ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

Multimediální podpora pro předmět KVD/PGM1P

BAKALÁŘSKÁ PRÁCE

Jiří Míka
Informatika se zaměřením na vzdělávání

Vedoucí práce: Mgr. Filip Frank, Ph. D.

Plzeň, 2024

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni, 23. dubna 2024

.....

vlastnoruční podpis

Poděkování

Na tomto místě bych rád poděkoval Mgr. Filipu Frankovi, Ph. D. za cenné rady, věcné připomínky a podporu při vypracovávání této práce. Také bych rád poděkoval doc. PhDr. Lucii Rohlíkové, Ph.D. za věcné připomínky při tvorbě kurzu v rámci předmětu KVD/ELVV.

Mé poděkování také patří rodičům za jejich podporu a možnosti, které vytváří pro mé studium.

ZDE SE NACHÁZÍ ORIGINÁL ZADÁNÍ KVALIFIKAČNÍ PRÁCE.

OBSAH

SEZNAM ZKRATEK

- CSS: Cascading Style Sheets
- HTML: Hypertext Markup Language
- JS: JavaScript
- LMS: Learning Management System
- VSC: Visual Studio Code

ÚVOD

Bakalářská práce se věnuje vytvoření multimediální podpory v LMS Moodle pro předmět KVD/PGM1P. Celá práce je rozdělena do čtyř kapitol, jež jsou dále členěny do podkapitol.

První kapitola se věnuje popisu a analýze předmětu KVD/PGM1P. Jsou zde popsány požadavky na studenta a vzdělávací obsah předmětu. Kapitola dále obsahuje zařazení programovacího jazyka používaného v rámci předmětu z hlediska vyššího programovacího jazyku. Dále z hlediska kompilovaných, interpretovaných a Just-In-Time kompilovaných jazyků. V závěru kapitoly se nachází popis vývojových prostředí, v nichž studenti plní seminární úlohy.

Druhá kapitola se soustředí na analýzu a popis právě těchto úloh, s nimiž se studenti setkávají v rámci seminářů. Analýza každé úlohy je rozdělena na čtyři části. V první části je vždy popsána výsledná funkcionalita úlohy a nachází se zde také obecný popis celé úlohy. Druhá část se poté zaměřuje na postup, nímž je možné seminární úlohu realizovat. Třetí část se poté vždy věnuje analýze častých chyb, které se vyskytují při realizaci úlohy. V poslední části jsou vždy stanoveny poznatky, které je nutné zmínit v příslušném multimediálním prvku, jenž bude popisovat realizaci této seminární úlohy.

Třetí kapitola se zabývá zásadami a principy pro návrh a vytváření vzdělávacích e-kurzů podle Maněny. Popsán je zde také samotný návrh vzdělávacího e-kurzu pro předmět KVD/PGM1P. Při návrhu našeho vzdělávacího e-kurzu jsme vycházeli právě z těchto zásad a principů. V samotném úvodu kapitoly se vykytuje popis LMS Moodle. Dále zde popisujeme, jakým způsobem byl navržen obsah a jakým způsobem vznikla struktura jednotlivých kapitol. V samotném závěru kapitoly je poté představen konečný seznam kapitol použitých v e-kurzu.

Poslední kapitola se poté věnuje samotné realizaci vzdělávacího e-kurzu pro předmět KVD/PMG1P v LMS Moodle. Také se zde vyskytuje popis tvorby jednotlivých multimediálních a interaktivních prvků.

1 PŘEDMĚT KVD/PGM1P

Předmět PGM1P, celý názvem Programování 1 pro vzdělávání, je vyučovaný na katedře výpočetní a didaktické techniky Pedagogické fakulty ZČU. Výuka probíhá v zimním semestru. Kreditové ohodnocení předmětu je stanoveno na čtyři kredity. Tyto údaje jsou platné v akademickému roku 2023/2024. [1]

1.1 POŽADAVKY NA STUDENTA

„Student úspěšně absolvuje předmět po získání zápočtu a následném splnění kombinované zkoušky. Znalosti studenta jsou ověřovány v průběhu semestru za pomoci seminárních úloh, průběžných zápočtových testů a dlouhodobé semestrální práce [2].“

1.1.1 PŘEDPOKLADY PRO ÚSPĚŠNÉ ABSOLVOVÁNÍ PŘEDMĚTU

Výuka předmětu KVD/PGM1P je řádně realizována ve druhém ročníku bakalářského studijního programu.

„Odborné znalosti a dovednosti, jež jsou předpokladem pro úspěšné zvládnutí PGM1P:

- *identifikace problémů souvisejících s algoritmickou složitostí,*
- *využití základních algoritmických konstrukcí,*
- *aplikace různých způsobů algoritmizace,*
- *analýza problémů a návrh vhodných řešení,*
- *řešení příkladů rozvíjející algoritmické myšlení ve zvoleném prostředí* [1].“

Tyto předpoklady vycházejí z jiného předmětu, a to KVD/ALGV. Ačkoliv předmět PGM1P navazuje na tento předmět, tak se nejedná o jeho prerekvizitu. [3]

Algoritmizace pro vzdělávání je v akademické roce 2023/2024 řádně vyučována v prvním ročníku bakalářského studijního programu a jejím hlavním cílem je rozvinout algoritmické myšlení studentů pomocí různorodých metod. [1]

Vedle odborných znalostí a dovedností by měl student disponovat i odpovídající obecnou způsobilostí, tj. schopnost samostatně plánovat své učení a schopnost posuzovat rizika jednotlivých variant řešení, včetně zvážení jejich kladů a záporů. [1]

1.1.2 PODMÍNKY PRO ZÍSKÁNÍ ZÁPOČTU

Zisk zápočtu je podmíněn úspěšným splněním dvou praktických zápočtových testů a uznáním semestrální práce. Docházka na seminář není podmínkou pro zisk zápočtu.

Testy jsou realizovány v rámci seminářů a slouží k ověření získaných znalostí studentů. Rozhodující pro uznání či neuznání testu je výsledná funkcionalita studentova řešení. V rámci hodnocení není posuzován zdrojový kód ani způsob, jímž student zadání řešil. Každý test je možné maximálně jednou opakovat. V takovém případě je studentovo povinností, zapsat se na příslušný opravný termín. Pokud tak neučiní, ztrácí právo na opravu. [2]

Po celou dobu řešení testového zadaní mají studenti k dispozici internetové připojení. V rámci internetu je studentům umožněno využívat jakékoliv dostupné zdroje. Tato možnost vychází z praxe, kde by studenti tyto zdroje také využili. Zakázaná je naopak komunikace mezi studenty a využívání příkladů ze seminářů v jakékoliv formě. Testové zadání typicky vychází z příkladů řešených na semináři. [3]

Semestrální práci představuje komplexní příklad, jejž je nutné vyřešit, do řádného termínu odevzdat a úspěšně obhájit. Práce nemá stanovený maximální počet odevzdání.

Obhajobou práce se prokazuje následující [2] [3]:

- porozumění řešeného příkladu,
- znalost zdrojového kódu,
- schopnost reagovat na dotazy vyučujícího.

V akademickém roce 2023/2024 došlo k doplnění podmínek obhajoby. Student má nyní k dispozici pouze jeden pokus. Lze předpokládat, že pokud student práci samostatně vypracoval, bude jejímu obsahu rozumět. [2]

Po úspěšném splnění všech těchto částí je studentovi udělen zápočet v rámci předmětu KVD/PGM1P a tím i možnost splnit zkoušku.

1.1.3 PODMÍNKY PRO SPLNĚNÍ ZKOUŠKY

Předmět je zakončen kombinovanou zkoušku, tzn. že se skládá ze dvou částí. První část je teoretická a ověřuje teoretické znalosti studenta. Druhá část poté zkoumá praktické dovednosti studenta. [2]

Teoretická část se skládá ze tří otázek, úspěšnost jejich zodpovězení poté ovlivňuje část praktickou. [2]

Praktická část je tvořena komplexním příkladem, jenž je rozdělen na tři varianty. Ty slouží jako úrovně se zvyšující se obtížností.

Vliv zodpovězení otázek na praktickou část je následující [2]:

- V případě zodpovězení všech tří otázek správně, stačí naprogramovat pouze základní variantu praktického příkladu. Tím student obdrží hodnocení dobře. Naprogramováním zbylých dvou variant může dosáhnout na hodnocení výborně.
- V případě zodpovězení dvou otázek správně, je potřeba naprogramovat dvě varianty praktického příkladu. Tím student obdrží hodnocení dobře. Naprogramováním zbylé varianty může dosáhnout na hodnocení velmi dobře.
- V případě zodpovězení pouze jedné otázky správně, je potřeba naprogramovat všechny varianty praktického příkladu. Tím student obdrží hodnocení dobře. V tomto případě nemůže student dosáhnout na lepší hodnocení.
- V případě, že student není schopen zodpovědět ani jednu otázku, není připuštěn k praktické části a obdrží hodnocení nevyhověl.

1.2 VZDĚLÁVACÍ OBSAH PŘEDMĚTU

Předmět si za hlavní cíl klade uvést studenty do problematiky programování v procedurálním jazyku a seznámit je se teoretickými a praktickými základy se zvláštním zřetelem pro vzdělávání. [1]

V rámci výuky jsou studenti seznámeni s následujícími tématy [1]:

- možnosti a techniky programování,
- typy programovacích jazyků,
- syntax a sémantika,
- datové typy,
- načítání dat ze vstupu a jejich zobrazení na výstupu,
- textové řetězce a matematické funkce,
- řídící struktury,
- práce s polem,
- funkce a parametry,
- práce se souborem,
- ladění programu,
- didaktické zásady výuky vybraných tematických celků.

Na základě úspěšného absolvování předmětu, a tudíž i porozumění výše zmíněných témat prokazuje student odborné dovednosti [1]:

- správný zápis syntaxe daného programovacího jazyku,
- znalost sémantiky daného programovacího jazyku,

- rozdělení programovacích jazyků,
- řešení úloh se vstupními a výstupními operacemi,
- odladění programu,
- využití vhodných funkcí a parametrů.

1.3 PROGRAMOVACÍ JAZYK POUŽÍVANÝ V RÁMCI PŘEDMĚTU

Všechny témata a veškerý vzdělávací obsah, jenž byl zmíněn v předešlé kapitole je studentům představen v rámci programovacího jazyku JavaScript (dále jen „JS“). [3]

Jelikož JS slouží pro vytváření scriptu webových stránek, jsou studenti okrajově seznámeni i s ostatními webovými technologiemi jako je HTML a CSS.

1.3.1 PROGRAMOVACÍ JAZYK JS

Marjin Haverbeke v knize „Eloquent JavaScript“ představuje JS následovně. *„JavaScript was introduced in 1995 as a way to add programs to web pages in the Netscape Navigator browser. The language has since been adopted by all other major graphical web browsers. It has made modern web applications possible— applications with which you can interact directly without doing a page reload for every action. JavaScript is also used in more traditional websites to provide various forms of interactivity and cleverness*. [4]“

David Flanagan v knize „JavaScript: The Definitive Guide“ definuje jazyk JS následovně. *„JavaScript is a high-level, dynamic, interpreted programming language that is well-suited to object-oriented and functional programming styles* [5].“ To tedy znamená, že se jedná o vysokoúrovňový, dynamický a interpretovaný programovací jazyk, který lze využít pro objektově orientované a funkcionální programování.

Haverbeke poukazuje také na skutečnost, že je JS velmi liberální programovací jazyk. To ovšem sebou nese nevýhody pro začínající programátory. *„JavaScript is ridiculously liberal in what it allows. The idea behind this design was that it would make programming in JavaScript easier for beginners. In actuality, it mostly makes finding problems in your programs harder because the system will not point them out to you*. [4]“

Ačkoliv předmětem této práce není charakteristika programovacího jazyku JS, dovolili bychom si vlastnosti zmíněné ve Flanganově definici krátce popsat a upřesnit.

1.3.2 VYSOKOÚROVŇOVÝ PROGRAMOVACÍ JAZYK

Vysokoúrovňový programovací jazyk je možné definovat následovně. „*A high-level programming language has a significant abstraction from the details of computer operation. It is designed to be easily understood by humans and for this reason they must be translated by another software* [6].“ Volně přeloženo to tedy znamená, že se jedná o programovací jazyk s významnou mírou abstrakce, jenž je navržen pro snadné porozumění lidmi. Z toho důvodu je nutné takový jazyk překládat jiným programem.

Tuto vlastnost také popisují Gabbrielli a Martini ve své knize „Programming Languages: Principles and Paradigms“ následovně. „*High-level languages are therefore suited to expressing algorithms in ways that are relatively easy for the human user to understand* [7].“ Vysokoúrovňové jazyky jsou navrženy tak, aby vyjadřovali algoritmy v podobě, která bude pro uživatele snadno pochopitelná.

