Úlohy pro výuku JavaScriptu z předmětu Programování 1

FILIP FRANK, MIROSLAV ZÍKA
Fakulta pedagogická ZČU v Plzni

Cíle předmětu Programování 1 pro vzdělávání

Předmět *Programování 1 pro vzdělávání* si klade za cíl seznámit studenty KVD FPE ZČU v Plzni se základy programování. Studenti jsou uvedeni do problematiky programování v programovacím jazyku JavaScript a seznámí se po teoretické i praktické stránce s elementárními základy programování se zvláštním zřetelem ke vzdělávání [1].

Tradičním problémem bývá velmi rozdílná úroveň jednotlivých studentů. Tento fakt přičítáme šířce oborů středních škol, jejichž studenti se mohou rozhodnout studovat pedagogickou fakultu (např. všeobecná gymnázia, obory potravinářské, zemědělské, elektrotechnické aj.). V průběhu seminářů je pak na vyučujícím, jakým způsobem tento problém vyřeší. Jako vhodný postup se jeví dát studentům možnost volby. Právě vzhledem k potenciálnímu užití z předmětu získaných poznatků ve vzdělávání jsou úkoly navrženy v rámci zásad a očekávaným výstupům v RVP ZV. Studenti jsou vedeni k rozdělování programů na dílčí řešitelné části, řešení problému více způsoby [2]. Tento výstup, resp. zkušenosti s řešením, je naplňován také v předchozím předmětu *Algoritmizace pro vzdělávání*, na který je navázáno právě předmětem programování 1.

Jednou z variant řešící různorodou úroveň studentů je kooperace nadanějších, zkušenějších studentů, se slabšími. Díky tomu má vyučující šanci individuálně pomoci studentům zvládající výuku s velkými obtížemi. Studenti, kteří mají jen drobné nejasnosti, je obvykle vyřeší právě vzájemnou spoluprací. Zmíněná forma výuky navíc pomáhá fixovat poznatky studentům s vyššími zkušenostmi a současně rozvíjí jejich pedagogické kompetence.

Druhou možností, kterou dostávají nadanější studenti, je práce na seminární práci, která je součástí zápočtu. Tento způsob zajistí aktivitu nadanějších studentů. Vyučující se tak může plně věnovat ostatním studentům a zodpovídat všechny dotazy.

Pro splnění předmětu je zapotřebí splnit zápočtové testy a obhájit semestrální práci. Zápočtové testy koncipujeme tak, aby v nich studenti využili postupy osvojené v průběhu seminářů. Při testu mohou používat internet a hledat pomocí něho způsob, jak zadané problémy v testu řešit. Tuto možnost jsme dali studentům proto, že při programování skutečného projektu budou tyto zdroje s velkou pravděpodobností stejně využívat. Jedinými pomůckami, které studenti využít nesmí, jsou komunikační platformy a příklady ze seminářů v jakékoli formě. Zákaz využití příkladů ze semináře je z důvodu, že problémy řešené v testu jsou typově podobné těm řešeným v rámci seminářů. Po získání zápočtu jsou studenti připuštěni ke zkoušce – ta se opět skládá z praktického příkladu, který vyučující následně zhodnotí. Od roku 2022 byla přidána teoretická část zkoušky, která studenty může zvýhodnit v praktické části. Čím budou studenti v teoretické části úspěšnější, tím méně bodů musí následně získat v praktické části.

Po absolvování předmětu získá student následující znalosti a dovednosti:

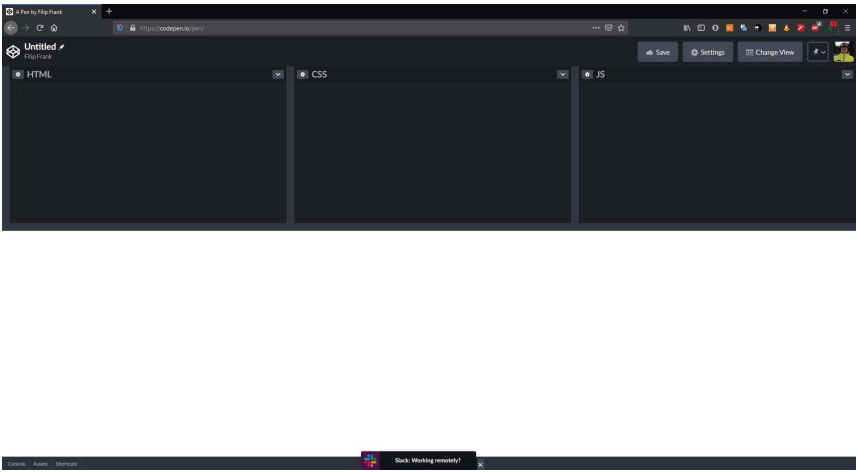
- řídí se příslušnými principy při využívání záznamu či jeho alternativy s ohledem na programovací jazyk,
- rozpozná možnosti, které nabízí použití různých typů programovacích jazyků,
- používá programovací jazyk v souladu s jeho syntaxí a sémantikou,
- řeší úlohy vztažené k vstupně/výstupním operacím (např. z klávesnice, na obrazovku, ve vztahu k souboru apod.)
- používá správné struktury programovacího jazyka spojené s logickými operátory,
- vybírá při programování správný cyklus pro řešení příslušného problému,
- odlaďuje program,
- využívá vhodné metody včetně parametrů [1].

Prostředí codepen.io

Prostředí *CodePen* umožňuje vytvářet webové stránky, přičemž okamžitě vykresluje jejich vzhled bez nutnosti znovunačtení stránky, čímž dává okamžitou zpětnou vazbu. Prostředí je rozděleno do 4 částí, jak vidíme na obr. 1.

V horní polovině obr. 1 jsou tři „boxy“ sloužící k zapsání daného kódu. Při programování pak uživatel přistupuje k jednotlivým částem izolovaně, podobně jako při linkování externího CSS nebo JavaScriptu. Výhodou zde

je, že nemusíme samotné napojení na HTML řešit, vše je již připravené. Z řad studentů se objevovaly námitky, že tímto způsobem si bez předchozích znalostí webových technologií neosvojí dříve zmíněné propojení (tj. využití tagů link a script). Zde narážíme na nevýhodu využívání JavaScriptu a další důvod volby prostředí CodePenu. Naším cílem není získat znalosti z oblasti tvorby webový stránek, zejména práce s HTML a CSS. Naším cílem je JavaScript, proto si můžeme dovolit vynechat propojování a přejít rovnou k tvorbě. Ze stejného důvodu mohou i příklady ze seminářů v následující kapitole působit trochu fádně, až zastarale. Naším cílem je totiž pochopení funkce na pozadí a řešení programátorských problémů, z nichž uživatel vidí pouze výsledek. V záhlaví každého z boxů je pak šipka, která umožňuje spustit obdobu debugeru, jenž červeně zvýrazní možné chyby v kódu.



Obr. 1 Prostředí CodePen

V dolní polovině student okamžitě vidí vykreslený výsledek svého kódu. V případě potřeby je možné autorefresh vypnout. To se může hodit při práci s cykly, kdy je nutno předejít opětovnému spuštění nedokončeného cyklu a jeho zacyklení.

Poslední z výhod prostředí CodePen je automatické ukládání. Kód práce totiž není uložen na lokálním počítači, ale webu Codepen.io, takže k němu má student kdykoli a odkudkoliv přístup. Vytvořený kód je ovšem možné exportovat do souboru .zip či publikovat na platformě GitHub.

Pokud bychom hledali možné nevýhody prostředí CodePen, mohly by spočívat především v problémech při zacyklení programu. V tomto případě zde totiž není žádný nástroj pro přerušení. Je potřeba počkat na prohlížečem vyvolané zastavení běhu z důvodu nadměrného využívání výkonu. Zacyklený kód navíc zpomalí celý počítač, takže je před zareagováním prohlížeče obtížné cokoli dělat. Druhou nevýhodou, se kterou jsme se však za čtyři roky setkali jen jednou, je přetížení serveru CodePen. V takovém případě je pak pro studenty problém se i přihlásit. Pokud se jim to však povede, práce už je dále plynulá.

Úlohy ze cvičení

Následující úlohy byly s drobnými obměnami používány již čtvrtým rokem ve výuce předmětu *Programování 1 pro vzdělávání*.

V následujících kapitolách si vždy představíme konkrétní úlohu, její cíl a postup řešení. Následně se budeme zabývat opakujícími se chybami, které se napříč ročníky objevují. Chyby si popíšeme a pokusíme se odhalit, na čem se zakládají.

