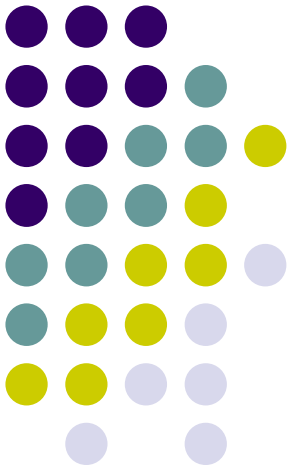
Klijentske tehnologije

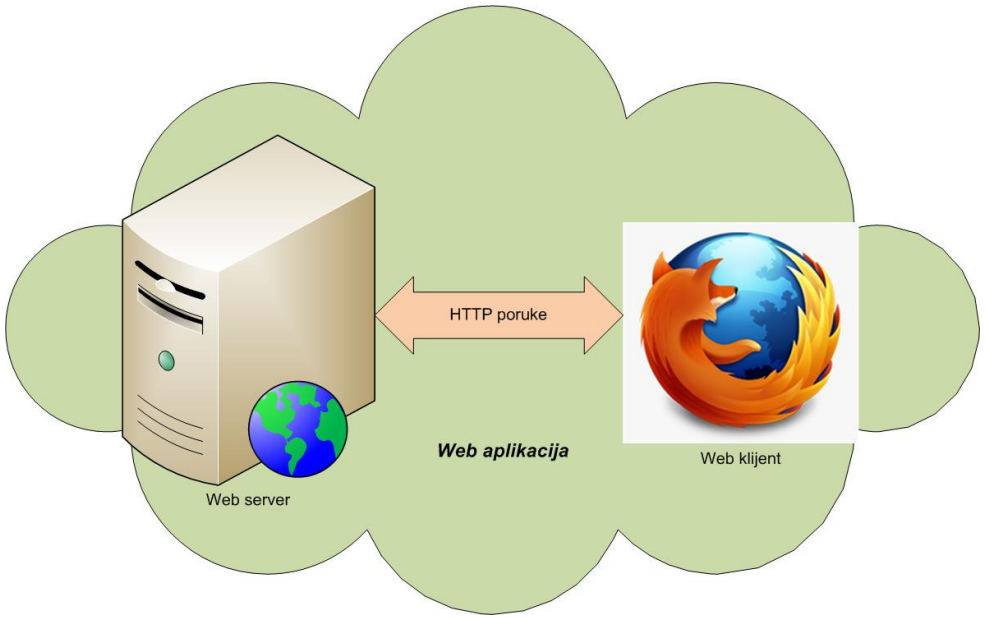
Maja Štula

ak. god. 2011/2012

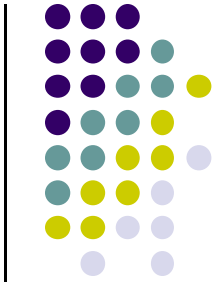


Web aplikacije

- Razvoj tehnologija koje se koriste u web aplikacijama dosta je usmjeren u razvoj tehnologija koje omogućavaju prebacivanje dijela poslovne logike web aplikaciju na stranu klijenta.
- Arhitektura web aplikacija je tipa klijent/server.

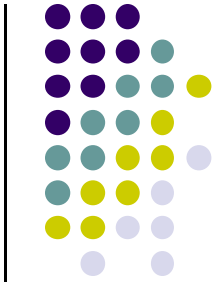


Web aplikacije



- Već smo spominjali JavaScript programski jezik koji se izvršava u klijentu i DOM model koji omogućava dinamički pristup dokumentima u web klijentu.
- Pored ovih tehnologija postoji još cijeli niz tehnologija na strani klijenta kao i implementacija tehnologija u različitim modelima.

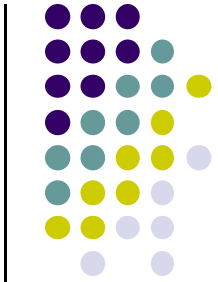
Web aplikacije



- Razvoj tehnologija na strani web klijenta ima dva glavna razloga:

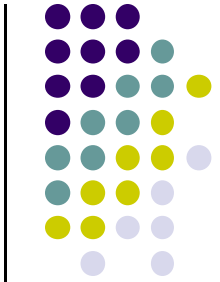
1. Prebacivanje dijela poslovne logike na stranu klijenta (dio posla obrade podataka obavlja klijent) - prednost prebacivanja dijela procesiranja na klijenta je rasterećenje Web servera i smanjenje prometa na mreži.
2. Približavanje web aplikacija klasičnim desktop aplikacijama prema izgledu i funkcionalnosti grafičkog sučelja prema korisniku.

Web aplikacije



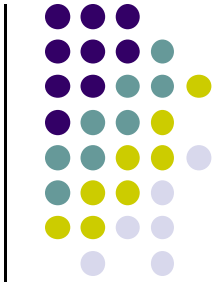
- Tradicionalne web aplikacije su radile na način da je sva obrada podataka (ili neka druga funkcionalnost web aplikacije) bila na strani servera, a klijent je prikazivao samo statički sadržaj u obliku HTML stranice.
- Najveći nedostatak ovakvih aplikacija je da sva interakcija aplikacije i korisnika ide preko servera.
- Svaki korisnikov zahtjev treba se proslijediti serveru. Klijentu se odgovor servera vraća opet kao statički sadržaj.

