Programiranje za Internet

Web tehnologije na strani klijenta

ak. god. 2023./24.

Web

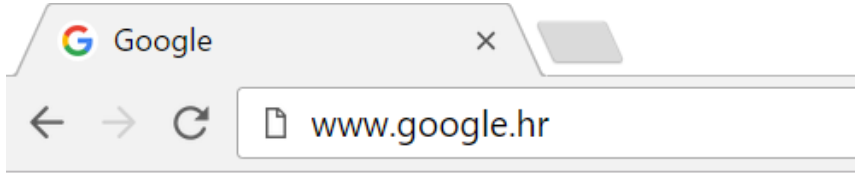
- Arhitektura weba je tipa klijent/server.
- Sadržaj dohvaćen s web servera isporučuje se korisniku u tijelu HTTP(S) odgovara.
- Može se raditi o statičkom sadržaj (HTML, CSS, …) smještenom na web serveru koji web server dohvaća i vraća korisniku na njegov HTTP(S) zahtjev.
- Može se raditi o dinamički kreiranom sadržaju korištenjem različitih web serverskih tehnologija (*backend* web aplikacije) često povezanih s bazama podataka.
- Dohvaćeni sadržaj (HTML, CSS, …) je potrebno na strani klijenta prikazati (iscrtati, renderirati) korisniku. Taj sadržaj može biti dinamički i na strani klijenta te izvršavati različite funkcije korištenjem različitih tehnologija (JavaScript, …) u klijentskoj web aplikaciji. (*frontend* web aplikacije)

Web preglednik

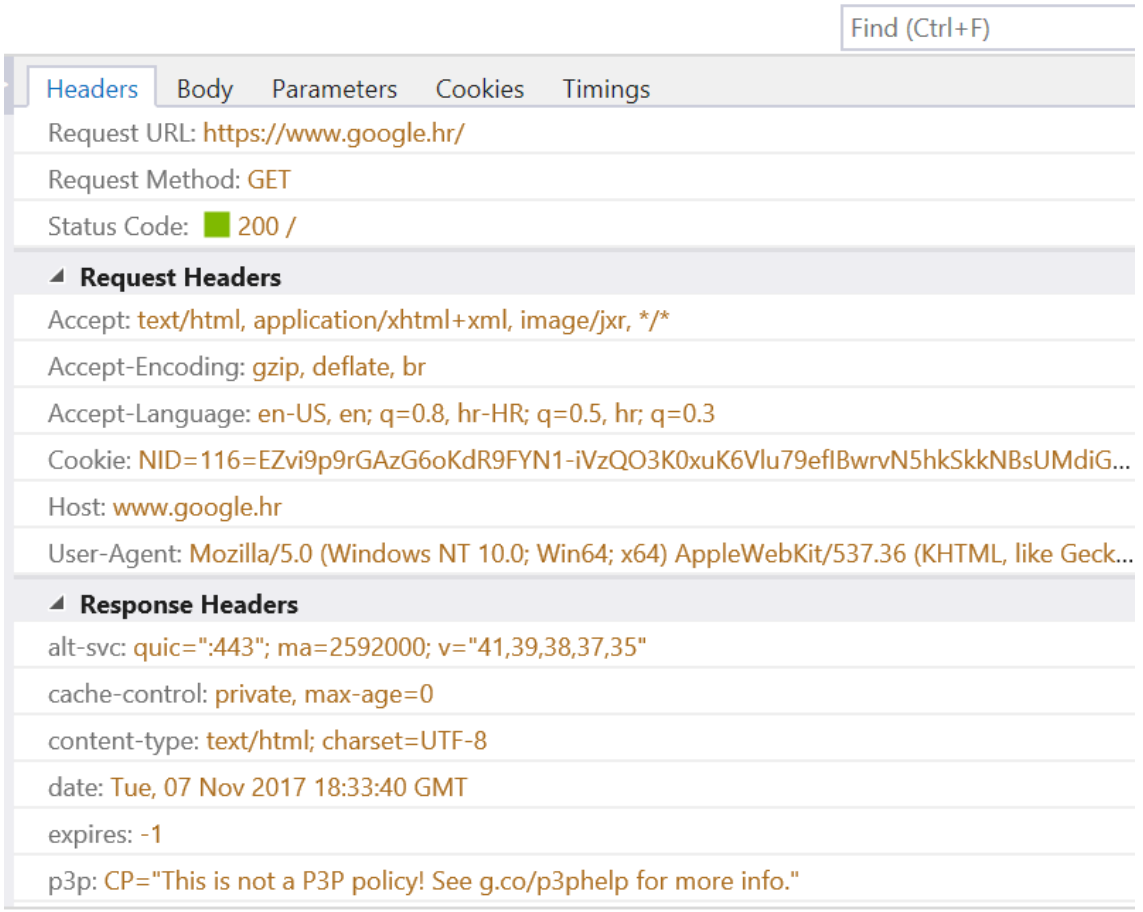
- Na strani web klijenta se koriste različite aplikacije web preglednika/*browsera* poput Chrome, Safari, Edge, FireFox, ... koje na osnovu korisnički zadane adrese u obliku URI-ja putem HTTP(S) protokola komuniciraju s web serverom i dohvaćaju sadržaj koji prikazuju korisniku.

