

Masarykova univerzita
FAKULTA INFORMATIKY



Systém pre sadzbu webových formulárov

Bakalárska práca

Miroslav Lehotský

Brno, jar 2016

Prehlásenie

Prehlasujem, že táto bakalárska práca je mojím pôvodným autorským dielom, ktoré som vypracoval samostatne. Všetky zdroje, pramene a literatúru, ktoré som pri vypracovaní používal alebo z nich čerpal, v práci riadne citujem s uvedením úplného odkazu na príslušný zdroj.

Miroslav Lehotský

Vedúci práce: RNDr. Pavel Minařík, PhD.

Podakovanie

Chcel by som poďakovať pánovi Mgr. Jaromírovi Vitekerovi za vypísanie témy ako aj za jeho čas a usmerňovanie pri tvorbe tejto práce.

Zhrnutie

Cieľom tejto bakalárskej práce bolo vytvoriť systém, ktorý uľahčí prácu pri vytváraní webových formulárov. Systém, ktorý je schopný manipulovať s už vytvorenými elementami a ich reálnymi veľkosťami v čase vykreslovania. Je schopný zoskupovať súvisiace formulárové elementy do skupín a následne ich aj vizuálne podporiť. Môže prijímať užívateľské preferencie na vykresľovanie týchto skupín ako aj obmedzenia na dostupný priestor v podobe maximálnej šírky a výšky. S prihliadnutím na tieto preferencie a obmedzenia sa snaží vyprodukovať vizuálne príjemnú sadzbu formulára.

Kľúčové slová

automatická sadzba, formulár, vykreslovanie, priestor, umiestňovanie, element

Obsah

Úvod	8
1 Vymedzenie pojmov automatická sadzba	9
1.1 Jednoduchý layout manager	9
1.2 Constraint Satisfaction	10
1.2.1 Typy obmedzení	10
1.2.2 Vyjadrenie obmedzení	12
1.2.3 Zisk obmedzení	13
1.2.4 Constraint Solver	14
1.2.5 Incosistency policies	15
1.3 Learning techniques	15
1.3.1 Marquise systém	15
1.3.2 ThingLab systém	15
1.3.3 Zhou výskum	16
2 Automatické generovanie webových formulárov	19
2.1 Automatické generovanie webových formulárov na základe spracovania prirodzeného jazyka	20
3 Grid Layout	23
3.1 CSS Grid Layout Module	23
3.1.1 Prispôsobenie voľnému priestoru	24
3.1.2 Nezávislosť na poradí v kóde	25
3.1.3 Definícia mriežky	26
3.1.4 Umiestňovanie položiek	27
3.1.5 Zarovnávanie položiek	27
3.2 jLayout	27
3.2.1 Border Layout	27
3.2.2 Grid Layout	28
3.2.3 Flex grid Layout	29
3.2.4 Flow Layout	30
4 Implementácia	31

4.1	Implementácia modulu FormLayoutManager	31
4.2	Načítanie elementov	31
4.3	Zoskupovanie elementov	32
4.4	Získ obmedzení.....	33
4.5	Vykresľovanie elementov.....	33
Záver.....		38

Úvod

V dnešnej technologicky vyspelej a informačnej dobe sa dennodenne na každom kroku stretávame s výmenou informácií prostredníctvom rôznych zariadení. Množstvo vymieňaných alebo zbieraných informácií nieje zanedbateľné a preto sa neustále hľadajú spôsoby ako tieto procesy urýchliť a uľahčiť.

Typickým príkladom hromadného zbierania informácií je formulár. Vo svete technológií a webových stránok to je webový formulár. Ten umožňuje zbierať rôzne typy informácií od ľudí. Každý takýto formulár je zväčša vytvorený nejakým človekom, ktorému to zaberie istý čas. Navyše formulár, aby sa v ňom ľahko orientovalo a interakcia užívateľa s ním bola čo možno najrýchlejšia a intuitívna, by mal spĺňať isté vizuálne a funkcionálne kritériá. O možnostiach automatizovania procesu tvorby webových formulárov pojednáva aj táto bakalárska práca.

Prvá kapitola práce vymedzuje pojem automatickej sadzby v počítačovom prostredí. Skúma metódy a už existujúce systémy a ich prístupy k riešeniu tejto problematiky. V tejto kapitole sú spomínané najmä systémy založené na tzv. spatial constraints, čo predstavuje obmedzenia a pravidlá kladené na vykresľovanie. Ako ďalšie sú v tejto kapitole spomenuté tzv. learning techniques. Sú to inteligentné systémy, ktoré sa dokážu učiť za pochodu. Nachádza sa tu stručný opis existujúcich systémov založených na tejto metóde ako aj princíp ich fungovania.

Druhá kapitola pojednáva o automatickom generovaní webových formulárov na základe spracovania písaného prirodzeného jazyka. Sú tu tiež uvedené príklady a stručný opis princípu fungovania takéhoto systému.

V tretej kapitole je spomenutý ďalší možný prístup k rýchlejšiemu a ľahšiemu vytváraniu webových formulárov. Konkrétne sa jedná o tzv. grid systém, ktorý je založený na delení priestoru do mriežky. Je tu konkrétne spomenutý pripravovaný CSS modul, ktorý by mal čoskoro byť oficiálne vypustený ku komerčnému použitiu.

Štvrtá kapitola je kapitola posledná a obsahuje popis vlastnej implementácie systému pre vytváranie webových formulárov. Ide o implementáciu v jazyku Javascript, ktorá sa stará o rozmiestňovanie predom vytvorených formulárových elementov v jazyku HTML. Vytvorený systém je schopný pracovať s rôznymi preferenciami a obmedzeniami získanými od užívateľa.

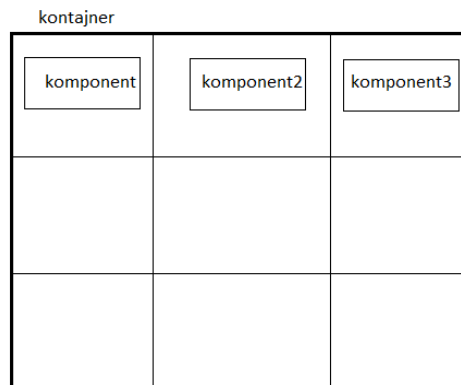
1 Vymedzenie pojmov automatická sadzba

Vo webovom prostredí pod slovným spojením automatická sadzba rozumieme automatický proces rozhodovania o veľkosti a pozícii vykreslovaných prvkov počítačovým programom [1]. Programy, schopné automatickej sadzby webových komponent naberajú na popularite každým dňom z dôvodu rastúceho množstva dát a neschopnosti ich manuálnej prezentácie.

V nasledujúcej kapitole budú postupne prebrané možné prístupy a metódy implementácie automatickej sadzby, ich výhody a nevýhody, stručne popísané ich fungovanie.

1.1 Jednoduchý layout manager

Predstavuje jednoduchý program často vkladaný ako súčasť UI toolkitov pre vytváranie formulárov pomocou klikania bez nutnosti písania kódu. Manager určuje veľkosť a pozíciu prvkov, ktoré spravuje na základe stanovených obmedzení a parametrov zadaných programátorom. Pre zjednodušenie používa kontajner, ktorý vymedzuje oblasť pre všetky vykreslované komponenty. Použitím kontajnera sa zabráni prípadnému presiahnutiu komponenty z požadovanej oblasti. Typické jednoduché layout manageri povolujú určovanie pozície len v horizontálnom alebo vertikálnom smere, v prípade použitia kontajnera v podobe mriežky určenie pozície komponenty konkrétnymi súradnicami a špecifikovanie maximálnej resp. minimálnej veľkosti komponenty. V prípade tejto metódy sa jedná o hierarchiu samostatne spravovaných kontajnerov (Obr. 1.1). Z dôvodu rozhodovania managera o vykreslovaní komponent za behu, je systém spoľahlivý pri rôznych veľkostiach obrazoviek.



Obr. 1.1: Rozdelenie kontajnera do mriežky

1.2 Constraint Satisfaction

Najviac skúmaná a najpoužívanejšia metóda v oblasti automatickej sadzby. Je založená na tom, že každá sadzba je reprezentovateľná množinou obmedzení a pravidiel, ktorá nieje predom pevne definovaná ale každý systém môže mať definovanú vlastnú množinu obmedzení. Obmedzenie(Constraint) je vzťah, ktorý musí byť dodržaný a je to užitočný nástroj pri budovaní grafických rozhraní.

