poziva funkcije 'izracunaj' s različitim operacijama
var result1 = operation(5, 3, add);
var result2 = operation(8, 2, subtract);

console.log('Rezultat zbrajanja: ' + result1); // 8
console.log('Rezultat oduzimanja: ' + result2); // 6
```

Anonimna funkcija

```
(function() {  
    var x = 1;  
    return x;  
})
```

```
(function() {  
    var x = 1;  
    return x;  
})()
```

Anonimna funkcija je funkcija koja nema ime. Može biti izravno dodijeljena varijabli, ili obuhvaćena zagradama (npr. da ne zagađuje globani scope).

- Immediately Invoked Function Expression (IIFE) tj. **samopozivajuća anonimna funkcija**.
- Anonimna funkcija nema ime i stoga se može koristiti samo ako je dodijelite varijabli, pozovete je odmah nakon deklaracije ili je proslijedite kao argument nekoj drugoj funkciji.

Anonimna funkcija

- Anonimna funkcija je funkcija koja nema ime. Može biti izravno dodijeljena varijabli, ili obuhvaćena zagradama (npr. da ne zagađuje globani scope).

```
var anonimnaFunkcija = function() {  
    console.log("Ovo je anonimna funkcija.");  
};
```

```
// Pozivanje anonimne funkcije  
anonimnaFunkcija();
```

Closure

U računalnoj znanosti, closure je kombinacija objekta funkcije i dosega (scopea) unutar kojeg su varijable funkcije razriješene.

- Sve funkcije u JavaScriptu implementiraju closure tehniku
- Doseg (scope) varijabla funkcije je onaj unutar kojeg je funkcija definirana, a ne unutar kojeg je pozvana

Closure

```
function makeFunc() {  
  const name = 'Mozilla';  
  
  function displayName() {  
    console.log(name);  
  }  
  return displayName;  
}  
  
const myFunc = makeFunc();  
myFunc();
```

Funkcionalno programiranje

- Funkcionalno programiranje je jedna od programskih paradigmi, uz npr. objektno-orientirano programiranje. Više o tome u React modulu!

Funkcionalno programiranje je programska paradigma koja računanje smatra kao evaluaciju matematičkih funkcija i izbjegava promjenu stanja i promjenjivih podataka.

Funkcije

Pratite upute u Instructions.md
Trajanje vježbe: 45 minuta



Napredni JavaScript

OOP JavaScript - 1.dio

Objektno-orientirano programiranje

Objektno-orientirano programiranje je jedna od programskeih paradigmi koja koristi objekte kao modele za računanje.

Programska paradigma je način klasifikacije programskih jezika na temelju njihovih značajki.
Jezici se mogu svrstati u više različitih paradigmi.

OOP, tl;dr: Sve je objekt!

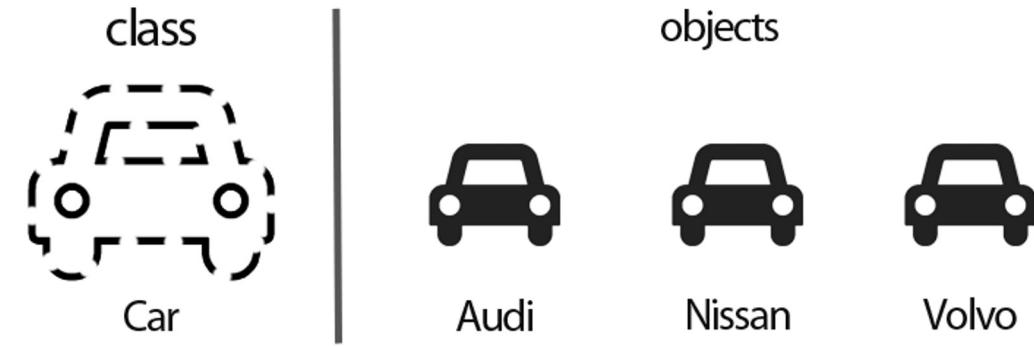
Neki poznati OOP jezici: [Java](#), C++, C#, Python, PHP...

JavaScript OOP

- JavaScript je dizajniran da bude objektno-orijentiran jezik
- Većina tipova podataka u JavaScriptu su zapravo objekti (čak i funkcije)
- Jezici koji podržavaju objektno orijentirano programiranje (OOP) obično koriste **nasljedivanje** (metoda i svojstava) u svrhu ponovne upotrebe postojećeg koda i proširivost u obliku klasa ili prototipa

Primjer: OOP u Javi (tzv. klasično nasljeđivanje)

- Baziran na klasama
- Objekti se stvaraju iz klase
- Ti objekti se nazivajuinstancama klasa



Primjer: OOP u Javi (tzv. klasično nasljeđivanje)

```
class Dog {  
    public String bark() {  
        return 'Wuf Wuf';  
    }  
}
```

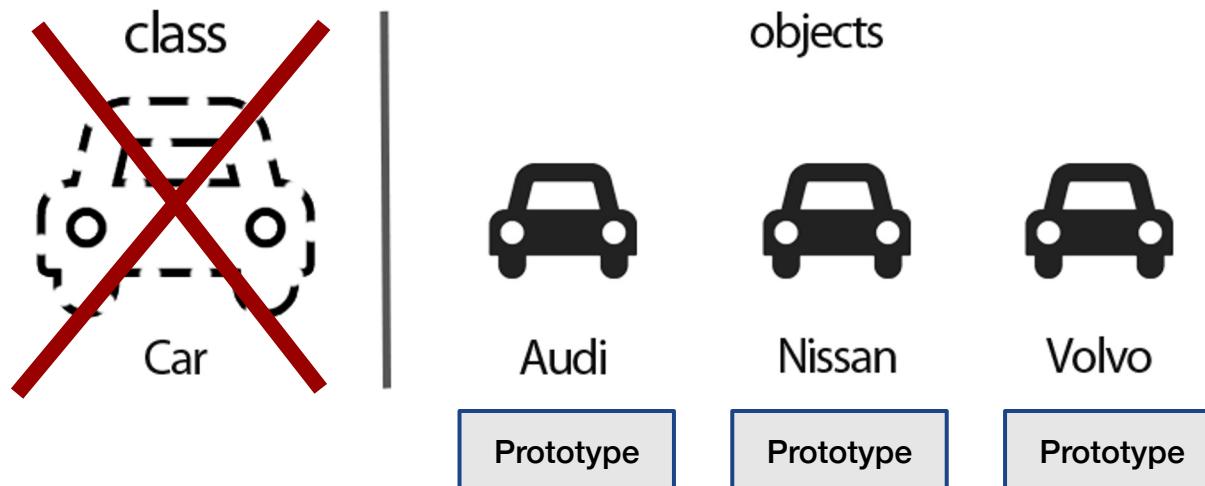
```
var dog = new Dog();  
dog.bark();
```

```
class GoldenRetriever extends Dog {  
    public String fetch() {  
        return 'Ball!';  
    }  
}
```

```
var goldenRetriever = new GoldenRetriever();  
goldenRetriever.fetch();  
goldenRetriever.bark();
```

OOP u JavaScriptu

- Baziran na prototip načinu nasleđivanja. JavaScript nema klase!
 - Pa, zapravo.. ima... 😊



Prototype objekt

- Svaki JavaScript objekt ima drugi JavaScript objekt (ili null, ali to je rijetko) povezan s njim
- Ovaj drugi objekt poznat je kao prototip, a prvi objekt nasljeđuje svojstva od prototipa

```
var person = {  
    name: 'John',  
    surname: 'Doe',  
    <prototype>: Object { ... }  
};  
  
console.log(person);
```

Probajte pokrenuti
ovaj kod, pogledajte
rezultat u konzoli.

Prototype

- "Prototype" se odnosi na svojstvo koje postoji za svaki JavaScript objekt. Svaki objekt u JavaScriptu ima referencu na svoj prototip (koji je također objekt) preko svojstva "prototype".
- Ovo svojstvo obično povezuje konstruktor funkciju (klasu) sa objektima koji su kreirani koristeći tu konstruktor funkciju.
- Na primjer, ako imate konstruktor funkciju Person, možete dodati metode ili svojstva sviminstancama tog konstruktora tako što ćete ih postaviti na njegov "prototype". **Sve instance Person objekta će naslijediti ta svojstva i metode preko prototipa.**

```
// Constructor function for creating Person objects
function Person(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

// Adding a method to the prototype of Person objects
Person.prototype.fullName = function () {
    return this.firstName + ' ' + this.lastName;
};

// Creating two instances of the Person object
var person1 = new Person('John', 'Doe');
var person2 = new Person('Jane', 'Doe');

// Calling the method from the prototype on the instances
console.log(person1.fullName()); // Output: 'John Doe'
console.log(person2.fullName()); // Output: 'Jane Doe'
```

Proto

- U JavaScriptu, `__proto__` je interno svojstvo (engl. "property") objekta koje omogućava pristup prototipnom objektu tog objekta.
- Prototipni objekt je drugi objekt koji se koristi kao model ili "predak" za datu instancu objekta.
 - Međutim, **nije preporučljivo** koristiti "proto" direktno jer se radi o internom svojstvu JavaScript enginea koji nije standardiziran i može biti nestabilan u različitim okruženjima.

