Promjena paradigme rasporeda elemenata na mrežnim stranicama implementacijom rešetaka stilskog jezika

Leh, Ana

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Humanities and Social Sciences / Sveučilište Josipa Jurja Strossmayera u Osijeku, Filozofski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:142:303032

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

FILOZOFSKI FAKULTET
SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

Repository / Repozitorij:

FFOS-repository - Repository of the Faculty of Humanities and Social Sciences Osijek

zir.nsk.hr

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Dvopredmetni studij nakladništva i informacijske tehnologije

Ana Leh

Promjena paradigme rasporeda elemenata na mrežnim stranicama implementacijom rešetaka stilskog jezika

Diplomski rad

Mentor: izv.prof.dr.sc. Boris Badurina

Sumentor: dr. sc. Tomislav Jakopec

Osijek, 2018.

Sveučilište J.J. Strossmayera u Osijeku
Filozofski fakultet Osijek
Odsjek za informacijske znanosti
Dvopredmetni studij nakladništva i informacijske tehnologije

Ana Leh

Promjena paradigme rasporeda elemenata na mrežnim stranicama implementacijom rešetaka stilskog jezika

Diplomski rad

Društvene znanosti, Informacijske i komunikacijske znanosti, Organizacija i informatika

Mentor izv.prof.dr.sc. Boris Badurina

Sumentor: dr. sc. Tomislav Jakopec

Osijek, 2018.

Sažetak

Tri godine trebalo je da nastane prvi stilski jezik nakon dolaska mrežnih stranica. No ispravan način postavljanja rasporeda elemenata na njoj nije došao još mnogo godina nakon toga. Prve mrežne stranice imale su unaprijed postavljenu veličinu stranice i nisu se mijenjale na temelju širine preglednika. U kasnim 90.-ima te ranim 2000.-ima pojavljuje se tablični raspored elemenata koji je od strane stručnjaka negativno prihvaćen. Nakon njega slijedi jedan od bolje prihvaćenih rasporeda elemenata, koji je umjesto fiksnih mjernih jedinica prešli na relativnu mjernu jedinicu, odnosno postotke: tečni raspored elemenata. Slijedi prilagodljivi dizajn koji je omogućio da isti kod rezultira stranicama prilagođenim za računala, tablete i mobitele te je postao jedan od najvećih otkrića u povijesti dizajna na mrežnim stranicama. Fleksibilni raspored elemenata prva je naznaka ispravnog rasporeda elemenata čija su svojstva zamijenila neke trikove koja su uključivala svojstva nenamijenjena za raspored elemenata. Dok je fleksibilni raspored elemenata postajao sve popularniji, nekoliko stručnjaka marljivo je radilo na rasporedu elemenata implementacijom rešetaka, za koji uvjeravaju da je napokon kompletno ispravan raspored elemenata. Rešetke su donijele pregršt prednosti, a omogućeno je i njihovo kombiniranje s drugim alatima kao što su okvirni sustavi, pretprocesori i postprocesori stilskog jezika. Budućnost je nepoznata, no predviđa se da su rešetke novi smjer mrežnih stranica prema kojima razvojni programeri trebaju ići. Najvažnije od svega je da korisnici shvate kako rešetke ne donose rješenja za cijeli dizajn na mrežnim stranicama, već da je korištenje različitih alata i modula iz stilskog jezika pravi odgovor za modernu i jedinstvenu mrežnu stranicu danas.

Ključne riječi: *raspored elemenata, rešetke, alati, mrežna stranica*

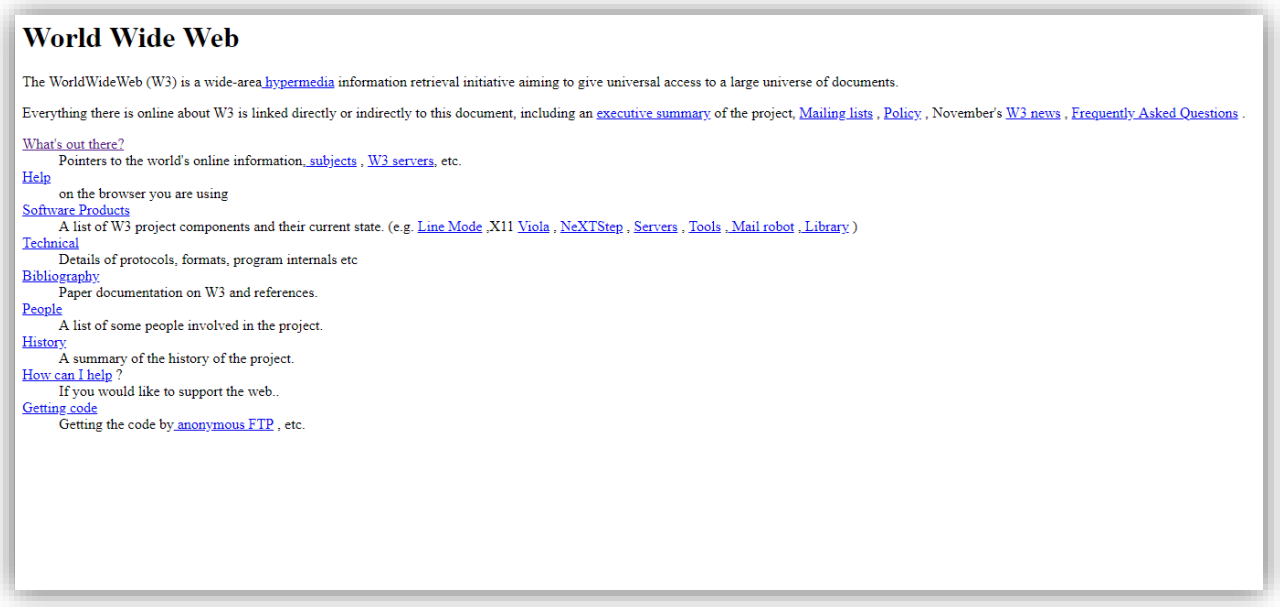
Sadržaj

1. Uvod ... 1
2. Povijest rasporeda elemenata na mrežnim stranicama ... 3
2.1. Statični raspored elemenata ... 3
2.2. Tablični raspored elemenata ... 3
2.3. Tečni raspored elemenata ... 4
2.4. Raspored elemenata i prilagodljivi dizajn ... 5
2.5. Fleksibilni raspored elemenata ... 7
3. Raspored elemenata implementacijom rešetaka ... 9
3.1. Terminologija rešetaka stilskog jezika ... 9
3.1.1. Prostor rešetaka ... 10
3.1.2. Stavka rešetaka ... 10
3.1.3. Linija rešetaka ... 11
3.1.4. Ćelija rešetaka ... 12
3.1.5. Traka rešetaka ... 13
3.1.6. Područje rešetaka ... 14
3.1.7. Međuprostor rešetaka ... 15
3.2. Definiranje rešetaka stilskog jezika ... 16
3.2.1. Eksplicitni raspored elemenata implementacijom rešetaka ... 16
3.2.2. Frakcijska jedinica ... 17
3.2.3. Imenovanje područja rešetaka ... 18
3.2.4. Stenografska svojstva položaja rešetke i predloška rešetke ... 19
3.2.5. Implicitni raspored elemenata implementacijom rešetaka ... 20
3.2.6. Ponavljanje linija rešetaka ... 20
3.2.7. Automatsko ponavljanje i pristajanje ... 21
3.2.8. Funkcija minmax() ... 22
3.3. Upiti svojstava ... 22

4. Načini pisanja 23
4.1. Latinski sustav 23
4.2. Arapski sustav 24
4.3. Hanski sustav 25
4.4. Mongolski sustav 26
5. Fleksibilni raspored elemenata i raspored elemenata implementacijom rešetaka 27
6. Alati pristupnog dijela i raspored elemenata implementacijom rešetaka 30
6.1. Okvirni sustavi 30
6.2. Pretprocesori 31
6.3. Postprocesori 32
7. Budućnost rasporeda elemenata implementacijom rešetaka 33
8. Zaključak 35
LITERATURA 37
PRILOZI 39
Prilog 1. Hrvatsko - engleski rječnik pojmova 40
Kronološki 40
Abecedni 41
Prilog 2. Popis CSS svojstava rasporeda elemenata implementacijom rešetaka i fleksibilnog rasporeda elemenata 43
Raspored elemenata implementacijom rešetaka 43
Fleksibilni raspored elemenata 43

1. Uvod

Dana 6. kolovoza 1991. prva mrežna stranica postala je dostupna i vidljiva svima. Tvorac stranice, kao i uostalom tvorac World Wide Web-a, koji je omogućio dostupnost i samo stvaranje mrežnih stranica, bio je Tim Berners-Lee. Stranica s sastojala od naslova te teksta koji je davao upute kako stvoriti mrežnu stranicu te objašnjavao što je to hipertekst (Slika 1).



Slika 1. Izgled prve mrežne stranice

Sami vizualni prikaz stranice podsjeća na tekstualni dokument, sličan primjerice Word dokumentu. Od samog početka postojala je potreba za većim utjecajem na izgled stranice od strane samih autora, no prvi pravi prijedlog stilskog jezika događa se 1994., objavljuje ga Håkon Wium Lie, a naziva ga Cascading Style Sheet (CSS). Prvi nacrt CSS-a doveo je do zaključka kako HTML nikad ne bi trebao postati jezik za definiranje izgleda stranice, te da je za to potreban drugi mehanizam, koji je pronađen u stilskom jeziku. Otada, pa sve do danas, CSS se značajno razvio, a skupa s njim i izgled, odnosno grafički dizajn mrežnih stranica. Umjesto mrežne stranice koja podsjeća na običan tekstualni dokument, danas imamo mrežne stranice u bojama, vizualno ukrašene slikama, animacijama, raznim fontovima i slično. No, kada pogledamo raspored elemenata na stranicama, on je gotovo pa jednak od samih početaka CSS-a. Dolaskom blogova, nastao je raspored elemenata koji je omogućio smislenu organizaciju većeg broja elemenata na

mrežnim stranicama. Dolaskom razvojnih cjelina, takozvanih okvirnih sustava[1], omogućena je prilagodljivost[2] mrežnih stranica. Osim ta dva događaja, raspored elemenata na stranicama[3] nije se značajnije mijenjao. Postoje razni trikovi kako poboljšati raspored elemenata na stranici, atributi kao što su *floats* ili trikovi kao što su *clearbox*, koji će biti detaljnije opisani kasnije u radu, no zaključak raznih stručnjaka diljem svijeta je sljedeći: raspored elemenata na stranici od samih početaka bio je pokvaren, i svi trikovi i načini korišteni do sada bili su kao da osoba pokušava šalicu razbijenu u desetke dijelova zalijepiti ljepilom: neke pukotine i dalje će se nazirati, a pokuša li jedan razbijeni dio odvojiti od polomljene šalice, riskira odvajanje i rušenje i ostalih dijelova. Logičan je zaključak da stručnjaci pokušavaju naći rješenje za „pokvareni raspored elemenata stranice“ te način kako ga popraviti. U 2012. godini predstavljen je operativni sustav Windows 8, od strane Microsoft-a i, iako nije postigao naklonost korisnika, predstavio je nešto za što se vjeruje da će omogućiti popravak rasporeda elemenata na stranici: raspored elemenata implementacijom rešetaka stilskog jezika[4]. Te rešetke stilskog jezika prisutne su u izborniku Windows-a 8, koji je po prvi put omogućio dvodimenzionalnost stranice, koja je navedena kao jedna od glavnih prednosti najnovijeg rasporeda elemenata. Pet godina bilo je potrebno da rešetke stilskog jezika postanu dostupne za mrežne stranice te budu podržane u većini najpoznatijih preglednika, a najpoznatiji pobornici, kao što su Rachel Andrew, Morten Rand-Hendriksen, Jen Simmons, Eric Meyer i ostali, uvjeravaju razvojni svijet informacijske tehnologije da će rešetke stilskog jezika značiti kompletno novo poglavlje za grafički izgled mrežnih stranica.

Svrha ovog rada je upoznati korisnika s rasporedom elemenata implementacijom rešetaka stilskog jezika, predstaviti njegove dijelove, kao i kompletnu terminologiju, usporediti ga s drugim tehnologijama i u konačnici zaključiti što sve raspored elemenata implementacijom rešetaka stilskog jezika donosi u razvojni svijet informacijskih tehnologija.

[1] eng. framework
[2] eng. responsive
[3] eng. layout
[4] eng. grid layout

2. Povijest rasporeda elemenata na mrežnim stranicama

Prvotni cilj mrežnih stranica bio je podijeliti s ljudima razne dokumente zbog čega je Internet na svojim počecima bio samo zbirka raznih tekstova. Prva promjena, kao i svaka nakon toga, nastaje nakon što se javlja potreba većeg broja korisnika za nekim novitetima. Prva potreba bile su različite boje i fontovi u tekstu, što se prvo i dogodilo. Potom je došla potreba za većom kontrolom nad samim izgledom mrežnih stranica od strane korisnika, nakon čega je predstavljen stilski jezik CSS koji je to i omogućio. Svaka nova konzistentna i učestala potreba ili upotreba novih elemenata dovodila je do novih poboljšanja i mogućnosti u stilskom jeziku. Iako se kroz kratku povijest CSS-a nije pojavilo rješenje za jednostavan i organiziran raspored elemenata na mrežnim stranicama, on se ipak razvijao kroz razne elemente i zaobilazne puteve. Nove specifikacije dovodile bi do novih „trikova“, ali niti jedan od njih nije davao rješenje za ispravnim rasporedom elemenata, zbog čega se konstantno tražio novi način.

2.1. Statični raspored elemenata

Prve mrežne stranice koristile su unaprijed postavljenu veličinu stranice i nisu se mijenjale na temelju širine preglednika, odnosno bile su statične. Imale su trajnu širinu od 960 piksela i nisu se mijenjale. Stranice su izgledale tako više godina, sve do pojave medijskih upita[5] i prilagodljivih mrežnih stranica.[6] Danas je upotreba statičnog rasporeda elemenata vrlo rijetka.