Web preglednik

1. Korisnik definira traženi URI:

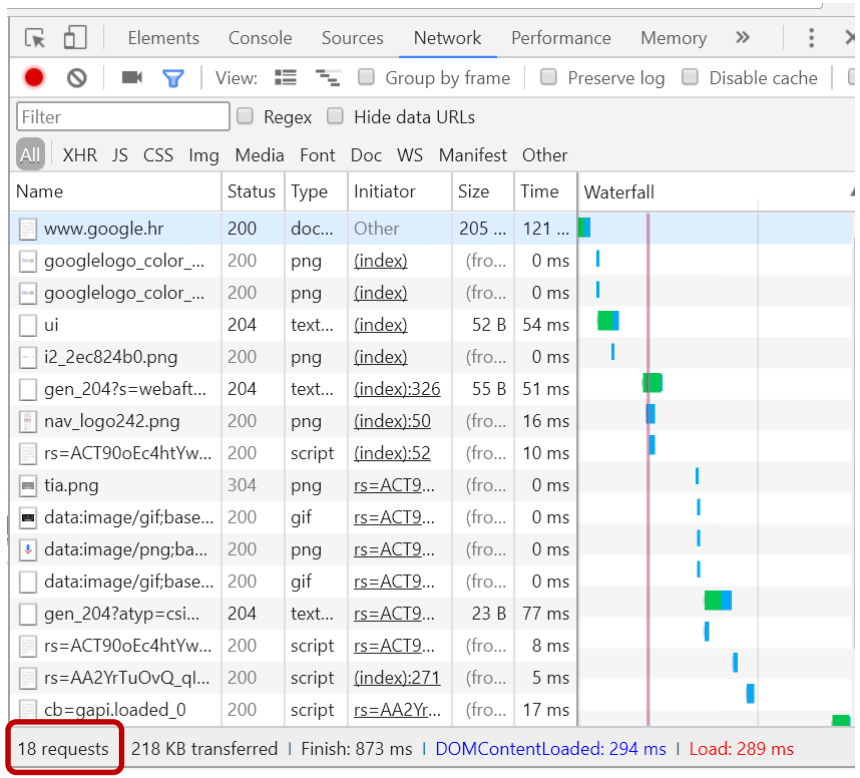


2. Preglednik na osnovu te informacije, informacija sadržanih u postavkama preglednika te u *cacheu* stvara HTTP(S) zahtjev:



Web preglednik

3. Ukoliko je potrebno preglednik stvara i dodatne HTTP(S) zahtjeve da bi dohvatio sadržaj:



- Preglednik "zna" kako i kada treba stvoriti dodatne HTTP(S) zahtjeve. Npr. za svaki *img* HTML element, preglednik će sam stvoriti novi HTTP(S) zahtjev prema URI-ju navedenom u *src* atributu HTML elementa kako bi dohvatio sliku i prikazao je korisniku u pregledniku.

Web preglednik

- *Cache* (skrivena memorija, predmemorija) preglednika je dio diska na koji preglednik pohranjuje datoteke dohvaćenog sadržaja s weba.
- To uključuje HTML, CSS, JS datoteke, slike (gif, jpg,….), *cookie* datoteke, itd.
- Preglednik upravlja tim sadržajem, osvježava ga po potrebi ili korisniku pruža sadržaj iz *cachea* u svrhu ubrzavanja rada i smanjenja prometa na mreži.
- Privremeni *cache* je memorija u kojoj preglednik, dok je aplikacija aktivna u memoriji, čuva dohvaćeni sadržaj.

