Poravnanje Grid elemenata

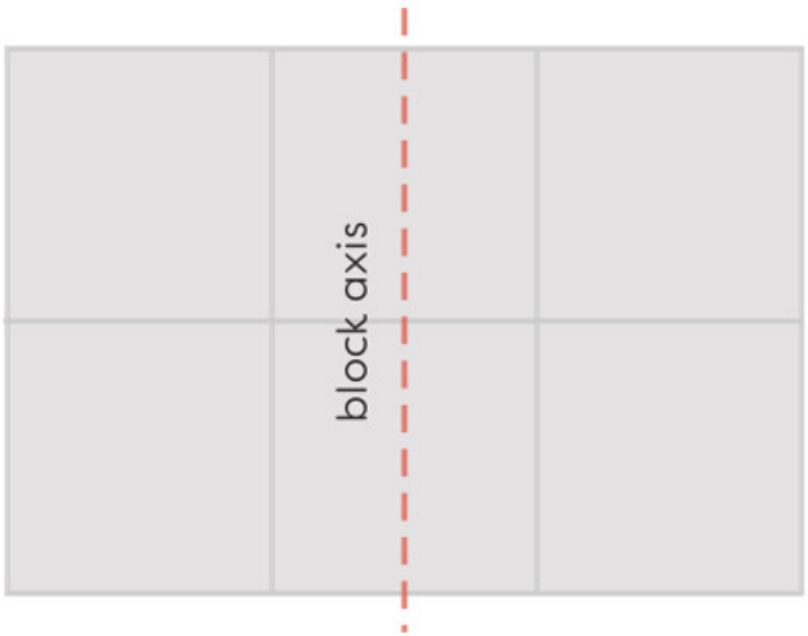
U prethodnim lekcijama prikazane su osnovne tehnike za konfigurisanje CSS Grida. Tako su obrađeni pojmovi kao što su kolone i redovi, prored, raspoređivanje elemenata na osnovu brojeva linija, oblasti i imenovane oblasti... Ipak, do sada nije bilo reči o poravnanju Grid elemenata. Stoga će lekcija pred vama biti posvećena takvoj problematici.

Ose CSS Grida

Kako bi se na pravi način razumela logika na osnovu koje funkcioniše poravnanje Grid elemenata, neophodno je razumeti pojmove osa Grid sistema. S obzirom na to da je reč o dvodimenzionalnom sistemu, CSS Grid poseduje dve ose:

- block (vertikalna) osa, odnosno osa kolona;
- inline (horizontalna) osa, odnosno osa redova.

Block osa ilustrovana je slikom 22.1.



Slika 22.1. Block osa Grid sistema

Vertikalna, odnosno block osa je ona osa na osnovu koje se raspoređuju block elementi. Upravo zbog takve osobine ova osa je i dobila svoj naziv.

Inline osa ilustrovana je slikom 22.2

Slika 22.2. Inline osa Grid sistema

Elemente unutar Grid sistema moguće je poravnavati po dve upravo ilustrovane ose. U nastavku lekcije biće ilustrovano kako tako nešto postići. Za realizaciju primera biće korišćen Grid sledeće strukture:

```
<div id="grid1">
      <div id="box1" class="box">Element 1</div>
      <div id="box2" class="box">Element 2</div>
      <div id="box3" class="box">Element 3</div>
      <div id="box4" class="box">Element 4</div>
      <div id="box5" class="box">Element 5</div>
</div>
```

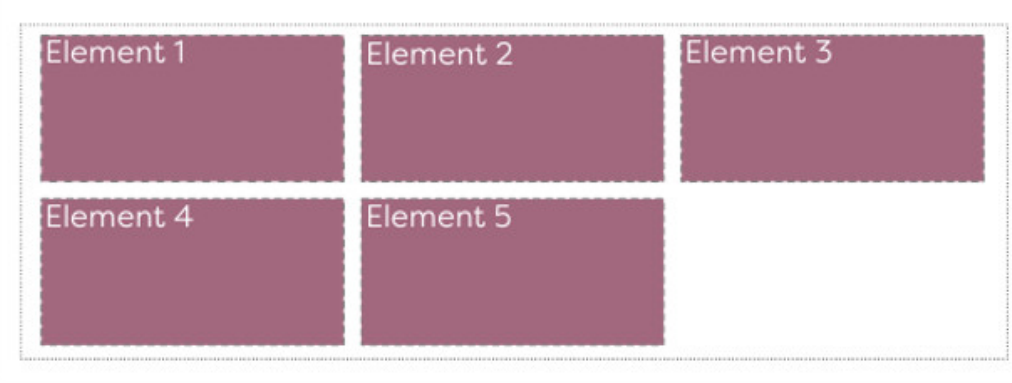
Ovakvi HTML elementi biće stilizovani na sledeći način:

```
#grid1 {
      display: grid;
      grid-template-columns: auto auto auto;
      grid-auto-rows: minmax(100px, auto);
      gap: 16px;
}

.box {
      background-color: #914E67;
      color: white;
}
```