2.2. Tablični raspored elemenata

U kasnim 90.-ima te ranim 2000.-ima, razvojni programeri posegnuli su za nekim, poprilično ozbiljnim trikovima kako bi dobili prividno uspješni raspored elemenata na mrežnim stranicama. Tehnologije koje su postale popularne i korištene su kao trikovi, ali u kompletno krivom smislu, bile su tablice i okviri.[7] Za vrijeme razvoja HTML-a i pokušaja izdavanja novih verzija kao što su HTML+ ili HTML 2.0, tablice su bile široko rasprostranjene i opsežno usvojene, no ne i službeno dokumentirane. Jedna od najvećih prednosti tablica, koje su poslužile kod rasporeda elemenata na mrežnim stranicama, bila je mogućnost određivanja širine i visine svakog pojedinog stupca. No

[5] eng. media query
[6] Usp. Pettit, Nick. Which Layout? Static, Liquid, Adaptive, or Responsive.URL: http://blog.teamtreehouse.com/which-page-layout (2018-07-20)
[7] Usp. Tables for Layout? Absurd. URL: https://thehistoryoftheweb.com/tables-layout-absurd/ (2018-07-21)

tablice su bile namijenjene za podatke, a ne za određivanje rasporeda elemenata na mrežnim stranicama. Prikaz u preglednicima nije bio niti približno uspješan, jer preglednici nisu mogli u trenutku učitavanja stranice izračunati mjere stupaca, a cijelu stranicu i preglednici i drugi uređaji tumačili su kao tablicu te pokušali analizirati podatke unutar nje što je često dovodilo do raspada cijele stranice.[8] Slična situacija događala se i kad su se umjesto tablica koristili okviri. Bez obzira na to, razvojni programeri u tim tehnologijama vidjeli su najbolje alternative za raspored elemenata na stranicama. U konačnici, službeni standardi su ih dostigli, što je prisililo ljude da se maknu od takve prakse.

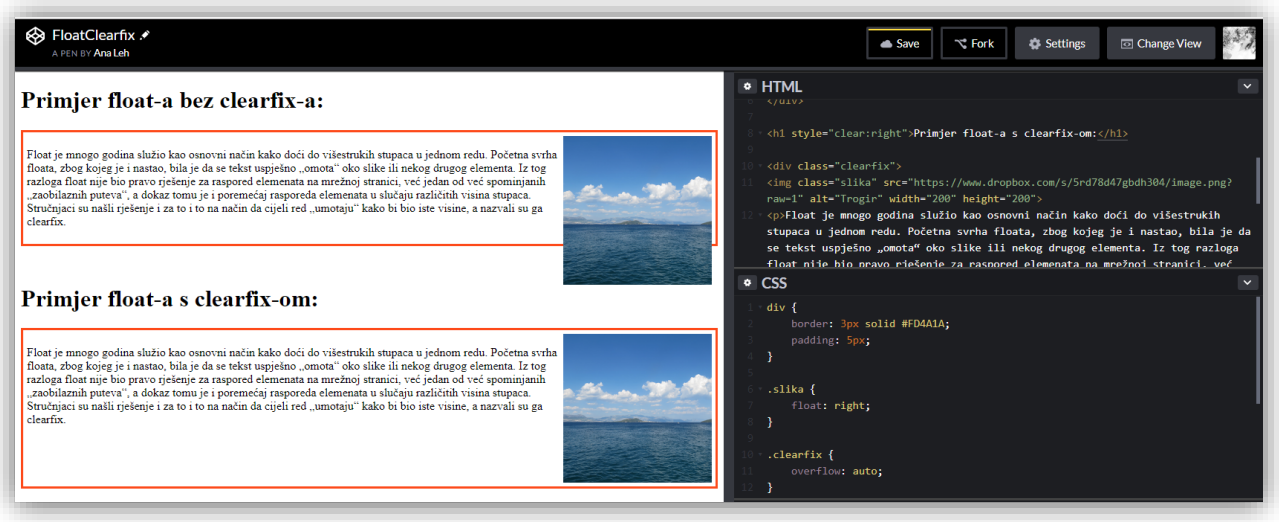
2.3. Tečni raspored elemenata

Iako je pojava stilskog jezika unaprijedila izgled stranica, Internet se i dalje mogao predstaviti kao jedna velika knjiga čije su mrežne stranice izgledale kao stranice tiskane u knjizi, uz dodatak hiperteksta. Dio stručnjaka koji su vidjeli dalje od toga, naveli su neke karakteristike Interneta, koje su prije bile predstavljene kao problem, kao glavne prednosti. John Allsopp, autor knjige „A Dao of Web Design“, govori kako bi dizajneri i programeri trebali prigrliti te prednosti, od kojih je najvažnija fleksibilnost, umjesto da ju pokušaju kontrolirati.[9]

Prvi konkretan korak prema tome bila je promjena mjere fontova s fiksne na relativnu (mjerna jedinica em). Umjesto statičnog rasporeda elemenata iskazanih u pikselima, počeli su se koristiti postotci što je dovelo do tečnog rasporeda elemenata[10]. Kod takvog rasporeda stranica bi se širila i skupljala, zavisno od veličine zaslona na kojem je stranica otvorena. Takav raspored elemenata postao je iznimno popularan i kompletno je promijenio način razmišljanja o mrežnim stranicama. No, sam raspored elemenata na stranici i dalje je ostajao isti, sve dok se nije shvatila prednost CSS atributa *float*. *Float* je mnogo godina služio kao osnovni način kako doći do višestrukih stupaca u jednom redu. Početna svrha *floata*, zbog kojeg je i nastao, bila je da se tekst uspješno „omota“ oko slike ili nekog drugog elementa. Iz tog razloga *float* nije bio pravo rješenje za raspored elemenata na mrežnoj stranici, već jedan od već spominjanih „zaobilaznih puteva“, a dokaz tomu je i poremećaj rasporeda elemenata u slučaju različitih visina stupaca. To je rezultiralo

[8] Usp. Isto.
[9] Usp. The (Mostly) Complete History of Layout on the Web Part 1: Liquid Cool. URL: https://thehistoryoftheweb.com/mostly-complete-history-layout-web-part-1-liquid-cool/ (2018-07-21)
[10] eng. liquid layout/fluid layout

još jednim trikom i to na način da se cijeli red „umota“ kako bi bio iste visine, a nazvali su ga *clearfix* (Slika 1).



Slika 2. Primjer CSS atributa *float* s i bez *clearfix-a*

U 2004. Cameron Adams predlaže novi raspored elemenata kojeg je nazvao raspored elemenata ovisno o razlučivosti[11]. Koristeći JavaScript, Adams je osmislio tehniku otkrivanja rezolucije zaslona kojeg bi korisnik koristio, a zatim izmijenio stil na temelju rezultata.[12] Tehnika nikad nije bila široko prihvaćena, ali je predvidjela buduću popularnu tehnologiju koja se koristi i danas, a biti će objašnjenja u idućem potpoglavlju: medijske upite.

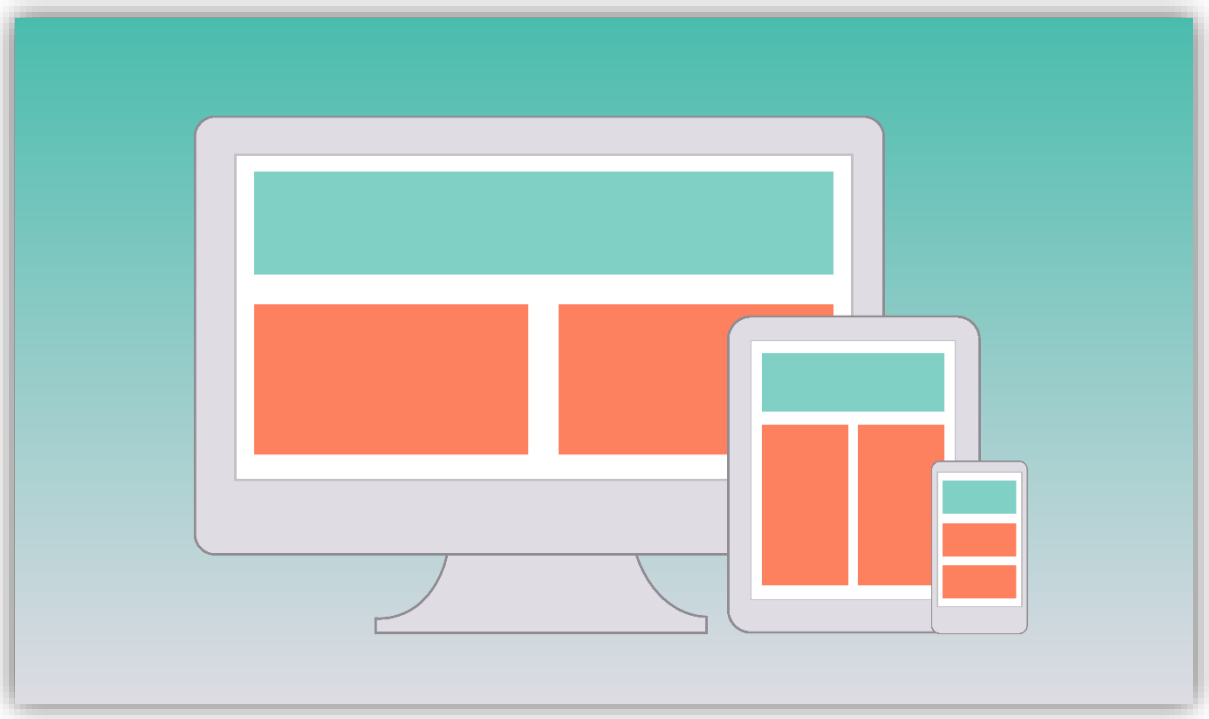
2.4. Raspored elemenata i prilagodljivi dizajn[13]

„Povijest prilagodljivog dizajna je povijest rasporeda elemenata na mrežnim stranicama.“[14] Fleksibilni dizajn postao je dominantan nakon što su se razvojni programeri uspjeli odmaknuti od utjecaja dizajna tiskanih materijala. Od samih početaka te dominacije, u pozadini je W3C[15], međunarodna zajednica koja razvija otvorene standarde kako bi se osigurao dugoročni razvoj mrežnih stranica, radio na novoj tehnologiji za CSS, a nazvali su ju medijski upiti. Medijski upiti

[11] eng. resolution dependent layout
[12] Usp. Isto.
[13] eng. responsive design
[14] The (Mostly) Complete History of Layout on the Web Part 2: Responsive Design. URL: https://thehistoryoftheweb.com/mostly-complete-history-flexible-web-layout-part-2-responsive-design/ (2018-07-21)
[15] World Wide Web Consortium

nudili su, i danas nude, mogućnost specificiranja karakteristika odabranog uređaja, kao što su razlučivost i širina preglednika, kao i promjenu CSS atributa na temelju rezultata. Tako bi razvojni programeri bili u mogućnosti napraviti kompletno drugi stil za uređaje velike razlučivosti, kao i za one male. Iako su zaživjeli tek u 2009., predlagani su sve od samih početaka CSS-a i to već od strane tvorca CSS-a. Isti su odbijeni u prvoj verziji stilskog jezika iz razloga što su razvojni programeri prvu verziju htjeli učiniti što jednostavnijom za primjenu. U drugoj verziji CSS-a medijski upiti bili su implementirani, no ne i u preglednicima što je ponovno odgodilo njihovu upotrebu. Konačna prekretnica dogodila se pojavom iPhone-a. Steve Jobs predstavio je prvi iPhone 2007. godine s vlastitim Safari preglednikom koji je radio isključivo s najnovijim verzijama HTML-a i CSS-a. Mrežne stranice bile su preširoke za zaslon od 320 piksela, pa čak i s tečnim rasporedom elemenata.[16] Prvo rješenje bilo je napraviti kompletno nove stranice namijenjene isključivo za iPhone. No, u isto vrijeme postojali su planovi i za iPad čiji je zaslon bio veći te se očekivao i dolazak raznih novih uređaja različitih veličina, što je značilo da bi se trebala praviti nova mrežna stranica za svaki novi tip uređaja. To se nije svidjelo Ethan-u Marcotte-u, koji je s timom u tvrtki Filament Group radio na novoj mrežnoj stranici za Boston Globe. Tvrtka nije htjela imati dva različita koda za istu stranicu. Marcotte je također bio i član WBS-a[17], u kojem je s ostalim kolegama bio svjestan da je iPhone samo još jedan dokaz kako su Internet i njegove mrežne stranice fleksibilne. S obzirom na sve te činjenice, Marcotte je odlučio naći rješenje kombinirajući više tehnologija koje su se pojavljivale u povijesti. U konačnici je došao do već spomenutih medijskih upita. Koristeći njih, Marcotte je predstavio novi pristup rasporedu elemenata na mrežnim stranicama, a nazvao ga je prilagodljivi dizajn. Dva dizajna, jedan za veće zaslone, drugi za manje, i samo jedan kod (HTML dokument).[18] Predstavio je prilagodljivi dizajn (Slika 2) u članku za elektronički magazin *A List Apart*. S tim se dogodio ogromni zaokret u industriji, a prilagodljivi dizajn postao je jedan od najvećih otkrića u povijesti dizajna na mrežnim stranicama.[19]

[16] Usp. Isto.
[17] Web Standards Project
[18] Usp. Isto.
[19] Usp. Isto.