Obmedzenie(Constraint) môže definovať rôzne vzťahy:

- Udržanie konzistentnosti medzi samotnými dátami a ich vykreslením
- Udržanie konzistentnosti medzi rôznymi zobrazeniami rovnakých dát
- Definíciu formátovania dát na obrazovke
- Špecifikovanie udalostí/animácií, ktoré sa vykonajú pri rôznych manipuláciách s dátami
- Určenie atribútov objektov animácií, napr. rýchlosť ...

O výhodách a nevýhodách tejto metódy jej zložitosti a formách, v ktorých môžu byť obmedzenia definované a o problémoch ich zisku a tvorby pojednávajú najbližšie podkapitoly.

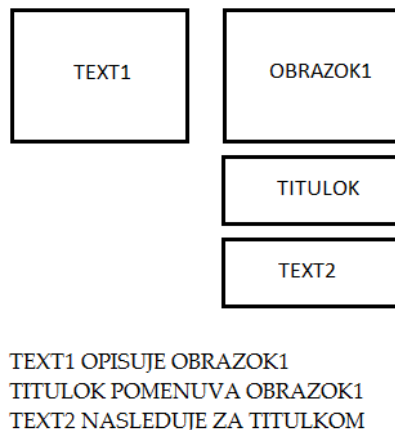
Typy obmedzení

Pri automatických vykreslovacích systémoch typu Constraint Satisfaction rozlišujeme dva typy obmedzení a to na základe toho, akým spôsobom opisujú pozíciu resp. rozmery komponenty.

Abstract Constraints

Definujú obmedzenia a vzťahy medzi vykreslovanými komponentami s vysokou úrovňou abstrakcie. Ich využitie je vhodné najmä v interaktívnych systémoch kde ako premenné vstupujú preferencie užívateľa, ktorý na ich definíciu nepotrebuje žiadne technické poznatky. Sú tiež vhodné pre formátovanie vizuálnych aspektov komponent ako veľkosť písma, ohraničenie komponenty...

Forma týchto obmedzení je typu: „*TEXT1 OPISUJE OBRAZOK1*“



Obr. 1.2: Opis sadzby pomocou abstract constraints.

Samotné abstraktné obmedzenia nedokážu jednoznačne určovať veľkosť alebo pozíciu vykreslovanej komponenty. Preto sú používané takmer vždy spoločne so Spatial Constraints, ktoré sú opísané v podkapitole nižšie. Ich spolupráca prebieha na základe funkcionality súčasti systému zvaného abstract-spatial translator, ktorý môže jedno abstraktné obmedzenie preložiť do ľubovlného počtu spatial obmedzení. Práve táto časť býva najzložitejšiou pri tvorbe automatického vykreslovacieho systému.

Spatial Constraints

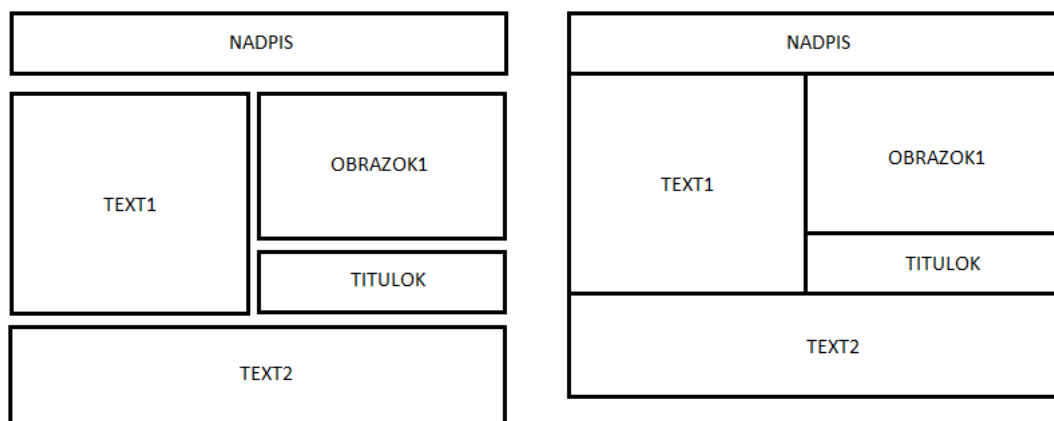
Vyjadrujú vzťahy medzi dvomi komponentami geometrickým spôsobom. „Predstavujú dôležitý abstraktný prostredný level medzi geometrickými vlastnosťami vykreslovaných komponent a ľudským vnímaním vzťahov medzi nimi.“[6] Jednoznačne definujú umiestnenie a veľkosť komponenty. Spôsob reprezentácie komponent, ktoré majú byť vykreslené zohráva pri spatial constraints úlohu, pretože môže spôsobovať obmedzenia ich definície.

Forma týchto obmedzení je typu: „*TEXT1 SA NACHÁDZA NAD OBRAZOK1*“



Obr. 1.3: Opis sadzby pomocou spatial constraints.

Spatial Constraints sú vo veľkom merítku využívané pri umiestňovaní popisov ku komponentám. Riešia problémy aby popis nebol príliš vzdialený ani príliš blízko od komponenty, ktorú opisuje a tiež aby k jednej komponente nebolo umiestnených viacero popisov. Sú tiež niekedy používané pre zvýšenie efektivity constraint solveru. Napríklad môže byť použité obmedzenie, ktoré povoľuje zmenu veľkosti komponent určitého typu len v jednom smere.



Obr. 1.4: Rozdiel medzi sadzbou určenou pomocou spatial constraints(vľavo) a sadzbou určenou pomocou abstract constraints(vpravo).

Vyjadrenie obmedzení

V tejto podkapitole sú opísané možné prístupy použité pre vyjadrenie obmedzení.

Formálna gramatika

Návrh formálnej gramatiky pre vyjadrenie obmedzení je efektívne riešenie, ale v prípade komplexnej formálnej gramatiky to môže byť veľmi zložité na implementáciu [1]. Výhodou použitia formálnej gramatiky je, že množina obmedzení nieje obmedzená a veľkosť množiny obmedzení pre systém závisí od konkrétnej implementácie.

Boolean predikáty

Ďalšou veľmi silnou metódou pre vyjadrenie obmedzení je metóda založená na boolean predikátoch. Zjednodušuje niektoré veci oproti použitiu formálnej gramatiky tým, že množina obmedzení nieje obmedzená len samotnou implementáciou, priestor vyjadrenia je limitovaný použiteľnými prvkami.

Zisk obmedzení

V tejto podkapitole sa rozoberajú možné prístupy zisku pravidiel a obmedzení pre vykresľovanie prvkov.

Interaktívna špecifikácia

Jedným zo spôsobov, ako získať pravidlá alebo obmedzenia pre sadzbu v použiteľnej forme je interaktívna špecifikácia. Jedná sa o metódu, pri ktorej je vyžadovaná interakcia s užívateľom. Systémy implementujúce tento prístup k zisku pravidiel môžu používať dva typy prístupov pri samotnej sadzbe. Jedným z nich je, že systém najprv vyžaduje užívateľov vstup pre špecifikáciu pravidiel, druhý prístup je, že systém najprv automaticky a samostatne vytvorí sadzbu, a následne povoľuje užívateľovi špecifikovať dodatočné pravidlá pre lepšie prispôsobenie umiestnenia prvkov. Nevýhodou interaktívnej špecifikácie je, že systémy, ktoré túto metódu používajú niesú plne automatizované.

Automatický zisk

Táto metóda je v súčasnej dobe ešte stále nedostatočne prebádaná, ale možnosť jej využitia je zrejmá z príbuzných oblastí. Jedná sa o plne automatizovaný zisk pravidiel a obmedzení potrebných pre sadzbu elementov. Jeden prístup je použiteľný najmä v súvislosti s prirodzeným jazykom a systémami, ktoré používajú automatické generovanie obsahu v prirodzenom jazyku. Pri použití tejto metódy v systéme s automatickým generovaním obsahu v prirodzenom jazyku sa jedná väčšinou o automatické generovanie abstraktných obmedzení.

Ďalším použiteľným prístupom je vytiahnutie abstraktných obmedzení zo vzťahov entít v SQL databáze. V rámci tejto metódy netreba generovať ani predávať dodatočné informácie ale pre získanie pravidiel a obmedzení sú použité dáta, pôvodne určené pre iné účely.

