Web stranice i različite veličine ekrana

Nijedna web stranica se ne bi trebala praviti samo da radi i izgleda funkcionalno na jednoj rezoluciji ili veličini ekrana. Trebamo biti svjesni da se web stranice koriste putem raznih uređaja: mobitela, tableta, smart tv-a, laptopa (raznih veličina), širokih ekrana, ekrana velike rezolucije (4K) itd. Postoje tri osnovne stvari koje trebate realizovati da bi vaša stranica bila funkcionalna na raznim rezolucijama:

1. media queries – kontrola koji stilovi se upotrebljavaju na kojim veličinama ekrana
2. fluid grids – fluidni layouti
3. skalabilne slike

Kod dizajniranja web stranica trebao bi se pratiti princip postepenog unaprijeđivanja. Prvo kreirate osnovni izgled stranice koji sadrži osnovne (core) informacije, a onda u zavisnosti od dodatnih mogućnosti ekrana prikazujete i dodatne informacije koje se inače ne bi mogle prikazati. Kroz sljedeća poglavlja dat ćemo pregled osnovnih tehnika kako ostvariti cilj (ispravan prikaz web stranice na raznim uređajima).

Web stranice i layouti

Postoje razne varijante layouta:

1. Layout **sa fiksnom širinom** – prednost ovakvih layouta je što nam je unaprijed poznata širina stranice i prostor kojim raspolažemo, ali to dolazi sa manom, jer je prikaz ovakvih layouta na ekranima koji su uži od zadate širine odsječen (jedan dio stranice se ne vidi) ili ako je ekran širi od zadane širine tada imamo neplanirani prazan prostor.

2. **Fluidni** layouti – dimenzije na ovakvim layoutima su određene procentima, a ne fiksnim iznosima u pikselima. Dosta problema koje layouti sa fiksnom širinom imaju su riješeni u slučaju fluidnih layouta. Sami fluidni layouti nisu dovoljni da bi stranica izgledala dobro na svim uređajima jer mogu uzrokovati da su određeni elementi u layoutu preširoki ili preuski na nekim ekranima. Da bi triješili ovaj problem potrebne su nam i ostale stvari opisane u narednim poglavljima.

3. **Elastični** layouti – rade na sličan način kao i fluidni, ali se kod ovih layouta veličine definišu u relativnim jedinicama poput **em**. Jedinica em označava odnos dužine u odnosu na veličinu fonta, tako da je 2em dužina dva puta veličine fonta, a 0.5em pola veličine fonta. Prednost ovih layouta je ukoliko korisnik povećava ili smanjuje veličinu fonta tako se povećavaju/smanjuju elementi u layoutu.

Primjer:

Slika 1: Lijevo layout sa veličinom fonta od 14px, desno sa veličinom od 21px

4. **Hibridni** layouti – ovi layouti kombinuju tehnike iz ostale tri vrste layouta kako bi zadovoljile specifične potrebe. Npr. neki elementi stalno trebaju biti iste veličine tada na tom dijelu stranice koristimo fiksne veličine, a na ostalim procente i sl.

Svi layouti, osim layouta sa fiksnom širinom, će se koristiti za izradu responzivnih web stranica.

Veličine fontova

Postoje tri glavna načina kako postaviti veličinu fonta:

1. Postaviti veličinu **u pikselima** – potencijalni problem sa ovakvim pristupom je što veličina piksela na nekim ekranima s velikom gustinom piksela po inču, a malom dijagonalom može učiniti da tekst od 12px bude nečitljiv i to što skaliranje ovakvih veličina nije podržano u nekim web preglednicima.

2. Postaviti veličinu **u procentima** – Procenat govori o tome kolika je veličina fonta trenutnog elementa u odnosu na roditelja, 50% znači pola veličine fonta roditeljskog elementa i sl.

3. Postaviti veličinu fonta **u em jedinicama** – 1em odgovara veličini fonta postavljenog za dati element, odnoso 16px ukoliko veličina nije postavljena. Najčešći način kako se postavlja i koristi em jedinica je da se postavi veličina fonta body-a na **62.5%** što znači da će 1em iznositi 10px (62.5% od 16px je 10px) i tada je rad sa ovim jedinicama znatno lakši.