Na ovaj način kreiran je jedan CSS Grid sa pet elemenata. Tri kolone su kreirane eksplicitno, korišćenjem svojstva `grid-template-columns`, dok je kreiranje redova prepušteno CSS Gridu. S obzirom na to da Grid poseduje pet elemenata, u takvoj situaciji će biti obavljeno automatsko kreiranje dva implicitna reda unutar Grida. Na takve implicitno kreirane redove utiče se korišćenjem svojstva `grid-auto-rows`. Vrednošću ovog svojstva je definisano da će automatski kreirani redovi imati minimalnu visinu od 100px, dok će maksimalna visina zavisiti od visine sadržaja.

Upravo opisani CSS Grid ilustrovan je slikom 22.3.



Slika 22.3. Grid koji će biti korišćen u nastavku lekcije

Poravnanje Grid elemenata po block osi

Za poravnavanje Grid elemenata po block osi moguće je koristiti dva CSS svojstva:

- `align-items` – svojstvo koje se koristi za poravnavanje svih Grid elemenata po block osi; upravo zbog toga se definiše na Grid kontejneru;
- `align-self` – svojstvo koje se koristi za poravnavanje pojedinačnih elemenata po block osi; definiše se na Grid elementima.

Razlika između dva upravo navedena svojstva za poravnavanje elemenata po block osi veoma je jasna. Svojstvo `align-items` se koristi kada je potrebno sve elemente Grida poravnati na identičan način, dok se svojstvo `align-self` koristi za poravnavanje pojedinačnih elemenata. Najznačajnije vrednosti koje ova svojstva mogu imati ilustrovane su tabelom 22.1.

Vrednost	Opis
`start`	element/elementi se poravnavaju po gornjoj ivici trake
`end`	element/elementi se poravnavaju po donjoj ivici trake
`center`	elementi se centriraju po block osi
`stretch`	elementi se razvlače tako da popune dostupan prostor po block osi
`baseline`	elementi se poravnavaju po osnovnoj liniji teksta
`normal`	podrazumevana vrednost; u najvećem broju slučajeva, ova vrednost se ponaša kao i vrednost `stretch`, osim kod elemenata kod kojih je odnos stranica značajan (npr. slike); kada je odnos stranica značajan, ova vrednost se ponaša kao vrednost `start`

Tabela 22.1. Vrednosti svojstava `align-items` i `align-self`

Efekat upravo navedenih vrednosti ilustrovan je slikom 22.4.

Slika 22.4. Različita poravnanja Grid elemenata po block osi

Na slici 22.4. bitno je primetiti da vrednosti `start`, `end`, `center` i `baseline` utiču i na veličinu Grid elemenata, koja će u slučaju korišćenja navedenih vrednosti biti taman tolika da obuhvati sadržaj koji se unutar njih nalazi. Vrednosti `stretch` i `normal` čine da elementi zauzmu kompletan dostupni prostor po block osi. Naravno, tako nešto je za `normal` vrednost tačno samo ukoliko proporcije Grid elementa nisu značajne.

Upravo prikazane vrednosti mogu se koristiti i na svojstvu `align-self`. Ipak, svojstvo `align-self` definiše se na pojedinačnim elementima, pa tako dozvoljava poravnanje svakog elementa pojedinačno.

