FRONTEND MĀJASLAPAS IZSTRĀDE

10. lekcija - JavaScript klases un Scope

LATVIJAS UNIVERSITĀTE
BIZNESA, VADĪBAS UN EKONOMIKAS FAKULTĀTE

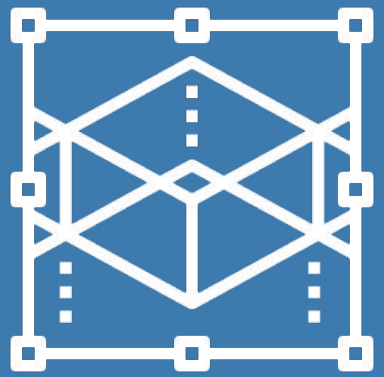
VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020

EIROPAS SAVIENĪBA
Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ

ESF projekts Nr. 8.4.1.0/16/I/001 ''Nodarbināto personu profesionālās kompetences pilnveide"



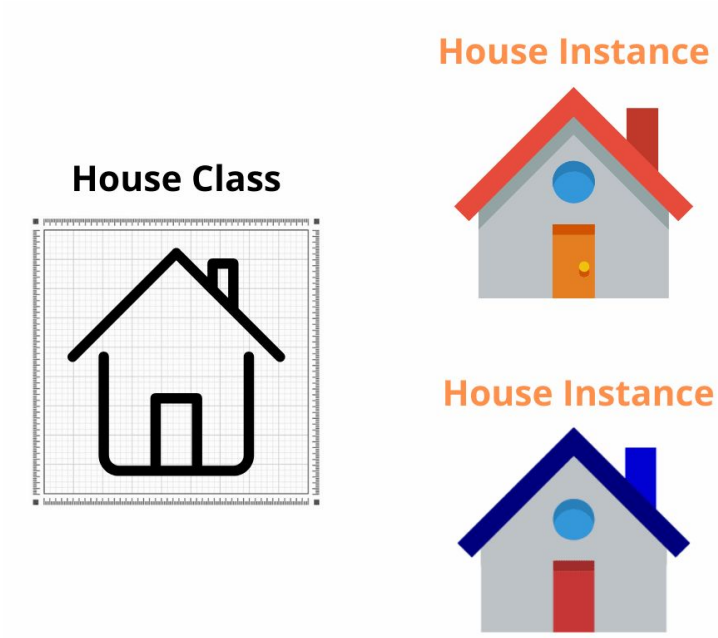
JavaScript klases

Kas ir klase?

Kopš ES6 ir iespējams izveidot klases. Klase ir šablons objektu izveidei. Klasiskajās programēšans valodās klases ir fundamentāla šo valodu daļa, tomēr JavaScript tas ir salīdzinoši jauns papildinājums.
Klasi var uztvert kā objekta rasējumu. Definējam klasi vienreiz - izveidojam cik vien objektus nepieciešams, izmantojot definēto klasi jeb rasējumu.
Katrs jaunizveidotais objekts ir klases instance un ir nesaistīts ar citām klases instancēm.
Līdz šim esam izmantojuši šādas JavaScript iekļautās klases:

- **Date**
- **URL**
- **URLSearchParams**

Atšķirībā no mainīgo nosaukumiem - klases nosaukums vienmēr jāsāk ar lielo burtu.



Klases inicializēšana

Ar klases inicializēšanu saprotam jaunas instances jeb objekta izveidi ar klasē definētajām īpašībām un metodēm.
Piem., lai iegūtu jaunu **Date** objektu, izveidojam mainīgo, kurā tiks saglabāta šī jaunā klases instance:

```
let date = new Date();
```

Tagad varēs im izmantot mainīgo **date** un visas **Date** klasē definētās metodes un īpašības.
Klases inicializējot tās var saņemt arī parametrus gluži kā funkcijas, piem., **URL** klase sagaida URL string vērtību kā parametru, kad tā tiek inicializēta:

```
let urlObj = new URL("https://some-url.example");
```

Klases definēšana

Kad mums ir nepieciešams izveidot pašiem savu klasi, svarīgi norādīt atslēgvārdu **class** un klases kodā norādīt funkciju **constructor()**. Klases sintaktiskais skelets izskatītos šādi:

```
class Car {
   constructor() {};
}
```

Tieši **constructor()** metode mums ļauj izmantot parametrus kā funkcijās, kad inicializējam jaunu klases instanci.

```
class Car {
   constructor(brand, model) {
        this.brand = brand;
        this.model = model;
   };
}
```

Tad inicializējot jaunu klases instanci varam izmantot parametrus:

```
let myCar = new Car("Volvo", "S60");
```

Un piekļūt tās īpašībām:

```
console.log(myCar.model) // S60
```

This atslēgvārds

this atslēgvārds norāda objektu. Kādu objektu - atkarīgs no tā kā šis **this** tiek izsaukts:

Kāda objekta metodē vai klases insatncē (arī objekts), **this** būs vienāds ar pašu objetu, kurš satur šo metodi.