Výsledky zde uváděné se zakládají na dlouhodobém, přímém, nestrukturovaném, zúčastněném pozorování. Švaříček tvrdí, že pro studium školní třídy je zúčastněné pozorování vhodné, protože nijak zásadně nenarušuje sociální interakce a edukační procesy ve škole. O nestrukturovaném pozorování mluvíme proto, že v průběhu let jsme sledovali, jak si studenti při práci počínají a jaké chyby dělají. Necílili jsme však na konkrétní jev. Až později se ze získaných dat daly identifikovat opakující se chyby a chování, které u studentů napříč lety pozorujeme [3].

Nahodile pak probíhaly nestrukturované rozhovory se studenty. Zde se jednalo zejména o studenty, kteří udělali opakovanou chybu, nebo naopak nechybovali tam, kde chybovali prakticky všichni. Využili jsme zde otevřené otázky, které vždy směřovali na konkrétní námi pozorovaný jev. Díky otevřeným otázkám jsme měli možnost zjistit úhel pohledu studenta na řešení úlohy, samotnou úlohu i jeho postoj k chybě a práci s ní.

Kompletní seznam úloh v předmětu Programování 1 pro vzdělávání

1. Úvod do HTML, JS a CSS

   Úloha uvádí studenty do problematiky propojení HTML, JS a CSS. Studenti vytváří bloky stránky, které následně upraví kaskádovými styly. Na závěr se po stisknutí tlačítka mění obrázky na stránce pomocí JS.

2. Smilíci

   Úloha je zaměřena na práci s proměnnými. Využívají se zde různé přenosy a získávání proměnných. Mimo jiné je zde možno vidět, že různé cesty vedou ke stejným výsledkům, ovšem s různou efektivitou.

3. Diktát

   Úloha je zaměřena na práci se stringem, tj. textovým řetězcem. Studenti upravují uživatelem zadaný text. Z tohoto textu odstraňují zvolené části. Zároveň musí zajistit, aby se odstraněné části daly se zvýrazněním vrátit. Součástí je ošetření vstupu pomocí podmínek.

4. Hádání čísel

   Úloha je zaměřena především na cyklus while. Studenti vygenerují náhodné číslo z rozsahu zadaným uživatelem a následně je pomocí modálního okna uživatelem hádána hodnota čísla. Uživatel dostává zpětnou vazbu, zda se trefil, nebo jestli jeho odhad moc velký, či malý.

5. Ovocné hrátky s polem

   Úloha je zaměřena na seznámení studentů s polem. Studenti vytvoří pole a naplní jej ovocem. Uživatel musí mít možnost nechat si vypsat konkrétní položku z pole, přidávat věci do pole, mazat z něho položky nebo celé pole vypsat.

6. Auta a cykly

   Úloha se zaměřuje na cyklus FOR a práci s polem, ve kterém jsou objekty. Díky objektům studenti zjistí, že funkce předdefinované v JavaScriptu nelze použít vždy. Naučí se manuálně procházet a pracovat s polem pomocí zmíněného cyklu. Zároveň budou pracovat s jednotlivými vlastnostmi objektu. Díky tomu studenti vidí výhody použití objektu oproti využití několika proměnných.

7. Úvod do canvasu a odrážecí kulička

   Úloha si klade za cíl seznámit studenty s možnostmi canvasu (tj. kreslícího plátna). Začínáme na jednoduchých čarách a objektech, které studenti obarvují a pozicují. Následně přecházíme k jednoduché animaci postavené na rychlém vykreslování a mazání.

8. Náhodný graf

   Úloha prohlubující předchozí úvod do canvasu ukazuje jeho možné praktické využití při reprezentaci hodnot. Pro zopakování studenti generují daný počet náhodných hodnot omezeného rozsahu, z nichž následně

určí maximum, minimum a aritmetický průměr. Každou hodnotu vykreslí do sloupcového grafu, kde označí maximum zeleně a minimum červeně. Celý graf protíná přímka znázorňující průměrnou hodnotu.

9. Prohazování obrázků na tlačítko s možností textového vyhledávání

   Jedná se o první typovou úlohu používanou u zápočtového testu. Studenti musí vytvořit obdobnou stránku, jako u příkladu *Úvod do HTML, JS a CSS*. Přidaný je třetí obrázek, v původní úloze jsou jen dva. Dále zde studenti musí vyřešit textové zadávání pro změnu na konkrétní obrázek. Textové zadávání řeší pomocí přijímaných konkrétních slov. Pod obrázky dále zobrazují historii textově zadávaných obrázků.

