

TypeScript Flexbox Flexlayout

Dr. Mlađan Jovanović
mjovanovic@singidunum.ac.rs

Sadržaj

- ◆ **TypeScript:**
 - ◆ Instalacija i korišćenje
 - ◆ Tipovi podataka
 - ◆ Deklaracija i definicija varijabli i funkcija
 - ◆ Interfejsi
 - ◆ Klase
 - ◆ Prevođenje u JS (primeri)
- ◆ **Flexbox:**
 - ◆ Direktive za definisanje rasporeda u CSS fajlu
- ◆ **Flexlayout:**
 - ◆ Direktive za definisanje rasporeda u templejtu (HTML)

Uvod u TypeScript

- ◆ Nadskup JavaScript jezika
 - JS kod predstavlja validan TS kod
- ◆ TS dodaje elemente JS jeziku radi jednostavnije upotrebe
 - Objektno-orientisani (OO) elementi
- ◆ TS kompajler prevodi TS kod u JS kod koji se zatim izvršava
- ◆ Instalacija TS kompajlera:
- ◆ Najpre je potrebno instalirati Node.js kao JS kompajler i izvršno okruženje
- ◆ Lokalna
 - Za svaki Angular projekat kreiran u Visual Studio Code razvojnom okruženju (**ng new**) automatski se dodaje podrška za TS
- ◆ Globalna
 - `npm install -g typescript`
 - `tsc --version`

Lokalna TS instalacija u Angular projektu

tsconfig.json

```
{  
  "compileOnSave": false,  
  "compilerOptions": {  
    "baseUrl": "./",  
    "outDir": "./dist/out-tsc",  
    "sourceMap": true,  
    "declaration": false,  
    "module": "es2015",  
    "moduleResolution": "node",  
    "emitDecoratorMetadata": true,  
    "experimentalDecorators": true,  
    "importHelpers": true,  
    "target": "es5",  
    "typeRoots": [  
      "node_modules/@types"  
    ],  
    "lib": [  
      "es2018",  
      "dom"  
    ]  
  }  
}
```

Build Angular aplikacije

ng serve – lokalno pokretanje Angular aplikacije

ng build --prod – kreiranje produkcionе verzije aplikacije

kreira **dist** direktorijum (distribution) sa izvšnom verzijom
koja se može pokrenuti pomoću Web servera (Apache)

prod opcija kreira izvršni kod što je moguće manje veličine

package.json

```
"scripts": {  
  "ng": "ng",  
  "start": "ng serve",  
  "build": "ng build",  
  "test": "ng test",  
  "lint": "ng lint",  
  "e2e": "ng e2e"  
},
```

TS tipovi podataka

number

string

boolean

null

undefined

object

function

any

Primitivni tipovi:

“osnovni tipovi” na osnovu kojih se mogu izvesti složeniji tipovi

Složeni tipovi: izvedeni iz osnovnih tipova

any:

Specijalan tip koji može predstavljati bilo koju vrednost

TypeScript tipovi podataka 1/2

- ◆ Osnovni tipovi:

- ◆ **Boolean**

```
let isDone: boolean = false;
```

- ◆ **Number** (floating point broj)

```
let decimal: number = 6;
```

- ◆ **String** (upotreba i korišćenje identični kao u JS)

```
let fullName: string ='Mladjan Jovanovic';
```

```
let sentence: string ='Hello, my name is ${ fullName }';
```

- ◆ **Nizovi** (upotreba i korišćenje identični kao u JS):

- ◆ Primitivan tip

```
let list: number[] = [1, 2, 3];
```

- ◆ Generički tip

```
let list: Array<number> = [1, 2, 3];
```

TypeScript tipovi podataka 2/2

- ◆ **Enum** element nabranja

```
enum Color {Red, Green, Blue}
```

```
let c: Color = Color.Green;
```

- ◆ **Any** (korisno ukoliko nismo unapred sigurni u tip vrednosti promenjive u vreme pisanja programa)

```
let notSure: any = 4;
```

```
notSure = "maybe a string instead";
```

```
notSure = false;
```

TS je strogo tipiziran jezik, ukoliko se tip ne navede on se prvi put implicitno izvodi iz tipa vrednosti koja se dodeljuje, ali se ne može menjati !

```
let name = "Hello";
```



```
let name: string = "Hello";
```

```
let someValue = 1;
```



```
let someValue: string = 1;
```

TS deklaracija varijabli i funkcija

`let variableName: TypeName;`

`const CONSTANT_NAME: TypeName;`

`function functionName(param1: Type1, param2: Type2):
ReturnType {}`

`let age: number = 20;`

`const ALPHABET: string =
"ABCDEFGHIJKLMNOPQRSTUVWXYZ";`

`function divides(divisor: number, dividend: number):
boolean { }`

object tip je identičan kao u JS

```
let dog = {  
  type: 'mammal',  
  name: 'dog',  
  sounds: ['woof', 'bark', 'yip', 'ruff'] };
```

```
let enigma = {  
  rotors: [],  
  lamps: [],  
  keys: [] };
```

```
let profile = {  
  name: "Mladjan Jovanovic",  
  imageUrl: "http://image.url/img.png",  
  language: "English" };
```

Interfejsi (interface)

- ◆ TS interfejsi definišu **strukturu** objekata
- ◆ Interfejsi nisu objekti, tj. object tip
- ◆ Interfejsi nemaju funkcionalnosti već opisuju isključivo strukturu
- ◆ Korišćenje – definisanje složenijih struktura aplikacije koje se sastoje od atributa (property)

```
interface InterfaceName {  
    property1: Type1  
    property2: Type2  
}
```

- ◆ Opcioni atributi se navode korišćenjem ? na kraju naziva atributa

```
interface Point {  
    x?: number  
    y?: number  
}
```

Nasleđivanje interfejsa

- ◆ Interfejsi se mogu nasleđivati (**extend**)

```
interface Shape {  
    color: string;  
}  
  
interface Square extends Shape {  
    sideLength: number;  
}  
  
let square = <Square>{};  
square.color = "blue";  
square.sideLength = 10;
```

Interfejs – nasleđivanje i instanciranje

```
interface SquareConfiguration {
    width: number
    height: number
    color: string
}

interface Square extends SquareConfiguration {
    area: number
    perimeter: number
}

function createSquare(config: SquareConfiguration): Square {
    return {
        width: config.width,
        height: config.height,
        color: config.color,
        area: config.width * config.height,
        perimeter: config.width * 2 + config.height * 2
    }
}

let square = createSquare({width: 2, height: 2, color: "Green"});
console.log("The new square's color is: " + square.color);
```

Klase (class)

- ◆ Predstavljaju složen tip, identično kao u OO programiranju
- ◆ Mogu se posmatrati i kao interfejsi sa funkcijama
- ◆ Korišćenje identično kao u OO programiranju:
- ◆ Nasleđivanje (**extends**)
 - Izvedene klase moraju pozivati konstruktor nadređene klase (**super**)
- ◆ Kreiranje objekta klase (**new**)
- ◆ Pristup atributima klase (**this**)
 - This unutar tela klase ukazuje na tekući objekat
 - readonly – atribut čija se vrednost ne može menjati (navodi se konstruktoru)
- ◆ Modifikatori vidljivosti
 - public – podrazumevano za sve metode i attribute
 - private – privatan atribut (nije vidljiv van objekta klase)
 - protected – private + vidljivi u izvedenim klasama
- ◆ Accessor metode (**get/set**)
 - Novije verzije TS omogućavaju pristup atributima bez `_` simbola !
- ◆ Statički atributi (**static**)
 - Zajednički za sve instance klase

Klase – kreiranje i instanciranje

```
class Profile {  
    readonly name: string;  
    private imageUrl: string;  
    statuses: string[];  
    constructor(initialName: string, initialImage: string) {  
        this.name = initialName;  
        this.imageUrl = initialImage;  
        this.statuses = [];  
    }  
    getImage(): string {  
        return this.imageUrl;  
    }  
    setImage(newImage: string): void {  
        this.imageUrl = newImage;  
    }  
    addStatus(newStatus: string): void {  
        this.statuses.push(newStatus);  
    }  
}  
let mladjan = new Profile("Mladjan Jovanovic", "https://image.url/");
```

Vidljivost TS klase i interfejsa

- ◆ Prilikom definisanja klase ili interfejsa može se korisiti ključna reč **export**
- ◆ Značenje ključne reči je isključivo da će klasa ili interfejs biti vidljivi u čitavoj Angular aplikaciji
- ◆ Korišćenje ključne reči nije obavezno prilikom definisanja klase ili interfejsa !

TS funkcije

- ◆ Imaju identično značenje i korišćenje kao u JS

- ◆ Imenovana funkcija:

```
function add(x: number, y: number): number {  
    return x + y; }
```

- ◆ Anonimna funkcija:

```
let myAdd = function(x: number, y: number): number  
{ return x + y; };
```

- ◆ Void funkcija – funkcija koja ne vraća vrednost:

```
function warnUser(): void {  
    console.log("This is my warning message"); }
```

TS funkcije

- ◆ Opcioni parametri:

```
function buildName(firstName: string, lastName?: string) :  
string {  
    if (lastName)  
        return firstName + " " + lastName;  
    else  
        return firstName;  
}
```

- ◆ Podrazumevani parametri:

```
function buildName(firstName: string, lastName =  
"Smith") : string {  
    return firstName + " " + lastName;  
}
```

TS funkcije - primer

```
function getMonth(month: number): string {  
    switch (month) {  
        case 1: return "January";  
        case 2: return "February";  
        case 3: return "March";  
        case 4: return "April";  
        case 5: return "May";  
        case 6: return "June";  
        case 7: return "July";  
        case 8: return "August";  
        case 9: return "September";  
        case 10: return "October";  
        case 11: return "November";  
        case 12: return "December";  
        default: return "Not a month!";  
    }  
}  
  
function logMonth(month: string): void {  
    console.log(getMonth(parseInt(month)));  
}
```

TS iteratori (iterators)

◆ **for ... in**

- Iterator koji vraća **ključeve** objekta nad kojim iterira

◆ **for ... of**

- Iterator koji vraća **vrednosti** objekta nad kojim iterira

```
let list = [4, 5, 6]
for (let i in list) {
    console.log(i); // '0','1','2'
}
for (let i of list) {
    console.log(i); // '4','5','6'
}
```

TS nabrojivi tip (enum)

- ◆ Omogućava definisanje skupa imenovanih konstanti
 - Analogno Java enum

```
const enum Direction { Up, Down, Left, Right, }
```

U primeru iznad Up dobija vrednost 0, Down vrednost 1, itd.