Web aplikacije



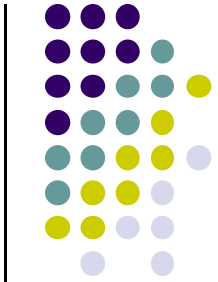
- Korištenjem tehnologija koje su na raspolaganju na strani klijenta dio procesiranja može se prebaciti na klijenta.
- Na taj način se dobivaju tzv. bogate Internet aplikacije (*Rich Internet Applications* - RIA).
- Pojam RIA uvela je Macromedia 2002.

Tipovi klijenata



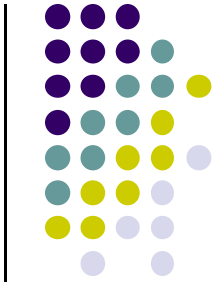
- Arhitektura web aplikacija je tipa klijent/server.
- Kod arhitekture softverskog sustava tipa server/klijent koriste se dva tipa klijenata:
  - "Tanki klijent" (*thin client*) – je minimalni klijent; minimalno korištenje resursa hosta; obrada podataka (poslovna logika) je na strani servera.
  - "Debeli klijent" (*fat, rich, thick client*) – je klijent koji obavlja i dio posla obrade podataka; dio poslovne logike je prebačen na stranu klijenta.

Tipovi klijenata Web aplikacija



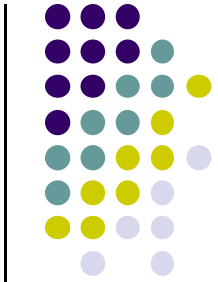
- Kako se dio aplikacije izvršava i na klijentu korisničko sučelje može imati opcije koje nisu na raspolaganju kod tankog klijenta.
- Složenija funkcionalnost može uključivati bilo što što se može implementirati na strani klijenta npr. nekakav izračun (http://www.calculator.com/calcs/calc_sci.html), obradu podataka s forme prije slanja i sl.
- Osim toga Web aplikacije se žele što više približiti desktop aplikacijama prema izgledu i funkcionalnosti grafičkog sučelja prema korisniku.
- Primjer: http://maps.google.com/

Tipovi klijenata Web aplikacija



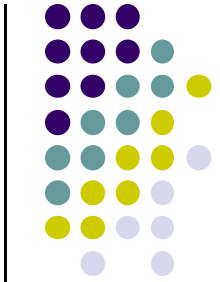
- Nedostatci RIA aplikacija:
  - Korisnik ima mogućnost zabraniti izvođenje JavaScript koda ili instalaciju ActiveX kontrole tj. ima mogućnost potpunog onemogućavanja funkcionalnosti aplikacije.
  - Kako se RIA aplikacije izvode u tzv. “pješčaniku” (*sandbox*) ograničen mi je pristup resursima računala. Sandbox je sigurnosni mehanizam koji se koristi za izvođenje nepouzdanog koda. Postavke ograničenja mogu se mijenjati. Ukoliko aplikacija ne može pristupiti nekom resursu neće ispravno funkcionirati.
  - RIA aplikacije u pravilu ne zahtijevaju instalaciju (*ActiveX*), ali se trebaju dohvatiti sa servera barem jednom (ukoliko ih klijent kešira ne treba ih više dohvaćati) (C:\Documents and Settings\Maja\Local Settings\Temporary Internet Files).

Tehnologije na strani klijenta



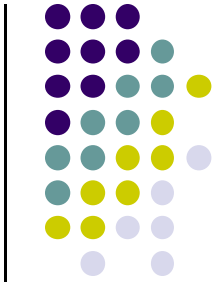
- Prenošenjem funkcionalnosti web aplikacija na stranu klijenta tj. postavljanjem standarda bavi se najviše W3 konzorcijum.
- Tako se W3C Web Applications (WebApps) Working Group bavi standardizacijom klijentskih API-ja. (http://www.w3.org/2008/webapps/)

Tehnologije na strani klijenta



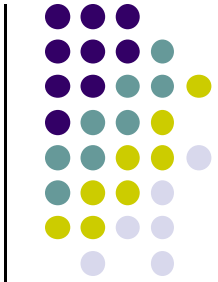
- JavaScript je skriptni programski jezik koji se izvršava na strani klijenta. Većina web klijenata ima podršku za JS. U kombinaciji sa CSS i DOM pruža dosta funkcionalnosti.
- XMLHttpRequest objekt je osnova Ajax tehnologije. Prvi ga je implementirao Microsoft na IE5 kao ActiveX kontrolu koja je uspostavljala HTTP konekciju u pozadini otvorene stranice i dohvaćala XML podatke.
- Implementiran je kompatibilno Microsoftovom od Mozilla 1.0 (Netscape 7) verzije kao dio pretraživača. To je napravio i Apple od Safari 1.2.

Tehnologije na strani klijenta



- JavaScript i XMLHttpRequest su podržani u svim web klijentima i za korištenje ovih tehnologija na strani klijenta nije potrebno instalirati dodatne aplikacije (plugin-ove, addon-ove i sl.).
- Pored ovih tehnologija postoji još cijeli niz tehnologija na strani klijenta koje zahtjevaju dodatne aplikacije kako bi se mogli izvršavati (flash player, applet viewer, ...).