Slika 3. Prilagodljivi dizajn

2.5. Fleksibilni raspored elemenata[20]

Iako su razvojni programeri stvarali funkcionalne i privlačne mrežne stranice koristeći trikove koji su uključivali CSS atribute *float*, *margin* i *position*, bili su svjesni da prvotna svrha tih atributa nije bila za bolji raspored elemenata i da nisu nastali zbog istoga. Kako je CSS sazrijevao te su preglednici postojali moćniji, pojavile su se nove alternative za raspored elemenata,[21] ovaj put isključivo stvorene za sam raspored elemenata. Raspored elemenata koji je postao široko rasprostranjen nazvan je fleksibilni raspored elemenata, a glavna svrha mu je učiniti elemente fleksibilno promjenjivima kako bi se bolje prilagodili dostupnom prostoru, bez potrebe za već spomenutim atributima. Ideja o fleksibilnom rasporedu elemenata prvi put se pojavljuje prije 2008. godine, a prvi nacrt objavljen je u 2009. godini. No, algoritam je bio spor, komponente u nacrtu bile su nejasne te su rezultati bili različiti na različitim preglednicima.[22] U 2011. Tab Atkins postaje urednik fleksibilnog rasporeda elemenata i unutar iste godine objavljuje dva nova nacrta. Verzija iz 2012. bila je predlagana od strane W3C-a, a najnoviji je objavljen u 2015. godini nije bio predstavljen kao konačna verzija, već i dalje kao nacrt, što je značilo da može biti ažuriran ili

[20] eng. flexbox
[21] Gasston, Peter. The Book of CSS3: A Developer's Guide to the Future of Web Design. San Francisco: No Starch Press, 2015. Str. 185.
[22] Usp. Flexbox History. URL: http://annairish.github.io/historicizing/history (2018-07-23)

zamijenjen u bilo kojem trenutku.[23] Fleksibilni raspored elemenata početak je novog smjera gdje su stvoreni novi atributi čija je svrha vezana isključivo za raspored elemenata te početak zatvaranja svih „zaobilaznih puteva“ otvaranjem novih, direktnih puteva do cilja, odnosno čistog i jednostavnog koda koji omogućuje raspored elemenata kakvog su razvojni programeri otpočetka priželjkivali. Ono što je uslijedilo nakon fleksibilnog rasporeda elemenata je, već spomenuti, raspored elemenata implementacijom rešetaka stilskog jezika, od kojeg se očekuje da napokon omogući stvaranje stranice i njenog izgleda bez ikakvih trikova, a hoće li to očekivanje ispuniti, samo budućnost može reći. Na razvojnim programerima je da se s njim upoznaju, iskušaju ga te odluče je li takav način rasporeda elemenata stvarno najbolji, a od čeg se raspored elemenata implementacijom rešetaka stvarno sastoji, koja mu je terminologija i kako funkcionira, biti će objašnjeno u narednim poglavljima.

[23] Usp. Isto.

3. Raspored elemenata implementacijom rešetaka

Kao što je već ranije navedeno, izlazak Windows 8 nije izazvao oduševljene kod korisnika i stručnjaka, no dao je viziju za kompletno novi raspored elemenata, no što je još važnije, način kako do njega doći. Tako je Microsoft istovremeno s objavom Windows 8 objavio i prijedlog za raspored elemenata implementacijom rešetaka stilskog jezika, u studenom 2012. godine. Istovremeno je rasla popularnost fleksibilnog rasporeda elemenata, koji se činio kao konačno rješenje za pravi raspored elemenata na mrežnim stranicama. No, fleksibilni raspored elemenata mogao se kodirati u jednom smjeru, horizontalnom ili vertikalnom, odnosno jednodimenzionalno. Raspored elemenata implementacijom rešetaka omogućio je dvodimenzionalnost elemenata (više o razlikama između ta dva rasporeda elemenata u idućim poglavljima). I dok je fleksibilni raspored elemenata postajao sve popularniji, nekoliko stručnjaka tiho je radilo na kompletnoj implementaciji rešetaka stilskog jezika, sigurni da je to pravi način kako doći do ispravnog rasporeda elemenata.[24] Tako je prvi put, u 2017. godini, omogućen CSS atribut *display:grid;*, i po Mortenu Randu-Handriksenu, mrežna stranica prvi put ima ispravan raspored elemenata.

U idućim potpoglavljima opširno će se opisati i praktično prikazati cijela terminologija rešetaka stilskog jezika te pružiti čitatelju uvid u njenu funkcionalnost i jednostavnost, od opisa nove mjere koja rešetke čine toliko poželjnima, do opisa elemenata u rešetkama stilskog jezika koje tek trebaju biti implementirane i podržane u svim poznatijim preglednicima.

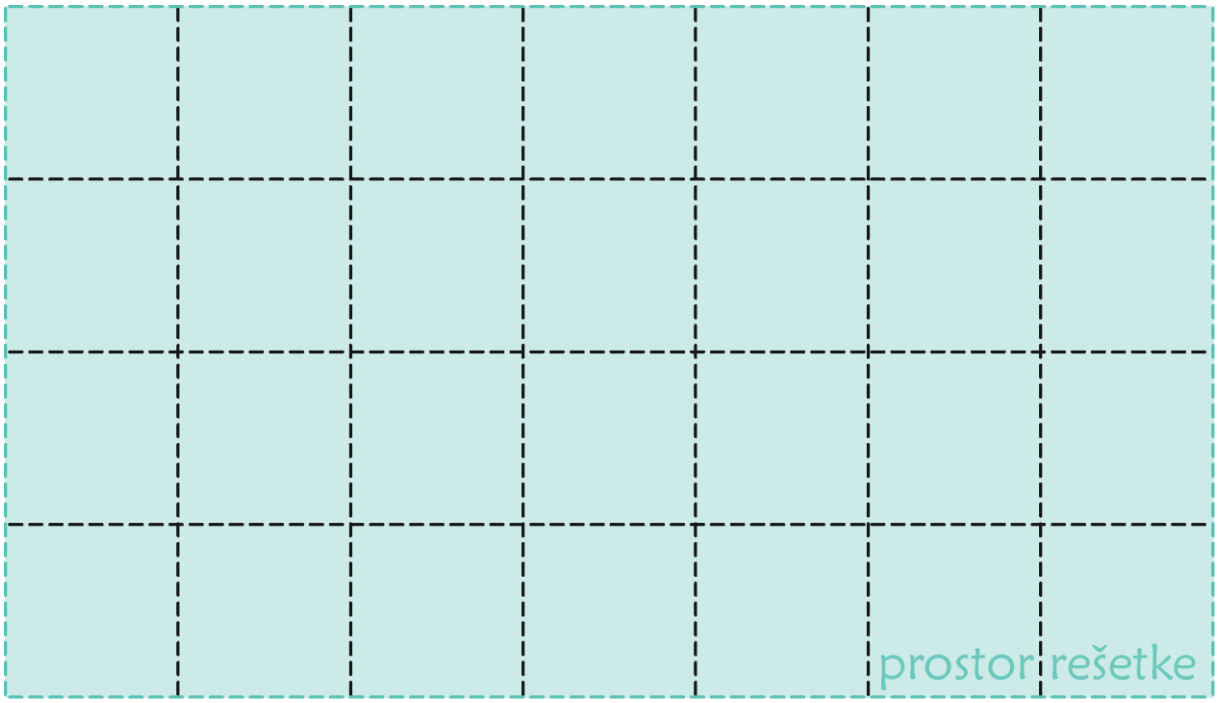
3.1. Terminologija rešetaka stilskog jezika

S pojavom rasporeda elemenata implementacijom rešetaka pojavili su se i novi pojmovi koji su dio njega, a prije nisu postojali. Osnovni pojmovi, od kojih se mrežna stranica implementirana rešetkama sastoji su: **prostor** rešetaka[25], **stavka** rešetaka[26], **linija** rešetaka[27], **ćelija** rešetaka[28], **traka** rešetaka[29], **područje** rešetaka[30], **međuprostor** rešetaka[31].

[24] Usp. Rand-Hendriksen, Morten. CSS Grid Changes Everything (About Web Layouts). WebCamp Zagreb 2017. Plaza centar. Zagreb, 7.10.2017. [Predavanje]
[25] eng. grid container
[26] eng. grid item
[27] eng. grid line
[28] eng. grid cell
[29] eng. grid track
[30] eng. grid area
[31] eng. grid gap / gutters

3.1.1. Prostor rešetaka

Prostor rešetaka je element koji djeluje kao granica i postavlja dimenzije rešetke (Slika 3).[32] Uspostavlja kontekst oblikovanja rešetke, odnosno područje u kojem se stvaraju rešetke. Elementi su postavljeni u skladu s pravilima rešetke umjesto prijašnjih rasporeda elemenata. Može se zamisliti kao tablica koja stvara kontekst tablice unutar nje same. Iako je usporedba pogodna, ipak nije jednaka i rešetke su daleko moćnije od same tablice.[33]

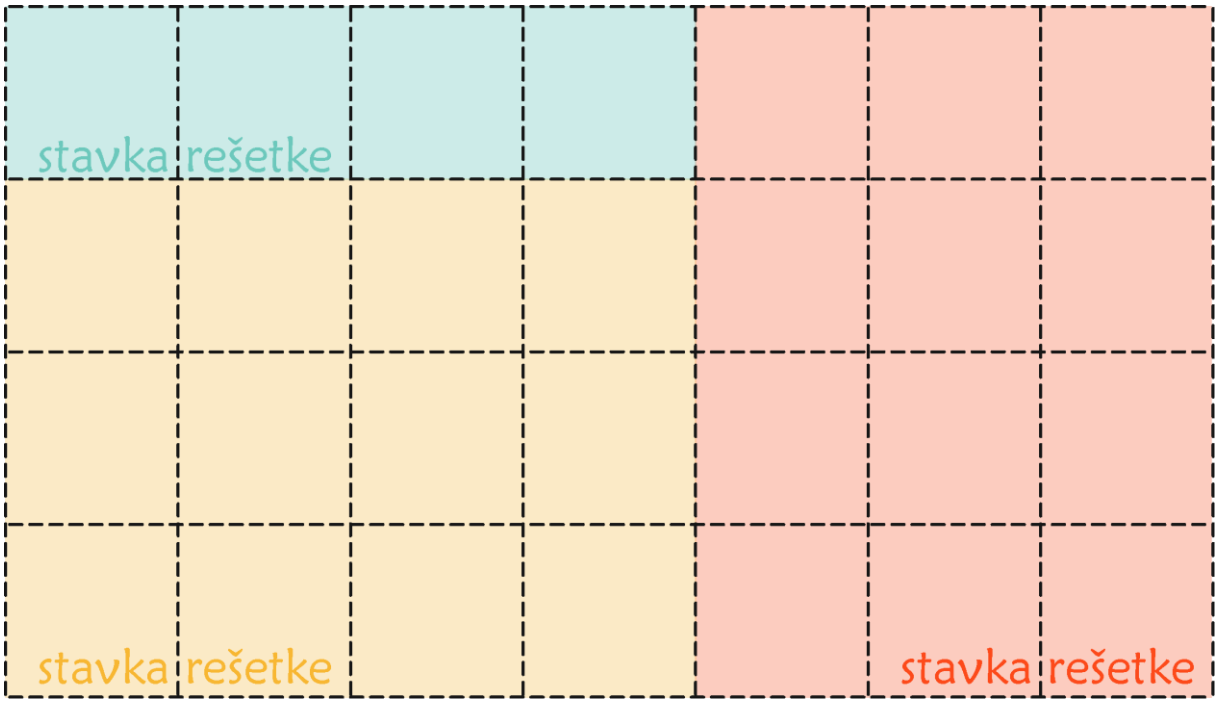


Slika 4. Prostor rešetke[34]

3.1.2. Stavka rešetaka

Stavka rešetaka je element koji slijedi ili drugačije rečeno, je ispod prostora rešetki („potomak[35]“ prostora rešetki)[36] Sudjeluje u rasporedu elemenata implementacijom rešetaka i dodatno ga oblikuje (Slika 4).[37]

[32] Usp. Gasston, Peter. Nav. dj. Str. 210.
[33] Usp. Mayer, Eric A. Grid Layout in CSS: Interface Layout for the Web. Beijing, Boston, Farnham, Sebastopol, Tokyo: O'Reilly, 2016. Str. 4.
[34] Usporediti s https://youtu.be/ojKbYz0iKQE?t=31s
[35] eng. child element
[36] Usp. Rand-Hendriksen, Morten. Nav. dj.
[37] Usp. Mayer, Eric. Nav. dj. Str. 4.