Abstrakcí tedy rozumíme, co největší přiblížení srozumitelnosti programovacího jazyku lidem. Podle míry abstrakce je možné určovat o jak vysokoúrovňový programovací jazyk se jedná. [6]

1.3.3 DYNAMICKÝ PROGRAMOVACÍ JAZYK

Programovací jazyk můžeme považovat za dynamický v případě, pokud umožňuje modifikovat program při jeho běhu. Z anglického jazyka „Run-time“ modifikace. Takové jazyky obvykle také disponují vlastností dynamických datových typů. [8]

U jazyků s dynamickými datovými typy není nutné přiřazovat datový typ proměnné při její deklaraci. Ten je přiřazen automaticky za běhu programu, podle hodnoty přiřazené do proměnné. Takovou vlastností také disponuje jazyk JS. [9]

1.3.4 JUST-IN-TIME KOMPILOVANÝ JAZYK

Flanagan uvádí, že se jedná o interpretovaný jazyk, což může být ovšem zavádějící. JS je ve skutečnosti Just-in-time (dále jen „JIT“) kompilovaný programovací jazyk. [10]

U interpretovaného jazyku je zdrojový kód interpretován pomocí interpretu. Interpret je speciální program, jenž zdrojový kód čte a vykonává řádek po řádku. Syntaktická správnost zápisu se provádí až za běhu programu samotného. Interpretované jazyky jsou pomalejší než jazyky kompilované. V případě změny zdrojového kódu není nutné provádět jeho překlad. [11] [12]

U kompilovaného jazyku dochází ke kompilaci zdrojového kódu pomocí kompilátoru na strojový kód. Neboli na instrukce, které jsou srozumitelné pro procesor, jenž bude program následně vykonávat. Tím vzniká samostatně spustitelný program. Syntaktická správnost je kontrolována při překladu zdrojového kódu. V případě jeho změny je nutné celý proces překladu opakovat. [11] [12]

JIT kompilovaný neboli dynamicky kompilovaný jazyk, vhodně spojuje vlastnosti interpretovaných a kompilovaných jazyků. Během interpretace zdrojového kódu dochází k jeho monitorování. Na základě toho jsou určeny nejpoužívanější části, jež kompilátor kompiluje na strojový kód. [13]

1.4 VÝVOJOVÉ PROSTŘEDÍ POUŽÍVANÉ V RÁMCI PŘEDMĚTU

Do začátku akademického roku 2023/2024 byla výuka realizována v prohlížečovém vývojovém prostředí Codepen.io. Od akademického roku 2023/2024 je pro účely výuky využíváno vývojové prostředí Visual Studio Code.

1.4.1 VÝVOJOVÉ PROSTŘEDÍ CODEPEN.IO

Jedná se o vývojové prostředí, jež je realizováno uvnitř webového prohlížeče. Podporuje jazyky HTML, CSS a JS. Tím umožňuje tvorbu webových stránek.[14]

Výhodou tohoto vývojového prostředí je automatické zobrazování provedených změn ve zdrojovém kódu přímo na webové stránce bez nutnosti jejího obnovení. Další výhodou je separace HTML, CSS a JS. Ty zde fungují jako tři samostatné části, které jsou mezi sebou automaticky provázány, tudíž zde studenti nemusejí řešit problematiku propojení externích souborů CSS a JS s dokumentem HTML. [3]

Vytvořený program, uvnitř tohoto prostředí tzv. pen, je uložen na uživatelském účtu studenta, což umožňuje studentovi přístup ke svému projektu odkudkoli. [3]

Jako největší nevýhodou vývojového prostředí Codepen.io se jeví zacyklení programu, tento problém popisují Frank a Zíka následovně. „*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* [3].“

1.4.2 VÝVOJOVÉ PROSTŘEDÍ VISUAL STUDIO CODE

Visual Studio Code (dále jen „VSC“), vývojové prostředí vyvíjené společností Microsoft. VSC poskytuje nástroje pro tvorbu kódu a jeho následné odladění. [15]

VSC umožňuje práci s většinou programovacích jazyků pomocí stažitelných rozšíření. Jazyky potřebné pro plnění obsahu předmětu PGM1P jsou součástí VSC již ve výchozím nastavení. [16]

Práce ve VSC oproti Codepen.io staví již studenty před výzvu správy pracovního adresáře a nutnost propojování externích souborů se scriptem a styly do dokumentu HTML. Mezi hlavní benefity však patří možnost krokování, sledování proměnných kukátkem a použití komplexnější verze našeptávače. To vše přispívá k snazšímu odladění programu. Studenti se zároveň seznámí s reálným vývojovým prostředím, a ne pouze s výukovým nástrojem.

1.5 ZÁVĚR PŘEDMĚTU KVD/PGM1P

Úvodní kapitola s názvem „Předmět KVD/PGM1P“ si jako hlavní cíl kladla analyzovat a popsat předmět KVD/PGM1P. V první části kapitoly jsme se věnovali popisu a analýze podmínek, jež musí studenti splnit pro úspěšné absolvování předmětu. Zjistili jsme, že každý student by měl před zahájením studia předmětu KVD/PGM1P disponovat odbornými znalostmi a dovednostmi, ty slouží jako předpoklad pro úspěšné absolvování předmětu. Tyto předpoklady by studenti měli získat v předmětu KVD/ALGV. Pro zisk zápočtu před zkouškou musí studenti úspěšně splnit dva praktické zápočtové testy a úspěšně obhájit semestrální práci. V akademické roce 2023/2024 došlo k zásadní změně, kdy na obhajobu semestrální práce mají studenti pouze jeden pokus. Zkouška předmětu KVD/PGM1P je kombinovaná, kdy studenti začínají teoretickou části a v případě splnění minimálních podmínek jsou následovně připuštěni k části praktické. V závěru první části kapitoly jsme se věnovali popisu vzdělávacího obsahu.

Druhá část úvodní kapitoly byla věnována programovacímu jazyku JS a vývojovým prostředím Codepen.io a VSC, používaných v rámci předmětu KVD/PGM1P. Pro stanovení vlastností jazyku JS jsme využili definici D. Flanagana. Následně jsme tyto jednotlivé vlastnosti popsali a programovací jazyk zařadili. Do akademického roku 2022/2023 byla výuka předmětu realizována ve vývojovém prostředí Codepen.io. Od akademického roku 2023/2024 je studentům představována práce v plnohodnotném vývojovém prostředí VSC.

Mimo popis a charakteristiky jsou tyto dvě vývojové prostředí porovnána a jsou představeny možné klady a zápory každého z nich.

2 ÚLOHY POUŽÍVANÉ V PŘEDMĚTU KVD/PGM1P

Námi zkoumaná sada se skládá ze všech úloh, které byly použity ve výuce, v akademickém roce 2023/2024. Všechny tyto úlohy jsou koncipovány podle vzdělávacího cíle předmětu. Vypracování seminárních úloh není kontrolováno vyučujícím a studenti nemají žádnou povinnost tyto úkoly odevzdávat. Ovšem, zápočtové testy jsou založeny na velmi podobné nebo stejné logice jako úlohy ze seminářů. [2] [3]

Frank a Zíka poukazují na problém, který může negativně ovlivnit průběh výuky. „*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.)* [3].“

Pokud víme o velké rozdílnosti úrovní studentů, můžeme předpokládat, že se při řešení seminárních úloh bude vyskytovat množství různých chyb. Z toho důvodu je pro nás důležité jednotlivé úlohy důkladně popsat, analyzovat a nejčastěji vyskytující chyby objevit. Pomocí tohoto zjištění můžeme následně vhodně připravit multimediální prvky pro e-kurz.

2.1 PRVNÍ SEMINÁRNÍ ÚLOHA

Úvodní úloha představuje studentům základy HTML, CSS a programovacího jazyku JS. Studenti se seznamují se syntaxí jednotlivých částí a získávají představu, jakou funkci na webové stránce, každá tato část plní. [3]

2.1.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, jež bude umožňovat přepínat mezi dvěma obrázky. Přepínání studenti realizují pomocí tří tlačítek, kde dvě tlačítka slouží vždy pro zobrazení příslušného obrázku. Tlačítko třetí provede cyklicky změnu zobrazeného obrázku. Na příslušných místech musí být také zobrazeno jméno studenta a název předmětu. [3]

Všechny prvky budou sdruženy do rodičovského elementu a umístěny na jeho střed. Rodičovský element musí mít nastavenou barvu pozadí. Veškerý text na webu bude zobrazen ve fontu Arial. Texty s názvem předmětu a jménem studenta budou mít

nastavený styl podle zadání. Tlačítka, pomocí kterých je realizováno přepínaní obrázků, budou ohraničena rámečkem. [3]

Obrázek 1: První seminární úloha (Zdroj: vlastní)

2.1.2 REALIZACE ÚLOHY

Studenti v prvním kroku vytvoří HTML strukturu. K tomu využijí následující HTML značky (ty budou v zbytku práce zapsány pouze slovně, nikoliv ve správném syntaktickém zápisu HTML):

- **Značka div**: Rodičovský element, jenž studentům umožní sdružit všechny ostatní elementy a následně je vystředit. Studenti využijí tohoto principu i pro sdružení tlačítek, aby mohli následně aplikovat ohraničení.
- **Značka p**: Textový odstavec, studenti pomocí této značky zobrazí název předmětu.
- **Značka h1** – **h6**: Nadpis první až šesté úrovně. Studenti pomocí této značky zobrazí své jméno.
- **Značka button**: Tlačítka, pomocí nichž studenti realizují přepínaní obrázků.
- **Značka img**: Značka umožňující zobrazení obrázku na stránce.

Při tvorbě struktury HTML se studenti seznámí i se základními atributy:

- **Atribut id**: Jednoznačný identifikátor, uvnitř CSS selektor, pomocí kterého studenti přistupují k elementu a mohou pro něj vytvářet pravidla.
- **Atribut class**: Třída, studenti mohou aplikovat na více elementů a mohou vytvářet pravidla pro všechny elementy s příslušnou třídou najednou.
- **Atribut alt**: Alternativní popisek, jenž se zobrazí, pokud nedojde k načtení obrázku. Studenti tento atribut využijí pro porovnávání zobrazených obrázků uvnitř JS.

- **Atribut src**: Zdrojová cesta k obrázku. Studenti musí zadat správnou cestu podle adresářové struktury svého projektu.
- **Atribut onclick**: Typ události, studenti pomocí tohoto atributu provážou příslušnou funkci s příslušným tlačítkem.

Po vytvoření struktury HTML je možné aplikovat styly CSS. U vytváření pravidel v tomto případě není nutné trvat na konkrétním pořadí. Kroky jsou v seznamu uvedeny v pořadí, v jakém se jeví logické:

- Nastavení fontu Arial pro všechen text.
- Zarovnání rodičovského elementu na střed stránky a vystředění všech prvků vněm.
- Nastavení barevného pozadí rodičovského elementu a nastavení jeho vhodné šířky. Při nastavení barevného pozadí, studenti následovně snadněji pozorují změny na rozměrech rodičovského elementu.
- Ohraničení tlačítek, to se aplikuje ovšem na celou šířku rodičovského elementu. Studenti musejí následně upravit šířku kontejneru v němž jsou tlačítka uložena a zarovnat jej opět na střed.
- Nastavení barvy zobrazovaných textů a doplnění efektu podtržení u textu s názvem předmětu.

Studenti se také seznámí se základním principem kaskád uvnitř CSS, kdy zjistí, že mohou upravovat text, pomocí selektoru ID, jenž se nachází uvnitř třídy. [3]

V posledním kroku studenti vytvoří funkce uvnitř JS pro zobrazování příslušných obrázků. Studenti se při tvorbě těchto funkcí seznámí s možnostmi, jak přistupovat k HTML elementům uvnitř JS. [3]

Při vytváření funkce pro zobrazení příslušného obrázku by studenti měli postupovat následovně [3]:

- Vytvoření funkce pomocí klíčového slova *function*.
- Přistoupení k HTML elementu pomocí tečkové notace uvnitř JS.
- Nastavení atributů alt a src pomocí tečkové notace na příslušné hodnoty.
- Provázání funkce s tlačítkem uvnitř HTML.

Takto vytvořenou funkci pro zobrazení příslušného obrázku mohou studenti zkopírovat a vhodně upravit. Následně ji mohou využívat pro zobrazení obrázku druhého.

Poslední funkci a způsob její realizace při výuce popisují Frank a Zíka následovně: „*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 [3].“

Od akademického roku 2023/2024, kdy je výuka realizována uvnitř vývojového prostředí VSC, si studenti ukládají obrázky do svých pracovnách adresářů. Z toho důvodu není možné zobrazené obrázky porovnávat pomocí atributu src, jako tomu bylo při realizaci úlohy uvnitř vývojového prostředí Codepen.io. Důvodem je lišící se hodnota atributu src, kterou získáváme uvnitř HTML a hodnota uvnitř JS. Proto je nutné využít pro porovnávání zobrazeného obrázku atribut alt.