Web preglednik

- Korisnik može upravljati *cacheom* djelomično, konfigurirajući postavke (npr. koliko diska *cache* može zauzimati, zabrana spremanja i sl.) ili "prisilno" brišući *cache*.
- Ali izvan toga, korisnik ne može upravljati time kako *browser* radi sa svojim *cacheom*. Stoga je prilikom razvoja i *debugiranja* web aplikacija potrebno voditi računa o tome.
- Preglednici pokušavaju "sakriti" *cache* datoteke od korisnika smještajući ih u zaštićene direktorije, kreirajući posebne datoteke binarnog formata i sl.

Chrome lokacija *cachea*:

- Windows 7/8/10/11: C:\Users\[USERNAME]\AppData\Local\Google\Chrome\User Data\Default\Cache
- Mac OS X: /Users/[USERNAME]/Library/Caches/Google/Chrome/

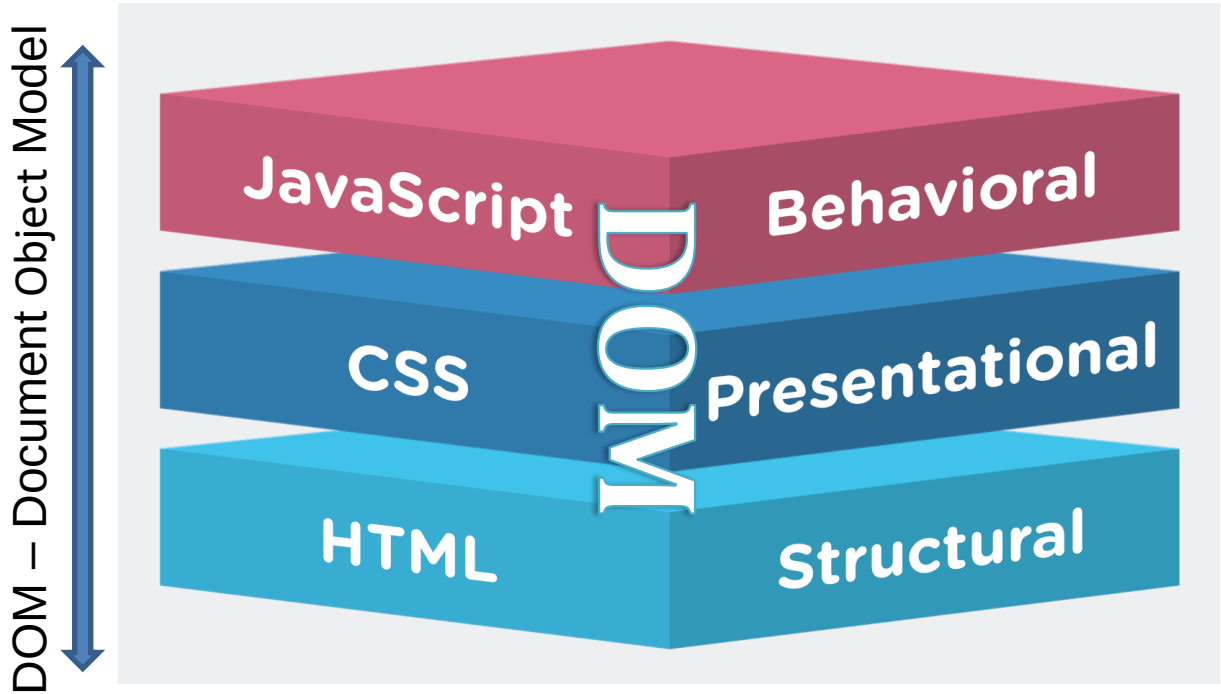
Web klijentske tehnologije

Web klijentske tehnologije

- Na raspolaganju nam je niz tehnologija, a kontinuirano se razvijaju nove.
- Tehnologije na strani web klijenta možemo podijeliti na:
  - Standardni dio preglednika (HTML, CSS, JavaScript, Ajax,…)
  - Potrebni dodatni mehanizmi (*plug-in*, *add-on*) za njihovo korištenje (Flash, JavaApplet, ...)
- Ili ih možemo podijeliti na:
  - Osnovno okruženje (JavaScript)
  - Dodatno okruženje (jQuery, AngularJS, Bootstrap, ….)

