

Gymnázium, Praha 6, Arabská 14

Předmět Programování



MATURITNÍ PRÁCE

Knihovna pro záznam kotev v textu

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V dne

Petr Chalupa

ANOTACE

Práce se zabývá návrhem algoritmů, které by umožnili ukládání tzv. textových kotev (označení, poznámky aj.) do statického i dynamického textu (formátu XML) tak, aby je bylo možné opětovně do textu vložit i po jeho úpravě (a případně vyhodnotit chybu při vkládání). Takovýto program by pak měl být použitelný jako knihovna např. pro webové aplikace.

KLÍČOVÁ SLOVA

Algoritmus; textová kotva; XML; knihovna; webová aplikace

ABSTRACT

The thesis deals with the design of algorithms that would enable the storage of so-called text anchors (labels, notes etc.) in static and dynamic text (XML format) so that they can be re-inserted into the text even after its editing (and possibly evaluate the error during insertion). Such a program should then be usable as a library for e.g., web applications.

KEY WORDS

Algorithm; text anchor; XML; library; web application

OBSAH

1	ZADÁNÍ	4
2	ÚVOD	5
3	TERMINOLOGIE.....	6
3.1	STATICKÝ TEXT	6
3.2	DYNAMICKÝ TEXT	6
3.3	TEXTOVÁ KOTVA	6
3.3.1	Z POHLEDU PROGRAMU	6
4	ALGORITMUS VYTVOŘENÍ KOTVY.....	7
5	ALGORITMUS ULOŽENÍ KOTVY	8
6	ALGORITMUS VLOŽENÍ KOTVY	9
7	KNIHOVNA	10
7.1	ARCHITEKTURA	11
7.2	POUŽITÍ.....	11
8	DEMO	12
8.1	FUNKCE	12
8.2	GENEROVÁNÍ TEXTU	12
9	ZÁVĚR	13
10	POUŽITÉ ZDROJE.....	14
11	SEZNAM OBRÁZKŮ	15
12	SEZNAM UKÁZEK KÓDU	15

1 ZADÁNÍ

Téma: Knihovna pro záznam kotev v textu

Autor: Petr Chalupa

Vedoucí práce: Mgr. Jan Lána

Popis: Knihovna s algoritmy pro zapamatování vložených kotev (označení, poznámek aj.) do textu uživatelem. Cílem je, aby fungovala i pro dynamický text, tedy aby se kotvy automaticky přizpůsobovaly změnám v konkrétním textu (v rámci možností) a případně aby poskytla „zpětnou vazbu“ ohledně chyb - např. nepovedené zařazení do textu apod. Měla by fungovat na formátu XML (HTML) - použití primárně ve webových aplikacích, jako je například projekt Digitálního učebnicového systému, kterého jsem spoluautorem.

Platforma: JS/TS, Vue.js

2 ÚVOD

V dnešní době se mnoho textů přesouvá do digitální podoby. Mnoho z nich jsou původně fyzické texty, jako jsou například knihy. Další takové texty mohou být ty nově vznikající, jako jsou například články. Takové texty jsou zpravidla dostupné na internetu, a proto se nabízí možnost poskytnout jeho uživatelům užitečné nástroje pro manipulaci s nimi. Motiv této práce je tedy vytvořit takový nástroj, který umožní vkládat a manipulovat s textovými kotvami.

Tato práce se tedy zaměřuje na problematiku vkládání, ukládání a opětovného vkládání textových kotev do statického i dynamického textu ve formátu XML. Pomocí navrhnutých algoritmů může uživatel označit klíčové body v textu a používat je i po aktualizacích původního textu. V případě, že nastane po změně původního textu problém, uživatel by se o něm měl dozvědět přívětivou cestou.

Cílem této práce tedy je vytvořit knihovnu, která bude sloužit jako nástroj pro manipulaci s textovými kotvami, a která bude využitelná ve webových aplikacích. Implementace této knihovny poskytne uživatelům flexibilitu a efektivitu při práci s textem, přičemž bude zajišťovat nejen správnost manipulace s kotvami, ale i detekce a řešení chyb, které by mohly nastat v průběhu procesu.

3 TERMINOLOGIE

Pro tuto práci je nutné definovat několik základních pojmů, které jsou pro její správné chápání zásadní. Tyto pojmy se týkají typů textu, nad kterými algoritmy knihovny operují, a také co znamená textová kotva jako taková.

3.1 STATICKÝ TEXT

Statický text je chápán jako řetězec znaků (jeho délka není relevantní), který se v průběhu času nemění. Takový text se dá v ideálním případě rozdělit na odstavce, věty a případně slova a znaky. Pracovat s takovým textem tedy lze předvídatelně.

3.2 DYNAMICKÝ TEXT

Dynamický text se odlišuje od statického tím, že se v průběhu času mění, což může vést ke ztížení práce s ním a ke zkomplikování operací na něm prováděných – v extrémních případech až k jejich úplnému selhání. Operace prováděné na dynamickém textu jsou funkční i na statickém textu.

3.3 TEXTOVÁ KOTVA

Textová kotva je pojem, který označuje specifický bod v textu, který je definován svojí pozicí (cesta/souřadnice apod.), a který je v ideálním případě nehybný. Kotva je charakteristická zejména svojí odlišností od textu, pokud rozšíříme její definici na právě dva sousední body, označující začátek a konec kotvy, protože poté začne mít význam i její vizuální charakteristika – např. zabarvené pozadí. Z toho vyplývá například použití pro označování částí textu, což je hlavní motivace této práce.

