**Uvod u JavaScript jezik**

**00:17:38**

Jedinica: 19 od 30

**+Rezime**

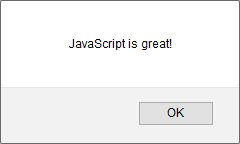
U dosadašnjem toku kursa, predstavljene su dve tehnologije koje se u kombinaciji koriste za kreiranje HTML5 aplikacija. Reč je o tehnologijama HTML5 i CSS3. Ipak, u svim lekcijama za nama, nije bilo reči o programiranju, s obzirom da se nije govorilo o programskim jezicima. HTML i CSS su opisni jezici koji se koriste da definišu sadržaj stranice i njenu vizuelnu komponentu. U takvoj konstelaciji stvari, nema govora o programiranju, već isključivo o kodiranju.

Pun potencijal jedna HTML5 aplikacija dobija korišćenjem jezika za klijentsko programiranje – JavaScript.

Na početku je najbolje da se sa JavaScript jezikom upoznamo na jednom praktičnom primeru. Sledeći primer ilustruje jednu liniju JavaScript koda:

|  |  |
| --- | --- |
| 1  2  3 | <script type="text/javascript">   window.alert("JavaScript is great!");  </script> |

Unutar specijalnih script tagova, o kojima je inače već bilo reči u uvodnim lekcijama ovoga kursa nalazi se jedna linija JavaScript koda. Kada browser u procesu parsiranja stranice dođe do prikazane linije koda, izvršiće prikaz pop-up prozora, kao na slici 19.1.



*Slika 19.1 – PopUp prozor prikazan korišćenjem JS-a*

**Šta je JavaScript?**

JavaScript je dinamički, interpretirani, programski jezik, najpoznatiji po korišćenju kao skript jezik web stranica. JavaScript je zasnovan na prototipovima, podržava objektno orijentisano, imperativno i deklarativno programiranje.



*Slika 19.2 – Nezvanični logo JS-a*

Izvorni kod napisan JavaScript jezikom interpretira se od strane browsera, te se na taj način sprovodi izvršavanje logike definisane ovim jezikom. S obzirom da postoji veliki broj browsera i da je implementacija JS interpretera prepuštena proizvođačima istih, jasno je da je neophodno postojanje određenog standarda, kada su u pitanju sintaksna pravila JS jezika. Tako je JavaScript jezik zasnovan na standardu koji se naziva ECMAScript.

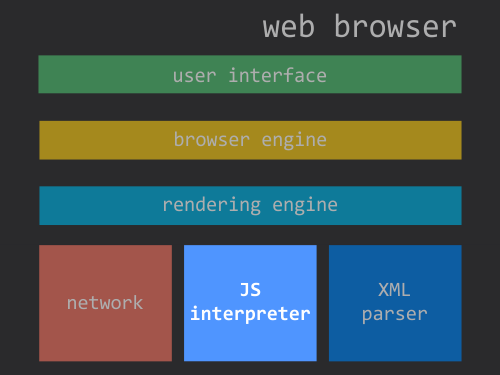
**ECMAScript**

ECMAScript je jezička specifikacija koja čini osnovu JavaScripta. ECMAScript je standardizovan od strane internacionalne neprofitne organizacije *Ecma International*, koja se brine o razvoju ove specifikacije. Tako su tokom vremena razvijane razne verzije ove specifikacije, sve od 1997. godine, kada se pojavila prva verzija ECMAScripta. Poslednjih godina primenjuje se pravilo jednogodišnjeg osvežavanja ECMAScript specifikacije.

ECMAScipt specifikacija veoma je bitna za kompanije koje žele da omoguće izvršavanje JavaScript jezika unutar svojih softverskih proizvoda. Naravno, primarno se misli na proizvođače web browsera, stoga interpreteri JS jezika, koji su sastavni deo web browsera, moraju da poštuju pravila definisana ECMAScript specifikacijom. Tako se dolazi do raznih implementacija JS jezika od strane interpretera JS koda. Neki od najpoznatijih JS interpretera su [SpiderMonkey](https://www.link-elearning.com/linkdl/opisPojma.php?id=144308), [V8](https://www.link-elearning.com/linkdl/opisPojma.php?id=144309), [JavaScriptCore (Nitro)](https://www.link-elearning.com/linkdl/opisPojma.php?id=144310) i [Chakra](https://www.link-elearning.com/linkdl/opisPojma.php?id=144311).

**Kako JavaScript funkcioniše?**

U prethodnim pasusima, već su pomenute neke vrlo značajne komponente koje učestvuju u funkcionisanju JavaScript jezika. JavaScript jezik zasnovan je na ECMAScript specifikaciji, a izvršava se unutar browsera, od strane specijalnih komponenti koje se nazivaju interpreteri. Slika 19.3 ilustruje strukturu browsera, sa posebnim akcentom na JS interpreter.



