

AJAX

Internet programiranje
IV godina, ETF Banjaluka

AJAX

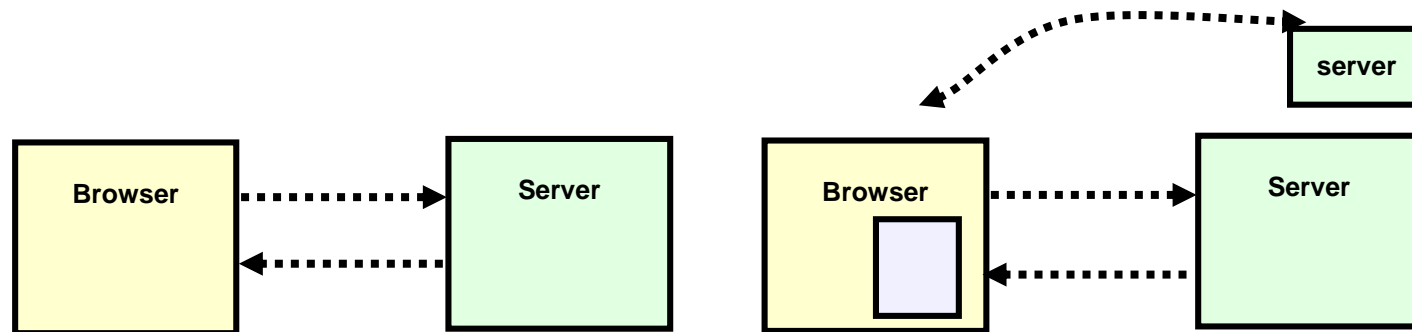
- ▶ Ajax (Asynchronous JavaScript and XML) – termin koji opisuje tehniku razvoja web aplikacija koje su bogatije po svom sadržaju i ugodnije za rad od standardnih web aplikacija
- ▶ razvijen početkom 2005. godine
- ▶ ogromna ekspanzija
- ▶ razlozi pojavljivanja:
 - klasične Web aplikacije bazirane na HTML + HTTP (POST i GET zahtjevi) “nisu interaktivne”
 - sve više zahtjeva za aplikacijama koje se izvršavaju u okviru Web čitača, bez instalacije
- ▶ “aktivni” sadržaj u okviru web čitača:
 - Java apleti – nisu u potpunosti uspjeli – nemaju interakciju sa HTML kodom
 - Flash i Flex – ozbiljna alternativa, nije univerzalno podržana
 - podrška za Flash do kraja 2020. godine – Adobe
 - Druge tehnologije:
 - Microsoft Silverlight (end of life – 2012, još podržan u IE i Operi), JavaFX, Adobe AIR

AJAX

- ▶ Ajax nije programski jezik
- ▶ Ajax nije nova tehnologija
 - koristi se duži niz godina
 - postala je popularna sa novim modernijim browser-ima i novijim Web aplikacijama
 - Google Search, Google Maps, Google Gmail, Google Suggest, MSN Mail
- ▶ Ajax koristi JavaScript da se ponovo obrati serveru
 - manje kompletnih obraćanja serveru
 - ne zamjenjuje se početna stranica
 - JavaScript se koristi da pošalje XML zahtjev
 - Server kasnije šalje rezultate nazad
 - JavaScript zamijeni dio DOM modela
- ▶ Ajax nije framework ili biblioteka
 - to je application pattern
 - mnogi framework-ci ga podržavaju

AJAX

- ▶ Ajax omogućava da web stranice budu update-ovane asinhrono, razmjenom malih količina podataka između servera i klijenta, pri čemu se osvježavaju samo dijelovi web stranice, a ne čitava stranica
- ▶ kod tipičnih Web aplikacija, svaki zahtjev dovodi do kompletnog osvježavanja stranice browser-a



- ▶ Ajax aplikacija počinje izvršavanje na isti način, a nakon inicijalnog učitavanja stranice, JavaScript kod dovlači dodatne podatke u pozadini, i osvježava specifične regione stranice