Neizmantots funkcijā vai objektā, **this** būs globālais objekts jeb window objekts.

Izmantos funkcijā, kas nav kāda objekta īpašība, **this** bus globālais objekts jeb window objekts.

Notikuma klausītāja(event listener), **this** būs elements, kurš izsauca šo notikumu.

Klašu mantojamība

Jebkura klase var mantot jeb iegūt piekļuvi kādas citas klases īpašībām un metodēm izmantojot **extends** extends atslēgvārdu.

```
class Car {
  constructor(brand) {
    this.carname = brand;
  }

  present() {
    return 'I have a ' + this.carname;
  }
}

class Model extends Car {
  constructor(brand, model) {
    super(brand);
    this.model = model;
  }

  show() {
    return this.present() + ', it is a ' + this.model;
  }
}
```

super() metode

super() ir noderīga, kad izmantojam **extends** jeb mantojam citas klases īpašības. Tieši šī metode iepriekšējā piemērā par mantošanu mums ļauj izmantot **Car** klases īpašības un metodes iekš **Model**. Izsaucot **super()** - mēs inicializējam kontruktoru klasē no kā mantojam.
Tieši **constructor()** metode izveido jauno objekta instanci un atgriež tā īpašības.

Scope

Globālais scope

Pirms ES6 JavaScript bija pieejams tikai viena veida scope - globālais scope.
Tas aizvien ir ļoti noderīgs situācijās, kas vēlamies lai kāds mainīgais, funkcija vai objekts ir pieejams jebkurā citā funkcijā vai objektā.
Tomēr ieteicams global scope atstāt tikai to, kas ir absolūti nepieciešams. Kods, kas satur visu globālajā scope ātri kļūst neuzturams un bieži rasties dažādas kļūdas kodā.
Piemēram līdz šim esam strādājuši ar tādiem globālā scope objektiem un klasēm kā:

- **window**
- **document**
- **Date**
- **URL**
- **URLSearchParams**

Bloka scope

Kopš ES6 varam izmantot mainīgo deklarācijas **let** un **const**. Tieši šie atslēgvārdi mums ļauj paslēpt mainīgos no globālas pieejamības un padara tos pieejamus tikai noteiktajā koda blokā. Ar koda bloku saprotam kodu, kas atrodas starp figūriekavām **{ }**. Piemēram **for** cikls **if() { } else { }** un jebkura funkcijas deklarācija **() => { }** izmanto šīs figūriekavas, lai definētu koda bloku.

```
if(true)  {
    let x = 2;
   const y = 3;
}
console.log(x) // undefined
console.log(y) // undefined
```

x un **y** mainīgajiem var piekļūt tikai **if() { }** koda blokā jeb kodā kas atrodās starp figūriekavām **{ }**

```
const person = () => {
        let name = “Full name”;
        let age = 43;
}
console.log(name) // undefined
console.log(age) // undefined
```

name un **age** mainīgajiem var piekļūt tikai **person()** funkcjas deklerācijas koda blokā jeb kodā kas atrodās starp figūriekavām **{ }**.

Novecojuši var deklerācija

Kaut arī koda piemēros interneta resursos bieži var atrast mainīgo deklerāciju **var** - to nav ieteicams izmantot, jo jebkurā koda blokā deklarēts mainīgais ar **var** atslēgvārdu būs pieejams jebkurā citā koda blokā, proti, globālajā scope. Un kā jau minēts iepriekš - globālajā scope jāatrodās tikai absolūti nepieciešamajam jeb kam tādam, kas tiešām ir izmantojams viscauri projekta kodam un jābūt pieejamam jebkurā koda blokā (funkcijā, objektā u.c.).

```
if(true)  {
   var globallyAvailableVariable = “I'm not that useful to be global..”;
}
```

```
console.log(globallyAvailableVariable) // “I'm not that useful to be global..”
```

Mainīgais **globallyAvailableVariable** ir pieejams un izmantojams arī ārpus **if() { }** koda bloka un “piesārņo” globālo scope.

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020

EIROPAS SAVIENĪBA
Eiropas Sociālais fonds

IEGULDĪJUMS TAVĀ NĀKOTNĒ

ESF projekts Nr. 8.4.1.0/16/I/001 ''Nodarbināto personu profesionālās kompetences pilnveide"

VADĪBAS UN UZŅĒMĒJDARBĪBAS MĀCĪBU CENTRS

Programmas nosaukums