Web klijentske tehnologije

- Tehnologije na strani web klijenta možemo podijeliti i obzirom na namjenu:
  - Strukturiranje sadržaja
    - HTML (HyperText Markup Language)
  - Prezentacija sadržaja
    - CSS (Cascading Style Sheets)
  - Interaktivnost sadržaja
    - JavaScript



MEHANIZAM PRIKAZA

- U početku su preglednici bili monolitne aplikacije koje su koristile različite tehnike procesiranja teksta da bi parsirali HTML kôd u vizualnu prezentaciju.
- Danas su preglednici modularni i sastoje se od mehanizma prikaza i ostatka aplikacije.
- Mehanizam prikaza (*layout engine, rendering engine*) je softver koji "čita" web sadržaj (HTML, XML, slike, …) i prikaz sadržaja (CSS, XSL, …) i prikazuje (vizualno) formatirani sadržaj na ekranu. Mehanizam prikaza "iscrtava" sadržaj stranice na ekranu i ustvari radi najveći dio posla web klijenta.
- Mehanizam prikaza je modul u pretraživačima, e-mail klijentskim aplikacijama i sličnim aplikacijama koji trebaju prikazati web sadržaj.

MEHANIZAM PRIKAZA

Engine	Developer(s)	Software license	Leading application	Target application(s)	Programming language
Blink[note 1]	Google, Opera, Samsung, Intel, others[2]	GNU LGPL, BSD-style	Google Chrome	Google Chrome & Opera from 15.0	C++
Dillo	Dillo developers	GNU LGPL	Dillo	Dillo	C
EdgeHTML[note 2]	Microsoft	Proprietary	Edge	Edge	C++[3]
Gecko	Netscape/Mozilla Foundation	MPL	Mozilla Firefox	Mozilla Firefox & Mozilla Thunderbird	C++
Goanna[note 3]	Moonchild Productions	MPL	Pale Moon	Pale Moon & FossaMail	C++
GtkHTML[note 4]	GNOME	GNU LGPL	Novell Evolution	Novell Evolution	C
Hubbub	Andrew Sidwell	MIT[4]	NetSurf	NetSurf	C
iCab[note 4]	Alexander Clauss	Proprietary	iCab	iCab	?
KHTML	KDE	GNU LGPL	Konqueror	Konqueror & KMail	C++
NetFront	Access Co.	Proprietary	NetFront	NetFront	?
Presto	Opera Software	Proprietary	Opera	Opera[note 5]	C++[5]
Prince	YesLogic Pty Ltd	Proprietary	Prince	Prince (formerly called Prince XML)	Mercury
Tasman[note 4]	Microsoft	Proprietary	Microsoft Entourage	Internet Explorer for Mac & Microsoft Entourage	?
The Bat!	Ritlabs	Proprietary	The Bat!	The Bat!	Delphi
Trident[note 4]	Microsoft	Proprietary	Internet Explorer	Internet Explorer	C++[6]
Servo	Mozilla Foundation	MPL			Rust
WebKit[note 6]	Apple, KDE, Nokia, BlackBerry, Palm, others	GNU LGPL, BSD-style	Apple Safari	Apple Safari	C++
XEP	RenderX	Proprietary	XEP	XEP	Java

Različiti mehanizmi prikaza: https://en.wikipedia.org/wiki/Comparison_of_web_browser_engines

Web aplikacije

- Tradicionalne web aplikacije su radile na način da je sva dinamičnost odnosno izvršavanje različitih funkcionalnosti web aplikacije bilo na strani servera. Web klijent je prikazivao samo statički sadržaj vraćne sa web servera u obliku HTML stranice.
- Najveći nedostatak ovakvih aplikacija je sporost i veliko opterećenje web servera, jer se za bilo kakvu funkciju aplikacije, korisnikov zahtjev treba proslijediti web serveru, a odgovor servera se vraća web klijentu kao statički sadržaj.
- Razvoj tehnologija koje se koriste u web aplikacijama dosta je usmjeren u razvoj tehnologija koje omogućavaju prebacivanje dijela poslovne logike web aplikacija na stranu klijenta.

Web klijentske tehnologije

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

1. Prebacivanje dijela poslovne logike na stranu klijenta tako da dio posla obrade podataka obavlja web klijent, prebacivanjem dijela procesiranja na klijenta se rasterećuje web server i smanjuje promet na mreži.
2. Približavanje web aplikacija tradicionalnim desktop aplikacijama prema izgledu i funkcionalnosti grafičkog sučelja prema korisniku.

Web klijentske tehnologije

- Kod arhitekture softverskog sustava tipa server/klijent razlikujem dva tipa klijenata:
  - "Tanki klijent" (*thin client*) je minimalni klijent kod kojeg je obrada podataka odnosno poslovna logika aplikacije u potpunosti na strani servera.
  - "Debeli klijent" (*fat, rich, thick client*) je klijent koji obavlja i dio posla obrade podataka te je dio poslovne logike prebačen na stranu klijenta.
- Kada 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, obradu podataka s forme prije slanja i sl.