AJAX



etf banja luka

etf banja luka

etf banja luka upis 2015

etf banja luka rang lista



Inspector Console Debugger Style Editor Performance Memory Network Storage DOM

Filter URLs

Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms	2.56 s	5.12 s
200	GET	search?client=psy-ab&ni=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	902 B	417 B	→ 96 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	1.05 KB	944 B	→ 88 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	907 B	506 B	→ 86 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	959 B	586 B	→ 84 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	927 B	554 B	→ 82 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	917 B	540 B	→ 87 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	917 B	552 B	→ 86 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	929 B	588 B	→ 90 ms		
200	GET	search?client=psy-ab&hl=en-BA&gs_rn=64&gs_ri=ps...	www.google.com	xhr	json	955 B	737 B	→ 89 ms		

11 requests | 6.74 KB / 10.14 KB transferred | Finish: 8.32 s

AJAX

- ▶ programerski model Ajax-a fokusira se na klijenta – većina Ajax aplikacija su kompletno serverski nezavisne (može da bude J2EE, PHP, ASP.NET, SOAP, ...)
- ▶ AJAX je baziran na standardima i predstavlja kombinaciju:
 - XMLHttpRequest objekta (za asinhronu razmjenu podataka sa serverom)
 - JavaScript/DOM (za prikaz i interakciju)
 - CSS
 - XML (često se koristi za razmjenu podataka)
- ▶ AJAX aplikacije su nezavisne od čitača i platforme
- ▶ kod tehnologija prije Ajax-a slični rezultati su se mogli postići pomoću JavaScripta i sakrivenih frame-ova, ali Ajax podržava standardniji (i razumljiviji) pristup

AJAX

- ▶ 3 koraka:
 - pozivanje URL-a (generisanje zahtjeva) iz JavaScript koda na klijentskoj strani
 - obrada zahtjeva na serverskoj strani i ispis odgovora
 - nakon što je odgovor završen, integrisanje u DOM (Document Object Model)

AJAX

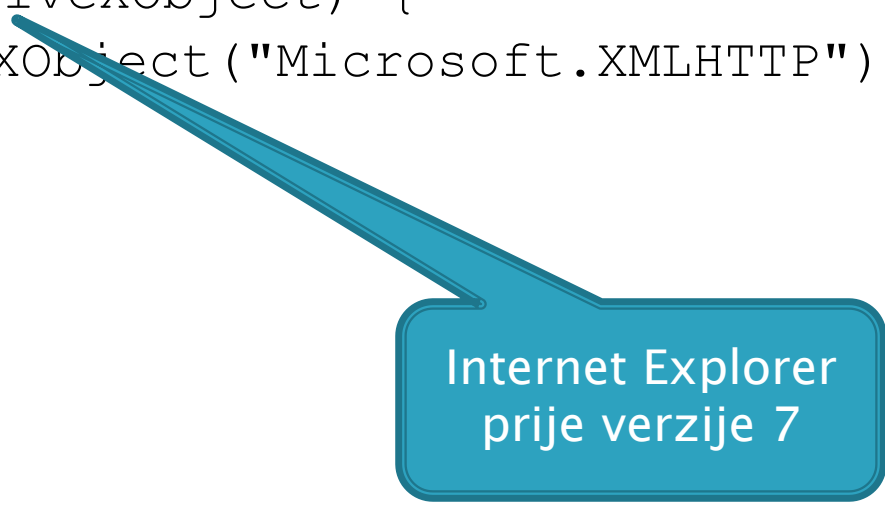
- ▶ može se steći utisak da AJAX umanjuje saobraćaj prema serveru zbog veće dostupnosti interfejsa na strani klijenta
- ▶ AJAX aplikacije, u opštem slučaju, zbog velikog broja zahtjeva prema serveru uzrokuju povećanu količinu saobraćaja

AJAX

- ▶ AJAX – koncept kojim se opisuje interakcija *XMLHttpRequest* objekta na klijentskoj strani sa serverskim skriptama
- ▶ moderni web čitači podržavaju *XMLHttpRequest* objekt (javascript objekat)
- ▶ kreiranje *XMLHttpRequest* objekta
 - `xmlhttp=new XMLHttpRequest();`