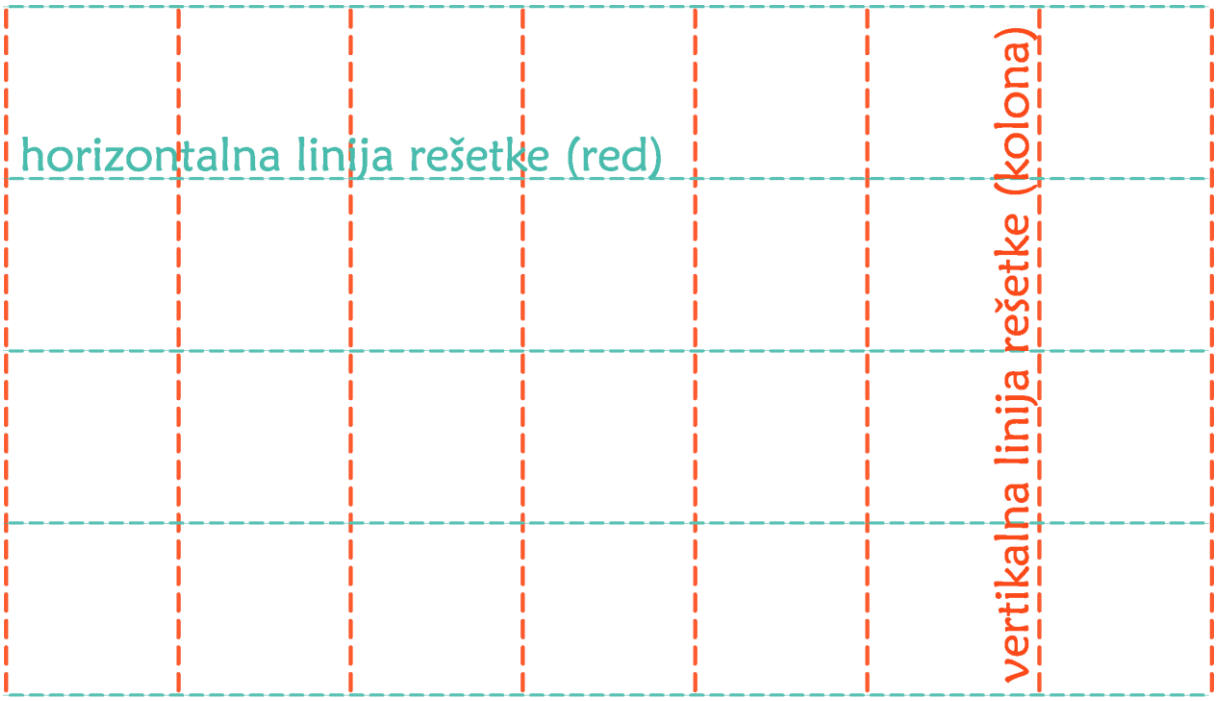


Slika 5. Stavka rešetke[38]

3.1.3. Linija rešetaka

Linije rešetaka su linije razdjeljivanje između redaka i stupaca (Slika 5). Linije su pojmovne, što znači da nisu uistinu prikazane na mrežnoj stranici.[39] Na svakoj strani stupca ili retka postoji linija rešetke. Stavka retka se odnosi na linije retka kako bi odredila svoj položaj unutar rešetke pomoću svojstava za postavljanje rešetke. Ta svojstva omogućuju autoru da odredi položaj stavke rešetke, a imena svojstava su: *grid-row-start*, *grid-column-start*, *grid-column-end*, *grid-row*, *grid-column* i *grid-area*.[40]

[38] Usporediti s https://youtu.be/ojKbYz0iKQE?t=46s
[39] Usp. Gasston, Peter. Nav. dj. Str. 210.
[40] Usp. CSS Grid Layout Module Level 1, 2017. URL: https://www.w3.org/TR/css-grid-1/ (2018-07-25)



Slika 6. Linija rešetaka[41]

3.1.4. Ćelija rešetaka

Ćelija rešetke je stvorena od svakog križanja kolone i retka, kao ćelije u tablici.[42] Ćelija je najmanja jedinica rešetke prilikom postavljanja elemenata rešetke.[43] Svaka ćelija graniči s četiri linije, niti jedna linija ne prolazi kroz nju (Slika 6).[44]

[41] Usporediti s https://youtu.be/ojKbYz0iKQE?t=1m
[42] Usp. Gasston. Nav. dj. Str. 210.
[43] Usp. CSS Grid Layout Module Level 1. Nav. dj.
[44] Usp. Mayer, Eric. Nav. dj. Str. 6.

Slika 7. Ćelija rešetke[45]

3.1.5. Traka rešetaka

Traka rešetaka je prostor između dva ili više retka rešetaka (Slika 7). Trake reda su horizontalne, dok su trake kolona vertikalne.[46] Svaka traka rešetke ima funkciju odrediti veličinu, koja određuje koliko će kolone ili redovi biti široki ili visoki. Susjedne trake rešetaka mogu se odvojiti međuprostorom rešetaka, no inače su spojene.[47]

[45] Usporediti s https://youtu.be/ojKbYz0iKQE?t=1m17s
[46] Usp. Rand-Hendriksen, Morten. Nav. dj.
[47] Usp. CSS Grid Layout Module Level 1. Nav. dj.

Slika 8. Traka rešetaka[48]

3.1.6. Područje rešetaka

Područje rešetaka je jedna ili više ćelija koja označava prostor u kojem će stavka rešetke biti postavljena.[49] Kao i ćelija rešetke, graniči s četiri linije koje sudjeluju u određivanju veličine traka rešetki (Slika 8). Područje rešetke može imati svoje vlastito ime koje odredi korisnik pomoću CSS svojstva *grid-template-areas* (više u potpoglavlju Imenovanje područja rešetaka). Stavka rešetke se dodjeljuje području rešetke pomoću svojstva *grid-placement*.[50]

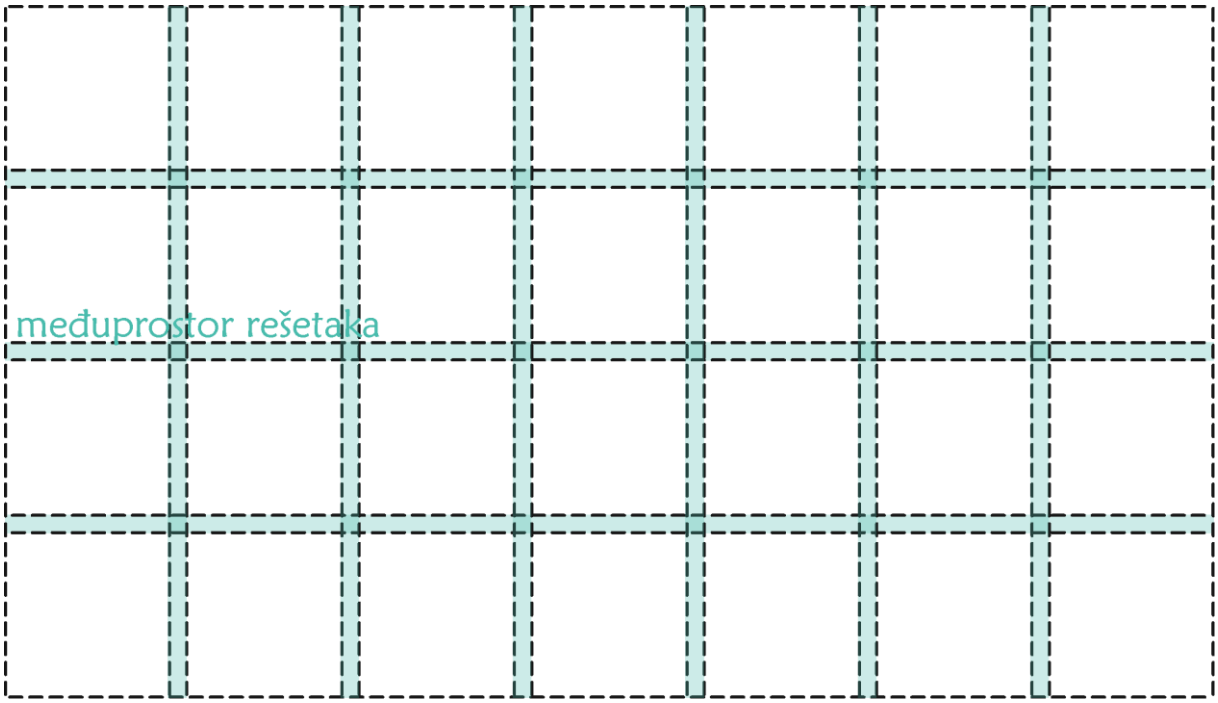
[48] Usporediti s https://youtu.be/ojKbYz0iKQE?t=1m27s
[49] Usp. Gasston, Peter. Nav. dj. Str. 210.
[50] Usp. CSS Grid Layout Module Level 1. Nav dj.

Slika 9. Područje rešetaka[51]

3.1.7. Međuprostor rešetaka

Međuprostor rešetaka je prazan prostor između traka rešetaka (Slika 9).[52] Stvoren je tako da širi liniju rešetke između traka rešetaka, a podsjeća na svojstvo *border-spacing* u kontekstu tablice, zato što stvara prostor između ćelija rešetki, kao i zbog toga što se može postaviti samo jedna vrijednost razmaka za svaku os, a to čini kroz svojstva *grid-row-gap* i *grid-column-gap*.[53]

[51] Usporediti s https://youtu.be/ojKbYz0iKQE?t=1m20s
[52] Usp. Rand-Hendriksen, Morten. Nav. dj.
[53] Usp. Mayer, Eric. Nav. dj. Str. 57.



Slika 10. Međuprostor rešetaka[54]

3.2. Definiranje rešetaka stilskog jezika

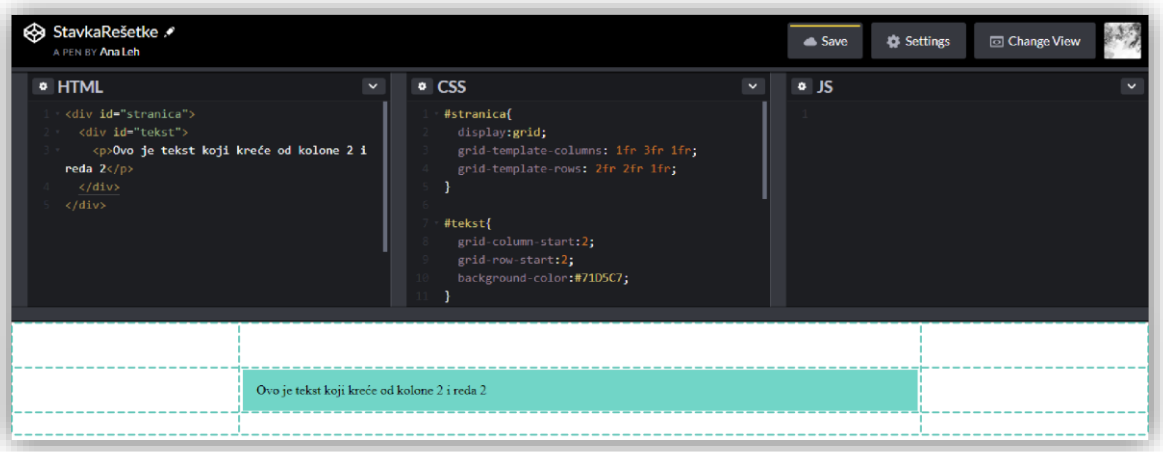
Prvi korak u stvaranju rasporeda elemenata implementacijom rešetaka je deklariranje prostora rešetke. Deklarira se svojstvom *display: grid;*. To svojstvo stvoriti će prostor rešetaka na razini blokova. Sljedeći korak je definirati trake rešetaka, odnosno retke i stupce. Trake se mogu definirati s preciznim brojem stupaca i redaka (eksplicitan način) ili se može dopustiti da se trake stvaraju u odnosu na njegov sadržaj (implicitan način). Moguće je i kombiniranje ta dva načina.[55]

3.2.1. Eksplicitni raspored elemenata implementacijom rešetaka

U eksplicitnom se rasporedu može definirati specifični broj traka rešetaka tako da se odredi njihova veličina koristeći svojstva *grid-template-columns* i *grid-template-rows*. U tim svojstvima može se postaviti više vrijednosti koje su prostorno razdvojene te svaka može imati svoju, jedinstvenu veličinu. Na primjer, *grid-template-columns: 20% 60% 20%;* određuje tri kolone koje ispunjavaju

[54] Usporediti s https://youtu.be/ojKbYz0iKQE?t=1m33s
[55] Usp. Gasston, Peter. Nav. dj. Str. 211.

prostor na mrežnoj stranici i to tako da su podijeljene u postotcima koji su ranije navedeni. Osim postotaka, kao vrijednosti možemo koristiti i druge mjerne jedinice kao što su pikseli (px), no preporuka za rešetke je nova mjerna jedinica nazvana frakcija[56] (fr), koja će detaljnije biti opisana u zasebnom potpoglavlju. Sve unutar prostora rešetke postaje stavka te rešetke. Postavljaju li se stavke u eksplicitnom rasporedu elemenata, mora se točno odrediti gdje će stavka započeti, a gdje završiti. Svojstva kojima se to određuje su *grid-column-start* i *grid-row-start*. Primjerice, stavi li se vrijednost 2 kod oba svojstva, stavka će započeti na koloni broj 2 te na redu broj 2 (Slika 10). Početna, odnosno zadana vrijednost je uvijek 1, tako da će u slučaju neodređenih vrijednosti za ta dva svojstva, stavka započeti na redu broj 1 te na koloni broj 1.



Slika 11. Primjer stavke rešetke u eksplicitnom rasporedu elemenata implementacijom rešetaka

Osim svojstava kojim se određuje početak neke stavke, postoje svojstva s kojima se određuje kraj kolone ili reda. Svojstva su *grid-row-end* te *grid-column-end*. Ne odrede li se ta svojstva, stavka ostaje u jednoj ćeliji u kojoj je i postavljen njen početak, u slučaju da nema sadržaja koji se „prelijeva“ preko nje.[57]

3.2.2. Frakcijska jedinica[58]

Kao što je navedeno u prijašnjem potpoglavlju, s dolaskom rasporeda elemenata implementacijom rešetaka, omogućena je i nova mjerna jedinica frakcija (fr). Frakcija se pojavljuje onda kada se

[56] eng. fraction
[57] Usp. Isto. Str. 212-214.
[58] eng. fractional unit

želi razdvojiti kompletni dostupni prostor. Najjednostavniji primjer bio bi razdvojiti prostor rešetke u 4 jednake frakcije, što će izgledati ovako: *grid-template-columns: 1fr 1fr 1fr 1fr;*. U tom slučaju, primjer je isti kao da smo napisali *grid-template-columns: 25% 25% 25% 25%;*. Želi li razvojni programer dodati petu kolonu, u slučaju postotaka mora ponovno računati koliko je potrebno da sve kolone budu iste širine, dok kod frakcijske jedinice može samo dodati peti *1fr*.[59] Još jedna prednost koju frakcijska jedinica ima naspram postotak je ta što se frakcijska jedinica može miješati s mjernim jedinicama kao što su *px* i *em*. Primjerice, želi li razvojni programer mrežnu stranicu s jednim stupcem širine *15em* te nakon njega dva stupca jednake veličine, u svojstvo *grid-template-columns* upisati će vrijednost *15 em 1fr 1fr*. S postotcima to nije moguće, osim ako razvojni programer ne zna točnu širinu prostora rešetke i spreman je dodatno računati.[60]