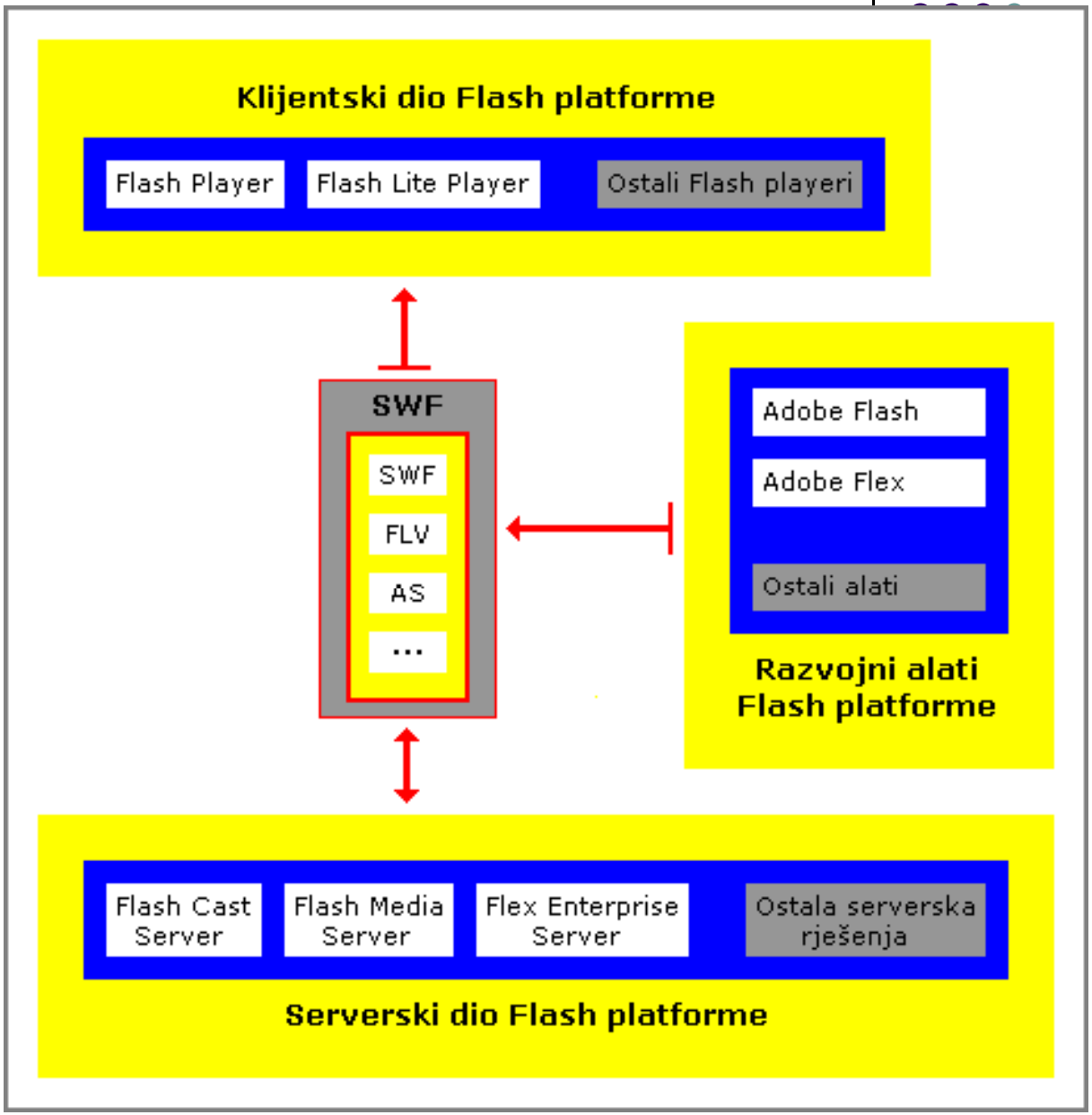
Flash



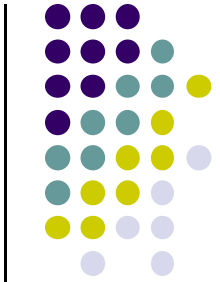
- Da bi se izvršio Flash (Macromedia, Adobe) sadržaj, web klijent treba imati neku verziju Flash playera.
- Flash Player je podržan na IE, FireFox-u i Operi i distribuiran je kao besplatni plugin.
- Flash Player je u biti virtualni stroj koji izvršava bajt kôd zapisan u SWF datotekama.
- Primjeri:
- http://flash-creations.com/notes/intro_flashexamples.php
- http://naldzgraphics.net/inspirations/45-excellent-examples-of-flash-websites-design/

Flash

- Flash tehnologija pokriva i stranu klijenta i stranu servera.
- Omogućava razvoj složenih multimedijalnih aplikacija, ali zahtjeva podršku za tu tehnologiju.