10. Neodbytný vrchní

    Úloha vznikla jako opravný zápočtový test. První zápočtový test je postaven na úloze *Auta a cykly*, která byla změněna na účtenku. Úloha je jinak úplně stejná. Účtenka však vykazovala vysokou míru neúspěchu, proto byla přesunuta až do zkouškového testu.

    Neodbytný vrchní je simulátorem pobytu v hospodě. Po usednutí se zákazníka ujme vrchní, který se neustále ptá na jeho objednávku. Toto je řešeno modálním oknem a cyklem. Vrchní skončí s dotazy až ve chvíli, kdy uživatel klikne na tlačítko Zrušit. Poté studenti řeší odchod a formu placení – buď jednotlivých položek, všeho najednou či dokonce pokus o odchod bez placení.

11. Spojnicový graf s uživatelským vstupem

    Úloha je postavena na znalostech získaných u práce se sloupcovým grafem. Uživatel má možnost zadávat omezené hodnoty, které bude graf schopen vykreslit. Hodnoty se ukládají do pole vypisovaným pod generovaným spojnicovým grafem. Ze všech hodnot si studenti určí maximum, minimum a aritmetický průměr. Uživatel musí mít možnost zvolené hodnoty mazat zadáním pořadí hodnoty.

S ohledem na rozsah celého článku se budeme v tomto příspěvku zabývat pouze třemi prvními úlohami.

1. Úvod do HTML, JS a CSS

Cvičení studentům představuje jednotlivé části webu, kterými jsou v našem případě HTML, JavaScript a CSS. Studenti v úvodní úloze vnímají rozdílné úlohy jednotlivých jazyků.

Postupně jsou představeny základní tagy a jejich atributy. Studenti si nejprve připraví obsah HTML, který ale prozatím není formátován CSS ani oživován pomocí JavaScriptu. Úvodní obrázek je nalinkován pomocí URL cesty přímo do HTML.

Dalším krokem je přidání CSS. Nejprve nastavíme veškerý zobrazovaný text na Arial a obarvíme jej na zvolenou barvu. Díky tomu se studenti mohou seznámit s rozdílem mezi CLASS a ID, kdy pochopí, že pomocí ID mohou upravit konkrétní text, i když je v rámci dané třídy. Obarvení pomocí ID použijeme u jména, které obarvíme na bílou barvu. Dále zvýrazníme tlačítka zeleným rámečkem. Rodičovský element div, který nám sdružuje všechny prvky, obarvíme zvolenou barvou a všechny jeho prvky vystředíme.

Obr. 2 Úloha Úvod do HTML, JS a CSS

Kód úlohy je k dispozici na https://codepen.io/filip-frank/pen/wELGQv.

Následně se přesuneme na JavaScript, který v této úloze bude velmi jednoduchý. Připravíme si funkci `pikachu()`, kterou navážeme na tlačítko se stejným nápisem. Při jeho stisku se původní obrázek změní na obrázek jmenovaného pokémona. Takřka identickou funkci si připravíme pro druhé tlačítko, kde bude jiný obrázek. Tímto způsobem můžeme přepínat mezi dvěma obrázky pomocí dvou tlačítek. Lze zde také zařadit změnu stylu. Typicky měníme barvu textu, podtržení a podobně.

Závěrem tohoto úkolu se studenti velmi okrajově seznamují s podmínkou `if`. Vzhledem k tomu, že jde o první úlohu, problematiku řešíme zejména algoritmicky a kód si v tuto chvíli spíše ukážeme. Cílem je vytvořit tlačítko `Změň`, které bude cyklicky měnit obrázky mezi sebou. Studenti sami musí přijít na logiku, že pokud bude zobrazen obrázek X, pak tlačítko změní obrázek na Y a naopak.

Pozorované chyby

V představené úloze studenti příliš nechybují, neboť se jedná zejména o práci s HTML a CSS. Úroveň první úlohy je ve všech částech (HTML, CSS i JS) velmi nízká, nejčastější chyby jsou proto spojeny spíše s překlepy. Výjimečně se objeví chyba, kdy student zaměňuje atributy CLASS a ID. V sekci s JavaScriptem se však opakuje častá chyba, kdy studenti kontrolují, zda je zobrazen obrázek X a následně při vyhodnocení dosadí obrázek X místo Y. Vidíme zde tedy, že špatně vyhodnocují podmínky a na kladné větvi se chovají, jako na záporné.