3.2.3. Imenovanje područja rešetaka

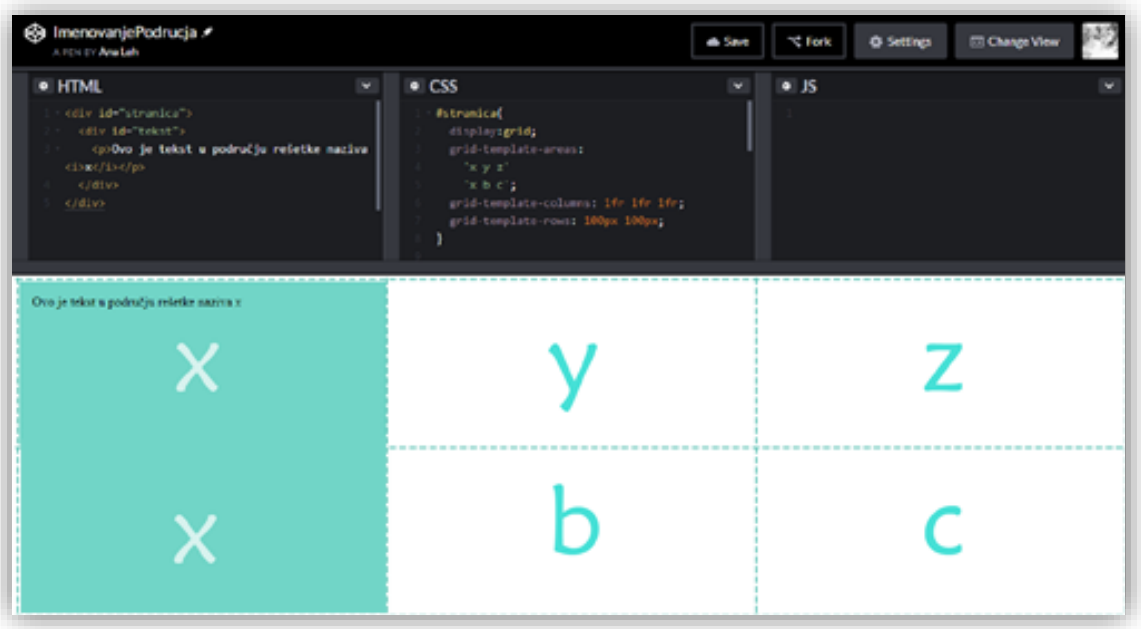
Osim što se stavka rešetke može postaviti s obzirom na brojeve koji se postave, stavka se može postaviti i u unaprijed nazvano područje sa svojstvom *grid-template-areas*. S tim svojstvom području se može dati jedinstveno ime, koje god razvojni programer želi.[61]

Slika 12. Područja rešetaka na mrežnoj stranici sa zadanim imenima

Primjerice, postave li se tri jednake kolone, razvojni programer može napisati *grid-template-areas: 'x y z';*, što znači da će ime prve kolone s lijeva biti *x*, ime kolone u sredini *y* te ime zadnje kolone *z* (Slika 11). Kako bi se stavka rešetke postavila unutar unaprijed nazvanog područja, koristi se svojstvo *grid-area*. Tako će razvojni programer koji želi stavku staviti u posljednju kolonu s lijeva, svojstvu dati vrijednost *z*, kako je područje nazvano. Svaki niz vrijednosti u svojstvu *grid-template-areas* predstavlja jedan redak u prostoru rešetke. Želi li razvojni programer novi redak,

[59] Usp. Mayer, Eric Nav. dj. Str. 13.
[60] Usp. Gasston, Nav. dj. Str. 213.
[61] Usp. Isto. Str. 217.

samo doda novi niz, što izgleda ovako: *grid-template-areas: ' x y z' 'x b c';*. U tom primjeru (Slika 12.) se može vidjeti kako je prva kolona drugog reda također nazvana *x*, što znači da će se područje rešetke proširiti i ići će od prvog do drugog reda prve kolone.[62]



Slika 13. Primjer stavke rešetke postavljene u određeno područje rešetke

3.2.4. Stenografska svojstva položaja rešetke i predloška rešetke

Pisanje četiriju pojedinačnih svojstava za postavljanje elemenata u raspored elemenata implementacijom rešetaka čini se bespotrebno, pogotovo ako postoji jednostavniji i kraći način. Svojstva koja daju rješenje su *grid-column* i *grid-row*, a svaka od njih je skraćenica (duža verzija sadrži *-start* ili *-end* nakon spomenutih svojstava). Tako će, primjerice, umjesto *grid-column-start:2; grid-column-end:3;* pisati *grid-column:2/3;*. Također je moguće i kombiniranje sva četiri svojstva u jednom stenografskom svojstvu, *grid-area* koja obuhvaća sva prijašnja svojstva. Osnovna sintaksa je *grid-area: row-start / column-start / row-end / column-end;*, odnosno primjer *grid-area: 1 / 2 / 3 / 3;*.[63]

Osim stenografskih svojstava kod položaja rešetaka, samo postavljanje predloška rešetaka također se može postaviti s jednim svojstvom, umjesto s tri. Tako umjesto *grid-template-columns*, *grid-template-rows* i *grid-template-areas* razvojni programer može upotrijebiti samo *grid-*

[62] Usp. Isto. Str. 218.
[63] Usp. Isto. Str. 216.

template. Predložak glasi: *grid-template: grid-template-columns / grid-template-rows;*, a primjerak je sljedeći: *grid-template: 1fr 1fr 1fr / 'x y z' 80px 'x b c';* (umjesto *grid-template-areas: 'x y z' 'x b c'; grid-template-columns: 1fr 1fr 1fr; grid-template-rows: 80px auto;*)[64]

3.2.5. Implicitni raspored elemenata implementacijom rešetaka

Za razliku od eksplicitnih rešetaka, koje su određene specifičnom duljinom vrijednosti, implicitne rešetke određene su njihovim sadržajem.[65] Kod eksplicitnih rešetaka koriste se navedena svojstva *grid-template-rows, grid-template-columns* i *grid-template-areas*. Kod implicitnih rešetaka koriste se druga svojstva, a nazivaju se: *grid-auto-rows* i *grid-auto-columns*. Također je prisutno i svojstvo *grid-auto-flow* koje kontrolira samostalno postavljanje stavki rešetke.[66] Implicitne rešetke koriste se kada razvojnom programeru nije važno koliko redova ili stupaca ima, već da svaka stavka ima svoje mjesto u rešetki. Nakon što se odrede spomenuta *auto* svojstva, rešetka će se automatski prilagoditi veličini koja odgovara stavci. U slučaju da stavka nema određeno mjesto u prostoru rešetke, za nju vrijedi početna vrijednost svojstva *grid-column* i *grid-row*, a to je vrijednost 1. To početno ponašanje može se promijeniti sa spomenutim *grid-auto-flow* svojstvom, kojemu je početna vrijednost *row* i koju razvojni programer mora promijeniti u *column* kako bi automatski dodavala nove stupce, a broj stupaca ovisiti će o broju stavki koje su prisutne u rešetki.[67]

3.2.6. Ponavljanje linija rešetaka

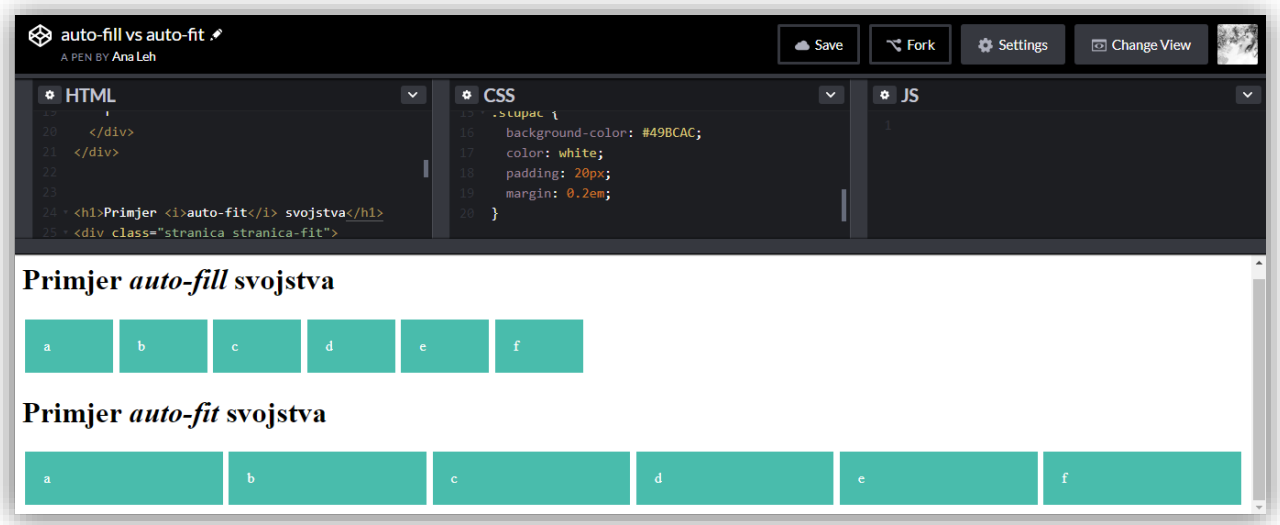
Ako se razvojni programer nađe u situaciji da želi postaviti 10 jednakih stupaca, a ne želi pisati 10 puta iste vrijednosti, koristiti će *repeat()*. U toj funkciji razvojni programer postaviti će koliko puta želi da se stupac (ili red) ponovi te koja je veličina svakoga od njih. Primjer izgleda ovako: *grid-template-columns: repeat(10, 1fr);*, što znači da je postavljeno 10 stupaca, svaki od njih je širok 1 frakciju, odnosno dostupan prostor podijeljen je na 10 jednakih dijelova. U isto svojstvo također se može dodati i međuprostor rešetaka, te umjesto posebnog dodavanja *grid-gap* svojstva, razvojni

[64] Usp. Isto. Str. 218-219.
[65] Usp. Isto. Str. 219.
[66] Usp. CSS Grid Layout Module Level 1. Nav. dj.
[67] Usp. Gasston, Peter. Nav dj. Str. 220.

programer napisati će: *grid-template-columns: repeat(10, 1em 1fr);*, što znači da će se prije svakog stupca dodati i međuprostor rešetke veličine 1em.[68]

3.2.7. Automatsko ponavljanje i pristajanje

Postavi li razvojni programer vrijednost *column* u svojstvo *grid-auto-flow*, u slučaju većeg broja stavki rešetke, uzrokovati će „prelijevanje“ u stupcu. Stavke neće otići u novi red ako je širina prozora preuska jer je razvojni programer rekao izričito rekao pregledniku da ponovi stupce određen broj puta. Kako bi postigao omotavanje te spriječio „prelijevanje“, razvojni će programer u funkciji repeat() koristiti ključne riječi *auto-fit* i *auto-fill*.[69] Te ključne riječi reći će pregledniku da se pobrine za veličinu stupaca i omatanje elemenata, što će uzrokovati uklapanje elemenata u redove kada širina nije dovoljno velika kako bi sve stavke stale u stupce bez prelijevanja. Razlika između njih je jednostavna, iako možda malo zbunjujuća dok se ne dobije vizualni prikaz istih (Slika 13). *Auto-fill* ispunjava red sa što više stupaca i stvara implicitne stupce svaki put kada se stupac može uklopiti. *Auto-fit* se, s druge strane, prilagođava trenutno dostupnim stupcima tako da ih širi kako bi zauzeli što više od dostupnog prostora.[70]



Slika 14. Vizualni prikaz svojstva *auto-fill* i *auto-fit*

[68] Usp. Isto.
[69] Usp. Soueidan, Sara. Auto-Sizing Columns in CSS Grid: 'auto-fill' vs 'auto-fit', 2017. URL: https://css-tricks.com/auto-sizing-columns-css-grid-auto-fill-vs-auto-fit/ (2018-07-27)
[70] Usp. Isto.

3.2.8. Funkcija minmax()

Minmax() je CSS funkcija koja definira raspon veličine veće ili jednako *min*-u i manji ili jednak *max*-u. Sadrži dva spomenuta parametra, *min* i *max*. Može biti iskazana u dužini, u postotku ili s frakcijskom jedinicom te u vrijednostima *max-content, min-content* ili *auto*.[71] Može se koristiti u svim svojstvima s kojima se određuje veličina unutar prostora rešetke. Primjer je: *grid-template-rows: minmax(250px,3fr);*.

3.3. Upiti svojstava[72]

Najnovije verzije većine preglednika omogućuju implementaciju rešetaka. No ostaje pitanje što s onim korisnicima koji koriste starije preglednike, kako oni mogu koristiti mrežnu stranicu s rešetkama? Odgovor leži u upitima svojstava koji su dio CSS modula uvjetnih pravila[73], a isti sadrže i medijske upite. I jedan i drugi alat funkcioniraju slično, a umjesto pravila *@media* što se koristi kod medijskih upita, u CSS kodu napisati će se @supports pravilo za upite svojstava.[74] Primjer koda je sljedeći: *@supports (display: grid) { // kod koji će se prikazati samo ako su rešetke podržane u pregledniku (komentar) }*. U slučaju da preglednik razumije *display:grid;* svojstvo, primijeniti će sve što je unutar zagrada zadano. U protivnom će preskočiti sve specifikacije.[75] To znači da će razvojni programer prvo napisati stari CSS, a potom nakon njega novi unutar upita svojstava. I Jen Simmons i Rachel Andrew govore kako nastaje zbunjenost kod shvaćanja što su točno upiti svojstava. Oni ne služe kao dodatna provjera je li preglednik ispravno implementirao CSS svojstvo. „To nije čarobni štapić za uklanjanje pogrešaka preglednika.“[76] Upiti svojstava dostupni su u zadnjih desetak verzija Firefox-a, Chrome-a i Opere, no nisu podržani niti u jednoj verziji Internet Explorer-a, Opere Mini, Blackberry preglednika te UC preglednika.[77]

[71] Usp. Minmax(): URL: https://developer.mozilla.org/en-US/docs/Web/CSS/minmax (2018-07-27)
[72] eng. feature query
[73] eng. CSS Conditional Rules Module
[74] Usp. Andrew, Rachel. The New CSS Layout. New York: A Book Apart, 2017. Str. 102.
[75] Usp. Simmons, Jen. Using Feature Queries in CSS, 2016. URL: https://hacks.mozilla.org/2016/08/using-feature-queries-in-css/ (2018-07-29)
[76] Isto.
[77] Usp. Isto.