Nabrojivi tip radi po principu auto-inkrementa

```
let directions = [Directions.Up,  
                 Directions.Down,  
                 Directions.Left,  
                 Directions.Right]
```

Ručno prevođenje TS

- ◆ U Visual Studio Code razvojnom okruženju otvoriti neki od primera (na primer, Interface.ts)
- ◆ U Terminal prozoru se pozicionirati u direktorijum primera i kompajlirati TS klasu naredbom
tsc Interface.ts
- ◆ Rezultat prevođenja će biti JS fajl sa istim nazivom (Interface.js)
- ◆ JS fajl se može izvršiti iz Terminala naredbom
node Interface.js
- ◆ Uporediti izgled TS i JS fajla !
 - Za klasu (Class.ts), interfejs (Interface.ts) i funkciju (Function.ts)
 - **klasa (class) -> prototip (Prototype)**

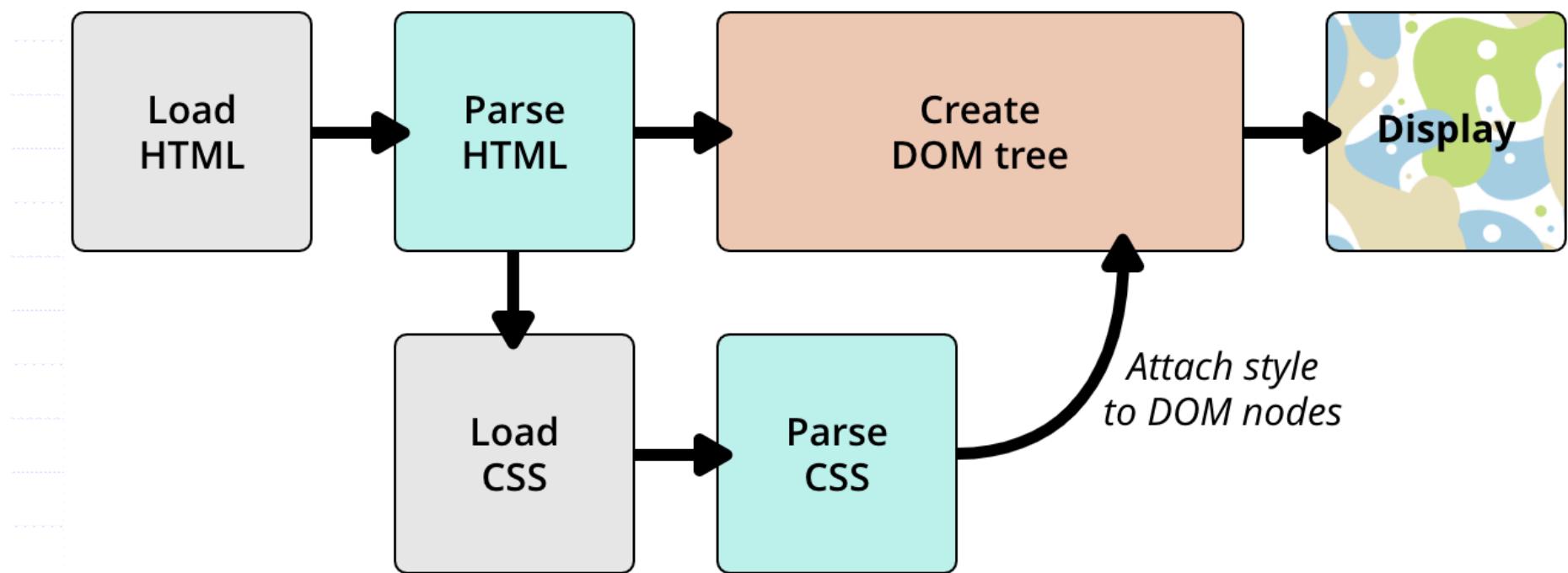
TypeScript sažetak i resursi

- ◆ Instalacija i korišćenje
- ◆ Tipovi podataka
- ◆ Deklaracija i definicija varijabli i funkcija
- ◆ Interfejsi
- ◆ Klase
- ◆ Prevođenje u JS (primeri)
 - TS Klasa -> JS Prototip
- ◆ Zvanična dokumentacija:
<https://www.typescriptlang.org/docs/handbook/basic-types.html> (handbook sekcije)

TypeScript materijali

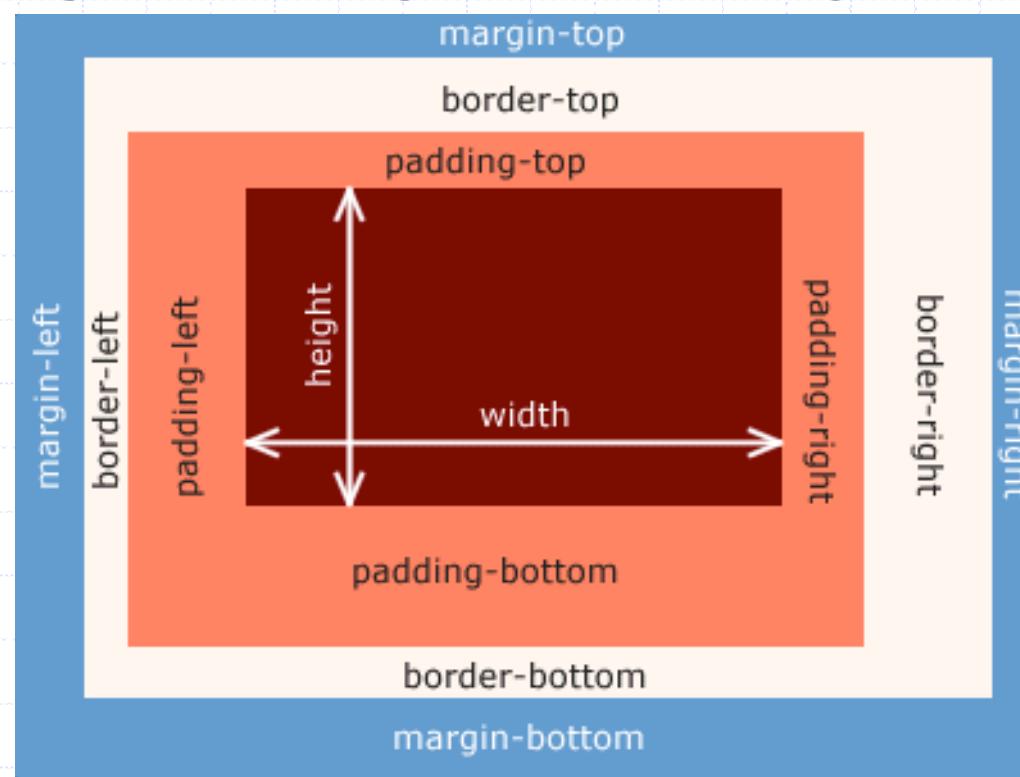
- ◆ Primer klase:
- ◆ Class.ts i Class.js (rezultat prevodenja TS klase u JS prototip)
- ◆ Primer interfejsa:
- ◆ Interface.ts i Interface.js (rezultat prevodenja TS interfejsa u JS funkciju)
- ◆ Primer funkcije:
- ◆ Function.ts i Function.js (rezultat prevodenja TS funkcije u JS funkciju)

Kako funkcioniše CSS ?



CSS box model

- ◆ **Margine** (margin) određuju prostor sa **spoljne strane** granica kontejnera
- ◆ **Pading** (padding) određuje prostor sa **unutrašnje strane** granica kontejnera



CSS veličine i boje

- ◆ height: visina elementa
- ◆ width: širina elementa
- ◆ max-height: maksimalna visina elementa
- ◆ max-width: maksimalna širina elementa
- ◆ min-height: minimalna visina elementa
- ◆ min-width: minimalna širina elementa

- ◆ color: frontalna boja elementa (tekst)
- ◆ background-color: pozadinska boja elementa

CSS fontovi

- ◆ font-family: "Helvetica Neue", "Arial", sans-serif;
- ◆ font-style: italic;
- ◆ font-weight: 100;
- ◆ font-size: 2em;
- ◆ letter-spacing: 1px

CSS jedinice mere

◆ **em:** skalabilna jedinica mere

- Na primer, 1em odgovara tekućoj veličini fonta. Ako je veličina fonta 12pt, $1\text{em}=12\text{pt}$, $2\text{em}=24\text{pt}$, $.5\text{em}=6\text{pt}$ i slično
- Popularna zbog skalabilnosti i responsive dizajna

◆ **px:** pikseli displeja računara kao fiksne jedinice

- Nije skalabilna jedinica – ne prilagođava se dimenzijama displeja

◆ **pt:** fiksna jedinica mere, $\text{pt}=1/72 \text{ inch}$

- Koristi se u štampi dokumenata

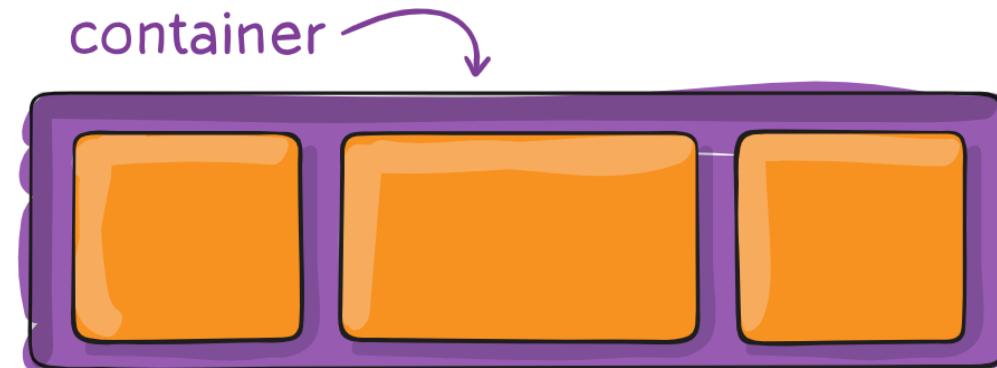
◆ **%:** skalabilna jedinica mere, sa tim što se modifikacije tekuće veličine izražavaju kao procenti

Šta je Flexbox ?

