Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Miha Gašperšič

Razvoj orodja za napredno upravljanje vsebin spletne strani

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00458/2013
Datum: 12.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA GAŠPERŠIČ**

Naslov: **RAZVOJ ORODJA ZA NAPREDNO UPRAVLJANJE VSEBIN SPLETNE STRANI**

DEVELOPMENT OF A TOOL FOR AN ADVANCED WEB PAGE CONETENT MANAGEMENT

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi predstavite sistematičen razvoj orodja za upravljanje vsebin spletne strani, ki omogoča učinkovito izdelavo le-te brez podrobnega poznavanja spletnih tehnologij. Orodje naj izkorišča danosti jezika HTML5, usmerjeno pa naj bo predvsem v učinkovito oblikovanje grafične podobe spletnih strani. Naloga naj predstavi sistematičen razvoj orodja od ideje do uporabe končnega izdelka.

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Miha Gašperšič, z vpisno številko **63080253**, sem avtor diplomskega dela z naslovom:

Razvoj orodja za napredno upravljanje vsebin spletne strani

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 13. septembra 2013 Podpis avtorja:

Iskreno se zahvaljujem mentorju viš. pred. dr. Igorju Rožancu za pomoč in nasvete pri izdelavi diplomskega dela. Zahvaljujem se tudi Tanji H., ki je lektorirala moje diplomsko delo. Zahvala gre tudi družini, sošolcem in prijateljem, ki so me tekom študija podpirali in mi stali ob strani ter pripomogli k uspešnemu zaključku mojega študija.

Kazalo

Seznam uporabljenih kratic in simbolov

AJAX	angl. *Asynchronous JavaScript and XML* – asinhroni JavaScript in XML
API	angl. *Application Programming Interface* – vmesnik za programiranje aplikacij
CMS	angl. *Content Management System* – sistem za upravljanje vsebin
Cookies	angl. *Cookie* – piškotek
CSS	angl. *Cascading Style Sheets* – kaskadna slogovna polja
DOM	angl. *Document Object Model* – objektni model dokumenta
Event	angl. *Event* – dogodek
HTML	angl. *Hyper Text Markup Language* – jezik za označevanje besedila
JSON	angl. *JavaScript Object Notation* – notacija objektov JavaScript
Local Storage	angl. *Local Storage* – lokalna shramba podatkov
MIT License	angl. *Massachusetts Institute of Technology License* – dovoljenje Inštituta za tehnologijo Massachusetts

MPL	angl. *Mozilla Public License* – javno dovoljenje Mozilla
PDF	angl. *Portable Document Format* – odprt standard za izmenjavo elektronskih dokumentov
SGML	angl. *Standard Generalized Markup Language* – sestav namenjen za pripravo zvrsti spisov
SVG	angl. *Scalable Vector Graphics* – umerljiva vektorska grafika
W3C	angl. *World Wide Web Consortium* – mednarodni inštitut, kjer člani organizacije in javnost sodelujejo ter skupaj razvijajo standarde za splet
WCMS	angl. *Web Content Management System* – spletni sistem za upravljanje vsebin
XHTML	angl. *Extensible HyperText Markup Language* – označevalni jezik HTML, usklajen s sintakso XML
XML	angl. *Extensible Markup Language* – razširljiv označevalni jezik
XSLT	angl. *Extensible Stylesheet Language Transformations* – jezikovne pretvorbe razširljivih slogovnih polj
XSS	angl. *Cross-site scripting* – vrsta računalniške ranljivosti, ki jo navadno najdemo v spletnih aplikacijah
XUL	angl. *XML User Interface Language* – označevalni jezik, ki so ga razvili pri Mozilli, namenjen pa je preprosti in hitri izdelavi grafičnih uporabniških vmesnikov

Povzetek

Cilj diplomskega dela je predstavitev primera razvoja orodja za grafično upravljanje s spletnimi stranmi WebsiteEdit, ki bo uporabniku omogočal napredno urejanje in izdelavo spletne strani. Prvi del diplomskega dela je namenjen predstavitvi tehnologij in orodij, ki se uporabljajo pri razvoju modernih spletnih aplikacij ter so bila uporabljena pri razvoju orodja WebsiteEdit. Predstavitev tehnologij se osredotoča predvsem na teoretični opis tehnologij, kot so označevalni jezik HTML5, kaskadna slogovna polja, skriptni jezik JavaScript, ter uporabo namenskih knjižnic jQuery, jQuery-UI ter BlobBuilder in FileSaver. Diplomsko delo obravnava tudi uporabna orodja ter vstavke, ki se uporabljajo pri razvoju aplikacij. V prvem delu opisuje tudi implementacijo in uporabo lokalne shrambe podatkov. Predstavi sisteme za upravljanje vsebin ter opiše njihove prednosti in slabosti.

Drugi del diplomskega dela vsebuje predstavitev razvoja orodja za grafično upravljanje s spletnimi stranmi WebsiteEdit. Najprej je opisana ideja, nato pa sledi podroben opis analize in specifikacije zahtev. Diplomsko delo nadaljuje z opisom načrta ter implementacije orodja WebsiteEdit. Ob koncu se dotakne načina testiranja orodja WebsiteEdit ter opredeli načrt izboljšave, uporabnost in nadaljnji razvoj. Rezultat praktičnega dela diplomske naloge je orodje za grafično upravljanje s spletnimi stranmi WebsiteEdit, ki omogoča enostavno izdelavo atraktivne multimedijske spletne strani.

Ključne besede: spletne tehnologije, spletni brskalnik, spletna aplikacija, WebsiteEdit.

Abstract

The aim of the thesis is to present an example of WebsiteEdit tool development which will allow users advanced editing and developing of a website. The first part of the thesis is intended to present the technologies and tools that are used in the development of modern web applications and have also been used in the WebsiteEdit tool development. While describing the technologies we focused on the theoretical description of technologies such as HTML5 markup language, Cascading Style Sheets, JavaScript scripting language, and the use of libraries such as jQuery, jQuery-UI, BlobBuilder and FileSaver. The thesis also touches upon useful tools and plugins that are used when developing applications. In the first part, the thesis describes the implementation and the use of Local Storage. It also describes the Content Management Systems, their advantages and disadvantages.

The second part of the thesis presents the WebsiteEdit tool development. Firstly, it describes the idea of a WebsiteEdit tool followed by a detailed description of the analysis and specification requirements. The thesis continues with a description of the WebsiteEdit tool plan and its implementation. Finally, the thesis touches upon the method of testing of the tool WebsiteEdit and specifies a plan of improvement, the tool's usability and further development. The result of the practical part of the thesis is a WebsiteEdit graphical tool for managing websites, which makes it easy to produce attractive multimedia websites.

Keywords: web technologies, web browser, web application, WebsiteEdit.

Poglavje 1

Uvod

Uporaba svetovnega spleta je postala vsakodnevno opravilo velikega števila ljudi. Razvoj je prinesel ogromno novih in naprednih funkcionalnosti, ki vsakemu uporabniku svetovnega spleta ponujajo možnost uporabe najrazličnejših storitev tako na nivoju namiznih kot tudi spletnih aplikacij. Svetovni splet predstavlja eno bolj razširjenih oblik pretoka podatkov in informacij, pri kateri sodelujejo uporabniki svetovnega spleta.

Splet, kot ga poznamo danes, je bistveno bolj razvit in dograjen, kot je bil v preteklosti, saj končnemu uporabniku ponuja enostavno in hitro interpretacijo podatkov. Najnovejše tehnologije ponujajo uporabniku prikaz najrazličnejših multimedijskih vsebin, podatkov in informacij, ki kot celota predstavljajo vizualni format zapisa informacij ter ponudijo uporabniku celovit pregled nad pomembnimi podatki.

Vzporedno z razširjenostjo svetovnega spleta so se širile tudi potrebe po izdelavi namenskih orodij in aplikacij, ki bi uporabniku omogočale enostavno, hitro in predvsem učinkovito uporabo svetovnega spleta. Namen svetovnega spleta je vsakemu uporabniku omogočiti kreativno predstavitev podatkov, interpretacijo ter deljenje lastnih idej, znanja in informacij ter njihovo predstavitev na spletu. Veliko podjetij širom po svetu ponuja brezplačne spletne rešitve, ki pripomorejo k preprostemu širjenju podatkov in informacij na svetovnem spletu.

Ob vsakodnevni uporabi svetovnega spleta smo prišli do spoznanja, kako pomembno in koristno je uporabljati obstoječe rešitve, ki uporabniku omogočajo polno funkcionalno delo s svetovnim spletom. V sklopu diplomskega dela smo se odločili, da bomo razvili orodje, ki bo uporabniku ponudilo možnost grafičnega upravljanja s spletnimi stranmi. Kot poznavalci svetovnega spleta smo seznanjeni s podobnimi rešitvami sistemov, ki uporabniku poenostavijo izdelavo modernih spletnih strani. Ob uporabi omenjenih rešitev morajo biti uporabniki vsaj deloma ozaveščeni o načinu delovanja svetovnega spleta, okvirno morajo prepoznati namen tehnologij za predstavitev podatkov ter prednosti, ki nam jih ponuja svetovni splet.

V ta namen bomo izdelali orodje za grafično upravljanje s spletnimi stranmi, ki bo implementirano v najnovejših tehnologijah ter bo uporabniku ponudilo posebno izkušnjo razvoja spletnih strani. Orodje bo ponujalo možnost izdelave napredne in atraktivne multimedijske spletne strani z uporabo spletnega brskalnika. Uporabnik bo imel na voljo funkcionalnosti, s katerimi si bo lahko popolnoma prilagodil grafični izgled urejane spletne strani ter imel možnost vključevanja multimedijske vsebine, kot so slike in videi. Od implementacij ostalih podobnih rešitev se bo naš sistem razlikoval po enostavnosti namestitve, preprostosti uporabe, odličnem grafičnem izgledu ter minimalnem pretoku podatkov med spletnim strežnikom in odjemalcem.

Poglavje 2

Orodja in tehnologije

2.1 Spletne tehnologije

S prvimi podanimi koncepti svetovnega spleta so se začele vzporedno z njim razvijati tehnologije in spletne aplikacije, ki skrbijo, da se uporabniku prikažejo zahtevane informacije ali podatki v ustrezno formatirani obliki.

Razvila se je prva različica svetovnega spleta, imenovana "Web 1.0" [2]. Vsebina prvega zasnovanega koncepta predstavitve podatkov na svetovnem spletu je zajemala enostaven prikaz podatkov zahtevane spletne strani uporabnika. Bistvena značilnost prve različice koncepta svetovnega spleta je bila predstavitev statičnih podatkov. Spletne strani so bile vnaprej definirane in se niso dinamično prilagajale uporabniški izkušnji. V grafičnem smislu je bila implementirana podpora tabel, s katerimi je lahko uporabnik poljubno pozicioniral elemente po površini spletne strani.

Predstavitev podatkov na svetovnem spletu je kmalu postala senzacija in je motivirala številne uporabnike in razvijalce, ki so imeli v tistem času možnost sodelovanja pri širjenju in objavljanju najrazličnejših podatkov in informacij, s katerimi so si izdelali svojo predstavitveno spletno stran. Tako je bila v letu 2004 zasnovana nova različica koncepta svetovnega spleta [2]. Nov koncept je bil nadgradnja obstoječega z dodanimi novimi funkcionalnostmi, ki so uporabniku ponudile ogromno novih možnosti predstavitve podatkov

na svetovnem spletu. Novost je bila funkcionalnost na strani odjemalca. Dodana je bila zmožnost uporabe skriptnega jezika za dinamično in interaktivno prikazovanje spletne strani. Prav tako je prišlo do razvoja na področju asinhronega JavaScripta z implementacijo označevalnega jezika XML, poimenovanega AJAX. Nova tehnologija je omogočala osveževanje podatkov na spletni strani brez posebne zahteve uporabnika.

Po zasnovi koncepta se je začel razvoj številnih knjižnic in dodatkov, ki so omogočile uporabniku enostavno implementacijo in prikaz podatkov na spletni strani. Izdelane so bile knjižnice, kot so jQuery, MooTools, Dojo Toolkit, YUI Library in Extjs [2]. Z implementacijo novih tehnologij sta bili v ta namen razviti tudi dve tipični obliki predstavitve podatkov, in sicer v obliki XML ali kot notacija objektov JavaScript, poimenovana JSON.

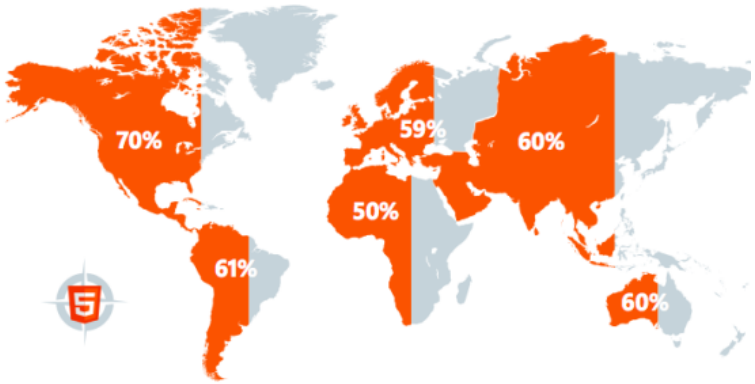
Ob zasnovi koncepta so se počasi začele razvijati tudi tehnologije, ki so uporabniku omogočile dinamično predstavitev podatkov in polno kontrolo nad shranjevanjem podatkov. Razvili so se skriptni jeziki, kot so PHP, Ruby, Pearl, Python, JSP in ASP [2]. Značilnost omenjenih jezikov je dinamično generiranje spletne strani na strani strežnika glede na podane parametre zahtevka uporabnika ter nato vrnjena in prikazana v spletnem brskalniku uporabnika. Najbolj razširjena oblika shranjevanja podatkov strežniških skriptnih jezikov je bila v namenskih podatkovnih bazah.

Implementacija novega koncepta je uporabnikom ponudila številne nove možnosti prikaza spletne strani ter omogočila podporo izdelave dinamične in interaktivne spletne strani [2]. V začetku leta 2008 se je zbralo več predstavnikov različnih skupnosti, ki so zasnovali prvi delujoči javni osnutek za implementacijo podpore prikaza multimedijskega gradiva z nadgradnjo obstoječe tehnologije HTML. Namen nadgradnje je bil implementacija tehnologije za podporo enostavnega hranjenja podatkov končnega uporabnika in implementacija standardiziranega ter enostavnega prikaza multimedijskih vsebin. Tako je nastala nova tehnična specifikacija zahtev za implementacijo novih funkcionalnosti prikaza podatkov na svetovnem spletu pod oznako “HTML5”.

2.1.1 Označevalni jezik HTML5

HTML5 je označevalni jezik za oblikovanje in predstavitev vsebine na svetovnem spletu, ki se glede na implementacijo bistveno razlikuje od prejšnje različice standarda za prikaz podatkov na svetovnem spletu HTML [2]. HTML5 je peta različica standarda HTML, ki je bila ustanovljena leta 1990 in standardizirana kot HTML4 leta 1997 ter izdana decembra 2012 pod priporočilom skupnosti W3C. Glavna cilja nove specifikacije HTML5 sta izboljšanje podpore za predstavitev multimedijske vsebine na spletu in podpore pravilnemu prikazu spletne strani na različnih vrstah naprav. HTML5 naj bi vključeval in podpiral tako HTML4 kot tudi HTML1 in HTML2.

Glede na poročilo, ki je bilo izdano 30. septembra 2011, je 34 izmed 100 najbolj obiskanih spletnih strani uporabljalo novo zasnovano tehnologijo HTML5 [2]. Slika 2.1 jasno prikazuje delež uporabe nove tehnologije HTML5 po posameznih celinah.



Slika 2.1: Delež uporabe označevalnega jezika HTML5 [5].

HTML5 skrbi tudi za pravilno interpretiranje in prikazovanje podatkov, ki so bili predmet opazovanja različnih implementacij specifikacij HTML [2]. Tako so snovalci specifikacije HTML5 poskrbeli, da je lahko zapis značk v označevalnem jeziku HTML5 v dveh oblikah, in sicer v obliki HTML ali XHTML. Vključuje podrobne procesne modele, ki poskrbijo, da se spletna stran prikaže pravilno ne glede na obliko, v kateri je bila zapisana. Tukaj

tiči tudi razlog, zakaj je standard HTML5 primeren za prikaz spletnih strani na več različnih napravah. S pomočjo implementirane tehnologije namreč poskrbi, da se opisana spletna stran v označevalnem jeziku prikaže enovito, celovito in neodvisno od naprave in pogona, ki skrbi za izrisovanje spletne strani v brskalniku.