Pitanje

Poravnanje pojedinačnih elemenata po block osi unutar Grida, moguće je postići svojstvom:

- **`align-self`**
- `self-align`
- `align-items`
- `items-align`

Objašnjenje:

Za poravnanje Grid elemenata po block osi moguće je koristiti CSS svojstva `align-items` i `align-self`. Svojstvo `align-self` koristi se za poravnanje pojedinačnih elemenata po block osi, a svojstvo `align-items` za poravnanje svih elemenata odjednom.

Poravnanje elemenata unutar Grida po inline osi

Za poravnanje Grid elemenata po inline osi koriste se sledeća dva CSS svojstva:

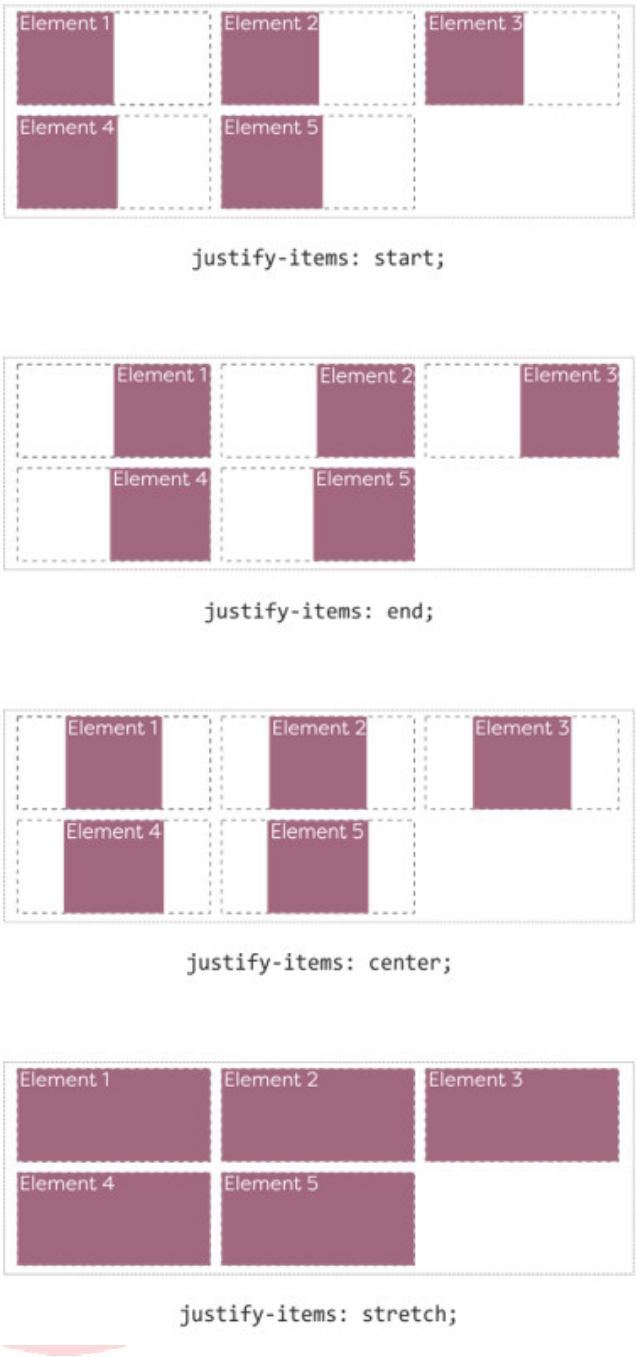
- `justify-items` – svojstvo koje se koristi za poravnavanje svih Grid elemenata po inline osi; definiše se na Grid kontejneru;
- `justify-self` – svojstvo koje se koristi za poravnavanje pojedinačnih Grid elemenata po inline osi; definiše se na Grid elementima.

Najznačajnije vrednosti svojstava `justify-items` i `justify-self` ilustrovane su tabelom 22.2.

Vrednost	Opis
`start`	element/elementi se poravnavaju po levoj ivici trake
`end`	element/elementi se poravnavaju po desnoj ivici trake
`center`	elementi se centriraju po inline osi
`stretch`	elementi se razvlače tako da popune dostupan prostor po inline osi
`normal`	podrazumeva vrednost; u najvećem broju slučajeva, ova vrednost se ponaša kao i vrednost `stretch`, osim kod elemenata kod kojih je odnos stranica značajan (npr. slike); kada je odnos stranica značajan, ova vrednost se ponaša kao vrednost `start`

Tabela 22.2. Vrednosti svojstava justify-items i justify-self

Efekti koje vrednosti iz tabele 22.2. proizvode ilustrovani su slikom 22.5.