Constraint Solver

Jedna z najdôležitejších komponent v systéme automatickej sadzby založenej na metóde constraint satisfaction. Okrem toho, že systém je schopný získať pravidlá a obmedzenia, potrebuje ich vedieť aj zlúčiť a vyhovieť pokiaľ možno všetkým z nich, presne na to slúži constraint solver. V tejto podkapitole sú rozpísané možné prístupy k jeho implementácii.

Lokálne techniky

Technika je založená na tom, že riešením problému sa zaoberá zdola-hore. To znamená, že najprv sa snaží vyriešiť menšie lokálne problémy, až potom sa ich snaží skombinovať do väčších globálnych riešení. V rámci tejto metódy a constraint solveru sa rozlišujú dve možné implementácie.

Prvá z nich je rozdelená do dvoch nezávislých fáz. V prvej fáze systém vyrieši čo najväčšiu časť izolovaných lokálnych obmedzení, v druhej fáze sa snaží tieto lokálne riešenia skombinovať do finálneho riešenia.

Druhá, je založená na tom, že systém vyrieši najprv istú časť izolovaných obmedzení a následne iteratívne rieši obmedzenia medzi prvkami už vyriešenými a nevyriešenými.

Globálne techniky

Techniky, ktoré riešia problém zhora-dole. Nesnažia sa riešiť množinu problémov ich izolovaním a následným kombinovaním ale snažia sa celú množinu vyriešiť globálne. V rámci týchto techník sa používajú napr. iteratívne aproximačné metódy.

Z odborných štúdií v oblasti týchto popísaných techník usudzujeme, že lokálne techniky sú časovo a výpočtovo menej náročné ako globálne ale čelia problému, že kombináciou lokálnych riešení sa často nedostáva optimálnemu finálnemu riešeniu. Globálne techniky sú z pohľadu finálneho riešenia viacerých obmedzení lepšie, vyžadujú si ale viac výpočtových zdrojov a času.

Inconsistency policies

So stúpajúcou veľkosťou množiny obmedzení a pravidiel stúpa aj šanca na ich vzájomné konflikty. Preto pri návrhu systému treba počítať aj s takým prípadom a byť schopný vygenerovať sadzbu aj vtedy.

Jedným možným riešením tohto problému pri použití formálnej gramatiky pre špecifikáciu obmedzení je navrhnutie formálnej gramatiky tak, že v nej nemôžu vzniknúť cykly a tým pádom ani konflikty.

Ďalším možným riešením je pridávanie a dedenie priorít samotných obmedzení a pravidiel. Takým spôsobom pri konflikte dvoch pravidiel systém vyberie ten s vyššou prioritou. Pridanie priorít k pravidlám však nerieši problém konfliktov úplne, keďže dva rôzne pravidlá môžu mať rovnakú prioritu. V takom prípade sa používajú jednoduché techniky výberu ako: prvý vybraný alebo je použitá náhodná technika výberu.

1.3 Learning techniques

Systémy, ktoré sa dokážu učiť samé sú využiteľné aj v oblasti automatickej sadzby prvkov. V súčasnosti však zatiaľ každý systém na sadzbu, ktorý sa dokáže aspoň čiastočne sám učiť, sa učí na základe interakcie s užívateľom. Nejedná sa teda o systémy plne automatizované. V tejto podkapitole budú spomenuté existujúce systémy na sadzbu, ktoré spadajú do tejto kategórie.

Marquise systém

Interaktívny nástroj, ktorý umožňuje vytváranie nových grafických rozhraní pomocou demonštrácie, bez nutnosti programovania [2]. Okrem vytvárania a rozmiestňovania jednotlivých prvkov umožňuje tiež definovať chovanie prvkov pri manipulácii s nimi. Systém je možné spustiť v dvoch módoch.

Prvý mód je training mód, v ktorom užívateľ manipuluje s vykreslenými komponentami. Manipulácie sú systémom zaznamenané, v podobe vhodných dát, ktoré sú v prípade automatického návrhu grafického rozhrania použité pre optimálne rozmiestnenie prvkov, ktoré sa dá následne pozrieť v druhom móde Show. Ak systém nedokáže na základe zozbieraných dát z interakcie s užívateľom vygenerovať vhodné spatial constraints, užívateľ je vyzvaný pre ich ručné zadanie.

ThingLab systém

Podobný systému Marquise, ThingLab sa však líši prostredím, pretože v jeho prípade užívateľ interaguje s programom pomocou webového prehliadača. Je oproti systému Marquise aj komplexnejší, keďže disponuje obdobou tréningového módu systému Marquise aj pre animácie [4].

Zhou výskum

Tento výskum sa snažil prebádať možné prístupy v oblasti learning techniques pre automatické generovanie grafického obsahu. Proces automatického grafického generovania definuje ako proces mapovania množiny vstupov na vizuálny výstup. Tento proces rozdeľuje do viacerých úrovní, popísaných v tejto podkapitole.

Pre potreby učenia a získavania obmedzení(constraints) systémom samostatne, bol proces učenia rozdelený do troch kategórií:

- Information Learning Space
- Visual Learning Space
- Rule Learning Space

Information Learning Space

Pre mapovanie vstupných dát na vizuálny výstup je potrebné definovať návrhové pravidlá. Táto kategória určuje, že vstupné dáta musia obsahovať 3 typy informácií:

- Presentation data – samotné zobrazované dáta
- Presentation intents – ciele užívateľa, ktoré chce s dátami naplniť, napr. súhrn pacientových údajov
- Presentation context – určuje cieľovú skupinu, ktorá bude dáta zobrazovať

Uvedené informácie o vykresľovaných dátach nemusia byť vždy vopred známe alebo uvedené, v takom prípade sa o to stará learning technika. Tá dokáže napr. navrhnúť cieľ dát na základe objemu dát.

Visual Learning Space

Pre umožnenie zisťovania podobnosti vizuálnych objektov je potrebné aby bolo tieto objekty možné istým spôsobom merať. Vo výskume pre tento účel používajú euklidovskú vzdialenosť, v tomto prípade je každý vizuálny objekt vyjadrený množinou vlastností:

Podobnosť dvoch objektov(v_i, v_j) = $\sum_{k=1}^K w_k (f_i[k] - f_j[k])^2$

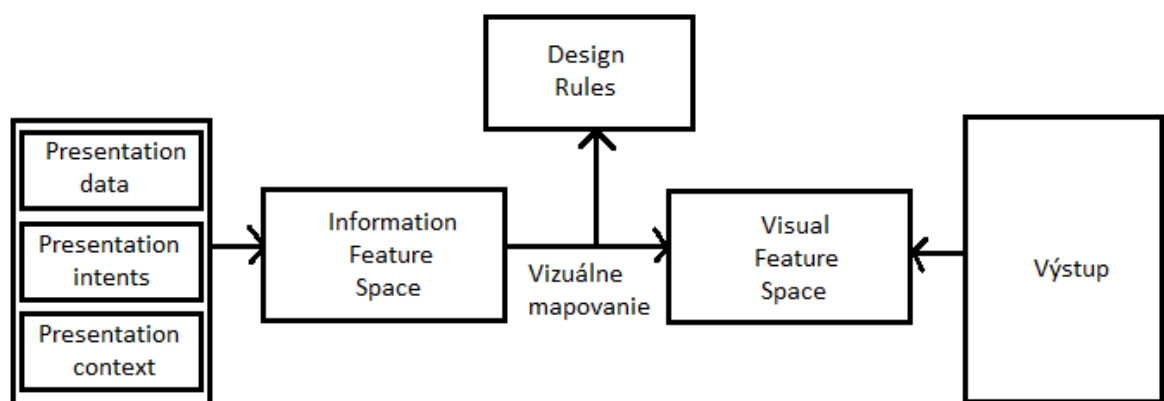
- K = celkový počet vlastností objektu
- $f_i[k]$ = k-tá vlastnosť objektu v_i
- w_k = váha $f[k]$

Práve pre proces rozhodnutia, ktoré vlastnosti je relevantné zohľadňovať je použitá technika learning. V prípade výskumu sa poskytla systému množina vizuálnych sadzieb s rôznymi úrovňami podobnosti objektov, z ktorých sa systém naučil o vzťahoch medzi rôznymi objektami.

Rule Learning Space

Proces learningu na tejto úrovni predstavuje učenie sa pravidiel vyššie úrovňových ako v prípade visual space learningu. Na tejto úrovni sa systém snaží naučiť ako zakódovať/specifikovať určitý typ dát. Stará sa o naučenie princípov použitých na mapovanie medzi systémovým vstupom a výstupom.