V novo zasnovani specifikaciji je bilo dodanih veliko funkcionalnosti in elementov označevalnega jezika HTML5 [2]. Med številnimi novimi funkcionalnostmi so bili dodani atributi za podporo elementov videa, audia in polja izrisovanja grafičnih oblik kot tudi podpora za prikaz vsebine SVG. Dodana je bila tudi podpora za MathML, ki služi za prikaz zapisa matematičnih formul. Vse funkcionalnosti so zasnovane na način, da so enostavne za uporabo in združljive med številnimi napravami. Z dodanima značkama elementov videa in audia lahko uporabnik na spletno stran enostavno vključi datoteko zvoka ali videa. Dodani so bili tudi elementi, kot so na primer section, article, header in nav, ki služijo za enostavno obogatitev semantične vsebine dokumentov. Nove lastnosti so bile uvedene za isti namen, medtem ko so bili odstranjeni nekateri odvečni elementi in atributi. Skupnost se je odločila, da bo spremenila obstoječe elemente in jih standardizirala. Tako so značke a, cite in menu postale standardizirani elementi novega označevalnega jezika HTML5. Prav tako sta API in DOM postala standard in del specifikacije. HTML5 nekoliko podrobneje opredeljuje zahtevano obdelavo neveljavnega dokumenta. Sintaktične napake se v vseh spletnih brskalnikih obravnavajo enako in imajo skladen prikaz.

HTML5 med drugim uvaja tudi elemente in atribute, prikazane na Sliki 2.2, ki se širše uporabljajo na sodobnih spletnih straneh [2]. Nekateri elementi so pomenske zamenjave za uporabo generičnega bloka (element `<div>`) in medvrstični elementi (`<span>`), na primer `<nav>` (blok navigacije spletne strani), `<footer>` (običajno se nahaja na dnu spletne strani) ali `<audio>` in `<video>` namesto značke `object`. Nekateri elementi iz specifikacije HTML 4.01 so bili opuščeni, vključno s povsem predstavitvenimi elementi, kot so `<font>` in `<center>`, katerih učinki so že dolgo nadomeščeni z bolj zmoglji-

vimi kaskadnimi slogi. Prenovljeno je bilo tudi obnašanje vpliva skriptnega jezika JavaScript na implementiran DOM spletnega brskalnika.

HTML 5

Tag	Info	V	Attributes*
<!-- -->	comment	4 / 5	none
<!DOCTYPE>	document type	4 / 5	none
<a>	hyperlink	4 / 5	href | hreflang | media | ping | rel | target | type
<abbr>	abbreviation	4 / 5	*standard attributes***
<acronym>	acronym	4	-
<address>	address element	4 / 5	*standard attributes***
<applet>	applet	4	-
<area>	area inside an image map	4 / 5	alt | coords | href | hreflang | media | ping | rel | shape | target | type
<article>	article	5	*standard attributes***
<aside>	content along side from the page content	5	*standard attributes***
<audio>	sound content	5	*autoplay | controls | end | loopend | loopstart | playcount | src | start*
<b>	bold text	4 / 5	*standard attributes***
<base>	base URL for all the page links	4 / 5	href | target
<basefont>	Base font for the document	4	-
<bdo>	direction of text display	4 / 5	dir
<big>	big text	4	-

Tag	Info	V	Attributes*
<embed>	external interactive content or plugin	5	height | src | type | width
<eventsource>	target for events sent by a server	5	src
<fieldset>	fieldset	4 / 5	disabled | form
<figure>	group of media content, and their caption	5	*standard attributes***
<font>	text font, size, and color	4	-
<footer>	footer for a section or page	5	*standard attributes***
<form>	form	4 / 5	action | data | replace | accept | accept-charset | enctype | method | target
<frame>	sub window	4	-
<frameset>	set of frames	4	-
<h1> to <h6>	header 1 to header 6	4 / 5	*standard attributes***
<head>	information about the document	4 / 5	none
<header>	header for a section or page	5	*standard attributes***
<hr>	horizontal rule	4 / 5	*standard attributes***
<html>	html document	4 / 5	manifest | xmlns

Slika 2.2: Najpogosteje uporabljene značke označevalnega jezika HTML5 [6].

Sintaksa HTML5 zaradi podobnosti med sintakso starejših različic označevalnega jezika HTML ne temelji več na označevalnem jeziku SGML [2]. Specifikacija HTML5 podpira vse prejšnje različice zapisa označevalnega jezika HTML. V kolikor želimo spletno stran predstaviti v obliki HTML5, moramo na vrh dokumenta vstaviti zapis interpretacije podatkov v dokumentu `<!DOCTYPE html>`, s katerim povemo, da je zapis spletne strani v označevalnem jeziku HTML5.

V specifikacijo HTML5 je vključena tudi zmožnost prilagajanja ob nepravilni sintaksi dokumenta. HTML5 je zaradi standardizacije zasnovan tako, da lahko brskalniki ugotovijo nekatere vrste nepravilnosti v dokumentu in jih skušajo sami odpraviti [2]. V nasprotju s standardom HTML 4.01 lahko spe-

cifikacija HTML5 podaja podrobna pravila za leksikalno razčlenjevanje dokumenta z namenom, da lahko različni brskalniki dosežejo skladenjsko enak rezultat v primeru nepravilne sintakse.

Vzporedno z označevalnim jezikom se je razvijala tudi tehnologija, ki nudi podporo označevalnemu jeziku HTML – ob predstavitvi podatkov ponudi zmožnost zagotavljanja celovite grafične oblike spletne strani. V ta namen je bila implementirana nova tehnologija, ki se je razvijala vzporedno s svetovnim spletom, imenovana kaskadna slogovna polja.

2.1.2 Kaskadna slogovna polja 3

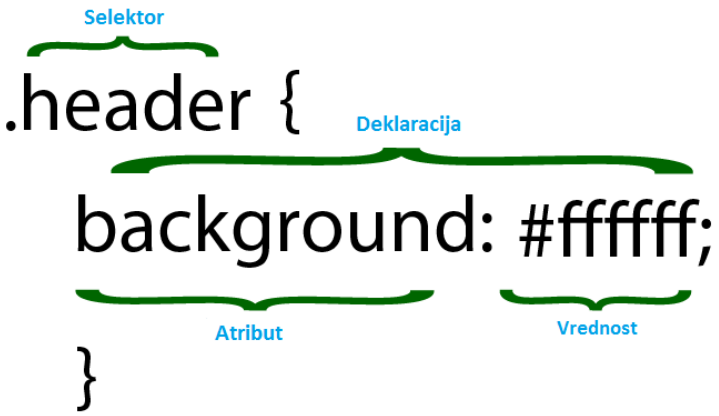
Kaskadna slogovna polja 3 so opis oblikovnega sloga, ki se uporabljajo za opis predstavitve izgleda in formata semantike dokumenta, napisanega v označevalnem jeziku [2]. Najpogosteje se uporabljajo na spletnih straneh, ki uporabljajo zapis spletne strani v označevalnih jezikih HTML ali XHTML. Uporabljajo se lahko tudi za oblikovanje dokumentov ostalih vrst zapisov, kot so XML, SVG ali XUL.

Namen kaskadnih slogovnih polj je predvsem omogočiti ločitev vsebine dokumenta, ki je napisana v označevalnem jeziku HTML, od predstavitve dokumenta, vključno z elementi, kot so postavitev, barva in pisava. Ta ločitev lahko izboljša dostopnost vsebin, zagotovi večjo fleksibilnost in nadzor v specifikaciji predstavitev. Slogovna polja zmanjšajo zapletenost in ponavljanje oblikovanja strukturirane vsebine. Omogočajo tudi enako označevanje strani s pomočjo predstavitve različnih stilov za različne načine uporabe, kot so izrisani na zaslonu ali v tiskani obliki. Uporabljajo se lahko tudi za prilagoditev prikaza spletne strani na različnih velikostih zaslona ali napravah, na katerih želi uporabnik pregledovati spletno stran. Vseeno pa lahko prihaja do neskladij, na primer ko uporabnik specificira določeno vrsto pisave, ki je na določenih sistemih ni. V tem primeru izbrana privzeta vrednost in želena vsebina, vsebovana v slogovnih poljih, ne bo prikazana in uporabljena na spletni strani.

Kaskadna slogovna polja imajo vnaprej določeno tabelo prioritet, po ka-

teri se ravnajo, če ima določen element definirano več kot eno lastnost za isti atribut. V tem primeru se uporabi oznaka, ki je definirana najnižje v drevesni strukturi dokumenta.

Kaskadna slogovna polja uporabljajo preprosto in enostavno skladnjo ter veliko angleških ključnih besed za poimenovanje atributov.



Slika 2.3: Struktura zapisa selektorja v CSS3.

Slog je sestavljen iz seznama pravil. Vsako pravilo ali sklop pravil je sestavljeno iz enega ali več selektorjev in iz bloka deklaracij atributov. Tipičen primer zapisa pravila je jasno prikazan na Sliki 2.3 z opisom sestavnih delov selektorja. V poljih se selektorji uporabljajo za poimenovanje atributov, nad katerimi bo določen slog uporabljen. Vsak selektor se lahko nanaša na:

- elemente specifičnega tipa (na primer element h2),
- elemente specifičnega atributa (atributa id in class),
- elemente, ki so odvisni od postavitve v dokumentu.

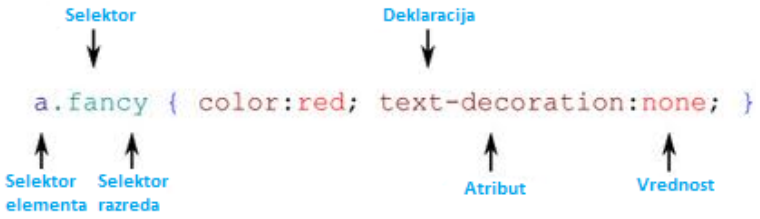
Pseudorazredi se uporabljajo s selektorji za zagotovitev oblikovanja na podlagi informacij, ki so zunaj drevesne strukture dokumenta [2]. Najpogosteje uporabljen pseudorazred je `:hover`, ki določa vsebino samo, ko uporab-

nik obišče element. Selektorju je dodan kot `a:hover` ali kot `#id:hover`. Poleg omenjenih pseudorazredov lahko uporabnik doda elementu tudi razrede, kot so `:link` in `:visited`. Način zapisa selektorja je lahko kombiniran z različnimi ostalimi možnostmi, še posebno v CSS 2.1. Uporabnik si tako zagotovi odlično fleksibilnost in specifičnost snovanja ter oblikovanja strukture dokumenta.

Pred uporabo kaskadnih slogovnih polj so se vse lastnosti oblik zapisovale v značke elementov HTML. To pomeni, da so se v značke označevalnega jezika HTML zapisovale vse velikosti barv, oblik, okvirjev in postavitev. Z uporabo kaskadnih slogovnih polj lahko definiramo stil za vse elemente v dokumentu v ločeni datoteki, kjer se s pomočjo naslavljanja sklicujemo na posamezne elemente, ki jih želimo oblikovati. S pomočjo ločevanja zapisa postavitve in oblike spletne strani lahko dosežemo tudi to, da je napisana spletna stran v označevalnem jeziku berljiva in enostavna za urejanje in vzdrževanje. Na primer, če je urejevalec spletne strani želel uporabiti isto slogovno polje za vse značke tipa `h2`, je moral vsaki znački dodati ustrezen oblikovni zapis, kar pa je predstavljalo izjemno težko in nepregledno vzdrževanje spletne strani.

Podatke o kaskadnih slogovnih poljih lahko zagotovimo iz različnih virov. Informacije o oblikah posameznih elementov so lahko zapisane v sami definiciji slogovnih polj ali pa jih zapišemo v zunanjo datoteko. Zunanjo slogovno datoteko lahko v dokument uvozimo tako, da vključimo značko `link` in v atributu `href` podamo pot do oblikovne datoteke. Poleg različnih slogovnih polj se lahko uporablja tudi več različnih zapisov oblik za posamezne naprave ali različice spletnih brskalnikov. Slednje nam je lahko vodilo pri izdelavi spletne strani, saj lahko na ta način zagotovimo optimalno delovanje spletne strani in si olajšamo vsako nadaljnje vzdrževanje spletne strani, saj bo zapis zaradi ločevanja podatkov v več datotek pregleden, enostaven in lepo berljiv.

Slog z najvišjo prioriteto v vsakem slogovnem polju nadzira prikaz vsebine. Vrednosti vseh oblik definicij elementov, ki niso definirane za posamezne podelemente, se prenesejo do njihovih potomcev oziroma do elementov,



Slika 2.4: Struktura zapisa selektorja z razredom v CSS3.

ki so definirani v območju elementa, kateremu je določeno slogovno polje dodeljeno. Primer uporabe zapisa slogovnega polja z razredom je razviden iz Slike 2.4.

Eden pomembnih ciljev kaskadnih slogovnih polj je uporabnikom zagotoviti večjo kontrolo nad predstavitvami [2]. Če nekomu ne odgovarja določeno slogovno polje nad elementom, lahko slednjega zamenja z drugim z uporabo druge predloge. Uporabnik načeloma lahko izbira med več slogovnimi polji, seveda če sama aplikacija oziroma predstavitvena spletna stran podpira implementacijo menjave slogovne oblike spletne strani.

Slogovna polja zagotavljajo uporabniku celovito in popolno kontrolo nad vizualno predstavitvijo podatkov na spletni strani. S pomočjo uporabe pravilnih slogovnih polj lahko uporabnik izdela interaktivno in atraktivno predstavitveno spletno stran.

2.1.3 Skriptni jezik JavaScript

Vsaka spletna stran, ki je zapisana v označevalnem jeziku in vsebuje kaskadna slogovna polja, lahko samostojno predstavlja celoto. Prvi spletni brskalniki so že imeli podporo za izvajanje skriptnega jezika na spletni strani. Končni uporabnik je dosegel možnost, da je spletni strani s pomočjo DOM dodal funkcionalnosti in poživil ter poenostavil samo predstavitev podatkov. Uporabniška izkušnja se je izkazala za zelo dobro, zato vsi sodobni spletni brskalniki podpirajo implementacijo skriptnega jezika JavaScript. Predstavljajmo si, da bi morali izdelati spletno stran, na kateri bi želeli izračunati

poštevanko stokratnika poljubnega števila. Ob snovanju algoritma opazimo, da bi bil problem izvedljiv na več različnih načinov, vendar bi se zahtevnost spletne strani kaj kmalu zelo povečala. Ob nepravilni implementaciji omenjene aplikacije bi lahko ob ustvarjanju spletne strani ugotovili, da je napisana spletna stran neprimerna za nadaljnje urejanje, predvsem zaradi zahtevnosti aplikacije.

V ta namen imajo brskalniki podprto implementacijo izjemno zmogljivega in razširjenega skriptnega jezika JavaScript, ki omogoča dinamično dodajanje in urejanje spletne strani.

Jezik JavaScript je skriptni jezik, ki je bil prvotno implementiran kot del spletnih brskalnikov [2]. Končni uporabniki so lahko dosegli interakcijo spletnega brskalnika z uporabnikom ter jim ponujali dodatne enostavne funkcionalnosti in asinhrono komunikacijo s spletnim strežnikom. Skriptni jezik JavaScript je prototipni jezik in ima prvorazredne funkcije. Na sintakso programskega jezika je vplival programski jezik C. Veliko skupnega ima tudi s programskim jezikom Java, saj so po njem poimenovane številne funkcije, vključuje pa tudi podobno sintakso. Jezika sicer nimata nobene utemeljene podobne primerjave v sami strukturi zasnove ter interpretiranja programske kode.

Skriptni jezik JavaScript se poleg uporabe na spletnih straneh uporablja tudi v dokumentih PDF, namiznih vtičnikih operacijskega sistema Windows in v specifično prilagojenih brskalnikih [2]. Novejša in hitrejša različica pogona JavaScript se uporablja tudi za implementacijo in podporo spletnih strani po tehnologiji strežnik-odjemalec. V ta namen je bila razvita tehnologija NodeJS, ki je trenutno ena najbolj razširjenih in uporabnih odprtokodnih tehnologij. Podpira številne funkcionalnosti, tako dostopanje do fizičnih datotek na lokalnem mediju kot dinamično asinhrono komuniciranje s spletnim strežnikom NodeJS.