RIA aplikacije

- Korištenjem tehnologija koje su na raspolaganju na strani klijenta dio poslovne logike aplikacije može se prebaciti na klijenta. Na taj način se dobivaju tzv. bogate internet aplikacije (*Rich Internet Applications* - RIA). Pojam RIA uvela je Macromedia 2002. godine opisujući web aplikacije koje su koristile nove tehnologije na strani web klijenta.
- Postavljanjem različitih standarda vezanih uz web klijentske tehnologije, između ostalog i funkcionalnostima web aplikacija u *frontendu* na strani web klijenta, bavi se W3 konzorcij (https://www.w3.org/standards/), a danas sve više standarda preuzima WHATWG (*The Web Hypertext Application Technology Working Group*) (https://spec.whatwg.org/).

W3 konzorcijum

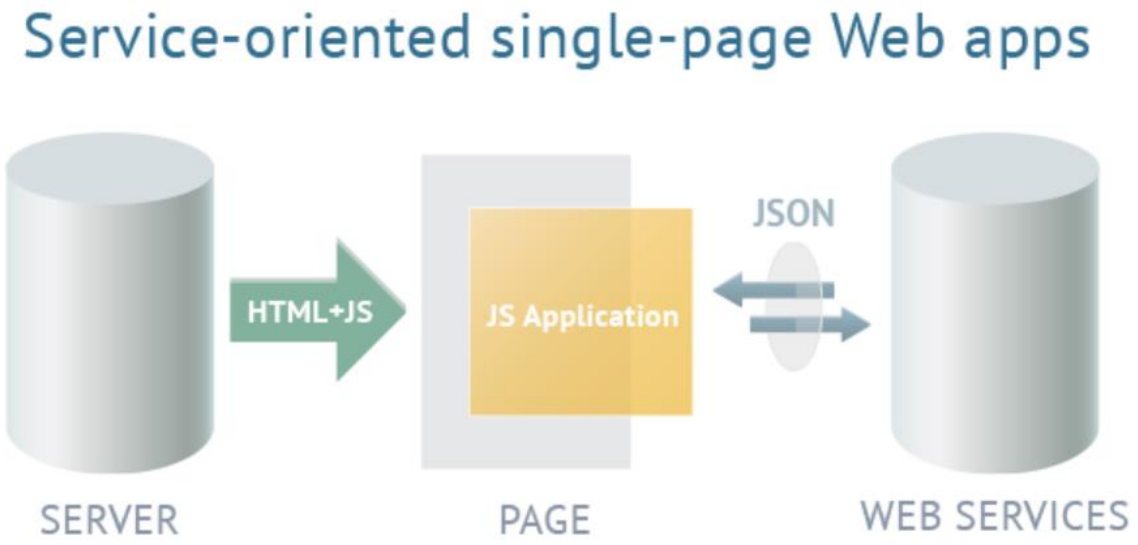
- W3C (*World Wide Web Consortium*) je organizacija osnovana 1994. kroz suradnju MIT-a i CERN-a uz podršku DARPA-e i Europske Komisije.
- Cilj W3C je razvoj web standarda, osiguravanje otvorenosti weba tj. mogućnost pristupa svima bez obzira na resurse, obrazovanje, …
- Razvoj weba ne može biti nečije vlasništvo, standardi trebaju biti otvoreni, a to je cilj i kompanijama vezanim uz web pa su tako članovi W3C i Google, Microsoft, Amazon, Facebook, Apple i još gotovo 400 kompanija zainteresiranih za razvoja weba. https://www.w3.org/membership/list/
- W3C standardi se nazivaju W3C preporuke (*recommendations*). W3C koordinira i rad drugih organizacija vezan uz web poput IETF-a.

RIA aplikacije

- Nedostatci RIA aplikacija:
  - Korisnik ima mogućnost zabraniti izvođenje JavaScript koda (chrome://settings/content/javascript?search=javascript) ili neke druge tehnologije (instalaciju ActiveX kontrole, java applet *plugin*, …) 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, ali se trebaju dohvatiti sa servera barem jednom. Ako ih klijent kešira ne treba ih dohvaćati ponovo.