Systém opísaný vo výskume tiež používa Confidence Factor(CF), používaný pre ohodnotenie návrhových pravidiel, ktorý sa použije v prípade konfliktu.



Obr. 1.5: Rule Learning Space diagram [5].

Kvôli schopnosti porovnania rôznych dát a tvorby pravidiel musia byť dáta reprezentované uniformne. Pre takúto reprezentáciu sa v systéme používajú tzv. learning features, čo je množina deskriptorov pre dáta. Keďže rôzne dáta môžu vyžadovať rôzne množiny deskriptorov, zadefinovali sa v systéme dva priestory learning features pre dva základné typy dát:

- Information Feature Space
- Visual Feature Space

Information Feature Space

Priestor pozostávajúci z troch podpriestorov:

- Data Feature Space
- Intent Feature Space
- Context Feature Space

Všetky tieto podpriestory obsahujú vlastnosti dát špecifické pre daný priestor. Do množiny deskriptorov vstupuje ešte parameter data closeness, ktorý určuje sémantickú príbuznosť dát, na základe čoho môže systém navrhnúť zoskupenie dát.

Visual Feature Space

Slúži pre opis vlastností a funkcionalít vizuálnych objektov, podľa čoho sa tento priestor rozdeľuje na dva podpriestory:

- Visual Objects
- Visual Techniques

Visual Objects

Poskytuje opis vlastností vizuálnych objektov v dvoch podobách:

- Structural Features – typ štruktúry objektu
- Perceptual Features – vzťahy k iným objektom

Visual Techniques

Opisujú, akú rolu hrá daný vizuálny objekt v celom vizuálnom vykreslení.

Stratégie učenia(Learning Strategies)

V tejto podkapitole sú opísané štyry základné stratégie, používané pre účely samostatného učenia sa systému:

- Learning from examples
- Learning from observation
- Learning from instruction
- Learning by analogy

Learning from examples

Metóda, pri ktorej sa systém snaží naučiť pravidlá a obmedzenia z už existujúcich príkladov, ktoré sú mu predložené. Pre potreby naučenia je potrebné veľké množstvo príkladov. Každý existujúci príklad je vyjadrený pomocou dvoch vektorov:

Príklad_i = {f_i, t_i}

- f_i = (f_i[1], ..., f_i[K]), je množina dizajnových vlastností
- t_i = (t_i[1], ..., t_i[M]), je množina dizajnových akcií

Learning From Observation

Iný prístup k učeniu systému. Táto stratégia predstavuje spôsob, pri ktorom sa systém snaží sám vyvodzovať závery na základe všimania si charakteristík relevantného prostredia. Takáto metóda je výpočtovo náročná, preto boli vyvinuté viaceré zjednodušovacie techniky a algoritmy ako napr. clustering algoritmy.

Learning From Instruction

Spôsob, pri ktorom je možné rozširovať existujúce pravidlá inštrukciami od užívateľa. Často používaný je tzv. reinforcement learning spôsob, pri ktorom sú systémom naučené pravidlá opakovane upravované a prispôbované na základe spätnej väzby získanej od užívateľa. Užívateľ môže známkovať výsledky automatickej sadzby, na základe čoho systém aktualizuje prioritu jednotlivých pravidiel.

Learning by Analogy

Technika, v ktorej sa systém snaží vytvoriť nové pravidlá z už existujúcich pravidiel, ktoré opisujú situácie podobné, ako tie, pre ktoré chce definovať nové pravidlá.

2 Automatické generovanie webových formulárov

Formulár je veľmi užitočná súčasť webovej stránky, pretože funguje ako médium na prenos informácií medzi užívateľom a systémom. Webový formulár by môže byť interaktívny a jednoduchý na použitie, ak je navrhnutý v zmysle návrhových zvyklostí [3]. Použitie komponent vo formulári by malo priamo odrážať o zber akých informácií od užívateľa na danej stránke ide. Preto je málokedy možné použiť rovnaký formulár pre dve alebo viac stránok a je teda treba vytvárať formuláry osobitne pre každú stránku. Tvorba samotného formulára zaberie istý čas, najmä menej skúseným webdizajnérom, preto sa neustále hľadajú spôsoby, ktoré by

proces vytvárania webových formulárov dokázali zautomatizovať a šetriť tak čas vývojárov.

V tejto kapitole sú opísané možné prístupy k riešeniu automatického generovania webových formulárov.

2.1 Automatické generovanie webových formulárov na základe spracovania prirodzeného jazyka

Systémy určené na automatické generovanie webových formulárov s využitím užívateľského vstupu v podobe prirodzeného jazyka sú priamo určené pre urýchlenie a zautomatizovanie procesu tvorby formulára s prihliadnutím na užívateľsky definované špecifikácie. Celý proces prebieha v niekoľkých fázach:

- Prečítanie užívateľského vstupu
- Pochopenie užívateľského vstupu
- Vytiahnutie dôležitých informácií
- Vygenerovanie HTML kódu formulára
- Vytvorenie .html súboru s vygenerovaným HTML kódom

Prečítanie užívateľského vstupu

Táto fáza je jednoduchá, ide len o zisk užívateľsky definovaných dát prostredníctvom definovaného vstupu.

Pochopenie užívateľského vstupu

Užívateľský vstup definujúci pravidlá pre vykresľovanie a použitie komponent vo formulári je v podobe prirodzeného jazyka a teda vyzerá nasledovne:

„Vykresli text box s popisom Name s maximálnou veľkosťou 25.
Vykresli text box s popisom Priezvisko s maximálnou veľkosťou 25.
Vykresli dva radio buttony s menom Pohlavie a s hodnotami Muž a Žena.
Vykresli tlačítko pre odoslanie.“

Systém následne nad týmito dátami spustí definované algoritmy na parsovanie prirodzeného jazyka a kategorizuje jednotlivé slová napr. na slovesá, prídavné mená, podstatné mená ...

Vytiahnutie dôležitých informácií

V tejto fáze sa pokúša o vytiahnutie relevantných informácií a kľúčových slov, na základe ktorých identifikuje typy požadovaných komponent formulára (Obr. 2.1).

```
Input_1.label = "Name"
Input_1.name = "text1"
Input_1.type = "Text"
Input_1.value = ""
Input_1.maxlength = 25

Input_2.label = "Priezvisko"
Input_2.name = "text2"
Input_2.type = "Text"
Input_2.value = ""
Input_2.maxlength = 25

Input_3.label = "Pohlavie"
Input_3.name = "pohlavie1"
Input_3.type = "radio"
Input_3.value = "Muž"

Input_4.name = "pohlavie1"
Input_4.type = "radio"
Input_4.value = "Žena"

Input_5.type = "submit"
```

Obr 2.1: Podoba získaných relevantných informácií

Vygenerovanie HTML kódu formulára

Keď už má systém vytiahnuté kľúčové slová zo vstupu, čo vykonal v predchádzajúcej fáze, nieje problém vygenerovať samotný HTML kód formulára. Na vykresľovanie formulára systém používa tabuľky. Z uvedeného ukážkového užívateľského vstupu a následného vytiahnutia kľúčových slov, by mohol výsledný HTML kód formulára vyzeráť nasledovne:

```

<html>
<body>
<form action="" method="post">
<table>
<tr>
<td>
<label>Meno</label>
<input type="text" name="text1" value="" maxlength=25>
</td>
</tr>
<tr>
<td>
<label>Priezvisko</label>
<input type="text" name="text2" value="" maxlength=25>
</td>
</tr>
<tr>
<td>
<label>Pohlavie</label>
<input type="radio" name="pohlavie1" value="Muž">
<input type="radio" name="pohlavie1" value="Žena">
</td>
</tr>
<tr>
<td>
<input type="submit">
</td>
</tr>
</table>
</form>
</body>
</html>

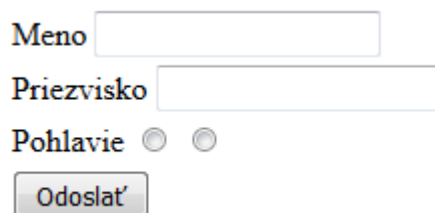
```

Obr. 2.2: Výsledný kód v značkovacom HTML jazyku

Vytvorenie .html súboru s vygenerovaným HTML kódom

Posledná fáza je veľmi jednoduchá, v nej systém len vytvorí nový .html dokument, do ktorého vloží celý vygenerovaný HTML kód a následne túto stránku poskytne ako výsledok.