3.3.1 Z POHLEDU PROGRAMU

Pro algoritmické operace pozbývá význam vizuální reprezentace kotvy, ale je zcela nutné, aby byla každá kotva unikátní např. díky UUID¹. Kotvy jsou zároveň chápány jako nejmenší možné celky, které jsou pospojované do jednolitého bloku, který může pak kotvu reprezentovat vizuálně jako celek. Důvodem pro toto rozlišení na kotvy (Anchor) a bloky kotev (AnchorBlock) je to, že ve formátu XML se každá kotva vkládá do páru tagů, který ohraničuje např. odstavec textu, ovšem označený text může přesahovat přes více než jeden takový úsek textu.

¹ UUID – Universally Unique Identifier ~ Univerzálně Unikátní Identifikátor

4 ALGORITMUS VYTVOŘENÍ KOTVY

K vytvoření kotvy je nejdříve potřeba impulsu od uživatele. Takový impuls musí následovat po označení libovolné oblasti textu – horizontálně i vertikálně. Vše je situováno tak, aby bylo možné tvořit kotvy zvlášť pro každý jeden definovaný blok textu (rootNode). Pokud výběr přesahuje za hranice tohoto bloku, je selekce nesprávná a algoritmus končí.

V případě, že výběr v textu je validní, je možné pokusit se o vytvoření kotvy. Algoritmus zpracovává jednotlivé části výběru (objektu Selection) – tzv. Range. Nejprve je nutné získat nejmenší možný blok, který obsahuje celý Range (tzv. commonAncestorContainer) – to znamená nejbližšího společného předka počátečního a koncového bloku. Zbytek práce může probíhat už pouze v tomto bloku. Z něho jsou dále vybrány všechny jeho potomci – tzv. Node, které jsou textové (TEXT_NODE) a zároveň jsou součástí označení. Ty jsou vkládány do pole v pořadí jejich výskytu a následně jsou vyřazeny (nahrazeny null) ty, které jsou již součástí nějakého Anchoru, aby se zabránilo vzniku překryvu. V cestě k němu skrze DOM² nesmí tedy existovat žádný další Anchor.

Pro každý souvislý úsek těchto Nodů je pak vytvořen AnchorBlock, který kromě referencí na jednotlivé menší části nese i další přídavné informace, které jsou pro celou kotvu společné (barva a data). Pro první Anchor prvního AnchorBloku je použit startOffset z Range a pro poslední Anchor posledního AnchorBlocku je použit endOffset z Range. Zbytek pokrývá všechn doposud nepokrytý text mezi nimi. Nakonec AnchorBlock spojí interně všechny své Anchory (leftJoin, rightJoin). Tím algoritmus končí.

Přidat diagram!!!

² DOM – Document Object Model ~ Objektový Model Dokumentu

5 ALGORITMUS ULOŽENÍ KOTVY

Něco málo k serializaci

6 ALGORITMUS VLOŽENÍ KOTVY

Pár checků a pak to samé jako vytvoření

7 SPECIÁLNÍ FUNKCE

V knihovně se vyskytují některé funkce, které mají velmi specifické využití a zaslouží si větší pozornost. Ty hlavní jsou v této kapitole popsány a podrobně vysvětlené.

xPath...

7.1 FUNKCE INVERTHEXCOLOR()

Tato funkce slouží k získání kontrastově obrácené barvy k zadané barvě. Funkce má jako argument barvu v HEX formátu a stejně tak vrací barvu v HEX formátu. Jako kontrastově obrácená barva se v tomto případě myslí buď černá (#000000), nebo bílá (#ffffff), jelikož funkce je využita při přebarvování Anchoru tak, aby byla barva textu vždy dostatečně kontrastivní s barvou jeho pozadí. Funkce podporuje argument v 3 nebo 6 znakovém zápisu i s možností vynechat „#“. Nejdříve je u každé barvy zajištěn 6 znakový zápis – tzn. převod v případě potřeby a poté její číselné rozložení na jednotlivé prvky R, G a B.

Nakonec...

<https://stackoverflow.com/questions/3942878/how-to-decide-font-color-in-white-or-black-depending-on-background-color/3943023#3943023>

8 KNIHOVNA

Celý projekt je koncipován jako knihovna pro použití ve webovém prostředí; přesněji přímo ve webových aplikacích. Od toho se také odvíjí architektura projektu a styl jeho vývoje. Důraz byl například kladen velmi na omezení využívání dalších knihoven (dependencies). Celá knihovna je pak dostupná v registru npm³ pod názvem `dynamic-text-anchors`. Díky npm je jednoduché publikovat nové verze knihovny přímo z GitHub repozitáře nebo knihovnu jednoduše sémanticky verzovat (`major.minor.patch`). Npm ostatním vývojářům poskytuje přehledné informace o knihovně, jako je například odkaz na demo nebo README projektu.

8.1 ARCHITEKTURA

8.2 POUŽITÍ

Celý proces od instalace po užití je objasněn v README. Po instalaci je potřeba knihovnu pouze importovat:

```
`import DTA from "dynamic-text-anchors";`
```

Poté je nutné vytvořit objekt DTA, do jehož konstruktoru se vkládá element, v němž má být možné operovat s kotvami:

```
`const dta = new DTA(rootElement);`
```

Následně je již možné používat všechny veřejné metody knihovny.

³ npm, Inc. – správce JS balíčků (knihoven) pro Node.JS

9 DEMO

9.1 FUNKCE

9.2 GENEROVÁNÍ TEXTU

10 ZÁVĚR

11 POUŽITÉ ZDROJE

Aktuální dokument neobsahuje žádné prameny.

12 SEZNAM OBRÁZKŮ

Nenalezena položka seznamu obrázků.

13 SEZNAM UKÁZEK KÓDU

Nenalezena položka seznamu obrázků.