2. Smilíci (Práce s proměnnými, parametrem a následně s podmínkami)

V úkolu *Smilíci* se studenti seznamují s možnostmi JavaScriptu: jak přenášet informaci pomocí proměnné a parametru nebo jak ji získat přímo z HTML prvku. Vzhledem ke snaze, věnovat co největší čas JavaScriptu, je z řešení zcela vynecháno stylování pomocí CSS.

Na začátku studenti vytvoří HTML strukturu podle hotové předlohy. I když je HTML značkovací, ne programovací jazyk, nepovažovali jsme za vhodné dávat studentům vypracované HTML. Je důležité, aby se sami naučili propojovat různé prvky HTML, v tomto případě zejména tlačítka, s JavaScriptem sami.

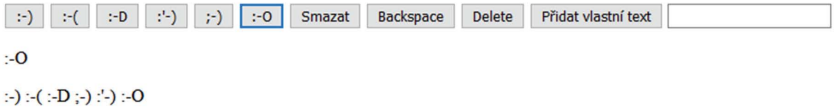
Po vytvoření struktury, bez jakéhokoli nalinkování, přichází první úkol, kdy se studenti seznámí s prací s proměnnou. Jejich úkolem je definovat si proměnnou se stringovou hodnotou požadovaného smilíka. V prvním případě se jedná o usměvavého smilíka. Po kliknutí na dané tlačítko by se měl požadovaný smilík zobrazit pod řadou tlačítek. Zároveň by se měl smilík přidat na konec řady (pod obslužnými tlačítky), čímž se nám vytváří historie stisknutých tlačítek. V tomto případě vystačíme s klasickým stringem. Studenti tak musí vyřešit, jak zobrazit nového smilíka a současně ho vhodně přidat ke dříve zobrazeným.

Studenti mají dále za úkol u tlačítek měnit způsob, jakým získávají hodnotu vypisující se pod tlačítka a do zobrazené historie. Postupně si tak vyzkouší možnost získání textu z popisku tlačítka, přenášet jej pomocí parametru funkce nebo dokonce zapisovat pouze text bez použití proměnné. Díky tomuto postupu studenti porovnávají náročnost, univerzálnost a pochopitelnost jednotlivých řešení. O nejvhodnějším způsobu je v závěru hodiny vedena diskuse. Obvykle mají studenti tendence upínat se pouze k jedné variantě, jako té nejlepší. Tradičně jí bývá přenos parame-

trem, který sice studenti považují za nejsložitější, ale zároveň v něm vidí velkou univerzálnost.

V další části úkolu pak je nutno implementovat tlačítko Smazat, které maže celou historii. Tlačítko Backspace, které maže znak po znaku historii od konce, a tlačítko Delete mazající řadu od začátku. Tato tlačítka jsou v další hodině ošetřována tak, aby v případě, že není co mazat, na tuto skutečnost upozornily.

Posledním úkolem v této úloze je možnost přidat vlastní slovo, nebo klidně jen znak. Zde musí studenti nejen znak přidat stejným způsobem, jako to dělali u smilíků (tedy na konec řady i posledního stisknutého), v rozšíření musí také validovat vstup pro kontrolu, zda uživatel skutečně něco zadal. Pokud by ve vstupním poli bylo prázdno, mělo by to být opět ohlášeno.



Obr. 3 Úloha Smilíci

Kód úlohy je k dispozici na https://codepen.io/filip-frank/pen/dgGZYy.

Pozorované chyby

V tomto úkolu často se vyskytují chyby, zejména v jeho první části se zobrazením jednoho smilíka, pramení spíš ze syntaktické stránky. Problém je poměrně jednoduchý – je potřeba přenést informaci. Většinou se jedná o chyby, kdy studenti otáčí přiřazování do proměnných. Místo smilíka se do proměnné pokouší přiřazovat neexistující proměnnou do smilíka (tj. dochází k prohazování levé a pravé strany přirazování).