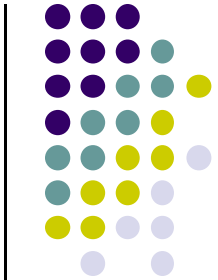


Silverlight



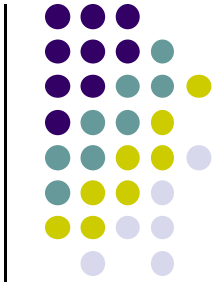
- Microsoft Silverlight je Microsoftova tehnologija za prikaz i animaciju površina grafičkih elemenata u web preglednicima (na neki način odgovor na Flash).
- Oslanja se na XAML (eXtensible Application Markup Language) markup jezik za kreiranje složenih, dinamičkih korisničkih sučelja, odnosno na .NET 3.x.
- Koristi se za razvoj grafičkog sučelja web aplikacija, a za izvršavanje Silverlight aplikacije potrebno je također imati instaliran player u web pretraživaču. Primjer

Java aplet



- Java aplet je Java aplikacija koja je napisana za izvođenje u pretraživaču.
- Java aplet se uključuje u HTML stranicu posebnim tagom.
- Za izvođenje apleta pretraživač treba imati instaliran aplet viewer (za većinu pretraživača postoji *plugin*).
- http://java.sun.com/applets/
- http://docs.oracle.com/javase/tutorial/deployment/applet/

Java applet - primjer



- Uključivanje appleta u html stranicu vrši se preko posebnog taga <applet>:

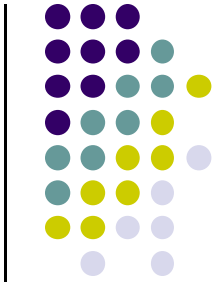
```
<applet code=Applet1.class width="200" height="200"> Your
    browser does not support the applet tag. </applet>
```

- Ili preko tagova <object> i <embed>:

```
<object classid="clsid:8AD9C840-044E-11D1-B3E9-
    00805F499D93" width="200" height="200"> <PARAM
    name="code" value="Applet1.class"> </object>

<embed code="Applet1.class" width="200" height="200"
    type="application/x-java-applet;version=1.6.0"
    pluginspage="http://java.sun.com/javase/downloads"/>
```

Java applet - primjer



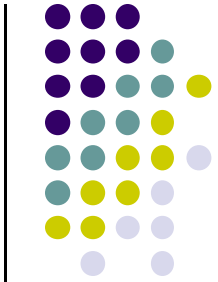
- Pretraživač će applet pokušati dohvatiti u istom direktoriju u kojem se nalazi i stranica sa uključenim appletom. Ukoliko se applet nalazi negdje drugdje atribut CODEBASE sadrži relativni ili apsolutni URL do appleta.

  < applet code=Applet1.class codebase="http://www.fesb.hr/~kiki/" width="200" height="200"> </applet>

- Prenošenje parametara u applet vrši se pomoću taga PARAM umetnutog u applet tag. Na ovaj se način omogućava da korisnici appleta mogu prilagođavati applet pomoću parametara bez mijenjanja kôda.

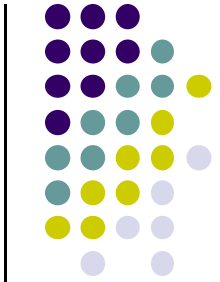
  <APPLET CODE=*AppletSubclass*.class WIDTH=*anInt* HEIGHT=*anInt*> <PARAM NAME=*parameter1Name* VALUE=*aValue*> <PARAM NAME=*parameter2Name* VALUE=*anotherValue*> </APPLET>

Java applet - primjer



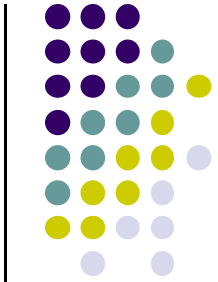
- Applet se uvijek definira kao podklasa klase java.applet.Applet ili klase javax.swing.JApplet.
- Elementi grafičkog sučelja, poput botuna i sl., definirani su u imenskom prostoru javax.swing.
- Potrebno je pregaziti metodu init() koja inicijalizira applet.
- Metode koje se još često koriste su stop (poziva se prilikom zaustavljanja appleta), start, destroy (poziva se kada se applet uništava) i paint (poziva se kada se radi iscrtavanje appleta).

Java applet - primjer



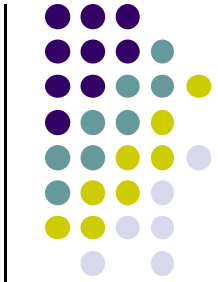
- Applet klasa obavezno treba biti definirana kao public!!!
- Imenski prostor java.awt.Graphics potrebno je uključiti zbog iscrtavanja tj. metode paint za iscrtavanje.
- Applet može otvarati konekciju u pozadini, ali je zbog sigurnosti ograničeno da se konekcija može otvarati samo prema onom serveru sa kojeg je applet i dohvaćen.