Skriptni jezik JavaScript je bil formaliziran po jezikovnem standardu ECMAscript in se uporablja predvsem kot del spletnega brskalnika (na strani odjemalca), kar omogoča uporabniku programski dostop do računalniških

komponent brez dostopanja v okolje sistema odjemalca [2].

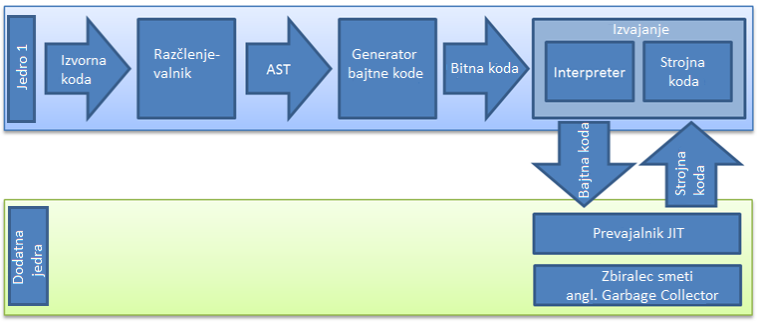
Najpogostejša uporaba skriptnega jezika JavaScript je pisanje funkcij, ki so vgrajene ali vključene v označevalni dokument spletne strani ter iteracija z DOM [2]. Nekateri primeri te uporabe so:

- nalaganje nove vsebine spletne strani ali pošiljanje podatkov na spletni strežnik s pomočjo tehnologije AJAX brez ponovnega osveževanja spletne strani,
- animacije, skrivanje in prikazovanje ter razširjanje in premikanje spletnih elementov,
- interaktivna vsebina, predvajanje glasbe in videov,
- validacija vnosnih vrednosti spletnega obrazca,
- posredovanje informacij o uporabniških bralnih navadah in aktivnostih brskanja na različnih spletnih straneh.

Interpreter skriptnega jezika JavaScript poganja programsko kodo lokalno v uporabniškem brskalniku, zato se lahko brskalnik zelo hitro odzove na uporabniške dogodke. S skriptnim jezikom JavaScript lahko zaznamo uporabniška dejanja na spletni strani, ki jih označevalni jezik HTML sam po sebi ni zmožen zaznati.

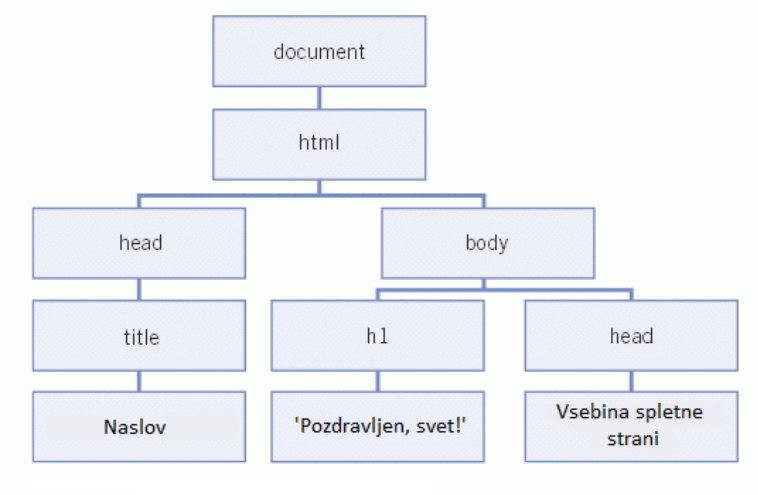
Interpreter JavaScript, poznan tudi kot tolmač JavaScript, interpretira izvorno kodo, napisano v skriptnem jeziku JavaScript, in jo ustrezno izvaja [2]. Prvi interpreter skriptnega jezika JavaScript je izdelal Brendan Eich. Dodano funkcionalnost je implementiral v programskem jeziku C in ji dodal kodno ime SpiderMonkey. Interpreter je od prve različice doživel že vrsto sprememb in vsebuje večje število novih dodanih funkcionalnosti. Podrobnejši pregled nad interpretiranjem skriptnega jezika JavaScript je prikazan na Sliki 2.5.

Zaradi razširjenosti uporabe skriptnega jezika JavaScript v spletnih brskalnikih je ta postal eden ciljnih jezikov za mnoga druga okolja, čeprav



Slika 2.5: Proces interpretiranja in izvajanja skriptnega jezika JavaScript.

njegov namen sprva ni bil tak. Kljub omejitvam hitrosti izvajanja jezika, je JavaScript doživel presenetljivo dobro izvajanje v realnem času. Posledica razširjenosti skriptnega jezika je privedla tudi do težav testiranja spletnih aplikacij na različnih okoljih. V kolikor želimo polno delujočo funkcionalnost v vseh spletnih brskalnikih, moramo vso napisano kodo v skriptnem jeziku JavaScript preveriti v vseh spletnih brskalnikih, kjer bomo izdelano aplikacijo ali spletno stran uporabljali. Tako se izognemo morebitnim težavam pri uporabi spletne strani.



Slika 2.6: Drevesna struktura elementov objektnega modela dokumenta.

Vmesnik DOM za upravljanje s spletnimi stranmi ni del standarda ECMAscript ali del jedra interpreterja JavaScript [2]. Uradno je DOM standardiziran kot ločen vmesnik. V praksi je lahko implementacija DOM med različnimi spletnimi brskalniki popolnoma različna. Enostaven primer razčlenjene oblike elementov DOM je razviden s Slike 2.6. V določenih primerih lahko dva brskalnika implementirata funkcionalnost popolnoma različno. V tem primeru mora izdelovalec spletne strani poskrbeti, da se bo spletna stran obnašala enako v vseh podprtih spletnih brskalnikih. V kolikor se želimo izogniti težavam z združljivostjo med spletnimi brskalniki, je rešitev dokaj enostavna. Implementiramo programsko kodo, ki preverja tip brskalnika, s katerim uporabnik obišče določeno spletno stran. Enostaven zapis funkcije v skriptnem jeziku JavaScript je prikazan na Sliki 2.7. Na podlagi pridobljenih informacij spletni strani povemo, katero skriptno kodo naj uporabi pri upravljanju funkcionalnosti spletne strani. Obstajajo tudi bolj enostavne rešitve. Veliko podjetij se je in se ukvarja z razvijanjem spletnih vtičnikov, ki poskrbijo, da se spletna stran izvaja neodvisno od brskalnika, vendar pa omenjeni vtičniki niso popolna rešitev težav z združljivostjo med različnimi spletnimi brskalniki.

Vzroki, da vtičniki ne delujejo vedno najbolje, se pojavljajo, zato ker uporabnik:

- uporablja staro ali redko različico brskalnika, nezdružljivo z DOM,
- uporablja naprave, ki ne podpirajo izvajanja skriptne kode JavaScript,
- ima onemogočeno izvajanje skriptne kode ali
- uporablja brskalnik, ki podpira samo prepoznavo zvoka.

V kolikor želijo podpreti pregled spletne strani uporabnikom, ki so omejeni z združljivostjo spletnega brskalnika in spletne strani, se večina razvijalcev spletnih strani opira na prakse, ki spletno stran razvijajo v smeri največje možne podpore funkcionalnosti uporabnikov. Če uporabnik nima

ustreznega spletnega brskalnika z ustrezno podprtim skriptnim jezikom JavaScript, lahko še vedno uporablja spletno stran. Ta mora biti uporabna kljub temu, da uporabnik ne more koristiti nekaterih naprednih ali dodatnih funkcionalnosti, ki jih spletna stran ponuja. Drugi pristop k razvoju spletne strani je tudi ta, da razvijalec spletne strani najprej vgradi funkcionalnosti in jih preveri v vseh brskalnikih, nato pa začne z razvojem spletne vsebine.

```
function factorial(n) {
    if (n === 0) {
        return 1;
    }
    return n * factorial(n - 1);
}
```

Slika 2.7: Enostaven primer rekurzivne funkcije, napisane v skriptnem jeziku JavaScript.

Razširjena uporaba skriptnega jezika JavaScript s sabo prinaša tudi slabe stvari. Skriptni jezik JavaScript in DOM lahko zaradi svoje razširjenosti in splošne uporabe povzročita možnost zlorabe funkcionalnosti. V izogib težavam imamo na voljo več različnih rešitev. Omenjeno spletno stran lahko zaženemo z osnovnimi varnostnimi omejitvami, s katerimi v spletnem brskalniku onemogočimo izvajanje določenih funkcionalnosti skriptnega jezika JavaScript ali celo popolnoma preprečimo njegovo izvajanje.

Najbolj razširjena možnost povzročitve škode uporabniku se nanaša na varnostni problem, ki je prepoznan pod pojmom "cross-site scripting" ali XSS [2]. Varnostna ranljivost XSS se pojavi v primeru, ko poskuša napadalec vstaviti zlonamerno kodo na spletno stran uporabnika ter se nato v primeru zagona skripte okoristiti. Napadi, ki se nanašajo na zlorabo varnostnih ranljivosti XSS, se ponavadi nanašajo na spletne strani za spletno bančništvo.

V ta namen imajo nekateri brskalniki implementirano varnostno pregrado, ki poskuša preprečiti napade XSS na nivoju samega spletnega brskalnika [2].

Brskalniki napad največkrat prepoznajo po vključitvi tujega spletnega naslova v spletno stran uporabnika. V kolikor brskalnik prepozna nedovoljene akcije spletne strani, omenjeni spletni strani onemogoči izvajanje skriptnega jezika in uporabnika opozori o nevarnosti. Varnostne ranljivosti XSS se lahko dogodijo tudi zaradi napak v implementaciji obstoječih ali novih funkcionalnosti v spletnih brskalnikih.

Naslednja večja varnostna ranljivost, ki jo je vredno omeniti, je ponarejanje ali urejanje odtisov, ki se beležijo v piškotkih spletne strani [2]. Napadalec lahko, v kolikor mu je dovoljeno izvajanje skriptnega jezika na spletni strani, prebira ali celo ureja vrednosti piškotkov spletne strani. Slednje lahko privede do varnostne ranljivosti, če se ciljna spletna stran zanaša na podatke, ki so v njih shranjeni. Če napadalec pridobi poverilce za prijavo v sistem, lahko prijavo v sistem zlorabi z umetnim ustvarjanjem vrednosti piškotkov. V tem primeru lahko napadalec dobi nepooblaščen dostop do vseh podatkov na spletni strani, v kolikor napadena spletna stran nima implementirane ustrezne varnostne zaščite.

2.1.4 Knjižnica jQuery

Z implementacijo in podporo izvajanju skriptnega jezika v vseh modernih brskalnikih so se začele razvijati tudi knjižnice, ki pripomorejo k bistveno izboljšani uporabniški izkušnji uporabe spletne strani. V ta namen je bila izdelana knjižnica jQuery, katere namen je poenostavitev dela z DOM na spletni strani.

```
function setElementsAbsolutePosition() {
    $("#content :not(.ui-resizable-handle)").each(function(){
        $(this).css({top: $(this).position().top, left: $(this).position().left });
    }).each(function(){ $(this).css('position', 'absolute')});
}
```

Slika 2.8: Enostaven primer nastavljanja absolutne pozicije elementov s knjižnico jQuery.

Knjižnica jQuery je osnovana na podpori vseh novejših brskalnikov in je združljiva z vsemi različicami spletnih pogonov. Knjižnico je razvil John Resig v začetku leta 2006, nato pa je bila v lasti več razvijalcev, ki so knjižnico posodabljali in dopolnjevali [2]. Uporaba omenjene knjižnice je zelo razširjena, uporabna in popularna, saj jo med 10.000 najbolj obiskanimi spletnimi stranmi uporablja več kot 65 % le-teh. Knjižnica je izdana pod licenco MIT in je odprtokodna brezplačna rešitev. Namen sintakse knjižnice jQuery je zagotoviti uporabniku enostavno sprehajanje po dokumentu, izbiranje elementov DOM, ustvarjanje animacij, izdelavo aplikacij z implementacijo tehnologije AJAX in enostavno upravljanje z dogodki. Enostaven zapis skriptne kode s pomočjo knjižnice jQuery je prikazan na Sliki 2.8. Knjižnica ponuja možnost ustvarjanja vstavkov. Slednje omogoča razvijalcem, da lahko ustvarijo abstrakcije na nizkonivojskem upravljanju s knjižnico, upravljajo z animacijami, naprednimi efekti in visokonivojskimi urejanimi gradniki. Modularen pristop k razvoju knjižnice jQuery omogoča ustvarjanje zmogljive in dinamične spletne strani.

Knjižnica jQuery uporabniku ponuja:

- izbiranje elementov DOM z uporabo odprtokodnega selektorja, poimenovanega “Sizzle”,
- sprehajanje po elementih DOM in njihovo urejanje,
- manipulacijo elementov DOM, ki temeljijo na selektorjih CSS,
- dogodke,
- efekte in animacije,
- AJAX,
- razširljivost z možnostjo razvoja novih vtičnikov,
- podporo izvajanju na več različnih vrstah brskalnikov,
- združljivostne metode,

- različne pripomočke.

Zaradi odlične razširljivosti knjižnice jQuery se je na spletu pojavilo veliko število njenih abstraktnih knjižnic, ki poenostavijo razvoj nekaterih gradnikov. Z uporabo omenjenih knjižnic lahko minimiziramo implementacijo gradnikov in funkcionalnosti, kot so AJAX, spletne storitve, tabele podatkov, dinamični seznami, označevalni jezik XML in orodje XSLT, dogodki, upravljanje s piškotki ter še ogromno ostalih. Številni abstraktni vtičniki jQuery so dosegljivi na repositoriju GitHub, kjer lahko vsak posameznik pomaga pri njihovem razvoju.

2.1.5 Knjižnica jQuery-UI

Najpomembnejša in tudi najbolj uporabna razširitev abstraktne knjižnice jQuery je vtičnik jQuery-UI [2]. Vtičnik jQuery-UI je abstrakcija nizkonivojskih animacij in naprednih efektov, ki temeljijo na osnovi knjižnice jQuery. Prvo različico omenjenega vtičnika je izdal John Resig leta 2007 pod licenco MIT. Vtičnik je brezplačna odprtokodna rešitev in je prosto dosegljiva vsem uporabnikom.

Omenjena knjižnica ponuja napredne možnosti dodajanja atraktivnih animacij spletnim elementom, omogoča spreminjanje velikosti in postavitve elementov ter številne nove funkcionalnosti, ki uporabniku olajšajo delo z implementacijo preprostih in najbolj uporabljenih elementov spletnih strani [2]. Vse funkcionalnosti, ki jih ponuja vtičnik, zagotavljajo možnost popolnega urejanja gradnikov in so popolnoma prilagodljive uporabnikovim željam in zahtevam.

Najpomembnejše implementacije gradnikov so:

- gumbi s posodobljenimi animacijami in izgledom,
- meniji, ki atraktivno prikažejo uporabniški meni,
- vrstice napredkov, ki atraktivno prikazujejo vrstico stanja napredka,

- zavihki, ki enostavno in učinkovito ponujajo možnost uporabe zavihkov,
- drsniki, ki ponujajo odlične možnosti vizualnega prilagajanja drsnika na spletni strani,
- izbirnik datumov, ki na učinkovit in enostaven način prikaže vse možnosti iteriranja z datumom.

Med najpomembnejše efekte vtičnika pa sodijo:

- animacija barv, kjer lahko uporabnik animirano spreminja barve gradnikov,
- različni efekti pojavitev, kjer lahko uporabnik izbira načine prikaza in skrivanja gradnikov,
- prehodi med različnimi oblikovnimi stili, kjer lahko uporabnik določi dogodke, ki spremenijo oblikovni stil določenega gradnika.

Pomembna funkcionalnost vtičnika je tudi napredno pozicioniranje elementov, kjer lahko na enostaven način dosežemo učinkovito definicijo postavitve elementov. V tem primeru lahko gradnikom spreminjamo absolutno ali relativno postavitev ter spreminjamo njihovo tretjo dimenzijo oziroma atribut `z-index`.