Pri reálnom použití HTML kódu z tohto príkladu by samotný formulár vyzeral nasledovne:



Obr 2.3: Vizuálna prezentácia vyššie uvedeného HTML kódu

3 Grid Layout

Je založený na tom, že rozdeľuje priestor do mriežky. Mriežka je množina neviditeľných vertikálnych a horizontálnych čiar, ktoré sú použité pre umiestňovanie rôzneho obsahu na webovej stránke, časopise alebo novinách [9]. Rozdelením priestoru, na menšie podpriestory, ktoré vznikajú práve pretínaním vertikálnych a horizontálnych čiar, je značne uľahčené zarovnávanie a samotné umiestňovanie elementov v 2D priestore. Pri vykresľovaní sú elementy vykresľované tak, aby vyplňali celočíselný počet buniek mriežky [8].

V poslednom čase tento prístup k sadzbe elementov v oblasti webových stránok a ich designu nabera na popularite avšak jeho podpora v prehliadačoch je prakticky nulová. Vlastné implementácie snažiac sa napodobniť takýto systém za pomoci Javascriptu a detailného nastavovania pozícií jednotlivých elementov sú nedostatočne flexibilné pri zmene obmedzení kladených na priestor vykresľovania.

3.1 CSS Grid Layout Module

Jedná sa o implementáciu sady CSS tagov v prehliadačoch, ktoré uľahčia samotnú sadzbu elementov a ich zarovnanie.

Ponúka možnosť rozdelenia dostupného priestoru do riadkov a stĺpcov použitím vertikálnych a horizontálnych čiar. Užívateľ je následne schopný jednoducho identifikovať pozície a veľkosti prvkov čo mu pomáha pri samotnom umiestňovaní prvkov, ktoré sú umiestňované do tzv. grid areas, čo predstavuje istú oblasť mriežky.

Grid Container

Základným predpokladom pre vykresľovanie do mriežky pomocou tohto modulu je definovanie kontajneru, tzv. grid container. V rámci tohto kontajneru je priestor rozdelený do mriežky. Mriežka definuje oblasti, do ktorých je možné umiestňovať jednotlivé vykresľované elementy.

Grid Lines

Modul používa pre vytvorenie 2 typy čiar, horizontálne, ktoré sú rovnobežné s x-ovou súradnicovou osou a čiary vertikálne, ktoré sú rovnobežné s y-ovou súradnicovou osou. Každá z týchto čiar existuje na oboch koncoch riadkov a stĺpcov. Na každú z týchto čiar sa dá odkazovať pomocou číselného indexu alebo užívateľom špecifikovaným menom.

Grid Track

Jedná sa o priestor vymedzený dvomi paralelnými čiarami. Toto pomenovanie môže teda odkazovať na riadok alebo stĺpec. Každému tracku je priradená tzv. sizing funkcia, ktorá eviduje o koľko môže riadok resp. stĺpec narásť.

Grid Cell

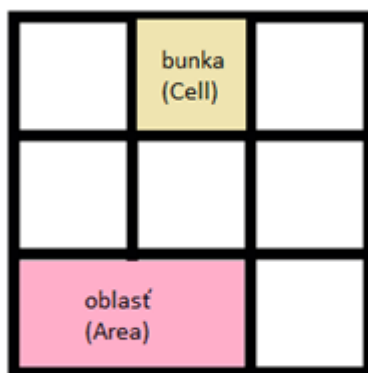
Je to bunka, vzniká prienikom riadka a stĺpca. Je to najmenšia adresovateľná jednotka v rámci mriežky, do ktorej je možné umiestniť element.

Grid Area

Je logický priestor pozostávajúci z jednej alebo viacerých susedných buniek, do ktorých je možné umiestniť element. Jej referencia je možná cez čiary, ktoré ju ohraničujú alebo priradeným menom, buď implicitným alebo explicitne zadaným užívateľom. Každý element, patriaci do rodičovského kontajnera mriežky je umiestnený v určitej oblasti.

Grid Item

Položka umiestnená v rodičovskom kontajnery, každý element, ktorý je pridany do mriežky v HTML kóde, je obalený do grid item. Výnimkou je text, obsahujúci len biele znaky, takýto textový element je ignorovaný a zaobchádza sa s ním ako by mal nastavenú „display: none“ vlastnosť.

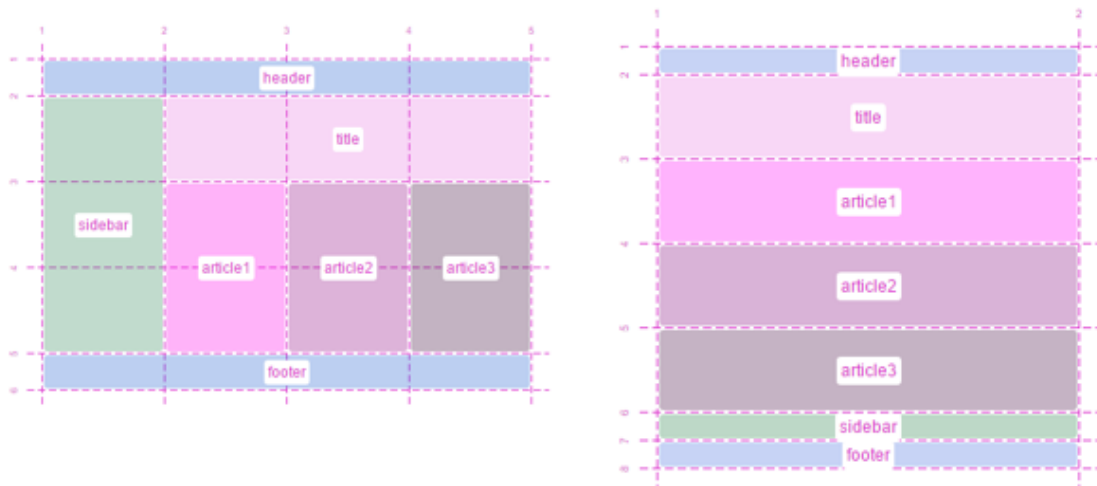


Obr. 3.1: Mriežka a jej súčasti

Prispôsobenie voľnému priestoru

CSS Grid Layout modul poskytuje možnosť definície a pomenovanie viacerých oblastí v rámci dostupného priestoru, pričom o samotné prispôbovanie týchto oblastí voľnému priestoru sa stará automaticky modul. Veľké uplatnenie tejto

funkcionality sa nájde určite pri zobrazovaní stránok na zariadeniach s rôznymi veľkosťami obrazoviek alebo v prípade mobilných zariadení pri zmene ich orientácie, tj. na šírku alebo výšku. Iným spôsobom ako docieľiť takýto efekt je použitie skriptovacieho jazyka s pracnou kontrolou mnohých pozícií pri zmenách veľkosti voľného priestoru.



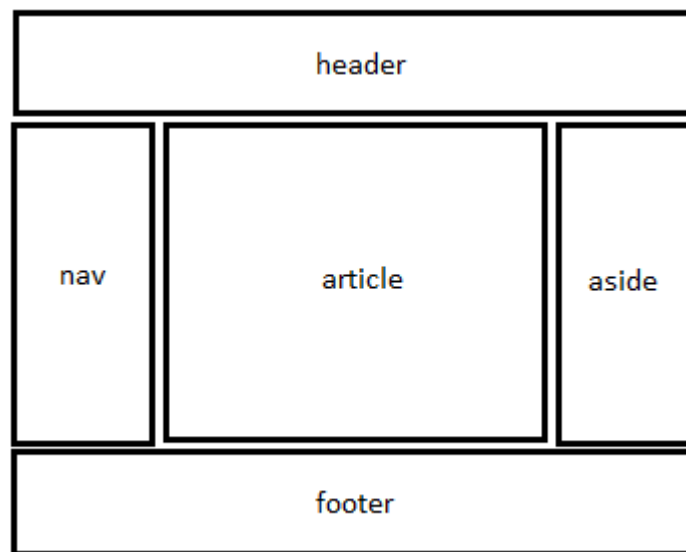
Obr. 3.2: Zmena vykreslenia v závislosti od zmeny orientácie obrazovky[9].

Nezávislosť na poradí v kóde

Mnohé web stránky vykresľujú svoj obsah lineárne, tj. vykresľujú elementy v poradí, v akom sa vyskytujú v samotnom kóde. CSS Grid Layout Module dokáže oddeliť vizuálnu a logickú časť elementov.

