FRONTEND MĀJASLAPAS IZSTRĀDE

6. lekcija - cikli, funkcijas un notikumi(events)

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"

Cikli

for

Ar **for** ciklu varam veikt kāda koda bloka izpildi noteiktu reižu skaitu. Šim ciklam svarīgas ir 3 izteiksmes.

izteiksme 1:
let i = 0; - izveido mainīgo i jeb index kura vērtība tiks mainīta pēc katra cikla izpildes (iterācija). Šis mainīgais kalpos kā skaitītājs.

izteiksme 2
i < 10; - izveido nosacījumu, kas tiek pārbaudīts pirms katra cikla izpildes (iterācija). Cikls turpinās izpildīties kamēr nosacījuma vērtība būs true.

izteiksme 3
i++; - darbība, kas tiks izpildīta katra cikla beigās.
Šajā gadījumā katra cikla beigās i vērtība tiks palielināta par 1.
i++; ir saīsināts pieraksts izteiksmei **i = i + 1;**

```
for (izteiksme 1; izteiksme 2; izteiksme 3) {

    // kods ko izpildīt noteiktu reižu skaitu

}
```

```
for (let i = 0; i < 10; i++) {

    // kods ko izpildīt 10x

}
```

forEach

forEach cikls ir metode massīviem - arrays.
Ar to mēs varam iterēt caur visiem masīva elementiem un pie katra elementa cikla izpildīt kodu.

forEach(); metodē kā parametrs ir jānorāda funkcija, kas tiks izpildīta katrā iterācijā. Viens no veidiem kā šo funkciju norādīt ir ar sintaksi **() => {}**

forEach(() => {}); Tomēr šajā brīdī kā parametrs norādītā funkcija nepilda nekādas darbības.
Ja funkcijai tiks norādīts viens parametrs, tas būs elements kura iterācijas kārta ir pienākusi.

(item) => {}
Ja funkcijai tiks norādīts arī otrs parametrs, tas būs šī elementa indeks masīvā.

(item, index) => {}
Šajā brīdī jau varam veikt ko jēgpilnu katrā iterācijā.

forEach((item, index) => { // kods ko izpildīt katrā iterācijā **}**

Funkcijas parametru nosaukumus varam brīvi izvēleties.

```
let exampleArray = ["el 1", "el 2", "el 3"]

// ar 1 parametru - masīva elementu
exampleArray.forEach((currentEl) => {
    console.log("Array element", currentEl);
}

// ar 2 parametriem - masīva elementu un indeksu
exampleArray.forEach((currentEl, index) => {
    console.log("Array element",  currentEl);
    console.log("Index of the element",  currentEl);
}
```

Funkcijas

Funkcijas sintakse

Funkcijas ir rīki ko paši izgatavojam konkrētu uzdevumu veišanai. Funkcija var saņemt neierobežotu skaitu parametru, kas nepieciešami darbību veikšanai. Kaut arī aizvien bieži sastopama pirms ES6 funkcijas pieraksta sintakse, ieteicams izmantot ES6 sinatki.

ES6

```
let someFunction = (parametrs 1, parametrs 2, ... ) => {
  // funkcijas koda bloks
}
```

Prims ES6:

```
function nosaukums(parametrs 1, parametrs 2, ... ) {
  // funkcijas koda bloks
}
```

```
let someFunction = function(parametrs 1, parametrs 2, ... ) {
  // funkcijas koda bloks
}
```

Funkciju izpilde

Funkciju deklarēšanā ir jānorāda vai tā atgriezīs vai neatgriezīs rezultātu.
Ja funkcija atgriež rezultātu, tās izpildi saglabājot mainīgajā - mainīgais iegūs funkcijas atgriezto vērtību.

Neatgriežot vērtību

```
// deklarējam globālu mainīgo, kas tiks modificēts
let someVar;

// deklarējam funkciju un norādam kādus mainīgos tā saņems
let someFunction = (x, y) => {
  someVar = x + y
}

// izpildām funkciju un padodam tai parameturs
someFunction(2 ,3);

// izvadām globālā mainīgā vērtību
console.log(someVar); // izvadītais someVar rezultāts ir 5
```

Atgriežot vērtību

```
// deklarējam globālu mainīgo, kas tiks modificēts
let someVar;

// deklarējam funkciju un norādam kādus mainīgos tā saņems
let someFunction = (x, y) => {
  return  x + y
}

// izpildām funkciju un padodam tai parameturs
someVar = someFunction(2 ,3);

// izvadām globālā mainīgā vērtību
console.log(someVar); izvadītais rezultāts ir 5
```