Single-page web aplikacije

- Danas je uobičajeno rješenje korištenje tzv. *single-page* web aplikacija (SPA). Taj naziv označava da se s web servera dohvaća samo jedna cijelokupna HTML stranica unutar koje se prikazuju nove/promijenjene informacije koje se dohvaćaju HTTP(S) zahtjevima u "pozadini" otvorene stranice obično prema web servisima ili web API-jima korištenjem različitih tehnologija.
- Izvođenjem JavaScript koda na klijentu, s web servera se dohvaćaju različite informacije, često u JSON formatu i prikazuju korisniku "unutar" "prve" dohvaćene stranice.



Web klijentske tehnologije

- Web klijentske tehnologije se kontinuirano razvijaju i mijenjaju. Cijeli niz tehnologija koji je nekada bio popularan za programiranje na strani web klijenta danas se više gotovo i ne koristi poput *appleta*, Flasha, i dr.
- JavaScript jezik je danas standardno podržan od svih preglednika. Uz mehanizam prikaza koji koristi preglednik, JS *engine* je glavna karakteristika preglednika. (https://medium.com/@acparas/browsers-rendering-engines-js-engines-bea42b77a182)

	2022 1 Oct	2022 1 Nov	2022 1 Dec	2023 1 Jan	2023 1 Feb	2023 1 Mar	2023 1 Apr	2023 1 May	2023 1 Jun	2023 1 Jul	2023 1 Aug	2023 1 Sep	2023 1 Oct	2023 29 Oct
None	2.0%	1.9%	2.7%	2.0%	1.8%	1.7%	1.5%	1.4%	1.3%	1.3%	1.3%	1.2%	1.2%	1.2%
JavaScript	98.0%	98.0%	97.3%	98.0%	98.2%	98.3%	98.5%	98.6%	98.6%	98.7%	98.7%	98.8%	98.8%	98.8%
Flash	1.4%	1.4%	1.4%	1.5%	1.5%	1.5%	1.5%	1.4%	1.4%	1.4%	1.4%	1.3%	1.3%	1.3%
Java	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	<0.1%	0.1%	<0.1%	<0.1%	<0.1%

https://w3techs.com/technologies/history_overview/client_side_language/all

Flash 

- Flash je multimedijalna softverska platforma Adobe kompanije (Macromedia). Omogućava razvoj složenih multimedijalnih aplikacija, ali zahtjeva podršku za tu tehnologiju.
- Da bi se izvršio Flash sadržaj, web klijent treba imati neku verziju Flash *playera*. Flash Player je podržan na IE, FireFoxu itd. i distribuiran je kao besplatni plugin. Flash Player je u biti virtualni stroj koji izvršava bajt kôd zapisan u SWF datotekama.
- Kompanija Adobe je objavila da od 2020. više ne razvija i podržava Flash.
- Apple nikada nije ugradio podršku za Flash na svojim mobilnim uređajima.

https://naldzgraphics.net/inspirations/45-excellent-examples-of-flash-websites-design/

Silverlight

- Microsoft Silverlight je Microsoftova platforma (obustavljen razvoj 2012., a podrška za platformu kao što je ispravljanje bugova i sl. je najavljena do 2021.) za razvoj složenih multimedijalnih aplikacija za prikaz i animaciju površina grafičkih elemenata u web preglednicima. To je bio na neki način odgovor na Flash tehnologiju.
- Oslanja se na XAML (*eXtensible Application Markup Language*) markup jezik za kreiranje složenih, dinamičkih korisničkih sučelja, odnosno na .NET 3.x.
- Koristio se za razvoj grafičkog sučelja web aplikacija, a za izvršavanje Silverlight aplikacije potrebno je također imati instaliran *player* u web pretraživaču.

Java applet

- Java *applet* je Java aplikacija koja je napisana za izvođenje u pregledniku.
- Java *applet* se uključuje u HTML stranicu posebnim tagom u HTML 4, ali od HTML 5 više nije podržan.
- Za izvođenje *appleta* preglednik treba imati instaliran aplet *viewer* koji je postojao za većinu starijih preglednika u obliku *plugin*.
- Oracle koji je vlasnik ove tehnologije najavio je od verzije JDK 9 (2016.) obustavljanje podrške za Java *applete*.
- Sve ove tehnologije su danas zastarjele i moderni preglednici više ne uključuju podršku za njih.

Tehnologije na strani klijenta

- ActiveX aplikacija je temeljena na Microsoft COM (*Component Object Model*) modelu. Koristila se pretežno za razvoj dodatnih komponenti IE (ActiveX 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. Većina pretraživača ima integriranu podršku za SVG. (https://www.w3.org/Graphics/SVG/)