Callback

Callback je funkce, která je předána jako argument jiné funkci a je vykonána později, obvykle po dokončení určité operace nebo při vzniku konkrétní události. Callbacky jsou důležitou součástí asynchronous programming a event-driven programming.

Používají se zejména v jazyce javascript, ale lze se s nimi setkat i v mnoha dalších programovacích jazycích.

Jak Callback funguje

Namísto okamžitého vykonání funkce je funkce předána jiné části programu, která ji zavolá ve vhodný okamžik.

```
Funkce A ↓ Předá Callback ↓ Proběhne operace ↓ Callback je spuštěn
```

Jednoduchý příklad

```
function pozdrav() { console.log("Ahoj!"); }

function provedAkci(callback) {
callback();
}

provedAkci(pozdrav);
```

Výstup:

```
Ahoj!
```

Funkce `pozdrav()` je předána jako callback a vykonána uvnitř funkce `provedAkci()`.

Anonymní Callback

Velmi často se používají anonymní funkce:

```
setTimeout(function() { console.log("Čas vypršel."); }, 1000);
```

Po uplynutí jedné sekundy je callback automaticky spuštěn.

Callback a události

Callbacky jsou běžnou součástí práce s událostmi.

```
button.addEventListener("click", function() { console.log("Kliknutí."); });
```

Anonymní funkce zde představuje callback, který se spustí při události `click`.

Callback a asynchronní operace

Při načítání dat nebo práci se soubory se callback vykoná až po dokončení operace.

Příklad:

```
nactiData(function(data) { console.log(data); });
```

Program mezitím může pokračovat v dalších činnostech.

Callback Hell

Při větším počtu vnořených callbacků může vzniknout problém známý jako Callback Hell.

Příklad:

```
akce1(function() { akce2(function() { akce3(function() { akce4(function() {
        });
    });
});

});
```

Takový kód je obtížně čitelný a složitě se udržuje.

Moderní alternativa

V moderním JavaScriptu se callbacky často nahrazují pomocí:

promise async await

Příklad s Promise:

```
fetch("/data") .then(response => response.json()) .then(data =>
console.log(data));
```

Výhody

Jednoduchý princip. Podpora asynchronních operací. Efektivní práce s událostmi. Nízké nároky na výkon.

Nevýhody

Při větším množství callbacků vzniká nepřehledný kód. Obtížnější ladění chyb. Horší čitelnost u složitějších aplikací.

Oblasti použití

Zpracování událostí. Síťová komunikace. Časovače. Práce se soubory. API požadavky. Animace. Asynchronní operace.

Související pojmy

event event handler event loop event listener asynchronous programming promise async await javascript node.js

Shrnutí

Callback je funkce předaná jiné funkci za účelem jejího pozdějšího vykonání. Callbacky tvoří základ práce s událostmi a asynchronními operacemi v JavaScriptu i dalších programovacích jazycích. Přestože jsou stále široce používány, u složitějších aplikací jsou často nahrazovány technologiemi Promise a Async/Await, které nabízejí přehlednější zápis kódu.