Slika 22.5. Različita poravnanja Grid elemenata po inline osi

Sa slika 22.4. i 22.5. može se videti da poravnanja po block i inline osama funkcionišu na identičan način – naravno, uz razliku kad je reč o referentnoj osi po kojoj se vrši poravnanje. Na slici 22.5. može se videti da, baš kao u prethodnom primeru, vrednosti `start`, `end` i `center` čine da elementi budu taman toliko široki koliko je potrebno da obuhvate sadržaj koji se unutar njih nalazi. I ovde važi pravilo za `normal` vrednost kao i kod block ose: vrednost `normal` ima identičan efekat kao i vrednost `stretch` ukoliko se primenjuje nad elementom čije proporcije nisu značajne. U protivnom, `normal` se ponaša kao `start`.

Napomena

Efekat vrednosti `start` i `end` primarno zavisi od usmerenja teksta unutar HTML dokumenta. Ukoliko je usmerenje teksta sleva nadesno, efekat je kao na prikazanim slikama. Ipak, ukoliko je usmerenje teksta zdesna nalevo, vrednosti `start` i `end` imaju obrnuti efekat.

Poravnanje Grid traka

U nekim situacijama može doći do toga da trake unutar kojih se raspoređuju Grid elementi budu manje od kompletnog Grid elementa. Kako bi se stvorila takva situacija, osobine Grida koji je korišćen u dosadašnjem toku lekcije biće blago izmenjene:

```
#grid1 {
    display: grid;
    grid-template-columns: 160px 160px 160px;
    grid-auto-rows: minmax(100px, 140px);
    gap: 16px;
    height: 60vh;
    border: 1px dashed;
}
```

Izmene se odnose na širine kolona koje su sada eksplicitno postavljene na 160px. Takođe, ograničena je i visina redova, tako da oni sada ne mogu biti viši od 140px. Na kraju, na kompletnom Grid kontejneru je definisan okvir, kako bi se lakše uočile njegove granice. Sve ove promene stvoriće prikaz kao na slici 22.6.

Slika 22.6. Grid unutar koga postoji slobodan prostor po block i inline osama

Sa slike 22.6. je potpuno jasno da je zbog ograničavanja veličine kolona i redova Grid kontejner većih dimenzija od svojih traka za prikaz elemenata. CSS Grid poseduje dva svojstva koja je moguće koristiti kako bi se uticalo na poravnanje traka u ovakvim situacijama:

- `align-content` – za poravnanje po block osi;
- `justify-content` – za poravnanje po inline osi.

Poravnanje Grid traka po block osi

Svojstvo **align-content** koristi se za vertikalno poravnanje, odnosno za poravnanje po block osi ili osi kolona.

Najznačajnije vrednosti svojstva `align-content` predstavljene su tabelom 22.3.

Vrednost	Opis
`start`	horizontalne trake se poravnavaju po početku block ose
`end`	horizontalne trake se poravnavaju po kraju block ose
`center`	horizontalne trake se centriraju po blok osi
`space-between`	horizontalne trake se raspoređuju ravnomerno unutar kontejnera, pri čemu se prva traka smešta na početak, a poslednja na kraj kontejnera
`space-around`	horizontalne trake se ravnomerno raspoređuju sa jednakim praznim prostorom pre i posle
`stretch`	horizontalne trake se proširuju kako bi se zauzeo kompletan dostupni prostor po block osi, ali samo ukoliko je visina redova postavljena na `auto`

Tabela 22.3. Vrednosti align-content svojstva

Efekti koje proizvode različite vrednosti `align-contet` svojstva prikazani su slikom 22.7.

Slika 22.7. Efekti različitih vrednosti align-contet svojstva

Poravnanje Grid traka po inline osi

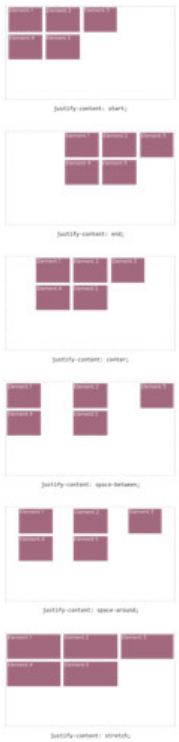
Svojstvo **justify-content** koristi se za horizontalno poravnavanje traka, odnosno za poravnanje po takozvanoj inline osi ili osi redova.