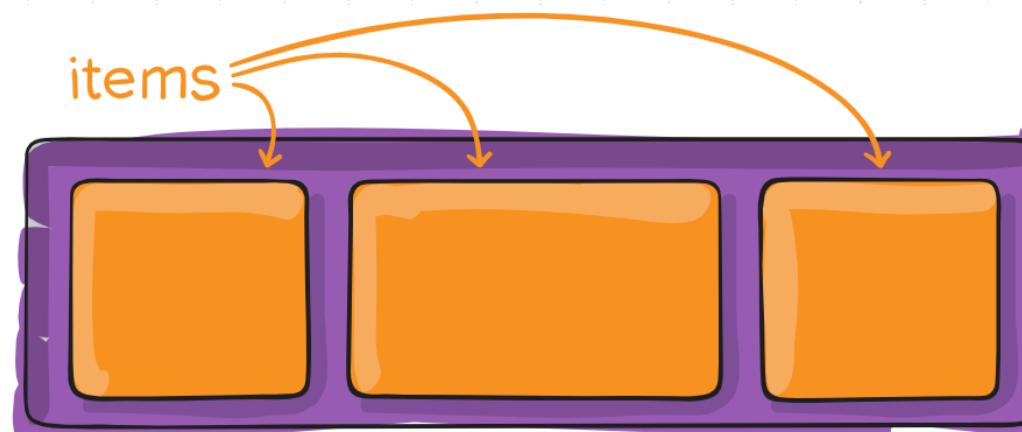
- ◆ Flexbox predstavlja W3C preporuku od Oktobra 2017 koja pruža efikasan načina za raspoređivanje, ravnanje i podelu prostora komponenti Web korisničkih interfejsa
- ◆ Čak i kada njihova veličina nije unapred poznata ili je dinamička (*flex*)
- ◆ Osnovne ideje:
- ◆ Mogućnost kontejnera da menja veličinu i raspored elemenata dece u cilju optimalnog iskorišćenja raspoloživog prostora ekrana
- ◆ Fleksibilnost u pogledu definisanja vizuelnog koordinatnog sistema za raspoređivanje komponenti
 - Block-layout je baziran na vertikalnom rasporedu
 - Inline-layout je baziran na horizontalnom rasporedu

Flexbox - osnovne komponente

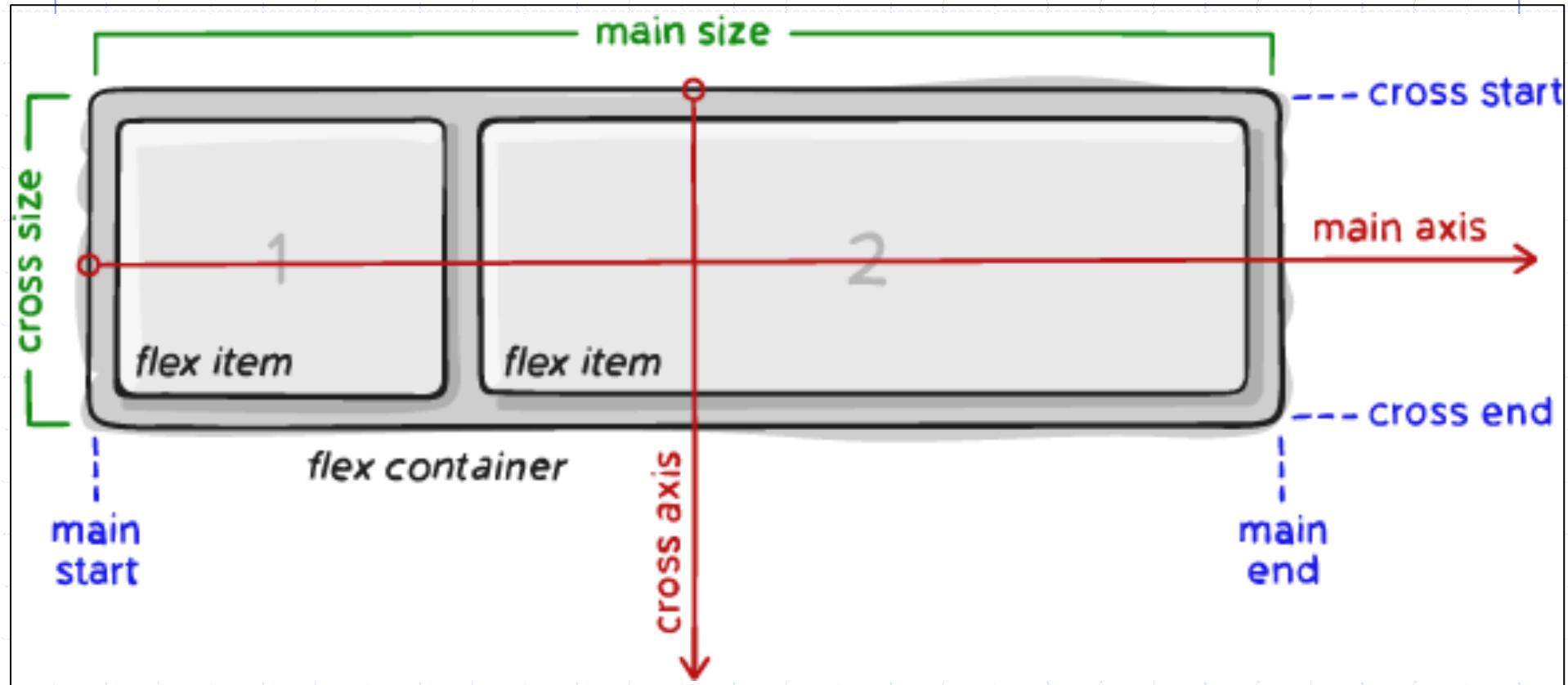
- ◆ Roditelj (*container*)



- ◆ Dete (*item*)



Flexbox – elementi rasperda



Flexbox – elementi rasporeda

- ◆ Deca (items) se raspoređuju po glavnoj osi (**main-axis**) ili upravnoj osi (**cross-axis**)
- ◆ Glavna osa je **podrazumevano horizontalna** i deca se raspoređuju sa leva (**main-start**) na desno (**main-end**)
 - Orijentacija se može menjati programski (**flex-direction**)
- ◆ Upravna osa je uvek normalna na glavnu osu. Njena orijentacija zavisi od orijentacije glavne ose
- ◆ Deca se (podrazumevano) raspoređuju odozgo (**cross-start**) na dole (**cross-end**)
- ◆ Raspored se definiše pomoću CSS naredbi, tj. direktiva



Direktive rasporeda roditeljskih elemenata

Roditeljski element - **display**

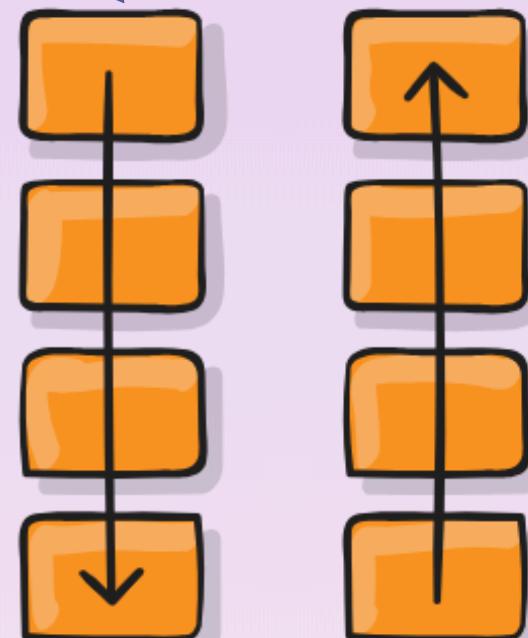
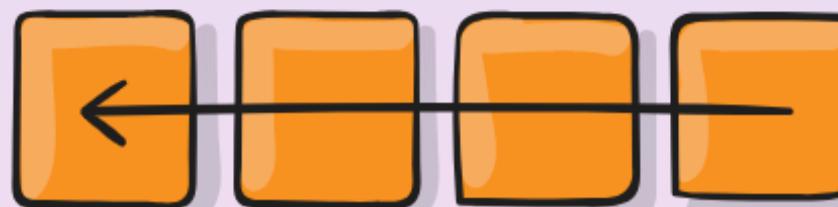
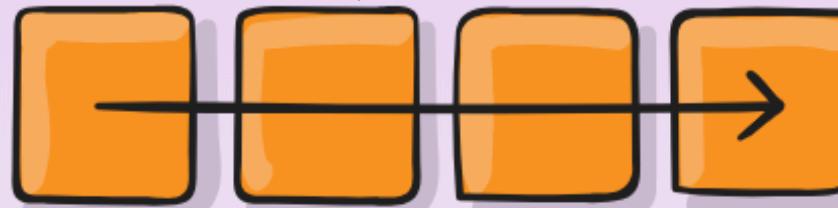
- ◆ Proglašava roditelja Flexbox kontejnerom
- ◆ Podrazumevan *raspored* dece:
- ◆ Jeden red, sa leva na desno, sa mogućim preklapanjem ukoliko nema prostora po glavnoj osi
- ◆ Deca ne povećavaju veličinu (tj. širinu) po glavnoj osi
- ◆ Deca mogu povećati veličinu (tj. visinu) po upravnoj osi u slučaju raspoloživog prostora

```
.container {  
    display: flex; /* or inline-flex */  
}
```

Roditeljski element – **flex-direction**

- ◆ Definiše orijentaciju i smer glavne ose (**main-axis**) na osnovu kojih se raspoređuju deca

```
.container {  
    flex-direction: row | row-reverse | column | column-reverse;  
}
```



Roditeljski element – **flex-wrap**

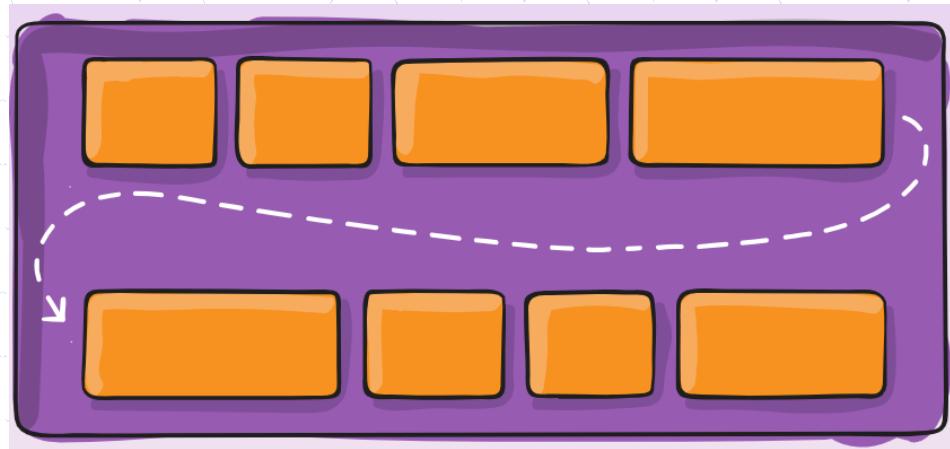
- ◆ Elementi deca se pokušavaju smestiti u jedan red
- ◆ Direktiva omogućava dodatnu fleksibilnost
- ◆ Definiše redosled dece po horizontali, tj. da li se smeštaju u jedan red ili se mogu nastavljati u novim redovima