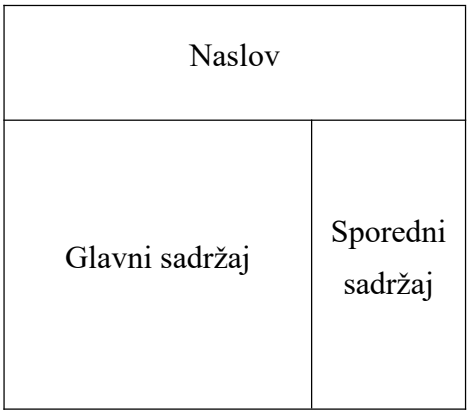
Mrežni layouti (grid layouts)

Postavljanje elemenata u mrežu čini stranicu urednom, konzistentnom, predvidljivom (lakom za korištenje), promjenjivom (olakšane izmjene). Kada crtate mrežu na papiru poznate su vam dimenzije papira, kako smo ranije rekli, veličinu ekrana ne možemo tek tako predvidjeti pa pribjegavamo raznim tehnikama kako posložiti stvari bez da odredimo fiksne veličine širine i visine. Za rad sa grid layoutima postoji veliki broj gotovih rješenja poput: flexboxgrid-a, foundation-a, bootstrap-a i sl. Korištenje ovih frameworka nećemo raditi u sklopu ovog predmeta, ovdje je cilj da se upoznamo sa idejama koje stoje iza ovih rješenja.

Kreirajmo layout za sljedeći HTML dokument čiji body glasi:

```
<div class="container">
<div class="zaglavlje">
<h1>Naslov</h1>
</div>
<div class="glavni">
<p>Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Vestibulum tortor quam, feugiat vitae, ultricies eget, tempor sit amet, ante. Donec eu libero sit amet quam egestas semper. Aenean ultricies mi vitae est. Mauris placerat eleifend leo. Quisque sit amet est et sapien ullamcorper pharetra. Vestibulum erat wisi, condimentum sed, commodo vitae, ornare sit amet, wisi. Aenean fermentum, elit eget tincidunt condimentum, eros ipsum rutrum orci, sagittis tempus lacus enim ac dui. Donec non enim in turpis pulvinar facilisis. Ut felis. Praesent dapibus, neque id cursus faucibus, tortor neque egestas augue, eu vulputate magna eros eu erat. Aliquam erat volutpat. Nam dui mi, tincidunt quis, accumsan porttitor, facilisis luctus, metus</p>
</div>
<div class="side">
<dl>
   <dt>Definition list</dt>
   <dd>Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</dd>
   <dt>Lorem ipsum dolor sit amet</dt>
   <dd>Consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</dd>
</dl>
</div>
</div>
```

Željeni izgled stranice je:



Ukoliko fiksno odredimo širinu ekrana CSS bi glasio:

```
.container{ width: 948px; }
.side{ float: right; width: 300px; }
.glavni{ float: left; width: 624px; }
```

Tada bi stranica na ekranima sa širinom od 948px izgledala ispravno, dok bi kod ekrana sa manjom širinom bila odsječena horizontalno.
Ukoliko iskoristimo ideju iz prošlog poglavlja i primijenimo fluidni layout tj. veličine iz piksela pretvorimo u procente tada CSS glasi:

```
.glavni{
    float: left;
    margin-right: 2.5316456%; /* 24px / 948px */
    width: 65.8227848%; /* 624/948 */
}
.container{
    width: 95%;
    padding: .625em 1.0548523% 1.5em; /* 10px/16px, 10px/948, 24px/16px */
    margin: auto 0;
}
.side {
    float: right;
    width: 31.6455696%; /* 300/948 */
}
```

U ovom slučaju stranica izgleda isto na ekranima sa širinom od 948px, dok je na drugim ekranima održan odnos da približno 60% stranice sadrži glavni sadržaj, a 30% sporedni.

Problem sa ovakvim rješenjem su slike. Ukoliko stavimo sliku u sporedni i/ili glavni sadržaj ona će zadržati svoju veličinu i pokvariti odnose veličina tekst:slika i sl. Prva stvar koja se mora uraditi je staviti da slike ne dobijaju veličinu iz HTML tj. atributi širine i visine ne bi smjeli biti postavljeni, a u CSS potrebno je postaviti širinu koristeći postotke. Pored postavljanja širine slike u postocima potrebno je staviti da slika ne prelazi izvan okvira svog roditeljskog elementa, to radimo koristeći CSS property **max-width** i njega postavljamo na 100%, što znači da se slika može proširiti najviše do širine svog roditeljskog elementa. Sa ovakvim stilom imamo stranicu koja je fluidna i upotrebljiva na raznim uređajima.

Media queries

Fluidni layouti su odlična stvar, ali ne rješavaju sve probleme. Ukoliko nam je potrebna veća kontrola nad tim kako će se sadržaj prikazivati pod raznim uslovima tada koristimo media queries.

