13 TIMERY, async programovanie

možno si v budúcnosti povieme viac o tzv. **asynchrónnom programovaní**. je to len ďalšie veľké, komplikované slovo pre vec, ktorú už celý čas robíme 🙂 lebo ty už vieš, že furt píšeme funkcie. keď máš viac riadkov, ktoré majú spoločnú misiu, obalíš ich do funkcie. a funkciu potom môžeš:

- zavolať priamo
- použiť v cykle
- hodiť ju do `fetch`
- spustiť ju ak nastane `click`
- ...

no a niektorú funkcie sa spustia v budúcnosti 😊 ak dáš funkciu do `click` ona leží v kóde a čaká. kým nastane `click`. čiže až budúcnosti. ak niekto klikne. cez `fetch` si vyžiadaš dáta. a tvoja funkcia vo `fetch` sa spustí až budúcnosti. v momente, keď tie dáta dorazia.

a možno najjasnejší reprezentant je...

TIMER (setTimeout, setInterval)

tento výpis do konzole príde o 2,5 sekundy:

```
setTimeout(function() {
  console.log('ahoj z budúcnosti!')
}, 2500);

// skratena "arrow function" verzia zapisju
setTimeout(() => console.log('ahoj'), 2500);
```

si to predstav jak tie kuchyňské minutky: ty varíš a nastavíš si budík na 5 minút. medzitým nečakáš. ty makáš a varíš ďalej. a o 5 minút začne vrieskať. rovnako `setTimeout` v javascripte. javascript nastaví budík na 2,5 sekundy a medzitým pokračuje a spúšťa ďalší kód. ale o 2,5 sekundy sa spustí funkciu, ktorá vypíše "ahoj z budúcnosti!"

`setTimeout` od teba chce 2 veci:

- funkciu, ktorá sa má spustiť
- počet mili-sekúnd, o ktoré sa spustiť

`setInterval` je podobný, akurát on funkciu bude **spúšťať opakovane**. `setTimeout` funkciu spustí jedenkrát. `setTimeout` ju spúšťa dokola, v časovom intervale, ktorý mu nastavíš.

setTimeout NA ODPOČET

tú funkciu môžeš do setTimeout poslať priamo, ako v príklade hore. ale na hodine sa niekto spýtal, či si môžeš vytvoriť funkciu aj "klasickým spôsobom" a vložiť ju do timera. a môžeš! napríklad aha... povedzme, že máš premennú:

```
let seconds = 6;
```

a chceš toto číslo znížiť **každú sekundu**. čiže odpočítavanie. countdown. fɪⁿal couⁿtdoʷⁿ. europe. ¹⁹⁸⁶. spravíš funkciu:

```
function tick() {
  seconds = seconds - 1;
  console.log(seconds);
}
```

a cez `setTimeout` povieš, aby javascript spustil funkciu s názvom `tick` o `1000 milisekúnd` (jednu sekundu):

```
setTimeout( tick, 1000 );
```

čas píšeme v **mili**-sekundách. tisícina sekundy. v tomto prípade názov funkcie napíšeš **bez zátvoriek()**. pretože `tick()` by znamenalo *"spusti funkciu okamžite, teraz, OKAMŽITE!"* a to tu nechceš.

v tomto prípade sa funkcia `tick` spustí **jedenkrát**. do konzole vypíše číslo 5. ak chceš, aby sa funkcia `tick` spúšťala dokola, môžeš ju v nej zavolať znova:

```
function tick() {
  seconds = seconds - 1;
  console.log(seconds);
  setTimeout(tick, 1000); // a sekundu sa zavola znova
}
```

na domácu úlohu si môžeš skúsiť spraviť odpočet pomocou `setInterval` a dorobiť, aby pri nule vystrelili konfety 🎉 pretože prečo nie! trošku zábavy do toho javascriptu 🙂

NA ČO SA REÁLNE POUŽÍVAJÚ TIMERY

na všetko možné vždy! toto rozhodne nie vyčerpávajúci zoznam, ale na:

- **odpočítavanie** sme videli
- **pravidelný FETCH na dáta** gmail pravidelne kontroluju, či neprišiel email
- **slideshow, carousels** eshop ťa chce oklamať, aby si si kúpila produkt
- **pravidelné ukladanie dát** gmail ukladá text, keď píšeš

- **notifikácie** príde otravná notifikácia, je tam 10 sekúnd, odíde
- **(de)aktivácia buttonov** občas chceš zablokovať button na pol sekundy, aby sa človek ja neviem... nezaregistroval 10x na stránke, ak kliká jak blázon
- **videohry** ich používajú nonstop a na všetko... keď zomrieš a tvoj panák sa znova objaví, sekundu bliká a je nezraniteľný. timer. alebo zakúzliš spell, trvá 20 sekúnd, kým ho môžeš použiť znova. timer. zomrem na starobu, ak budem vymenúvať ďalšie príklady, v hrách sú timery úplne všade.
- **debounce** netflix má formulár. začneš písať názov filmu. on automaticky vyhľadáva podľa prvých napísaných písmen. často ale nevyhľadáva vôbec okamžite. naopak čaká, kým na povedzme 350 milisekúnd písať prestaneš. pretože toto vyhľadávanie je často náročná, dlho trvajúca operácia, kedy sa musí spojiť s databázou a vyhodnocovať, ktoré filmy na základe tvojej predošlej aktivity odporučiť práve tebe... a toto nechceš robiť 200x za sekundu. tak čakáš, kým človek reálne na chvíľu prestane písať. až potom vyšleš žiadosť!

a tisíce ďalších príkladov. ale to by mi odpadli prsty a ako potom budem písať kód? budem sa *rozprávať* s počítačom?? NEBUDEM.

setInterval A JEDNA ZVLÁŠTNOSŤ

ukazoval som `setTimeout`. `setTimeout` spustí kód neskôr. později. po pauze. naproti tomu `setInterval` spúšťa kód dokola. tak často, ako mu povieš. na odpočítavanie by bol možno lepší práve `setInteval`.

skús ho použiť. uvidíš, že sú veľmi podobné. ale majú **jeden zvláštny rozdiel**. napíšem ho sem, ale pokojne to ignoruj. v programovaní je kopa špecialitiek, ktorú nie sú dôležité absolútnu väčšinu času 😊 programovanie musí byť strašidelné, ak sa snažíš všetky tieto informácie držať v hlave. nemusíš!! dôležité sú premenné, funkcie, cykly a to, aby si nemala v kóde bordel. ostatné sú srandičky. takže ak ťa to zaujíma, prečítaj si. ak ťa to nezaujíma, vyhoď z hlavy. nezáleží na tom. každopádne:

`setTimeout` sa snaží dodržať pauzu medzi odpalmi. v našom `tick()` príklade hore bude medzi každým spustením funkcie 1 sekundová pauza. ale predstavme si, že by kód v tick funkcii trvalo počítaču spustiť 5 sekúnd. `setTimeout` by povedal *"ok, funkciu trvalo spustiť 5 sekúnd, teraz 1 sekundu počkáme, potom ju spustíme znova"*

naproti tomu `setInterval` by nepovedal nič a OKAMŽITE funkciu spustil znova 😊 a potom by povedal *"ok, kód máme spúštať každú sekundu, ale táto funkcia trvala 5 sekúnd, takže máme 4 sekundový časový sklz OKAMŽITÝ ODPAL ZNOVA"*. jo? `setInterval` sleduje ako dlho trval beh funkcie a snaží sa podľa toho upraviť dĺžku čakania medzi odpalmi. ak mu nastavíš 1 sekundu, ale spustenie funkcie trvalo pol sekundy, spustí ju znova o pol sekundy. `setTimeout` by ju spustil o 1 sekundu.

pre `setTimeout` je dôležitá **pauza** medzi odpalmi. ale `setInterval` sa snaží dodržať **interval** odpalov.

ok, teraz toto vyhoď z hlavy, stiahni si priložený kód 👇 a hraj sa s ním ♥