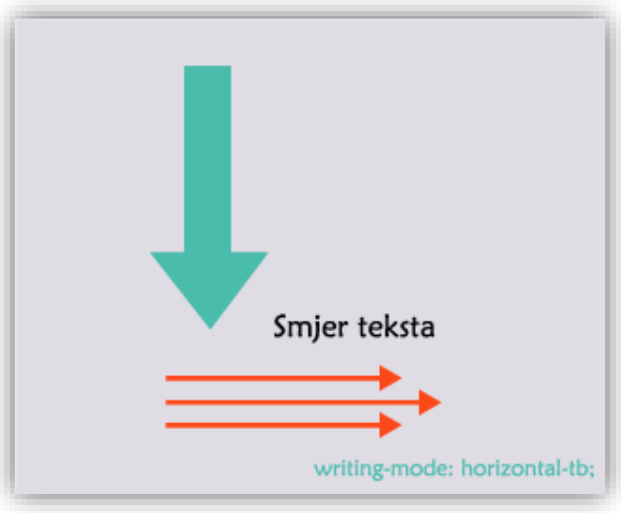
4. Načini pisanja[78]

Za sve korisnike koji cijeli život pišu latiničnim pismom sve mrežne stranice izgledale su uvijek isto, odnosno redoslijed rasporeda elemenata na stranici je uvijek isti: horizontalni tekst koji ide s lijeva na desni. Takav način pisanja također je i najčešći u svijetu, točnije pokriva 70% stanovništva.[79] Naravno, to ne znači da je jedini, niti da se oni koji poznaju drugi način pisanja i cijeli život se koriste njime moraju prilagoditi najčešće korištenom načinu pisanja. Svijet je velik i raznolik, a ništa drugačije nije ni s Internetom. Kako bi se redoslijed rasporeda elemenata mogao što jednostavnije odrediti, CSS je osigurao svojstvo *writing*-mode koji pripada CSS načinu pisanja. CSS način pisanja je CSS modul koji definira različite međunarodne načine pisanja.[80] Svojstvo *writing-mode* sadrži pet mogućih vrijednosti: *horizontal-tb, vertical-rl, vertical-lr, sideways-rl, sideways-lr*. Iako se vrijednosti mogu koristiti za razne stvari, ipak su namijenjene za četiri glavna sistema pisanja: latinski, arapski, hanski i mongolski.[81] Razlog zbog kojeg je način pisanja bitan za raspored elemenata implementacijom rešetaka (i obrnuto) jest što su rešetke omogućile svojstva koja više nisu ograničena na lijevo i desno, već na početak i kraj (dodatno pojašnjeno u potpoglavljima). Jenn Simmons također navodi kako joj je bolje razumijevanje načina pisanja uvelike pomoglo bolje razumjeti raspored elemenata implementacijom rešetaka, kao i fleksibilni raspored elemenata.[82] Zbog toga slijedi detaljniji opis svakog od sistema pisanja u idućim potpoglavljima.

4.1. Latinski sustav

Latinski sustav dobio je ime po svim jezicima koji koriste latinicu, uključujući engleski, španjolski, njemački i slično. No, i razna druga pisma koriste latinski sustav. Razvojni programeri koji stvaraju mrežnu stranicu s ovim sustavom ne moraju postaviti nikakvo svojstvo, niti vrijednost, jer je taj sustav već zadan kao početna vrijednost, a postavljen je tako da ide horizontalno s vrha prema dnu te s lijeva na desno (Slika 16). No, ako se već određuje, to se najčešće radi u početnom *<html>* elementu tako da se doda svojstvo *dir="ltr"*.[83]

[78] eng. Writing Modes
[79] Usp. Simmons, Jen. CSS Writing Modes, 2016. URL: https://24ways.org/2016/css-writing-modes/ (2018-07-30)
[80] Usp. Writing Modes. URL: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Writing_Modes (2018-07-30)
[81] Usp. Simmons, Jen. Nav. dj.
[82] Usp. Isto.
[83] Usp. Isto.

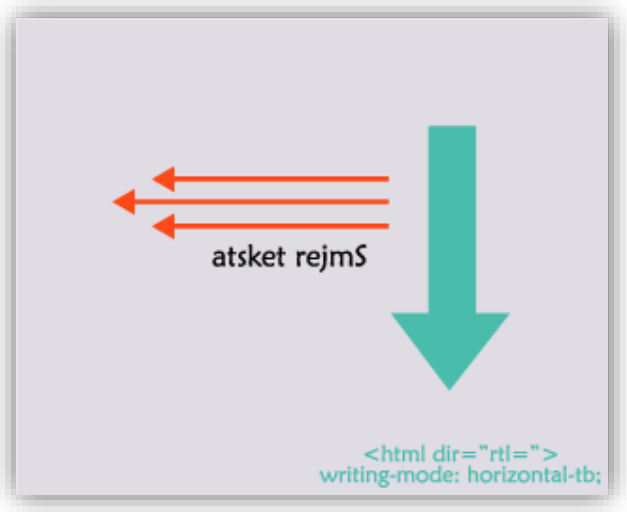


Slika 15. Prikaz načina pisanja latinskim sustavom[84]

4.2. Arapski sustav

Arapski, hebrejski i nekoliko drugih jezika koriste redoslijed s desna na lijevo. Redoslijed je, pak, i dalje horizontalan. Mrežne stranice ne raspoređuju samo tekst s desna na lijevo, cijeli raspored elemenata ide s desno na lijevo (Slika 17). Ovdje novi rasporedi elemenata (fleksibilni i rešetke) pokazuju još prednosti, jer u svojim svojstvima umjesto dosadašnjih vrijednosti *left* i *right* koriste *start* i *end*, što znači da nije određeno s koje će strane biti početak, a s koje kraj. U konačnici, redoslijed se postavlja isto kao i kod latinskog sustava, u <*html*> elementu postavlja se svojstvo *dir*, no umjesto *ltr* vrijednosti koja se postavlja kod latinskog sustava, ovdje se postavlja *rtl* vrijednost.[85]

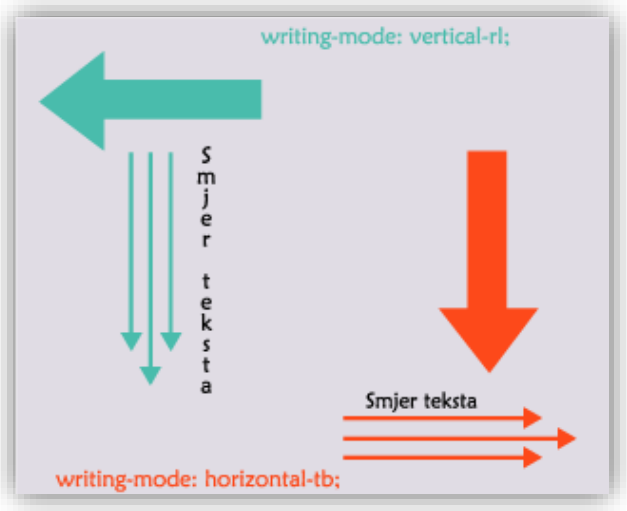
[84] Usporediti s https://24ways.org/2016/css-writing-modes/
[85] Usp. Isto.



Slika 16. Prikaz načina pisanja arapskim sustavom[86]

4.3. Hanski sustav

Hanskom sustavu pripadaju kineski, japanski, korejski i slični jezici. Stranice s hanskim sustavom znaju biti u vertikalnom načinu pisanja, no i u horizontalnom (Slika 18). Nisu rijetka niti kombiniranja ta dva načina, zbog čega se preporuča da razvojni programer za stranicu ostavi zadanu horizontalnu vrijednost, te da vertikalne elemente dodatno specificira (primjer: *div.tekstclanka{ writing-mode: vertical-rl;}*.[87]

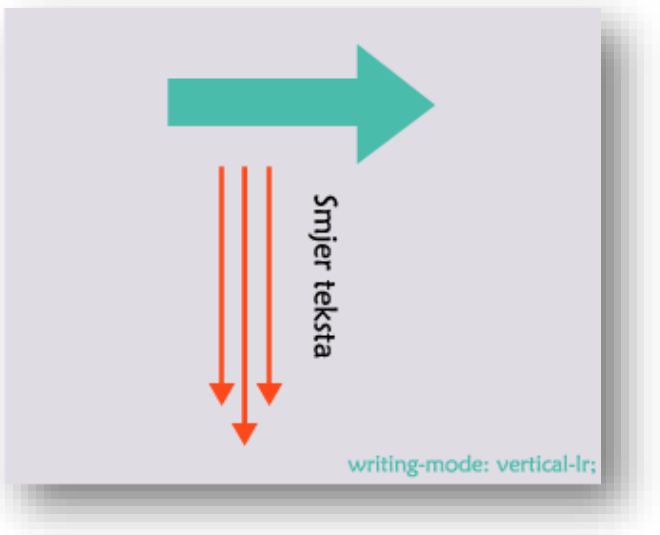


Slika 17. Prikaz načina pisanja hanskim sustavom[88]

[86] Usporediti s https://24ways.org/2016/css-writing-modes/
[87] Usp. Isto.
[88] Usporediti s https://24ways.org/2016/css-writing-modes/

4.4. Mongolski sustav

Mongolski je vertikalni jezik, a elementi su postavljeni s lijeva na desno. Iako se na prvu sve čini jednostavno i jasno, znakovi su izokrenuti naopačke i vrh mongolskih znakova ne ukazuje na lijevo prema početnom rubu smjera blokova, već desno (Sika 19). Način na koji mongolski jezik funkcionira definirao je rezultat vrijednosti *vertical-lr* u svojstvu *writing-mode*.[89]

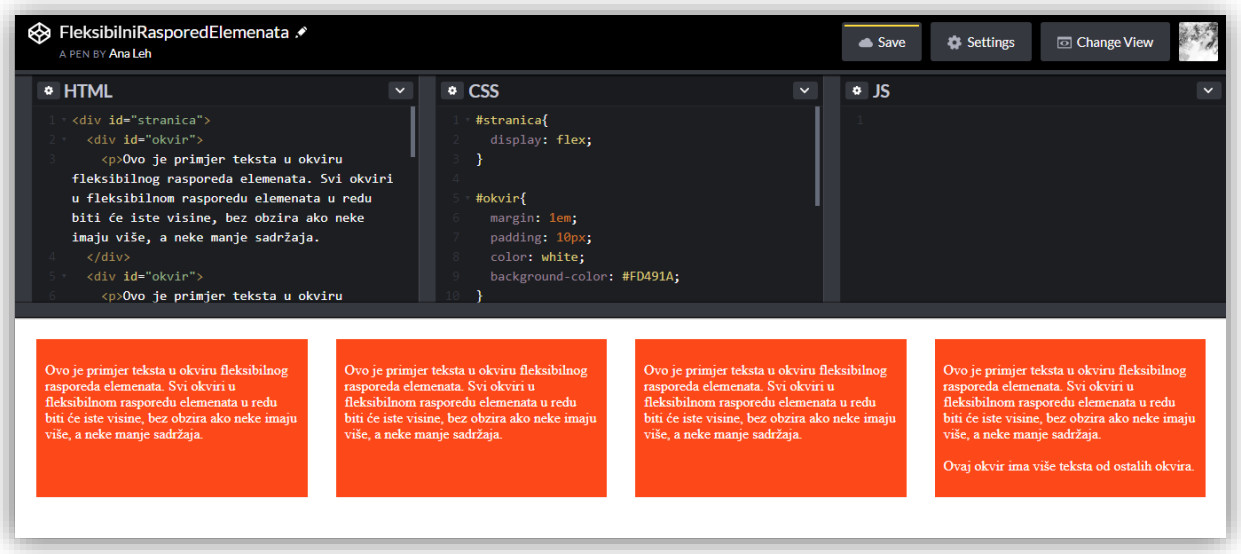


Slika 18. Prikaz načina pisanja mongolskim sustavom[90]

[89] Usp. Isto.
[90] Usporediti s https://24ways.org/2016/css-writing-modes/

5. Fleksibilni raspored elemenata i raspored elemenata implementacijom rešetaka

Ranije u radu spomenuto je kako se raspored elemenata stvarno počeo mijenjati s dolaskom fleksibilnog rasporeda elemenata, no za razliku od rasporeda elemenata implementacijom rešetaka, nisu prikazana njegova svojstva niti praktični primjeri. U ovom poglavlju biti će navedene razlike između oba rasporeda elemenata. Umjesto svojstva *display* kod kojeg je vrijednost kod rešetaka bila *grid*, kod fleksibilnog rasporeda elemenata vrijednost će biti *flex*. S time će stavke postati fleksibilne, mijenjajući visinu s obzirom na fleksibilni prostor. Svi okviri[91] u fleksibilnom rasporedu elemenata u redu biti će iste visine, bez obzira ako neke imaju više, a neke manje sadržaja (Slika 14).[92]