*Slika 19.3 – Struktura web browsera*

Kao i svaki programski jezik, JavaScript poseduje veliki broj rezervisanih reči, kojim se kreira logička struktura programa. Takođe, svaki jezik poseduje i skup pravila kojima se kod piše i takav skup pravila se drugačije naziva sintaksa jezika. Sintaksa jezika je, kao što je malopre rečeno, definisana i usklađena ECMAScript specifikacijom. Interpreter je komponenta koja mora da poznaje sintaksu jezika koji interpretira. To je slučaj i sa JS interpreterima. Kada browser prilikom parsiranja web stranice naiđe na JS kod, posao interpretiranja takvog koda prepušta se interpreter komponenti, koja na osnovu sintakse jezika utvrđuje kakav efekat je potrebno da napisani kod proizvede.

**Razvoj JavaScript jezika**

JavaScript jezik kreiran je u maju 1995. godine od strane Brendan Eicha, koji je u to vreme radio za kompaniju [Netscape](https://www.link-elearning.com/linkdl/opisPojma.php?id=144312). Uzimajući u obzir veoma spore internet konekcije, koje su korišćenje u to vreme, bilo je potrebno napraviti tehnologiju koja bi podatke koji se prosleđuju serveru validirala na klijentu, bez potrebe za slanjem zahteva i gubljenjem nepotrebnog vremena. Upravo je to bila jedna od osnovnih funkcija novokreiranog jezika JavaScript.

JavaScript nije oduvek bio poznat pod tim imenom. Prvobitno ime je glasilo Mocha, da bi u septembru 1995. bilo promenjeno u *LiveScript*. Nakon dobijanja dozvole od kompanije [Sun](https://www.link-elearning.com/linkdl/opisPojma.php?id=144313), novonastali jezik dobija svoje konačno ime – *JavaScript*. Imenovanje je bilo isključivo marketinški orijentisano, s obzirom da je u to vreme programski jezik Java uživao ogromnu popularnost. Tako budući najpopularniji klijentski skriptni jezik dobija svoje konačno ime.

Godine 1996. i 1997. kreira se ECMA specifikacija, kako bi ostali proizvođači browsera bili u mogućnosti da implementiraju podršku za jezik JavaScript. Sve to dovodi do objavljivanja zvanične ECMA-262 specifikacije kao opšteprihvaćenog standarda, a JavaScript postaje najpopularnija implementacija takvog standarda.

Ni Microsoft korporacija, proizvođač najpopularnijeg web browsera u to vreme, nije bila izuzetak po pitanju implementacije JavaScripta u svoj browser. Ipak, Microsoft se odlučio da svoju implementaciju ECMAScript standarda nazove JScript, prevashodno zbog problema vezanih za autorska prava na ime *Java*. U početku, Microsoft je sarađivao i implementirao funkcionalnosti definisane ECMAScript specifikacijom, ali je tokom vremena postalo jasno da nema nameru da u potpunosti sarađuje. Tako se Microsoft implementacija ECMAScripta u izvesnoj meri razlikovala od implementacija ostalih kompanija. Izvesne razlike postoje i danas.

Godina 2005. predstavlja još jednu značajnu prekretnicu u razvoju JavaScripta. To je godina kada je definisana kovanica **Ajax**, koja predstavlja set tehnologija za kreiranje web aplikacija, gde JavaScript kao jezik ima centralnu figuru. To rezultuje intenzivnim uključenjem open-source zajednice u razvoj brojnih biblioteka zasnovanih na JavaScript jeziku. Tako nastaju biblioteke kao što su jQuery, Prototype, Dojo i druge.

Danas je JavaScript jezik koji je prešao dug put, od jednostavnog validatora podataka na klijentskoj strani do bogatog jezika, bez koga se ne može zamisliti ozbiljna web aplikacija. Tako JavaScript danas omogućava asinhrono slanje zahteva, manipulaciju HTML elementima, rukovanje CSS stilovima, definisanje animacije, upravljanje lokacijom, kontrolu multimedije i još mnogo toga.

