08 ANIMATE DAY.JS, knižnice v praxi

LIBRARIES (knihovny 📚)

keď si do `.js` súboru vytvoríme funkciu `financial()` alebo `createNewCard()`, pridáme javascriptu novú schopnosť. naučili sme javascript nové veci. ty vieš do kódu pridať funkcie, ktoré si sama napíšeš. ale rovnako tam vieš pridať funkcie, **ktoré napísal niekto iný**. napríklad formou knižníc. knihoven. libraries.

na hodine som ukázal ako použiť https://animate.style a https://day.js.org/. kód náješ v #soubory, konkrétne `03b-animacie-produkty.zip` a `03c-dayjs-format-datumu.zip`. ale tieto dve knižnice tu neboli dôležité. **dôležitý bol postup**. tie dve boli príklad, na ktorom som ukázal, ako môžeš pracovať s dokumentáciou.

NPM INSTALL vs CDN

každá knižnica má svoju **dokumentáciu**. každá má sekciu o tom, ako knižnicu nainštalovať. budeš tam vidieť príkaz ako tento `npm install chart.js`.

tomu sa budem venovať na ďalších hodine. ak to chceš už teraz, ja som o tom už **NAPÍSAL DETAILNÝ ČLÁNOK TU**: https://gist.github.com/yablko/47cb93224a8e297df3c5f8776ede2c46

prečítaj si ho. `npm` je program, ktorý v dnešnom svete **potrebuješ** pre reálnu prácu s javascriptom. text vysvetlí o čo ide a ako ho nainštaluješ do svojho počítača. sprav to. potom získaš schopnosť používať `npm install` príkazy, ktoré budeš vidieť v snáď každej dokumentácii.

CDN je skratka pre Content Delivery Network. v budúcnosti budeme používať NPM. zatiaľ nepoužívame NPM. a teda ak chceš používať nejakú knižnicu, **hľadaj CDN spôsob inštalácie** 🙂

každá naša appka začína súborom `index.html`. k nemu vieš pripojiť CSS súbory pomocou `<link>` a JS súbory pomocou `<script>`. CDN je v podstate odkaz na externý .css alebo .js súbor, ktorý takto pripojíš do svojho `index.html` súboru. príklad znova nájdeš v tých `03b-animacie-produkty.zip` a `03c-dayjs-format-datumu.zip`.

DOKUMENTÁCIA A MLÁTENIE HLAVY O STENU

každá knižnica má na webe svoju **dokumentáciu**. ty **budeš** používať knižnice. takže tvoja práca je nájsť dokumentáciu. v nej nájdeš sekciu o inštalácii. a nájdeš tam vysvetlenie všetkého. a nájdeš tam príklady kódu a toho, ako sa daná knižnica používa. ty nepíšeš kód z hlavy. ty v dokumentácii nájdeš, aký kód máš písať.

opakujem: ty sa naučíš základy programovania a základy jazyka javascript. to boli prvé 4 hodiny tohoto kurzu. tých magických 7 konceptov. to sa naučíš. a potom zvyšný kódu nepíšeš z hlavy. ten si vzďy dohľadávaš.

niektoré dokumentácie sú lepšie, iné sú horšie. **každá jedna z nich bude frustrujúca**. budeš si mlátiť hlavu o stenu. je to bežná súčasť tvojej práce. prejde 5 rokov, 5 rokov budeš programovať, po 5 rokov si otvoríš ďalšiu dokumentáciu a budeš si mlátiť hlavu o stenu. **je to bežná súčasť tvojej práce.** pretože programovanie ťa neustále núti používať nástroje a knižnice a postupy, ktoré sú pre teba nové. veci, ktoré sú nové, sú ťažké a frustrujúce.

ľudia majú predstavu, že programátor si ráno sadne za počítač a magicky začne písať správne znaky vždy v správnom poradí do prázdneho súboru a potom ide domov. vôbec. programovanie je neustále riešenie problémov. furt niečo nefunguje, furt musíš používať novú vec, furt sa babreš v nastaveniach nejakého programu, ktorý furt nefunguje a googliš, čítaš, ľudí sa pýtaš a mlátiš si hlavu o stenu a skúšaš návody a postupy a 95% z nich absolútne nefunguje, ale potom narazíš na tých 5%, ktoré fungovať začnú a buď zdvihneš ruky vo víťazstve alebo rozmlátiš počítač kladivom. **a toto je programovanie.** a nezáleží na tom, či kódiš mesiac alebo 15 rokov.