2.1.3 CHYBY PŘI REALIZACI ÚLOHY

Vyskytující chyby při realizaci první úlohy v rámci semináře popisují Frank a Zíka: „*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* [3].“

Od akademického roku 2023/2024 můžeme pozorovat chybu, již se v minulých letech při realizaci výuky nevyskytovala. Ve vývojovém prostředí Codepen.io byly části CSS a JS automaticky provázané s částí HTML. Uvnitř vývojového prostředí VSC ovšem žádný takový vztah neplatí a části s CSS a JS je nutné k dokumentu HTML připojit. Pokud tak studenti neučiní, není následně možné zjišťovat funkcionalitu těchto částí na stránce.

2.1.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace první seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

- V rámci CSS vysvětlit rozdíl mezi selektory id a class.
- Při porovnávání zobrazených obrázků zdůraznit a vysvětlit logiku, kterou se řídíme a na základě níž budou podmínky vyhodnoceny.
- Propojení externích částí CSS a JS s částí HTML.

2.2 DRUHÁ SEMINÁRNÍ ÚLOHA

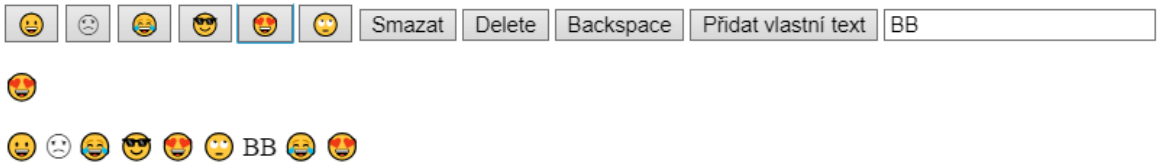
Druhá seminární úloha představuje studentům práci se vstupními hodnotami, proměnnými, funkcemi, funkcemi s parametrem a podmínkami. [3]

2.2.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, již bude umožňovat po stisku tlačítka zobrazit příslušný znak. Dále bude možné zadávat a následně zobrazovat text. Zobrazení bude realizováno ve dvou textových odstavcích, kde jeden bude sloužit pro zobrazení posledního stisknutého znaku a druhý jako historie všech již zobrazených znaků a textů. [3]

Aplikace dále bude umožňovat smazat oba dva textové odstavce, provádět mazání historie odpředu a odzadu. Pro mazání se stanoví podmínky, jež budou kontrolovat, zdali je možné nějaké hodnoty mazat. [3]

Tato úloha neobsahuje žádnou práci s CSS, studenti pouze vytvoří základní strukturu HTML a provedou její oživení pomocí kódu JS. [3]



Obrázek 2: Druhá seminární úloha (Zdroj: vlastní)

2.2.2 REALIZACE ÚLOHY

Studenti v prvním kroku opět vytvoří základní strukturu HTML. K tomu využijí následující HTML značky:

- **Značka button:** Tlačítka, studenti pomocí nich realizují výpis jednotlivých znaků, přidání vlastního textu a veškeré mazání.
- **Značka input:** Vstup, studenti pomocí této značky zajistí možnost zadávat vstupní data.
- **Značka p:** Textové odstavce, budou umožňovat zobrazení znaků a textů v požadovaném tvaru.

Oproti předešlé seminární úloze se studenti neseznámí s žádnými novými atributy HTML.

Po studentech v této úloze není vyžadována jakákoliv práce s kaskádovými styly. Tudíž se po vytvoření struktury HTML začínají studenti zabývat tvorbou kódu v JS. Prvním úkolem v této části je zajistit zobrazení znaků z příslušných tlačítek. To je na seminářích realizováno

formou vzájemné spolupráce mezi studenty a vyučujícím, jenž postupně seznámí studenty s následujícími čtyřmi možnostmi řešení [3]:

- Deklarace proměnné a inicializace požadovaného znaku uvnitř funkce v JS. Následné vypsání této proměnné do příslušných textových odstavců.
- Získání a uložení požadovaného znaku z příslušného tlačítka v části HTML do proměnné a následné vypsání této proměnné.
- Vypsání požadovaného znaku jako textového řetězce do příslušných textových odstavců.
- Využití funkce s parametrem, kde parametr bude představovat požadovaný znak. Při zavolání této funkce v části HTML se parametr požadovaným znakem nahradí.

Studentům je zároveň představeno několik různých způsobů, kterými je možné realizovat výpis do textových odstavců. Důležité je po vytvoření každé funkce její provázání s příslušným tlačítkem v části HTML [3].

Dalším úkolem je načtení vlastního textu a jeho následné zobrazení. Text bude zobrazován pouze v textovém odstavci představující historii zobrazených znaků. Studenti realizují tento dílčí úkol následovně [3]:

- Vytvoření proměnné a uložení hodnoty HTML prvku input.
- Zobrazení této proměnné do textového odstavce v HTML části.

Studenti musí také zajistit kontrolu zadaného textu. V případě, že dojde k zadání prázdného řetězce, budeme na tuto skutečnost upozorněni a nedojde k vložení do historie znaků.

V posledním dílčím úkolu studenti realizují smazání zobrazených textů, to bude prováděno následujícími třemi způsoby [3]:

- Smazání všeho textu zobrazeného v textových odstavcích.
- Smazání vždy prvního znaku v historii zobrazených znaků.
- Smazání vždy posledního znaku v historii zobrazených znaků.

Vyučující seznámí studenty s metodami textových řetězců, jenž studenti následně vhodně využijí při vytváření příslušných funkcí. Ty musí studenti také vhodně ošetřit. V případě, že se v textových odstavcích nevyskytuje žádný znak, a tudíž není co mazat, budeme na tuto skutečnost upozorněni. [3]

2.2.3 CHYBY PŘI REALIZACI ÚLOHY

Častou syntaktickou chybou je špatná deklarace proměnné. Studenti zaměňují strany hodnoty a proměnné u přiřazovacího operátoru. Místo přiřazení hodnoty do proměnné provádí přiřazení proměnné do hodnoty. [3]

Další častou syntaktickou chybou je neprovázání funkce z části JS s příslušným tlačítkem v části HTML. [3]

Chybou, která již není syntaktická, je poté špatná realizace výpisu zobrazovaných znaků do historie. Tuto chybu popisuje Frank a Zíka následovně: „*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* [3].“

Poslední chybou je špatné stanovení podmínky pro kontrolu mazaných odstavců, kdy studenti neví, vůči čemu mají proměnnou porovnávat. [3]

2.2.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace druhé seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

- Zdůraznit pravidla, která platí při deklaraci proměnných.
- Nutnost provázat funkci z JS s příslušným elementem v části HTML.
- Vysvětlení, jak je možné řetězit textové řetězce. Představení a popsání jednotlivých variant.
- Porovnávání obsahu textových řetězců, jak je možné zjistit, jestli textový řetězec neobsahuje žádnou hodnotu.

2.3 TŘETÍ SEMINÁRNÍ ÚLOHA

Třetí seminární úloha představuje studentům podrobněji práci s textovými řetězci a vhodné využití metod textových řetězců, které JS obsahuje. [3]

2.3.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, která bude umožňovat načítat vlastní text, ten bude zobrazen v příslušném textovém odstavci. Následně bude možné v tomto textu skrývat určité znaky. Ty budou nahrazeny jiným určitým znakem. U takto upraveného textu bude možné provést kontrolu, kdy se všechny skryté znaky objeví zpět na svém původním místě.

Ovšem nyní budou zobrazeny červeně a velkým písmem. Poslední funkcí aplikace bude možnost zobrazený text smazat. [3]

Jako u předešlých úloh, je po studentech vyžadováno, aby pomocí podmínek ošetřili konkrétní případy, jež by mohly nastat.

V této úloze se objevuje i malá část práce s CSS, kdy je požadováno, aby se část s načtením textu a část zobrazující tento text a umožňující úpravu nacházeli vedle sebe. [3]

Obrázek 3: Třetí seminární úloha (Zdroj: vlastní)

2.3.2 REALIZACE ÚLOHY

Studenti musí opět v prvním kroku vytvořit základní strukturu HTML k tomu využijí následující HTML značky:

- **Značka div**: Rodičovský element, jenž studentům umožní rozdělit HTML prvky na dvě příslušné části. Zároveň umožní následné zobrazení těchto částí vedle sebe.
- **Značka p**: Textový odstavec, bude sloužit pro zobrazení načteného texu a následně textu upraveného.
- **Značka h1** – **h6**: Nadpis první až šesté úrovně, studenti pomocí této značky zobrazí popis části pro zadávání textu a části pro jeho úpravu.
- **Značka button**: Tlačítka, pomocí niž studenti realizují funkce načtení zadaného textu, skrytí jednotlivých znaků, provedení kontroly zobrazeného textu a také jeho smazání.
- **Značka textarea**: Zadávací oblast, do níž bude zadáván text, který budeme chtít následně zobrazit.

Po vytvoření HTML struktury, je nutné zajistit zobrazení příslušných částí vedle sebe. Aby bylo možné toto provést, musí mít příslušné prvky div nastavenou stejnou hodnotu u atributu class. Tím dojde k zařazení obou části do stejné třídy a je možné nastavit příslušné pravidlo uvnitř CSS [3].

Prvním dílčím úkolem v části JS je pro studenty zajistit načtení a zobrazení zadávaného textu. Načtení textu musí být ošetřeno následujícími podmínkami [3]:

- Pokud není zadán žádný text, a i přes to se uživatel rozhodne provést načtení, bude o tomto faktu upozorněn. Zároveň je nutné zajistit, aby v takovém případě nedošlo k načtení prázdného textu a spuštění procedur umožňující jeho úpravu.
- Pokud je délka zadaného textu kratší než sto znaků, je uživatel upozorněn, že se jedná o krátký text a je uveden přesný počet znaků v zadaném textu.
- Pokud se délka zadaného textu nachází v rozsahu sto až dvě stě znaků, je uživatel upozorněn, že se jedná o středně dlouhý text a je uveden přesný počet znaků v zadaném textu.
- Pokud je délka zadaného textu větší než dvě stě znaků, je uživatel upozorněn, že se jedná o dlouhý text a je uveden přesný počet znaků v zadaném textu.

Dále musí studenti vytvořit funkci, již bude zajišťovat nalezení a nahrazení požadovaného znaku jiným znakem. Tato funkce je realizována následujícím způsobem [3]:

- Vytvoření funkce s parametry, kde parametry budou reprezentovat typ hledaných znaků, náhradní znak a jednotlivé znaky, které chceme nahradit.
- Uložení zobrazovaného textu v HTML části do proměnné uvnitř funkce, to umožňuje následně snazší práci při procházení textu.
- Hledání požadovaného znaku probíhá cyklicky, do té doby, dokud se požadovaný znak vyskytuje ve vytvořené proměnné.
- Při nalezení hledaného znaku dochází vždy k jeho nahrazení znakem určeným pro náhradu.
- Takto upravený text uvnitř proměnné je následovně zpětně zobrazen v části HTML.

Také zde studenti musí kód ošetřit pomocí podmínek. V případě, že se požadovaný znak v textu nenachází budeme na tuto skutečnost upozorněni a nedojde ke spuštění výše popsané funkce.

Funkci zajištující kontrolu zobrazeného textu, zároveň zobrazení textu původně zadaného s červeně zobrazenými znaky, které byly nahrazeny, studenti realizují následovně [3]:

- Porovnání původně zadaného textu s textem upraveným. V případě shody těchto textů není nutné zobrazený text měnit, nedošlo totiž zatím k žádné jeho úpravě.
- V případě, že se texty neshodují je nutné pracovat s textem původním. V něm se požadované znaky zamění za znaky stejné ovšem červenou barvou zvýrazněné.
- Takto upravený původní text je následně zobrazen v příslušném textovém odstavci.