```
// Create a base object
var animal = {
  type: 'Unknown',
  speak: function () {
    console.log('The ' + this.type + ' makes a sound.');
  }
};

// Create a derived object with __proto__
var dog = { __proto__: animal, type: 'Dog' };

// Now, 'dog' inherits properties and methods from
// 'animal'
dog.speak(); // Output: "The Dog makes a sound."
console.log(dog.type); // Output: "Dog"
```

Zaključak

- Ukratko, `prototype` se koristi za definiranje zajedničkih metoda i svojstava kod funkcija-konstruktora, dok se `__proto__` koristi za pristup i naslijedivanje metoda i svojstava putem prototipnog lanca kod instanci objekata.

VAŽNE NAPOMENE:

- `__proto__` je zastario, izbjegavajte njegovo korištenje - više na [MDN-u](#).
 - Umjesto `__proto__`, koristite `Object.create()` ili `Object.getPrototypeOf()` i `Object.setPrototypeOf()` (koristiti s oprezom)

OOP – 1. dio

Pratite upute u Instructions.md
Trajanje vježbe: 25 minuta

"Sposobnost razmišljanja o onome što želite u JavaScriptu i prevodenja toga u kod je ključna vještina."

- Chris Pine



Napredni JavaScript

OOP JavaScript - 2.dio

Primjer: OOP u JavaScriptu

- Za razliku od, npr. Java, JavaScript nema definirana privatna, javno dostupna, zaštićena ili privilegirana svojstva objekata, ali mogu se ostvariti kroz principe closure-a.

```
function Dog() {  
    this.hasSat = false;  
  
    this.bark = function() {  
        return 'Wuf Wuf';  
    }  
  
    this.doSit = function() {  
        this.hasSat = true;  
    }  
  
}  
  
var dog = new Dog();  
dog.bark();
```

Konstruktor

- Konstruktor je funkcija dizajnirana za inicijalizaciju novostvorenih objekata.
- Kritična značajka poziva konstruktora je da se prototip svojstvo konstruktora koristi kao prototip novog objekta.

```
function Dog() {  
    this.hasSat = false;  
    this.bark = function() {  
        return 'Wuf Wuf';  
    }  
    this.doSit = function() {  
        this.hasSat = true;  
    }  
}  
  
var dog = new Dog();  
dog.bark();
```

Riječ „new”

- Riječ `new` ispred bilo koje funkcije pretvara poziv funkcije u poziv konstruktora.
 - Stvara se potpuno novi prazni objekt
 - Novi prazni objekt povezuje se sa svojstvom prototipa te funkcije
 - Isti novi prazni objekt postaje `this` za kontekst izvršenja tog poziva funkcije
 - Ako ta funkcija ništa ne vraća, onda implicitno vraća ovaj objekt.

```
function Dog() {  
    this.hasSat = false;  
    this.bark = function() {  
        return 'Wuf Wuf';  
    }  
    this.doSit = function() {  
        this.hasSat = true;  
    }  
  
    var dog = new Dog();  
    dog.bark();
```

Riječ „this” u konstruktor funkciji

- U funkciji koja je invocirana riječju new (konstruktor funkciji), riječ **this** se odnosi na samu funkciju.
- Metode i varijable koje su deklarirane sa riječju this pripadaju objektu te funkcije.
- Ne zaboravite:
 - Kod “normalnih” funkcija riječ this se odnosi na globalni objekt.

```
// Constructor function for creating Person objects
function Person(firstName, lastName) {
    // Properties
    this.firstName = firstName;
    this.lastName = lastName;

    // Method
    this.fullName = function () {
        return this.firstName + ' ' + this.lastName;
    }
}

// Creating instances of the Person object
var person1 = new Person('John', 'Doe');
var person2 = new Person('Jane', 'Smith');

// Accessing properties and calling methods
console.log(person1.firstName); // Output: "John"
console.log(person2.lastName); // Output: "Smith"
console.log(person1.fullName()); // Output: "John Doe"
```

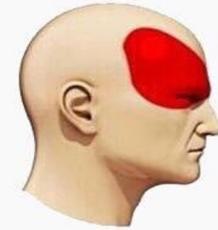
Klase

- ES6 je dodao ključnu riječ class, kako bi jezik približili klasičnoj sintaksi OOP jezika (Java, C++), ali ništa se u pozadini nije promijenilo.

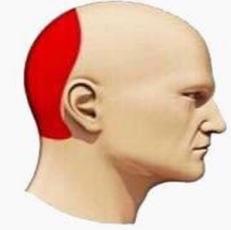
```
class Dog {  
    ...  
}
```

Types of Headaches

Migraine



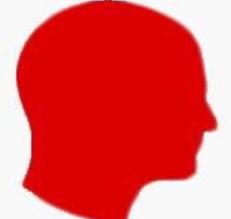
Hypertension



Stress



JavaScript



Klase

- Također, imamo par novi izraza koje nam pomožu u radu sa "klasama".
- **constructor** - metoda koja se zove prilikom inicijalizacije objekta
- **extends** - za nasljeđivanje, kao u klasičnom OOPu

```
class Golden extends Dog {  
    constructor(breed) {  
        this.breed = breed;  
    }  
}
```

Prepišite Dog i GoldenRetriever klase iz OOP - 2.dio lekcija sa novim class izrazima!

Call, bind, apply

- Svaka funkcija ima metode pozivanja, vezivanja i primjene (.call, .bind, .apply).
- Ove se metode mogu koristiti za postavljanje prilagođene vrijednosti "this" u kontekst izvršenja funkcije.
- Bind vam omogućuje da postavite ovu vrijednost sada, dok vam omogućuje izvršavanje funkcije u budućnosti, jer vraća novi objekt funkcije.

	time of function execution	time of this binding
function object <code>f</code>	future	future
function call <code>f()</code>	now	now
<code>f.call()</code> <code>f.apply()</code>	now	now
<code>f.bind()</code>	future	now

Bind()

```
const automobil = {
  brzina: 0,
  ubrzanje: function () {
    this.brzina += 10;
    console.log('Auto ubrava na' + this.brzina + ' km/h.');
  }
};

const ubrzanjeFunkcija = automobil.udrzanje;

// Sada, ako pozovemo ubrzanjeFunkcija, 'this' će biti undefined jer nema vezu s objektom 'automobil'
ubrzanjeFunkcija(); // Pogreška!

// Međutim, možemo koristiti 'bind()' kako bismo vezali funkciju za objekt 'automobil'
const vezanaUbrzanjeFunkcija = automobil.udrzanje.bind(automobil);

// Sada će 'this' u funkciji biti objekt 'automobil' kad god je pozovemo
vezanaUbrzanjeFunkcija(); // Radi ispravno
```

Prototip metode

- Prototipne metode u JavaScriptu su funkcije koje su dodane objektima putem njihovih prototipa (prototipnih objekata). To znači da su dostupne svim instancama određenog tipa objekta!
- Kada definirate funkciju kao prototipnu metodu za neki objektov prototip, svaka instanca tog objekta automatski će naslijediti tu metodu.
- To omogućuje zajedničko dijeljenje funkcionalnosti među svim objektima tog tipa, što čini kod čišćim i efikasnijim.

```
function Dog() {  
    this.hasSat = false;  
}  
  
// Dodavanje prototipne metode 'bark'  
Dog.prototype.bark = function () {  
    return console.log('Wuf Wuf');  
}  
  
// Dodavanje prototipne metode 'sit'  
Dog.prototype.sit = function () {  
    this.hasSat = true;  
}  
  
// Stvaranje nove instance objekta Dog  
var dog = new Dog();  
  
// Pozivanje prototipne metode na objektu  
dog.bark();
```

JavaScript – OOP – 2. dio

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 15 minuta

"Programiranje je kao umjetnost dizajniranja, ali s kodom."
- Damian Conway



Napredni JavaScript

Obrasci programskog koda

Obrazac (pattern)

- Predložak ili konvencija
- Rješenje za česte probleme u programiranju
 - Rješava:
 - Potrebu za ponavljanjem istog koda na više mesta
 - Kompliciran kod pojednostavljenjem apstrakcije
 - Probleme komunikacije tima pružanjem zajedničkog jezika opisivanja sličnih problema

Tipovi obrazaca

- Dizajn obrasci

Gang of Four obrasci za OOP jezike (Java, C++)

- Obrasci u kodu

Vezani za specifični programske jezike

- Antiobrasci (antipattern)

Obrasci koji kreiraju više problema nego što rješavaju

Dizajn obrasci za OOP

Singleton

Factory

Iterator

Decorator

Strategy

...

U OOP-u Singleton podrazumijeva samo jednu instancu klase. U JavaScriptu je objekt literal već singleton (nema klasa):

```
var person = {  
    name: 'John Doe'  
}
```

Dizajn obrasci za OOP

Singleton

Factory

Iterator

Decorator

Strategy

...

- Koristi se za kreaciju objekata. Ima sljedeće funkcije:
 - Pruža sučelje za stvaranje objekata koji se mogu mijenjati nakon stvaranja.
 - Logika za kreiranje objekata je centralizirana, pojednostavljuje i bolje organizira naš kod
 - Više na itnext.io/easy-patterns-simple-factory-b946a086fd7e

```
// Define a constructor function for different types of vehicles
function Car(make, model) {
  this.make = make;
  this.model = model;
}

function Bicycle(type) {
  this.type = type;
}

// Create a VehicleFactory that can create different types of vehicles
function VehicleFactory() { }

// Add a method to the factory to create cars
VehicleFactory.prototype.createCar = function (make, model) {
  return new Car(make, model);
};

// Add a method to the factory to create bicycles
VehicleFactory.prototype.createBicycle = function (type) {
  return new Bicycle(type);
};

// Usage of the factory
const factory = new VehicleFactory();
const myCar = factory.createCar('Toyota', 'Camry');
const myBicycle = factory.createBicycle('Mountain');

console.log(myCar); // Output: Car { make: 'Toyota', model: 'Camry' }
console.log(myBicycle); // Output: Bicycle { type: 'Mountain' }
```

Module obrazac

- Koristi se za **kreiranje JS paketa**, tj. samostalnih dijelova koda, koji se mogu dodavati, zamijeniti, mijenjati po volji bez opasnosti da se nešto poremeti u ostatku aplikacije/web stranice
- Par pojmljiva koje bi bilo dobro proučiti prije implementacije module obrazca:
 - Imenski prostor (namespace)
 - Samopozivajuće anonimne funkcije
 - Privatni i povlašteni članovi objekta
 - Deklaracije paketa o kojima ovisimo