Používaná mriežka je celá definovaná v CSS, čo znamená že okrem rodičovského HTML elementu, ku ktorému je pripojená nieje potrebná žiadna dodatočná definícia riadkov alebo stĺpcov mriežky v HTML. Z tejto vlastnosti vyplýva nezávislosť medzi vizuálnym poradím vykreslenia elementov a ich poradím v kóde. Táto vlastnosť podporuje zas o niečo lepšiu separáciu vizuálnej manipulácie s elementom ako jeho umiestňovanie, zarovnávanie alebo zmenu veľkosti od logickej časti. Zmeny vykonávané s vykreslením elementu a jeho štylovaním sa neprejavujú nikde inde. Za pomoci tejto separácie je jednoduchšie udržiavať kód dobre štruktúrovaný a čitateľný.

Väčšina štruktúr webových stránok, čo sa týka logického usporiadania je rovnaká a to taká, že sa v kóde elementy vyskytujú v poradí v akom sú na stránke aj vykresľované (Obr. 3.3).



Obr. 3.3: Typická štruktúra webovej stránky [10].

Jej logická štruktúra v bežnom lineárnom usporiadaní a pomocou grid layoutu:

<pre> <header>...</header> <article>...</article> <nav>...</nav> <aside>...</aside> <footer>...</footer> </pre>	<pre> main { display: grid; grid: "h h h" "a b c" "f f f"; grid-template-columns: auto 1fr 20%; } article { grid-position: b; min-width: 12em; } nav { grid-position: a; /* auto min-width */ } aside { grid-position: c; min-width: 12em; } </pre>
---	---

Obr. 3.4: HTML kód typickej štruktúry stránky (vpravo), CSS kód tej istej štruktúry stránky za pomoci modulu Grid Layout Module []

Definícia mriežky

Definovanie mriežky vo vnútri kontajnera je možné určiť explicitne aj implicitne.

Explicitná mriežka

Mriežka sa zostavuje za pomoci troch atribútov:

- grid-template-rows

- grid-template-columns
- grid-template-areas

Tieto atribúty udávajú fixný počet riadkov a stĺpcov mriežky. Proporcie takto definovanej mriežky nemusia byť konečné, v prípade vloženia elementov do mriežky, ktoré budú presahovať jej hranice, bude mriežka automaticky zväčšená prostredníctvom zväčšenia riadkov a stĺpcov, prostredníctvom atribútov:

- grid-auto-rows
- grid-auto-columns

Implicitná mriežka

Vzniká automaticky v situácii, keď sú do kontajnera vložené elementy presahujúce hranice explicitne vytvorenej mriežky. Kontajner v takom prípade automaticky vygeneruje nové čiary mriežky v požadovanom smere, čím definuje nové grid tracks. Tieto novo-definované čiary spolu s explicitne vytvorenou mriežkou tvoria implicitnú mriežku.

Umiestňovanie položiek

Každá položka má oblasť v mriežke, ktorú okupuje, táto oblasť je množina buniek v mriežke. Táto oblasť definuje pre položku blok, ktorý má určené zarovnanie pre svoj obsah. Bunky, ktoré položka okupuje, priamo ovplyvňujú veľkosť riadkov a stĺpcov mriežky.

Zarovnávanie položiek

Po rozmiestnení položiek do oblastí môžu byť položky zarovnané. Pre zarovnanie v rámci oblasti mriežky sa implicitne používajú margin atribúty. Defaultne je v rámci mriežky nastavené zarovnanie položky na stretch, to znamená, že položka je natiahnutá aby vyplňala celú oblasť, v ktorej sa nachádza.

3.2 jLayout

Nástroj na rozmiestňovanie a zarovnávanie elementov v podobe Javascriptovej knižnice. Poskytuje celkovo štyri rôzne algoritmy a prístupy k rozmiestňovaniu elementov.

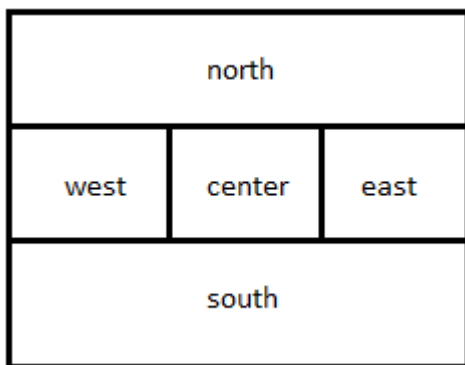
Border Layout

Prvý z algoritmov, ktoré `JLayout` poskytuje. V rámci tohto prístupu sa priestor, do ktorého sa vykresluje rozdelí na päť oblastí. Každý vykreslovaný element je umiestnený práve do jedného z tejto oblasti. Oblasti sú nazvané:

- center: vykreslený v strede daného kontajneru
- north: vykreslený nad strednou oblasťou
- south: vykreslený pod strednou oblasťou
- east: vykreslený naľavo od strednej oblasti
- west: vykreslený napravo od strednej oblasti

Vykreslenie každej tejto oblasti je voliteľné a každá oblasť je vykreslená len raz, algoritmu sa zadáva objekt, ktorý môže obsahovať nasledujúce vlastnosti:

- `hgap`: horizontálne odsadenie medzi dvomi komponentami
- `vgap`: vertikálne odsadenie medzi dvomi komponentami



Obr. 3.5: Rozdelenie priestoru do 5 oblastí [11].

```
var BorderLayout = JLayout.BorderLayout({
    west: myWestComponent,
    center: myCenterComponent,
    north: myNorthComponent,
    vgap: 5
});

borderLayout.layout(myContainer);
```

Obr. 3.6: Ukážka kódu border layoutu [11].

Grid Layout

Tento algoritmus vykresluje dané komponenty do mriežky. Každý vykreslovanej komponente nastaví rovnakú veľkosť. Rozmery mriežky môžu byť zadávané užívateľom. Algoritmus berie ako parameter objekt, ktorý môže obsahovať:

- `hgap`: horizontálne odsadenie medzi susednými bunkami
- `vgap`: vertikálne odsadenie medzi dvomi bunkami
- `rows`: počet riadkov mriežky
- `columns`: počet stĺpcov mriežky
- `items`: pole elementov, ktoré budú vykreslené
- `fill`: smer vykreslovania elementov, horizontálny alebo vertikálny

1	2
3	4

Ak je zadany len jeden z parametrov rows/columns ten druhý sa dopočíta za pomoci počtu elementov.

Ak nieje zadany ani jeden z parametrov rows/columns, počet stĺpcov sa nastaví na počet elementov a počet riadkov na 0.

Obr. 3.7: Rozdelenie priestoru do mriežky [11].

```
var gridLayout = jLayout.grid({
    rows: 2,
    columns: 2,
    items: [myComponent1, myComponent2, myComponent3, myComponent4]
});

gridLayout.layout(myContainer);
```

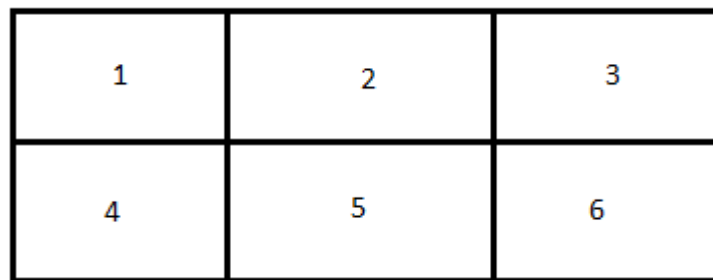
Obr. 3.8: Ukážka kódu grid layoutu [11].

Flex grid Layout

Vykreslí komponenty do mriežky s flexibilným počtom riadkov a stĺpcov, počet riadkov a stĺpcov môže byť zadany užívateľom. Predávaný objekt algoritmu, môže mať nasledujúce vlastnosti:

- hgap: horizontálne odsadenie medzi susednými bunkami
- vgap: vertikálne odsadenie medzi dvomi bunkami
- rows: počet riadkov mriežky
- columns: počet stĺpcov mriežky
- items: pole elementov, ktoré budú vykreslené

Rovnako ako v prípade algoritmu Grid, parametre rows a columns sú voliteľné a algoritmus dokáže vykresliť komponenty aj bez ich špecifikácie.