Poslední funkce, pomocí které bude realizováno mazání je založena na stejné funkcionalitě a logice jako stejná funkce v předešlé seminární úloze. [3]

2.3.3 CHYBY PŘI REALIZACI ÚLOHY

V této úloze se často vyskytuje chyba v části HTML, kde studentům činí problémy udržet přehlednost kódu, při sdružování jednotlivých prvků do prvku div. Taková chyba poté znemožňuje správné využití CSS a zobrazení části v požadovaném rozložení. [3]

Další chybou je také opomenutí skutečnosti, že programovací jazyk JS je case-sensitive. Rozlišuje tedy velká a malá písmena. To poté studentům znemožňuje provést nahrazení všech požadovaných znaků. [3]

Jako u předešlých seminárních úloh se i zde často vyskytuje chyba spojená se správným stanovením podmínek a jejich následovným vyhodnocením. Nejčastěji se to projevuje u podmínek zjišťující délku zadávaného textu, kde studenti nesprávně použijí příslušné operátory a vytvoří tak chybné intervaly. [3]

Nejčastěji vyskytující chyba je spojena s funkcí zajišťující kontrolu zobrazeného textu. Tuto chybu popisuje Frank a Zíka následovně: „*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.* [3].“

2.3.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace druhé seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

- Vhodný zápis a vnořování prvků uvnitř HTML. Možnost odsazení vnořených prvků pro lepší přehlednost HTML.
- Upozornit na základní vlastnosti JS, nutnost při hledání znaků vyhledávat malá i velká písmena.
- Stanovení a vysvětlení logiky uvnitř podmínek. Správné využívání logických operátorů a správné vyhodnocení celé podmínky.
- Vysvětlení funckionality a logiky funkce zajišťující kontrolu zobrazovaného textu. Nutnost využití původně zadaného textu pro možnost realizace kontroly.

2.4 ČTVRTÁ SEMINÁRNÍ ÚLOHA

Čtvrtá seminární úloha seznamuje studenty cyklem s podmínkou na začátku. Studenti v této úloze také pracují s matematickou třídou, již využijí pro generování náhodného čísla. Dále se seznámí s možností navrátit hodnotu z funkce.

2.4.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, již bude umožňovat hádat náhodně vygenerované číslo. Číslo bude vygenerováno v rozsahu, jenž bude stanoven pomocí dvou vstupních hodnot. Tipování náhodného čísla probíhá uvnitř modulárního okna. Při zadání čísla se dozvíme, zdali námi hledané číslo je menší než náš tip, větší než náš tip nebo se nám podařilo číslo uhádnout. V takovém případě dojde k přerušení hádání a dojde k výpisu hodnoty hledaného čísla a množství pokusů, které jsme na uhádnutí potřebovali.

Také v této úloze musí studenti ošetřit pomocí podmínek konkrétní stavy a události, které by se mohly vyskytnout. Úloha neobsahuje žádnou práci s CSS, studenti pracují pouze s částí HTML a JS.

Obrázek 4: Čtvrtá seminární úloha (Zdroj: vlastní)

2.4.2 REALIZACE ÚLOHY

Stejně jako u předešlých úloh i zde musí studenti jako první vytvořit základní strukturu HTML. K tomu využijí následující značky:

- **Značka p**: Textový odstavec, bude sloužit pro zobrazení hodnoty uhádnutého čísla a množství pokusů, jenž k tomu bylo za potřebí.
- **Značka h1** – **h6**: Nadpis první až šesté úrovně, studenti pomocí této značky zobrazí popis aplikace.
- **Značka button**: Tlačítko, pomocí jehož studenti realizují funkci načtení dvou čísel, pomocí kterých vznikne rozsah hádaného čísla.
- **Značka input:** Dva vstupy, do nichž bude možné zadávat krajní hodnoty rozsahu pro generování náhodného čísla.

Studenti se při tvorbě struktury HTML seznámí s atributem placeholder, jenž umožňuje do vstupů vložit výchozí text.

Prvním dílčím úkolem při tvorbě kódu v JS je nutnost vytvořit funkci pro zadávání čísel. Zajistit načtení dvou čísel by studentům nemělo činit obtíže, protože obdobnou problematiku řešili již v předešlých úlohách. Tato funkce bude také jediná, která bude provázaná s částí HTML. Všechny ostatní funkce budou poté volány uvnitř této funkce v části JS. Také zde je nutné pomocí podmínek ošetřit možné situace, které by mohly ovlivnit výslednou funkcionalitu programu. Studenti tedy musí při načítání čísel ošetřit následující:

- Kontrola, zdali jsou obě zadané hodnoty číselného typu. V případě, že by tomu tak nebylo, budeme na tuto skutečnost upozorněni a není možné spustit generování náhodného čísla.
- Kontrola hodnot zadaných čísel. Na samotné stránce není možné zajistit, aby hodnota pro spodní hranici rozsahu byla nižší než ta horní. To se musí provést právě až v části JS. V případě, že dojde k zadání hodnot, kde spodní bude větší než horní, musí v kódu dojít k přiřazení těchto hodnot do příslušných proměnných, aby bylo možné číslo správně vygenerovat.
- Kontrola, že zadané hodnoty nejsou duplicitní. Poté by nebylo možné vytvořit číselný rozsah.

V případě, že program splní tyto podmínky, máme úspěšně uložené dvě hodnoty ve dvou příslušných proměnných. Studenti, tedy musí vytvořit a následně zavolat ve vhodném místě novou funkci s parametry. Vstupními parametry budou při volání funkce právě tyto dvě proměnné. Tato funkce vygeneruje na základě předaných hodnot náhodné číslo a následně jej vrátí do příslušné proměnné. Funkce bude vytvořena následovně:

- Vytvoření funkce a příslušných parametrů.
- Využití vhodných metod matematické třídy uvnitř JS.
- Využití návratové hodnoty.

V posledním dílčím kroku je nutné vytvořit funkci, jíž bude umožňovat hádání vygenerovaného čísla. To bude realizováno cyklicky pomocí modulárního okna. Cyklus bude probíhat do té doby, dokud nedojde k uhodnutí čísla nebo se nerozhodneme hádání ručně ukončit. Studenti jsou povinni v této funkci ošetřit následující:

- V případě, že dojde k zadání menšího čísla, než je číslo vygenerované, budeme na tuto skutečnost upozorněni.

- V případě, že dojde k zadání většího čísla, než je číslo vygenerované, budeme na tuto skutečnost upozorněni.
- V případě, že uhádneme číslo, dojde k přerušení cyklu a provede se výpis s hodnotou čísla a počtem pokusů, jenž byly k uhodnutí za potřebí do příslušného textového odstavce.
- V případě, že se rozhodneme hádání ručně ukončit, budeme na tuto skutečnost upozorněni a bude nám odhaleno hádané číslo.

2.4.3 ZÁVĚR ÚLOHY

Tato úloha byla zařazena do sady seminárních úloh až v akademickém roce 2023/2024. Z toho důvodu nebylo zatím možné stanovit často vyskytující se chyby při realizaci úlohy.

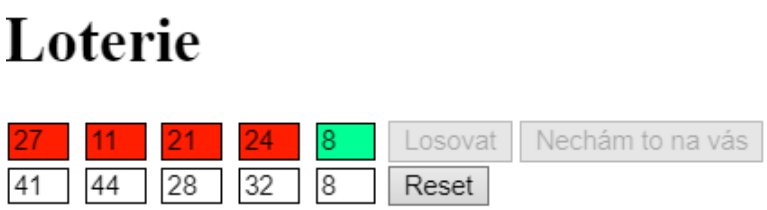
2.5 PÁTÁ SEMINÁRNÍ ÚLOHA

Tato seminární úloha byla nově zavedena do výuky v akademickém 2023/2024. Úloha je zejména zaměřena na práci s polem, ale také využívá již získaných poznatků studentů z oblasti funkcí s parametrem a řídících struktur. Doposud se jedná o nejkomplexnější úlohu.

2.5.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, jíž bude umožňovat uživateli zadávat čísla z předem stanoveného intervalu. Uživatel bude mít také možnost vynechat zadávání vlastních čísel a bude moci využít náhodné generování čísel ze stejného intervalu. Uživatelovo vstupní čísla budou následně porovnána se sadou dalších, náhodně vygenerovaných čísel a bude hledaná případná shoda. O shodě či neshodě zadaných čísel bude uživatel informován pomocí upozorňovacího okna a také za pomocí vhodného vizuálního zobrazení. Tato úloha obsahuje také značné množství událostí a stavů, které je nutné ošetřit pomocí podmínek.

Úloha obsahuje také práci se styly, kdy je vždy nutné zajistit zbarvení políčka obsahujícího čísla příslušnou barvu. Podle shody či neshody s náhodně vygenerovanými čísly.



Obrázek 5: Pátá seminární úloha (Zdroj: vlastní)

2.5.2 REALIZACE ÚLOHY

Již tradičně je nutné na začátku vytvořit základní strukturu HTML. Oproti předešlým úlohám je ovšem práce v této části složitější a vyžaduje od studentů zamyšlení se nad následující prací v ostatních částech. To je způsobeno tím, že budeme načítat více vstupních hodnot, zvíce vstupů najednou. K vytvoření základní struktury HTML budou v tomto případě využity následující značky:

- **Značka input:** Vstupy budou sloužit pro zadávání jednotlivých hodnot uživatele. Zároveň budou vstupy využity pro zobrazení náhodně vygenerovaných čísel, s nimiž budou uživatelova čísla porovnávána.
- **Značka button:** Pomocí tlačítek bude realizováno potvrzení zadání uživatelovo čísel. Možnost pro uživatele si nechat svá čísla náhodně vygenerovat a také provést restart celé aplikace.

Klíčové v této části je vhodné využití atributů. S jejich pomocí bude možné realizovat načítání jednotlivých vstupních hodnot a také provádět výpis výstupních hodnot. Pomocí vhodných atributů bude taky možné následně aplikovat příslušné styly. Oproti předešlým úlohám budeme provádět změnu hodnoty atributu přímo uvnitř části JS. Pro správnou realizaci budou klíčové následující atributy:

- **Atribut id**: Vlastnost je použita na všechny prvky, důležité je ovšem zvolení vhodné hodnoty vlastnosti u vstupů. Zde musíme zvolit hodnoty, které budou jednoduše umožňovat načítat vstupní čísla při průchodu cyklem v části JS.
- **Atribut class**: Aplikovaná na všechny vstupy, podle shody či neshody čísla dojde k přiřazení do příslušné třídy a následné aplikaci stylu.
- **Atribut readonly**: Použitu u vstupů sloužící pro zobrazení výstupních hodnot, zamezuje zadávání a přepisování. Jedná se o obdobu atributu disabled, s nímž se již studenti seznámili.

V prvním kroku se jeví jako logické vytvoření funkce umožňující načtení uživatelem zadaných čísel. My si vytvoříme pomocnou funkci, jíž bude kontrolovat správnost načtení čísel, ale zatím nebude provádět žádné porovnávání. Načítaní realizujeme z více vstupů najednou, proto využijeme cyklus s daným počtem opakování. Ten nám umožní zkontrolovat hodnoty jednotlivých vstupů. Zde se projevuje poprvé nutnost vhodného využití *atributu ID*, kdy při každém průchodu uvnitř cyklu musíme přistoupit k dalšímu vstupu a získat jeho hodnotu. Takové přistupování je realizované pomocí vhodného využití indexu při průchodu cyklem a spojením s pevně daným názvem HTML prvku.

Při kontrole načítání čísel musíme ošetřit následující:

- Kontrola, zdali došlo k vyplnění všech vstupů určených pro zadávání vstupních hodnot.
- Nutnost zadávat pouze čísla, není možné zadat jakoukoliv jinou hodnotu.
- Kontrola duplicitních hodnot, možnost zadávat pouze unikátní hodnoty.
- Kontrola číselného intervalu zadávaných čísel, možnost zadávat čísla pouze z předem daného intervalu.

Funkce slouží pouze jako pomocná pro kontrolu zadávaných hodnot a bude následně využita jako kontrola pro spuštění funkce porovnávající čísla. Z toho důvodu je funkce ukončena návratem logické hodnoty.

V dalším dílčím kroku je možné vytvořit funkci, již bude provádět samotné načtení, vygenerování a následnou kontrolu čísel. Tato funkce bude obsahovat více pomocných funkcích a bude uvnitř ní také voláno více funkcí s parametry. Funkce bude realizována následovně:

- Vytvoření funkce samotné a na začátku funkce deklarování tří proměnných představující tři různá pole. První pole bude sloužit pro ukládání načtených čísel uživatele. Druhé pole bude obsahovat vygenerovaná čísla, s nimiž budeme uživatelova čísla porovnávat. Třetí pole bude obsahovat případná, shodující se čísla.
- Využití již před vytvořené funkce zajišťující kontrolu načítaných hodnot. V případě, že kontrola proběhne úspěšně vstupujeme dovnitř cyklu s daným počtem opakování. V opačném případě dojde k upozornění uživatele na zjištěné chyby.
- Při každém průchodu cyklem dochází k načtení vstupní hodnoty z příslušného vstupu a vložení této hodnoty do příslušného pole. Provádí se také vygenerování náhodného čísla pomocí vytvořené funkce a jeho vložení také do příslušného pole. Toto číslo je taky vždy rovnou vypsáno do příslušného výstupu.
- Po provedení načtení a vygenerování čísel je nutné provést kontrolu těchto dvou polí. K tomu poslouží také vytvořená pomocná funkce. Tato funkce představuje studentům práci s vnořenými cykly, kdy dochází k postupnému porovnávání vždy hodnoty z jednoho pole se všemi ostatními hodnotami z pole druhého.
- V případě, že dojde ke shodě mezi hodnotami, je tato hodnota vložena do příslušného pole.
- V závěru je nutné zkontrolovat, zdali doopravdy došlo ke shodě mezi čísly. V takovém případě dojde k výpisu tohoto pole prostřednictvím upozorňovacího okna.