2.1.6 Knjižnici BlobBuilder in FileSaver

Pomembnejši knjižnici, ki ju je priporočljivo omeniti, sta tudi BlobBuilder in FileSaver [7, 4]. Obe knjižnici sta prosto dosegljivi na spletu in primerni za uporabo v spletu, ko želi uporabnik izdelati objekt podatkov in ga vrniti uporabniku.

Knjižnica BlobBuilder je zanimiva predvsem zato, ker uporabniku ponuja enostavno upravljanje s podatki na spletni strani brez posega strežniškega dela. Zgenerira lahko poljubne objekte oziroma tipe podatkov, ki jih lahko

nato ali uporabi na spletni strani ali pa jih preprosto vrne uporabniku. Zelo dober primer uporabe te knjižnice je izvoz podatkov iz spletne strani, kjer lahko s pomočjo skriptnega jezika JavaScript in omenjene knjižnice poberemo podatke s spletne strani in jih vrnemo uporabniku, brez da bi ponovno zahtevali podatke s spletnega strežnika.

Knjižnica FileSaver služi enostavnemu upravljanju s shranjevanjem datotek preko brskalnika oziroma drugače rečeno, ko uporabnik zgenerira objekt podatkov s pomočjo knjižnice BlobBuilder, lahko zgeneriran objekt vrne uporabniku preko pretoka podatkov brskalnika, nakar ima uporabnik možnost vrednost shraniti lokalno na računalniku. Knjižnica je razširitev več implementacij knjižnic in funkcionalnosti ter omogoča enostavno in varno uporabo pretoka podatkov od spletnega brskalnika do uporabnika.

V kolikor želimo zgenerirati objekt s knjižnico BlobBuilder, v skriptnem jeziku ustvarimo objekt tipa BlobBuilder ter nato v spremenljivko s pomočjo funkcije `.append(string)` dodajamo vrednosti, ki jih želimo imeti vsebovane v objektu. Ko imamo celoten objekt zgeneriran, ga pretvorimo s pomočjo funkcije `.toBlob()` v blob objekt, ki ga lahko nato s pomočjo knjižnice FileSaver vrnemo uporabniku preko pretoka podatkov brskalnika. Predtem lahko konstruktorju podamo tudi želeno ime datoteke in način pretoka podatkov, glede na tip zgeneriranih podatkov s pomočjo knjižnice BlobBuilder.

2.2 Lokalna shramba podatkov

Ob zasnovi nove specifikacije protokola HTML5 so snovalci želeli novo implementacijo hranjenja podatkov. Podali so nov koncept hranjenja podatkov, želeli so namreč doseči hranjenje trajnih podatkov izključno na strani odjemalca.

Pred snovanjem nove specifikacije protokola HTML5, ko še ni bilo poznanega izraza spletne shrambe, so se vsi podatki hranili v datotekah, imenovanih “piškotki”. Hranjenje podatkov v omenjeni obliki je bilo zelo neučinkovito. Hranjeni so bili lahko podatki v največji velikosti 4 KB. Če izdelovalec

spletne strani ni pazil na podatke, ki so bili zapisani v piškotkih, je lahko prihajalo do različnih varnostnih zlorab, kar pa je vodilo v raznorazne napade na spletne strani, prevare ter kraje zaupnih podatkov. Slabost hranjenja podatkov v tako imenovanih "piškotkih" je med drugim tudi ta, da so se ob vsakem zahtevku po osveženi spletni strani poleg vsebine nove spletne strani prenašali tudi piškotki. Prenos podatkov je znatno upočasnil pretok in prikaz spletne strani. Kot rešitev pomanjkljivosti implementacije "piškotkov" so želeli ustvariti nov koncept hranjenja podatkov, ki bi lahko vseboval veliko prostora za hranjenje podatkov, bil celoten čas na strani odjemalca in se ne bi osveževal ob zahtevkih po osvežitvi spletne strani. Uspelo jim je ustvariti strukturo novega načina hranjenja podatkov, ki je bistveno drugačen od implementacije trenutnih različic hranjenja podatkov na strani odjemalca. Tako je nastal nov koncept hranjenja podatkov, imenovan *lokalna shramba podatkov*.

2.2.1 Zgodovina hranjenja podatkov

Ob začetku razvoja spletnih brskalnikov je bil spletni brskalnik Internet Explorer med vsemi najbolj razvit in je imel implementiranih veliko različic hranjenja podatkov. Prva implementacija hranjenja podatkov v podobni tehnologiji, ki jo je Microsoft implementiral, se je imenovala "userData" [8]. Tehnologija je omogočala hranjenje podatkov v velikosti do 64 KB na eno domeno spletne strani ter desetkratnik te velikosti za spletne strani, ki so bile vključene v omrežje intranet. Prva oblika hranjenja podatkov se je izkazala za zelo učinkovito in uporabno, vendar v praksi ni nikoli dobro zaživela. Koncept je imel implementiran le brskalnik Internet Explorer, medtem ko so ostali ponudniki spletnih brskalnikov omenjeno tehnologijo šele začeli razvijati. Ob koncu leta 2002 je združba Adobe predstavila novo funkcionalnost hranjenja podatkov in jo poimenovala "Flash Cookies". Nov koncept je omogočal hranjenje lokalno deljenih objektov v velikosti do 100 KB podatkov med objekti, ki so bili v implementaciji tehnologije Flash. V nekaj letih po implementaciji "userData" metode hranjenja podatkov se je razvilo še nekaj

manj znanih protokolov hranjenja podatkov. Bolj znan protokol, ki se je razvil eno leto po Microsoftovi različici, se je imenoval "Dojo Toolkit".

Ob koncu leta 2007 je podjetje Google izdalo odprtokodni vstavek, poimenovan "Gears", s katerim so dodali določene nove funkcionalnosti k implementaciji hranjenja podatkov [8]. Koncept je temeljil na že obstoječih rešitvah in tehnologijah hranjenja podatkov.

Ob snovanju nove specifikacije so bile ključne točke v razvoju nove tehnologije izvzete iz že poznanih tehnologij in orodij, katerih namen je bil hranjenje podatkov. Nastal je nov, specificiran in standardiziran koncept hranjenja podatkov, ki so ga kasneje implementirali vsi sodobni spletni brskalniki [8].

2.2.2 Lastnosti lokalne shrambe podatkov

Lokalna shramba HTML5 je nov standard za hranjenje podatkov, ki se opira na shranjevanje podatkov v obliki ključa in vrednosti v sami integraciji spletnega brskalnika [8]. Podobno kot piškotki se podatki ohranijo, ko uporabnik zapusti določeno spletno stran ali ko zapre spletni brskalnik. Bistvena razlika med "piškotki" in spletno shrambo je v zahtevkih po osvežitvah spletnih strani. V implementaciji spletne shrambe se podatki ne osvežujejo konstantno in ne obremenjujejo pretoka podatkov osvežene spletne strani. Za razliko od standardnih "piškotkov", ki so ob vsakem zahtevku po osvežitvi spletne strani poleg vsebine poslali tudi nove vrednosti, lahko enake podatke hranimo v lokalni shrambi podatkov. Podatki se v lokalni shrambi osvežujejo na podlagi zahtevka skriptnega jezika oziroma po dejanju, ko pisec spletne strani želi, da se podatki na spletni strani osvežijo. Tako lahko dosežemo izjemno dobro kvaliteto prenosa in prikaza spletnih strani, saj se lahko vsi podatki, ki jih aplikacija potrebuje, shranijo že ob zagonu spletne strani. Naloga aplikacije je nato pravilen izris oziroma prikaz le-teh na spletni strani.

Omejitev trenutne implementacije spletne shrambe v vseh brskalnikih je največja možna velikost hranjenja podatkov, ki znaša 5 MB [8]. Velikost je trenutno določena s standardom in se v prihodnosti najverjetneje ne bo spreminjala. V kolikor uporabnik preseže dovoljeno mejo hranjenja podat-

kov v spletni shrambi, dobi opozorilo spletnega brskalnika z vsebino, da je presegel dovoljeno količino podatkov, ki jih lahko hrani v spletni shrambi. Izpisana napaka je bila standardizirana in je v vsakem brskalniku implementirana kot opozorilo `QUOTA_EXCEEDED_ERR`, če uporabnik preseže dovoljeno velikost hranjenih podatkov.

Lokalna shramba ima odlično podporo za sledenje spremembam v spletni shrambi [8]. Programsko lahko določimo dogodke, kaj se zgodi v primeru, ko se določena vrednost v spletni shrambi spremeni. Dogodek `storage` je podprt v vseh novejših spletnih brskalnikih. Sama implementacija deluje enako v vseh spletnih brskalnikih, zato izdelovalec spletne strani nima težav z združljivostjo oziroma kompatibilnostjo napisane programske kode v različnih spletnih brskalnikih.

2.2.3 Uporaba lokalne shrambe podatkov

Lokalna shramba se lahko uporablja na več različnih načinov. Vsi brskalniki podpirajo na nivoju okna privzeta imena za `sessionStorage` in `localStorage` [8]. Konceptualno sta oba načina shranjevanja podatkov v lokalno shrambo podobna, razlika je le v tem, da imata oba specifičen namen hranjenja podatkov. Tako lahko uporabnik v shrambo seje shrani vse podatke in do njih dostopa preko spremenljivke seje, vendar se ob izhodu iz spletnega brskalnika vsebina ne shrani. V kolikor želi uporabnik trajno shraniti vrednosti, lahko uporabi lokalno shrambo, kjer so mu podatki na voljo do preklica veljavnosti oziroma dokler uporabnik ne počisti vrednosti podatkov spletne shrambe.

V kolikor želimo uporabiti eno od omenjenih shramb, moramo pred uporabo preveriti, ali naš spletni brskalnik ponuja možnost uporabe spletne shrambe [8]. Enostavno lahko preverimo v pogojnem stavku, kjer kot argument podamo ime spletne shrambe. Če je rezultat pogoja pozitiven, vemo, da brskalnik podpira spletno shrambo, v nasprotnem primeru pa lahko izpišemo obvestilo, da spletna shramba ni podprta. Ko imamo vse potrebno preverjeno, lahko v shrambo seje shranimo vrednosti v obliki `sessionStorage.setItem('key', 'value');`, kjer vrednost `key` predstavlja ime ključa, pod katerim želimo

shraniti podatke, ter vrednost `value`, ki predstavlja naše konkretne podatke, ki jih želimo shraniti. Če želimo podatke shraniti v lokalno shrambo, pokličemo funkcijo lokalne shrambe `localStorage.setItem('key', 'value');` in na isti način shranimo vrednosti ključa in podatkov v lokalno shrambo. Shranjeni podatki veljajo do preklica oziroma dokler jih uporabnik ne pobriše.

Za pridobitev podatkov iz shrambe lahko enostavno pokličemo metodo `.getItem('key')` nad shrambo, ki smo jo uporabili, in dobimo vrnjeno vsebino. Podatke pridobimo s klicem metode `localStorage.getItem('key')`, pri kateri niz `key` predstavlja ključ hranjenih podatkov.

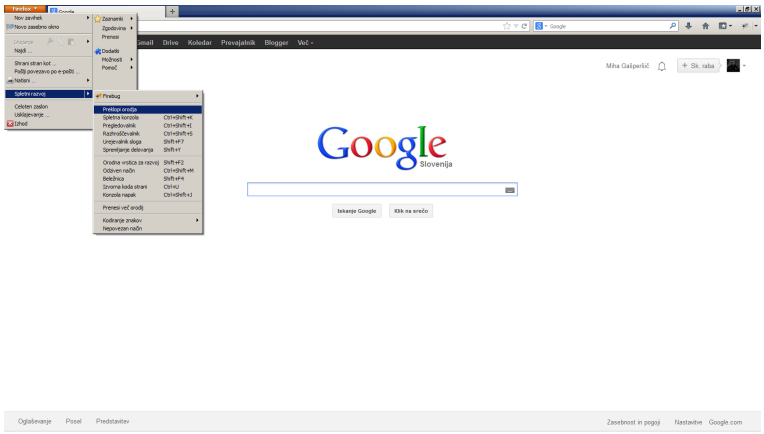
2.3 Razvojno okolje

Razvijalci spletnih aplikacij ali strani se opirajo na integrirana razvojna okolja in orodja, ki jim lajšajo vsakodnevno razhroščevanje in izdelavo produktov. V ta namen je na spletu na voljo ogromno brezplačnih in plačljivih različic namiznih aplikacij, vtičnikov in pripomočkov, ki lajšajo razvijanje spletnih strani in aplikacij. Med najpomembnejšimi aplikacijami, ki predstavljajo pomemben korak k razvoju spletnega produkta, je okolje, v katerem bomo spletno stran ali aplikacijo pregledovali. V ta namen se bomo oprli na spletni brskalnik Mozilla Firefox, ki trenutno velja za enega bolj priljubljenih brezplačnih spletnih brskalnikov. V pomoč pri razhroščevanju napak spletnih strani nam lahko služi razhroščevalnik Firebug, ki je odličen pripomoček pri pregledovanju elementov postavitve spletne strani. Za samo izdelavo spletne strani bomo uporabili brezplačno in odprtokodno integrirano razvojno okolje Komodo Edit.

2.3.1 Spletni brskalnik Mozilla Firefox

Spletni brskalnik Mozilla Firefox je brezplačna različica odprtokodnega grafičnega brskalnika, ki ga je razvilo in ga še vedno razvija podjetje Mozilla Corporation skupaj s prostovoljci [12]. Uporabniški vmesnik spletnega br-

skalnika je prikazan na Sliki 2.9. Brskalnik vsebuje odlično podporo za razvoj novih vstavkov, ki končnim uporabnikom omogočajo, da si lahko poljubno prilagodijo delovno površino spletnega brskalnika v namen, ki je njim najljubši. Podpira veliko število programskih standardov prikaza spletnih strani in je podprt v vseh različicah operacijskega sistema, tako na Windows, Linux kot na MacOS. Spletni brskalnik Mozilla Firefox ponuja tudi odlična integrirana razvojna orodja za razhroščevanje in opazovanje izpisov spletnih strani. Pri razvoju spletnih strani se lahko razvijalci zanašajo na kakovostna in učinkovita orodja.



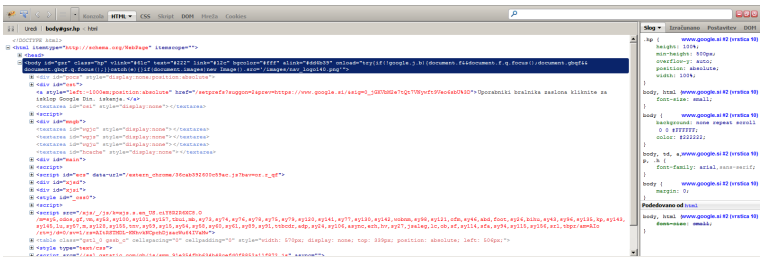
Slika 2.9: Uporabniški vmesnik spletnega brskalnika Mozilla Firefox.

V odraz kvalitetne izdelave spletnega brskalnika končnemu uporabniku je spletni brskalnik Mozilla Firefox prejel ogromno število nagrad, ki potrjujejo, da je varen, učinkovit in enostaven.

2.3.2 Razhroščevalnik Firebug

Razhroščevalnik Firebug je spletno razvojno orodje, ki ponuja odlične možnosti razhroščevanja in pregledovanja pri izdelavi spletne strani [11]. Podpira realno-časovno pregledovanje in urejanje elementov v spletnem brskalniku Mozilla Firefox in ponuja popolno možnost pregledovanja izvajanja skriptnega jezika JavaScript. Minimalističnost uporabniškega vmesnika vtičnika,

prikazan na Sliki 2.10, ponuja tudi odlične možnosti testiranja varnosti in pregleda hitrosti izvajanja spletne strani v realnem času.



