

19. JS nizovi i f-je za njihovu obradu

JS nizovi se mogu kreirati na 2 načina:

1º let prvi = [1,2,3];

2º let drugi = new Array(1,2,3);

U oba slučaja je rezultat novi niz i ne postoji nikakva razlika. Niz može da se inicijalizuje na sledeći način;

```
let prvi = [42, "Web", null, 168, "Levo"];
```

```
let drugi = new Array("Desno", 52, null);
```

Broj elemenata niza može se dobiti korišćenjem atributa LENGTH.

```
console.log(drugi.length);
```

let prvi = [7]; → kreira se niz sa 1. elementom (sedmicom)

let drugi = new Array[7]; → kreira se prazan niz dužine 7

Ova dva niza NISU ista

prvi.push("new", 163); → prvi posle dodavanja ima 3 elementa

drugi.push("new", 163); → drugi posle dodavanja ima 9 elemenata

let prvi = [7];
let drugi = new Array(7);
prvi[5] = "new"; → ima 6 elemenata, ✓
drugi[5] = "new"; i šesti je jednak "new"
→ ima i dalje 7 elemenata,
šesti je jednak "new", a
ostali su prazni

U JS-u se mogu napraviti i asocijativni
nizovi, pri čemu se konisti modifikovana for
petlja

let treći = [];
treći["a"] = 16;
treći["b"] = 44;
treći[4] = 12;
treći[3] = "new";

```
for (var key in treći)  
    console.log(treći[key]);
```

Ne može treći[i] klasično
Prvo se ispisuju svi elementi kod
kojih je indeks intijer (od 0
do length-1), a zatim ostali kojima
je indeks string (po redosledu
ubaciranja u niz) ne po
abecednom
redosledu

19. JS nizovi i ~~f~~-je za njihov Obradu

Nad nizovima se primenjuju različite metode kao npr.:

→ **forEach**

```
let cetvrti = [54, 17, "x", 666, undefined, "yyz"];  
cetvrti.forEach(function(vrednost, indeks){  
    console.log(vrednost, indeks);  
});
```

54 0
17 1
x 2

→ **map (povratna vrednost je niz)**

```
let cetvrti = [54, 17, "x", 666, null, "yyz"];  
let peti = cetvrti.map(x => "x" + x);  
peti = ["x54", "x17", "xx", "x666", "xnull", "xyyz"];
```

→ **filter (povratna vrednost je niz)**

```
let peti = cetvrti.filter(x => (x > 20));  
peti = [54, 666];
```

→ **reduce (povratna vrednost nije niz)** → ^{niz pretvara}
^{v jednu vrednost}

```
let cetvrti = [1, 2, 3, 4, 5, 6, 7];
```

```
let proizvodi = cetvrti.reduce((acc, x, indeks) => {
```

```
    console.log(indeks);
```

```
    return acc * x;
```

```
y, 1);
```

Pocetna vrednost acc

proizvodi = 5040;

$1 * 2 * 3 * 4 * 5 * 6 * 7$

Izvršava funkciju u argumentima.

```
function sum(acc, x){  
    return acc+x;  
}
```

let suma=cetvrti.reduce(sum); $\xrightarrow{\frac{28}{1}}$

Nad svakim elementom niza se primjenjuje prosleđena f-ja; koja kao ulazni argument (i to PRVI) prima vrednost akumulatora, i koja vraća novu vrednost akumulatora, koja se zatim prosleđuje funkciji pozvanoj za sledeći element niza, itd.);

Početna vrednost akumulatora se prosleđuje kao DRUGI argument!

23. ES6 klase

Klase su u stvari "specijalne fje" i sintaksa klase ima dve komponente:

- Izrazi klasa (Class Expressions)
- Deklaracije klasa (Class Declarations)

Kao što imamo izraze i deklaracije funkcija (analogija) za deklaraciju klasa koristimo ključnu reč class

```
class Pravougaonik {  
    constructor(visina, sirina) {  
        this.visina = visina;  
        this.sirina = sirina;  
    }  
}
```

Razlika između deklaracija f-ja i deklaracija klase je to što se deklaracije f-ja HOIST-uju, a deklaracije klasa NE.

```
const p = new Kvadrat(); → greška  
class Kvadrat {}
```

Izrazi klase (još jedan način za definiciju klase)

Izraz klase može da ima ime, a i ne mora.