```
// Modul za rad s brojevima
var BrojeviModul = (function () {

    // Privatna funkcija
    function privatnaFunkcija(x, y) {
        return x + y;
    }

    // Javne funkcije i varijable
    return {
        zbrajanje: function (a, b) {
            return privatnaFunkcija(a, b);
        },
        oduzimanje: function (a, b) {
            return a - b;
        }
    };
})();

// Korištenje modula
console.log(BrojeviModul.zbrajanje(5, 3)); // Ispisuje 8
console.log(BrojeviModul.oduzimanje(10, 4)); // Ispisuje 6
```

Import / Export Paketa

```
// math.js
export function zbrajanje(a, b) {
    return a + b;
}

export function oduzimanje(a, b) {
    return a - b;
}

// main.js
import { zbrajanje, oduzimanje } from './math.js';

console.log(zbrajanje(5, 3));
console.log(oduzimanje(5, 3));
```

Obrasci programskog koda

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Naučiti programirati je kao učiti novi jezik. Što više jezika znate, to ste bogatiji."
- Unknown



Napredni JavaScript

ES6 standard - 1.dio

ES6 (ECMAScript 6)

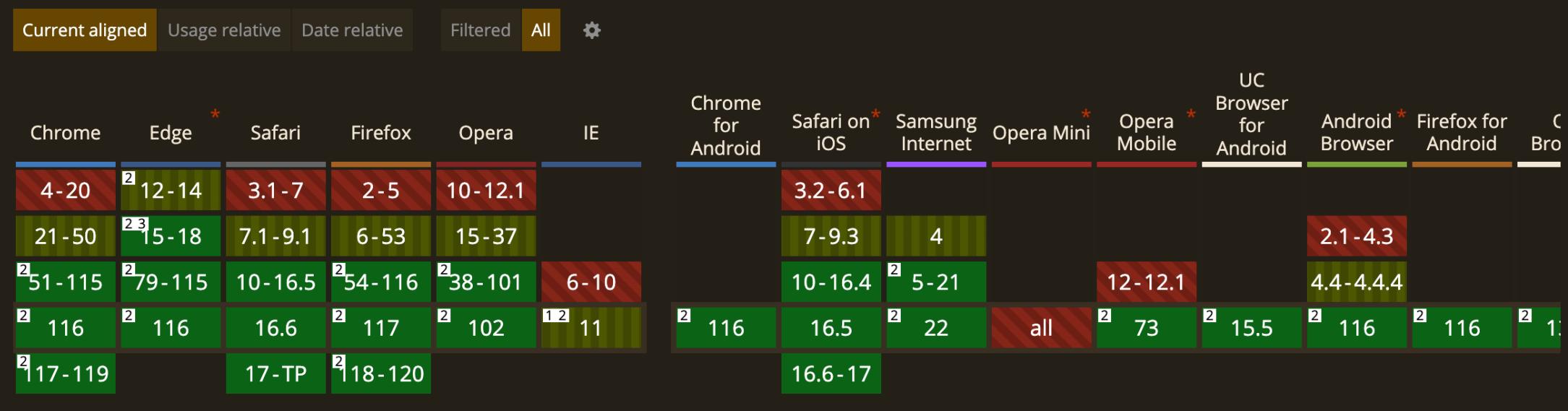
ES6 ili ECMAScript 2015 standard je specifikacija jezika ECMAScript koji definira standard za implementaciju JavaScripta.

- Za sve preglednike koji ne podržavaju ovaj standard koristimo Babel transpiler koji pretvara kod u prijašnju verziju JavaScripta.

ECMAScript 2015 (ES6) - OTHER

Usage
Global
% of all users  ?
94.23% + 1.2% = 95.43%

Support for the ECMAScript 2015 specification. Features include Promises, Modules, Classes, Template Literals, Arrow Functions, Let and Const, Default Parameters, Generators, Destructuring Assignment, Rest & Spread, Map/Set & WeakMap/WeakSet and many more.



“use strict”

Svrha direktive "use strict" je:

- Naznačiti da je kod koji slijedi (u skripti ili funkciji) strogi kod
- Strogi način rada olakšava pisanje "sigurnog" JavaScript-a
- Strogi način rada prethodno naznačenu "lošu sintaksu" pretvara u stvarne pogreške u kodu

ES6 novosti

1. Nove definicije varijabli (let, const)
2. Template literali
3. “Spread” operator
4. “Arrow” funkcije
5. Klase i moduli
6. Promises
7.

Let deklaracija

- Prednosti let deklaracije (naspram var):
 - Označuje da planiramo mijenjati tu varijablu
 - Opseg varijable je ograničen unutar bloka, ne funkcije

```
function example() {  
let x = 10; // Deklaracija varijable x unutar ove funkcije  
  
if (true) {  
    let y = 20; // Deklaracija varijable y unutar ovog bloka (if)  
    console.log(x); // Možemo pristupiti varijabli x unutar ovog bloka  
    console.log(y); // Možemo pristupiti varijabli y unutar ovog bloka  
}  
  
// Ovdje ne možemo pristupiti varijabli y jer je izvan njenog opsega (scope)  
console.log(x); // Možemo pristupiti varijabli x jer je vidljiva unutar cijele funkcije  
}  
  
example();
```

const deklaracija

- Ima slično ponašanje kao let deklaracija (block scope), osim što se vrijednost varijable ne može mijenjati.

Funkcionalno programiranje izbjegava promjenu stanja i mijenjanje varijabli, tako da je **const** deklaracija puno bolja od **let** ili **var** u 99% slučajeva.

Razlike

- Promjenjivost (Mutable vs. Immutable):
 - `let` se koristi za deklariranje promjenjivih varijabli - to znači da možete mijenjati njihovu vrijednost nakon što su deklarirane.
 - `const` se koristi za deklariranje konstanti - **vrijednost konstante ne može se mijenjati** nakon što je postavljena.

```
let broj = 5;
broj = 10; // Promjenjivo, može promijeniti vrijednost

const pi = 3.141;
pi = 4; // Greška, vrijednost konstante ne može se promijeniti
```

Razlike

- Opseg (Scope):
 - Varijable deklarirane s "let" imaju opseg (scope) ograničen na blok unutar kojeg su deklarirane (na primjer, unutar funkcije ili if petlje).
 - Varijable deklarirane s "const" također imaju blokovski opseg, ali dodatno se ponašaju kao konstante, što znači da ih ne možete re-deklarirati u istom opsegu.

```
// LET
if (true) {
    let x = 5; // x ima opseg samo unatr ove if petlje
}

console.log(x); // Greška, x nije definiran

// CONST
const y = 10;
if (true) {
    let y = 5; // Možemo deklarirati 'y' unutar if petlje jer ima drugačiji opseg
}

console.log(y); // Ispisuje vanjsku konstantu 'y' (10)
```

Template literali (string interpolation)

- Umjesto konkatenacije stringova pomoću + operatora, koristimo **backtickove**.

```
function ispisiImena(prvo, drugo, treće) {  
    console.log(prvo);  
}  
  
const x = ['Ivan', 'Marija', 'David'];  
ispisiImena(...x); // Rest operator – ubrzo
```

“Spread” operator (...)

- Omogućava nam **dekonstrukciju** niza na njegove elemente.

“Rest” operator (...)

- Omogućava nam definiciju funkcije sa **nepoznatim** brojem argumenata.

```
function ispisiImena(...imena) {  
    console.log(imena);  
}  
  
const x = ['Ivan', 'Marija', 'David'];  
ispisiImena(x);
```

ES6 Module – 1. dio

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Kodiranje je igra koja se igra s logikom i maštom."
- Unknown



Napredni JavaScript

ES6 standard - 2.dio

Za one koji žele znati više

- w3schools.com/js/js_es6.asp
- medium.com/beginners-guide-to-mobile-web-development/introduction-to-es6-c4422d3c5664

ES6 novosti

1. Nove definicije varijabli (let, const)
2. Template literali
3. “Spread” operator
4. “Arrow” funkcije
5. Klase i moduli
6. Promises

Arrow funkcije

- Omogućuju kraću sintaksu za definiranje funkcija (bez zagrada, ključnih riječi function i return, ili kombinacija).

Napišite ovu funkciju u
ES5 notaciji!

```
const bark = () => 'Wuf wuf';  
forEach(myArray, element => console.log(element));  
forEach(myArray, element => {  
    console.log(this);  
    return element;  
});
```

Obećanja (promises)

- Obećanje je objekt koji će rezultirati jednom vrijednosti koja može biti ili riješena ili neriješena (odbačena).

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

```
    if (...) {  
        return resolve;  
    }  
    return reject;  
})  
.then()  
.catch();
```

Egzekutor funkcija mora vratiti ili resolve ili reject callback funkciju.

.then reagira na stanje fulfilled promise objekta
.catch reagira na stanje rejected (npr. na neki error)

Obećanja (promises) - primjer

- Promises se mogu nositi sa negativnim ishodima akcije
- Promises su koncept u JavaScriptu koji se koristi za rukovanje asinkronim operacijama i upravljanje njihovim rezultatima ili greškama. Promises pružaju elegantan način za obradu operacija koje traju neko vrijeme, kao što su HTTP zahtjevi, čitanje datoteka ili bilo koja druga operacija koja se izvršava asinkrono.