**JavaScript jezik se prvi put pojavio unutar browsera:**

 Netscape

 Chrome

 Opera

 Mozilla

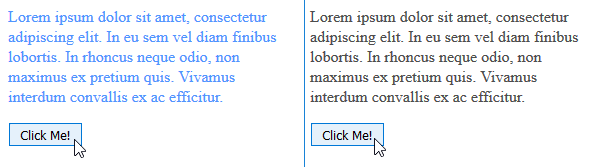
**Prvi JavaScript primer**

Nakon uvodnog izlaganja o jeziku JavaScript, vreme je da se pažnja preusmeri na jedan praktičan primer koji ilustruje jednu od mogućnosti JavaScript jezika.

|  |
| --- |
| **Napomena**    *Cilj narednog primera je kratka demonstracija jedne od mogućnosti JavaScript jezika. Potpunije razumevanje narednog primera i kreiranje sličnih funkcionalnosti zahteva savladavanje jezičkih principa JavaScripta.* |

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | <body>   <p id="text">Lorem ipsum dolor sit amet, consectetur adipiscing elit. In eu sem vel diam finibus lobortis. In rhoncus neque odio, non maximus ex pretium quis. Vivamus interdum convallis ex ac efficitur.</p>   <button id="button" type="button">Click Me!</button>     <script type="text/javascript">      document.getElementById("button").addEventListener("click", styleText);      var counter = 0;      function styleText(){       counter++;       if(counter%2 == 1)     {      document.getElementById("text").style.color = "#4F95FF";     }     else{      document.getElementById("text").style.color = "#4D4D4D";     }    }   </script>  </body> |

Prikazani primer sadrži jedan paragraf i jedan button element. Klikom na button element menja se boja teksta paragrafa između plave i sive, naizmenično. Slika 19.4 ilustruje takvu situaciju.



*Slika 19.4 – Klikom na dugme, menja se boja teksta*

Kada se prvi put klikne na dugme *Click Me!*, boja teksta postaje plava. Sledeći klik tekst paragrafa boji u sivo, pa zatim opet u plavo i tako naizmenično.

Prikazani primer je samo kratka demonstracija jedne od mogućnosti JavaScript jezika. Mogućnosti su praktično ograničene samo maštom i umešnošću programera. Da bismo bili u stanju da kreiramo funkcionalnosti slične prikazanoj, neophodno je prvo se pozabaviti osnovnim jezičkim pravilima JavaScript jezika.

**Gde pisati JavaScript?**

Pre nego što se počne sa pisanjem koda, potrebno je znati gde je moguće njega pisati. Slično kao i kod CSS-a, i JavaScript kod se može pisati na nekoliko različitih lokacija. Generalno, postoje tri načina za uključivanje JavaScript koda u jedan HTML dokument:

* implementacija JavaScript koda između tagova <script> i </script>
* uključivanje JavaScript koda sa eksternog izvora
* implementacija JavaScript koda kroz proceduru za obradu događaja

|  |
| --- |
| **Napomena**    *Primeri u nastavku lekcije se odnose na prikazivanje gde se JavaScript može pisati, sama funkcionalnost koda će vam biti detaljnije objašnjena u narednim lekcijama.* |

**Implementacija JavaScript koda između tagova <script> i </script>**

Da bi se JavaScript kod implementirao u HTML dokument, potrebno je naglasiti početak i kraj skripte navođenjem tagova:

|  |  |
| --- | --- |
| 1  2  3 | <script>   //JavaScript code  </script> |

Ovakvo definisanje tagova za pisanje JavaScript koda funkcionisaće bez problema u HTML5 dokumentima, jer se od ove verzije JavaScript smatra podrazumevanim script jezikom. Ipak, kako bi se osigurala kompatibilnost i sa dokumentima starije verzije HTML-a, potrebno je postaviti i atribut type:

|  |  |
| --- | --- |
| 1  2  3 | <script type="text/javascript">   //JavaScript code  </script> |

Upravo ovaj pristup uključivanja JavaScript koda je iskorišćen u prvom primeru, u prethodnom pasusu ove lekcije. Obično se script blok sa JavaScript kodom smešta na kraj body elementa, ali to nije pravilo. Sledeći primer ilustruje script element sa JavaScript kodom u strukturi jedne HTML stranice:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <!DOCTYPE html>  <html>  <head>  <title>First page</title>  </head>  <body>    <script type="text/javascript">  alert('Hello!');  </script>    </body>  </html> |

U prikazanom primeru, script element je jedini element unutar body dela. Ipak, u praksi će HTML dokument sadržati brojne elemente, i u takvoj situaciji je veoma bitno gde se piše script element sa JavaScript kodom. Naime, JavaScript kod se izvršava onoga trenutka kada browser do njega dođe prilikom parsiranja stranice. Pritom, browser čita stranicu od vrha ka dnu.

Za demonstriranje efekta postavljanja JS koda na različite pozicije unutar HTML dokumenta, može da posluži primer iz poglavlja *Prvi JavaScript primer*. Ukoliko se u ovom primeru script element sa JS kodom navede na početku body dela, pre ostalih HTML elemenata, prilikom njegovog izvršavanja doći će do greške (slika 19.5).

https://www.link-elearning.com/linkdl/coursefiles/1166/PH5JC_19_05.jpg

*Slika 19.5 – Greška prilikom izvršavanja JS koda*

Greška sa slike 19.5 jasno stavlja do znanja da JavaScript kod nije uspeo da pronađe traženi element. Razlog je i više nego očigledan. S obzirom da se stranica parsira od vrha ka dnu, u trenutku izvršavanja JS koda, browser još nije izvršio parsiranje elementa, te stoga JS kod nije u mogućnosti da pronađe element po id vrednosti. Zbog ovoga je veoma bitna lokacija na koju se JS kod smešta.