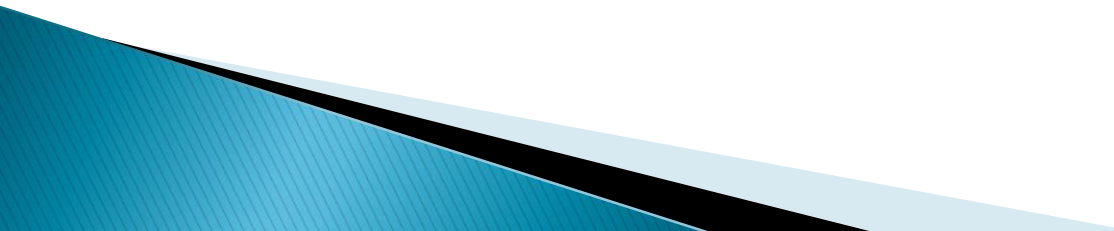
AJAX – kreiranje XMLHttpRequest objekta

```
function getRequestObject() {  
    if (window.XMLHttpRequest) {  
        return(new XMLHttpRequest());  
    } else if (window.ActiveXObject) {  
        return(new ActiveXObject("Microsoft.XMLHTTP"));  
    } else {  
        return(null);  
    }  
}
```



Internet Explorer
prije verzije 7

AJAX – slanje zahtjeva

- ▶ GET ili POST
 - ▶ GET je jednostavniji i brži od POST i može se koristiti u većini slučajeva
 - ▶ POST se može koristiti kada:
 - se šalje velika količina podataka serveru – POST nema ograničenje veličine (npr. maksimalna veličina URL u IE je 2083 karaktera)
 - kada se šalju osjetljiviji podaci – POST je robusniji i sigurniji
- 

AJAX – slanje zahtjeva

```
function sendRequest() {  
    var request = getRequestObject();  
    request.onreadystatechange = function() {  
        handleResponse(request)  
    };  
    request.open("GET", "ajax.jsp", true);  
    request.send();  
}
```

za GET zahtjev

GET zahtjev, URL
koji se poziva,
asinhroni poziv

kod koji se poziva
kada server odgovori

Svaki put kad se
promijeni stanje
readyState property-ja
automatski se poziva

readyState property
čuva stanje
XMLHttpRequest
objekta

JavaScript dozvoljava
anonimne funkcije

AJAX – slanje zahtjeva

`open(method,url,async)`

- ▶ `method`: tip zahtjeva - GET ili POST
- ▶ `url`: lokacija datoteke na serveru
- ▶ `async`: `true` (asinhroni) ili `false` (sinhroni)

`send(string)` – šalje zahtjev serveru

- ▶ `string`: koristi se samo kod POST zahtjeva, parametri sa vrijednostima

AJAX – slanje POST zahtjeva

```
xmlhttp.open("POST","ajax_test.asp",true);  
xmlhttp.setRequestHeader("Content-  
    type","application/x-www-form-urlencoded");  
xmlhttp.send("operand1=ABC&operand2=DEF");
```

setRequestHeader(header,value) – dodaje HTTP headere u zahtjev

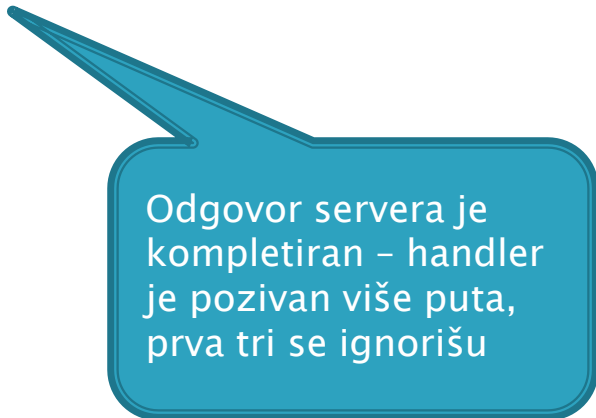
- ▶ header: specificira naziv headera
- ▶ value: specificira vrijednost headera

