07 VYRÁBAME HTML, document object model

naša žltá appka zobrazuje 3 šedé boxy. nazval som si ich `card`, každý z nich je takzvaný `card`. na hodine som napísal javascript funkciu, ktorá vie vytvoriť nový card a "akoby" ho pridať na správne miesto do HTML kódu a tým pádom sa zobrazí na obrazovke:

```
function createNewCard(title, content) {
  const container = document.querySelector('.card-list');
  const newCard = document.createElement('li');

  newCard.innerHTML = `
    <h3>${title}</h3>
    <p>${content}</p>
  `;

  container.appendChild(newCard);
}
```

ak začínaš:

> je v poriadku, ak zatiaľ nerozumieš tomu, ako presne tá funkcia funguje. čo tie riadky v jej tele znamenajú. je to ok. ukazoval som ti funkcie ako `alert()` a `toFixed()` a ty ich používaš. akceptuješ, že existujú a vieš približne, aké je ich úloha a používaš ich. ber to tak, že toto je teraz funkcia, ktorá existuje v javascripte a môžeš ju používať 🙂 úloha `alert-u` je, že ti vyskočí okno s textom v prehliadači. úloha `createNewCard` je, že ti na obrazovke pribudne ďalší šedý box. ďalší card. ďalší element.
>
> keď používaš knižnice, často používaš funkcie, ktorým neruzumieš. ale vieš, čo robia. tak ich používaš. úprimne? nevieš, ako funguje `toFixed` 🙂 ale vieš, čo `toFixed` robí, tak to používaš. funkciu môžeš používať a ako presne funguje, si naštuduj vtedy, keď potrebuješ. možno, keď budeš robiť niektoré so https://discord.com/channels/1147117106450681876/1348937779983421503 😉

teraz ak v kóde napíšem:

```
createNewCard('ahoj', 'toto je krasny novy text');
```

tak na obrazovke pribudne nový šedý card, s nadpisom "ahoj" a ten druhý text bude v popise.

doteraz sme výsledky kódu vypisovali len do konzole, cez `console.log()`. ale konzola je len pre nás. pre programátorov. a teraz konečne vieme veci zobraziť aj bežnému, reálnemu používateľovi našej appky.

AUTOMATIZÁCIA

v prvom rade neviem, či automatizácia je správne slovo. ale pointa: pri webových appkách ty si vieš otvoriť `index.html` súbor a napísať tam 5 nových cardov ručne. ok. ale čo keď ti kolega dá zoznam 30 tisíc cardov, ktoré sa majú zobraziť? **o tom je programovanie.** ty napíšeš pár zvláštnych riadkov kódu, ktorých úloha je vyrobiť nový card. a teraz nezáleží na tom, či ich máš vyrobiť 5 alebo 5 miliónov. keď máš funkciu, tá funkcia sa spustí 5-milión-krát a ty máš vyložené nohy. chilluješ. **programovanie je, že ty povieš počítaču, čo má robiť a on to potom spraví hrozne rýchlo.** a našom prípade mu to hovoríme jazykom javascript.

potom tú našu funkciu vieš použiť napríklad na...

VYROBENIE CARDOV Z EXTERNÝCH ÚDAJOV

```
// fetch ziska z tejto adresy recepty
fetch('https://dummyjson.com/recipes')
    .then(res => res.json())
    .then(data => {
        // a pre kazdy jeden recept
        data.recipes.forEach(food => {
            // vyrobime novy card
            createNewCard(food.cookTimeMinutes, food.name);
        });
    });
```

👉 tu používame **cyklus**. bližšie vysvetlím na ďalšej hodine. a už teraz to máš vysvetlené tu https://discord.com/channels/1147117106450681876/1367270818803089519

keď mám funkciu, ktorá vyrobí nový card, tak ju viem použiť univerzálne. viem si vyrobiť card s mojimi vlastnými textami. ale čo je dôležitejšie: tie texty môžu prísť od kolegu. zo servera. zo súboru. z databázy. z excelu... doslova na tom nezáleží. ja mám proste dáta a potrebujem ich zobraziť na obrazvku. **a teraz na to mám funkciu.** a viem robiť reálne veci.

bonusové, hlbšie info:

DOM (DOCUMENT OBJECT MODEL)

súbor `index.html` je štart každého webového projektu. keď otvoríš ktorúkoľvek webstránku v prehliadači, ten začne sťahovať `index.html` súbor. tak funguje internet. prehliadač stiahne `index.html`, postupne ho číta po riadkoch a vytvára si štruktúru elementov: v kóde tej našej žltej webstránky evidentne existuje napríklad `main` element.