Java applet - primjer



```
import java.applet.Applet;
import java.awt.Graphics;
public class mojapplet extends  Applet
{
    public void init()
    {   }
    public void paint(Graphics g) {
            g.drawRect(0, 0, getSize().width - 1, getSize().height - 1);
            String tekst = getParameter("tekst");
            if (tekst != null)
            {       g.drawString(tekst, 5, 25);    }
            else
            {   g.drawString("Ovo nije preneseni tekst", 5, 25); }
   }
}
```

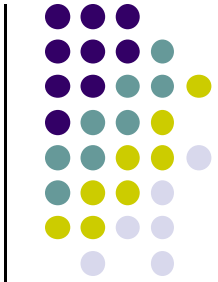
Java applet - primjer



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">
<HTML>
<HEAD>
</HEAD>
<BODY>
<applet code="mojapplet.class" width="300" height="100">
<param name="tekst" value="Tekst prenesen preko parametra">
</applet>
</BODY>
</HTML>
```

E:\predavanja\internet_2_sve\applet\mojapplet.html

Tehnologije na strani klijenta

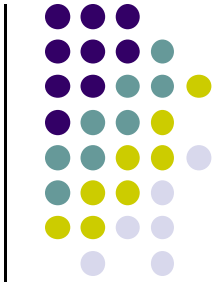


- ActiveX aplikacija je temeljena na Microsoft COM (Component Object Model) modelu. Koristi se pretežno za razvoj dodatnih komponenti IE (AcitveX kontrola).
- SVG (Scalable Vector Graphics) je jezik temeljen na XML-u za predstavljanje grafičkih elemenata kroz dvodimenzionalnu vektorsku grafiku. Može se integrirati s XML podacima. Ima podršku i za CSS i JavaScript. Određeni pretraživači imaju integriranu podršku za SVG.

  http://www.w3.org/Graphics/SVG/

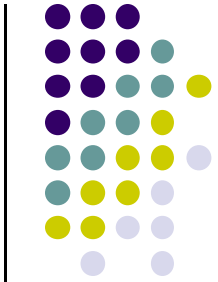
  http://www.w3schools.com/svg/svg_examples.asp

Tehnologije na strani klijenta



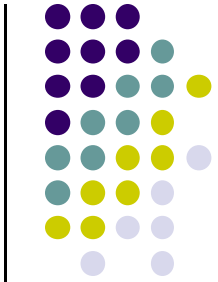
- W3C Web Applications (WebApps) Working Group bavi se dokumentiranjem postojećih API-ja poput XMLHttpRequest, ali i razvojem novih.
- W3C je definirao standard za XMLHttpRequest kao objekt koji sadrži predefinirane metode i svojstva.
- Kako se konkretno XMLHttpRequest implementira u pretraživaču nije specificirano.
- XMLHttpRequest objekt je osnova Ajax tehnologije.

Ajax



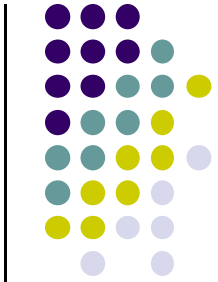
- Ajax je oznaka za asinkroni JavaScript i XML. To je Web tehnika za kreiranje interaktivnih web aplikacija. (izvor http://en.wikipedia.org)
- Asinkronost se odnosi na to da se odgovor servera na zahtjev klijenta može obraditi bez čekanja i zamrzavanja aktivnosti korisnika.
- Pojam Ajax je prvi put upotrijebio Jesse James Garrett u veljači 2005 u članku Ajax: A New Approach to Web Applications
- AJAX nije skraćenica. To čak nije ni novi koncept ili pojedinačna tehnologija već je to ime za skupinu tehnologija koje se već duže vremena koriste.

Ajax

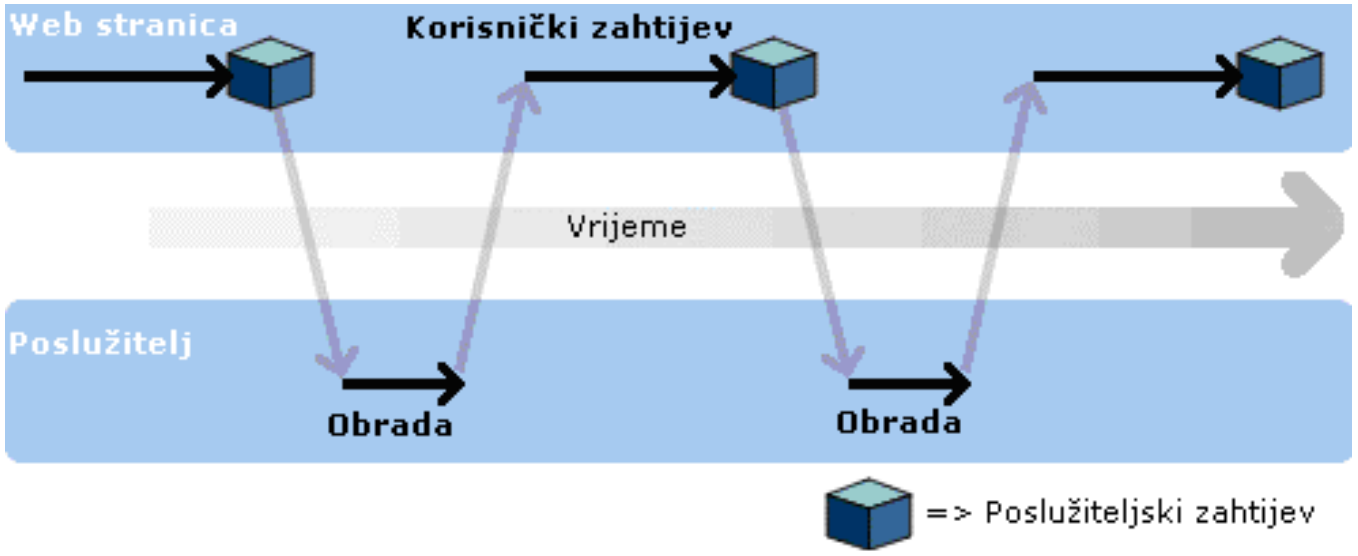


- Tehnologije uključene u Ajax:
  - **XMLHttpRequest** objekt za asinkronu razmjenu podataka s Web serverom. Zbog sigurnosti konekcija se može otvarati samo prema serveru sa kojeg je dohvaćena stranica sa XMLHttpRequest objektom.
  - HTML, XHTML i CSS za prikaz dohvaćenih podataka
  - JavaScript i DOM za pristup XMLHttpRequest objektu i elementima stranice
  - XML za zapis podataka
- Ajax je kombinacija XHTML, CSS, DOM, XML, asinkronog dohvaćanja podataka preko XMLHTTPRequest, i JavaScripta koji to sve povezuje.

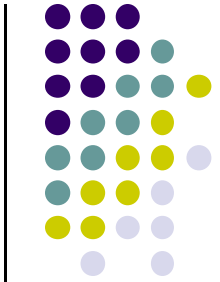
Tradicionalna komunikacija sa serverom



- Kod tradicionalnih aplikacija interakcija sa serverom je zahtijevala ponovno učitavanje stranice (*page reload*). Npr. kada bi korisnik ispunio formu i poslao zahtjev prema serveru sačekao bi stranicu odgovora od servera i tek onda bi mogao nastaviti s radom.

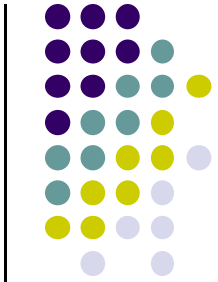


Tradicionalna komunikacija sa serverom

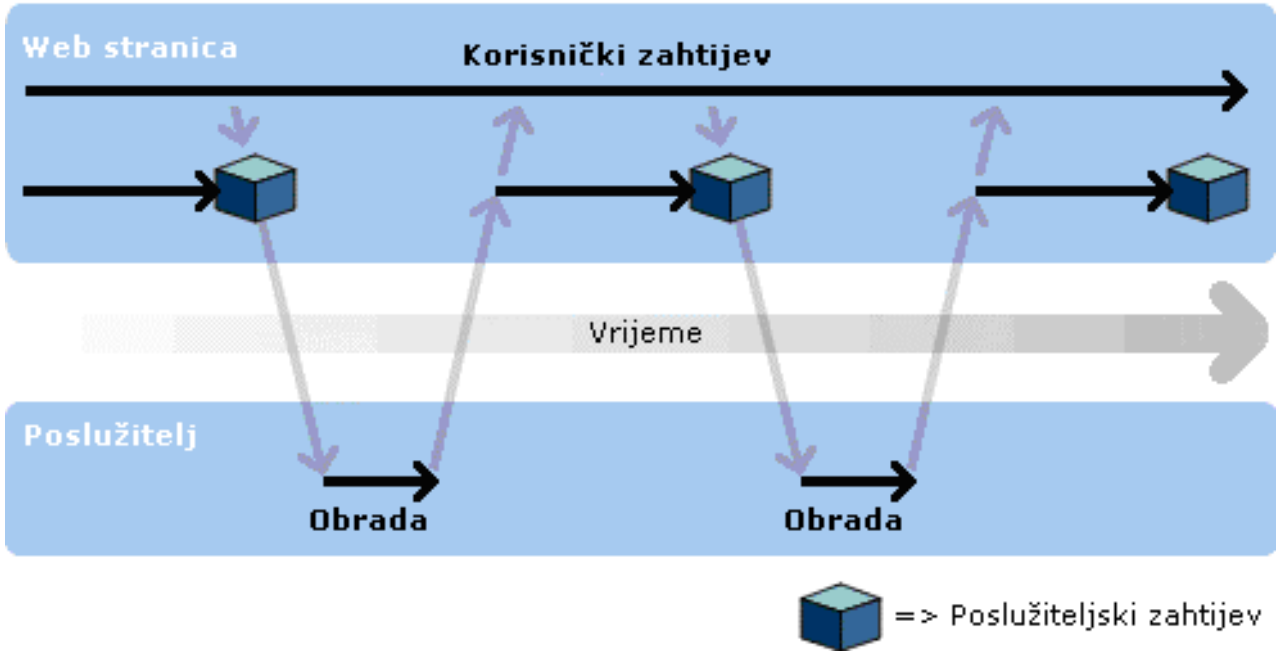


- Pozitivne strane:
  - Web aplikacije su bile dosta kompatibilne s pretraživačima.
  - Implementacija funkcionalnosti išla je preko običnih HTML kontrola.
- Negativne strane:
  - Rad korisnika je prekidan čekanjem na odgovor servera
  - Korištenje pojasne širine komunikacijskog kanala na ovaj način nije efikasno.

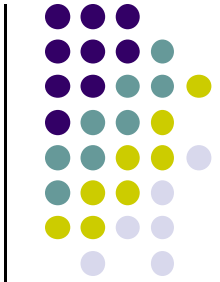
Ajax komunikacija sa serverom



- Pomoću Ajax modela serveru se može poslati zahtjev samo za dohvat potrebnih podataka, a ne cijele stranice.