AJAX – obrada odgovora

```
function handleResponse(request) {  
    if (request.readyState == 4 && request.status==200) {  
        alert(request.responseText);  
    }  
}
```



odgovor servera



Odgovor servera je kompletiran – handler je pozivan više puta, prva tri se ignorišu

vrste odgovora

responseText – odgovor kao string

responseXML – odgovor kao XML

AJAX – čest pristup

- ▶ primjer u brojnoj literaturi

```
var request;  
function getRequestObject() { ... }  
function sendRequest() {  
    request = getRequestObject();  
    request.onreadystatechange = handleResponse;  
    request.open("GET", "...", true);  
    request.send(null);  
}  
function handleResponse() {  
    if (request.readyState == 4) {  
        alert(request.responseText);  
    }  
}
```

- ▶ globalna request promjenljiva
- ▶ problem:
 - dva uzastopna poziva, odgovor na prvi za 10 sekundi, odgovor na drugi za 1 sekundu – race condition

AJAX – pravilan pristup

```
function getRequestObject() { ... }
```

```
function sendRequest() {  
    var request = getRequestObject();  
    request.onreadystatechange =  
    function() { handleResponse(request); };  
    request.open("GET", "...", true);  
    request.send(null);  
}  
function handleResponse(request) {  
    ...  
}
```

- ▶ lokalna request promjenljiva

AJAX – sinhroni zahtjev

- ▶ korišćenje `async=false` se ne preporučuje, može se koristiti za male i jednostavne zahtjeve
- ▶ JavaScript neće nastaviti da se izvršava, dok odgovor servera ne bude spreman
 - ako je odziv servera spor – aplikacija se može “zaglaviti”
- ▶ kada se koristi `async=false` nije potrebno koristiti `onreadystatechange` funkciju
 - preostali kod se treba nalaziti iza `send` funkcije

AJAX – onreadystatechange događaj

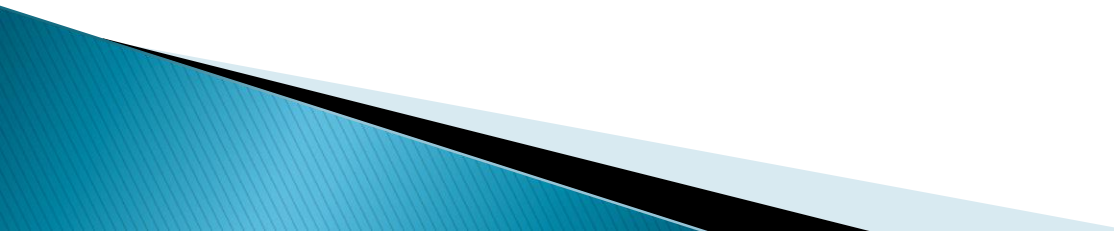
- ▶ 3 važne osobine XMLHttpRequest objekta:
 - onreadystatechange – čuva naziv funkcije koja se automatski poziva svaki put kada se readyState promijeni
 - readyState čuva status XMLHttpRequest objekta:
 - 0: zahtjev nije inicijalizovan
 - 1: uspostavljena konekcija sa serverom
 - 2: prihvaćen zahtjev
 - 3: procesiranje zahtjeva u toku
 - 4: zahtjev završen i odgovor je spreman
 - status
 - 200: OK
 - 404: Page not found

AJAX – callback funkcija

- ▶ Callback funkcija – funkcija koja se prosljeđuje kao parametar drugoj funkciji

```
function some_function2(url, callback) {  
    var httpRequest = new XMLHttpRequest();  
  
    httpRequest.onreadystatechange = function() {  
        if (httpRequest.readyState === 4 &&  
            httpRequest.status === 200) {  
            callback.call(httpRequest.responseXML);  
        }  
    };  
    httpRequest.open('GET', url);  
    httpRequest.send();  
}  
// poziv funkcije  
some_function2("text.xml", function() {  
    console.log(this);  
});  
console.log("this will run before the above callback");
```