```
function getX() {
    return Math.random() >= 0.5;
}

var wait = new Promise((resolve, reject) => {
    const x = getX();

    if (x) {
        return setTimeout(resolve, 1000);
    }
    return setTimeout(reject, 1000);
}

.then(() => console.log('Yay!'))
.catch(() => console.log('Oh no!'));
```

Obećanja (promises) - primjer

```
setTimeout(function(){  
    console.log('Yay!')  
}, 1000)
```



```
var wait = new Promise(function(resolve, reject){  
    return setTimeout(resolve, 1000);  
}).then(function() {  
    console.log('Yay!')  
});
```

Prepišite gore napisani
Promise sa arrow
funkcijama!

ES6 Module – 2. dio

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Nema kraja učenju."
- Robert E. Lee



Napredni JavaScript

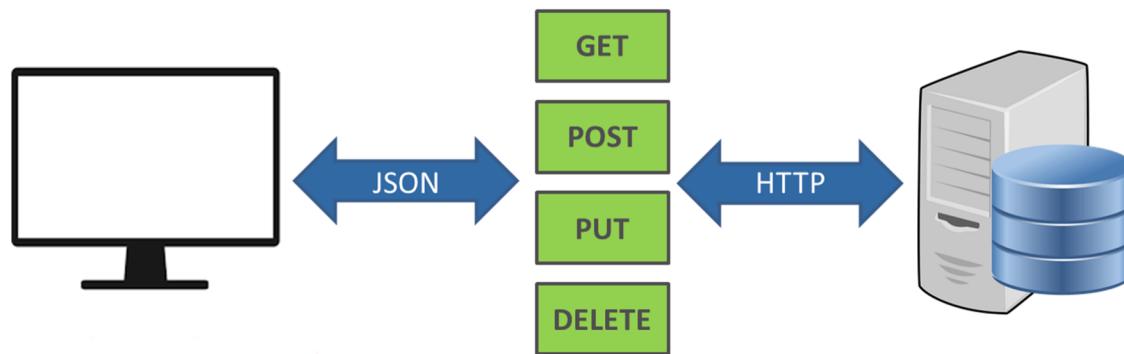
Rad s API-jima

Za one koji žele znati više

- taniarascia.com/how-to-connect-to-an-api-with-javascript
- restfulapi.net
- ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm
- w3schools.com/xml/ajax_intro.asp

API

- Application Programming Interface
- Sučelje aplikacijskog programa koje se može definirati kao skup metoda komunikacije između različitih softverskih komponenti.



HTTP request

Akcija	HTTP Metoda	Opis
Create	POST	Kreira novi resurs
Read	GET	Vrati resurs
Update	PUT/PATCH	Ažurira postojeći resurs
Delete	DELETE	Briše resurs

REST(ful) API

- REpresentational State Transfer
- Skup pravila kojih se programeri pridržavaju prilikom kreiranja API-ja. Jedno od tih pravila kaže da biste trebali biti u mogućnosti dobiti dio podataka (koji se zove resurs) kada se povezujete na određeni URL.
- GET <https://api.skufllic.net/movie/968051>

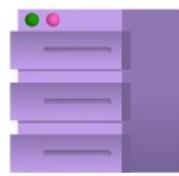
Povezivanje na REST API

- Krajnja točka (endpoint)
- Metoda
- Zaglavla (headers)
- Podaci (ili tijelo) (data/body)

Koristit ćemo **JSON format** za prijenos podataka između servera i aplikacije!

SOAP vs. REST APIs

upwork



Server



App

SOAP is like using an envelope

Extra overhead, more bandwidth required, more work on both ends (sealing and opening).

REST is like a postcard

Lighterweight, can be cached, easier to update.

SOAP

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:web="http://www.example.com/webservice">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <web:GetWeather>  
            <web:CityName>Zagreb</web:CityName>  
        </web:GetWeather>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Headers

- **Request headers** sadrži informacije o zahtjevu kao što su URL koji ste zatražili, metoda (GET, POST, HEAD), preglednik korišten za generiranje zahtjeva i druge informacije
- Primjer:
 - User-Agent: Mozilla/5.0 (<system-information>) <platform> (<platform-details>) <extensions>
 - User-Agent: "Mozilla/5.0 (Windows NT 10.0; WOW64; rv:41.0) Gecko/20100101 Firefox/41.0"
 - Više na developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent
- **Response headers** prima browser od servera nakon što korisnik pošalje zahtjev za određenu stranicu ili resurs i ono sadrži informacije kao što su kodiranje korišteno u sadržaju, poslužiteljski softver koji se koristi na poslužiteljskom stroju za generiranje odgovora i druge podatke
- Primjer:
 - nginx
 - Više na developer.mozilla.org/en-US/docs/Web/API/Response/headers

Response body

- Neobavezan dio HTTP odgovora koji sadrži podatke koje server šalje klijentu kao odgovor na zahtjev
- Tijelo odgovora sadrži obično tražene informacije, kao što su HTML stranica, **JSON** podaci, slike, ili bilo koji drugi sadržaj koji klijent treba

Request body

- Neobavezan dio HTTP zahtjeva koji sadrži podatke koje klijent šalje serveru
- Obično se koristi u zahtjevima kao što su POST, PUT, ili PATCH za slanje podataka serveru
 - Na primjer, ako ispunjavate online obrazac i pritisnete "Pošalji", podaci koje ste unijeli u obrazac bit će poslani u tijelu zahtjeva. Ovo tijelo može biti u različitim formatima, kao što su JSON, XML, HTML oblikovani podaci ili obični tekst, ovisno o potrebama aplikacije

AJAX

- Asynchronous JavaScript and XML
- Kombinacija `XMLHttpRequest` objekta ugrađenog u preglednik (za traženje podataka s web poslužitelja) i HTML DOM-a (za prikaz ili upotrebu podataka).

A_{↑↓}JAX

AJAX

Sa AJAX-om možemo bez učitavanja:

- Zatražiti podatke s poslužitelja
- Primati podatke s poslužitelja
- Poslati podatke poslužitelju

```
// Stvaranje instance
var request = new XMLHttpRequest();

// Nakon stvaranja objekta, možete postaviti različite
parametre zahtjeva kao što su metoda (GET, POST, itd.), URL
servera i asinkroni (true ili false) način izvršavanja
request.open('GET', endpoint, true);

request.onload = function () {
    // Morate definirati funkcije koje će se pozvati kad se
    dogode određeni događaji, poput uspješnog dohvaćanja
    podataka ili greške
};

//Konačno, možete poslati zahtjev prema serveru:
request.send();
```

Rad sa API-jima

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

"Nema kraja učenju."
- Robert E. Lee



Napredni JavaScript

Animacije

Za one koji žele znati više

- cssauthor.com/javascript-animation-libraries
- css-tricks.com/myth-busting-css-animations-vs-javascript
- developers.google.com/web/fundamentals/design-and-ux/animations/css-vs-javascript
- animejs.com/documentation

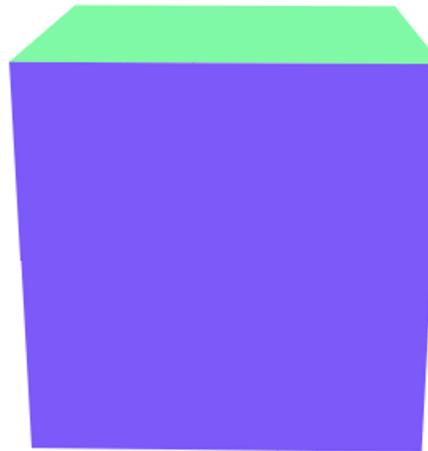
Mo.js - animation engine library



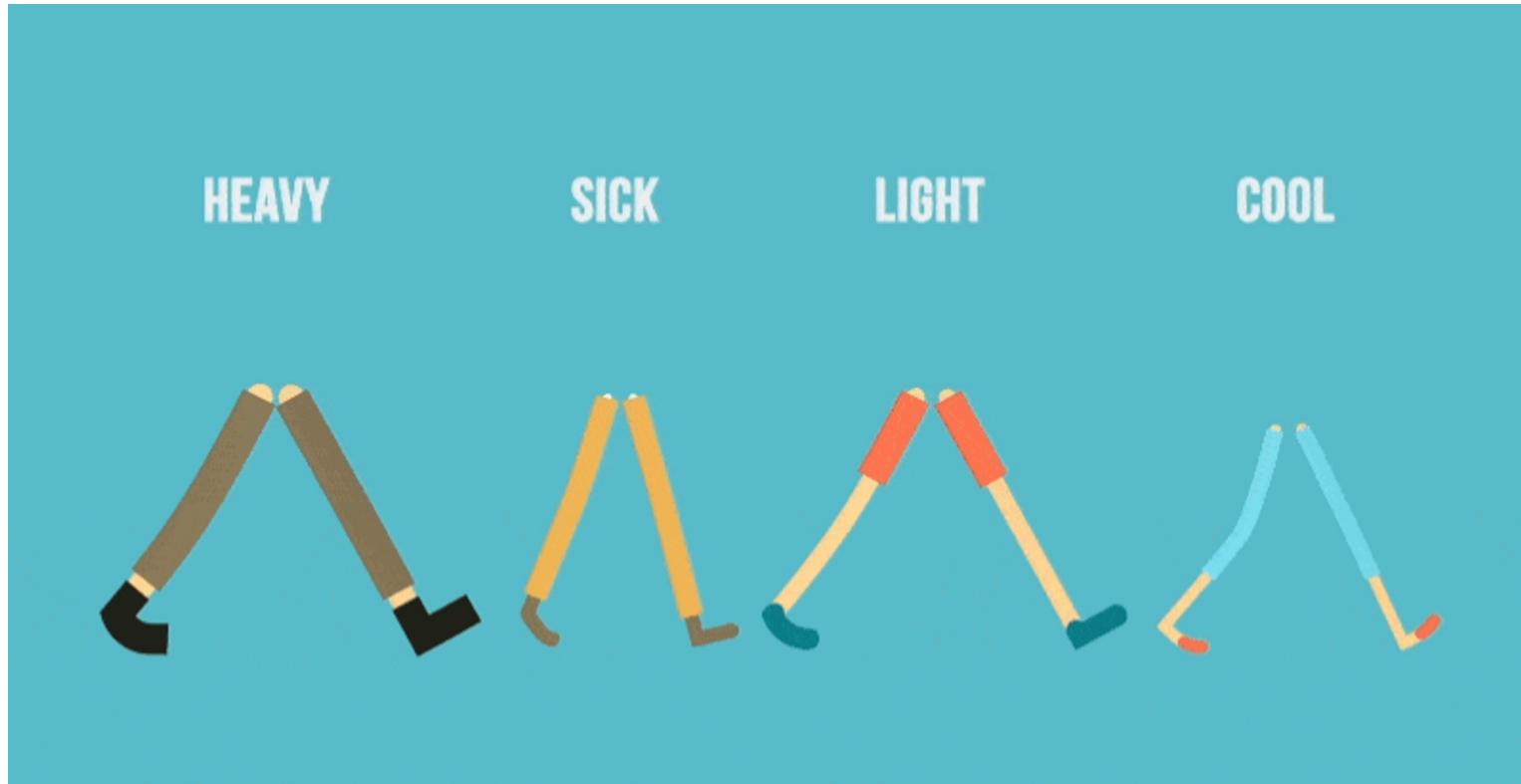
D3.js - za animiranje infografika



three.js - animacija 3D objekata



Greensock - animation engine library



JavaScript animacija vs. CSS animacija

- Zašto?
 - CSS kod neće skršiti stranicu ako nešto krene po zlu
 - CSS koristi browser hardware ubrzanja kad je moguće, JS nema tu privilegiju (obično)
 - JS kod je “skup”, tj. skuplji od CSS za obraditi
 - Ako se oslanjate na biblioteku za animiranje, može vam se ukinuti održavanje te biblioteke

Ako možete nešto animirati u CSS-u, **animirajte u CSS-u.**

JavaScript animacija vs. CSS animacija