Více zajímavé je řešení zobrazování historie. Zde je obvyklou chybou postup, kdy studenti jednoduše zopakují přiřazení smilíka do řady stejně, jako když jej jen zobrazovali. Výsledkem pak je, že místo prodlužující se řady je zobrazován jen právě stisknutý smilík. Je to proto, že je potřeba vždy vzít data z existujícího záznamu (je možné si pro ně připravit speciální proměnnou, ale není to nutné) a přidat k nim nejnovější stisknutou hodnotu. Po tomto zjištění obvykle studenti již řeší jen problém pořadí proměnných, aby byla poslední stisknutá data na konci.

V části s mazáním se studenti často zaseknou při tvorbě podmínky.

Neví, jakým způsobem kontrolovat délku historie. Obvyklé chybné pokusy jsou porovnání vůči ničemu tedy „`if rada=‘‘`, porovnání vůči nule „`if rada=0‘‘` nebo vůči mezeře „`if rada=‘ ‘‘‘`. Porovnání vůči prázdnému řetězci, které je v tomto případě správné, napadne jen málo studentů. Tento poznatek však bez problémů využijí ve chvíli kontroly uživatelského vstupu u finální části úlohy, v němž uživatel zadává vlastní text. Zde se již chyba neopakuje.

Velmi častou syntaktickou chybou, která se objevuje napříč celou úlohou, je přiřazení funkce k danému tlačítku. Jde o chybu, kterou si studenti jednoduše velmi často neuvědomí. V části s JavaScriptem je vše vytvořeno správně, ale chybí navázání na tlačítko pomocí atributu `onClick` (s možností přiřazování pomocí konstrukce `addEventListener` jsou studenti seznámeni později).

3. Diktát

Smyslem úlohy nazvané *Diktát* je seznámit studenty s prací se stringem, jeho vlastnostmi a možnostmi. Hlavním úkolem je vytvořit zadávací pole (textarea) pro uživatele, kam vloží či napíše svůj text a následně ho přebere do jiného html prvku. V něm bude možné z textu odstraňovat požadovaná písmena tak, aby vznikla klasická češtinářská doplňovačka. Mělo být možné také zobrazit řešení diktátu, kdy se hledané prvky zobrazí červeně a velkým písmem.

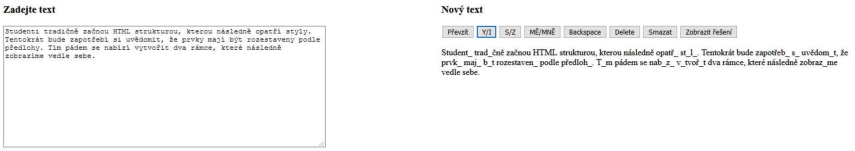
Studenti tradičně začnou HTML strukturou, kterou následně opatří styly. Tentokrát bude zapotřebí si uvědomit, že prvky mají být rozestaveny podle předlohy. Tím pádem se nabízí vytvořit dva rámce zobrazené vedle sebe.

Základní funkcí, kterou je potřeba vyřešit, je přenos textu z levé sekce do pravé, kde budou možné úpravy pomocí tlačítek. Studenti musí nejen dokázat přenést text, ale zároveň mají za úkol podle jeho délky vypsat, zda se jedná o krátký, dlouhý, nebo středně dlouhý diktát. Zde mají studenti možnost se kreativně vyjádřit v hláškách zobrazovaných v modálních oknech. Pokud by nebylo ve vstupním poli uživatelem nic zadáno a pokusil se stisknout tlačítko pro převzetí textu, musí vyskočit varovná hláška a zároveň se nesmí spustit ani procedury, které by přebíraly „nezadaný“ text.

Po vytvoření funkce pro převzetí musí studenti zpracovat odstranění požadovaných znaků či slabik. Zvolili jsme možnosti y/i, s/z, mě/mně, které bývají tradičními doplňovačkami na základních školách. Úkolem je najít požadovaný znak a nahradit jej v textu podtržítkem.

Na závěr by měli studenti vytvořit tlačítko zobrazující po stisknutí správné řešení. Tato funkcionalita je poměrně jednoduchá. Vezme se původní text a v něm se najdou znaky, které byly odstraněny. Ty se pak zobrazí červeně a velkým písmem.

Zbylá tlačítka z obr. 4, tedy `Backspace`, `Delete` a `Smazat`, jsou opakováním z předchozích úloh a jejich funkcionalita je stejná. Studenti ve většině případů nemají problém zopakovat již vytvořené řešení.