AJAX – web čitači

- ▶ testiranje na različitim Web čitačima
 - ▶ različite verzije Internet Explorer-a
 - ▶ Firefox
 - ▶ Opera
 - ▶ Chrome
 - ▶ Safari
 - ▶ ...
- 

AJAX – JSP, Servlet

- ▶ AJAX poziv ka:
 - JSP strani – treba izbjegavati mnogo Java koda u JSP stranama
 - Servletu – pogodni za izvršavanje mnogo Java koda

```
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    Date currentTime = new Date();
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss dd.MM.yyyy.");
    String message = "Sada je " + sdf.format(currentTime) + " godine!";
    out.print(message);
}
```

```
<input type="button" value="Show Server Time from JSP"
    onclick='ajaxAlert("ajax_server_time.jsp")' />
<input type="button" value="Show Server Time from Servlet"
    onclick='ajaxAlert("ShowTime")' />
```

AJAX – ubacivanje odgovora u HTML

- ▶ Dinamičko ubacivanje teksta
- ▶ HTML
 - `<div id="time-placeholder" class="ajaxResult"></div>`
- ▶ JavaScript
 - `document.getElementById(resultRegion).innerHTML = request.responseText;`
- ▶ Rezultat
 - `<div id="time-placeholder" class="ajaxResult">Sada je 07:10:46 30.12.2011. godine!</div>`
- ▶ Upozorenje
 - ubacivanje rezultata treba da rezultira legalnim HTML dokumentom

AJAX – slanje parametara

- ▶ GET ili POST metodom
- ▶ GET metoda:
 - konkatencija parametara i njihovih vrijednosti na URL adresu
 - kao kroz URL bar Web čitača
 - `ajax.jsp?par1=12&par2=34`

AJAX – slanje parametara – GET

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    int operand1 = Integer.parseInt(request.getParameter("operand1"));
    int operand2 = Integer.parseInt(request.getParameter("operand2"));
    out.print(operand1 + operand2);
}
```

```
function ajaxCalculate(address, field1, field2, resultRegion) {
    var operandValue1 = "operand1=" + getValue(field1);
    var operandValue2 = "operand2=" + getValue(field2);
    var finalAddress = address + "?" + operandValue1 + "&" + operandValue2;
    ajaxAlert2(finalAddress, resultRegion);
}
```

```
<input type="button" value="Calculate" onclick='ajaxCalculate("Calculator",
"operand1", "operand2", "calculator-placeholder")' />
<div id="calculator-placeholder" class="ajaxResult"></div>
```

AJAX – slanje parametara – POST

- ▶ Kolekcija podataka sa forme
 - potrebno je dati id svim komponentama
 - `<input type="text" id="some-id"/>`
 - čitanje vrijednosti
 - `var value1 = document.getElementById("some-id").value;`
 - URL-encode vrijednosti i kreirati query string
 - `var data = "var1=" + escape(value1);`
- ▶ Specificirati POST metodu, umjesto GET u pozivu funkcije “open”
 - `request.open("POST", address, true);`
- ▶ Specificirati encoding type za formu
 - `request.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");`
- ▶ Proslijediti query string funkciji send
 - `request.send(data);`

Ajax stilovi

- ▶ **sadržaj–centrični Ajax**
 - server šalje HTML kod
 - klijent samo ubacuje ovaj sadržaj u stranicu
 - može koristiti CSS za prilagođenje prikaza
- ▶ **podatak–centrični Ajax**
 - server šalje “čiste” podatke klijentu
 - klijent parsira podatke i kreira HTML
- ▶ **skript–centrični Ajax**
 - server šalje JavaScript funkcije klijentu
 - klijent izvršava funkcije
 - najmanje fleksibilan – server mora znati mnogo o klijentu

Ajax – rad sa XML podacima

- ▶ tretiranje Ajax odgovora kao XML
 - `var xmlDoc = response.responseXML;`
- ▶ uzimanje niza podelemenata (na bazi naziva taga)
 - `xmlDoc.getElementsByTagName(...)`
- ▶ uzimanje atributa elementa
 - `someElement.getAttribute(...)`
- ▶ sadržaj elementa
 - `someElement.firstChild.nodeValue`

Ajax – rad sa JSON podacima

▶ JSON

- JavaScript Object Notation – jednostavna tekstualna reprezentacija JavaScript objekata, podržana direktno u JavaScript-u

▶ Rad u JavaScript-u