Na základě vyhodnocení shody či neshody je u vstupů uvnitř JS změněna hodnota u atributu class. Následně podle toho dojde k aplikování příslušných stylů.

V posledním kroku realizujeme funkci, jíž umožní uvést celou aplikaci do svého původního stavu.

2.5.3 ZÁVĚR ÚLOHY

Tato úloha byla zařazena do sady seminárních úloh až v akademickém roce 2023/2024. Z toho důvodu nebylo zatím možné stanovit často vyskytující se chyby při realizaci úlohy.

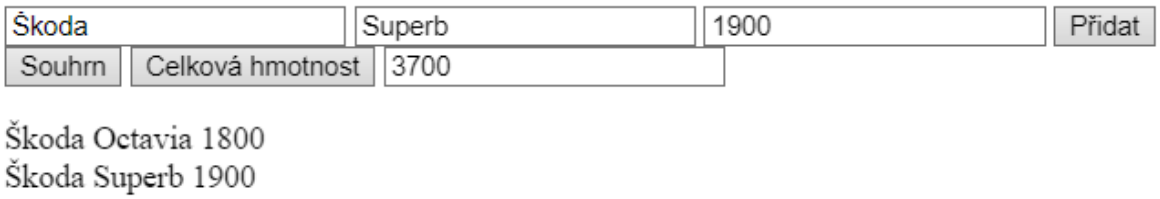
2.6 ŠESTÁ SEMINÁRNÍ ÚLOHA

Šestá seminární úloha je zejména zaměřena na práci s objekty, cykly a polem. Studenti jsou seznámeni s možnostmi vytváření objektů, jejich ukládáním do pole a prací s polem obsahující objekty. [17]

2.6.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, jíž bude umožňovat evidovat záznamy o automobilech. Uživatel bude moci vytvořit záznam, jenž bude uložen do pole jako jeden objekt. Aplikace bude umožňovat kdykoliv zobrazit všechny evidované záznamy a vedle toho bude možné zjisti celkovou hmotnost všech evidovaných záznamů.[17]

Úloha neobsahuje žádnou práci se styly.



Obrázek 6: Šestá seminární úloha (Zdroj: vlastní)

2.6.2 REALIZACE ÚLOHY

Před vytvářením kódu samotného je nutné vytvořit základní strukturu HTML. K tomu využijeme následující HTML značky:

- **Značka input:** Vstupy budou sloužit pro zadání jednotlivých informací o každém objektu. Jeden vstup bude opatřen atributem umožňující pouhé čtení a bude určen pro zobrazení celkové hmotnosti všech záznamů.
- **Značka button:** Tlačítka budou sloužit pro provázání s jednotlivými funkcemi. Bude pomocí nich realizováno přidávání jednotlivých záznamů, výpis celé evidence a celkové hmotnosti všech evidovaných záznamů.
- **Značka p:** Textový odstavec bude sloužit pro zobrazení celé evidence záznamů.

Jako první v části JS realizujeme funkci umožňující vytváření záznamů v evidenci. Zde jsou studenti zejména seznámeni se syntakticky správným zápisem objektů a přistupováním k jejich jednotlivým vlastnostem pomocí tečkové notace. Opět je nutné zajistit kontrolu zadávaných hodnot pomocí podmínek. Pokud jsou všechny stanovené podmínky splněny dojde k přidání objektu do příslušného pole [17].

Další realizujeme funkci zajišťující výpis celé evidence, zde studenti objeví skutečnost, že pole obsahující objekty není možné vypisovat jako pole doposud. Studenti jsou tedy seznámení s nutností přistoupit k objektu vždy na konkrétním indexu a přistoupit vždy k jeho požadované vlastnosti opět pomocí tečkové notace [17].

V posledním kroku zajistíme získání hmotnosti všech záznamů uvnitř evidence. Jedná se o velmi obdobný problém jako v předešlé funkci. Z toho důvodu je i toto řešení velmi podobné. Musíme zajistit pouze sečtení všech hodnot u požadované vlastnosti všech objektů a následně výsledek vypsat [17].

2.6.3 CHYBY PŘI REALIZACI

Často se vyskytuje chyba spojená s výpisem pole, v němž jsou záznamy evidovány. Studenti se pokouší pole vypsat tradičním způsobem, to ovšem není u pole obsahující objekty možné. [17]

Další chybou je obvykle špatné použití cyklu s daným počtem opakování, kde studenti chybují při zápisu podmínky a cyklus neproběhne podle požadovaných předpokladů. [17]

2.6.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace druhé seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

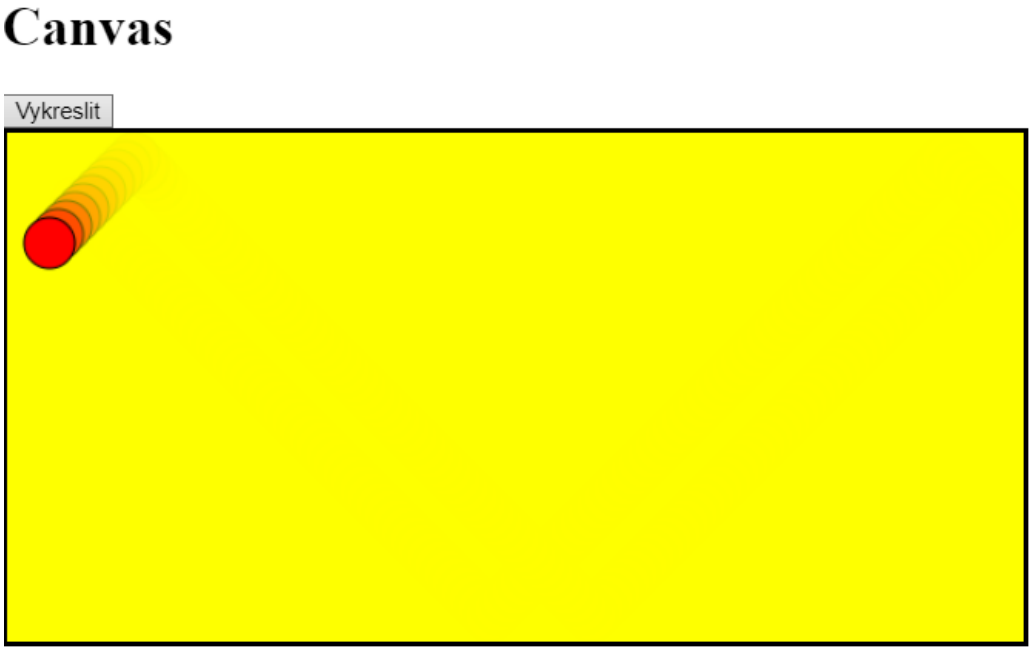
- Vysvětlení práce s polem obsahující objekty a realizace výpisu takového pole.
- Stanovení správné podmínky při průchodu cyklu s daným počtem opakování. Zdůraznění využití vlastnosti pole zjišťující jeho délku.

2.7 SEDMÁ SEMINÁRNÍ ÚLOHA

Sedmá seminární úloha seznamuje studenty s canvsem (tj. kreslící plátno) a jeho základními nástroji. Oproti předešlým úlohám se zde více vyskytuje spolupráce mezi vyučujícím a studenty. [17]

2.7.1 POPIS ÚLOHY

Tato úloha nemá oproti předešlým jasně stanovený cíl a je tvořena segmenty v nichž vyučující studentům představuje různé nástroje určené pro práci s canvasem. Výsledkem úlohy tedy není komplexní aplikace, nýbrž jednotlivé části, které by mohly být v komplexním příkladu využity. [17]



Obrázek 7: Sedmá seminární úloha (Zdroj: vlastní)

2.7.2 REALIZACE ÚLOHY

V této úloze se vyskytuje práce zejména v části JS, kdy v části HTML studenti vytvoří pouze základní strukturu pomocí následující značky:

- **Značka canvas:** Vytvoří plátno, na které je následovně možné kreslit a přidávat libovolné obrazce a tvary.

Po vytvoření plátna a spuštění aplikace se neprojeví žádné změny. Plátno se na stránce skutečně vyskytuje, ovšem ve výchozím nastavení má bílou barvu pozadí a tudíž splývá. V části se styly je tedy nutné nastavit pro plátno barvu pozadí, případně jeho ohraničení. Také je možné nastavit rozměry plátna.

V první části jsou studenti seznámeni s možností vytvoření obrazce pomocí jednotlivých křivek. Takový postup seznámí studenty se souřadnicovým systémem uvnitř plátna a se základními metodami. Ovšem toto řešení je nepraktické a nese s sebou značné nevýhody pro další práci s vykresleným obrazcem. [17]

Vzhledem k nedostatkům předchozího řešení jsou studenti v další části seznámeni s metodou umožňující rovnou vykreslit požadovaný obrazec. Studenti jsou v tomto případě také seznámeni s nutností otevírání a uzavíraní jednotlivých cest při vykreslování. To je jedna z klíčových vlastností ovlivňující výsledek vykreslení. [17]

Dalším krokem je následné uvedení obrazce do pohybu. To je realizováno pomocí zvyšujících a zmenšujících hodnot uvnitř proměnných, které použijeme jako souřadnicové body obrazce. Důležitá je zde nutnost již vždy vykreslený obrazec smazat, aby nezůstal vykreslená na plátně a tím bylo dosaženo efektu posunu. [17]

Posledním dílčím úkolem je vykreslení kruhu a jeho následné rozpohybování. Rozdíl oproti předešlým částem je v použité metodě pro vykreslení požadovaného obrazce. Také je zde chtěný efekt odrazu obrazce od hranic plátna a jeho následný pohyb opačným směrem. [17]

2.7.3 CHYBY PŘI REALIZACI

Chyba, již se vyskytuje zejména v první části při vykreslování křivek je způsobeno syntaktickým zápisem vykreslování. Studenti sice správně vytvoří křivku, ovšem následně ji zapomenou vykreslit pomocí příslušného příkazu. [17]

V druhé části se poté vyskytuje chyba, kdy studenti zapomenou zavřít a znovu otevřít cestu pro vykreslení konkrétního obrazce. [17]

Chyba, již se začíná vyskytovat ve třetí části při úvodním rozpohybování obrazce, kdy studenti zapomínají smazat celé plátno. Před tím, než začnou vykreslovat nový obrazec, tím zůstává na plátně vykreslené velké množství již nežádoucích obrazců. [17]

Ve všech částech se vyskytuje chyba, kdy studenti špatně vypočítají nebo určí souřadnice vykreslovaného objektu. Dále činí studentům problém vhodně určit, v jaký moment a jakým způsobem se mají obrazce odrazit od okraje plátna. [17]

2.7.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace druhé seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

- Zdůraznění správného syntaktického zápisu při vykreslování a mazání obrazců.
- Nutnost otevřít a zavřít vždy cestu pro vykreslení jednotlivých obrazců.

- Objasnění logiky při uvádění obrazců do pohybu a jejich chování při dosažení hranic plátna.