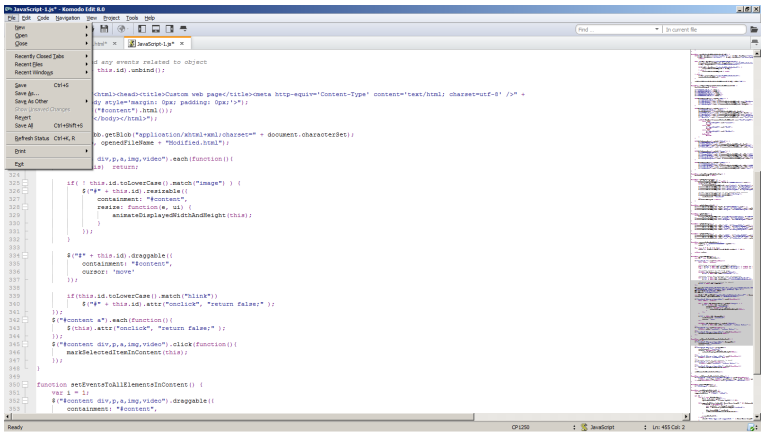
Slika 2.10: Uporabniški vmesnik razhroščevalnika Firebug.

Razhroščevalnik Firebug je odprtokodna rešitev in je bil izdan pod licenco BSD [11]. Prvo različico orodja je napisal Joe Hewit na začetku leta 2006. Pri nadaljnjem razvoju je sodelovalo večje število razvijalcev, ki so pripomogli, da so odprtokodno rešitev razhroščevalnika pripeljali na nivo, kjer lahko z gotovostjo trdimo, da ponuja najboljše možne načine razhroščevanja izdelave spletnih strani.

2.3.3 Integrirano razvojno okolje Komodo Edit

Pomemben pristop k izdelavi spletnih strani ali aplikacij je uporaba kvalitetnih integriranih razvojnih okolij, ki jih ponujajo razvijalcu orodja za lažje urejanje in izdelavo same spletne strani. Primer odličnega integriranega razvojnega okolja je Komodo Edit, prikazan na Sliki 2.11 [10].

Urejevalnik je odprtokodna rešitev, izdana pod licenco "Mozilla Public License", ki poskrbi, da lahko razvijalec izdela spletno stran z uporabo dinamičnih programskih jezikov. Med najbolj razširjenimi jeziki, ki jih urejevalnik podpira in jim nudi polno podporo, je razvoj spletnih strani v skriptnih jezikih Python, Perl, PHP, Ruby, nudi pa tudi podporo označevalnim jezikom, kot so CSS, HTML, XHTML, HTML5 in XML. Urejevalnik ima odlično podporo za razširljivost ter tako omogoči razvijalcem uporabo številnih vstavkov, ki so že prosto objavljeni na svetovnem spletu. Ponuja tudi odličen pregled



Slika 2.11: Uporabniški vmesnik integriranega razvojnega okolja Komodo Edit 8.

nad drevesno strukturo elementov DOM, inteligentnim dopolnjevanjem funkcij skriptnih jezikov, enostavno uporabo regularnih izrazov, napredne konzole ter integriran razhroščevalnik. Urejevalnik brezhibno deluje tako na operacijskih sistemih Windows, Linux kot tudi MacOS.

Poglavje 3

Sistemi za upravljanje vsebin

3.1 Sistemi CMS

Na svetovnem spletu obstaja veliko brezplačnih različic orodij, ki nam vsakodnevno lajšajo razvoj najrazličnejših aplikacij in računalniških vsebin. Tako se je vzporedno z razvojem spleta razvijalo veliko število brezplačnih in odprtokodnih rešitev za inteligentno urejanje, objavljanje in spreminjanje najrazličnejše spletne vsebine. Spletni sistemi in rešitve, ki so nastale v ta namen, so poznani pod pojmom *sistemi za upravljanje vsebin*.

Razvoj sistemov za upravljanje vsebin se je začel že v 90. [1]. letih prejšnjega stoletja za upravljanje spletnih strani, ki so služile kot pomemben vir pretoka podatkov. Sistemi so bili največkrat vključeni na spletne strani, ki so služile za bloganje, spletno nakupovanje, in spletne strani, ki so uporabnike obveščale o najrazličnejših informacijah in dogodkih.

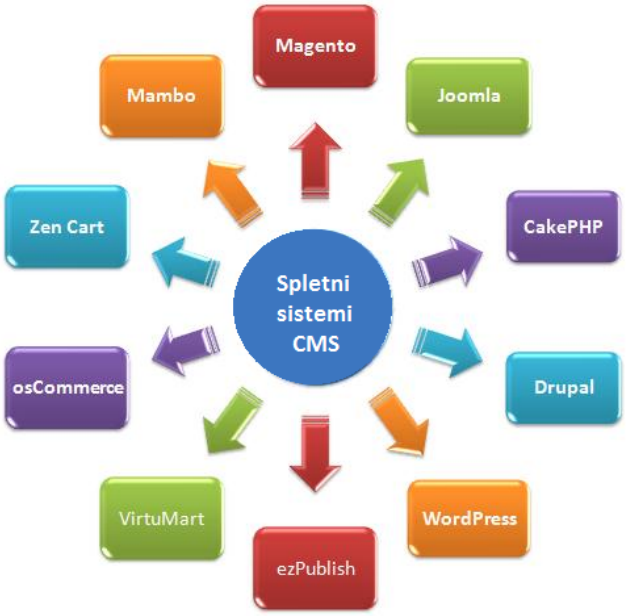
Namen sistemov za upravljanje vsebin je enostavno urejanje in upravljanje s spletno stranjo brez posebnega znanja označevalnega ali njemu podobnega jezika ter uporabe kaskadnih slogovnih polj [1]. Sistemi so v spletno stran integrirani tako, da uporabnik preko spletnih obrazcev in predlog izpolnjuje in dodaja vsebino na spletno stran ter grafično ureja postavitve in prikaz elementov na spletni strani. Drugi glavni namen sistemov je zagotavljanje hitrosti objave informacij. Z upravljalskimi sistemi lahko hitro

objavimo in uredimo novo dodano spletno vsebino, kar lahko uporabniku svetovnega spleta omogoča hitro osveževanje vseh pomembnih dogodkov in informacij, s tem pa pridobi na popularnosti vzdrževane spletne strani. Na svetovnem spletu obstaja veliko sistemov za upravljanje vsebin, katerih funkcionalnosti so lahko zelo različne glede na primerjavo posameznih sistemov. Nekaj primerov plačljivih in brezplačnih različic sistemov za upravljanje s spletnimi vsebinami lahko razberemo s Slike 3.1. V praksi so zastonjske različice sistemov za upravljanje vsebin enostavne za uporabo in funkcionalno primerljive s plačljivimi različicami. Plačljivi sistemi, ki jih razvijajo podjetja, so funkcionalno popolni in vsebujejo številne dodatne možnosti upravljanja in urejanja spletne strani. Večina osnovnih in naprednejših sistemov za upravljanje spletnih vsebin ponuja funkcionalnosti sistemov, kot so dodajanje vsebine preko spletnih obrazcev, ustrezno oblikovanje spletne vsebine, upravljanje z različicami spletne vsebine, označevanje, številčenje ter iskanje. Sistemi za upravljanje vsebin ponavadi predstavljajo tudi ključno točko v hranjenju najrazličnejših dokumentov, slik, videov, znanstvenih podatkov in besedil.

Sisteme za upravljanje spletnih vsebin lahko glede na namen kategoriziramo v tri večje skupine. Tipično jih lahko ločimo na spletne sisteme za upravljanje vsebin, komponentne sisteme za upravljanje vsebin ter sisteme za upravljanje vsebin podjetij [1]. Sistem za upravljanje vsebin, ki je za nas najbolj zanimiv in ga bomo tudi podrobneje opisali, se med drugim nanaša tudi na nadaljnji razvoj orodja za grafično urejanje spletnih strani. V ta namen bomo podrobneje predstavili spletne sisteme za upravljanje vsebin.

3.2 Spletni sistem CMS

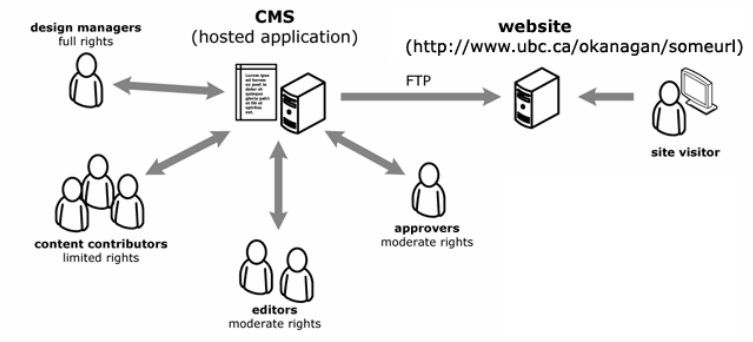
Spletni sistem za upravljanje vsebin ali spletni sistem CMS je sistem, ki ponuja urejanje spletnih vsebin ter sodelovanje skupin pri objavljanju vsebine na spletni strani [1]. S pomočjo administrativnih orodij lahko uporabnik z minimalno količino znanja programskih ali označevalnih jezikov izdela in



Slika 3.1: Seznam najpogosteje uporabljenih sistemov za upravljanje vsebin.

ureja svojo predstavitveno spletno stran. Omenjeni sistemi ponujajo funkcionalnosti, ki zagotavljajo več različnim uporabnikom, da lahko isto vsebino urejajo vzporedno in neodvisno. S celovitimi funkcionalnostmi sistemi ponujajo najrazličnejše možnosti urejanja, dodajanja in oblikovanja vsebine, ki jo želijo uporabniki predstaviti na spletni strani. Večina spletnih sistemov za upravljanje vsebin v ozadju uporablja skladišče vsebine ali podatkovno bazo za shranjevanje vsebine spletne strani, metapodatkov ali ostalih podatkov, ki so potrebni za celovito predstavitev spletne strani. Predstavitveni nivo sistemov za upravljanje vsebin skrbi za prikaz spletne strani odjemalcem, ki se povežejo na spletno stran. Za hitrejše in bolj optimalno delovanje večina naprednih sistemov uporablja predpomnenje spletne strani, da lahko, v kolikor uporabnik pošlje zahtevek po spletni strani, aplikacija v trenutku vrne želeno spletno vsebino. Večina sistemov ponuja urejanje spletne vsebine preko spletnega brskalnika, kjer obstaja ločena vstopna stran za prijavo v sistem urejanja spletnih vsebin. V praksi imajo manjše skupnosti skrbnika, ki s pomočjo sistema za upravljanje vsebin skrbi za posodabljanje spletne strani.

V večjih skupnostih pa ponavadi obstaja več skrbnikov spletnih strani.



Slika 3.2: Tipično delovanje sistemov za upravljanje vsebin [9].

Namen spletnega sistema za upravljanje vsebin je dinamično upravljanje z zbirko podatkov spletnega gradiva, vključno z označevalnimi dokumenti HTML, slikami, videi ter vsemi ostalimi multimedijskimi vsebinami [1]. Tipičen primer delovanja sistemov za upravljanje vsebin lahko razberemo s Slike 3.2. Spletni sistem olajša upravljanje z vsebinami, revizijami, urejanjem in časovnim vodenjem urejanja spletnih vsebin. Sistemi za upravljanje spletnih vsebin imajo ponavadi funkcionalnosti, ki zagotovijo izdelavo samodejnih predlog, upravljanje dostopa, enostavno razširljivost, enostavno urejanje vsebine, enostavno dodajanje novih funkcionalnosti, posodabljanje sistema, upravljanje poteka dela, sodelovanje, urejanje dokumentov, grafično urejanje elementov, podporo večjezičnosti in verzioniranje. Glede na tip sistemov za upravljanje vsebin pa ločimo sisteme, ki zagotavljajo urejanje in predogled vsebine pred objavo na spletni strani, ter sisteme, ki omogočajo urejanje in predogled spletne vsebine v realnem času. Poleg omenjenih dveh možnosti pa obstajajo tudi sistemi, ki uporabljajo obe tehniki pristopa k urejanju spletne strani.

3.3 Prednosti sistemov CMS

Zaradi popularnosti in razširjenosti uporabe sistemov za upravljanje spletnih vsebin se vse več skupnosti in posameznikov odloča za integracijo omenjenih sistemov na svoji spletni strani. Glavno prednost uporabe spletnih sistemov za upravljanje vsebin predstavlja dejstvo, da je veliko sistemov na voljo vsem uporabnikom brezplačno in brez stroškov namestitve [1]. Prav tako se je uporaba teh sistemov razširila zaradi enostavne prilagodljivosti, uporabe, upravljanja poteka dela, s čimer predstavljajo odlično orodje za optimizacijo spletnih strani.

3.4 Slabosti sistemov CMS

Čeprav imajo sistemi za upravljanje vsebin mnogo prednosti, imajo seveda tudi nekatere pomanjkljivosti. Kot njihovo bistveno pomanjkljivost lahko izpostavimo dejstvo, da so plačljive različice omenjenih sistemov lahko zelo drage, prav tako tudi njihovo vzdrževanje. Iz tega razloga se veliko skupnosti odloča za implementacijo brezplačnih rešitev. Z zahtevnostjo sistemov pa se lahko veča tudi odzivnost spletne strani. Če je sistem zahteven, mora strežnik za procesiranje podatkov porabiti bistveno več sistemskih virov, kot jih porabi enostaven sistem.

3.5 Razvoj sistema CMS v tehnologiji HTML5 z implementacijo spletne shrambe

Ob uporabi in poznavanju spletnih sistemov za upravljanje vsebin smo prišli do ugotovitve, da je trenutno zelo malo sistemov, ki bi končnemu uporabniku oziroma upravljalcu spletnih vsebin ponujali enostavno, minimalistično in uporabniku prijazno aplikacijo, ki bi temeljila na ločeni podpori integracije pri urejanju spletnih strani. Dobili smo idejo, da bi izdelali orodje, ki bi za razliko od ostalih spletnih sistemov namesto podatkovne baze oziroma

skladišč podatkov uporabljala izključno implementacije tehnologij na strani odjemalca oziroma izdelali bi orodje, ki za delovanje ne bi potrebovalo strežniške strukture. Pomembnejša ideja, ki se nam je porodila, je tudi zagotovitev nemotenega pregleda uporabniku in upravljanje spletne strani brez nepotrebnega procesiranja integracije sistema za upravljanje vsebin.

Tako smo se odločili, da bomo razvili orodje, ki nam bo omogočilo grafično upravljanje s spletnimi stranmi. Orodje bo uporabljalo podobne koncepte funkcionalnosti kot sistemi za upravljanje spletnih vsebin. Naš sistem se bo razlikoval v arhitekturni zasnovi ter bo implementiral lokalno skladiščenje podatkov na strani odjemalca. Orodje, ki ga bomo razvijali, je v nadaljevanju besedila poimenovano "orodje WebsiteEdit".

Poglavje 4

Razvoj orodja WebsiteEdit

4.1 Ideja orodja

Implementacija in razvoj sistemov za upravljanje vsebin se lahko zelo zapleteta, če želi razvijalec sistema podpreti vse bistvene in ključne funkcionalnosti uporabnega sistema. Po pregledu obstoječih tehnologij smo prišli do ugotovitve, da nekateri sistemi, ki upravljajo s spletnimi vsebinami, ne uporabljajo najnovejših tehnologij in njihovih implementacij. Tako se nam je porodila ideja, da bi izdelali orodje WebsiteEdit za podporo izdelave nove spletne strani. Po temeljitem pregledu trenutnih standardiziranih tehnologij smo prišli do ugotovitve, da je mogoče veliko funkcionalnosti, ki so v trenutnih sistemih za upravljanje vsebin, implementirati na enostaven način. Z uporabo novejših tehnologij se lahko razvoj sistemov za upravljanje s spletnimi vsebinami zelo poenostavi.

Ideja za razvoj orodja se je razvila iz preprostega stališča uporabnosti, saj je trenutno malo podobnih sistemov za upravljanje spletnih vsebin, ki bi podpirali razvoj spletne strani in njeno vzdrževanje na zelo enostaven način ter bi omogočali njeno urejanje na uporabniku prijazen način. Funkcionalnosti, ki jih želimo podpreti v novem orodju WebsiteEdit, so predvsem enostavno grafično urejanje spletne strani, enostavna možnost premikanja elementov po spletni strani ter možnost prilagajanja spletnih elementov glede na želje

uporabnika oziroma njegove izkušnje. Porodila se nam je tudi ideja, da bi ob razvoju orodja WebsiteEdit ponudili hranjenje podatkov na strani uporabnika brez strežniške strukture v ozadju delovanja orodja. Hranjenje želimo podpreti s pomočjo spletne shrambe, saj je trenutna implementacija funkcionalnosti v spletnih brskalnikih dobro podprta.