```
var someObject =  
{ property1: value1,  
  property2: value2, ... };
```

▶ U formi stringa – npr., kod prenosa preko mreže

- objektna reprezentacija smješta se unutar zagrada

Ajax – rad sa JSON podacima

- ▶ objekt
 - objekat se okružuje vitičastim zagradama
 - nazivi polja smještaju se unutar jednostrukih ili dvostrukih znakova navoda
 - znakovi navoda mogu biti izostavljeni ako je naziv polja legalan
 - dvotačke se koriste između naziva polja i njihovih vrijednosti
 - zarez se stavlja iza svakog para naziv polja – vrijednost polja
- ▶ vrijednosti polja
 - stringovi: koristiti jednostruke ili dvostruke znakove navoda
 - numeričke: nema potrebe za navodnicima
 - nizovi: CSV (comma separated values) unutar uglastih zagrada
- ▶ smještanje JSON-a u string
 - unutar zagrada i navodnika

Ajax – rad sa JSON podacima

- ▶ Konverzija JSON u JS objekat

- ▶ `var myJSON = '{ "ime":"Pera", "godine":31, "grad":"BL" }';`
- ▶ `var myObj = JSON.parse(myJSON);`
- ▶ `document.getElementById("test").innerHTML = myObj.name;`

Ajax – rad sa JSON podacima

- ▶ JSON i XML
- ▶ sličnosti
 - "self describing" (human readable)
 - hijerarhijski organizovani (vrijednosti unutar vrijednosti)
 - mogu biti prasilani i konzumirani od strane različitih programskih jezika
 - mogu biti „dohvaćeni“ pomoću XMLHttpRequest objekta
- ▶ razlike:
 - JSON nema „zatvarajući tag“
 - JSON je manji
 - JSON se brže čita i zapisuje
 - JSON može raditi s nizovima
 - XML se mora parsirati XML parserom, dok se JSON može parsirati JS funkcijom

Ajax – rad sa JSON podacima

▶ tipovi JSON podataka

- string
 - { "ime":"Marko" }
- broj
 - { "godine":30 }
- JSON objekat
 - {
 "osoba" : { "ime":"Pera", "godine":31, "grad":"BL" }
}
- niz
 - {
 "osobe":["Marko", "Ana", "Petar"]
}
- boolean
 - { "test":true }
- null
 - { "test":null }

Client-Side alati – JavaScript biblioteke sa podrškom za Ajax

- ▶ jQuery
 - <http://jquery.com/>
- ▶ Prototype
 - <http://www.prototypejs.org/>
- ▶ script.aculo.us
 - <http://script.aculo.us/>
- ▶ Dojo
 - <http://www.dojotoolkit.org/>
- ▶ Ext-JS
 - <http://extjs.com/>
- ▶ Yahoo User Interface Library (YUI)
 - <http://developer.yahoo.com/yui/>
- ▶ Google Closure Library
 - <http://code.google.com/closure/library>

Server-Side alati

- ▶ JSON/JSON-RPC
 - <http://www.json.org/>
 - <http://json-rpc.org/>

Hibridni (klijent-server) alati

- ▶ JSP custom tag biblioteke
 - kreirati tagove koji generišu HTML i JavaScript
- ▶ AjaxTags (nadgradnja script.aculo.us)
 - JSP custom tagovi koji generišu Ajax funkcionalnost
 - jednostavna sintaksa
 - <http://ajaxtags.sourceforge.net>
- ▶ Google Web Toolkit
 - pisanje koda u Javi, prevođenje u JavaScript
 - <http://code.google.com/webtoolkit/>
 - <https://ajax4jsf.dev.java.net/>
 - GWT/JSF Integration Toolkit