```
var boxElement =  
document.querySelector('.box');  
  
var animation = boxElement.animate([  
  {transform: 'translate(0)'},  
  {transform: 'translate(40px,'}  
  30px)}]  
, 2000);
```

```
.box {  
  animation-name: movingBox;  
  animation-duration: 2000ms;  
}  
  
@keyframes movingBox {  
  0% {  
    transform: translate(0, 0);  
  }  
  100% {  
    transform: translate(40px, 30px);  
  }  
}
```

Elementi animacije

Element	Primjeri motoda
Timeline	Delay, Stagger, Duration, Direction, Autoplay
Kontrole animacije	Play, Stop, Pause, Reverse
Easing efekti (ublažavanje)	Linear, Spring, Cubic bezier, Elastic
Keyframes	Definicija vrijednosti na zadanom keyframe-u

Animacije - vježba

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 45 minuta

"Kvaliteta je bolja od kvantitete. Jedna dobro napisana linija koda je bolja od tisuću linija lošeg koda."

- Steve Jobs



Upravljanje pogreškama

Upravljanje kvalitetom

- 2 sata predavanja, 5 sati vježbi

Cjeline

- Kvaliteta software-a
- Otkrivanje pogrešaka
- Optimizacija
- Testiranje sa Jest-om
- Vježba

Upravljanje pogreškama

Kontrola kvalitete

Za one koji žele znati više

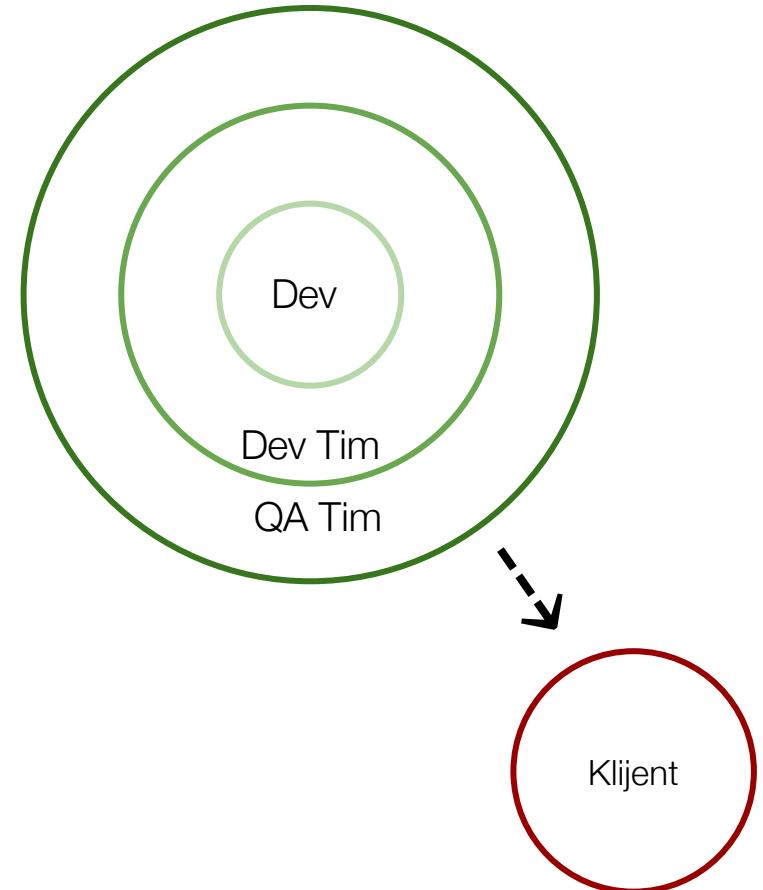
- smartbear.com/learn/automated-testing/best-practices-for-automation
- bryntum.com/blog/javascript-quality-assurance-pt-2-finding-bugs
- jslint.com
- jestjs.io
- cypress.io

Zašto testirati kod?

- Zato što želimo svojim klijentima dostaviti kvalitetan i profesionalan kod.
- Kao i sa svim u developmentu, što je ranije pronađen bug, to je manje sredstava, vremena i novca potrebno za njezino popravljanje.

Tko sve može detektirati bug?

- Developer - kod pisanja koda, kod testiranja aplikacije
- Development tim - kod testiranja aplikacije, kod provjere koda ili pomoću CI paketa
- QA tim - kod ručnog ili automatskog testiranja aplikacije
- **Klijent ili krajnji korisnik** - najgori scenarij, bug u produkciji šteti ugledu firme i košta više za popraviti

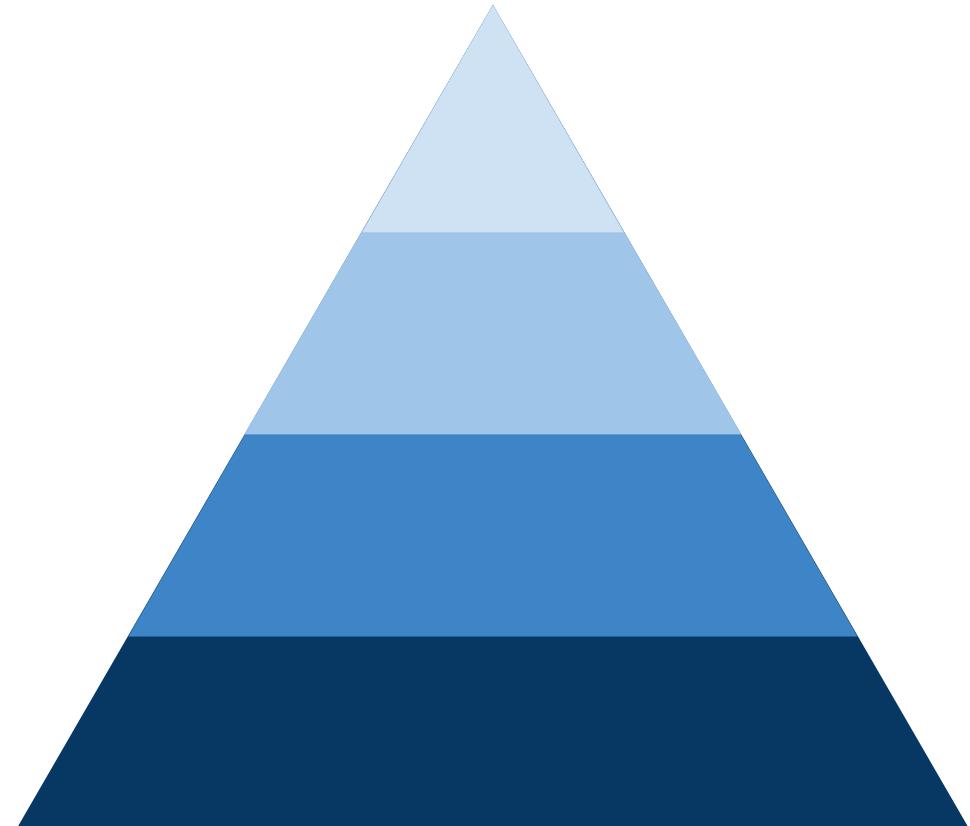


Pojmovi

- Quality control (**QC**) - kontrola kvalitete skup je aktivnosti koje kontroliraju kvalitetu proizvoda koji se razvija identificiranjem eventualnih grešaka.
- Quality assurance (**QA**) - osiguranje kvalitete softwarea, postupak QC-a je podskup koji spada pod QA.
- Test-driven development (**TDD**) - razvoj softwarea voden testiranjem pristup je razvoju koji se sastoji od pisanja testova, praćenog proizvodnim kodom i ponovnog faktoriranja po potrebi.

Razine testiranja

- **E2E** - provjera aplikacije kroz sve sustave, od backenda do frontenda
- **INTEGRACIJA** - provjera da više modula ispravno rade zajedno
- **UNIT** - provjera da individualni dijelovi (moduli) funkcioniraju (u izolaciji), alat: test framework
- **STATIČNO** - detekcija bugova i zatipaka dok se piše kod, alat: JSLint



Automatski testovi

Najviše koristi možete dobiti automatizacijom:

- Repetitivnih testova
- Testova koji imaju tendenciju biti uzrokovani ljudskom pogreškom
- Često korištenih funkcija visokog rizika
- Testova koji zahtijevaju puno vremena kad se izvode ručno



Kontrola kvalitete

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Kodiranje je lako. Razmišljanje o tome što treba kodirati je teško."
- Michael A. Butler



Upravljanje pogreškama

Otkrivanje pogrešaka

Za one koji žele znati više

- en.wikipedia.org/wiki/Lint_%28software%29
- en.wikipedia.org/wiki/Breakpoint
- developers.google.com/web/tools/chrome-devtools/javascript/reference
- jslint.com

Linting

- Linting je postupak provjere izvornog koda za programske, kao i stilističke pogreške.
- Linter je program koji podržava linting (provjeru kvalitete koda).
 - Linter uzima izvor JavaScript i skenira ga
 - Ako pronađe sintaksni ili strukturni problem, vraća poruku koja opisuje problem i približnu lokaciju unutar izvora

Linteri

Linter	Prednosti	Nedostatci
JSLint	Možete odmah početi rad s linterom	Nema konfiguracijski file
JSHint	Može se konfigurirati, samo bazični ES6	Ne možemo dodati vlastito pravilo
JSCS	Može se posve customizirati po potrebi	Ne detektira sve sintaksne bugove
ESLint	Najbolja ES6 podrška, vrlo fleksibilan	Zahtijeva konfiguraciju, spor

JSHint

Koristi .jshintrc file (JSON) za konfiguraciju pravila linanja.
Također se može koristiti inline.