Obr. 3.9: Rozdelenie priestoru do flexibilnej mriežky [11].

```
var flexGridLayout = jLayout.flexGrid({
    rows: 2,
    columns: 2,
    items: [myComponent1, myComponent2, myComponent3,
            myComponent4, myComponent5, myComponent6]
});

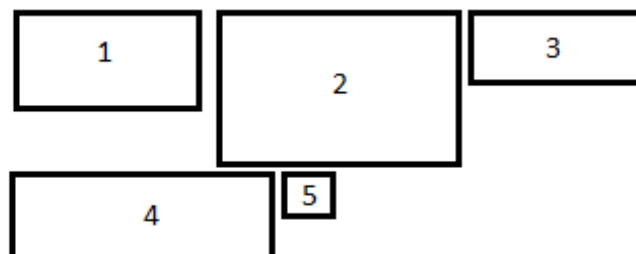
flexGridLayout.layout(myContainer);
```

Obr. 3.10: Ukážka kódu flexgrid layoutu [11].

Flow Layout

Komponenty sa vykresľujú na jeden riadok, ak sa už komponenta do riadku nezmestí, posunie sa na nový riadok. Objekt s parametrami môže vyzeráť:

- hgap: horizontálne odsadenie medzi susednými bunkami
- vgap: vertikálne odsadenie medzi dvomi bunkami
- alignment: zarovnanie elementu v riadku, 'left'/'center'/'right'
- items: pole elementov, ktoré budú vykreslené



Obr. 3.11: Vykresľovanie komponent do riadkov [11].

```
var flowLayout = jLayout.flow({
    alignment: 'center',
    items: [myComponent1, myComponent2, myComponent3,
            myComponent4, myComponent5]
});

flowLayout.layout(myContainer);
```

Obr. 3.12: Ukážka kódu flow layoutu [11].

4 Implementácia

V nasledujúcej kapitole je opísaná samotná implementácia systému pre automatickú sadzbu formulárových elementov.

4.1 Implementácia modulu FormLayoutManager

Systém je implementovaný v programovacom jazyku Javascript formou knižnice, resp. modulu. Celý kód programu sa nachádza v jednom súbore. Po importovaní tohto súboru v html súbore, je možné využívať funkcie manažéra.

V jazyku Javascript je so všetkým zaobchádzané ako s objektom, takže je to silno objektový jazyk. Čím sa však líši od mnohých iných objektových jazykov je, že v ňom neexistujú triedy a teda definícia objektu musí prebiehať inou formou ako definíciou triedy. Na definíciu objektu v Javascripte existuje viacero spôsobov. Implementovanie tohto systému je realizované formou jedného hlavného objektu, ktorý je definovaný tzv. formou object literals s menom FormLayoutManager. Jedná sa v podstate o jednu pomenovanú globálnu premennú, ktorá obsahuje ďalšie úrovne definovaných objektov a funkcií.

4.2 Načítanie elementov

Keďže hlavným cieľom tohto systému je uľahčenie a automatizovanie vykreslovania formulárových elementov, ako prvú vec, ktorá sa musí spraviť je načítanie elementov, ktoré sa majú vykreslovať.

Ide o funkciu s názvom addFormElements. Táto funkcia má jeden formálny parameter vo forme poľa. Pole musí obsahovať už predom vytvorené elementy,

ktoré sa budú následne vykreslovať pomocou manažéra. Poradie uvedenia elementov v predávanom poli sa prejaví pri vykresľovaní v prípade, že elementy nebudú pred vykreslením ďalej členené do skupín. Pre každý element v poli, reprezentovaný html kódom, ktorý sa predá funkcii `addFormElements` je vytvorená osobitná inštancia objektu elementu v manažérovi. Toto sa nevzťahuje na elementy typu `label`, ktoré slúžia ako popisok pre formulárový element. Popisy sú hneď pri načítaní asociované s formulárovými elementami, ku ktorým patria. Predanie toho istého formulárového elementu funkcii `addFormElements` viackrát nemá žiadny efekt, pretože tento scenár je funkciou kontrolovaný aj za pomoci funkcie `checkIfContains`.

4.3 Zoskupovanie elementov

“Grafika niekedy šetrí čas užívateľov potrebný pre vyhľadanie požadovaných informácií zoskupením súvisiacich informácií do jednej oblasti.” [7]

Jedna zo základných funkcionalít tohto systému je, že jednotlivé formulárové elementy je v rámci systému možné zoskupovať do skupín.

Pre vytvorenie novej skupiny slúži funkcia `addGroup`, ktorá má jeden formálny parameter, ktorým je meno skupiny. Následne je možné tejto skupine pridať elementy. Elementy nachádzajúce sa v jednej skupine sú podporené vizuálne. Pre každú skupinu je vytvorený osobitný `<fieldset>` tag, do ktorého sú následne vykresľované všetky elementy patriace do danej skupiny. Elementy sú teda istým spôsobom zviazané a pri ich vykreslení tvoria vizuálny celok. Navyše každej skupine je možné zadať preferenciu vo forme smeru vykresľovania v nej obsiahnutých elementov. Preferencia sa zadáva prostredníctvom funkcie `setPreferredOrientation`, ktorá berie jeden formálny parameter a to reťazec s kľúčovým slovom „horizontal“ alebo „vertical“. Ako ďalšia doplnková funkcionalita implementovaná v súvislosti so skupinami je funkcia `setShowName`, ktorá ako parameter berie pravdivostnú hodnotu „true“ a „false“, čím užívateľ môže špecifikovať, či chce aby bola skupina vykreslená aj s nadpisom obsahujúcim názov skupiny (Obr. 4.1).

The diagram illustrates two distinct groups of form elements, each enclosed in a light gray border. The first group, titled 'Login Details', contains four input fields: 'User Name', 'Password', 'Confirm Password', and 'First Name'. The second group, titled 'User Details', contains four input fields: 'Last Name', 'Gender' (a dropdown menu currently showing 'Female'), 'Email', and 'City'.

Obr. 4.1: Vykreslenie zoskupených elementov

4.4 Zisk obmedzení

Systému je tiež možné zadať obmedzenia pre vykresľovanie. Jedná sa o obmedzenia vo forme preferencií na maximálnu výšku a šírku formulára.

Obmedzenia je možné systému predať prostredníctvom na to, určených funkcií: `setMaxWidth` a `setMaxHeight`. Druhým možným spôsobom definovania obmedzení je prostredníctvom funkcie `setParent`, ktorá ako argument berie už existujúci a vykreslený kontajner. Ak kontajner obsahuje nastavené atribúty `max-width` a `max-height`, použijú sa pre načítanie obmedzení. Nastavením rodičovského kontajneru prostredníctvom funkcie `setParent` sa tiež určuje, kde bude formulár vykresľovaný. V prípade jeho určenia bude výsledný formulár vykreslený do tohto rodičovského formulára.

4.5 Vykresľovanie elementov

Samotný proces vykresľovania načítaných, prípadne aj zoskupených elementov s ohľadom na špecifikované obmedzenia a preferencie je komplexný problém. Systém sa snaží vyhovieť všetkým uvedeným preferenciám, často sú však v rozpore a niektoré sa musia uprednostniť na úkor tých druhých. Systém pracuje s už

vytvorenými elementami obsahujúcimi dáta, takže pracuje s reálnymi veľkosťami a nijak veľkosť elementov nemení pre potreby ľahšieho rozmiestnenia prvkov.

Celý proces vykreslovania je spustený bezparametrickou funkciou `layout`. Táto funkcia si postupne volá potrebné pomocné funkcie pre potreby procesu vykreslovania, ktoré si volajú pomocné funkcie pre špecifické im potrebné úkony. Pomocné funkcie si volá v poradí:

`getConstraints()`

Funkcia, ktorej jedinou úlohou je zisk obmedzení na rodičovský kontajner a vytvorenie kontajnera s menom „`FormLayoutManager_inner_div`“, do ktorého bude vykreslovať celý formulár.

`groupsLayout()`

Úlohou tejto funkcie je vykreslenie skupín a elementov patriacich do nich. V tejto fáze sa zohľadňuje prvá preferencia týkajúca sa skupín, a to konkrétne preferencia smeru vykreslenia elementov v rámci skupiny. Manažér sa snaží vykreslovať elementy podľa preferencie, v rámci tejto fázy vykreslovania elementov dochádza už aj k prihliadaniu na obmedzenie celkovej šírky formulára, ak bola zadaná. Ak je skupine zadaná preferencia vykreslovania v horizontálnom smere, manažér ju tak vykreslí a následne kontroluje celkovú šírku takto vykreslenej skupiny elementov (Obr. 4.2). V prípade, že takto vykreslená skupina presahuje maximálnu šírku, manažér sa ju pokúsi prekresliť vo vertikálnom smere (Obr. 4.3).