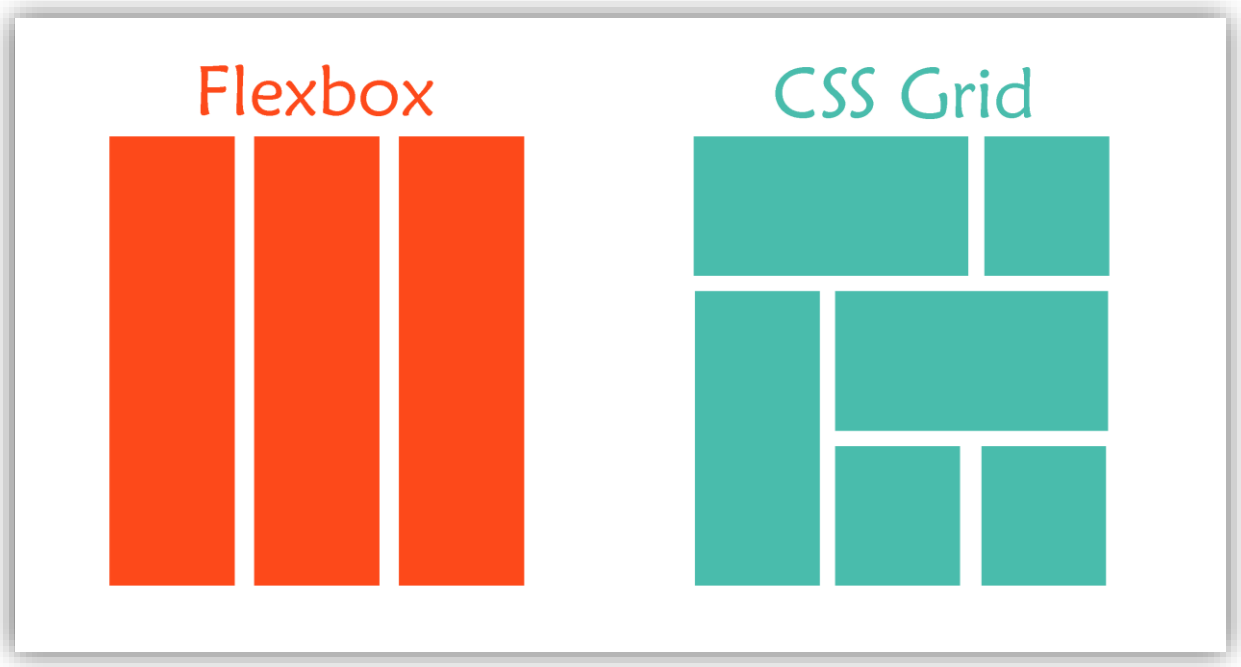


Slika 19. Primjer fleksibilnog rasporeda elemenata

Ako se nastave dodavati stavke i okviri, oni će se nastaviti nizati u istom redu. U konačnici red će se „preliti“. Kod takve situacije, dodaje se još jedno svojstvo: *flex-wrap: wrap;*, a također se mora dodati i novo svojstvo u svakom okviru (primjer *flex: 1 1 200px;*). Kod primjera šest stavki u fleksibilnom prostoru, to će rezultirati izgledom rešetke, po tri stupca u dva reda. No, ako se ukloni samo jedna stavka, donji red će se podijeliti na dva jednaka dijela gdje će biti smještene dvije posljednje stavke. S time, drugi red postati će novi fleksibilni prostor, kompletno se

[91] eng. boxes
[92] Usp. Andrew, Rachel. Nav. dj. Str. 34-37.

odvajajući od prvog reda. Neće pokušati smjestiti stavke u gornji ili donji red. Takav način rasporeda elemenata naziva se jednodimenzionalni raspored elemenata. Fleksibilni raspored elemenata može kontrolirati stavke ili u stupcu ili u redu, ne mogu se kontrolirati i jedno i drugo odjednom. [93] Postoje dodatni trikovi koji fleksibilni raspored elemenata čine više kao raspored elemenata implementacijom rešetaka, a oni uključuju svojstva kao što su *flex-grow*, *flex-shrink* i *flex-basis*. Oni mogu dati puno više kontrole kako se fleksibilne stavke ponašaju, no ako razvojni programer želi rešetke, bolje rješenje bi bilo koristiti svojstva rešetaka. Za razliku od fleksibilnog rasporeda elemenata, raspored elemenata implementacijom rešetaka čini točno ono po čemu je dobio i ime, stvara od mrežne stranice izgled rešetaka i što ga čini dvodimenzionalnim rasporedom elemenata (Slika 15).



Slika 20. Vizualni prikaz razlike fleksibilnog rasporeda elemenata i rasporeda elemenata implementacijom rešetaka[94]

Za razliku od fleksibilnog rasporeda elemenata, kod rasporeda elemenata implementacijom rešetaka nije potrebno dodati ikakva svojstva svojstvima koji se nalaze u prostoru rešetke, već će ćelije odmah poprimiti svojstva rešetke, čim u prostoru rešetke stoji svojstvo *display: grid;*. Kod fleksibilnog rasporeda elemenata, kako bi se stvorio međuprostor između stavki, potrebno je koristiti svojstvo *margin*, dok je kod rešetaka potrebno svojstvo *grid-gap*, stvoreno isključivo za

[93] Usp. Isto.
[94] Usporediti s http://maddesigns.de/css-grid-layout-2764.html

međuprostor između svojstava. Još jedna razlika između spomenutih rasporeda elemenata je ta što kod fleksibilnog rasporeda elemenata svojstva idu redom, dok kod rešetaka razvojni programer svojstva može postavljati redoslijedom kojim želi, bez obzira kojim redom su svojstva napisana u kodu, što je već navedeno i opisano ranije u radu.

6. Alati pristupnog dijela[95] i raspored elemenata implementacijom rešetaka

Razvoj pristupnog, odnosno prednjeg dijela ne znači samo raspored elemenata. Razvoj pristupnog dijela danas izgleda kompleksno, a razvojni programeri imaju nekoliko načina kako napraviti istu stvar na mrežnoj stranici. Cilj rasporeda elemenata implementacijom rešetaka nije poslati poruku kako je samo korištenje njega dovoljno za stvaranje mrežne stranice, već se preporuča kombiniranje različitih metoda kako bi se dobila mrežna stranica koja ispunjava sve kriterije IT[96] svijeta, o čemu će detaljnije biti napisano u idućim poglavljima. S obzirom na navedeno, daje se zaključiti kako se novi raspored elemenata zapravo uklapa u okruženje kakvo već postoji te da njegov dolazak ne znači novi početak pristupnog dijela, već nadopuna pristupnog dijela. Kao da autor piše novo, bitno poglavlje u knjizi, umjesto nove knjige. Alata koji se danas koriste kako bi se olakšao razvoj pristupnog dijela doista je mnogo te će u ovom poglavlju biti navedena samo neka od njih, ona koja su široko rasprostranjena u IT svijetu, kao i ona koja se već nekoliko mjeseci koriste zajedno s rešetkama.

6.1. Okvirni sustavi[97]

Danas se vode rasprave o činjeničnom stanju gdje sve mrežne stranice izgledaju isto ili slično. Razlog stoji u okvirnim sustavima, Bootstrap-u i Foundation-u kao vodećim sustavima, koji omogućavaju razvojnim programerima da stvaraju mrežne stranice s minimalnim znanjem pristupnog dijela.[98] Okvirni sustavi danas su izrazito popularni, a iako mrežne stranice izrađene uz njihovu pomoć možda ne izgledaju jedinstveno, postoji veća šansa da će izgledati ljepše, barem ako ju izrađuje netko tko nije dizajner mrežnih stranica (primjerice programeri). I Bootstrap[99] i Foundation[100] u svojim najnovijim verzijama već su uključili rešetke, a u slučaju da se rešetke koriste, Bootstrap je osim ograničenja od 12 stupaca, dodao i mogućnost jednostavnog raspoređivanja stupaca koje je razvojni programer stavio u jedan red.

Mnogi razvojni programeri, s druge strane, smatraju kako se okvirni sustavi uopće ne bi trebali koristiti, a istog su mišljenja i oni koji su radili na razvoju rešetaka. Okvirni sustavi nude korisnicima veliki broj rješenja te pojednostavljuju stvaranje pristupnog dijela mrežne stranice

[95] eng. front-end development
[96] informacijske tehnologije
[97] eng. framework
[98] Usp. Andrew, Rachel. Nav. dj. Str. 14.
[99] Usp. Grid System. URL: https://getbootstrap.com/docs/4.0/layout/grid/ (2018-07-28)
[100] Usp. The Grid. URL: https://foundation.zurb.com/grid.html (2018-07-28)

toliko da se korisnici nekad ne mogu sjetiti jednostavnih CSS svojstava jer ih nisu imali potrebu koristiti duže vrijeme zbog implementacije okvirnih sustava. Raspored elemenata implementacijom rešetaka ima nekoliko prednosti naspram okvirnog sustava kao što je Bootstrap, a jedan od najvažnijih je taj što rešetke omogućuju HTML da postane ono što je otpočetka trebao biti: označavanje sadržaja. Sve ostalo što se tiče izgleda pripada CSS-u. Tako da se unaprijed postavljene klase koje nude okvirni sustavi (kao što su *col-xs-4*, *col-lg-2* i slično), koje rezultiraju time da isti element ima više klasa ili identifikatora u rešetkama ne pojavljuju, već je postavljena jedna klasa ili jedan identifikator gdje se sve njegove karakteristike potom uređuju u CSS-u.[101] Postoji još nekoliko prednosti rešetaka naspram okvirnih sustava, no pitanje je vrijede li oni i danas s obzirom da su okvirni sustavi uključili rešetke u svoje najnovije verzije. Iako mnogi razvojni programeri savjetuju da se okvirni sustavi ne koriste, konstantno napredovanje sustava i pružanje mogućnosti korištenja najnovijih tehnologija objašnjava zašto su i dalje toliko popularni. No, žele li razvojni programeri napredovati, pisati će sami svoj kompletan CSS kod, a vjeruju li u raspored elemenata implementacijom rešetaka, smatrati će da su rešetke okvirni sustav sam za sebe i da niti jedan drugi nije potreban, barem kada je stvaranje rasporeda elemenata u pitanju.

6.2. Pretprocesori

„CSS pretprocesor je program koji vam omogućuje stvaranje CSS-a iz vlastite jedinstvene sintakse pretprocesora.“[102] Kao i okvirni sustavi, pretprocesori danas imaju važno mjesto u razvojnom svijetu mrežnih stranica te svakom godinom raste broj ljudi koji se bave CSS-om, a zapravo ga ne pišu.[103] Alati poput pretprocesora štede puno vremena, a jedan od uobičajenih načina korištenja pretprocesora je razdvajanje stilskog jezika u više manjih datoteka, tako da osobe odgovorne za različite dijelove aplikacije mogu raditi na pojedinačnim datotekama koje se zatim sastavljaju. Pretprocesori također omogućavaju postavljanje varijabli za zajedničke boje, fontove ili veličinu.[104] Vjerojatno najpoznatiji pretprocesor zove se SASS[105], a prate ga LESS[106] i Stylus. Sa SASS-om je moguće stvaranje rasporeda elemenata, koristeći malo matematike, a svoja rješenja nude i ostali pretprocesori, no dolaskom rasporeda elemenata implementacijom rešetki za dodatne izračune i kompliciranja stvaranja rasporeda elemenata nema potrebe s obzirom da je moguće

[101] Usp. Harald Borgen, Per. Why CSS Grid is better than Bootstrap for creating layouts, 2017. URL: https://hackernoon.com/how-css-grid-beats-bootstrap-85d5881cf163 (2018-07-28)
[102] CSS preprocessor. URL: https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor (2018-07-29)
[103] Usp. Andrew, Rachel. Nav dj. Str. 13.
[104] Usp. Isto.
[105] Syntactically Awesome Style Sheets
[106] Leaner Style Sheets

kombiniranje rešetaka i pretprocesora u izradi mrežnih stranica. Stručnjaci jesu za korištenje pretprocesora, no neki od njih i dalje napominju da bi razvojni programeri u neki situacijama trebali „pogledati svijet izvan tih alata“, jer je moguće da razvojni programer ograniči sam sebe s odabranim pretprocesorom i time propustiti neke od novijih tehnologija koje nude nove, bolje mogućnosti.[107]

6.3. Postprocesori

Za razliku od pretprocesora, koji se koriste prije stvaranja samog CSS-a, postprocesori se koriste nakon što je CSS stvoren i spreman za korištenje. Svrha im je poboljšavanje CSS-a te mijenjanje onog koda za koji se razvojni programer ne želi brinuti, kao što je primjerice automatsko preuzimanje prefiksa, izrada zamjenskih piksela za svaku „rem“ jedinicu koja je korištena i slično.[108] Jedan od alata postprocesora je i PostCSS, kojemu je zadaća transformiranje stilova pomoću JavaScript dodataka[109]. Najveća snaga PostCSS-a leži u njegovoj modularnosti, koja zahtjeva da se u kod dodaju samo one značajke koje su zaista potrebne što rezultira jednostavnijim stilskim jezikom[110], što mu je u konačnici zajedničko s rasporedom elemenata implementacijom rešetaka (osim što rešetke omogućavaju jednostavniji označiteljski jezik, odnosno HTML). I ne samo da se postprocesori i rešetke mogu kombinirati, na Internetu već postoje dodatci (PostCSS-a)[111] koji sadrže specifikacije rešetki koji čine CSS kod još jednostavnijim i još boljim.