```
{  
    "maxerr"      : 50,  
    "bitwise"     : true,  
    "camelcase"   : false,  
    "curly"       : true,  
    "eqeqeq"      : true,  
    "forin"       : true,  
    "immed"       : true,  
    "indent"      : 4  
}
```

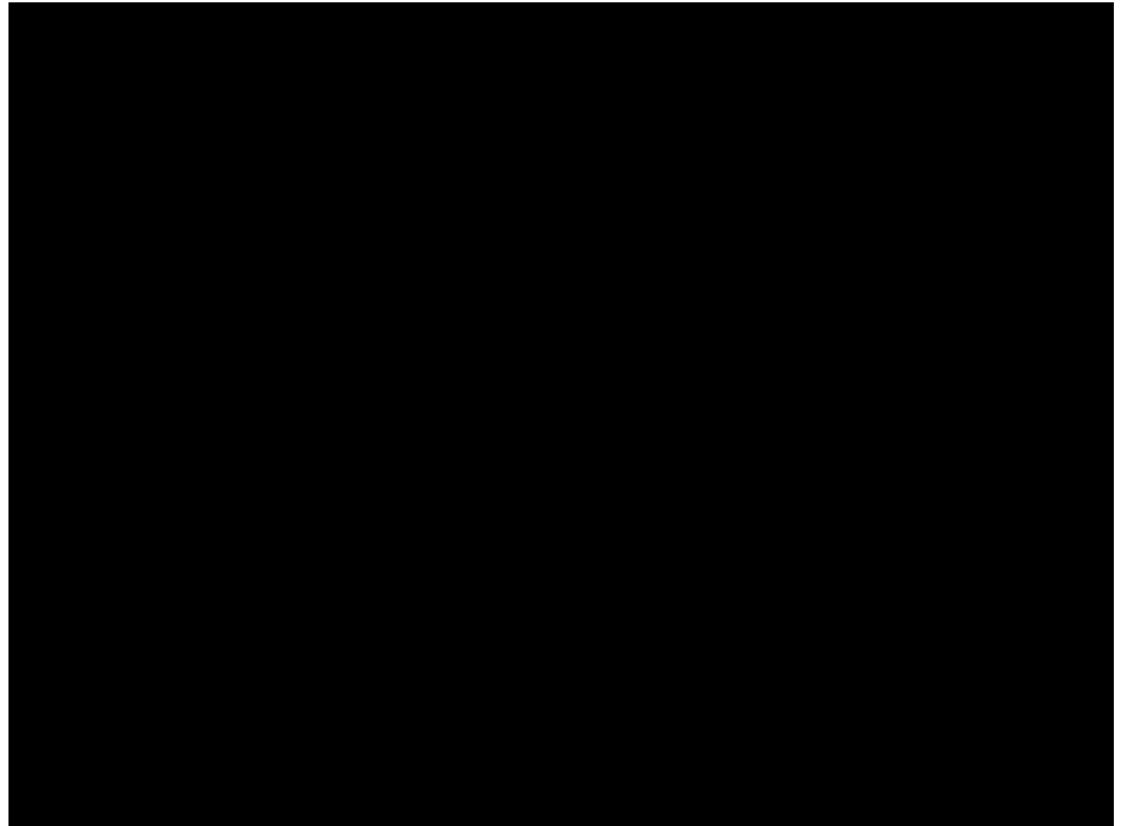
DevTools

- Chrome, Firefox, Edge DevTools skup je alata za web programere ugrađenih izravno u preglednicima i najčešće je korišten web developer alat.

Tab	Uloga
Elements	Za HTML i CSS
Console	Za debuggiranje grešaka i JS koda
Sources	Pregled svih izvora stranice i za postavljanje breakpointova
Performance	Praćenje redoslijeda izvođenja i utjecaja na brzinu stranice
Network	Prikaz redoslijeda loadanja resursa i detalja requesta

Točke prekida (breakpoints)

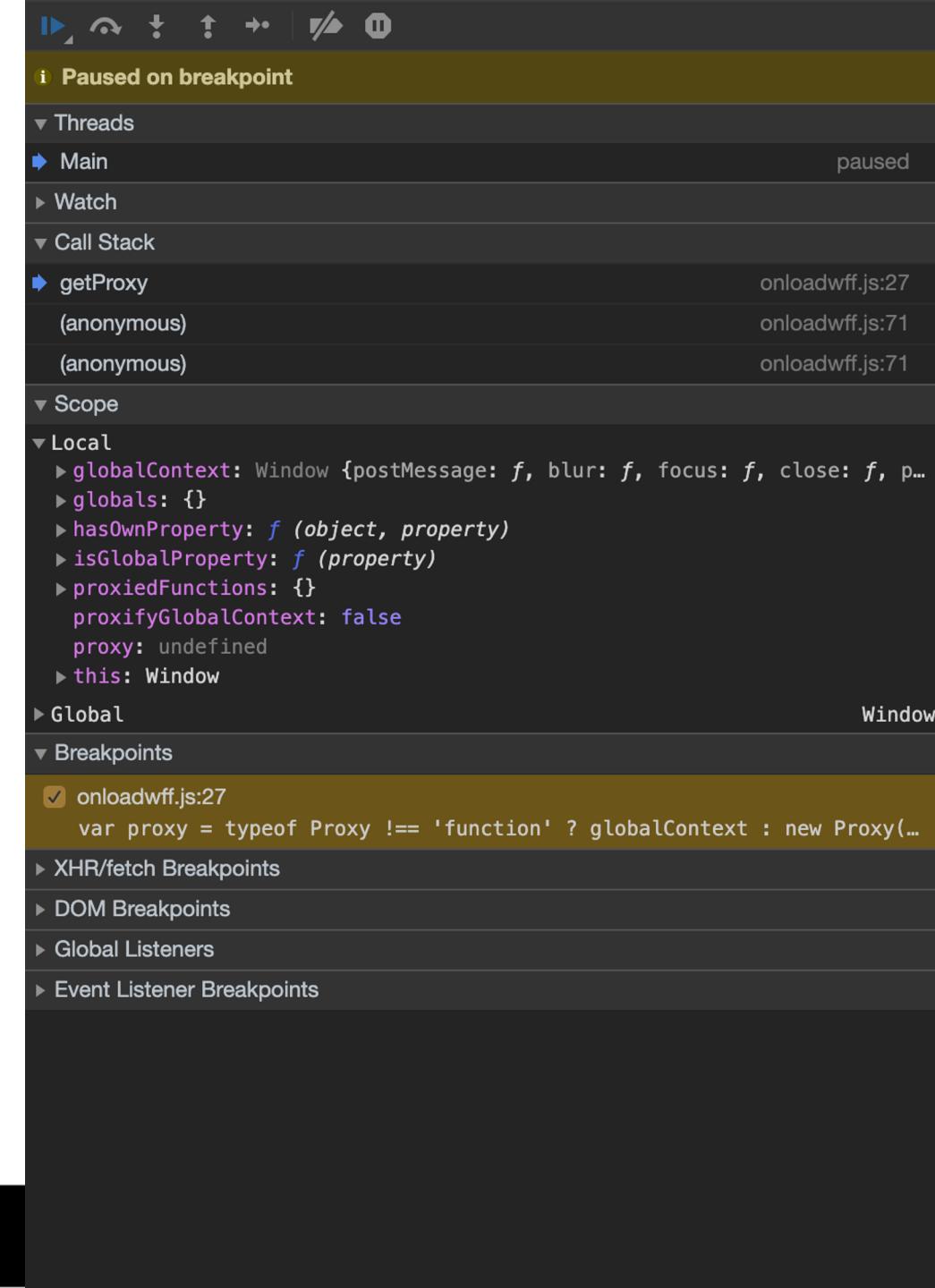
- Točka prekida je namjerno mjesto zaustavljanja u programu, postavljeno u svrhu uklanjanja pogrešaka.
- Neke od funkcija:
 - set breakpoint
 - remove breakpoint
 - resume script execution
 - pause script execution
 - step over to next function call



Google DevTools

Sources

- Prilikom breakpoint zastaja možemo dobiti informacije o:
 - Threadu na kojem se izvodi JS kod
 - Call stack-u koji je prethodio trenutnoj točci
 - Scopu varijabli



Otkrivanje pogrešaka

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 25 minuta

"Nikad ne prestaj učiti, jer su programski jezici, okoline i tehnologije uvijek u pokretu."
- Harvey Deitel



Upravljanje pogreškama

Optimizacija

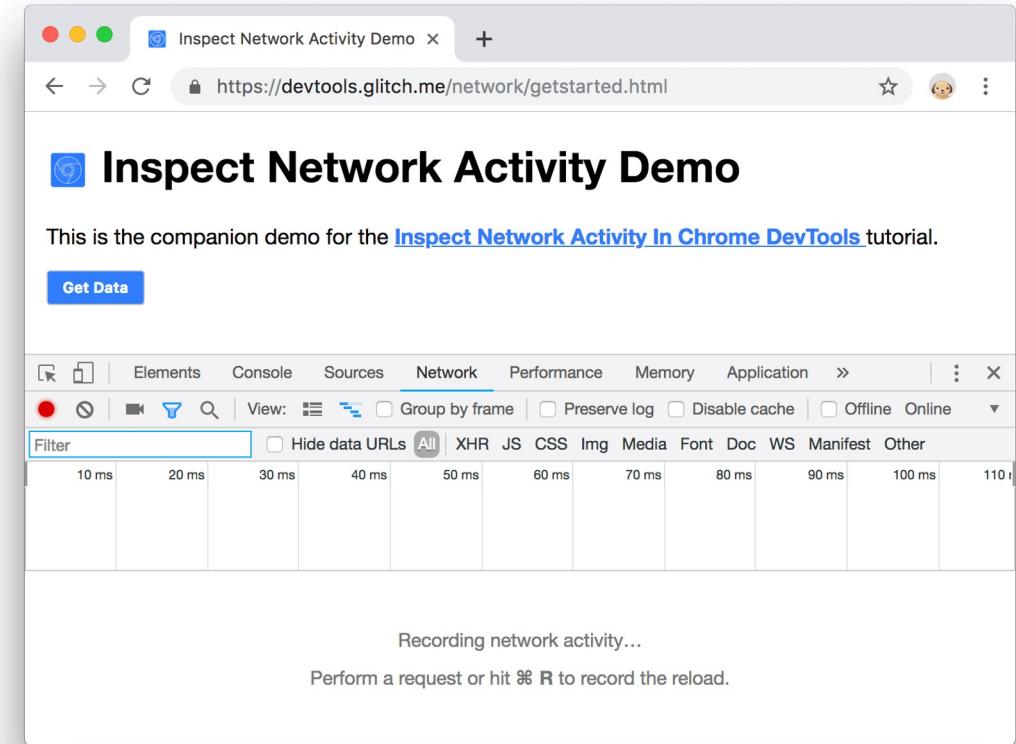
Za one koji žele znati više

- developers.google.com/web/tools/chrome-devtools/network/
- developers.google.com/web/tools/chrome-devtools/evaluate-performance/

Google DevTools

Network

- Network nam daje informacije o loadanom resursu:
 - Poredak loadanja resursa
 - HTTP status kod
 - Vrsti resursa
 - Što je uzrokovalo da se zatraži resurs
 - Koliko je trajao zahtjev

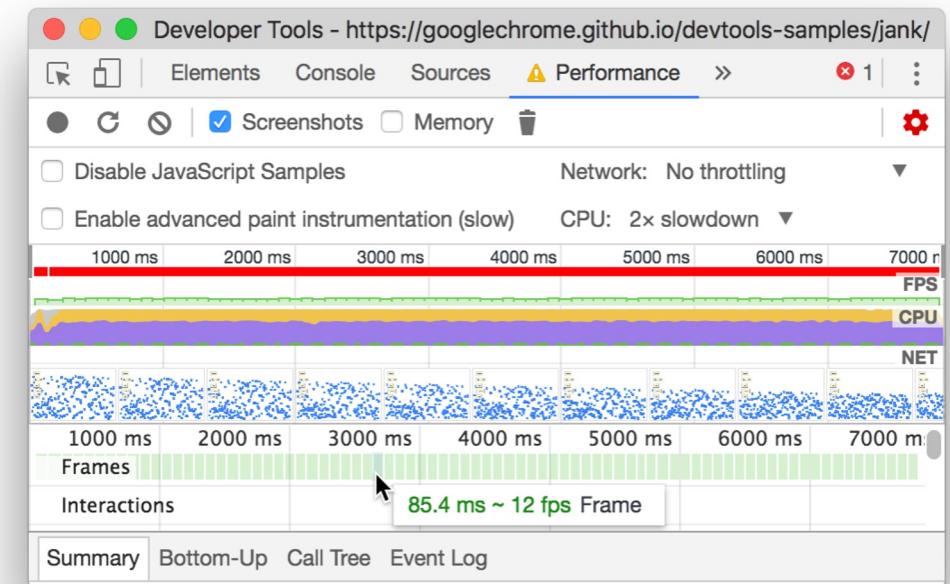


Google DevTools

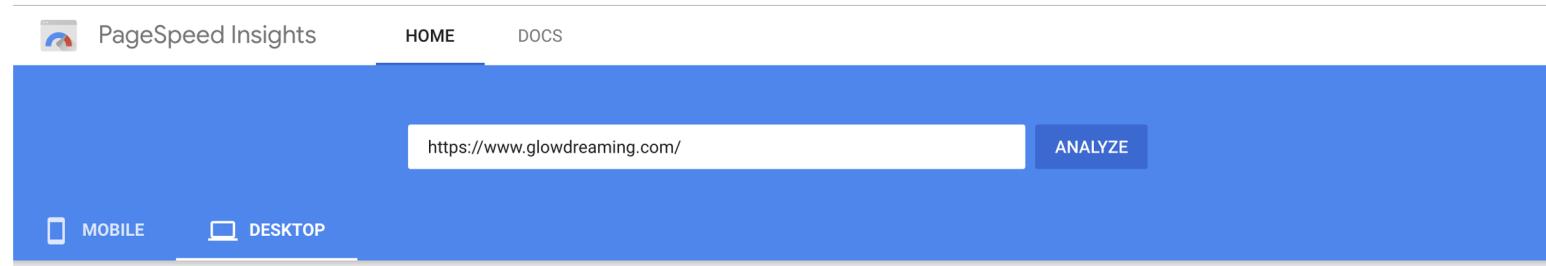
Performance

- Performance tab analizira kako vaša stranica radi dok se prikazuje (za razliku od učitavanja).
- Kompliciran je za shvatiti čak i za iskusnog developera i zahtijeva detaljno znanje o radu browsera.

Više o tome pročitajte na Chrome DevTools [referentnoj stranici](#)



PageSpeed Insights



The screenshot shows the PageSpeed Insights homepage. At the top, there's a navigation bar with the PageSpeed Insights logo, a HOME button, and a DOCS button. Below the navigation is a search bar containing the URL <https://www.glowdreaming.com/>, followed by a blue "ANALYZE" button. Underneath the search bar are two buttons: "MOBILE" and "DESKTOP". The main content area features a large circular progress meter with the number "53" in the center. Below the meter is the URL <https://glowdreaming.com/>. A legend below the URL indicates performance ranges: 0-49 (red), 50-89 (orange), and 90-100 (green). A small information icon is also present in the legend.

Field Data — Over the last 30 days, the field data shows that this page has a **Slow** speed compared to other pages in the [Chrome User Experience Report](#). We are showing the [90th percentile of FCP](#) and the [95th percentile of FID](#).

▲ First Contentful Paint (FCP) **2.6 s** ■ First Input Delay (FID) **57 ms**

41% 47% 12% 94% 5% 2%

Show Origin Summary

PageSpeed Insights

- Bilo koja stranica na internetu se može analizirati kroz Google alat zvan PageInsights. Ove su informacije važne i za SEO.
- On nam daje informacije o:
 - Ocjeni koju joj Google dodjeljuje na temelju kategorija koje prate brzinu izvođenja (za desktop i za mobile)
 - Primjećenim problemima i predloženim koracima koji se mogu poduzeti kako bi se ocjena povisila i stranica ubrzala

Don't repeat yourself

"Svako znanje mora imati jedinstven, nedvosmislen, autoritativan prikaz unutar sustava".

- Princip razvoja softvera usmjerenog na smanjenje ponavljanja uzoraka softvera, zamijenivši ga apstrakcijama ili koristeći normalizaciju podataka kako bi se izbjegla suvišnost.

Minifikacija JS-a

- Loadanje JavaScript koda (kao i HTML-a i CSS-a) se može smanjiti minifikacijom skripti, što smanjuje veličinu datoteke