.container{ flex-wrap: nowrap | wrap | wrap-reverse; }

nowrap: podrazumevano; deca se smeštaju u jedan red

wrap: deca se smeštaju u više redova.

wrap-reverse: deca se smeštaju u više redova, ali u obrnutom redosledu (od dna ka vrhu)



Roditeljski element – justify-content

- ◆ Definiše ravnjanje po glavnoj osi (main-axis) i raspodelu prostora između dece

```
.container {  
justify-content: flex-start |  
           flex-end |  
           center |  
          baseline |  
         stretch;  
}
```

Ravnjanje u odnosu na početak i kraj ose sa ravnomernim međuprostorom

Svako dete dobija jednaku porciju slobodnog prostora oko sebe

Udaljenost od početka/kraja ose i međuprostori između dece su jednaki

flex-start



flex-end



center



space-between



space-around



space-evenly



Roditeljski element – justify-content

- ◆ Primer:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox#justify-content

Roditeljski element – align-items

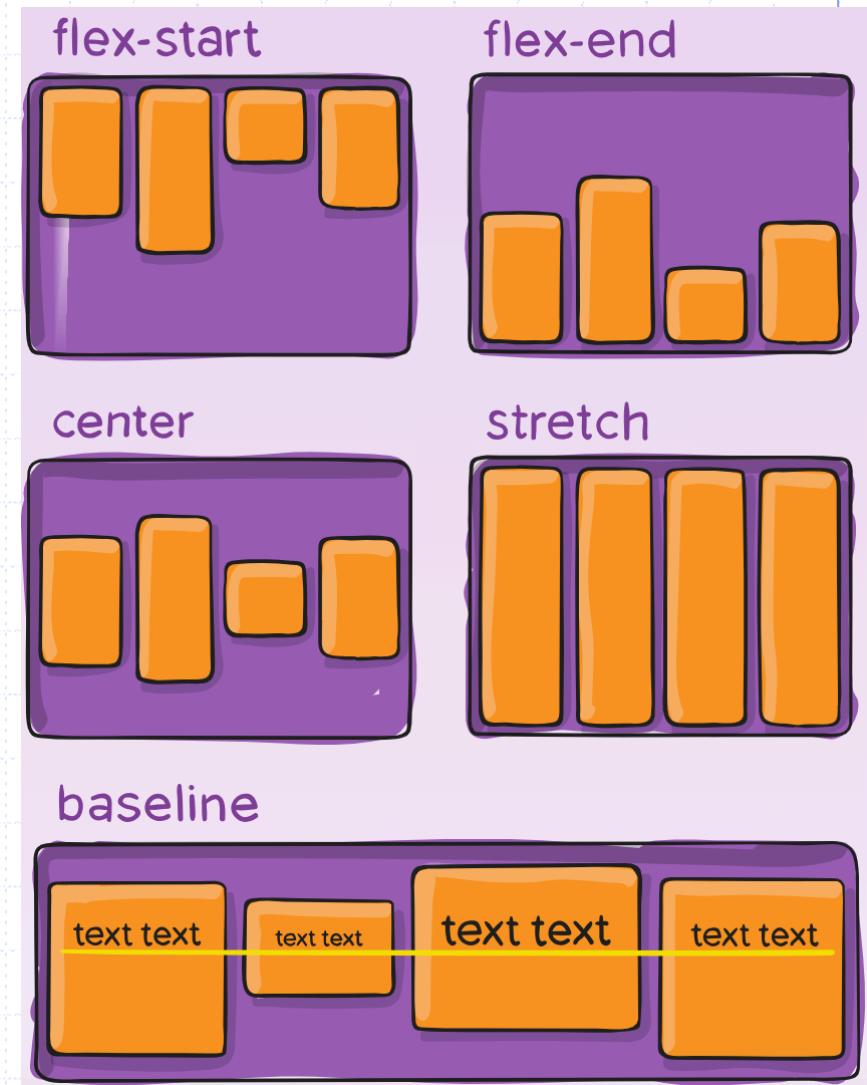
- ◆ Definiše ravnanje po upravnoj osi (cross-axis) i raspodelu slobodnog prostora između dece

```
.container {  
    align-items: flex-start |  
                flex-end |  
                center |  
                baseline |  
                stretch;  
}
```

stretch: podrazumevano
ponašanje; deca mogu redefinisati
svoju max veličinu

center: centriranje po upravnoj osi

baseline: ravnanje u odnosu na osu
koja predstavlja osnovnu liniju
teksta elemenata dece



Roditeljski element – align-items

◆ Primer:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox#align-items

Roditeljski element – align-content

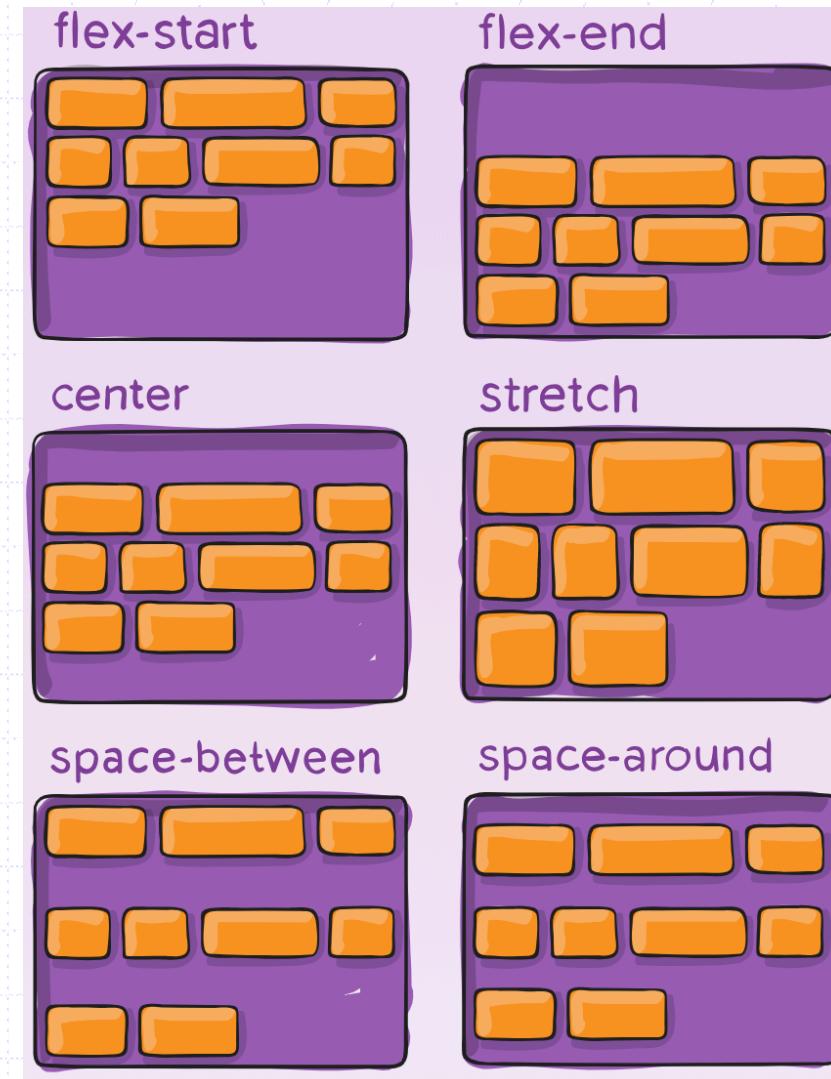
- ◆ Definiše ravnanje grupa, tj. linija dece po upravnoj osi (cross-axis) - samo u slučaju **više redova** dece

```
.container {
```

```
  align-content: flex-start |  
    flex-end |  
    center |  
    space-between |  
    space-around |  
    stretch;
```

```
}
```

stretch: podrazumevano ponašanje; dimenzije dece-redova su identične; deca mogu redefinisati svoju max veličinu
space-between: ravnomeran međuprostor sa ravnanjem uz granice
space-around: ravnomeran međuprostor uključujući i granice



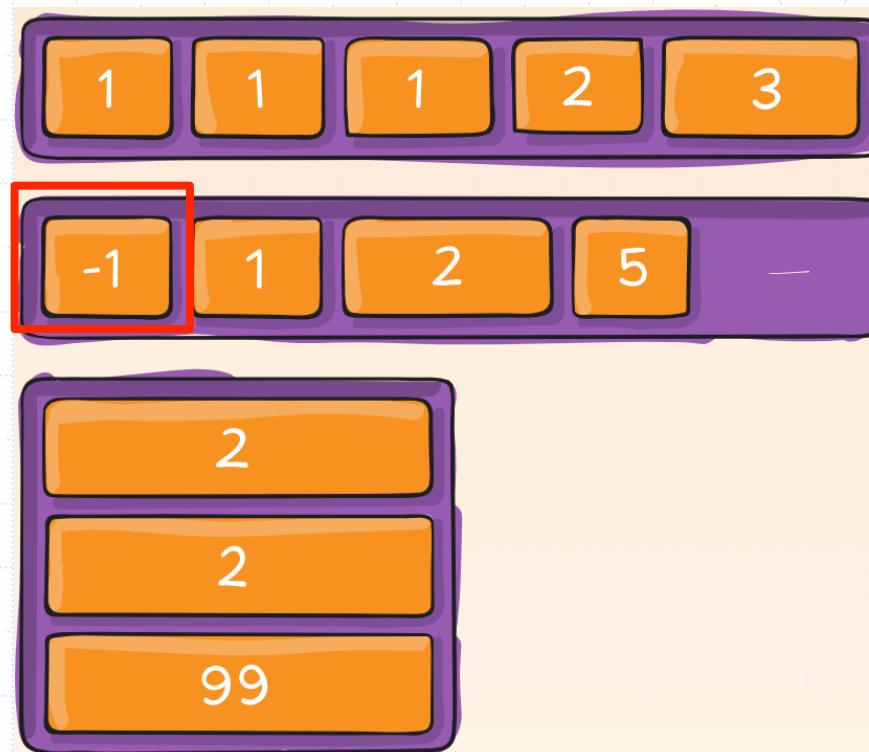


**Direktive rasporeda
elemenata dece**

Element dete – order

- ◆ Deca se podrazumevano prikazuju redosledom kojim su definisana
- ◆ Definiše redosled prikaza deteta u roditelju