Ob sami implementaciji spletne strani se nam je porodila tudi želja, da se uporabniku ponudi možnost uvažanja in izvažanja spletne strani, v kolikor bi jo želel med razvojem podrobneje pregledati v spletnem brskalniku. Izvažanje in uvažanje podatkov novo ustvarjene spletne strani naj bi bilo s stališča uporabniške izkušnje dinamično in enostavno. Zaželeno je, da uporabnik preko atraktivnih gumbov izbere akcije, ki bi jih želel izvesti, kot na primer shraniti izdelano spletno stran ali jo po potrebi ponovno uvoziti v urejevalnik spletnih strani.

Prav tako bi moralo orodje WebsiteEdit zagotoviti, da so uvoženi in izvoženi podatki konsistentni in da ustrezajo pravilom, ki jih orodje zahteva za pravilno delovanje. Ob izvozu ali uvozu podatkov obstoječe spletne strani, bi moralo orodje poskrbeti, da so vsi podatki primerno pripravljeni, da lahko uporabnik izvaja določene akcije oziroma spletno stran poljubno oblikuje. Prav tako se morajo vse sledi spletnega orodja ob izvozu urejane spletne strani odstraniti.

Med pomembnejšimi funkcionalnostmi orodja je tudi izdelava atraktivnega videza. Orodje bi imelo izdelan uporabniški vmesnik, ki bi uporabnika že po videzu prepričal o kvaliteti in uporabnosti orodja za urejanje spletne strani. Priporočljivo bi bilo, da se v smiselnih modulih uporabi funkcionalnost implementacije animacij skrivanja in prikazovanja elementov. Animirani deli orodja naj bi imeli atraktivne in smiselne prehode ter bi uporabniku zagotovili nemoteno upravljanje s spletno stranjo in po potrebi ponudili možnost izklapljanja animacij.

4.2 Analiza in specifikacije zahtev orodja

4.2.1 Želene funkcionalnosti orodja

Glede na podano idejo smo opredelili posamezne funkcionalnosti in zahtevke za izdelavo novega spletnega orodja. Opis želenih funkcionalnosti je okvirne narave in vsebuje približne opise točk funkcionalnosti, ki jih bo naše orodje vsebovalo.

Stališče uporabniškega vmesnika mora temeljiti na razvoju modularnih delov, ki morajo biti med sabo neodvisni in enostavno implementirani. Stališče je pomembna točka v razvoju spletnega orodja in je izhodiščne narave, nanj se bomo opirali skozi celoten razvoj orodja.

Sam uporabniški vmesnik orodja naj bo razdeljen na več polj, kjer bodo sistematično in konceptualno združene funkcionalnosti, ki jih bo orodje podpiralo. Tako želimo, da se v zgornjem delu orodja implementira zaglavje, v katerem bodo vsebovani podatki o imenu ter možnosti za uvoz in izvoz podatkov urejane spletne strani. Nadaljnja postavitev spletne strani naj bi temeljila na ločevanju urejevalnega dela in možnosti atributov, ki so lahko urejani. Na podlagi teh podatkov predlagamo, da se celotna vsebina spletne strani, ki jo uporabnik lahko ureja, prikaže uporabniku na sredini delovne površine, vsa možna polja in podatki, ki jih lahko ureja, pa se mu pokažejo v namenskem polju ob levi strani ekrana. Namensko polje za urejanje podatkov spletih gradnikov lahko vsebuje animacijo skrivanja in prikazovanja, prav tako pa je priporočljivo, da se vsebina spletnega orodja dinamično prikazuje in skriva. V namenskem polju naj bo rezerviran del polja, v katerem bodo prisotne možnosti klica akcij, ki jih lahko uporabnik izvaja nad vsemi elementi. Zagotoviti želimo tudi možnost uporabniškega naprednega izbiranja elementov v polju, kjer bo prikazana vsebina spletne strani. Dodane naj bodo tudi funkcionalnosti, ki bodo uporabniku omogočile enostavno grafično razširjanje in premikanje elementov po spletni strani.

Orodje naj uporabniku ponuja izdelavo okvirja poljubne velikosti, barve in postavitve. Prav tako naj ponudi možnost dodajanja poljubnega bese-

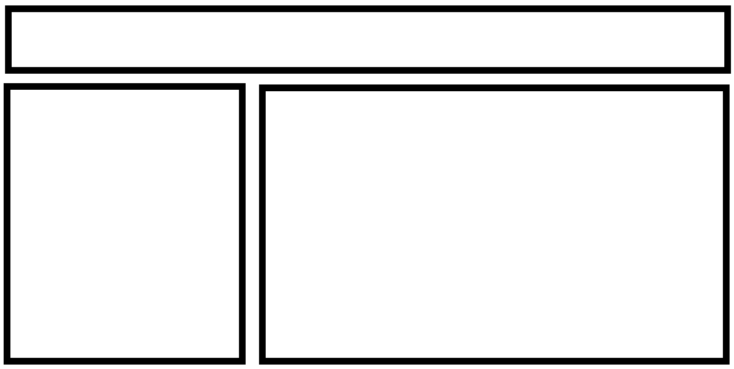
dila z možnostjo urejanja vseh atributov, ki se lahko dodajajo nad poljem besedila. Za povezovanje funkcionalnosti spletne strani naj orodje poskrbi z dinamičnim upravljanjem s spletnimi povezavami in ima polno podprte funkcionalnosti, s katerimi bo lahko ponudilo uporabniku navigacijo po spletni strani. Od multimedijskih vsebin mora uporabniku ponuditi možnost vključevanja slik in videov na spletno stran ter možnost urejanja velikosti in postavitve vseh gradnikov.

4.2.2 Specifikacija zahtev orodja

Glede na podane želje funkcionalnosti orodja bomo podrobneje opredelili posamezne module, ki jih bomo posamično implementirali v orodju WebsiteEdit. V prvem delu bomo opisali točno obnašanje in postavitev vseh večjih polj orodja na spletni strani. Nato se bomo lotili podrobnejše analize vsakega posameznega modula, kjer bomo opredelili funkcionalnosti, grafični izgled ter morebitne animacije elementov in okvirja na spletni strani orodja. Ob zaključku se bomo podrobneje dotaknili tudi grafičnega izgleda uporabniškega vmesnika in točneje določili izgled, postavitev in barve orodja. Vzporedno z definicijami modulov bomo pri posameznih elementih dodali tudi opise funkcionalnosti za implementacijo spletne shrambe v modulih.

Orodje bo razdeljeno v tri glavne okvirje, kjer bo vsak okvir podrobneje opisan in predstavljen s funkcionalnostmi, dizajnom in načinom interakcije z uporabnikom. Okvirna postavitev okvirjev je prikazana na Sliki 4.1.

Prvi okvir bo predstavljal zaglavje orodja in bo vključen v zgornji del spletnega brskalnika. V okvirju, kot je prikazano na Sliki 4.2, bo zapisano ime spletnega orodja ter dve funkcionalnosti uvoza in izvoza urejane spletne strani. Uporabnik bo ob kliku na izbiro gumba lahko uvozil že obstoječo spletno stran. Odprlo se mu bo pogovorno okno za izbiro spletne strani, ki jo bo želel uvoziti. Ob kliku na gumb izvoz, bo orodje uporabnika vprašalo po želenem imenu datoteke ter bo nato uporabniku ponudilo možnost, da bo ustvarjeno spletno stran shranil lokalno na računalnik in jo bo po potrebi preveril v spletnem brskalniku. Oba gumba bosta vsebovala atraktivno izbiro



Slika 4.1: Postavitev glavnih okvirjev orodja.

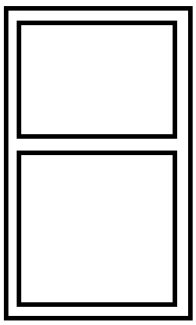
in bosta v skladu z začrtano obliko ter dizajnom orodja. Ozadje in oblika zaglavja bo prehod barv iz sive proti beli, in sicer v smeri od leve proti desni strani orodja. Na spodnji strani bo podana rdeča črta, pod njo pa senca za boljši izgled plošče.

Slika 4.2: Okvirna postavitev elementov v zaglavju spletne strani.

V drugem okvirju bodo vsebovane vse možne funkcionalnosti, ki jih lahko uporabnik izvaja nad določenimi elementi. Postavitev elementov drugega okvirja je predstavljena na Sliki 4.3. Okvir bo v orodju vključen na levo stran delovne površine brskalnika in se bo, v kolikor bo imel uporabnik izbrano možnost, animiral glede na vhod oziroma izhod miške v polje. Okvir se bo animiral z leve strani ekrana proti desni, v kolikor bo uporabnik želel okvir prikazati, ter iz desne proti levi, v kolikor bo želel uporabnik okvir skriti. V zgornjem delu okvirja bodo vsebovane vse ključne funkcionalnosti, ki se bodo lahko uporabljale nad vsemi elementi v urejevalniku spletnih strani. Funkcionalnosti, ki bodo na voljo vsem gradnikom, bodo brisanje in

dodajanje vnaprej določenih elementov, kopiranje in lepljenje že obstoječega elementa ter spreminjanje elementove vrednosti `z-index`, v kolikor bo želel uporabnik spremeniti položaj prikaza gradnika na spletni strani. Uporabnik bo lahko dodal na spletno stran pravokotnik, besedilo, spletno povezavo, slike in videe. V kolikor bo želel spreminjati vrednosti posameznih gradnikov, dodanih na spletno stran, bo lahko v polju, kjer bo prikazan predogled spletne strani, z miškinim klikom označil želeni atribut. Ob označbi atributa se bodo uporabniku nato v omenjenem okvirju prikazali vsi atributi, ki jih bo lahko spreminjal in definiral po svojih željah. Vsak element bo imel nekaj specifičnih lastnosti, vsem elementom pa bo skupno nastavljanje pozicije `x` in `y` glede na pozicijo elementa v polju za urejanje spletne strani. Gradniku pravokotnik bo lahko uporabnik spreminjal in urejal atribute, kot so barva pisave, širina in dolžina elementa ter pozicija elementa glede na levi in desni rob. Ob izbiri besedila bo lahko uporabnik spreminjal možnosti barve, stila, velikosti in težo pisave ter tudi samo besedilo, ki bo prikazano v izbranem gradniku. Ob vključitvi gradnika spletne povezave v spletno stran bo lahko uporabnik spreminjal ime povezave, povezavo do želene spletne strani ter način prikaza spletne strani. Pod načinom prikaza spletne strani pojmujemo prikazovanje spletne strani v trenutnem, novem ali predhodnem oknu spletne strani. Ko bo uporabnik vključil na spletno stran multimedijsko vsebino, ki bo lahko slika ali video, mu bo ponujena možnost spreminjanja atributov širine in dolžine, pozicije elementa glede na položaj v spletnem urejevalniku, besedila, v kolikor slika ne bo na voljo, ter pot do multimedijske vsebine. V kolikor bo želel uporabnik orodja spremeniti vrednost atributov spletne strani, bo najprej izbral želene vrednosti ter nato s potrditvijo na ustrezen gumb posodobil vrednosti na podanem gradniku. Oblika in ozadje okvirja bosta predstavljena s sivo barvo ozadja ter z rdečo črto zunaj okvirja. Z zunanje strani bo dodana senca. Posamezen sklop elementov na spletni strani bo ločen s črto ločnico, ki bo ločevala posamezne konceptualno podobne elemente. Prva ločnica bo ločevala ime okvirja z gumbi, ki bodo skupni vsem gradnikom, druga bo ločevala skupne funkcionalnosti s tistimi, ki se bodo

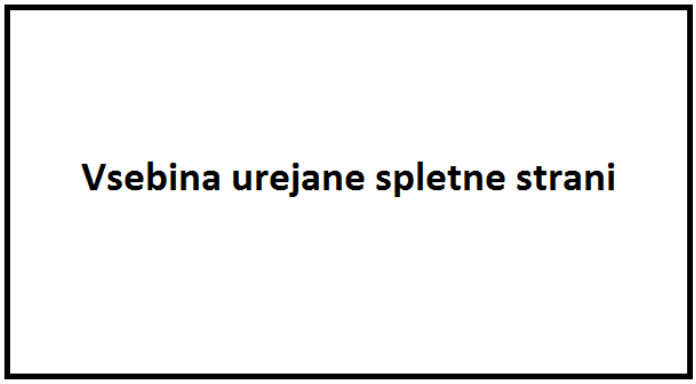
dinamično prilagajali glede na izbran atribut v polju vsebine.



Slika 4.3: Okvirna postavitev elementov stranske plošče orodja.

Tretji in hkrati zadnji okvir orodja pa bo vsebovan v sredini spletnega orodja in bo predstavljal glavni del funkcionalnosti našega orodja. V tem delu bo lahko uporabnik ročno z uporabo miške prilagajal elemente, jih premikal po delovni površini in spreminjal njihovo višinsko pozicijo. Dodatna funkcionalnost orodja bo ta, da bo lahko uporabnik ob kliku v polju izdelave spletne strani izbral določen gradnik, nakar se mu bodo v drugem okvirju prikazale možnosti gradnika, ki jih lahko ureja. V kolikor bo uporabnik izbral drug element, se mu bo prejšnji izbrani element odznačil ter obarval novo izbrani element. Uporabniku bo ponujena tudi možnost naprednega spreminjanja elementove dolžine in širine. Slednje bo lahko dosegel tako, da se bo z miško premaknil na rob elementa in s pritiskom ter vlečenjem levega miškinega gumba spremenil dolžinske vrednosti gradnika. Na podoben način bo lahko uporabnik spreminjal tudi pozicijo elementa na spletni strani. Imel bo možnost, da s klikom na element izbere gradnik in ga vleče po delovni površini. V kolikor bo uporabnik uporabljal eno od omenjenih funkcionalnosti, se mu bodo dinamično spremenjene vrednosti prikazale v drugem polju, tako da bo imel grafični in besedilni pregled nad elementi, ki jih bo spreminjal. Implementacija spletne shrambe bo določala, da se bo celotna vsebina delovne površine samodejno shranjevala vsakih petnajst sekund in se bo po potrebi, ob ponovnem zagonu orodja, ponovno prikazala v delovnem polju

orodja. Vsak novo dodani element bo imel najvišjo pozicijo atributa `z-index` med vsemi gradniki, in v kolikor bo možnost, še vnaprej definirano velikost ter naključno barvo. Delovna površina bo statična glede na širino in dolžino spletnega brskalnika in je ne bo moč spreminjati. Postavitev okvirja vsebine orodja WebsiteEdit je prikazana na Sliki 4.4. Barva ozadja tretjega okvirja orodja bo obarvana svetlo sivo z dodano rdečo obrobo in senco.



Slika 4.4: Okvirna postavitev elementov vsebine orodja.

Ozadje orodja bo na mestih, kjer ne bo imelo definiranih okvirjev, bele barve, brez dodanih senc in okvirjev. Širina in dolžina prvega okvirja bosta določeni vnaprej in se ne bosta spreminjali. Prav tako bosta dolžina in širina drugega okvirja določeni vnaprej in se ne bosta spreminjali. Tretjemu okvirju bo dolžina določena glede na dolžino delovne površine spletnega brskalnika, ki ji bomo odšteli dolžino drugega okvirja. Privzeta barva pisave bo rdeča, prav tako tudi vse obrobe elementov orodja.

4.3 Načrtovanje orodja

Na podlagi podrobne specifikacije zahtev orodja se bomo v nadaljevanju lotili podrobnejšega načrtovanja posameznih modulov orodja. Rezultat načrtovanja orodja bo podrobna specifikacija modulov, njihovo obnašanje ter sam

koncept zasnove uporabniškega vmesnika orodja.

4.3.1 Načrtovanje modulov orodja

Zelo pomemben korak tako pri načrtovanju kot implementaciji modulov orodja je njegova podrobnejša analiza in definicija, kako se mora posamezen modul obnašati, kakšne so njegove funkcionalnosti ter na kakšen način bodo le-te implementirane [3]. Trenutno obstaja veliko različnih teorij, načinov in pristopov k načrtovanju modernih aplikacij. Implementacije orodja WebsiteEdit se bomo lotili na klasičen način, ker je le-ta najbolj enostaven in trenutno najbolj primeren način izvedbe načrtovanja orodja glede na podane kriterije. S klasičnim načrtovanjem orodja želimo doseči, da bomo iz logičnega, implementacijsko neodvisnega modela izdelali fizičen, implementacijsko neodvisen model, ki bo maksimiziral stopnjo medsebojnih aktivnosti znotraj komponent, minimiziral število različnih elementov in odstranil relacije podvajanja med komponentami.