```
// return random number between 1 and 6
function dieToss() {
    return Math.floor(Math.random() * 6) + 1;
}
// function returns a promise that succeeds if a 6 is
tossed
function tossASix() {
    return new RSVP.Promise(function(fulfill, reject) {
        var number = Math.floor(Math.random() * 6) + 1;
        if (number === 6) {
            fulfill(number);
        } else {
            reject(number);
        }
    });
}
// display toss result and launch another toss
function logAndTossAgain(toss) {
    console.log("Tossed a " + toss + ", need to try
again.");
    return tossASix();
}

function logSuccess(toss) {}
```



```
function dieToss(){return Math.floor(6*Math.random())+1}function
tossASix(){return new RSVP.Promise(function(a,b){var
c=Math.floor(6*Math.random())+1;6==c?a(c):b(c)})}function
logAndTossAgain(a){return console.log("Tossed a "+a+", need to try
again."),tossASix()}function logSuccess(a){}
```

Zapamtite

Za optimizaciju JavaScripta na vašoj stranici:

1. Pišite optimiziran i **DRY** JS kod
2. Učitajte JS skriptu na ispravnom mjestu u HTML-u
3. Minificirajte JS skripte na produkciji
4. Kad je primjерено, učitajte JS kod preko **CDN-a**

Optimizacija

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Najbolji način za predviđati budućnost je stvarati je."
- Alan Kay



Upravljanje pogreškama

Testiranje koda

Za one koji žele znati više

- jestjs.io
- wanago.io/2018/08/27/testing-javascript-tutorial-types-of-tests-of-unit-testing-with-jest/

Testiranje koda

- Unit testing
- Functional testing
- Integration testing
- System testing
- Usability testing
- Software performance testing
- Load testing
- Snapshot testing
- Installation testing
- Regression testing
- Stress testing
- Acceptance testing
- Beta testing
- Volume testing
- Recovery testing

Jest test framework