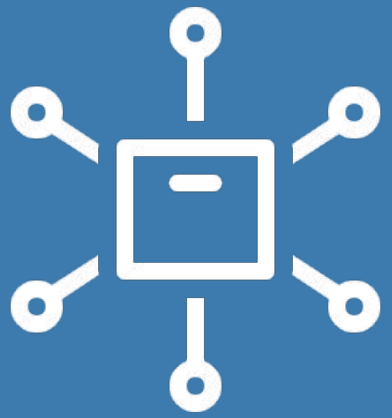
Try Catch bloks

Būtiski ir kontrolēti reaģēt uz kļūdām kodā (errors). Proti, apzināties momentus, kad kādā koda blokā varētu būt iespējama kļūda un paredzēt kodu šādam gadījumam. To mēs varam panākt ar Try Catch bloku.

```
let emailList = [“ EMail1@Mail.Com   ”, “  EMail2@Mail.Com ”, null, “   EMail3@Mail.Com ”];

let formatEmail = (item) => {
        try {
                item.trim();
                item.toLowerCase();
        }
        catch(err) {
                console.log(“Not able to format item:”, err);
        }
}

emailList.forEach(formatEmail(arrayItem))
```



Mijiedarbība ar DOM

DOM elementu meklēšana

Lai mijiedarbotos ar DOM elementiem, tos nepieciešams atrast ar kādu selektrou. Līdzīgi kā veidojām selektors CSS, arī JS funkcijās, kas tiek izmantotas DOM elementu atrašanai - jānorāda selektors.

Meklēt elementus pēc to id atribūta vērtības. Atgriež vienu elementu.
getElementById(selektors);

Meklēt elementus pēc to klases atribūta vērtības. Atgriež masīvu ar elementiem.
getElementsByClassName(selektors);

Meklēt elementus pēc to birkas nosaukuma (tag name). Atgriež masīvu ar elementiem.
getElementsByTagName(selektors);

Atgriež pirmo atrasto elementu, kas atbilst selektora vērtībai. Selektors norādīts brīvi kā CSS.
querySelector(selektors);

Atgriež masīvu ar visiem atrastajiem elementiem, kas atbilst selektora vērtibai. Selektors norādīts brīvi kā CSS.
querySelectorAll(selektors);

Darbības ar DOM elementiem

Ar DOM elementiem varam mijiedarboties, izmantojot to metodes vai mainot īpašību vērtības.

Metodes
Element.getAttribute("attributeName") - iegūt norādītā atribūta vērtību.
Element.setAttribute("attributeName", "value") - iestatīt norādītā atribūta norādīto vērtību.
Element.focus() - piešķirt fokusu elementam.
Element.scrollTo("x position", "y position") - navigēt(scroll) iekš elementa līdz norādītajām x un y koordinātām.
Element.scrollIntoView() - navigēt(scroll) skatu līdz elements ir redzams.

Īpašības
Element.classList - iegūt elementa klases atribūta visas vērtības.
Element.dataset - iegūt elementa data attribūta vērtības.
Element.style - iegūt elementa style atribūta visas vērtības.
Element.innerHTML - iegūt elementa HTML iekšējo saturu.
Element.outerHTML - iegūt elementa HTML apkārtējo saturu.
Element.innerText - iegūt elementa saturošo redzamā noformējuma teksta vērtības.
Element.textContent - iegūt elementa saturošo visa teksta vērtības, neatkarīgi no redzamības noformējuma.
Element.hidden - iegūt elementa redzamību, atgriež boolean vērtību. Nav saistīts ar CSS redzamības noformējuma.

Notikumi (events)

Notikumu(events) klausīšanās(listening)

Jebkurš DOM elements var signalizēt par kādu notikumu(event) - peles klikšķi, taustiņa nospiešanu, izmēra maiņu, navigāciju(scroll) un citiem notikumiem. Lai uz šiem notiekumiem reģētu, ir pieejamas metodes dēvētas par klausītājiem(listeners). Izveidot sasaisti starp notikumu un klausītāju var divos viedos.
Pilnu sarakstu ar iespējamiem events, skatīt šeit.

Events listener kā elementa īpašība/attribūts

```
element.onclick = (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.onscroll = (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.onkeypress = (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.onmouseeneter = (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
```

Event listener kā elementa metode

```
element.addEventListener('click', (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.addEventListener('scroll', (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.addEventListener('keypress', (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
element.addEventListener('mouseenter', (eventObject) => {
        // kods ko izpildīt līdz ar notikumu
}
```

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