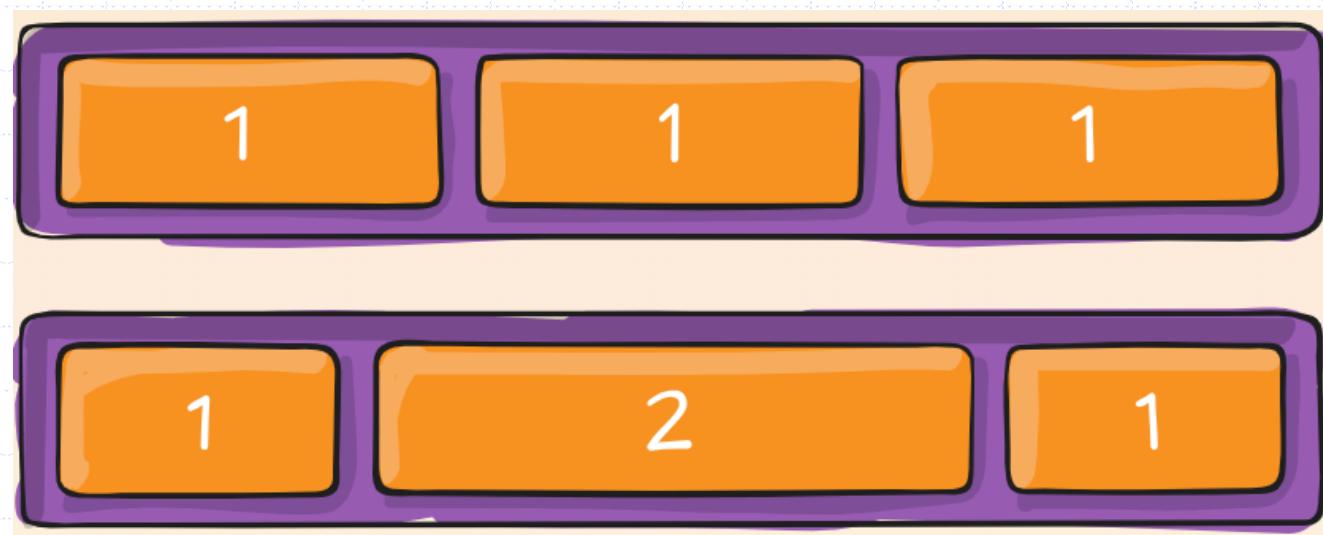
```
.item { order: <integer>; /* default = 0 */ }
```



Element dete – **flex-grow**

- ◆ Mogućnost deteta da povećava svoju veličinu u odnosu na glavnu osu (**main-axis**) u zavisnosti od slobodnog prostora
- ◆ Prihvata broj koji služi kao proporcija – udeo slobodnog prostora kontejnera koji dete može zauzeti u odnosu na drugu decu
- ◆ Podrazumevano je 0 (nema povećana veličine)

```
.item { flex-grow: <pozitivan broj>; /* default = 0 */ }
```



Element dete – **flex-shrink**

- ◆ Mogućnost deteta da smanjuje svoju veličinu u odnosu na glavnu osu (**main-axis**) u slučaju smanjenja prostora
- ◆ Prihvata broj koji služi kao proporcija – veća vrednost broja znači da će se element više smanjivati u odnosu na susedne elemente
- ◆ Minimalna veličina deteta se uzima u obzir ukoliko je zadata
- ◆ Podrazumevano je 0 (nema smanjenja veličine)
- ◆ U praksi se obično zadaje ista vrednosti kako bi se veličine susednih elemenata smanjivale ujednačeno

```
.item { flex-shrink: <pozitivan broj>; /* default = 0 */ }
```

Element dete – **flex-basis**

- ◆ Definiše podrazumevanu veličinu deteta po glavnoj osi (**main-axis**) tj. širini
- ◆ Vrednost može biti broj ili ključna reč
- ◆ Preduslov za primenu dve prethodne direktive, tj, da li će se veličina elementa povećavati (**flex-grow**) ili smanjivati (**flex-shrink**)
- ◆ Podrazumevena vrednost je **auto** što znači da će biti konsultovana veličina deteta ako je zadata
- ◆ Ako veličina nije zadata, veličina sadržaja deteta će definisati prostor koji ono zauzima

```
.item { flex-basis: <širina> | auto; }
```

Skraćeni oblik za više direktiva - flex

- ◆ Kombinuje povećanje veličine (**flex-grow**), smanjenje veličine (**flex-shrink**) i podrazumevanu veličinu (**flex-basis**)
- ◆ Vrednosti se navode po redosledu iznad
- ◆ Mogu se koristiti i simboličke konstante kao ispod

flex: initial = **flex: 0 1 auto** (podrazumevano)

flex: auto = **flex: 1 1 auto**

flex: none = **flex: 0 0 auto**

Skraćeni oblik za više direktiva - flex

- ◆ Primer:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox#Shorthand_values_for_the_flex_properties

Flexbox - sažetak

- ◆ Flexbox naredbe se navode u stilu Angular komponente (css datoteka)
- ◆ Princip korišćenja:
- ◆ Definisati klase stila u css datoteci
 - Naredbe koje se odnose na roditelja (.container)
 - Naredbe koje se odnose na decu (.item)
- ◆ Klase dodeliti kontejnerskim elementima templejta komponente (div, section)
 - `div class="container"` ili `div class="item"`

Materijal pripremljen na osnovu:

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Flexbox resursi

- ◆ Flexbox tutorijal:

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

- ◆ Flexbox MDN tutorijal:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox

- ◆ Flexbox sa definisanje globalnog layout-a Web sajta:

<https://www.sitepoint.com/flexbox-css-flexible-box-layout/>

Flexlayout - sadržaj

- ◆ Flexlayout:
- ◆ Definicija
- ◆ Terminologija
- ◆ Roditeljski elementi stila
- ◆ Elementi stila dece

Šta je Flexlayout ?

- ◆ Angular biblioteka koja sadrži HTML direktive (ili atribute) za definisanje fleksibilnih **rasporeda** elemenata interfejsa
- ◆ Predstavlja nadogradnju Flexbox-a koje je intuitivnija za korišćenje:
- ◆ Umesto upotrebe CSS stilova i sintakse, programer definiše raspored direktno u templejtu Angular komponente
- ◆ Slično Flexbox-u, postoje direktive templejta za roditeljske elemente (kontejneri) i direktive za elemente decu

Zbog čega Flexlayout ?

- ◆ Nastao kao potreba zajednice programera Web korisničkih interfejsa da definišu **fleksibilan** i **dinamički** raspored elemenata direktno u HTML dokumentu
- ◆ Fleksibilan: prilagodljiv displejima različitih dimenzija tako da obezbedi podjednako kvalitetan osećaj korišćenja za različite korisnike
- ◆ Dinamički: fleksibilan u vreme korišćenja, tj. izvršavanja aplikacije

Flexlayout korišćenje i paketi

- ◆ Flexlayout direktive se instaliraju kao dodatna biblioteka na nivou Angular projekta

```
npm install @angular/flex-layout --save
```

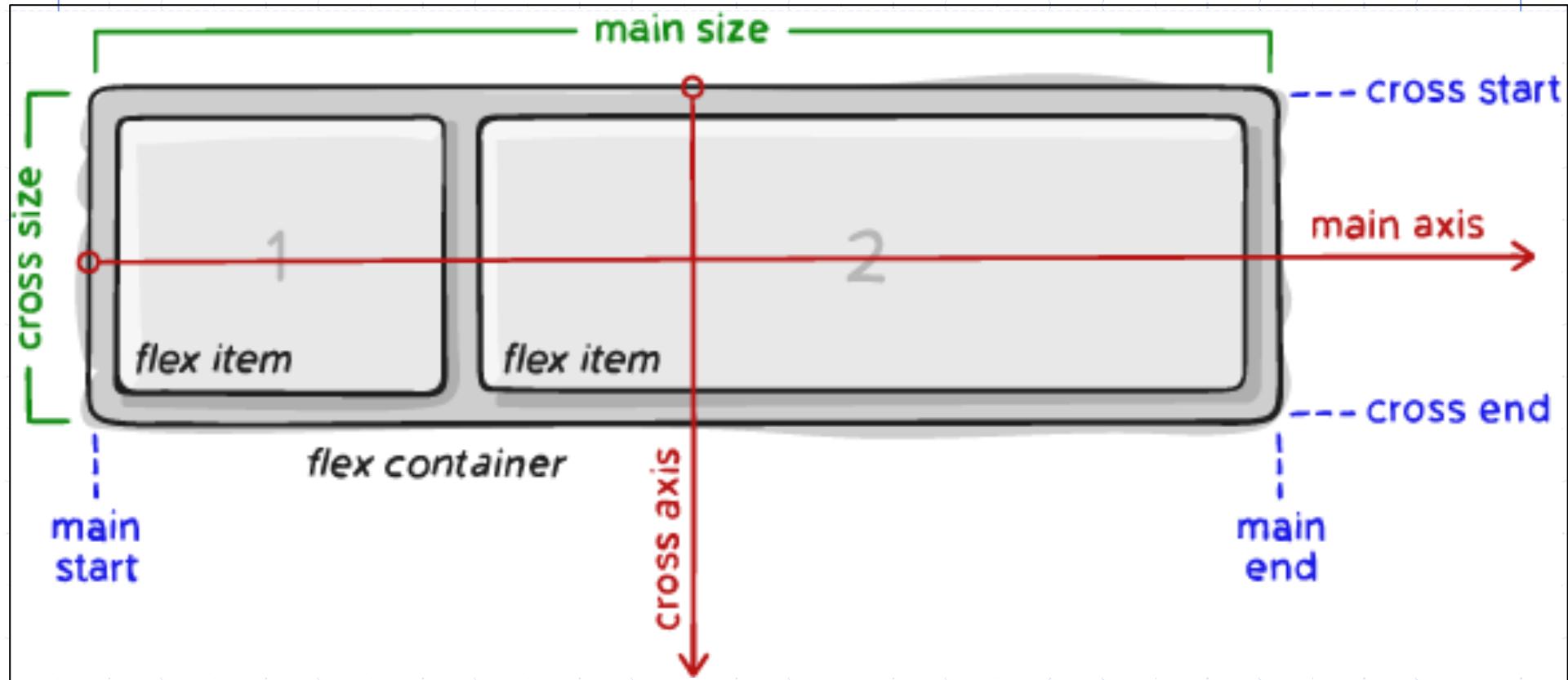
@angular/flex-layout biblioteka:

- ◆ **Static API**
- ◆ **Responsive API**

package.json

```
"dependencies": {  
  "@angular/animations": "^7.0.0",  
  "@angular/cdk": "^7.0.0",  
  "@angular/common": "^7.0.0",  
  "@angular/compiler": "^7.0.0",  
  "@angular/core": "^7.0.0",  
  "@angular/flex-layout": "7.0.0-beta.19",  
}
```