Jedan HTML dokument nije ograničen na jedan script element sa JS kodom. Tako je u jednom HTML dokumentu moguće imati proizvoljan broj script elemenata:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | <head>  <script type="text/javascript" >  document.write("JavaScript - HEAD");  </script>  </head>  <body>  <h1>Some text 1</h1>    <script type="text/javascript" >  document.write("JavaScript - BODY 1");  </script>    <p>Some text 2</p>    <script type="text/javascript" >  document.write("JavaScript - BODY 2");  </script>    <p>End of Document</p>  </body> |

**Uključivanje JavaScript koda sa eksternog izvora**

Prilikom razvijanja ozbiljnih aplikacija bilo kog tipa, nastoji se postići što bolja razdvojenost između različitih slojeva aplikacija. Kod HTML5 aplikacija, ti slojevi se mogu podeliti na:

* strukturni sloj – HTML elementi
* sloj stilizacije – CSS stilovi
* sloj klijentske logike – JS kod

Nastoji se da se kod svih navedenih slojeva drži zasebno, kako bi se na taj način popravila preglednost i podigla produktivnost. Baš kao što se i CSS opisi mogu smestiti u zaseban fajl, i JavaScript kod nije izuzetak.

Element script, koji je već korišćen u prethodnim primerima, podržava atribut src, kojim je moguće definisati URL fajla koji sadrži JS kod koji je potrebno uključiti u HTML dokument. Tako se uključivanje eksternog fajla sa JS kodom u HTML dokument može obaviti veoma jednostavno, korišćenjem samo jedne linije koda:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <head>  <title>Page 1</title>  </head>  <body>  <script type="text/javascript" src="popup.js"></script>  </body> |

U primeru je unutar body dela definisan script element sa src atributom, čija vrednost ukazuje na putanju do fajla koji sadrži JS kod. Takav fajl može imati sledeći sadržaj:

|  |  |
| --- | --- |
| 1 | alert('Hello'); |

Kao što se može primetiti, eksterni fajl sa JS kodom poseduje samo jednu liniju koda, i pritom kod nije potrebno smeštati unutar bilo kakvih blokova. Takođe, fajl koji sadrži JavaScript kod uglavnom poseduje ekstenziju .js, što nije obavezno da bi kod ispravno funkcionisao.

Korišćenje eksternih JavaScript fajlova nije dobro rešenje ako se radi o malom kodu koji se koristi samo na jednoj stranici. U takvom slučaju, browser će napraviti jedan zahtev više, s obzirom da će, pored .html fajla, morati da učita još jedan dodatni fajl sa JS kodom.

U takvim situacijama je bolje kod napisati u <script> tagovima na onim delovima stranice gde je to potrebno. Ipak, korišćenje eksternih JS fajlova može imati brojne, izuzetno značajne prednosti:

* razdvajanje velike količine JS koda od HTML-a
* korišćenje istog JS koda na više stranica
* JS kod unutar eksternog fajla lako se kešira od strane browsera, te na taj način nije neophodno svaki put iznova njega učitavati
* korišćenje eksternih JS fajlova sa drugih servera

**Implementacija JavaScript koda kroz proceduru za obradu događaja**

Poslednji način na koji je moguće uključiti JS kod u stranicu veoma je sličan uključivanju inline stilizacije u HTML dokument. Naime, na HTML elementima je moguće definisati slušač događaja (engl. event listener), koji zatim aktivira određeni JS kod, kada dođe do pojave nekog događaja. Sledeći primer to ilustruje:

|  |  |
| --- | --- |
| 1 | <p onclick="alert('Hello!'); alert('Hello, again!')">Click Me</p> |

U primeru, na p elementu, definisan je atribut onclick, a kao njegova vrednost je naveden JS kod. Na ovaj način, kada korisnik klikne na ovakav p element, pojaviće se dva popup prozora, jedan za drugim.

Cilj navedenog primera odnosi se na prikazivanje gde se JavaScript može pisati, dok će razumevanje koda i prikaz njegove aktivacije u browseru biti jasniji nakon razumevanja narednih lekcija.

|  |
| --- |
| **Napomena**    *Ovakav način implementiranja JS koda u HTML dokument je potrebno maksimalno izbegavati, s obzirom da otežava preglednost i testiranje. Takođe, pretplatu na različite događaje moguće je postići na mnogo elegantniji način, koji će biti prikazan nešto kasnije.* |