Obr. 4 Úloha Diktát

Kód úlohy je k dispozici na https://codepen.io/filip-frank/pen/qBNKjwm.

Pozorované chyby

Při tvorbě HTML struktury studenti často chybují v rozdělení rámců. Obvykle ztrácí přehled o svém kódu a nevidí, kde rámec začíná a končí. Výsledkem je, že mají následně poměrně závažné problémy s rozestavěním jednotlivých částí pomocí CSS. Jako dobré řešení se ukázalo apelovat na odsazování vnořených prvků, kdy jsou začátky a konce rámců dobře vidět.

Při převzetí textu se pak objevuje víc chyb, které ale nemají fatální dopad na celou funkčnost. Studenti správně zkontrolují, zda je něco v poli zadáno, pak ale kontrolují délku textu a zde dochází buď k překrývání intervalů, kdy diktát je jak středně dlouhý, tak krátký apod. Dochází také k vynechání jedné hodnoty, která je přesně na přelomu délky textu. Studenti například kontrolují, zda je text kratší než 150 znaků, pak je diktát krátký. Při kontrole, zda je text střední se pak ptají, zda je text delší než 150 znaků a zároveň kratší než 300. Tím pádem vynechávají délku 150. Situace se pak může opakovat, kdy místo použití ELSE pro dlouhý diktát specifikují delší než 300, čímž opět délka 300 vypadne. Studenti po demonstrování těchto chyb snadno doplňuje větší/menší nebo rovno, případně na poslední možnost využijí ELSE.

Dalším problematickým prvkem se stává samotné odstraňování požadovaných znaků. Studenti si správně vyhledají, že JavaScript má na vyhle-

dávání a nahrazení znaku funkci a bez problému ji využijí. Problém je, že funkce nalezne pouze první hledaný prvek, ale neprochází celý text. Tento problém studenti často sami ani nevyřeší, protože spoléhají na dodávané funkce. Řešení je projít si ucelené informace k funkci. Připravená funkce totiž obsahuje parametry, které umožňují procházet celý text. Druhé řešení je využití cyklu. V době řešení této úlohy však ještě cykly nebyly probrány, a tak většina studentů sáhne k řešení pomocí parametru připravené funkce.

Chybou, kterou dělá drtivá většina studentů, je také opomenutí faktu, že JavaScript je case sensitive, tedy rozlišuje velká a malá písmena. Častým problémem je, že studenti nahradí pouze malá písmena a velká zůstanou nenahrazena.

Posledním problémem je ve všech případech zobrazení řešení. Tuto problematiku vyřeší obvykle jednotky studentů v ročníku. Studenti se snaží získat původní text z upravovaného textu s podtržítky. Po několika marných pokusech se rozhodnou, že by bylo možné znovu převzít text z pole, do kterého nám zadává text uživatel. Toto řešení funguje do chvíle, než chtějí prezentovat své řešení vyučujícímu. Studenti totiž opomíjejí fakt, že s textem v poli může uživatel manipulovat. Ve chvíli, kdy vyučující text smaže a zkouší zobrazit řešení, se smaže i druhý text. Správným řešení zde je uložit text do duplicitní proměnné už při prvotním přebírání textu. V tom případě nám zůstane text původní, který můžeme zobrazit a pouze v něm zvětšit a začervenit požadované části.

Závěr

V článku jsou představeny tři úlohy z předmětu *Programování 1 pro vzdělávání*. Úlohy za sebou mají 4. rok nasazení ve výuce, a proto bylo možné představit opakované chyby, kterých se studenti při řešení dopouštějí. Úlohy reflektují velký rozptyl zkušeností studentů s programováním, přičemž při řešení úloh mohou studenti používat internet a různé zdroje, které by stejně v praxi používali. V rámci předmětu je vytvořeno celkem 8 úloh pro semináře a další 3 úlohy ke zkoušce. Další úlohy budou s ohledem na rozsah článku představeny v budoucnu.

Literatura

[1] Programování 1 pro vzdělávání. Západočeská univerzita, Plzeň.

[2] Rámcový vzdělávací program pro základní vzdělávání. Dostupné z: https://www.edu.cz/wp-content/uploads/2021/07/RVP-ZV-2021.pdf

[3] *Švaříček, Roman, Šeďová, Klára a kol.*: Kvalitativní výzkum v pedagogických vědách. Portál, Praha, 2014.