keď ľudia začínajú, majú pocit *"ja neviem čo robím, vôbec mi to nefunguje, asi som blbý a nikdy to nebudem vedieť"* **nope!** zvykni si na tento pocit, lebo ho majú **všetci**. ak zotrváš a budeš to robiť 10 rokov, jediná zmena je, že možno zmizne tá *"a nikdy to nebudem vedieť"* časť vety. zvyšok zostáva.

pretože tvoja robota je **neustále čítať dokumentáciu a externé zdroje a snažiť sa rozchodiť a pochopiť nové veci.** každá dokumentácia bude mať ukážky kódu a príklady a vysvetlenia. a mnohé z nich budú fajn a mnohé budú vágne a takmer nepochopiteľné. a tvoja úloha je skúšať tie kusy kódu písať a meniť a lepiť a vyhadzovať a skúšať nové tak dlho, kým to nezačne fungovať.

ja to opakujem dokola: keď si programátor ráno sadne za počítač, nevie, čo má písať. programovanie je, že 95% času ti to nefunguje. a si nasraný. a v konečnom dôsledku programovanie je kreatívna robota. a socha je z toho až na konci. keď to postupne opracuješ. dovtedy je to len hnusný kameň.

AI CHATBOT KAMARÁT?

čiže tvoja robota bude používať knižnice. a čítať dokumentáciu. a snažiť sa ju pochopiť. a zliepať kusy kódu z nej. občas to bude fajn fungovať, často to bude frustrujúce. **tu chatbot ako chatgpt alebo copilot môže byť fajn pomoc**. on tiež len "prečíta" tú dokumentáciu (plus možno pridá pár ďalších zdrojov), ale preformuluje to. do možno pochopiteľnejšieho jazyka? respektíve vieš mu dať konkrétnejšiu otázku a on odfiltruje nepotrebné veci. v dokumentácii je toho veľa, on ti vie pomôcť vytiahnuť tú časť, ktorú potrebuješ.

OVERENÝ ZOZNAM KNIŽNÍC?

v #chate staršej verziu kurzu raz padla otázka, že či existuje zoznam knižníc, ktoré sú overené a správne. moja odpoveď:

ja osobne neviem, či existuje nejaký definitívny, "objektívny" zoznam 🙂 dá sa napríklad googliť niečo na štýl *"most used npm packages"*, určite bude existovať kopa článkov, ale asi žiaden objektívny zoznam. zoraď si tie články podľa dátumu (alebo vyhložene hľadaj s textom "2025"), otvor niekoľko z nich a uvidíš, že niektoré odpovede sa budú opakovať. "komunita" sa každých pár rokov akoby zhodne na tom, ktorý je ten správny a ten sa teraz bude používať najčastejšie.

napríklad pri dátumoch to dlho bol `moment.js` ale keď si teraz otvoríš moment.js tak priamo na ich webe nájdeš text "*There may be better modern alternatives.*" a keď skúsiš hľadať "moment.js modern alternative" tak sa veľmi často bude opakovať práve `date-fns` alebo `dayjs` 🙂 ďalšia vec je, že programovacie jazyky sa vyvíjajú. v minulosti boli knižnice potrebné na veci, ktoré medzičasom pribudli do javascriptu samotného. a daná knižnica je teda nepotrebná.

všetky z nich budú mať svoj github a svoju podstránku na https://www.npmjs.com. na githube napríklad vieš zistiť, kedy bol kód knižnice naposledy upravený. ak bol naposledy upravený pred povezdme 2+ rokmi, je možné, že o knižnicu sa už nikto nestará a je na čase hľadať alternatívu. na npm zasa vieš zistiť, koľkokrát bola tento týždeň stiahnutá. keď tam vidíš, že každý týždeň je stiahnutá povedzme milióny krát, vieš, že je to populárna a používaná knižnica.

to sú všetko len také indície ale... 🙂 momentálne to vypadá tak, že `date-fns` alebo `dayjs` sú pre dátumy dobrá voľba. ale toto je vec na web svete, že **sa neustále (a príliš rýchlo) vyvíja a mení a je ťažké ukázať na definitívne zoznamy**. ale google, a niektoré odpovede sa budú často opakovať a pravdepodobne to bude ok 🙂

ja som do https://discord.com/channels/1147117106450681876/1369263008051892325 vymenoval niekoľko knižníc, ktoré sú overené a roky používané. ale ako vždy, tento svet je frustrujúco skúpy na do-kameňa-vytesané nemenné odpovede.