[107] Usp. Isto.
[108] Usp. Baumgartner, Stefan. Deconfusing Pre- and Post-processing, 2015. URL: https://medium.com/@ddprrt/deconfusing-pre-and-post-processing-d68e3bd078a3 (2018-07-29)
[109] eng. JavaScript plugins
[110] Usp. Isto.
[111] Usp. Pollet-Villard, Sylvain. CSS Grid Layout and PostCSS: Now Kiss!, 2017. URL: https://medium.com/@SylvainPV/css-grid-layout-and-postcss-now-kiss-5e35f61a6f00 (2018-07-29)

7. Budućnost rasporeda elemenata implementacijom rešetaka

Trenutno točan naziv rasporeda rešetaka implementacijom rešetaka je CSS raspored elemenata implementacijom rešetaka modul – Stupanj 1[112]. Trenutna verzija CSS-a je CSS3. Prve dvije verzije CSS-a bile su monolitne i svaka od njih imala je detaljno opisane specifikacije. Druga verzija CSS-a je također imala i dodatnu verziju naziva CSS2.1. CSS3 je imao sve dijelove CSS2.1 verzije, no podijelio ih je u module. Nakon toga, svaki modul mogao se uređivati zasebno. Zbog toga postoje moduli stupnja 3, kao što su selektori, *display* svojstvo i slično. Sve specifikacije koje su napredovale u stupnjevima postojale su u ranijim verzijama CSS-a, a neki su sazrjeli toliko da se trenutno nalaze na stupnju 4. S obzirom na to, logičan je zaključak zašto je raspored elemenata implementacijom rešetaka na stupnju 1 (kao i fleksibilni raspored elemenata). Rešetke nisu postojale u prijašnjim verzijama CSS-a. No, i rešetke i fleksibilni raspored elemenata do danas su uznapredovale toliko da se uskoro može očekivati i prijedlog za stupanj 2.[113]

Postoje specifikacije rešetaka koje su pozitivno ocijenjene no još nisu podržane u preglednicima. Možda i najočekivanija je specifikacija *subgrid*. Problem koji sada postoji u rešetkama je taj što će stavka postati dio rasporeda elemenata implementacijom rešetaka samo ako je direktan „potomak“ prostora rešetke. To znači da „potomak potomka“ neće biti prepoznat kao stavka rešetke. Razvojni programer može ga postaviti da bude prostor rešetke sam za sebe, no neće biti povezan s prijašnjim prostorom. *Subgrid* svojstvo nudi rješenje za opisani problem tako što će dopuštati „potomku potomka“ sudjelovanje u izvornom rasporedu elemenata. Svojstvo je prvotno bilo uključeno u Stupanj 1 rasporeda elemenata implementacijom rešetaka, no naknadno je premješten u Stupanj 2, što znači da će se za njegovu implementaciju morati još malo pričekati.[114] Uređivanje samih stavki ili područja rešetaka nije moguće, no postoji prijedlog da se isto omogući, kao i za stvaranje elemenata rešetaka koje sadrže neki drugi oblik osim pravokutnog.

Iako je prijašnje poglavlje usporedilo raspored elemenata implementacijom rešetaka i fleksibilni raspored elemenata, to ne znači da treba odabrati bolji i koristiti samo njega. I jedan i drugi raspored elemenata danas je dio CSS-a i razvojnog programera ništa ne sprječava da ih kombinira. I ne samo njih. CSS ne nudi jedno rješenje za savršenu mrežnu stranicu, već razne specifikacije između kojih će razvojni programeri moći birati najbolja rješenja za svoju mrežnu stranicu. S tom formulom dolazi se do jedinstvenih mrežnih stranica, umjesto da sve izgledaju

[112] eng. CSS Grid Layout Module Level 1
[113] Usp. Andrew, Rachel. Nav. dj. Str. 113-114.
[114] Usp. Isto. Str. 117.

slično ili jednako. Rešetke, fleksibilni raspored elemenata, načini pisanja, poravnavanje, prilagođavanje objekta[115], posebne mjerne jedinice za prikaz[116], *flow* svojstva, više stupaca[117],dimenzioniranje[118], sve su to alati koji mogu pomoći izraditi mrežnu stranicu. Kao što automehaničar ima cijelu kutiju s alatima i razne dijelove automobila popravlja s raznim alatima u kutiji, umjesto jednog univerzalnog alata koji bi popravio sve dijelove. To niti je moguće, niti je potrebno.

[115] eng. object fit
[116] eng. viewport units
[117] eng. multicolumn
[118] eng. sizing

8. Zaključak

Dana 6. kolovoza 1991. prva mrežna stranica postala je dostupna i vidljiva svima. 27 godina je, većina povjesničara bi se složila, poprilično nova povijest. No, mrežna stranica u tih 27 godina evoluirala je više nego jednom. Koliko novih tehnologija, koliko novih načina i specifikacija za mrežne stranice je bilo predstavljeno IT svijetu, vjerojatno nije moguće ni nabrojati. No, budućnost će pamtiti samo one najvažnije, one koje su imale utjecaja na mrežnu stranicu, i najvažnije, one koje su opstale. Hoće li raspored elemenata implementacijom rešetaka biti jedna od njih? Mnogi stručnjaci stilskog jezika misle pozitivno. U konačnici se dolazi do zaključka da su se baš rešetke i njihov raspored elemenata čekale sve ove godine i da upravo ono što one nude rješava višegodišnji problem neispravnog rasporeda elemenata. Dakako, raspored elemenata nije magičan alat koji će riješiti sve probleme razvojnog programera tijekom izrade mrežne stranice, niti se očekuje da će s dva, tri reda koda odrediti sve što je potrebno. Rešetke zahtjevaju pažnju, zahtjevaju istraživanje i što je najbitnije, zahtjevaju praksu. Nije moguće provjeriti koliko razvojnih programera danas koriste rešetke, godinu dana nakon što je *display:grid;* omogućen u preglednicima i možda se prividno dobiva osjećaj da raspored elemenata implementacijom rešetaka nije još doista zaživio. No, 5 godina je bilo potrebno da se rešetke uopće omoguće za korištenje (od njihovog prvog prijedloga od strane Windowsa). Svakim danom sve je više članaka, sve više knjiga, sve više videozapisa koji opisuju na koji način raspored elemenata implementacijom rešetaka funkcionira te zašto je alat kojeg bi svaki razvojni programer trebao koristiti. Za vjerovati je da će to rezultirati i sve većim brojem razvojnih programera koji će ga koristiti. Možda ga je najbolje usporediti sa samom mrežnom stranicom. Koliko je ljudi pregledavalo mrežne stranice 1991. godine, a koliko ga koristi danas? Kako je izgledala prva mrežna stranica, a kako izgledaju mrežne stranice u 2018. godini? Ranije u radu navedeno je kako je raspored elemenata implementacijom rešetaka tek na prvom stupnju, dok su neki drugi moduli stilskog jezika već na četvrtoj razini. Moduli koji godinama napreduju. Isto čeka i rešetke. Prva verzija uvijek je najslabija verzija. Slijedi još puno mogućnosti, puno poboljšanja u modulu rešetki. Za prave stvari potrebno je vrijeme. Stručnjaci koji već više godina proučavaju rešetke i pomogli su ih implementrirati niti u jednom trenutku nisu rekli da su rešetke rješenje za sve probleme rasporeda elemenata, da je to vrhunac, da bolje ne može. Rekli su da su rešetke smjer u kojem se mrežna stranica na dalje treba kretati. Čekalo se 27 godina da se zagrebe površina ispravnog rasporeda elemenata. Ako je potrebno, pričekati će se još malo. I u konačnici, ne smije se zaboraviti da je raspored elemenata samo jedan mali dio stilskog jezika i načina na koji se dolazi do skladnih i oku ugodnih mrežnih stranica. Nema razloga odmicati se od svojstava koji godinama

dobro funkcioniraju i ne koristiti ih u svrhu zbog kojih su i nastali. Svijet je veliko i raznoliko mjesto, Internet također. Ništa ne sprječava stilski jezik mrežne stranice, kao niti njegov raspored elemenata implementacijom rešetaka da bude jedna karika te raznolikosti.

LITERATURA

1. Andrew, Rachel. The New CSS Layout. New York: A Book Apart, 2017.
2. Baumgartner, Stefan. Deconfusing Pre- and Post-processing, 2015. URL: https://medium.com/@ddprrt/deconfusing-pre-and-post-processing-d68e3bd078a3 (2018-07-29)
3. CSS Grid Layout Module Level 1, 2017. URL: https://www.w3.org/TR/css-grid-1/ (2018-07-11)
4. CSS preprocessor. URL: https://developer.mozilla.org/en-US/docs/Glossary/CSS_preprocessor (2018-07-29)
5. Flexbox History. URL: http://annairish.github.io/historicizing/history (2018-07-23)
6. Gasston, Peter. The Book of CSS3: A Developer's Guide to the Future of Web Design. San Francisco: No Starch Press, 2015.
7. Grid System. URL: https://getbootstrap.com/docs/4.0/layout/grid/ (2018-07-28)
8. Harald Borgen, Per. Why CSS Grid is better than Bootstrap for creating layouts, 2017. URL: https://hackernoon.com/how-css-grid-beats-bootstrap-85d5881cf163 (2018-07-28)
9. Mayer, Eric A. Grid Layout in CSS: Interface Layout for the Web. Beijing, Boston, Farnham, Sebastopol, Tokyo: O'Reilly, 2016.
10. Minmax(): URL: https://developer.mozilla.org/en-US/docs/Web/CSS/minmax (2018-07-27)
11. Pettit, Nick. Which Layout? Static, Liquid, Adaptive, or Responsive.URL: http://blog.teamtreehouse.com/which-page-layout (2018-07-20)
12. Pollet-Villard, Sylvain. CSS Grid Layout and PostCSS: Now Kiss!, 2017. URL: https://medium.com/@SylvainPV/css-grid-layout-and-postcss-now-kiss-5e35f61a6f00 (2018-07-29)
13. Rand-Hendriksen, Morten. CSS Grid Changes Everything (About Web Layouts). WebCamp Zagreb 2017. Plaza centar. Zagreb, 7.10.2017. [Predavanje]
14. Simmons, Jen. CSS Writing Modes, 2016. URL: https://24ways.org/2016/css-writing-modes/ (2018-07-30)
15. Simmons, Jen. Using Feature Queries in CSS, 2016. URL: https://hacks.mozilla.org/2016/08/using-feature-queries-in-css/ (2018-07-29)
16. Soueidan, Sara. Auto-Sizing Columns in CSS Grid: 'auto-fill' vs 'auto-fit', 2017. URL: https://css-tricks.com/auto-sizing-columns-css-grid-auto-fill-vs-auto-fit/ (2018-07-27)

17. Tables for Layout? Absurd. URL: https://thehistoryoftheweb.com/tables-layout-absurd/ (2018-07-21)
18. The (Mostly) Complete History of Layout on the Web Part 1: Liquid Cool. URL: https://thehistoryoftheweb.com/mostly-complete-history-layout-web-part-1-liquid-cool/ (2018-07-21)
19. The Grid. URL: https://foundation.zurb.com/grid.html (2018-07-28)

PRILOZI

Prilog 1. Hrvatsko - engleski rječnik pojmova (kronološki i abecedni)

Prilog 2. Popis CSS svojstava rasporeda elemenata implementacijom rešetaka i fleksibilnog rasporeda elemenata

Prilog 1. Hrvatsko - engleski rječnik pojmova

Kronološki

Okvirni sustav – **Framework**

Prilagodljiv dizajn – **Responsive Design**

Raspored elemenata – **Layout**

Raspored elemenata implementacijom rešetaka stilskog jezika / rešetke – **Grid Layout**

Tečni raspored elemenata – **Liquid Layout / Fluid Layout**

Raspored elemenata ovisno o razlučivosti – **Resolution Dependent Layout**

Medijski upiti **- Media Query**

Fleksibilni raspored elemenata – **Flexbox**

Prostor rešetaka – **Grid Container**

Stavka rešetaka – **Grid Item**

Linija rešetaka – **Grid Line**

Ćelija rešetaka – **Grid Cell**

Traka rešetaka – **Grid Track**

Područje rešetaka – **Grid Area**

Međuprostor rešetaka – **Grid Gap / Gutters**

Potomak – **Child Element**

Frakcija **- Fraction**

Frakcijska jedinica – **Fractional Unit**

Upiti svojstava – **Feature Query**

CSS modul uvjetnih pravila – **CSS Conditional Rules Module**

Okviri – **Boxes**

Načini pisanja – **Writing Modes**

Alati pristupnog dijela – **Front-End Development**

JavaScript dodatci – **Javascript Plugins**

Raspored elemenata implementacijom rešetaka modul stupanj 1 – **Grid Layout Module Level 1**

Prilagođavanje objekata – **Object Fit**

Mjerne jedinice za prikaz – **Viewport Units**

Više stupaca – **Multicolumns**

Dimenzioniranje – **Sizing**

Abecedni

Alati pristupnog dijela – **Front-End Development**

CSS modul uvjetnih pravila – **CSS Conditional Rules Module**

Ćelija rešetaka – **Grid Cell**

Dimenzioniranje – **Sizing**

Fleksibilni raspored elemenata – **Flexbox**

Frakcija **- Fraction**

Frakcijska jedinica – **Fractional Unit**

JavaScript dodatci – **Javascript Plugins**

Linija rešetaka – **Grid Line**

Medijski upiti **- Media Query**

Međuprostor rešetaka – **Grid Gap / Gutters**

Mjerne jedinice za prikaz – **Viewport Units**

Načini pisanja – **Writing Modes**

Okviri – **Boxes**

Okvirni sustav – **Framework**

Područje rešetaka – **Grid Area**

Potomak – **Child Element**

Prilagodljiv dizajn – **Responsive Design**

Prilagođavanje objekata – **Object Fit**

Prostor rešetaka – **Grid Container**

Raspored elemenata – **Layout**

Raspored elemenata implementacijom rešetaka modul stupanj 1 – **Grid Layout Module Level 1**

Raspored elemenata implementacijom rešetaka stilskog jezika / rešetke – **Grid Layout**

Raspored elemenata ovisno o razlučivosti – **Resolution Dependent Layout**

Stavka rešetaka – **Grid Item**

Tečni raspored elemenata – **Liquid Layout / Fluid Layout**

Traka rešetaka – **Grid Track**

Upiti svojstava – **Feature Query**

Više stupaca – **Multicolumns**

Prilog 2. Popis CSS svojstava rasporeda elemenata implementacijom rešetaka i fleksibilnog rasporeda elemenata

Raspored elemenata implementacijom rešetaka

grid-template-columns	grid-template-rows	grid-template-areas
grid-template	grid-auto-columns	grid-auto-rows
grid-auto-flow	grid	grid-row-start
grid-column-start	grid-row-end	grid-column-end
grid-row	grid-column	grid-area
grid-row-gap	grid-column-gap	grid-gap

Fleksibilni raspored elemenata

align-content	align-items	align-self
flex	flex-basis	flex-direction
flex-flow	flex-grow	flex-shrink
flex-wrap	justify-content	order