Flexlayout elementi raspoređeni su nasleđeni od Flexbox-a

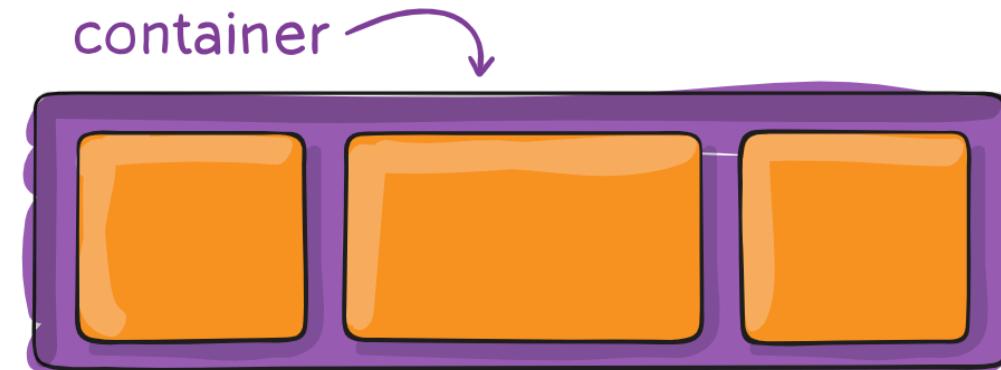


Flexlayout – elementi rasporeda

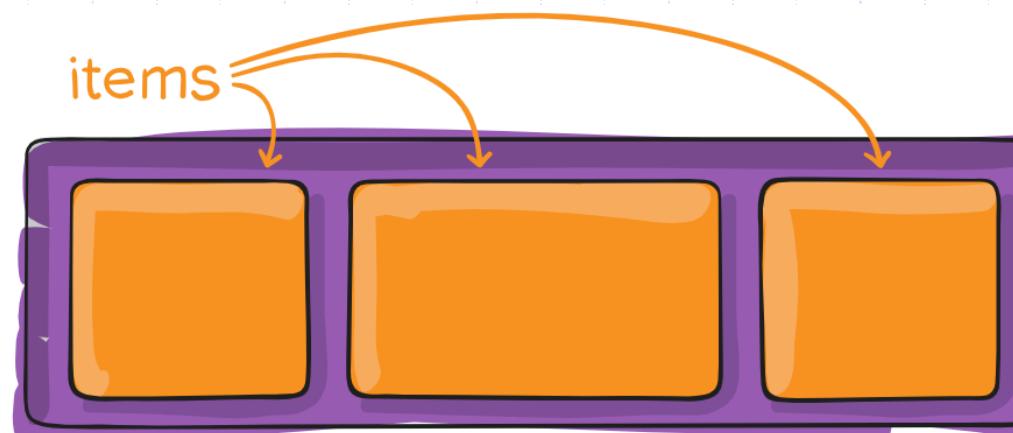
- ◆ Deca (items) se raspoređuju po glavnoj osi (**main-axis**) ili upravnoj osi (**cross-axis**)
- ◆ Glavna osa je **podrazumevano horizontalna** i deca se raspoređuju sa leva (**main-start**) na desno (**main-end**)
 - Orijentacija se može menjati programski (**flex-direction**)
- ◆ Upravna osa je uvek normalna na glavnu osu.
Njena orijentacija zavisi od orijentacije glavne ose
- ◆ Deca se (podrazumevano) raspoređuju odozgo (**cross-start**) na dole (**cross-end**)
- ◆ Raspored se definiše pomoću naredbi ili direktiva templejta

Osnovne komponente

◆ Roditelj (*container*)



◆ Dete (*item*)



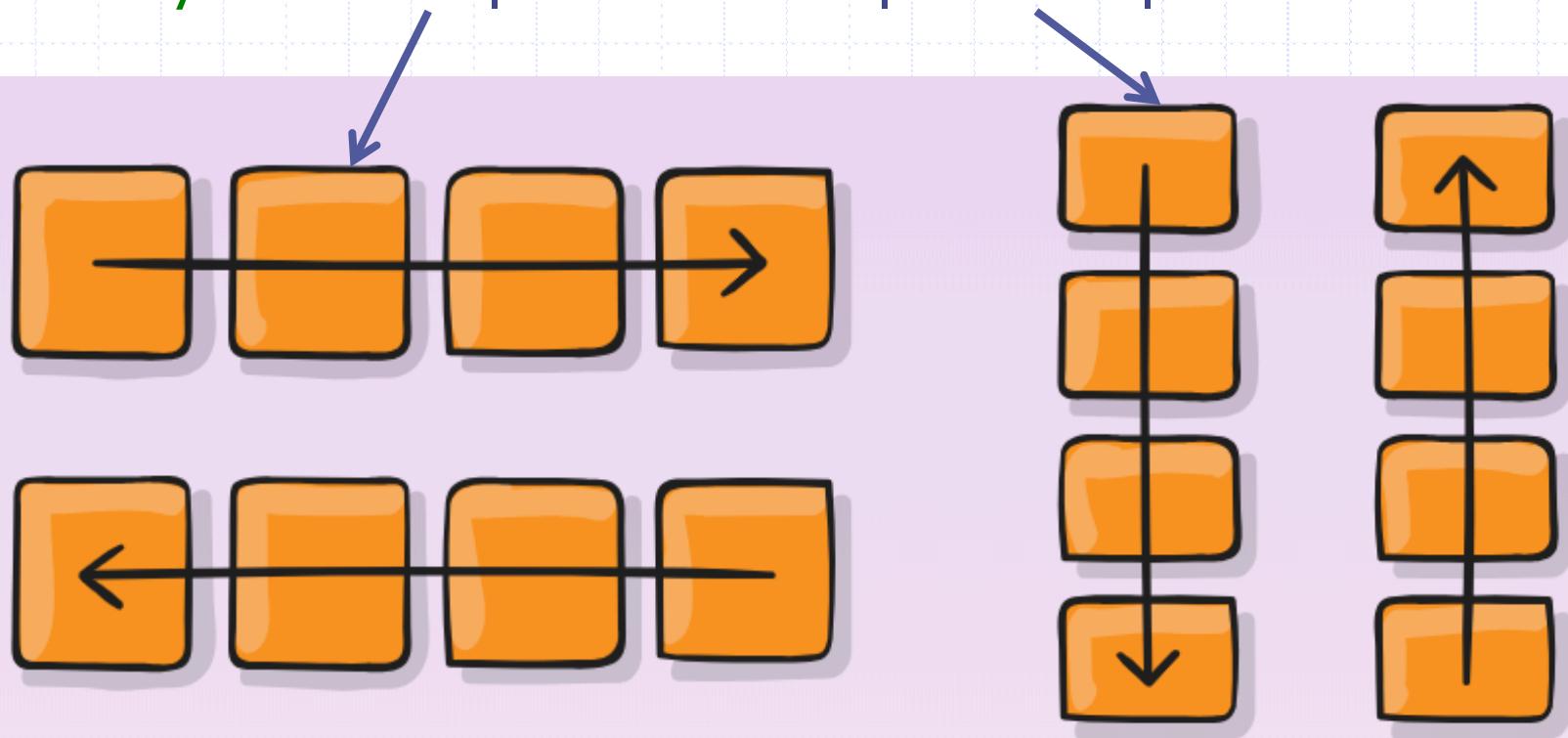


Direktive rasporeda roditeljskih elemenata

Roditeljski element – **fxLayout**

- ◆ Definiše orijentaciju i smer glavne ose (**main-axis**) na osnovu kojih se raspoređuju deca
- ◆ Flexbox ekvivalent: **flex-direction**

```
<div fxLayout="row | row-reverse | column | column-reverse">
```



fxLayout primer 1/2

```
<div fxLayout="row">  
  <div>1. One</div>  
  <div>2. Two</div>  
  <div>3. Three</div>  
  <div>4. Four</div>  
</div>
```

1. One

2. Two

3. Three

4. Four

fxLayout primer 2/2

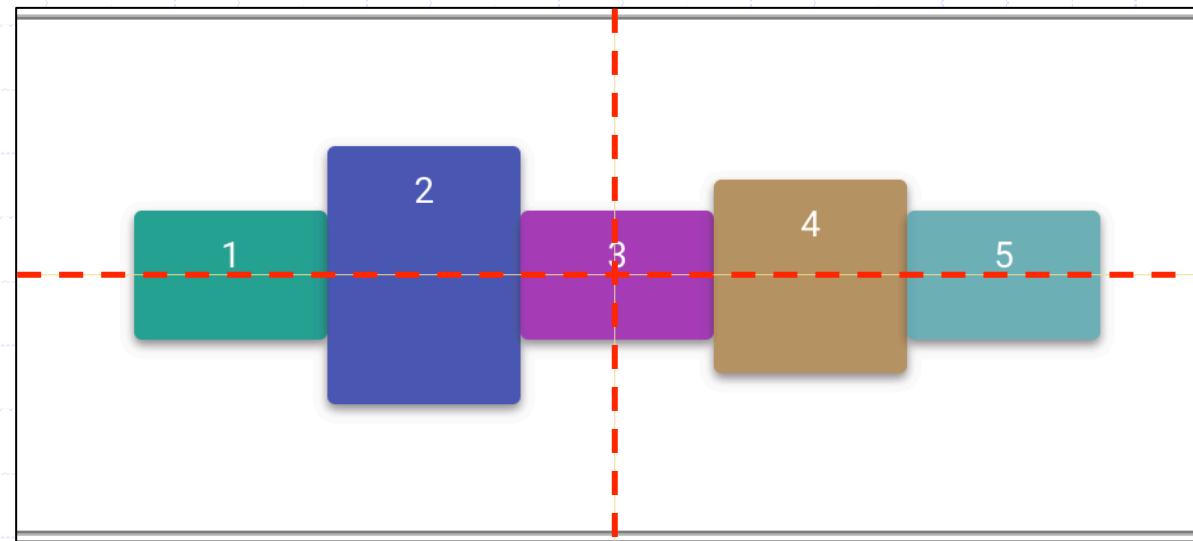
```
<div fxLayout="column">  
  <div>1. One</div>  
  <div>2. Two</div>  
  <div>3. Three</div>  
  <div>4. Four</div>  
</div>
```



Roditeljski element – **fxLayoutAlign**

- ◆ Definiše ravnjanje elemenata dece po glavnoj osi (**main-axis**) i opcionalno upravnoj osi (**cross-axis**)
- ◆ Prva vrednost koja se navodi predstavlja ravnjanje po glavnoj osi
- ◆ Druga vrednost koja se navodi predstavlja ravnjanje po upravnoj osi (opcionalno)

```
<div fxLayout="row" fxLayoutAlign="center center">...</div>
```