Ime datom izrazu klase je lokalno za telo klase

```
let Kvadrat = class {  
    constructor(a) {  
        this.a = a;  
    }  
    povrsina() {  
        return a*a;  
    }  
};  
var obj = new Kvadrat(5);  
obj.povrsina();
```

```
constructor(a) {  
    this.a = a;  
}  
povrsina() {  
    return a*a;  
};
```

```
this.a = a;  
return a*a;
```

```
let Kvadrat = class Kvadrat2 {  
    constructor(a) {  
        this.a = a;  
    }  
    povrsina() {  
        return a*a;  
    }  
};
```

```
constructor(a) {  
    this.a = a;  
}
```

→ console.log(Kvadrat.name)
izlazi "Kvadrat2"

```
povrsina() {  
    return a*a;  
};
```

Export class

24. ES6 nasleđivanje

ES6 podržava koncept nasleđivanja. Nasleđivanje je sposobnost programa da kreira nove entitete od postojećeg entiteta (ovde iz postojeće klase u novu klasu).

Klasa od koje se kreiraju nove klase je roditeljska klasa/super klasa.

Novonastale klase se nazivaju dete/podklasa. Nasleđivanje se obavlja pomoću ključne reči "extends".

Deca nasleđuju sva svojstva i metode (osim konstruktora) iz roditeljske klase.

```
class Shape {                                class Circle extends Shape{  
    constructor(a){                         disp(){  
        this.area=a;                      console.log(this.area);  
    }                                         }  
}                                         }  
var obj=new Circle(220);  
obj.disp(); → output = 220
```

28. DOM definicija i najčešća f-ja API-ja

DOM = Document Object Model, standardizovani API za strukturu dokumenta (HTML, XML, SVG, ...)

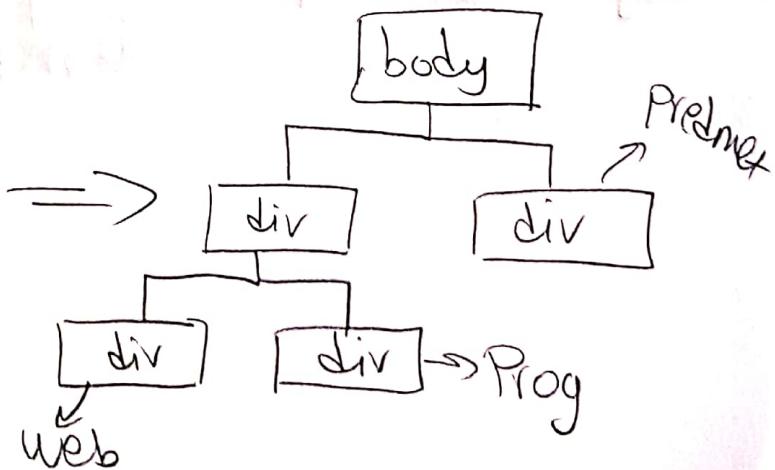
DOM nudi standardni skup objekata za predstavljanje HTML i XML dokumenta, kao i standardni interfejs za njihovu manipulaciju i pristup.

Pri uvođenju DOM-a je svaki web čitač imao drugačiji pristup objektima (stara koja je radila u Explorere nije radila u Netscape-u npr.).

Bez DOM-a ne bi bilo moguće pisati kod koji će raditi na svim web pretraživačima.

DOM svaki dokument vidi kao stablo (hijerarhiju čvorova)

```
<body>
  <div>
    <div>Web</div>
    <div>Prog</div>
  </div>
  <div>Predmet</div>
</body>
```



Najčešće korištene metode za pretragu HTML elemenata u DOM stablu:

1. `document.getElementById("nekiID");` vraca HTML element sa navedenim ID-om
2. `element.getElementsByTagName("button")` vraca niz HTML elemenata navedenog tipa koji su potomci element-a (element može = document)
3. `document.querySelectorAll("table + d input[type='checkbox'])` izvršava CSS selekciju i vraca niz elemenata
4. `document.querySelector("table + d input[type='checkbox'])` izvršava CSS selekciju i vraca prvi nadjeni element
5. `document.createElement("table")` kreira novi HTML element naziv HTML elementa
6. `Obj.innerHTML = "nesto"` menja sadržaj HTML elemenata
7. `Obj.style.color = "#66faaa"` menjanje boje HTML elementa
8. `Obj.value = 4` menjanje vrednosti elemenata (input elementi)