The image shows a web form with two rows of input fields. The first row contains 'User Name' and 'Password' labels followed by their respective input boxes. The second row contains 'Confirm Password' and 'First Name' labels followed by their respective input boxes. All labels are on the left and input boxes are on the right, indicating a horizontal layout.

Obr. 4.2: Vykreslenie prvej skupiny elementov v horizontálnom smere



The image shows the same web form as in Obr. 4.2, but the input boxes are stacked vertically. 'User Name' and 'Password' are in the first row, and 'Confirm Password' and 'First Name' are in the second row. This indicates a vertical layout where the horizontal space is insufficient for the horizontal arrangement.

Obr. 4.3: Vertikálne vykreslenie prvej skupiny po presiahnutí maximálnej šírky

othersLayout()

Táto funkcia sa stará o vykreslenie elementov, ktoré boli pridané do manažéra ale neboli priradené do žiadnej skupiny. Pre všetky tieto elementy sa vytvorí spoločná skupina s názvom „Others“ a následne sa vykreslí ako klasická skupina elementov.

leftAlignAll()

V rámci tejto a jej podporných funkciách sú všetky prvé elementy zľava, horizontálne zarovnané.

checkIfExceedHorizontally()

Funkcia vykonáva kontrolu presiahnutia maximálnej šírky niektorou z vykreslených skupín, ktoré mohlo vzniknúť v predchádzajúcej funkcii pri zarovnávaní elementov zľava. Ak došlo k presiahnutiu šírky v dôsledku zarovnania a smer vykreslenia elementov v tejto skupine je horizontálny, vykreslí skupinu vo vertikálnom smere.

checkIfExceedVertically()

Táto funkcia a jej pomocné funkcie sú obdobou predchádzajúcej, ale kontroluje presiahnutie maximálnej výšky. Navyše je táto funkcia omnoho komplexnejšia a zložitejšia, pretože sa už nesnaží o prekreslovanie vertikálnych skupín na horizontálne ale snaží sa nájsť nové miesta v pomyselnom druhom stĺpci, teda napravo od už vykreslených skupín. V prípade že sa podarí nájsť nové miesta pre dostatočný počet skupín, dokáže tým predísť presiahnutiu maximálnej výšky (Obr. 4.5).

rightAlignAll()

Dosiahnutím tejto funkcie v rámci procesu rozmiestňovania prvkov je sadzba formulára takmer pevne daná a nedochádza k žiadnym presunom skupín alebo elementov. V rámci tejto funkcie sa vykonáva zarovnávanie celých skupín zprava. Toto zarovnanie poskytuje estetické vylepšenie najmä v prípade použitia viditeľných okrajov jednotlivých fieldsetov.

Text input1	default value
Boolean input1	☒
Select input1	Second item
Boolean input2	☒
Select input2	Second item
Text input3	default value
Boolean input3	☒

Text input2	default value
Select input3	Second item
Boolean input5	☒

Boolean input4	☒
Select input4	Second item

Select input5	Second item
---------------	--

Obr. 4.4: Pôvodné vykreslenie skupín s elementami

Text input1	default value
Boolean input1	☒
Select input1	Second item
Boolean input2	☒
Select input2	Second item
Text input3	default value
Boolean input3	☒

Text input2	default value
Select input3	Second item
Boolean input5	☒

Boolean input4	☒
Select input4	Second item

Select input5	Second item
---------------	--

Obr. 4.5: Vykreslenie skupín s elementami po presiahnutí maximálnej výšky

4.6 Vyhodnotenie sadzby

Ako metriku pre umožnenie zhodnotenia úspešnosti vykreslenia formulára vzhľadom na získané obmedzenia a preferencie používa manažér objekt „metrics“. Tento objekt obsahuje zložky pre jednotlivé obmedzenia a preferencie. V priebehu procesu vykreslovania aktualizuje údaje v jednotlivých zložkách objektu. Pre preferencie smeru vykreslenia elementov v príslušnej skupine eviduje počet prípadov, kedy bolo vyhovené tejto preferencii. V prípade obmedzení vo forme maximálnej výšky a šírky formulára používa percentuálne vyjadrenie skutočnej veľkosti výsledného formulára vzhľadom na zadanú maximálnu výšku a šírku.

Záver

Cieľom práce bolo oboznámiť sa a preskúmať možné prístupy k riešeniu problematiky automatickej sadzby formulárových elementov a tiež vlastná implementácia systému tohto typu. V prvej časti práce je vymedzený pojem automatickej sadzby v prostredí počítačov a webových stránok. Sú tu spomenuté používané systémy a prístup ich riešenia na základe čoho som si bol schopný urobiť predstavu o automatickej sadzbe elementov potrebnú pre vytvorenie vlastnej implementácie.

Počas práce na vlastnej implementácii som sa naučil pracovať s v súčasnosti najrozšírenejšou Javascriptovou knižnicou jQuery. Tiež som si prehĺbil znalosti o značkovacom jazyku HTML a s ním úzko spojeným štýlovacím jazykom CSS.

Výsledkom práce je teda teoretická výskumná časť ako aj implementačná časť systému, ktorý môže byť použitý pre podporu automatickej sadzby webových formulárov.

Literatúra

- [1] S. Lok, S. Feiner. A Survey of Automated Layout Techniques for Information Presentations, Dept. of Computer Science: Columbia University, New York, 2001.
[<http://monet.cs.columbia.edu/publications.newer/layoutsurvey.pdf>]
- [2] B. A. Myers, R. G. McDaniel, D. S. Kosbie. Marquise: Creating complete user interfaces by demonstration. INTERCHI '93, Human Factors in Computing Systems, 1993.
[http://delivery.acm.org/10.1145/170000/169225/p293-myers.pdf?ip=147.251.230.94&id=169225&acc=ACTIVE%20SERVICE&key=D6C3EEB3AD96C931.BB3DE337C716FE30.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=618984699&CFTOKEN=90840118&acm_1463926228_2e53cf998b0087b7c1f18c1b63cf66da]
- [3] I. S. Bajwa, W. Aslam, S. I. Hyder. Speech Language Engineering System for Automatic Generation of Web based User Forms.
[<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.6864&rep=rep1&type=pdf>]
- [4] A. Borning, R. Duisberg: Constraint-based tools for building user interfaces. ACM Transactions on Graphics, 1986.
[http://delivery.acm.org/10.1145/30000/29354/p345-borning.pdf?ip=147.251.170.142&id=29354&acc=ACTIVE%20SERVICE&key=D6C3EEB3AD96C931.BB3DE337C716FE30.4D4702B0C3E38B35.4D4702B0C3E38B35&CFID=618911401&CFTOKEN=85729755&acm_1463910769_45283d9ab322ca48c9869843041b3b14]
- [5] M. X Zhou, S. Ma. Toward applying machine learning for to design rule acquisition for automated graphics generation. Technical report, IBM Watson Research Center, 1999.
[<https://www.aaai.org/Papers/Symposia/Spring/2000/SS-00-04/SS00-04-003.pdf>]
- [6] A. Borrmann, J. Hyvärinen, E. Rank. Spatial constraints in collaborative design processes. VTT – Technical Research Centre of Finland.
[http://www.andre-borrmann.de/docs/paper_ICE09.pdf]
- [7] S. M. Casner. A task-analytic approach to the automated design of graphic presentations. ACM Transactions on Graphics, 1991.
[http://vis.pku.edu.cn/course/Visualization_2013F/readings/Casner_TOG91_Task.pdf]

- [8] S. Feiner. A grid-based approach to automating display layout. In Proceedings of Graphics Interface '88, 1988.
[<http://graphicsinterface.org/wp-content/uploads/gi1988-26.pdf>]
- [9] P. Brosset. The future of layout with CSS: Grid Layouts, 2015.
[<https://hacks.mozilla.org/2015/09/the-future-of-layout-with-css-grid-layouts/>]
- [10] T. Atkins, E. J. Etemad, R. Atanassov. CSS Grid Layout Module Level 1, W3C, 2016.
[<https://drafts.csswg.org/css-grid-1/>]
- [11] B. Steain. jLayout – JavaScript Layout Algorithms, 2012.
[<https://github.com/bramstein/jlayout/blob/master/README.md>]