Roditeljski element – fxLayoutAlign

◆ Flexbox ekvivalent:

justify-content

1. vrednost – glavna osa

flex-start



flex-end



center



space-between



space-around



space-evenly

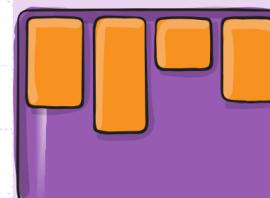


align-items

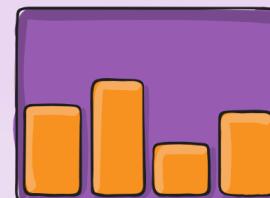
align-content

2. vrednost – upravna osa

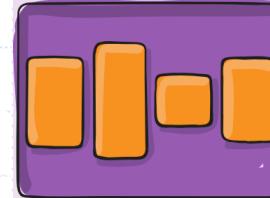
flex-start



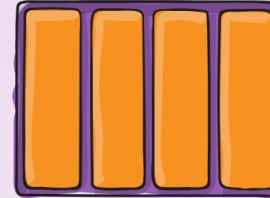
flex-end



center



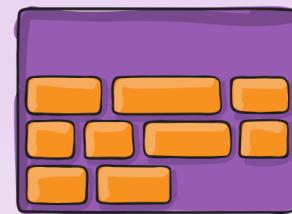
stretch



flex-start



flex-end



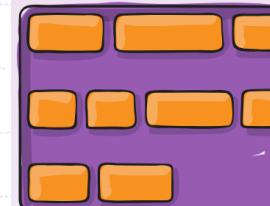
center



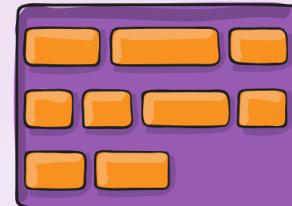
stretch



space-between



space-around



fxLayoutAlign primeri – nazivi i značenja ravnanja su identični kao za Flexbox !

Layout Children with 'layout-align'

1

2

3

4

5

Layout Direction

- row
- column

Alignment in Layout Direction (horizontal)

- none
- start (default)
- center
- end
- space-around
- space-between
- space-evenly

Alignment in Perpendicular Direction (vertical)

- none
- start
- center
- end
- stretch (default)

```
<div fxLayout="row" fxLayoutAlign="space-around center" >
```

Roditeljski element – **fxLayoutGap**

- ◆ Definiše **margin** oko elemenata dece kontejnera
- ◆ Koje margin se dodaju zavisi od orijentacije glavne ose:

`fxLayout="row" => horizontalne margine`

`fxLayout="column" => vertikalne margine`

Jedinice mere:

%

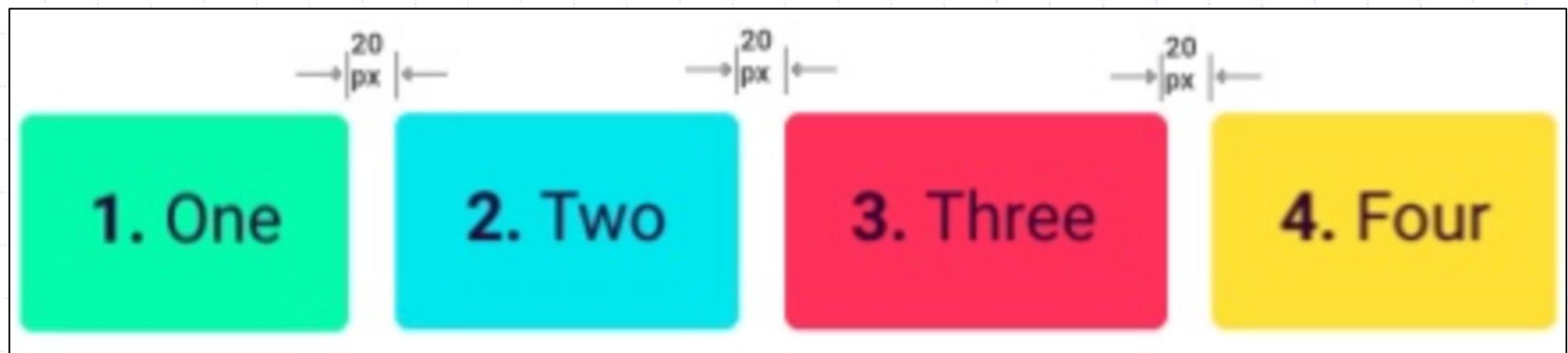
px

$1 \text{ vw} = 1\% \text{ širine prozora pretraživača (viewport width)}$

$1 \text{ vh} = 1\% \text{ visine prozora pretraživača (viewport height)}$

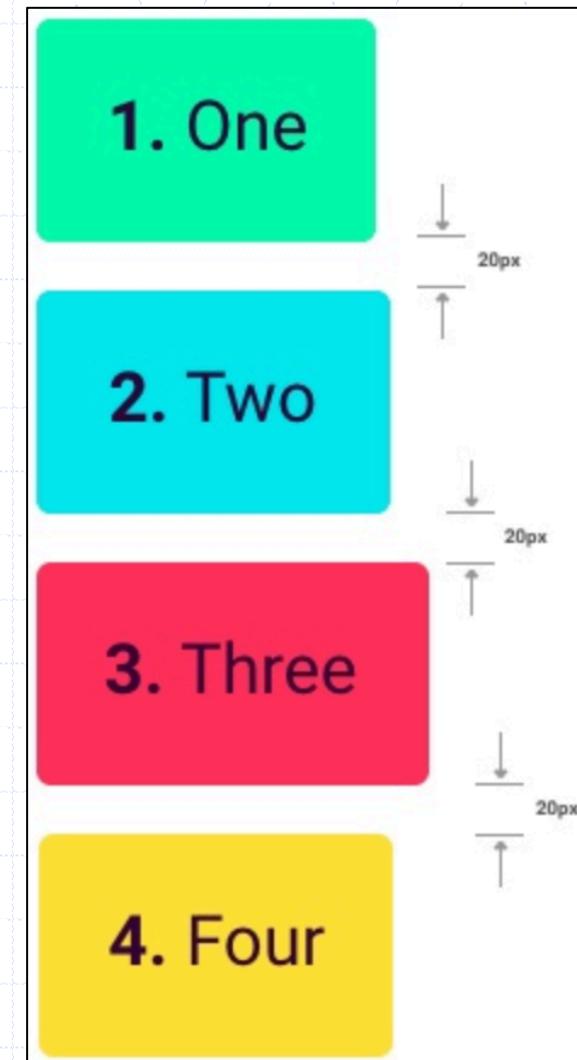
fxLayoutGap primer 1/2

```
<div fxLayout="row" fxLayoutGap="20px">  
  <div>1. One</div>  
  <div>2. Two</div>  
  <div>3. Three</div>  
  <div>4. Four</div>  
</div>
```



fxLayoutGap primer 2/2

```
<div fxLayout="column" fxLayoutGap="20px">  
  <div>1. One</div>  
  <div>2. Two</div>  
  <div>3. Three</div>  
  <div>4. Four</div>  
</div>
```





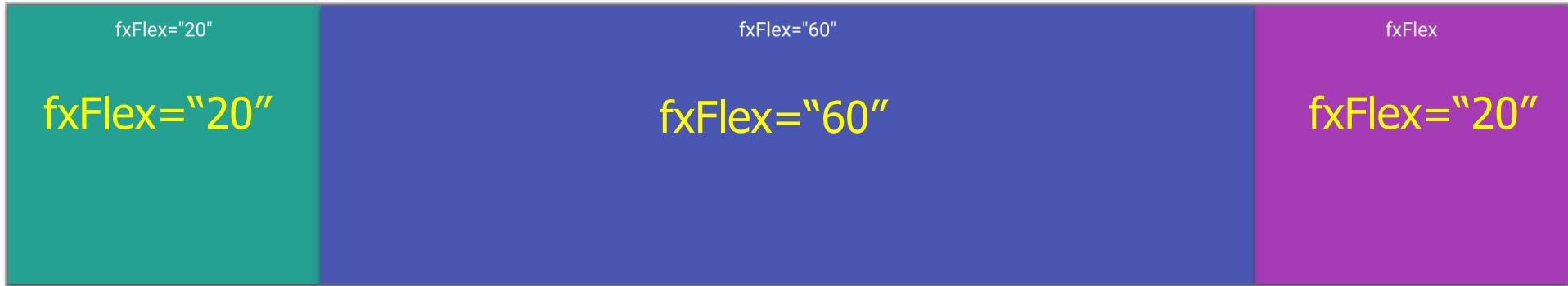
**Direktive rasporeda
elemenata dece**

Element dete – **fxFlex** 1/2

- ◆ Definiše promenu veličine elementa u kontejneru u odnosu na glavnu osu (**main-axis**) kao podrazumevanu veličinu, mogućnost povećanja veličine, i mogućnost smanjenja veličine
- ◆ Flexbox ekvivalent: **flex-grow**
- ◆ **Skraćena notacija:** navodi se jedna vrednost koja definiše podrazumevanu veličinu elementa i ravnomerno povećanje/smanjenje veličine
 - Ukoliko se ne navede vrednost, dete će zauzeti sav raspoloživ prostor po glavnoj osi roditelja
 - Ukoliko više elemenata dece navede direktivu bez vrednosti, ravnomerno će podeliti raspoloživ prostor
- ◆ Puna notacija: navode se tri vrednosti kao povećanje (**grow**), smanjenje (**shrink**) i podrazumevana veličina (**basis**)
- ◆ Jedinice mere: % px
 - 1 vw = 1% širine prozora pretraživača (viewport width)
 - 1 vh = 1% visine prozora pretraživača (viewport height)

