wantsome
The friendly IT Academy

Saptamana 9

Partea 1

Programare Front-End


Callbacks

Callbacks & JavaScript Event Loop

Callback: o functie care va fi executata dupa ce se termina executia unor instructiuni de cod incapsulate in cadrul unei alte functii

- functiile pot accepta ca argumente si alte functii - acestea se numesc **callback functions**

De ce avem nevoie de callbacks ?

- ***main process*** din browser este ***single threaded***
- daca ar avea loc executia unei operatii de lunga durata in cadrul unui ***single-threaded event loop***, procesul ar fi blocat

Callbacks & JavaScript Event Loop

```
function first(){
    console.log(1);
}

function second(){
    console.log(2);
}

first();
second();
```

```
function third(){
    console.log(A);
    setTimeout(
        () => console.log(B), 1000
    )
    console.log(C);
}
```

What do you expect to be printed here ?

Event Loop

https://flaviocopes.com/javascript-event-loop/

https://medium.com/front-end-weekly/javascript-event-loop-explained-4cd26af121d4

Callbacks

Imagine this blocking situation:

```
const orderPizza = flavour => {
  callPizzaShop(`I want a${flavour} pizza`);
  waits20minsForPizzaToCome(); // Nothing else can happen here
  bringPizzaToYou();
}

orderPizza('Hawaiian');
mopFloor();
ironClothes();
```

Callbacks

We actually want to be able to do something like this:

```
const orderPizza (flavor, smthToDoWhenPizzaArrives) {
  callPizzaShop(`I want a ${flavor} pizza`)
  whenPizzaComesBack {
    smthToDoWhenPizzaArrives //pseudo-code
  }
}

const layTheTable = _ => console.log('laying the table')

orderPizza('Hawaiian', layTheTable)
mopFloor()
ironClothes()
```

Callback

Ce se intampla atunci cand vom face comunica cu un server prin intermediul unui request ?

- In exemplul alaturat, se va printa mai intai 2 apoi 1

- JS nu a asteptat va astepta ca `first` sa se execute complet, ci va continua cu executia functiei `second`

Daca am **stabilit, scris,** la nivel de cod, o ordine a executiei, acest lucru nu este garantat !

- din acest motiv folosim **callbacks.**

```
function first(){
  // Simulate a code delay
  setTimeout( function(){
    console.log(1);
  }, 500 );
}
function second(){
  console.log(2);
}
first();
second();
```

Callback

Callback-urile reprezinta totodata o cale prin care ne asiguram ca o anumita functie sau instructiune de cod nu este executata pana in momentul in care a fost indeplinit un anumit task

```
function doHomework(subject, callback) {
  alert(`Starting my ${subject} homework.`);
  callback();
}

doHomework('math', function() {
  alert('Finished my homework');
});
```

```
function doHomework(subject, callback) {
  alert(`Starting my ${subject} homework.`);
  callback();
}
function alertFinished(){
  alert('Finished my homework');
}
doHomework('math', alertFinished);
```

PRACTICE: **Callbacks**

http://bit.do/exST

shorturl.at/cnE01