33. XML vs HTML

XML služi za prenos podataka, a ne za njihovu prezentaciju (za prezentaciju HTML)

XML elementi nisu predefinisani (kao u HTML-u), već ih korisnik sam definije (XML Schema)

34. XSL

XSL je porodica jezika koji se koriste za transformaciju i renderovanje XML dokumentata. Obuhvata 3 jezika

1. XSLT (XSL Transformation) - XML jezik za transformisanje XML dokumenta
2. XSL-FO (XSL Formatting Objects) - XML jezik za specificiranje vizuelnog formatiranja XML dokumenta
3. XPath (XSL Path Language) - jezik koji koristi XSLT, mada je dostupan i u drugim kontekstima i služi za adresiranje delova XML dokumenta.

XSL je poput CSS-a, samo za XML.

Opisuje kako treba prikazati XML dokument.

35. JSON → Java Script Object Notation

JSON je format podataka za razmenu koji koristi čitljiv tekst (čitljiv i razumljiv čoveku) da bi prenosio objekte podataka (koji se sastoje od atributa i tipova podataka niza).

Veliki uobičajen format podataka koji se koristi u asinhronoj komunikaciji servera i pretraživača.

var json = {
 "age": 24,
 "gender": "male",
 "name": "Bob"
};

JSON Objekat

Pristup json.age json.name

var jsonArray = [{ "name": "Bob", "age": "48" },
 { "name": "Ana", "age": "21" }];

jsonArray[0].name = "Bob"

36. JSON vs XML

JSON je čitljiviji za čoveka i ima manje reči (manje teksta za opis), JSON je takođe i brži (parsovanije XML-a je spor), i glomatno

JSON objekat

```
{"employees": [  
    {"name": "Bob", "lastname": "Smith"},  
    {"name": "Ana", "lastname": "Antic"}]}
```

XML ekvivalent JSON objekta

```
<employees>  
  <employee>  
    <name>Bob</name>  
    <lastname>Smith</lastname>  
  </employee>  
  <employee>  
    <name>Ana</name>  
    <lastname>Antic </lastname>  
  </employee>  
</employees>
```

37. AJAX

skup klijentskih tehnologija
- Asinhroni JavaScript i XML

To je skup tehnika za razvoj na web-u koje koriste web tehnologije na klijentskoj strani radi kreiranja asinhronih web aplikacija.

AJAX se koristi u google Drive-u, google maps-u, google suggest-u, Gmail, Hotmail, Yahoo mail, ...