Razvoj in načrtovanje orodja bosta vsebovala opis implementacije orodja v označevalnem jeziku HTML5 ter v skriptnem jeziku JavaScript z vključenimi knjižnicami jQuery, jQuery-UI ter implementacijo vtičnikov BlobBuilder in FileSaver. Pri načrtovanju orodja se bomo opirali na zmožnosti funkcionalnosti označevalnega jezika HTML5 ter ob možnosti vključitve njegove funkcionalnosti uporabili namenske strukture za implementacijo funkcionalnosti.

Zaglavje orodja bo glede na podano specifikacijo vsebovalo dve ključni točki, saj bo uporabniku ponujalo možnost uvoza in izvoza generirane spletne strani iz spletnega orodja. Implementacija modula za uvoz obstoječe spletne strani, zgenerirane z orodjem, bo delovala tako, da bo imel uporabnik na voljo gumb, s katerim se bo ob kliku odprlo pogovorno okno, kjer bo lahko izbral želeno spletno stran in jo uvozil v orodje. Postavitev elementov v zaglavju orodja WebsiteEdit je vidna na Sliki 4.5. Označevalni jezik HTML5 podpira značko, s katero lahko uporabnik ob dogodku klika na element spletne strani in uporabi pogovorno okno za izbiro datoteke. Namenska značka

je tipa datoteke in poimenovana input. Za primer lahko uporabimo format zapisa `<input type="file">`, kjer mu lahko kot atribut `id` podamo ime elementa, na katerega se nato v skriptnem jeziku sklicujemo. Dogodek klika se največkrat implementira kot klik na spletno povezavo, nakar ob kliku pokličemo funkcijo, ki bo nad značko izbire datoteke izvedla klic metode `.click()` in odprla pogovorno okno za izbiro datoteke. Za načrtovanje implementacije funkcionalnosti za shranjevanje vsebine, ki jo bo uporabnik izdelal, bomo uporabili implementacijo dogodkovnega klika, ob katerem se bo uporabniku sprocesirala spletna stran in vrnila vsebino spletne strani. V ta namen bo napisana namenska funkcija, ki bo poskrbela, da bo vnaprej preverila, ali obstaja vsebina spletne strani, ki jo bo uporabnik izdelal s pomočjo orodja. V kolikor bo vsebina prazna, bo orodje uporabnika obvestilo ter prekinilo s procesiranjem izvoza spletne strani. Če bo vsebina obstajala, bo moralo orodje s pomočjo vtičnika BlobBuilder zgenerirati vsebino spletne strani na dinamičen način, odstraniti bo moralo vse sledi uporabe orodja za izdelavo spletne strani. Tako bo moralo orodje poskrbeti, da gradniki ne bodo imeli nepotrebnih funkcionalnosti, ki jih orodje potrebuje pri svojem delovanju. Aplikacija bo morala prav tako omogočiti uporabniku izbiro imena vrnjene datoteke, kjer bo uporabnika v pogovornem oknu povprašala o želenem imenu spletne strani. Orodje bo nato s pomočjo funkcije `saveAs()` vrnilo zgeneriran objekt blob uporabniku s podanim parametrom imena datoteke. Ob koncu procesiranja bo moralo orodje poskrbeti za ponovno vzpostavitev vseh elementov v predhodno stanje, ki bo prisotno pred začetkom izvoza spletne strani.

Slika 4.5: Postavitev elementov zaglavja spletne strani.

Stranska plošča bo v zgornjem delu okvirja vsebovala implementacijo

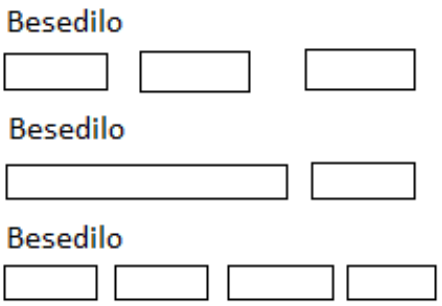
funkcionalnosti, ki jih bo lahko uporabnik izvajal nad vsemi elementi, ki bodo vključeni na spletno stran. Orodje bo tako ponudilo možnost brisanja, kopiranja in lepljenja elementov. Uporabniku bo omogočilo tudi dodajanje novih elementov ter spreminjanje višinske pozicije posameznih elementov s pomočjo atributa `z-index`. Vse funkcionalnosti, ki bodo implementirane, bodo implementirane s pomočjo značk gumb HTML5 ter bodo ob kliku na gumb sprožile akcijo, ki bo izvedla določeno spremembo.

V kolikor bo želel uporabnik odstraniti element s spletne strani, bo ob kliku na gumb odstranil element. Namenska funkcija mora poskrbeti, da se bodo odstranile vse akcije in relacije, ki jih bo vključeval omenjeni element. Uspešnost odstranjevanja elementa bo odvisna od izbranega elementa v polju vsebine. Če uporabnik ne bo imel izbranega elementa, se brisanje ne bo izvedlo, v kolikor pa bo imel element izbran, se bo izbrani element pobrisal s spletne strani. Uporabnik bo imel na voljo tudi kopiranje elementa. Slednje bo lahko, v kolikor bo izbral želeni element v polju vrednosti, ob kliku na ustrezen gumb skopiral vrednosti in jih začasno uskladiščil v orodju. Ob kliku na ustrezen gumb lepljenja bo lahko uporabnik podvojil element in s tem pridobil dva popolnoma enaka elementa z različnima atributoma `id`. Če uporabnik ob kliku na ustrezen gumb kopiranja ne bo imel izbranega elementa, se kopiranje ne bo izvedlo.

Uporabnik bo imel na voljo dodajanje novega spletnega elementa. V polju izbire bo lahko izbral element, ki ga bo želel dodati na spletno stran. Polje izbire bo imelo vnaprej določene značke, ki jih bo uporabnik lahko uporabil na spletni strani. Izbiral bo lahko med okvirjem, besedilom, povezavo ter multimedijskima vsebinama slike in videa. Ko bo uporabnik izbral vrednost, ki jo bo želel vključiti na spletno stran, se bo nato ob kliku na ustrezen gumb dodajanja v vsebino spletne strani dodal element z naključno zgeneriranimi lastnostmi, ki jih bo lahko nato uporabnik poljubno urejal. Gumb za dodajanje elementov na spletno stran bo v ozadju klical namensko funkcijo, ki bo preverila klicoči element in poskrbela, da se bo ustrezen element pojavil na spletni strani.

Uporabniku mora biti na voljo funkcionalnost, s katero bo lahko spreminjal višinsko lego elementa. Lega bo določena z atributom `z-index` in bo ob inicializaciji uvožene spletne strani dinamično zgenerirana. Ob vsakem novem dodanem elementu bo imel element najvišjo vrednost višinske pozicije, ki jo bo lahko uporabnik ročno urejal in prilagajal svojim potrebam ali željam. Uporabnik bo imel na voljo namenske gumbe, s katerimi bo imel možnost postavitve elementa na dno ali vrh spletne strani. Prav tako bo imel možnost izbire postavitve elementa za eno stopnjo nižje ali višje glede na akcijo klika ustreznega gumba. V kolikor bo želel premakniti izbran element v ozadje spletne strani, bo moralo orodje poiskati najnižjo vrednost atributa `z-index` ter ga dodeliti izbranemu elementu uporabnika. Na podlagi te informacije bo lahko nato orodje prilagodilo vrednost atributa `z-index` gradnika, ki ga bo želel uporabnik postaviti na dno spletne strani. S podobno implementacijo bo lahko uporabnik dosegel postavitev elementa na najvišji nivo med gradniki orodja, s tem da bo namesto najmanjše vrednosti atributa `z-index` uporabil najvišjo najdeno. Funkcionalnost premikanja posameznega atributa, z uporabo akcije premikanja na višji ali nižji nivo, bo implementirana na podlagi vrednosti trenutno izbranega gradnika, v kolikor se mu bo prištel potreben odmik glede na elementove atribute v seznamu vseh elementov. Ob dodajanju elementov na spletno stran ali uvozu spletne strani v brskalnik bo orodje dinamično dodelilo vrednosti atributa `z-index` vsem elementom z določenim odmikom, ki se bo nato uporabljal pri inicializaciji in postavitvi elementov na spletno stran, z uporabo funkcionalnosti premikanja elementa na nivo višje ali nižje. Postavitev statičnih elementov stranske plošče orodja je prikazana na Sliki 4.6. Vse omenjene funkcionalnosti premikanja elementov na različne nivoje bodo imele predpogoj, da v kolikor bo želel uporabnik spremeniti višinsko pozicijo elementa, bo moral biti gradnik izbran v polju vsebine. Če gradnik ne bo izbran, se spremembe nad višinsko pozicijo atributa ne bodo spremenile.

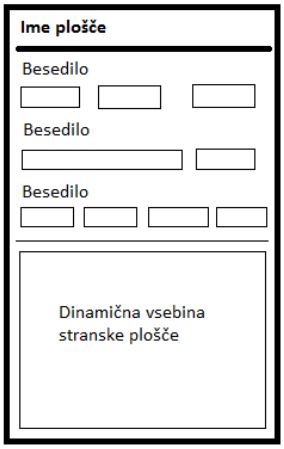
Pod rezerviranim prostorom za implementacijo funkcionalnosti, ki je skupna vsem gradnikom, bo orodje vsebovalo prostor, kjer se bo dinamično



Slika 4.6: Postavitev skupnih elementov stranske plošče orodja.

prikazovala možna vsebina atributov, ki jih bo lahko uporabnik urejal glede na izbran gradnik v vsebini urejane spletne strani. Ob kliku na gradnik v polju vsebine se bo uporabniku prikazal okvir z vsebovanimi atributi, ki jih bo lahko urejal v trenutnem kontekstu gradnika spletne strani. Orodje bo implementirano tako, da se bo za vsak gradnik, ki ga bo lahko uporabnik izbral s seznama možnih atributov dodajanja na spletno stran, ustvaril okvir z vsebovanimi urejanimi atributi. Ob izbiri gradnika se bo okvir prikazal, medtem ko se bo prejšnje izbrani okvir uporabniku skril. Do trenutka, ko uporabnik ne bo izbral nobenega elementa iz vsebine spletne strani, se mu ne bo prikazal noben okvir. Ob izbiri gradnika na spletni strani bo moralo orodje nato prikazati ustrezen okvir, v katerega bodo vnesene trenutne vrednosti gradnika, ki ga bo uporabnik izbral. Ob morebitnem urejanju gradnika bo moral imeti uporabnik na voljo gumb, s katerim bo potrdil spremembe nad izbranim gradnikom. Vrednosti se bodo dinamično spreminjale glede na interakcijo uporabnika z gradnikom v polju spletne vsebine. Vsak gradnik bo imel privzeto možnost urejanja dolžine in širine ter levi in zgornji odmik gradnika glede na pozicijo v vsebini spletne strani. Poleg privzetih atributov bo imel vsak element svoje dodatne tipične atribute, ki jih bo lahko uporabnik spreminjal. Tako bo lahko uporabnik gradniku pravokotnik spreminjal barvo ozadja, kjer bo lahko med vnaprej definiranimi barvami izbral želeno vrednost. Gradniku besedilo bo lahko uporabnik spreminjal barvo, stil, velikost ter težo. Uporabnik bo lahko v namenskem polju določil tudi samo

besedilo, ki bo nato prikazano v elementu besedila. Gradnik spletne povezave bo uporabniku omogočal urejanje imena povezave, mesto navigacije povezave ter mesto, kjer bo želel uporabnik odpreti želeno spletno stran. Omogočeno mu bo odpiranje spletne strani v trenutnem, novem in predhodnem oknu. Orodje bo moralo prav tako poskrbeti, da ne bo uporabnika ob kliku na spletno povezavo med urejanjem spletne strani preusmerilo na želeno spletno stran. Prav tako bo moralo orodje poskrbeti, da bo ob izvozu in uvozu spletne strani pravilno nastavilo vrednosti izklapljanja ali vklapljanja možnosti uporabe spletne povezave oziroma gradnika, ki ga bo uporabnik urejal. Poleg atributov urejanja bo imel uporabnik možnost ob kliku na gumb vse spremembe uveljaviti v gradniku spletne povezave. Ob izbiri gradnikov multimedijske vsebine spletne strani bo imel na voljo urejanje atributov dolžine in širine gradnika ter vir, kjer se bo želena prikazana vsebina na spletni strani nahaja. Vir se bo podal kot absolutna pot, pogojena z dejstvom, da bo želena vsebina že obstajala nekje na svetovnem spletu.



Slika 4.7: Postavitev elementov stranske plošče orodja.

Uporabnik bo imel na dnu plošče na voljo tudi možnost vklapljanja ali izklapljanja animacije skrivanja in prikazovanja plošče. V kolikor bo imel vključeno animacijo, se mu bo plošča skrila, ko jo uporabnik ne bo prekril z miško, ter prikazala, ko bo uporabnik prekril ploščo z miško. Implemen-

tacija funkcionalnosti bo zaradi boljše preglednosti in večje delovne površine primerna za uporabnike z manjšimi resolucijami monitorjev. Funkcionalnost bo implementirana na način, da bo uporabniku na voljo izklopno stikalo, s katerim se bo ob kliku vklopila ali izklopila animiranost plošče. Če bo imel uporabnik izbrano vrednost skrivanja plošče, se mu bo plošča ob neaktivnosti skrila, vendar bo določena dolžina roba ostala vidna, da bo lahko uporabnik ob naslednji interakciji s ploščo izbral atribute in vrednosti, ki jih bo želel urejati. V kolikor bo imel uporabnik onemogočeno animacijo skrivanja plošče, bo njena postavitev statična. Vrednost se bo ob prvi interakciji uporabnika s stikalom shranila v lokalno shrambo in bo uporabniku na voljo ob vsaki prijavi v orodje. Celotna vsebina končne postavitve elementov je prikazana na Sliki 4.7.

Polje vsebine spletne strani bo predstavljalo osrednjo funkcionalnost orodja, kjer bo imel uporabnik možnost grafičnega urejanja elementov. Na voljo mu bodo funkcije premikanja elementov po delovni površini, spreminjanja velikosti ter interakcijo povezanosti funkcionalnosti s ploščo atributov izbranega gradnika v polju. Orodje bo moralo v inicializacijskem postopku uvoza ali izdelave nove spletne strani poskrbeti, da bodo vsi gradniki pripravljeni za premikanje in spreminjanje velikosti ter ustrezno pripravljeni za urejanje. Tako bo moralo orodje ob inicializaciji poskrbeti, da bo vsem gradnikom dodalo funkcionalnosti iz knjižnice jQuery-UI, in sicer na način, da bo nad vsemi elementi vsebine poklicalo funkciji `.resizable()` in `.draggable()`, s katerima bo uporabniku omogočilo napredno urejanje spletne strani. Ob izvozu spletne strani bo moralo orodje poskrbeti, da bo vsebini spletne strani odstranilo omenjeni funkcionalnosti ter poskrbeti, da bo lahko izdelana spletna stran polno funkcionalno delovala na spletnem strežniku brez odvečnih delov spletne strani, ki jih pri prikazu ne bo potrebovala.

4.3.2 Načrtovanje uporabniškega vmesnika orodja

Naslednji večji sklop pri razvijanju orodja bo zasnova uporabniškega vmesnika. Ta del bo ključnega pomena in bo definiral izgled, postavitev in obliko

samega orodja [3]. Podrobneje bo opisoval, kako se morajo animacije odražati ter kakšna je njihova povezava z ostalimi gradniki in implementacijami animacij. Uspešnost dobrega načrta orodja temelji predvsem na dejstvu, da lahko s pomočjo dobrega grafičnega načrta orodja dosežemo enostavno uporabo, kljub njegovi morebitni zahtevni implementaciji. Tako bo na podlagi specifikacije zahtev orodja nastal načrt, ki bo natančno opisoval izgled uporabniškega vmesnika orodja.