Najznačajnije vrednosti svojstva `justify-content` ilustrovane su tabelom 22.4.

Vrednost	Opis
`start`	vertikalne trake se poravnavaju po početnoj ivici svog roditelja; to je obično leva ivica, mada može biti i desna, kada je definisan rtl sistem pisanja
`end`	vertikalne trake se poravnavaju po krajnjoj ivici svog roditelja; to je obično desna ivica
`center`	vertikalne trake se centriraju po inline osi
`space-between`	vertikalne trake se jednako raspoređuju duž inline ose, tako što su prva i poslednja traka prilepljene za ivice
`space-around`	vertikalne trake se jednako raspoređuju duž inline ose, sa jednakom količinom prostora oko svake trake
`stretch`	vertikalne trake se proširuju kako bi se zauzeo kompletan dostupni prostor po inline osi, ali samo ukoliko je širina kolona postavljena na `auto`

Tabela 22.4. Vrednost justify-content svojstva

Efekti različitih vrednosti svojstva `justify-content` ilustrovani su slikom 22.8.



Slika 22.8. Efekti različitih vrednosti justify-content svojstva

Primer 1 – Korišćenje CSS Grida za kreiranje jednostavnog layouta stranice

Neke od tehnika koje su ilustrovane u ovoj i prethodnim lekcijama sada će biti iskorišćene za kreiranje realnog primera upotrebe CSS Grida za realizaciju layouta jednostavne web stranice. Struktura CSS Grida na takvoj stranici će izgledati ovako:

```
<div id="grid-container">
	<header>Header</header>
	<article>
		<h1>Article Title</h1>
		<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.
		Phasellus gravida, enim nec fermentum ullamcorper, eros
		metus rhoncus ante, in ornare arcu purus id est. Duis
		viverra mi felis, nec feugiat tortor scelerisque ut. Ut
		efficitur ac ipsum a viverra.</p>
	</article>
	<aside>
		<h1>Sidebar</h1>
		<ul>
			<li>consectetur</li>
			<li>phasellus</li>
			<li>ullamcorper</li>
			<li>efficitur</li>
			<li>viverra</li>
		</ul>
	</aside>
	<footer>Footer</footer>
</div>
```

Upravo prikazani HTML elementi biće pretvoreni u CSS Grid na sledeći način:

```
body {
	font-family: sans-serif;
	color: white;
}

#grid-container {
	display: grid;
	grid-gap: 8px;
	grid-template-columns: 1fr 3fr;
	grid-template-areas:
	"header  header"
	"sidebar content"
	"footer  footer";

}

header, article aside, footer {
	padding: 16px;
}

header {
	grid-area: header;
	background-color: #5D7586;
}
```

```
article {
	grid-area: content;
	background-color: #914E67;
}

aside {
	grid-area: sidebar;
	background-color: #37424c;
}

footer {
	grid-area: footer;
	background-color: #6d4856;
}
```

Efekat koji se na ovaj način dobija ilustrovan je slikom 22.9.

Slika 22.9. Primer realizacije jednostavnog layouta korišćenjem CSS Grida

Primer 2 – Korišćenje CSS Grida za kreiranje prilagodljivog grida elemenata (1)

Naredni primer ilustrovaće kreiranje prilagodljivog grida korišćenjem CSS Grida. Primer će biti primarno realizovan korišćenjem funkcionalnosti koje su ilustrovane u prethodnoj lekciji i koje će omogućiti da se responsive ponašanje u potpunosti dobije bez upotrebe medija upita. HTML struktura će izgledati ovako:

```
<div class="container">
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
</div>
```

Element sa klasom `container` biće iskorišćen kao Grid omotač i na njemu će biti definisane sledeće osobine:

```
.container {
	max-width: 960px;
	width: 90%;
	margin: 0 auto;

	display: grid;
	grid-template-columns: repeat(auto-fit, minmax(144px, 1fr));
	grid-gap: 8px;
}
```

Kompletno responsive ponašanje biće dobijeno korišćenjem samo jedne linije:

```
grid-template-columns: repeat(auto-fit, minmax(144px, 1fr));
```