Element dete – **fxFlex** 2/2

<div fxLayout="row">



<div fxLayout="column">



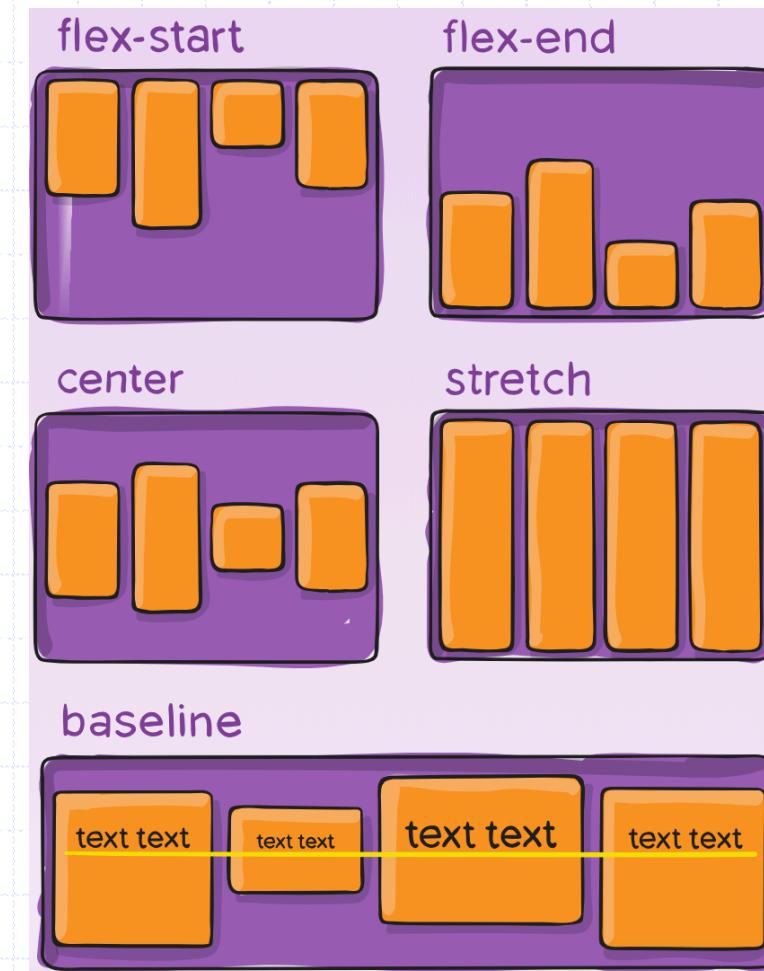
fxFlex predefinisane vrednosti (alias)

alias	Flexbox (css) značenje
grow	<code>flex: 1 1 100%</code>
none	<code>flex: 0 0 auto</code>
nogrow	<code>flex: 0 1 auto</code>
noshrink	<code>flex: 1 0 auto</code>

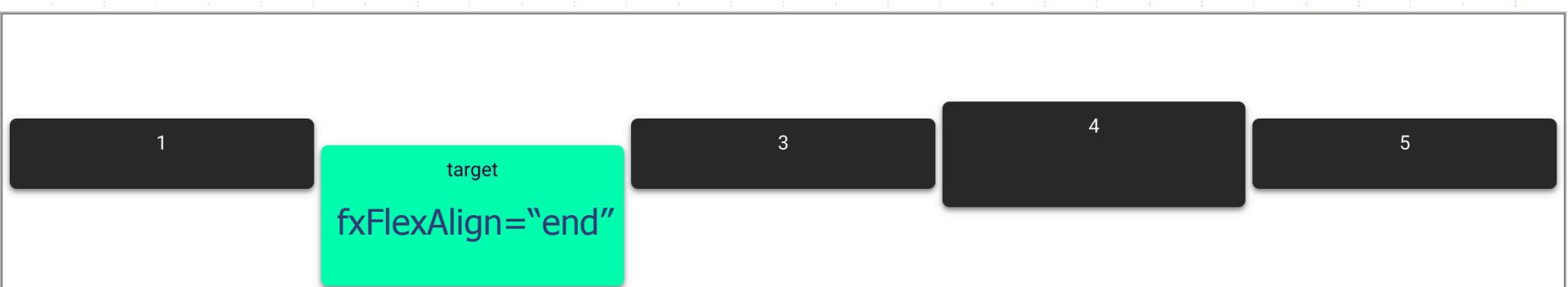
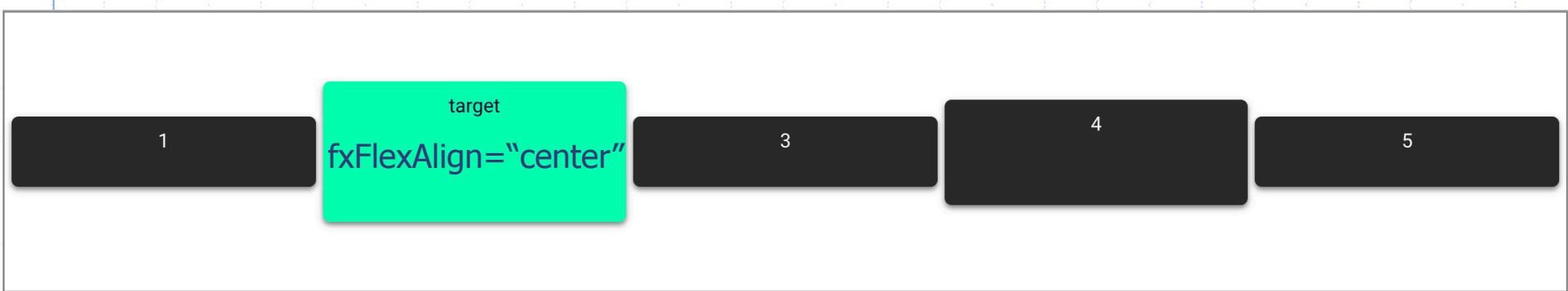
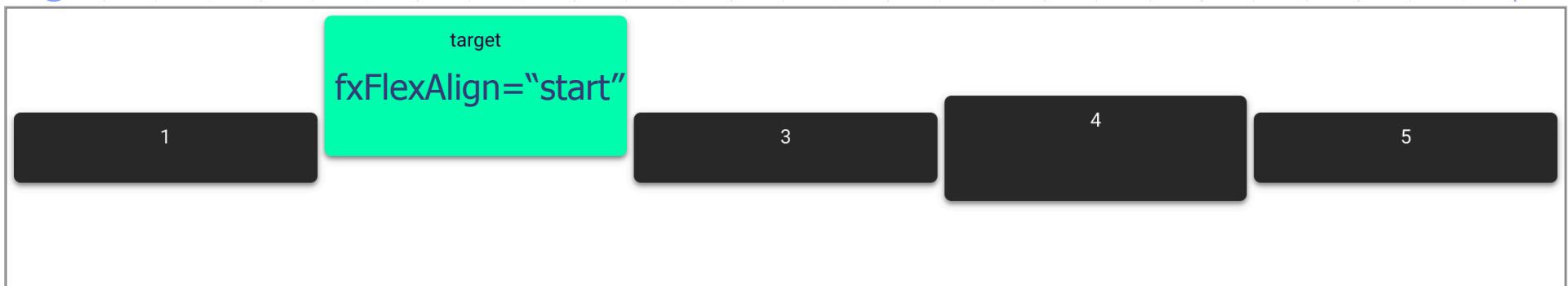
Element dete - **fxFlexAlign**

- ◆ Ravnanje na nivou deteta, redefiniše vrednost ravnjanja roditelja po uporednoj osi (**cross-axis**)

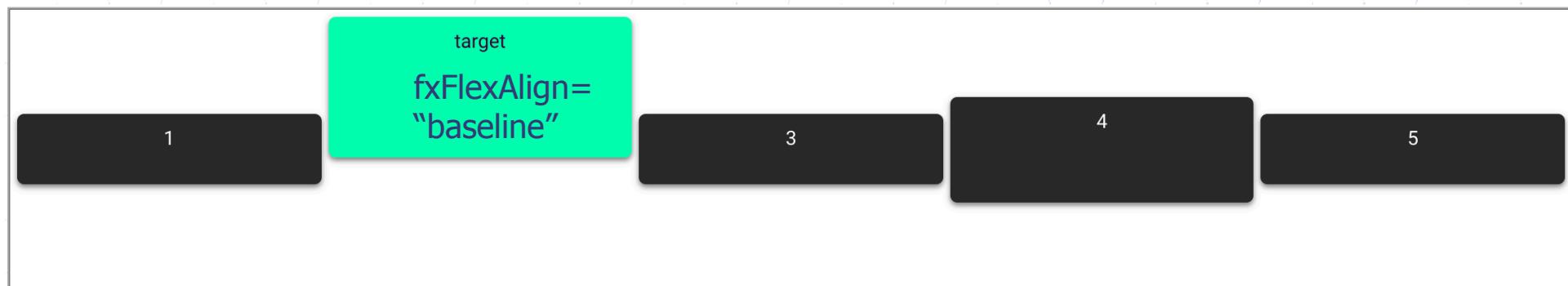
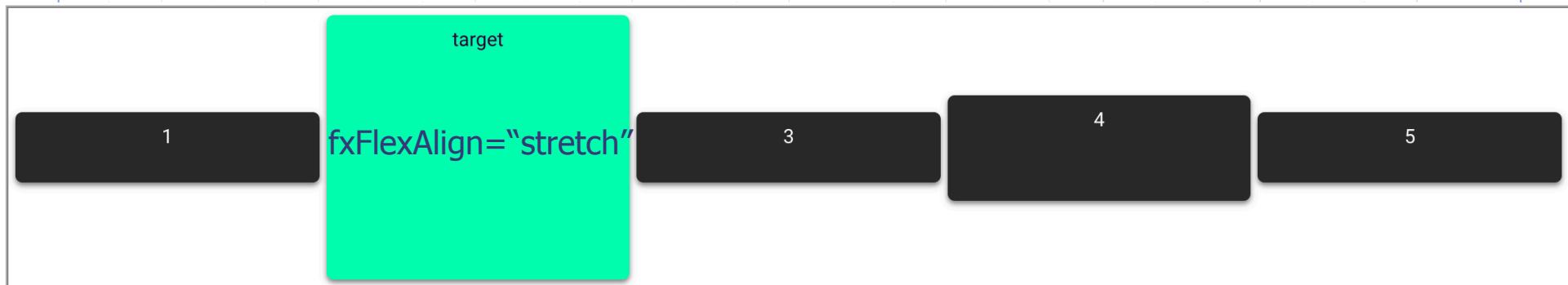
- ◆ Vrednosti:
 - start (= flex-start)
 - end (= flex-end)
 - center
 - baseline
 - stretch



fxFlexAlign primeri 1/2



fxFlexAlign primeri 2/2



Flexlayout - sažetak

- ◆ Flexlayout naredbe (direktive) se navode u templejtu Angular Material komponente (html datoteka)
 - Princip rada je identičan kao Flexbox
- ◆ Naredbe koje se odnose na roditelja (kontejner)
- ◆ Naredbe koje se odnose na decu

Materijal pripremljen na osnovu:

<https://github.com/angular/flex-layout/wiki/Declarative-API-Overview>

Flexlayout resursi

- ◆ Zvanična dokumentacija:

[https://github.com/angular/flex-layout/wiki/
Declarative-API-Overview](https://github.com/angular/flex-layout/wiki/Declarative-API-Overview)

- ◆ Online demo editor:

[https://tburleson-layouts-demos.firebaseio.com/#/
docs](https://tburleson-layouts-demos.firebaseio.com/#/docs)

TypeScript Flexbox Flexlayout

Dr. Mlađan Jovanović
mjovanovic@singidunum.ac.rs