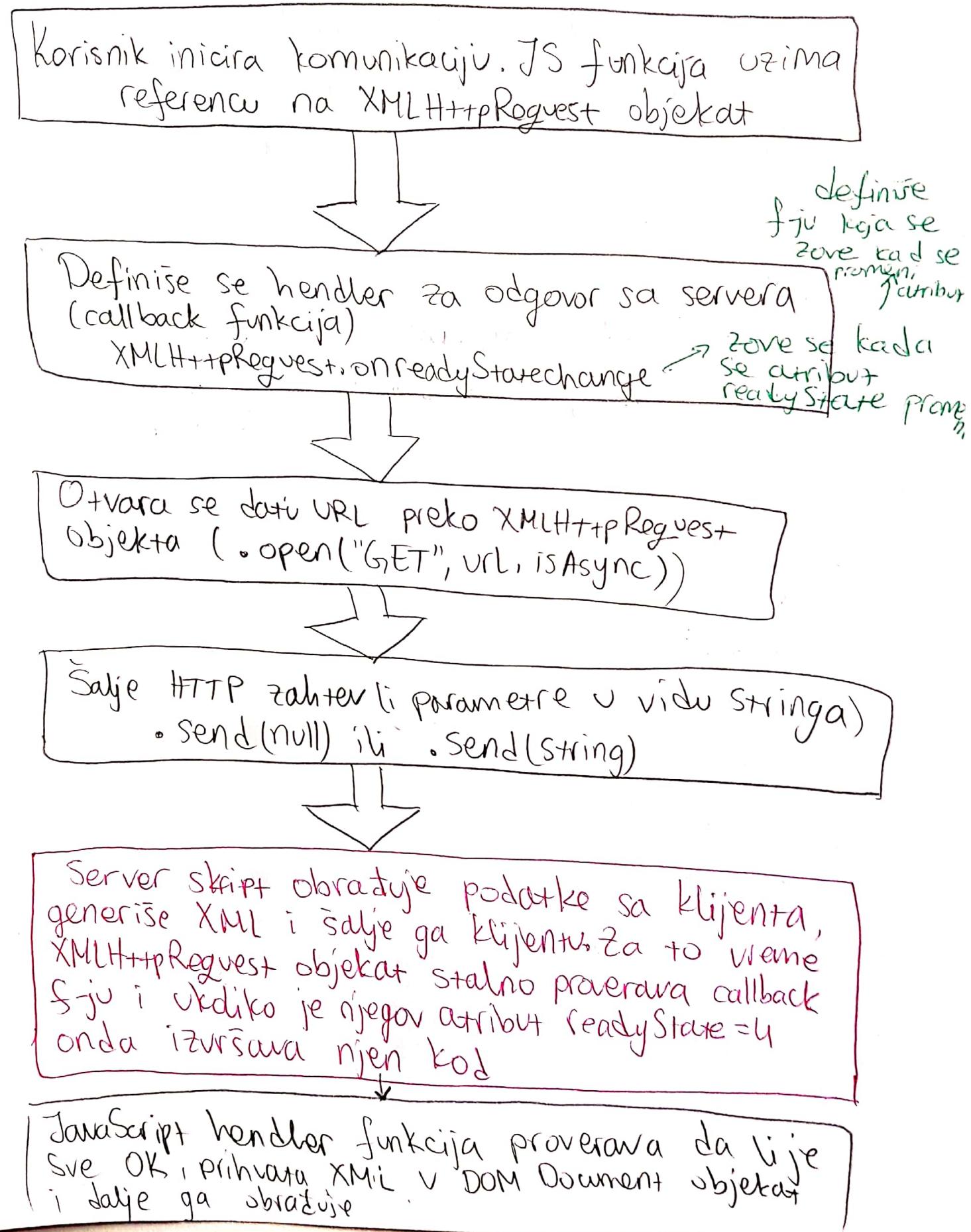
Korišćenjem AJAX-a, web aplikacije mogu da asinhrono šalju i primaju podatke sa servera, bez uticaja na trenutni izgled i trenutno ponašanje stranice.

Sloj razmene podataka je u potpunosti odvojen od prezentacionog sloja, što znači da je moguće dinamički menjati sadržaj stranice bez potrebe za ponovnim učitavanjem cele. Uprkos tome, ne mora se koristiti XML, najčešće se koristi JSON.

HTML i CSS se mogu zajedno koristiti za obeležavanje i stilizovanje podataka. DOM-u se pristupa pomoću JS-a radi dinamičkog prikaza podataka. JS i objekat XMLHttpRequest pružaju metode za asinhronu razmenu podataka između browsera i servera.

37. AJAX

Proces komunikacije



38. PHP opšte

PHP (HyperText Preprocessor) je serverski skriptni jezik dinamičkih tipova dizajniran za razvoj web aplikacija. Može se ugraditi u HTML, a može se koristiti u kombinaciji sa nekim šablonским sistemima (npr. Smarty).

PHP kod se piše u okviru `<?php //kod ?>`

Sve promenljive imaju prefiks \$ (`$age = 47;`).

Tip podataka ne mora da bude unapred specifikiran.

U PHP-u postoji mogućnost kreiranja klasa, kao i nasleđivanja.

Funkcije... glupo pitanje

39. PHP superglobalne promenljive

To su ugrađene promenljive koje su UVEK dostupne → može im se pristupati iz bilo koje funkcije, klase ili fajla. Ima ih 9:

1. \$GLOBALS

asocijativni niz

→ koristi se za čuvanje svih globalnih promenljivih iz bilo kog dela PHP skripte

`$x = 75;`

`$y = 43;`

`function add() {`

Samo ona nema donju crtu

`$GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];`

`} add();`

`echo $z;`

2. \$SERVER

→ sadrži informacije o headerima, putanjama i lokacijama skripti

`$_SERVER['SERVER_NAME']` → vratiča ime servera (www.elfatnac.rs)

3. \$_POST

→ sadrži informacije poslate sa klijenta preko POST metode

4. \$_GET

→ -||- GET metode

→ može da prikupi informacije sadržane u adresi (QueryString)