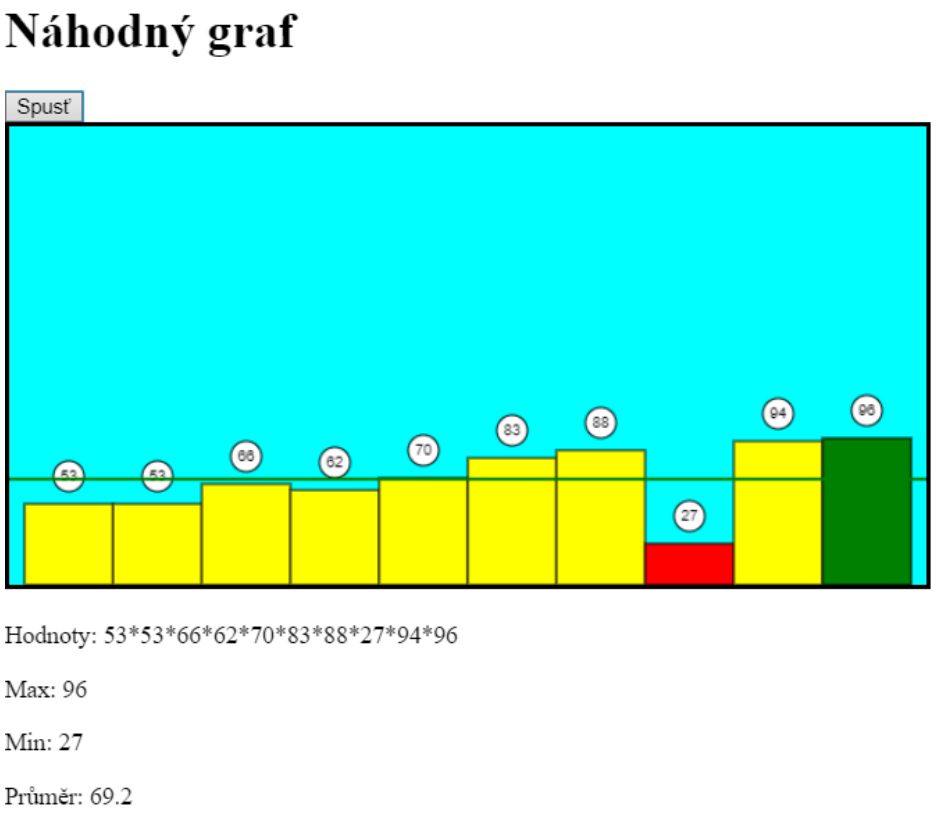
2.8 OSMÁ SEMINÁRNÍ ÚLOHA

Závěrečná seminární úloha navazuje na předešlou seminární úlohu a prohlubuje práci s canvasem a jeho nástroji. Mimo jiné zde ale také studenti uplatní své znalosti a dovednosti s cykly, poli a generováním náhodných čísel. [17]

Na velmi podobné funkcionalitě a logice této úlohy je vytvořen příklad v praktické části zkoušky.

2.8.1 POPIS ÚLOHY

Cílem je vytvořit webovou aplikaci, již bude generovat náhodné číselné hodnoty, které budou následně zobrazeny v podobě sloupcových grafů uvnitř canvasu. Z náhodně vygenerovaných hodnot budeme chtít vždy určit tu minimální, maximální a také hodnotu průměrnou. Tyto krajní hodnoty budou zobrazeny v canvasu příslušnou barvou a průměrná hodnota bude znázorněna pomocí přímky protínající všechny grafy v dané výšce. V akademickém roce 2023/2024 došlo k drobné modifikaci úlohy. Nyní musí studenti nad každým grafem zobrazit číselnou hodnotu, kterou graf reprezentuje. [17]



Obrázek 8: Osmá seminární úloha (Zdroj: vlastní)

2.8.2 REALIZACE ÚLOHY

Práce s částmi HTML a CSS je naprosto totožná s prací v předešlé seminární úloze. V HTML se vyskytuje tlačítko, pomocí kterého bude spuštěno vygenerování náhodných hodnot. Zároveň dojde k určení minimální, maximální a průměrné hodnoty. Spustí se také vykreslení všech příslušných obrazců a křivek do canvasu. [17]

Jako první je nutné vytvořit funkci, již nám vygeneruje náhodná čísla a uloží je do příslušného pole. Generování je vhodné ošetřit, abychom zamezili vzniku hodnot, které by následně byly v grafu nepatrné nebo by naopak přesahovaly mimo plátno. Pro generování využijeme cyklus s daným počtem opakování, kdy při každém průchodu vygenerujeme hodnotu, již následně uložíme do pole. [17]

Když máme připravené hodnoty v poli je nyní možné zjišťovat požadované krajní hodnoty a hodnotu průměrnou. Pro zjištění minimální a maximální hodnoty je vhodné využít již před vytvořenou matematickou metodu, která tyto hodnoty z pole automaticky určí. Funkci pro výpočet průměrné hodnoty musíme vytvořit za pomoci cyklu s daným počtem opakování. Uvnitř něho vypočítáme aritmetický průměr hodnot v poli. [17]

Takto vytvořené funkce můžeme vhodně využít i pro funkci vykreslující samotný graf, kde tyto krajní hodnoty chceme znázornit požadovanou barvou. U vykreslování grafů je problematická práce se souřadnicemi, protože souřadnice u každého obrazce udávají pozici jeho levého horního rohu. Proto je nutné hodnotu z pole vždy odečíst od celkové výšky plátna. Vykreslení a zbarvení jednotlivých obrazců je opět realizováno pomocí cyklu s daným počtem opakování. [17]

Poslední funkcí zajistíme vykreslení přímky znázorňující průměrnou hodnotu pole. Zde opět vhodně využijeme funkci, kterou jsme si již připravili a průměrnou hodnotu znázorníme opět odečtením této hodnoty od celkové výšky plátna. [17]

2.8.3 CHYBY PŘI REALIZACI

Jedna z častých chyb je způsobena vlastnostmi JS, kdy pracujeme s dynamickými datovými typy. Tudíž JS automaticky převede hodnoty uložené v poli na textový řetězec, a proto se následné matematické operace jeví jako chybné. [17]

Chyba, která se vyskytuje i v předešlé seminární úloze, je způsobena neuzavřením cesty při vykreslování jednotlivých obrazců. Poté dochází k nesprávnému vykreslení nebo obarvení následujících obrazců. [17]

Objevuje se zde taky logická chyba, kdy studenti špatně určují souřadnice vykreslovaných obrazců nebo zapomínají na posun mezi jednotlivými grafy, čímž dochází k vykreslení všech grafů na jednom místě. [17]

Poslední domnělou chybu studentů vysvětluje Frank a Zíka následovně: „*V této úloze se pak objevuje studenty domnělá chyba. Při generování náhodných čísel se někdy může stát, že se vygeneruje několik stejných čísel a zrovna jde o maxima nebo minima. V takovém případě, pokud je vše správně naprogramováno, se nám objeví několik zelených sloupců nebo několik červených sloupců. Jedná se o normální běh programu, protože vše jsou to maxima a minima. Studenti však často toto chování považují za špatné vyhodnocení a tráví poměrně dlouhý čas nad opravou. Situace obvykle končí dotazem na vyučujícího* [17].“

2.8.4 ZÁVĚR ÚLOHY

Vzhledem k zjištěním chybám, jež se nejčastěji vyskytují během realizace druhé seminární úlohy je nutné při vytváření multimediálního prvku, zabývajícího se řešením této úlohy, klást důraz na následující:

- Práce s dynamickými datovými typy, při provádění matematických operací a funkcí je nutné přetypovat proměnou na číslo.
- Otevření a uzavření cesty pro vykreslení křivek.
- Vysvětlení logiky souřadnice os X a Y pro správné zobrazení vykreslených grafů.
- Vysvětlení logiky celého příkladu a zdůraznění logiky podle níž se vykreslené grafy zobrazují.

2.9 ZÁVĚR ANALÝZY ÚLOH POUŽÍVANÝCH V PŘEDMĚTU KVD/PGM1P

Tato kapitola s názvem „Úlohy používané v předmětu KVD/PGM1P“ si kladla za cíl zanalyzovat a popsat jednotlivé úlohy realizované během seminářů předmětu KVD/PGM1P. Všechny úlohy, představené v této sadě, byly využity ve výuce v zimním semestru akademického roku 2023/2024. Každé úloze byla věnována samotná podkapitola, ve které jsme se zejména snažili úlohu představit a podrobně popsat na základě naší vlastní realizace. K zjištění chyb při realizaci jednotlivých úloh ve výuce posloužili zejména poznatky Franka a Zíky, kteří je popsali ve své publikaci na základě svých zkušeností

s výukou předmětu KVD/PGM1P. Vzhledem ke zjištěným chybám jsme vždy v závěru kapitoly stanovili body, na které je nutné v multimediálním prvku, popisující realizaci úlohy, vždy upozornit.

Analýza odhalila u každé úlohy několik chyb, kdy se v některých případech opakují chyby z předešlé úlohy. První častou chybou ve všech případech je špatný syntaktický zápis nebo neznalost sémantiky programovacího jazyku JS. Druhou nejčastější chybou je poté stanovení chybné logiky, kterou není možné úlohu následovně realizovat.

3 NÁVRH VZDĚLÁVACÍHO E-KURZU PRO PŘEDMĚT KVD/PGM1P

Naším cílem je navrhnout a vytvořit vzdělávací e-kurz v prostředí LMS Moodle, který bude sloužit jako studijní podpora pro předmět KVD/PGM1P. Součástí vytvořeného kurzu budou vedle studijních textů také námi vytvořené multimediální a interaktivní prvky.

3.1 LMS MOODLE

Pojem LMS (Learning Managment System) je Václavem Maněnou v knize „Moderně s Moodlem“ definován a představen následovně. „*LMS představuje komplexní e-learningovou platformu, která kromě nástrojů pro elektronické vzdělávání obsahuje další funkce pro řízení aktivit uživatelů v jednotlivých e-learningových kurzech i v rámci celého systému.* [18]“ LMS Moodle (dále jen „Moodle“) je tedy takovou platformou, která umožňuje pomocí předpřipravených nástrojů vytvářet a spravovat vzdělávací e-kurzy.

Moodle je open source platformou, jež byla založena v roce 2002 a jejím autorem je Australan Martin Dougiamas. Moodle se v dnešní době těší velké popularitě napříč celým světem. V březnu, roku 2020 dosáhl počet registrovaných uživatelů přes hodnotu sto devadesát milionů. V té době byl Moodle využíván přibližně na sto čtyřiceti pěti tisících unikátních webových stránkách. [18][19]

Otevřenost a dostupnost Moodlu je právě jednou z jeho největších předností. Nejedná se o platformu, kterou využívají pouze vzdělávací instituce. Je také hojně využíván organizacemi nebo firmami. Právě tyto komerční instituce se ve velké míře podílejí na jeho vývoji. [18]

„*Moodle umožňuje:*

- *Vytváření tematicky i časově orientovaných kurzů.*
- *Automatickou i ruční klasifikaci studentů a účastníků kurzu. Vytváření a používání úloh, banky úloh, rozdělení úloh dle obtížnosti, časové a další možnosti pro testování, klasifikaci nebo opakování studentů.*
- *Autentizaci uživatelů s externími systémy (email, adresáře uživatelů, Google, atp.). Systém umožňuje import uživatelů a kurzů z externích databází a zdrojů dat (např. formátu CSV).*
- *Propojení s externími aplikacemi a zdroji (např. vkládání a zpracování HotPotato, vkládání a přehrávání multimediálních souborů, integraci s Google diskem, Wikimédií, MS Office 365, atp.).*

- *Plnou kontrolu nad systémem (běh na vlastním nebo zprostředkovaném počítači), komunitní nebo placenou podporu komerčních firem. LMS Moodle nevyžaduje další poplatky za licence SW, ke svému běhu používá tzv. LAMP (Linux, Apache, MySQL, PHP).*
- *Používání různých škál hodnocení dle potřeby (zápočty, klasické i slovní známkování, známkování různých kritérií řešení úkolu – např. jazykovou správnost, odbornou kvalitu atp.). Sledování pokroku a plnění zadaných úkolů od studentů, sledování odpovědí studentů a export či import známek a výsledků do externích systémů.*
- *Diskutování v diskuzních fórech, vytváření společných materiálů (např. slovníků a databází – např. S popisnými fotografiemi fauny a flóry), tvorbu dotazníků a online studijních materiálů včetně delších knih s možností rychlého tisku nebo přechodu na určitou kapitolu knihy.*
- *Spolupráci např. S LaTeXem pro tzv. matematickou sazbu (generování obrázků s matematickými vzorci dle speciálního zápisu TeX)* [18].“

V tomto můžeme spatřit širokou využitelnost platformy Moodle. Pro nás je ovšem stěžejní hned první bod. Tedy možnost vytvořit a spravovat vzdělávací kurz. Ten námi vytvořený bude tedy tematicky orientovaný na předmět KVD/PGM1P.

3.2 NÁVRH OBSAHU A STRUKTURY E-KURZU PRO PŘEDMĚT KVD/PGM1P

Při návrhu a následné realizaci našeho vzdělávacího kurzu jsme usilovali zejména o vytvoření kvalitního obsahu kurzu. Obsah celého kurzu je také označován jako distanční text. Maněna tento pojem popisuje a definuje následovně: „*Vychází z odborných materiálů, které jsou metodicky zpracovány tak, aby studující mohl bez problému pracovat s tímto materiálem a současně byl dostatečně motivován a povzbuzován k dalšímu studiu. Distanční text je tedy text, který je svou strukturou a grafikou upravený tak, aby umožňoval samostudium.* [18]“ Našim hlavním cílem bylo tedy navrhnout obsah kurzu, který bude odpovídat výše zmíněným podmínkám.

3.2.1 NÁVRH DISTANČNÍHO TEXTU

Při návrhu distančního textu jsme dodržovali zásady, které popisuje Maněna [18]:

- Přizpůsobení textu cílové skupině,
- rozdělení textu na menší celky,
- srozumitelnost textu,
- názornost,
- motivace a aktivizace studenta.