- Taj zahtjev se šalje u pozadini stranice na kojoj korisnik trenutno radi.

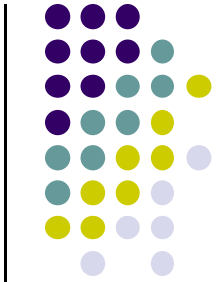


Ajax komunikacija sa serverom



- Pozitivne strane:
  - Korisnik se ne treba prekidati u radu.
  - Pojasna širina se štedi slanjem samo potrebnih podataka, a ne cijele stranice.
  - Mogućnost novih tipova korisničkih sučelja.
  - Ne zahtjeva dodatni softver kao Java ili Flash.

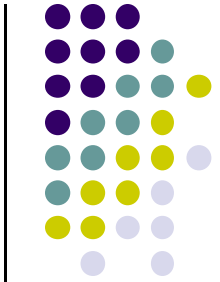
http://www.w3.org/TR/XMLHttpRequest



XMLHttpRequest sučelje

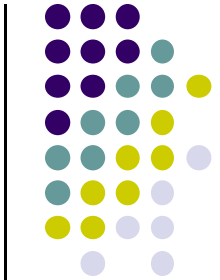
```
interface XMLHttpRequest
{ attribute EventListener onreadystatechange;
readonly attribute unsigned short readyState;
void open(in DOMString method, in DOMString uri);
void open(in DOMString method, in DOMString uri, in boolean async);
void open(in DOMString method, in DOMString uri, in boolean async, in DOMString user);
void open(in DOMString method, in DOMString uri, in boolean async, in DOMString user, in
     DOMString password);
void setRequestHeader(in DOMString header, in DOMString value) raises(DOMException);
void send();
void send(in DOMString data) raises(DOMException);
void send(in Document data) raises(DOMException);
void abort();
DOMString getAllResponseHeaders();
DOMString getResponseHeader(in DOMString header);
attribute DOMString responseText;
attribute Document responseXML;
attribute unsigned short status; // raises(DOMException) on retrieval
attribute DOMString statusText; // raises(DOMException) on retrieval };
```

Ajax



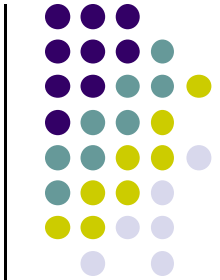
- Ajax programski model se bazira na događajima (*events*) i prikazima rezultata događaja.
- Događaji su neke korisničke akcije (npr. klik na botun na formi) koje rezultiraju pozivom funkcije koja koristi XMLHttpRequest objekt.
- DOM omogućava da se dohvaćeni podaci prikažu korisniku preko nekog elementa stranice.

XMLHttpRequest



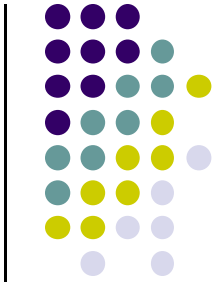
- Za dohvaćanje podataka sa servera XMLHttpRequest objekt koristi dvije metode:
  - open – kreira konekciju prema serveru
  - send – šalje zahtjev prema serveru.
- Podaci sa servera (nakon odgovora servera) smješteni su u jednom od sljedeća dva atributa XMLHttpRequest objekta:
  - responseXml – za dohvat XML datoteke
  - responseText – za dohvat obične tekstualne datoteke (uključujući i HTML stranicu).

XMLHttpRequest



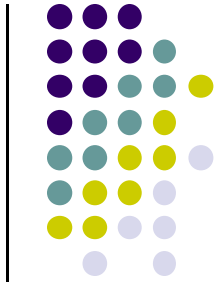
- Prije no što se krene u obradu podataka dohvaćenih sa servera trebamo biti sigurni da su podaci dohvaćeni do kraja, da je server server završio s procesiranjem zahtjeva poslanog preko XMLHttpRequest *send* metode ,….
- Stanje podataka definirano je u atributu *readyState* XMLHttpRequest objekta.
- Moguća stanja podataka i vrijednosti *readyState* atributa su:
  - 0 (*uninitialized*) – funkcija *open* još nije pozvana.
  - 1 (*loading*) - konekcija uspostavljena, ali funkcija *send* još nije pozvana.
  - 2 (*loaded*) – *send* je napravljen, u odgovoru primljena zaglavlja i statusni kôd odgovora.
  - 3 (*interactive*) – dohvaćanje podataka odgovora u tijeku, podaci su trenutno još uvijek djelomični.
  - 4 (*complete*) – završene sve operacije i dohvaćeni podaci.

XMLHttpRequest



- Prije no što se krene s obradom podataka treba sačekati da *readyState* atribut poprimi vrijednost 4 tj. da je prijem podataka sa servera na osnovu postavljenog zahtjeva *open* metodom završen.
- No kada je prijem podataka završen nismo sigurni da li je server vratio odgovor koji smo očekivali. Npr. ako pozivamo neku CGI skriptu ili nešto slično server može vratiti odgovor o greški (500 HTTP status kod).
- Provjeru odgovora možemo raditi ili parsiranjem primljenih podataka (npr. tražimo određeni string) ili provjerom atributa *status* koji sadrži HTTP statusni kod odgovora (npr. 200) ili *statusText* koji sadrži HTTP statusni kod u tekstualnom obliku (npr. OK).

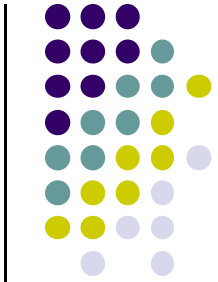
Ajax



- Prvi korak u programu je kreiranje instance objekta XMLHttpRequest:

```
if (window.XMLHttpRequest)     // Object of the current windows
{
     request = new XMLHttpRequest();     // Firefox, Safari, ...
      alert("Ovo je Firefox ili nesto sl.");
}
else
if (window.ActiveXObject)   // ActiveX version
{
   request = new ActiveXObject("Microsoft.XMLHTTP");    // Internet Explorer
   alert("Ovo je IE ili nesto sl.");
}
```

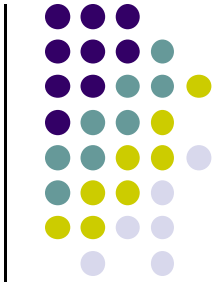
Ajax



- Obrada podataka koje šalje server smješta se u zasebnu funkciju koja se povezuje s atributom *onreadystatechange* koji sadrži "pokazivač" na funkciju (*EventListner*) koja će se izvršiti kada dođe do promjene stanja tj. *readyState* atributa:

  request.onreadystatechange = mojafunkcija();

Ajax

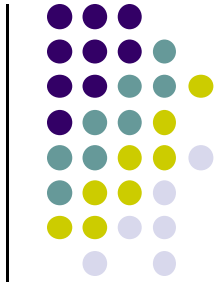


- Slanje zahtjeva prema serveru radi se u dva koraka:
  - open naredba definira GET ili POST zahtjev i URL prema kojem se taj zahtjev postavlja, zadnji argument metode je true za asinkronu konekciju.
  - send naredba šalje podatke s POST zahtjevom, dok se za GET zahtjev šalje prazni send.
- Primjer:

```
request.open("GET","kreiranje_objekta.html", true);
request.send(null);

request.open('POST', 'rating-submit.php', true);
request.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
request.send('rating=' + target.elements['rating'].value);
```

Ajax

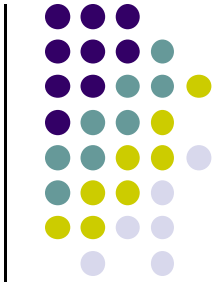


Starije verzije IE-a

```
//Kreiranje XMLHttpRequest objekta
function getHTTPObject_alarm()
{  var xmlHttp = false;
   /*@cc_on @*/
   /*@if (@_jscript_version >= 5)
   try {
     xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
   } catch (e) {
     try {
       xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
     } catch (e2) {
       xmlHttp = false;
     }
   }
   @end @*/
if (!xmlHttp && typeof XMLHttpRequest != 'undefined') {  xmlHttp = new XMLHttpRequest(); }
  return xmlHttp;
}
```

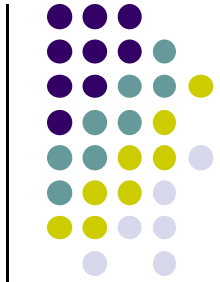
Primjer

http://vatra.fesb.hr/ajax_primjer/stranica.htm



```
<HTML><HEAD>
<script type="text/javascript">
var httpsl;
function getHTTPObject()
{
var xmlHttp = false;
if (!xmlHttp && typeof XMLHttpRequest != 'undefined') {  xmlHttp = new XMLHttpRequest(); }  return xmlHttp;
}
function funkcija_obradi()
{
if(httpsl.readyState == 4)       {
         var p_elementi = document.getElementsByTagName("P");
        p_elementi[0].innerHTML = httpsl.responseText;
         setTimeout(" obrada_podatak()",1000); }
        else       setTimeout("funkcija_obradi()",500);
}
function obrada_podatak()
{       httpsl = getHTTPObject(); // HTTP Object
        adresa="citaj_datoteku.php“;
        httpsl.open("GET",adresa, true);
        httpsl.onreadystatechange = funkcija_obradi();   httpsl.send(null);
}
</script> </HEAD><BODY onLoad="obrada_podatak()“><H1>Sadržaj datoteke </H1><p></p></BODY></HTML>
```

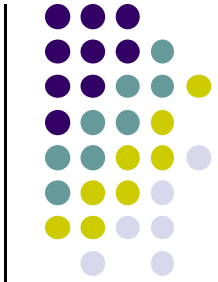
Primjer



```
<?php
$instream = fopen("datoteka.txt", 'r');
if($instream)
{
while(!feof($instream))
{
    $tmp = fgets($instream);
     echo $tmp;
}
}
else
echo "Ne mogu otvoriti datoteku";
?>
```

http://vatra.fesb.hr/ajax_primjer/citaj_datoteku.php

Tehnologije na strani klijenta



- Pored API standarda koji se razvijaju, razvija se i cijeli niz tzv. “proprietary” API-ja koji se temelje na nekoj od postojećih tehnologija (AJAX, Web servisi, …). Takve API-je razvijaju veliki site-ovi poput Googla, Amazona, Yahooa i drugih.
- Ti API-ji su skup metoda ili funkcija pomoću kojih je moguće pristupati unutarnjoj funkcionalnosti neke web stranice i pružaju korisnicima sučelje za pristup uslugama web stranice bez da moraju poznavati unutarnju strukturu same stranice.

  http://www.programmableweb.com/apilist