5. \$_FILES

6. \$_ENV

39. PHP superglobalne promenljive

7. \$_COOKIE

8. \$_SESSION

9. \$_REQUEST → sadrži podatke poslate sa klijenta preko GET ili POST metode

Podaci vidljivi
u adresi

Slanje veće
količine podataka
(ne vide se u
adresi)

40. Prenos podataka sa klijenta na server

41. PHP klase i objekti

Koristi se ključna reč `class`. Telo klase se nalazi između vitičastih zagrada nakon class sledi ime klase
class Macka{

 public ime='Flafi';

 public function mjawknij(){

 echo 'mjaw';

 public function imeMacke(){

 return \$this->ime;

}

}

`$this` → nestatičkim atributima se iz konteksta klase pristupa pomoću `$this->imeAttr`

Statičkim atributima se iz konteksta klase pristupa pomoću `self::imeAttr` → ne može sa \$klasa →

Atributi klase imaju 3 modifikatora pristupa:

→ public

→ private

→ protected

obavezno 1 modifikator

stoji uz atribut

(za metode ako ne стоји подразумева se `public`, za atributve NE)

Postoje 2 načina za pozivanje konstruktora:

→ direktno

→ preko promenljive

`$obj = new Macka();` `$imeKlase = 'Macka';`
`$obj -> mjakni();` `$obj = new $imeKlase();`
 ↗ ↗
 direktno preko promenljive

Pored globalnih konstanti, moguće je zadati konstante u okviru klase. Ključna reč je CONST.
Naziv konstante NEMA \$!

Pristup iz konteksta klase → self:: KONSTANTA

Pristup vanjske klase → imeKlase:: KONSTANTA

Atributi i metode mogu da imaju ISTI naziv.
\$this nije dostupno u okviru statičke metode!

Statičkim metodama možemo pristupati sa

→ \$promenljiva → imeMetodel() \$f = new Foo();
→ \$klasa:: imeMetodel() \$f -> fooStatic();
→ \$klasa::: imeMetodel() Foo::: fooStatic();

Statičkim atributima se pristupa samo sa ::

→ \$klasa:: staticAttr;
→ \$promenljiva:: staticAttr;

41. PHP klase i objekti -DOPUNA-

Postoje i apstraktne klase u PHP-u koje se ne mogu instancirati. Klase koje imaju barem 1 apstraktnu metodu, moraju biti apstraktne. Kad konkretna klasa nasleđuje apstraktnu, apstraktne metode roditeljske klase moraju biti definisane. Metoda implementirana u izvedenoj klasi mora biti ISTOZ ili VIŠEG stepena vidljivosti u odnosu na apstraktne metode u osnovnoj klasi (protected u osnovnoj, protected ili public u izvedenoj)

Interfejsi zadaju samo metode koje klase trebaju da implementiraju, bez specifikacije kako treba implementirati metode. Ključna reč **interface**. Sve metode interfejsa moraju biti javne (eksplicitno pisanjem **public**, ili podrazumevano bez pisanja mod pristupa). Ključna reč **implements** specificira da klasa implementira interfejs.

Interfejsi mogu metusobno da se nasleđuju (ključna reč **extends**). Klasa može da implementira više interfejsa (odvajaju se zarezom). Interfejs može da sadrži konstantu, a klasa koja implementira taj interfejs, NE može da predefiniše tu konstantu.

42. PHP nasleđivanje

Ključna reč kojom se označava da jedna klasa nasleđuje drugu je **extends**.

Dozvoljeno je samo jednostruko nasleđivanje.

Nasleđeni atributi i metode se mogu preklopiti u izvedenoj klasi (override), tako što se deklarišu atributi i metode sa istim imenom kao u osnovnoj.

Metodama osnovne klase se stavlja modifikator **final** da bi se zabranilo preklapanje.

Preklopljenim atributima i metodama osnovne klase se pristupa sa parent::<attribution-metoda>

Konstruktor je metoda koja se definise:

→ **construct (<lista argumentata>)**

dve donje
cite

Konstruktor izvedene klase NE zove automatski konstruktor osnovne klase.

Ne postoji overloading funkcija, pa klase mogu imati samo 1 konstruktor.

Destruktor je metoda oblika:

→ **destruct()** → Destruktor izvedene klase NE zove automatski destruktur osnovne

dve donje
cite