Distanční text byl měl být navržen a přizpůsoben vzhledem k vstupnímu dovednostem studentů. Od toho by se měla odvíjet složitost a jazyková srozumitelnost textu. Srozumitelnost je důležitá i u grafických prvků textu. [18]

Náš text byl přizpůsoben faktu, který byl již zmíněn v úvodu druhé kapitoly. A to sice, že vstupní dovednosti a vědomosti studentů jsou na velmi rozdílné úrovni. Text zásadně neobsahuje cizojazyčné pojmy. V případě, že tomu tak je, jsou pojmy detailně vysvětleny. Z toho důvodu se také v textu nenacházejí žádné přebrané grafické prvky, které by mohly obsahovat odborné cizojazyčné pojmy.

Při návrhu textu bychom měli vždy myslet na hlavní myšlenky a pod myšlenky jednotlivých témat. K tomu může pomoci vytvoření jednoduché osnovy v níž budou jednotlivá témata a pod témata rozepsána. Takto rozdělený text představuje jednotlivé kapitoly distančního textu. [18]

„*Rozumně zvažte zařazení multimediálních prvků v textu. V případě učebních textů velmi často platí, že méně je více*. [18]“

Při návrhu jsme se rozhodli, že text bude v celém kurzu doplňován pomocí multimediálních prvků následujícími způsoby:

- **Obrázky se syntaktickým zápisem**: Tyto obrázky budou pomocí popisků představovat studentům syntakticky správný zápis.
- **Obrázky s útržky kódu**: Tyto obrázky budou zobrazovat část kódu, jenž bude doplněn o komentáře popisující funkcionalitu této části.
- **Videa**: Text bude doplněn o videa vždy vystihující nebo rozšiřující hlavní myšlenku studijního textu. Dále budou videa také sloužit pro zobrazení realizace seminárních úloh, jež byly analyzovány ve druhé kapitole. Tyto videa budou doplněna o mluvený komentář.

Navrženy by měly být také úkoly, jež umožní studentům ověřit získané vědomosti a dovednosti v dané kapitole. Také by měl být navržen komplexní úkol, pro jehož vyřešení budou muset studenti aplikovat veškeré získané poznatky. [18]

Pro každou kapitolu jsme navrhnuli interaktivní činnost, která právě slouží k ověření získaných vědomostí a dovedností. Tato činnost bude ověřovat jak teoretické, tak praktické poznatky studentů. Do každé kapitoly je také zařazena jedna ze seminárních úloh, které byly analyzovány v druhé kapitole.

3.2.2 NÁVRH STRUKTURY KAPITOL

Distanční text by měl být rozdělen do co nejmenších částí, tyto části představují následně jednotlivé kapitoly. [18] Stejně jako při návrhu distančního textu jsme při návrhu struktury studijních kapitol dodržovali zásady, které stanovuje Maněna.

„*Každá kapitola by měla dodržovat následující strukturu:*

- *Cíle,*
- *časová náročnost,*
- *klíčové pojmy,*
- *výukový text,*
- *otázky,*
- *úkoly a cvičení,*
- *testy,*
- *souhrn,*
- *doporučená a použitá literatura* [18].“

Cíl by mě být vždy jednoznačný, konkrétní a splnitelný. Splnitelnost je důležitá jak ze strany učitele, tak ze strany studenta. Pro studenty představuje stanovený cíl, v úvodu každé kapitoly, důležitý informační prvek. Sděluje jim, jaké znalosti a vědomosti by po absolvování studijní kapitoly měli získat. [18]

Dalším důležitým informačním prvkem, v úvodu každé studijní kapitoly, je časová náročnost. Ta by měla studentům jasně říct, kolik času musí studiu kapitoly věnovat. Časová náročnost se nevztahuje pouze k přečtení a porozumění studijního textu. Zahrnuje také čas, který je nutný pro splnění všech úloh ověřující získané znalosti a dovednosti. Stanovení takového časového údaje je velmi subjektivní a pro každého studenta se potřebný čas ke studiu může lišit. Nemělo by však nikdy dojít ke stanovení příliš podhodnocené nebo nadhodnocené časové náročnosti. To může vést k demotivaci a pocitu méně cennosti studenta. Studenti mohou také studium vzdát hned v počátku, protože si budou myslet, že nemají tolik času. V případě, že je náročnost nadhodnocena a studenti kapitolu zvládnou v mnohem kratším čase, mohou poté nabýt dojmu, že je kurz příliš lehký a pro ně nemá smysl v něm dále studovat. [18]

Klíčové pojmy jsou obvykle cizojazyčné pojmy, zkratky nebo termíny, které ve výukovém textu nemůžeme vzhledem jeho k odbornému významu nahradit. Takové slovo by mělo být při jeho prvním výskytu, vždy důkladně vysvětleno. [18]

„*Výukový text představuje samotné nové učivo a je vhodné ho upravit podle zásad uvedených v předcházejícím textu (důraz na jednoduchost, srozumitelnost, názornost, aktivizace studujícího apod.)*. [18]“

V sekci, otázek, úkolů a cvičení budeme v naších kapitolách využívat interaktivní činnosti a úlohy, které byly popsány již v předešlé kapitole.

„*Každá kapitola by měla v závěru obsahovat jasné a stručné shrnutí toho nejdůležitějšího, o čem kapitola pojednávala. Souhrn má pro studující velký význam a usnadní jim např. opakování a přípravu na závěrečný test. Čím rozsáhlejší je kapitola distančního textu, tím větší význam má souhrn*. [18]“

My jsme se při návrhu kurzu rozhodli, že souhrny nebudeme provádět na konci celé studijní kapitoly, nýbrž jej provedeme vždy na konci každé studijní podkapitoly.

Pro prohloubení znalostí a vědomostí je možné v závěru zmínit doporučenou literaturu. [18]

3.2.3 KONEČNÝ NÁVRH KURZU PRO PŘEDMĚT KVD/PGM1P

Při navrhování struktury a rozložení obsahu jednotlivých kapitol a pod kapitol jsme vycházeli z plánu seminářů předmětu KVD/PGM1P pro akademický rok 2023/2024. Na základě toho jsme navrhli následující rozložení kapitol vzdělávacího kurzu:

- Úvod,
- Vývojové prostředí Visual Studio Code,
- Úvod HTML, CSS a JS,
- Proměnné a operátory,
- Funkce, funkce s parametrem,
- Textové řetězce,
- Řídící struktury,
- Pole,
- Objekty,
- Semestrální práce,

- Závěr.

3.3 ZÁVĚR NÁVRHU VZDĚLÁVACÍHO E-KURZU PRO PŘEDMĚT KVD/PGM1P

Tato kapitola s názvem „Návrh vzdělávacího kurzu pro předmět KVD/PGM1P“ si jako hlavní cíl kladla popsat zásady a principy podle nichž byl vzdělávací kurz v LMS Moodle navržen. V první části jsme se věnovali vysvětlení pojmu LMS a seznámí s platformou Moodle. Seznámili jsme se možnostmi jeho využití a také s jeho základními parametry, vlastnostmi a historií. Moodle je v dnešní době díky své otevřenosti a možnostem jeden z nejpoužívanějších systému LMS ve světe. Pro naše účely jsou nejdůležitější jeho možnosti a nástroje pro tvoření vzdělávacích kurzů.

V druhé části jsme vystihli zásady a principy, které by měli být splněny pro vytvoření efektivního a přínosného vzdělávacího kurzu. Těmito zásadami, které Maněna popisuje ve své knize „Moderně s Moodle“ jsme se následně řídili po celou dobu navrhování všech částí kurzu. Navržená forma distančního textu klade zejména důraz na jednoduchost, srozumitelnost a názornost. Těmito zásadami jsme se následně řídili i při návrhu všech multimediálních prvků, jež budou text v kurzu doplňovat.

Po navržení obsahu kurzu bylo možné navrhnout jeho členění do kapitol. Jejich návrhu a zásadám pomocí, kterých byly navrženy jsme se věnovali v poslední části této kapitoly. Při návrhu jsme mysleli na to, aby každá kapitola měla jasně stanovený jednoznačný a měřitelný cíl. Aby měla stanovenou časovou náročnost, která studentům usnadní přizpůsobit své časové možnosti pro studium. Každá námi navržená kapitola by taky měla obsahovat srozumitelný studijní text a aktivizační prvky.

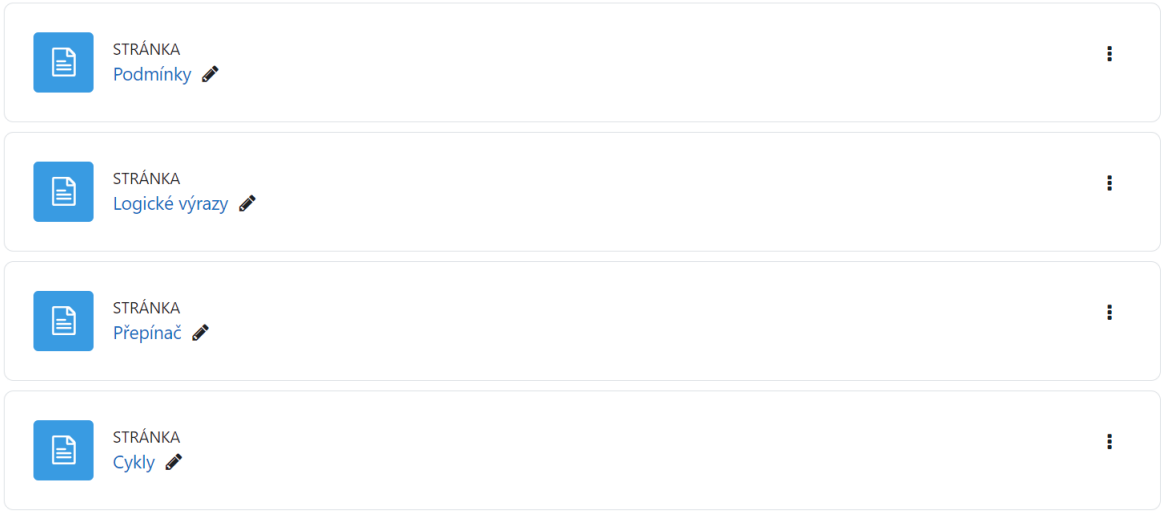
Po návrhu těchto jednotlivých částí bylo možné vytvořit konečný návrh rozložení kapitol uvnitř kurzu. Podle těchto návrhů bude následně celý kurz v Moodle realizován.

4 REALIZACE VZDĚLÁVACÍHO KURZU PRO PŘEDMĚT KVD/PGM1P

Po vytvoření návrhu a seznámení se zásadami a principy pro tvorbu vzdělávacího kurzu bylo možné realizovat náš vlastní kurz v Moodle pro předmět KVD/PGM1P. Naším cílem bylo také vytvořit nové multimediální prvky, jež byly následně do kurzu implementovány.

4.1 TVORBA VÝUKOVÉHO TEXTU

Výukové texty byly realizovány uvnitř jednotlivých studijních podkapitol. Podkapitoly byly vytvořeny pomocí studijní činnosti s názvem *„Stránka“* uvnitř Moodle (viz. Obrázek 1). Uvnitř této činnosti jsme pomocí textového pole a nástrojů pro úpravu textu připravili výukový text tak, aby se shodoval se zásadami a principy popsanými v předešlé kapitole.



Obrázek 9: Rozdělení kapitoly "Řídící struktury" na jednotlivé podkapitoly (Zdroj: vlastní)

Text uvnitř každé studijní podkapitoly je členěn do krátkých odstavců, v nichž jsou důležité pojmy vždy zvýrazněny tučně. Studijní texty uvnitř kurzu byly vytvořeny na základě rešerše následujících online zdrojů:

- MDN Web Docs,
- W3Schools.

4.2 TVORBA MULTIMEDIÁLNÍCH PRVKŮ

V kurzu jsou využity multimediální prvky ve třech podobách. Ty byly již popsány v předešlé kapitole. Pro vytvoření všech multimediálních prvků byly využity nástroje mimo Moodle. Vytvořené prvky byly do Moodlu následně pouze vloženy.