`main` element má tzv. potomkov: `ul`, `li`, `h3`... a naopak `h3` má rodičov: `li`, `ul`, `section`, `main`...

prehliadač si vytvorí takého pavúka. taký rodokmeň. prehľad o tom, ktoré HTML elementy existujú a aké sú medzi nimi vzťahy. tejto štruktúre hovoríme **DOM (Document Object Model)**.

web funguje na báze **rodičov** a **potomkov**. keď chcem vytvoriť nový element, potrebujem nájsť rodiča, do ktorého tento nový element pridám.

elementy sú matrioškoidné - zavrstvené do seba. elementy patria do iných elementov. u nás elementy `h3` a `p` **patria do** elementu `li`. `li` obsahuje tieto elementy. `li` je ich rodič. `h3` a `p` sú jeho deti.

JAVASCRIPTOM VYTVORÍME NOVÉ HTML ELEMENTY

a teda keď chcem javascriptom vyrobiť nový HTML element, môžem si takto rozpísať jednolivé potrebné kroky:

```
// najdem rodica, do ktoreho chcem pridavat nove HTML elementy

// pripravim si novy HTML element (li)

// novy element bude mat nadpis (h3) a text (p)

// vlozim ho do rodica
```

každú veľkú úlohu si vieš rozbiť na niekoľko maličkých. niekedy dostaneš zadanie a je toho strašne veľa. a je to strašidelné. pokús sa nájsť ten najmenší kúsok, ktorý vieš spraviť. keď máš prvý kúsok, oveľa ľahšie nájdeš druhý. nikdy neromýšľaj nad celým problémom. nesnaž sa celý problém držať v hlave. nájdi si tú jednu časť problému, ktorú dokážeš vyriešiť. tam začni. keď máš za sebou prvý rok, druhý bude jednoduchší.

ChatGPT?

čiže máš 4 úlohy, ktoré potrebuješ splniť. možno nevieš ako. možno sa spýtaš AI *"ako v javascripte vytvorim novy LI element a pridam don HTML kod a ako ho potom vlozim do existujuceho UL elementu?"* a možno dostaneš kód ako tento:

```
// najdem rodica, do ktoreho chcem pridavat nove HTML elementy
const cardContainer = document.querySelector('.cards ul');

// vytvorim novy HTML element (li)
const newCard = document.createElement('li');
newCard.innerHTML = '<h3>' + title + '</h3><p>' + text + '</p>';

// novy element bude mat nadpis (h3) a text (p)
let title = 'Nový nadpis';
let text = 'Toto je kúsok textu. Je krásny. Je priam nádherný. Čítaj ho
očami a užívaj si ako znie v tvojich ušiak, ak ho niekto vysloví.';

// vlozim ho do rodica
cardContainer.appendChild(newCard);
```

pamätaj si: tieto riadky majú jednu spoločnú misiu. jednu spoločnú úlohu. ich úloha je vytvoriť nový LI element a vložiť do ho spávneho UL zoznamu. dá sa povedať, že ich úloha je **vytvoriť nový card**. tento kód je trochu iný, ako ten môj vo výslednej funkcii. ale chcem ilustrovať, že keď nevieš ako presne niečo spraviť, máš prístup k celému internetu a ten je plný informácií.

mimochodom tomu `.cards ul` zápisu hovoríme `selector`. je to identické ako v CSS. ak v CSS chcem elementu, ktorý drží všetky cardy nastaviť výzor, napíšem `.card ul { ...sem dám výzor... }`.

FUNKCIA JE KUS KÓDU, KTORÝ MÁ JASNÚ ÚLOHU

pamätáš? keď máš riadky kódu, ktoré patria k sebe, ktoré majú jednu spoločnú úlohu: **sprav si z nich funkciu.** daj jej názov, ktorý tú spoločnú úlohu vystihuje. úloha týchto riadkov je vytvoriť nový card. funkcii dám názov `createNewCard`. a keď mám funkciu, viem ju používať opakovane.

hotovú funkciu máš hore v tomto texte. práve si javascript naučil novú vec. javascript získal schopnosť vytvárať nové cardy. napríklad takto:

```
createNewCard(
  'Prvy nadpis',
  'Prvy text <strong>toto bude tucne</strong>'
);

createNewCard(
  'Druhy',
  'Ahahaha'
);

createNewCard(
  'Ja som 3',
  'A tiez existujem.'
);
```

a vytvoril som 3 nové card elementy! takým kódom, z ktorého ma nebolia oči. a srdce a duša! pretože ak vidím `createNewCard()` poviem si *"aha... toto asi vytvorí nový card."*

a to je okamžite pochopiteľné. **a to chceš.**

navyše ak budeš potrebovať spraviť zmenu v kóde, stačí ju spraviť na jednom mieste: v tele funkcie. je vo vytvávarní nového cardu chyba? ok, viem **presne**, kde ju nájdem. vo vnútri `createNewCard`.

takže ak si vytvorím novú funkciu, viem ju používať znova a znova. ak jej dám pekný názov, tak nemám bordel. a ak existuje chyba, viem presne, kde ju nájdem ✨