Priča o pikselu i poremećaju višestruke ličnosti
Kada govorite o pikselu dobro pazite o kojem pikselu govorite. Piksel na vašem ekranu i piksel u CSS-u mogu, a i ne moraju biti isti. CSS piksel može zauzmati više piksela na samom uređaju. Ukoliko korisnik zumira stranicu do 400% tada jedan piksel definisan u CSS-u zauzima četri piksela na ekranu, ukoliko korisnik odzumira stranicu na 50% tada je širina i visina piksela prepolovljena. Ono što trebamo imati na umu je to da broj piksela na ekranu ostaje isti konstantno, dok se raspored CSS piksela mijenja.
Kako se čudne stvari dešavaju na nivou piksela to se prenosi i na nivo layouta. Na mobilnom uređaju postoje dvije širine, širina vidljivog prozora (širina uređaja) i širina prozora layouta. Ove dvije širine ne moraju biti iste (najčešće nisu). Prozor layouta možete smatrati velikom slikom, a prozor vidljivog možete smatrati kao prorez koji prikazuje dio velike slike i može se pomijerati u raznim pravcima kako bi sagledali veliku sliku. Kada koristimo mobilni uređaj pod zumiranjem navikli smo da se približi dio stranice, a ne da se čitav sadržaj poveća kao na desktop browserima. Veličine u CSS se odnose na prozor layouta i one se ne mijenjaju iako zumirate sadržaj na uređaju, mijenjaju se veličine u vidljivom prozoru. Ako rotirate uređaj vidljivi prozor se mijenja dok layout ostaje isti.

Postoji HTML tag koji omogućava da se specificira veličina prozora layouta tako da prozor vidljivog i prozor layouta budu usklađeni. Tag koji nam omogućava navedeno je **<meta>** tag i njegovi atributi **name=”viewport”, content=”directive,directive”,** gdje directive može biti:

- **width -** sa ovom direktivom specificiramo širinu layouta na uređaju, ukoliko je vrijednost fiksna (broj pixela) doći će do reskaliranja na širinu uređaja, najčešće se width postavlja na širinu ekrana uređaja sa **width=device-width**.
- **height** - ova direktiva nam omogućava da postavimo visinu layouta, u praksi se ne koristi često jer je korisnicima prirodno da visina layouta bude veća od visine prozora vidljivog, gdje pregledanje stranice vrške skroliranjem kroz sadržaj. Kao i kod širine i visina se može ograničiti na visinu ekrana uređaja (tada korisnik ne može skrolati sadržaj) sa **height=device-height**.
- **user-scalable -** sa ovom direktivom postavljamo osobinu da li korisnik može zumirati stranicu, ukoliko je vrijednost **user-scalable=no** tada korisnike ne može zumirati stranicu. Početna vrijednost ove direktive, ako je vi drugačije ne postavite, je **yes** i tako je u većini slučajeva najbolje. Rješenja prikaza stranice koja zahtijevaju da ova vrijednost bude na **no** su u suprotnosti sa očekivanjima korisnika i predstavljaju lošu praksu.
- **initial-scale** - ova direktiva ima vrijednost od 0.1 do 10.0 što predstavlja procenat koliko je sadržaj zumiran.
- **maximum-scale -** ovim postavljamo granicu do koje korisnik može zumirati stranicu, vrijednosti su od 0.1 - 10.0
- **minimum-scale -** ovom direktivom postavljamo granicu koliko korisnik može odzumirati stranicu, vrijednosti su od 0.1-10.0.

Ukoliko želimo da se stranica na mobilnim uređajima ne pojavljuje odzumirana i ako želimo da širina ekrana bude popunjena sa širinom stranice koristimo meta tag na sljedeći način:

```
<meta name=”viewport” content=”width=device-width” />
```

Napomena: Meta tag se dodaje unutar head taga.

Media quries - pregled

Media queries nam daju informacije o uređaju i omogućavaju da primjenjujemo određeni set stilova na jednom uređaju, a drugi set na drugom. Izgled jednog media query-a je:

```
@media [not|only] type [logički operator] (izraz){
    CSS pravila
}
```

Komponente ovog izraza su:

- type - tip uređaja - obavezan za sve media query-e. Uređaj može biti sve od projektora, monitora, pa do uređaja na Brajevom pismu za slijepe i slabovidne. Neki od tipova su:
  1. **all -** *svi uređaji***,**
  2. ***screen* -** *ekran u boji*,
  3. **tv -** *tv uređaj*,
  4. **print** - *prikaz za printanje i sl.*
- izraz - omogućava nam da provjerimo dodatna svojstva uređaja poput:
  1. **width** - *provjera širine ekrana, može biti sa prefiksom min ili max npr. min-width što znači da ekran treba biti najmanje širok koliko je zadano u uslovu, veličina se zadaje u pikselima(px), inčima (in), em-ovima i sl,*
  2. **height -** *provjera visina ekrana, može biti sa prefiksom min ili max kao i za* ***width***
  3. **orientation** - *orijentacija ekrana, može biti landscape ili portrait u zavisnosti od zadane orijentacije uređaja*
  4. **resolution -** *rezolucija, označava gustinu prikaza ekrana, može biti u tačkama po inču (dpi), tačkama po centimetru (dpcm).*
  5. **aspect-ratio -** *odnos širine i visine, predstavlja izraz u obliku a/b gdje su a i b cijeli brojevi koji govore o odnosu broja horizontalnih piksela prema broju vertikalnih piksela*
- logički operator - njime vezujemo više izraza u složenije izraze npr. ako želimo primijeniti neka pravila nad ekranom koji ima određenu širinu ekrana i određenu rezoluciju dva izraza vežemo sa **and**, ukoliko izraze odvajamo zarezom to predstavlja logičko ili, **not** ima značenje negacije cijelog media query-a, **only** sprječava browsere koji ne podržavaju media query-e da primijene navedeni stil.
- Gdje pisati media query-e? Media query se može pisati u samom CSS fajlu ili se može svaki stil za svaki media query izdvojiti u poseban CSS fajl, a media query pisati u HTML-u pri uključivanju na način da se u **<link>** tagu navede i atribut **media** koji će sadržavati media query. Primjer:

```
<link href=”style.css” media=”only screen and (min-width: 1300px)” />
```

Kod oba načina pisana media query-a sav stil će se downloadovati, a primjenjivat će se kada pravilo bude zadovoljeno (orijentacija promijenjena i sl.).

Postoje dva načina kako pisati stil:

- Mobile up - pišemo stil web stranice kao da će se izvršavati samo na mobilnim uređajima, a onda sa pravilima dodajemo i dodatne funkcionalnosti za veće ekrane, prednost ovog načina su što od početka pojednostavljujete dizajn i prikazujete najosnovnije informacije, kao i što rješavate problem da mobilni uređaji koji ne podržavaju media query-e imaju/dobijaju ispravan izgled web stranice.
- Desktop down- pišemo stil web stranice kao da će se prikazivati samo na desktop uređajima, a onda sa pravilima oduzimamo prikaz funkcionalnosti koji se ne mogu prikazati na mobilnim uređajima.

Više o media query-ima na:

- *https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries*
- *https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Testing_media_queries*
- *http://www.w3schools.com/css/css3_mediaqueries_ex.asp*

Responzivni sadržaj

Kada se pravi web stranica veliki dio pažnje oko sadržaja se treba posvetiti slikama. Jedna slika na jednoj veličini ekrana može biti adekvatna i informativna, dok na drugoj veličini ekrana može biti: sitna, mutna, prevelika i sl. Zbog toga bi bilo dobro po potrebi učitavati razne slike za razne ekrane. Slika visoke rezolucije će se prikazati i na ekranu sa niskom rezolucijom, ali pri tome smo potrošili nepotrebno vrijeme i bandwidth za skidanje podataka koji se neće moći prikazati na ekranu uređaja. Zbog toga se često koristi metoda da se naprave slike sa različitim rezolucijama koje će se učitavati po potrebi. Tako će uređaj sa ekranom visoke rezolucije dobiti sliku visoke rezolucije, a ekran niže rezolucije sliku koja je adekvatna njegovim mogućnostima prikaza.

Da slike ne bi remetile red koji smo uspostavili sa fluidnim templejtima i media query-ima trebale bi se primjeniti neke od sljedećih tehnika:

- Korištenje relativnih dimenzija slike, korištenje max-width svojstva, postavljanje širine u procentima i sl.
- **srcset** atribut img taga omogućava naznačavanje više fajlova slika od kojih će se jedna prikazivati u zavisnosti od osobina ekrana uređaja, kod web preglednika koji ne podržavaju ovaj atribut prikazuje se slika navedena u src atributu.
- Korištenje **picture** elementa (nije podržan u svim web preglednicima) koji omogućava da se učita određeni fajl u zavisnosti od media query-a koji je naveden u elementu djetetu **source** u njegovom atributu **media**.
- Zamjena slika korištenjem JavaScript-a - koristeći DOM property **window.devicePixelRatio** možemo dobiti širinu i visinu ekrana i na osnovu toga učitati sliku. Mana ovog pristupa je što se slike počinju učitavati tek nakon što je prošao proces parsiranja HTML-a i CSS-a.
- Učitavanje slike iz CSS-a - korištenjem background property-a može se putem CSS-a i korištenjem media query-a specificirati koja će se slika učitati ako je media query ispunjen. U CSS-u se tada postavlja vrijednost **background-image: url(url_slike)**.

Više o tehnikama za prikaz slika u responzivnim web stranicama na linku:

https://developers.google.com/web/fundamentals/design-and-ui/media/images