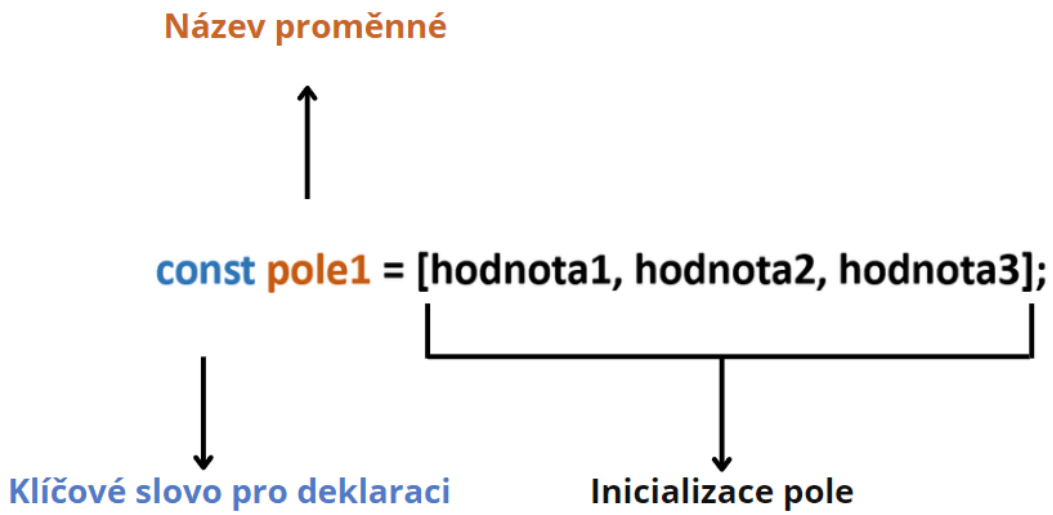
4.2.1 OBRÁZKY SE SYNTAKTICKÝM ZÁPISEM

Tyto obrázky slouží tedy zejména k ukázání a popisu syntakticky správného zápisu kódu. Při tvorbě jsme vždy vytvořili a následně upravili zápis uvnitř textové editoru MS Word (viz. Obrázek 2).

```
const pole1 = [hodnota1, hodnota2, hodnota3];
```

Obrázek 10: Vytvoření a úprava zápisu zachycující vytvoření pole uvnitř JS (Zdroj: vlastní)

Z takto připraveného textu jsme následně pomocí nástroje pro tvorbu snímků obrazovky vytvořili obrázek a vložili jej do online grafického editoru Canva. V tomto editoru jsme pomocí nástrojů provedli úpravy, čímž bylo dosaženo obrázku obsahující příslušné popisky (viz. Obrázek 3). Takto vytvořený obrázek byl následně vložen do příslušné studijní podkapitoly.



Obrázek 11: Úprava obrázku s popisem syntaktického zápisu (Zdroj: vlastní)

4.2.2 OBRÁZKY S ÚTRŽKY KÓDU

Obrázky sloužící k doplnění příslušného výukového textu. Kód byl v tomto případě vždy psán uvnitř vývojového prostředí Visual Studio Code a následně byl popsán pomocí komentářů přímo v jazyce JS (viz. Obrázek 4). Po vytvoření zápisu jsme opět využili nástroj

pro tvorbu snímků obrazovky. Takto vytvořený obrázek jsme přiložili k výukovému textu do příslušné studijní podkapitoly.

```
const automobily = ["Škoda","BMW","Audi","Toyota","Fiat"];

let vypis = automobily; // vypis = Škoda,BMW,Audi,Toyota,Fiat

vypis = automobily.toString(); // vypis = Škoda,BMW,Audi,Toyota,Fiat

vypis = automobily.join(" * "); // vypis = Škoda * BMW * Audi * Toyota * Fiat
```

Obrázek 12: Vytvořený prvek popisující metody sloužící pro výpis pole uvnitř JS (Zdroj: vlastní)

4.2.3 VIDEA

Pro tvorbu videí byl použit program pro vytvoření záznamu plochy a pro následnou úpravu byl použit program DaVinci Resolve. Videa vždy zachycují tvorbu kódu, kdy je tento proces doplněn po celou dobu audio komentářem. Všechna vytvořená a upravená videa byla nahrána na platformu YouTube a následně embedována uvnitř kurzu.

Ve studijních podkapitolách se vyskytují krátká videa, která slouží opět k doplnění výukového textu.

V závěru každé kapitoly se poté nachází video s ukázkou řešení příslušné seminární úlohy. Toto video jsme rozdělili vždy na menší části a ty od sebe rozdělili infografikou. Dále jsme pro tyto videa pomocí nástroje YouTube vytvořili časovou osu. Tím jsme se opět snažili dosáhnout zásady, aby byl celek rozdělen do co nejmenších částí. V těchto videích byl kladen důraz na vysvětlení a popis kritických míst seminárních úloh, které vyplynuly z jejich analýzy.

4.2.4 INTERAKTIVNÍ ČINNOSTI

Pro každou kapitolu jsme vytvořili sadu úkolů, které slouží k prohloubení získaných vědomostí a dovedností. V samotném závěru kapitoly se poté vždy nachází příslušná seminární úloha.

Pro vytvoření sady úkolů pro procvičení byla využita studijní činnost s názvem „*H5P*“. Taková sada úloh je vždy složena z otázek zkoumajících teoretické vědomosti studentů. Každá sada obsahuje otázky:

- Výběr správné možnosti,
- Výběr Ano/Ne,
- Výběr více správných možností.

V sadě se objevují ale také úlohy prohlubující praktické dovednosti. Připravili jsme takové příklady, kde studenti musí vytvořit syntakticky správný zápis, doplnit na příslušná místa správné části kódu nebo určit, zdali je kód syntakticky správně zapsaný.

Seminární úloha v závěru kapitoly je poté vložena pomocí studijní činnosti s názvem „*Úkol*“. Tato činnost bude umožňovat v budoucnu vyučujícímu automaticky kontrolovat odevzdání či neodevzdání úlohy. V takové části se vždy nachází písemný popis zadání úlohy. Aby měli budoucí studenti možnost lépe porozumět zadání a vyzkoušet výslednou funkcionalitu programu, pomocí HTML tagu iframe jsme na stránku vložili již vytvořený program ve vývojovém prostředí Codepen.io.

ZÁVĚR

Autor práce se seznámil s obsahem předmětu KVD/PGM1P a vytvořil pro něj multimediální prvky. První kapitola se v první části věnuje popisu předpokladů a podmínek, které jsou na studenty kladeny pro úspěšné absolvování předmětu. Druhá část je věnována popisu programovacího jazyku a vývojového prostředí využívaných při výuce předmětu. Shledáváme jako důležité se seznámit s veškerým obsahem předmětu KVD/PGM1P.

V kapitole věnované analýze seminárních úloh došlo k jejich podrobnému popisu. Byly zde také stanoveny často se vyskytující chyby při realizaci jednotlivých úloh během seminářů. Jako problematické vnímáme velké rozdíly ve stupních dovednostech jednotlivých studentů. Tento fakt může poté přispívat k široké škále chyb. Pro nás je seznámení se s úlohami stěžejním bodem pro následnou tvorbu multimediálních prvků. Ty budou popisovat realizaci jednotlivých úloh a budou upozorňovat na zjištěná kritická místa. Z toho důvodu vnímáme jako klíčové, důkladné seznámení se všemi seminárními úlohami.

Kapitola s názvem „Návrh vzdělávacího e-kurzu pro předmět KVD/PGM1P“ představuje zejména zásady a principy pro vznik moderního e-kurzu podle Maněny. Následná kapitola poté popisuje vznik a tvorbu námi vytvořeným multimediálních prvků, které jsou obsaženy v e-kurzu. Jako zásadní považujeme seznámení s těmito obecnými zásadami a principy, nimiž jsme se řídili při návrhu a realizaci našeho kurzu.

Kurz nebyl zatím nasazen a otestován v rámci výuky předmětu KVD/PGM1P. Z toho důvodu zatím není možné ohodnotit přínos námi vytvořeného e-kurzu, včetně jeho textového obsahu a multimediálních prvků. Můžeme předpokládat, že první zpětnou vazbu od studentů by bylo možné začít získávat až od akademického roku, v němž bude kurz uveden zařazen do výuky. To samé lze říct o získávání poznatků od vyučujících.

V budoucnu by bylo jistě vhodné kurz upravovat podle poznatků, které budou získány od vyučujících. Dále by bylo vhodné obsah kurzu upravovat na základě získané zpětné vazby ze strany studentů. Zde by bylo vhodné zjistit zejména přínos multimediálních prvků popisující realizaci seminárních úloh.

RESUMÉ

This bachelor thesis is called „Multimedia support for course KVD/PGM1P“. It is divided into four chapters. The first chapter deals with the description of the course. The second chapter examines and describes the set of tasks used in seminars. The third chapter describes principles which were used for the draft and creation of the multimedia support and e-course. The last chapter describes the creation of multimedia elements.

SEZNAM LITERATURY

[1] *Portál ZČU*. Online. Portál ZČU. C2007–2023. Dostupné z: https://portal.zcu.cz/portal/. [cit. 2023-11-15].

[2] *CourseWARE PGM1P*. Online. CoursWARE.ZČU.CZ. C2007–2023. Dostupné z: https://courseware.zcu.cz/portal/studium/courseware/kvd/pgm1p. [cit. 2023-11-16].

[3] FRANK, Filip a ZÍKA, Miroslav. Úlohy pro výuku JavaScriptu z předmětu Programování 1. Online. *MFI*. 2022, roč. 31, č. 4, s. 303–314. ISSN 1805-7705. Dostupné z: https://mfi.upol.cz/index.php/mfi/issue/view/49. [cit. 2024-04-17].

[4] HAVERBEKE, Marijn. *Eloquent JavaScript: a modern introduction to programming*. Third edition. San Francisco: No Starch Press, [2019]. ISBN 978-159-3279-509.

[5] FLANAGAN, David. *JavaScript: the definitive guide : master the world's most-used programming language*. Seventh edition. Beijing: O'Reilly, 2020. ISBN 978-1-491-95202-3.

[6] *High-level programming language*. Online. MDN Web Docs. C1998–2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/High-level_programming_language. [cit. 2023-11-28].

[7] GABBRIELLI, Maurizio a MARTINI, Simone. *Programming Languages: Principles and Paradigms*. Undergraduate Topics in Computer Science. London: Springer London, 2010. ISBN 978-1-84882-914-5.

[8] *Dynamic Languages*. Online. Javatpoint. C2011–2021. Dostupné z: https://www.javatpoint.com/post/dynamic-languages. [cit. 2023-11-28].

[9] *Dynamic typing*. Online. MDN Web Docs. C1998–2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/Dynamic_typing. [cit. 2023-11-28].

[10] *JavaScript*. Online. MDN Web Docs. C1998–2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [cit. 2023-11-28].

[11] *Interpreted vs Compiled languages*. Online. FreeCodeCamp. 2020. Dostupné z: https://www.freecodecamp.org/news/compiled-versus-interpreted-languages/. [cit. 2023-11-28].

[12] ZDRÁHAL, Jindřich. *Vývoj a rozdělení programovacích jazyků*. Online. OrgPad. Dostupné z: https://orgpad.com/o/Cv-Yo59zdAIbRmpIrRU4wG?token=APBNA_zvJPJYe0g4Y292V9. [cit. 2023-11-28].

[13] *Just in Time Compilation*. Online. FreeCodeCamp. 2020. Dostupné z: https://www.freecodecamp.org/news/just-in-time-compilation-explained/. [cit. 2023-11-28].

[14] *About CodePen*. Online. Codepen.io. C2023. Dostupné z: https://codepen.io/about. [cit. 2023-11-28].

[15] *Visual Studio Code FAQ*. Online. Visual Studio Code. C2023. Dostupné z: https://code.visualstudio.com/docs/supporting/faq#_what-is-the-difference-between-visual-studio-code-and-visual-studio-ide. [cit. 2023-11-28].

[16] *Programming Languages*. Online. Visual Studio Code. C2023. Dostupné z: https://code.visualstudio.com/docs/languages/overview. [cit. 2023-11-28].

[17] FRANK, Filip a ZÍKA, Miroslav. Úlohy pro výuku JavaScriptu z předmětu Programování 1 (2. díl). Online. *MFI*. 2023, roč. 32, č. 2, s. 141–151. ISSN 1805-7705. Dostupné z: https://mfi.upol.cz/index.php/mfi/issue/view/51. [cit. 2024-03-12].

[18] MANĚNA, Václav. *Moderně s Moodlem: jak využít e-learning ve svůj prospěch*. CZ.NIC. Praha: CZ.NIC, z.s.p.o., 2015. ISBN 978-80-905802-7-5.

[19] *Moodle History*. Online. Moodle.org. C2024. Dostupné z: https://docs.moodle.org/403/en/History. [cit. 2024-04-08].

SEZNAM OBRÁZKŮ, TABULEK, GRAFŮ A DIAGRAMŮ

PŘÍLOHY

V elektronických přílohách jsou obsaženy:

- Obrázky popisující syntaktický zápis.
- Obrázky s ukázkami kódu a komentáři.
- Videa popisující tvorbu seminárních úloh.
- Videa doplňující výukový text.
- Soubor s příponou .mbz obsahující zálohu hotového e-kurzu.