```
test(imeTesta, () => {  
    expect(testnaVrijednost).funkcijaPodudaranja;  
});
```

- Funkcije podudaranja: toBe, toEqual, toBeNull, toBeUndefined, toMatch, toBeClose, toContain, ...

Jest test framework - primjer

sum.js

```
function sum(a, b) {  
  return a + b;  
}  
module.exports = sum;
```

sum.test.js

```
const sum = require('./sum');  
  
test('adds 1 + 2 to equal 3', () => {  
  expect(sum(1, 2)).toBe(3);  
});
```



Mock funkcije

- Mock je tehnika zamjene dijelova koda koje ne možemo kontrolirati (npr. requesti prema API-ju) sa nečim što možemo predvidjeti i testirati.

```
test("returns undefined by default", () => {
  var mock = jest.fn();
  var result = mock("foo");

  expect(result).toBeUndefined();
  expect(mock).toHaveBeenCalled();
  expect(mock).toHaveBeenCalledTimes(1);
  expect(mock).toHaveBeenCalledWith("foo");
});
```

Jest test framework

Koraci instalacije:

- Instaliranje Jest-a preko npm-a
- Dodavanje test naredbe u package.json
- Pisanje testa (defaultno se stavljaju u __tests__ direktorij)
- Pokretanje testa nad fileom
- Ako je test prošao, super. Ako nije, onda poravak koda!

Testiranje s JEST-om

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Efikasno testiranje koda nije samo praksa, već je način razmišljanja."
- Unknown





JavaScript van
browsera

Cjeline

- JavaScript van browsera
- Progresivne aplikacije
- Mobilne aplikacije
- JavaScript na poslužiteljskoj strani

HTML5 razvojni koncepti

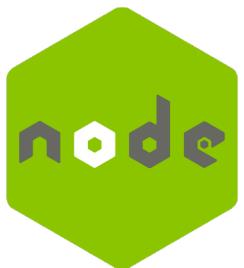
JavaScript izvan browsera

Za one koji žele znati više

- smashingmagazine.com/2017/03/beyond-browser-web-desktop-apps
- en.wikipedia.org/wiki/Node.js
- developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects

Node.js

- Node.js je open-source, cross-platformsko JavaScript run-time okruženje koje izvršava JavaScript kod izvan preglednika. Napisan je u C, C++ i JavaScript jezicima.
- Node.js omoguće "JavaScript posvuda", objedinjujući razvoj web aplikacija oko jednog programskog jezika, a ne različitih jezika na strani poslužitelja i klijenta.



Platforme

- Web preglednici
- Mobilne platforme
- Operacijski sustavi
- Dronovi
- Serveri

... JavaScript se danas može koristiti gotovo na svakoj platformi koja vam padne na pamet!

Možete pisati u JavaScriptu

Tip aplikacije	Tehnologije
Web aplikacija	Chrome, Firefox ...
Progresivna aplikacija	Service workers
Nativna mobilna aplikacija	React Native, Flutter
Hibridne mobilne aplikacije	Cordova, PhoneGap
Server aplikacije	Rhino
Desktop aplikacije	Express
E-knjige	Kindle, ...

Nativni i host objekti

- Objekti domaćina (host) su objekti koje opskrbljuje određeno okruženje.
- Na primjer, u browseru imamo Window objekt, dok node.js opskrbljuje objekte kao što je NodeList.
- Nativni objekti ili ugrađeni objekti standardni su ugrađeni objekti koje nudi Javascript. Nativni objekti ponekad se nazivaju i "globalni objekti" s obzirom da su oni objekti koje je Javascript pružio izvorno dostupan za upotrebu.

HTML5 razvojni koncepti

Progresivne aplikacije

Za one koji žele znati više

- developer.mozilla.org/en-US/docs/Web/Progressive_web_apps

Progresivne web aplikacije

- Progresivne web aplikacije koriste moderne web API-je zajedno s tradicionalnom strategijom progresivnog poboljšanja za izradu web-aplikacija na više platformi.
- Te aplikacije rade svugdje i pružaju nekoliko značajki koje im daju iste prednosti korisničkog iskustva kao i nativne aplikacije.

PWA

Progresivna aplikacija

- Sadržaj se može pronaći putem tražilica
- App se instalira, pa je dostupan na početnom zaslonu uređaja
- Možete ju podijeliti jednostavnim slanjem URL-a
- Neovisna je o povezanosti s internetom
- Progresivna, tako da je još uvijek dostupna na osnovnoj razini u starijim preglednicima
- Može slati obavijesti kad god je dostupan novi sadržaj
- Moguće je koristiti na bilo kojem uređaju sa zaslonom i preglednikom
- Sigurna, stoga je veza između vas i aplikacije osigurana protiv bilo koje treće strane koja pokušava dobiti pristup vašim osjetljivim podacima

PWA

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Kod je poezija."
- Unknown



HTML5 razvojni koncepti

Mobilne aplikacije

Za one koji žele znati više

— reactnative.dev

Nativne vs hibridne aplikacije

React Native i Flutter

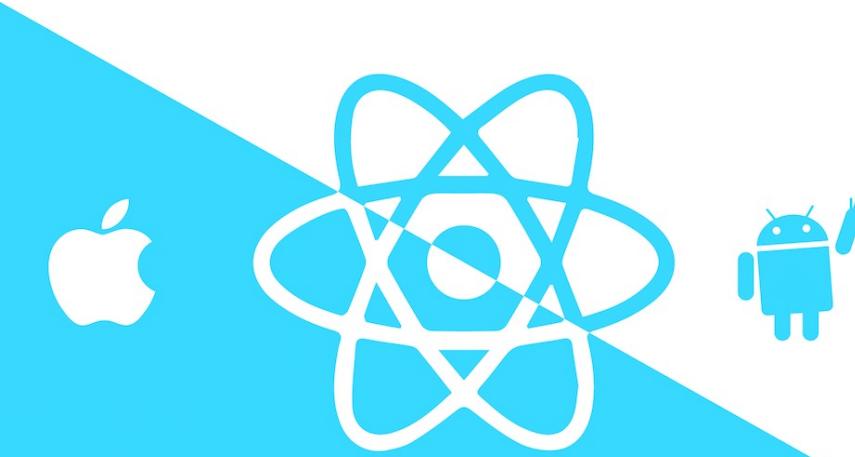
- Primjeri su tehnologija koje uzimaju aplikaciju napisanu u JavaScriptu i pretvaraju ju u nativni kod koji mobilne platforme mogu razumijeti i za koje su optimizirane.

Cordova i PhoneGap

- Uzimaju kod pisan u JavaScriptu i kreiraju dodatan “omotač” oko koda te na taj način komuniciraju sa sučeljem mobilnih platformi. Takve aplikacije su obično sporije od onih pretvorenih u nativne.

React Native

- Kreiran od strane Facebooka 2015. godine, koristi React framework kao osnovu.
- Omogućuje nam korištenje JavaScripta na Android i iOS mobilnim uređajima.



React Native

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

export default class HelloWorldApp extends Component {
  render() {
    return (
      <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
        <Text>Hello, world!</Text>
      </View>
    );
  }
}
```

Mobilne Aplikacije

React Native jezik je van opsega osnova JavaScript programiranja, stoga nastavljamo sa vježbom na našoj progresivnoj aplikaciji.

"Kvalitetan kod se ne događa slučajno. To je rezultat predanosti, strpljenja i promišljenog dizajna."

- Unknown



HTML5 razvojni koncepti

JavaScript na poslužiteljskoj strani

Za one koji žele znati više

– freecodecamp.org/news/how-to-make-a-beautiful-tiny-npm-package-and-publish-it-2881d4307f78/

Poslužiteljska strana

- JavaScript na poslužiteljskoj strani, poznat i kao Node.js, je okruženje koje omogućava izvršavanje JavaScript koda na strani poslužitelja (server-side). Tradicionalno, JavaScript je bio jezik za izvođenje koda na klijentskoj strani, u web preglednicima, ali razvoj Node.js-a je omogućio da JavaScript postane višenamjenski jezik koji se koristi i za izradu poslužiteljske strane aplikacija.

npm

- node package manager
- Pokrećemo ga kroz Terminal u folderu aplikacije.

npm config	Managira npm konfiguracijske fileove
npm init	Kreira package.json file
npm install <neki_node_paket>	Bez imena paketa instalira sve pakete u package.json Sa imenom instalira specifični paket
npm ls	Izlistava sve instalirane npm pakete
npm start	Pokreće paket ili naredbu definiranu u package.json
npm publish	Izdaje node modul na npm.js stranici

JavaScript na poslužiteljskoj strani

Pratite upute u datoteci Instructions.md
Trajanje vježbe: 30 minuta

"Programiranje je najbliže čarobnjaštvu koje ćemo ikada doći."
- Bjarne Stroustrup





Hvala na pažnji!