Na ovaj način je browseru rečeno da je potrebno da samostalno odredi broj kolona unutar Grida. Broj kolona će biti određen u zavisnosti od širine vidnog polja. Za minimalnu širinu kolona je postavljena vrednost od 144px. Tako će browser u svakom trenutku pokušati da unutar jednog reda smesti najveći mogući broj kolona, a da pri tome širina nijedne kolone ne bude manja od 144px. Na ovaj način, oslobođeni smo pisanja bilo kakvih medija upita, i sve što preostaje jeste definisanje jednostavne stilizacije za Grid elemente:

```
.box {
	height: 144px;
	background-color: #914E67;
}
```

Efekat ovakvog koda ilustrovan je animacijom 22.1.

Animacija 22.1. Primer prilagodljivog grida (1)

Napomena

Primer je moguće realizovati i korišćenjem sledeće linije:

```
grid-template-columns: repeat(auto-fit, minmax(144px, auto));
```

Jedina razlika ogleda se u upotrebi vrednosti `auto` umesto `1fr`. Na prvi pogled, efekat će biti identičan, ali prilikom nejednake količine sadržaja unutar kolona, vrednost `auto` će najviše prostora dati koloni sa najviše sadržaja. Takvog ponašanja nema prilikom upotrebe vrednosti `1fr`, kada sve kolone moraju biti identične širine.

Primer 3 – Korišćenje CSS Grida za kreiranje prilagodljivog grida elemenata (2)

Prethodni primer ilustrovao je kreiranje prilagodljivog grida koji je u potpunosti realizovan bez upotrebe medija upita. Ukoliko je potrebno preuzeti veću kontrolu nad prelomom redova i distribucijom kolona, funkcionalnosti CSS Grida mogu se upariti sa medija upitima.

HTML struktura će biti ista kao u prethodnom primeru:

```
<div class="container">
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
	<div class="box"></div>
</div>
```

Okružujući `div` element (`.container`) biće stilizovan na sledeći način:

```
.container {
	max-width: 960px;
	width: 90%;
	margin: 0 auto;

	display: grid;
	grid-template-columns: repeat(6, 1fr);
	grid-gap: 8px;
}
```

Ovoga puta je za vrednosti `grid-template-columns` svojstva definisana vrednost nešto jednostavnijeg oblika. Upotrebljena je funkcija `repeat()`, kojom je rečeno da je potrebno stvoriti šest kolona, čije će širine biti postavljene na `1fr`. Na taj način će sve kolone podjednako podeliti prostor unutar jednog reda.

Prilikom promene osobina vidnog polja, ovakav grid će adekvatno reagovati, korišćenjem sledećih medija upita:

```
@media screen and (max-width: 1024px) {
	.container {
	grid-template-columns: repeat(3, 1fr);
	}
}

@media screen and (max-width: 768px) {
	.container {
	grid-template-columns: repeat(2, 1fr);
	}
}

@media screen and (max-width: 414px) {
	.container {
	grid-template-columns: repeat(1, 1fr);
	}
}
```

Unutar upravo prikazanih medija upita, utiče se na broj kolona grida na različitim širinama vidnog polja. Efekat koji se na ovakav način dobija ilustrovan je animacijom 22.2.

Animacija 22.2. Primer prilagodljivog grida (1)

Rezime

- CSS Grid poseduje dve ose: osu kolona (block ili vertikalna osa) i osu redova (inline ili horizontalna osa).
- Za poravnanje Grid elemenata po block osi moguće je koristiti CSS svojstva `align-items` i `align-self`.
- `align-items` je svojstvo koje se koristi za poravnanje svih Grid elemenata po block osi.
- `align-self` je svojstvo koje se koristi za poravnanje pojedinačnih elemenata po block osi.
- za poravnanje Grid elemenata po inline osi, koriste se CSS svojstva `justify-items` i `justify-self`.
- `justify-items` je svojstvo koje se koristi za poravnanje svih Grid elemenata po inline osi.
- `justify-self` je svojstvo koje se koristi za poravnanje pojedinačnih Grid elemenata po inline osi.
- Poravnanje kompletnih Grid traka se može postići korišćenjem svojstava `align-content` i `justify-content`.