Zaglavje orodja bo vsebovalo implementacijo barvnega prehoda iz sive v belo barvo. Celotna implementacija orodja WebsiteEdit bo osnovana na izgledu in podpori tehnologije CSS3 ter bo vključevala napredne filtre in gradiente za popolnejši in moderniziran videz orodja. Zaglavje orodja se bo po dolžini prilegalo širini delovne površine spletnega brskalnika ter bo imelo fiksno višino petdesetih pikslov, kot je razvidno s Slike 4.8. V spodnjem delu bo orodje vključevalo rdečo črto ter senčenje. Implementacijo gradienta ozadja spletnega orodja bomo lahko dosegli z uporabo predefiniranega atributa, poimenovanega `-moz-linear-gradient`, ki mu bomo s parametri podali širino, smer ter začetno in končno barvo, ki jo bomo želeli uporabiti pri implementaciji barve na spletni strani. Senčenje je v novejših brskalnikih podprto z dodanim atributom `box-shaddow`. V zaglavju bo orodje vsebovalo naslov, njegova postavitev se bo prilegala spodnji poziciji rdeče črte z odmikom od leve strani delovne površine ekrana. Na desno stran bosta vzporedno z velikostjo imena orodja dodana atraktivna gumba osivene barve, ki bosta jasno ponazarjala njuno funkcionalnost. Ob uporabnikovem prekritju miške z gumbom se bo gumb obarval v rdečo barvo, v primeru izhoda miške iz polja gumbov pa se bo obarval nazaj v sivo barvo.

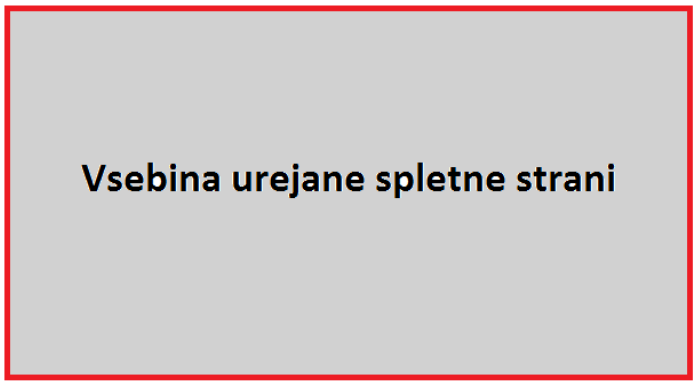
Slika 4.8: Grafična oblika elementov zaglavja.

Plošča z atributi izbranega gradnika bo pripeta na levo stran delovne površine ekrana in bo vsebovala statično osiveno barvo ozadja brez prelivanja barv. Vsi gumbi in elementi, ki se bodo dinamično prilagajali na spletno stran, bodo imeli privzeto črno barvo ozadja z rdečo obrobo z implementacijo senčenja ter belim napisom elementa. Posamezna funkcionalno podobna skupina gumbov bo združena v skupine, kjer bomo z rdečim napisom na spletni strani povedali, kateri funkcionalnosti pripada določena skupina elementov. Grafični izgled vsebine stranske plošče orodja WebsiteEdit je razviden s Slike 4.9. Tako bomo imeli gumbe za brisanje, kopiranje in dodajanje strnjene na vrhu pod ločnico imena v eno vrsto ter medsebojno ločene s praznim prostorom, poimenovane `Element:`. V naslednji vrsti bo orodje uporabniku ponujalo možnost dodajanja elementov na spletno stran s poimenovanim naslovom `Add:`. V možnost urejanja višinske pozicije elementa pa bo uporabniku na voljo skupina gumbov, ki bodo prisotni na spodnji strani ločnice in bodo predstavljeni z imenom `Move:`. Naslov plošče bo zapisan nad elementi ter obarvan rdeče. V polju dinamičnega prilagajanja elementov se bodo uporabniku prikazovali elementi, poravnani v tabeli, ki bodo imeli statično dolžino. V kolikor bo manjše število elementov, kot bo namenjenega prostora v plošči, se bodo atributi poravnali glede na vertikalni položaj, in sicer vedno proti vrhu. Če bo imel uporabnik vključeno možnost skrivanja plošče, se mu bo ta animirala v dolžini ene sekunde ter dinamično prilagajala pozicijo spletne strani orodja. Dolžina in širina plošče bosta statični in se ne bosta spreminjali niti v primeru, ko bo imel uporabnik vključeno animacijo plošče.

Vsebina spletne strani bo postavljena v osrednji del orodja in bo predstavljala delovno površino orodja. Višina spletne strani bo fiksne dolžine, medtem ko se bo širina prilagajala glede na delovno površino spletnega brskalnika. Uporabnik bo lahko z vključitvijo animacije povečal velikost delovne površine in s tem pridobil na velikosti urejane spletne strani. Privzeta barva ozadja delovne površine bo svetlo sive barve, z dodano rdečo obrobo delovne površine in senčenjem, kot je prikazano na Sliki 4.10. Izbranemu elementu

Slika 4.9: Grafična oblika elementov stranske plošče.

v vsebini spletne strani bo dodano senčenje ob njegovem robu. Uporabniku bo dovoljeno elemente premikati samo znotraj vsebine spletne strani.



Slika 4.10: Grafična oblika elementov vsebine urejane spletne strani.

4.3.3 Načrt hranjenja podatkov v spletni shrambi

Celoten načrt hranjenja podatkov se močno razlikuje glede na implementacijo hranjenja podatkov v ostalih znanih sistemih za upravljanje vsebin.

Na podlagi specifikacije in zmožnosti bo hranjenje podatkov implementirano pri samem odjemalcu, kar ponuja številne dodatne možnosti. Uporabnik bo lahko uporabljal bistveno hitrejše pomnenje podatkov, manjšo obremenitev spletne povezave in pretoka podatkov ter, kar je najpomembnejše, ne bo mu potrebo skrbeti za postavitev namenskih strežnikov za hranjenje podatkov. Orodje bo uporabniku ponujalo možnost samodejnega shranjevanja podatkov v pomnilniku brskalnika, kjer mu bodo podatki na voljo tudi v primeru, ko bo uporabnik zaprl spletni brskalnik. Vsebina spletne strani se bo shranjevala v petnajstsekundnem intervalu, ki ga ne bo moč spreminjati ter bo uporabniku omogočila naložitev podatkov v vsebino spletne strani ob zagonu orodja. Orodje bo uporabniku ponujalo tudi možnost shranjevanja nastavitve stanja animacije plošče.

4.4 Implementacija orodja

Vsaka aplikacija vsebuje točko v razvojnem ciklu, ko se posamezni moduli in grafični uporabniški vmesnik implementirajo v delujočo aplikacijo. Zahtevnost izdelave aplikacije je odvisna od načina in kvalitete podanega načrta aplikacije, ki jo želimo izdelati [3]. Zato je zelo pomembno, da pred začetkom implementacije aplikacije poskrbimo, da izdelamo kvaliteten načrt aplikacije, ki mu nato sledimo ob razvoju aplikacije.

Razvoj aplikacije lahko poteka na več različnih načinov. Najbolj uporaben in razširjen je način implementacije aplikacije z inkrementalnim razvojem, kjer razvijalec aplikacijo razvija po delih [3]. Podobnega prijema se bomo lotili pri implementaciji spletnega orodja za podporo izdelave spletne strani. Probleme smo razdelili na več podproblemov, ki jih bomo nato posamično implementirali v naše orodje.

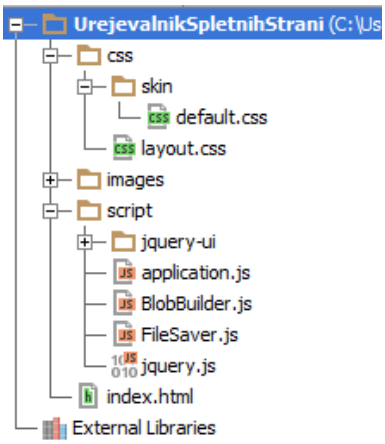
4.4.1 Ogrodje in postavitev orodja

V prvem delu razvoja spletnega orodja smo najprej zasnovali vse potrebne module in okvirno izdelali uporabniški vmesnik orodja. V načrtu orodja je

jasno razvidno, kako so posamezni moduli pozicionirani, kakšna je njihova postavitev ter kakšno je njihovo obnašanje. V ta namen bomo v prvem koraku izdelali spletno stran in ji sprva definirali vse potrebne atribute v zaglavju spletne strani, da bo lahko spletni brskalnik določil razvoj spletne strani v označevalnem jeziku HTML5. Slednje definiramo v zaglavju orodja z določilom `<!DOCTYPE html>`. Na ta način povemo prevajalniku spletnih strani interpretacijo spletne strani kot označevalni jezik HTML5. Definiramo tudi metapodatke ter lokalno in strukturalno ustvarimo datoteke na lokalnem disku, v katerih bomo shranjevali podatke o obliki, postavitvi in funkcionalnosti. Odločili smo se, da bomo grafični uporabniški vmesnik ločili v dve namenski datoteki, v kateri bomo ločeno vključili postavitev elementov spletne strani, ter v drugo sam grafični izgled vseh elementov na spletni strani. Obe datoteki smo vključili v ustvarjeno mapo `css`, v kateri bodo vsebovane vse datoteke, ki se bodo navezovale na sam izgled spletne strani. Prednost uporabe ločene implementacije grafičnega in postavitvenega dela je v tem, da lahko definiramo več različnih preoblek in uporabimo eno postavitveno datoteko. Ni nam potrebno skrbeti za morebitno podvajanje podatkov, ker je grafični in postavitveni del orodja ločen. Podobno ustvarimo tudi mapo z imenom `scripts`, v kateri bomo hranili datoteke, ki jih bo naše orodje uporabljalo pri svojem delu. V mapo smo shranili datoteke, ki smo jih ustvarili sami ter knjižnice, ki jih bomo uporabili pri sami implementaciji orodja. Na tem mestu bodo tako shranjene knjižnice, kot so jQuery, jQuery-UI, FileSaver ter BlobBuilder. Ko imamo pripravljeno okvirno strukturo datotečnega sistema, poiščemo vse zgoraj omenjene knjižnice na spletu in jih vključimo v ustrezne mape. Podrobnejša drevesna struktura datotek je prikazana na Sliki 4.11.

Nato na našo predhodno ustvarjeno spletno stran dodamo potrebno zaglavje ter ustrezno vključimo vse datoteke na spletno stran. Na ta način si pripravimo okolje, v katerem bomo lahko postopoma razvijali spletno stran.

Ko imamo spletno stran pripravljeno, ji lahko dodamo polja, v katerih bomo hranili vsebine posameznih okvirjev, opisanih v načrtu orodja. V ta



Slika 4.11: Implementirana struktura postavitve orodja WebsiteEdit.

namen ustvarimo tri gradnike tipa `div` ter jim konceptualno podamo ime v atributu `id`. Na te elemente se bomo v prihodnosti sklicevali, v kolikor bomo želeli v skriptnem jeziku poseči v funkcionalnost posameznih elementov.

Zaradi statične postavitve in enostavnosti implementacije zaglavja bomo vse potrebno implementirali že v začetku razvoja orodja. V namensko datoteko za upravljanje postavitev orodja podamo atribute, v katerih specificiramo postavitev imena orodja. Ime v orodju zasidramo na levo stran delovne površine ekrana, medtem ko na desno stran vključimo dve polji. V polji bomo v prihodnje implementirali funkcionalnosti za uvoz in izvoz podatkov. Ko imamo slednje pripravljeno, se lahko lotimo implementacije ozadnje kode gumbov v zaglavju orodja. Elementoma dodamo sklic na namenski funkciji, ki jima bomo v bodoče implementirali funkcionalnost uvoza in izvoza podatkov. Ob spodnjem robu zaglavja dodamo orodju še predvideno črto iz načrta.

4.4.2 Stranska plošča orodja

V prejšnji iteraciji razvoja spletnega orodja smo dodali vse potrebne okvirje, v katere bomo vključili ustrezne dodatne gradnike za pravilen prikaz spletne strani. V načrtu orodja je podrobneje predstavljena postavitev elementov

v stranski plošči, ki bo uporabniku služila kot vstopna točka za urejanje vseh gradnikov v vsebini spletne strani. Implementacija načrta orodja sledi dodajanju imena plošče v zgornjem delu orodja, kjer pod naslov dodamo ločnico, ki bo ločevala polje z imenom plošče, od polja, kjer bodo prisotni elementi, ki so skupni vsem gradnikom urejanja na spletni strani.

Na podlagi načrta orodja dodamo na spletno stran ustrezne gumbe ter jih medsebojno ločimo glede na opisan načrt. V zgornji del okvirja dodamo osem gumbov, ki jih konceptno združimo glede na načrt orodja in jim podamo imena. Na spletno stran vključimo tudi izbirni seznam, kjer bo imel uporabnik na voljo izbire gradnika, ki ga bo želel vključiti na spletno stran. Ob dodajanju gradnikov na spletno stran uporabimo tabelo, v kateri razvrstimo elemente glede na podano pozicijo na spletni strani. Ob koncu tabele dodamo tudi ločnico, ki bo mejila statični del od dinamičnega.

Nato se lotimo razvoja dinamičnega dela stranske plošče, v katerem implementiramo pet okvirjev, v katerih bomo hranili vrednosti za predstavitev podatkov posameznega izbranega gradnika. Ustvarimo ustrezno število okvirjev značk tipa `div` ter jih vključimo na spletno stran. Vsem elementom dodamo atribut `hidden`, s katerim jim povemo, da so skriti. Vse elemente nato ustrezno poimenujemo oziroma jim podamo ustrezna imena za atribute `id`. Nato v vsebino vsakega okvirja ustrezno dodamo še gumbe in polja, ki smo jih v načrtu orodja ustrezno definirali za predstavitev gradnikovih atributov izbranega elementa. Vse elemente v okvirjih spletne strani ustrezno pozicioniramo in prilagodimo glede na načrt orodja.

4.4.3 Polje vsebine urejanja orodja

Implementacija funkcionalnosti v polju urejanja vsebine spletne strani je bila v prvi iteraciji implementirana do nivoja, kjer je bil postavljen okvir, v katerega bo vključena vsebina. Polje urejanja vsebine spletne strani se mora glede na načrt orodja dinamično prilagajati delovni površini spletnega brskalnika. Funkcionalnost prilagajanja v orodju implementiramo na način, da se ob polni naložitvi spletne strani izvede klic funkcije, ki bo poskrbela,

da bo primerno izračunala dolžine okvirja, ter nastavila potrebne vrednosti okvirju. Tako dodamo znački `<body>` atribut `onLoad`, v katerem podamo ime klicoče funkcije. Klic bo predstavljal našo izhodiščno točko, kjer bomo klicali funkcije za prilagajanje uporabniškega vmesnika orodja.

Implementacijo prilagajanja dolžine okvirja lahko dosežemo tako, da vrednosti dolžine delovne površine ekrana odštejemo dolžino stranske plošče in odmik levega in desnega roba. Novo izračunano vrednost nato nastavimo okvirju vsebine ter tako prilagodimo polje. Višina polja je fiksna in je enaka višini stranske plošče.

4.4.4 Implementacija funkcionalnosti orodja

Funkcionalnost in postavitev orodja smo pripravili do nivoja, kjer lahko začnemo z implementacijo posameznih funkcionalnosti v modulih. Celotne implementacije funkcionalnosti se bomo lotili ločeno ter jo razvijali po delih.

Sprva se bomo lotili razvoja stranske plošče orodja. Pred razvojem smo se odločili, da bomo trenutno izbrani element v vsebini spletne strani hranili v spremenljivki, v kateri bo shranjen celoten objekt. Prednost slednjega je predvsem v tem, da so nam na voljo vsi atributi, ki jih ima želen gradnik. Začnemo z implementacijo brisanja elementov s spletne strani. V kolikor obstaja vrednost v namenski spremenljivki za hranjenje trenutno izbranega objekta, nad spremenljivko pokličemo metodo `.remove()`, ki bo poskrbela, da bo izbran element odstranjen s spletne strani. Funkcionalnost kopiranja lahko implementiramo na podoben način, za razliko od prejšnje vrednosti imamo podano novo spremenljivko, v kateri bo hranjen želen kopiran objekt. Ob kliku na ustrezen gumb nato funkcija preveri, ali ima uporabnik določeno vrednost spremenljivke trenutno izbranega elementa. V kolikor je pogoj izpolnjen, se vrednost podatkov podvoji in shrani v namensko funkcijo kopiranja. Ko uporabnik izbere možnost lepljenja, se vrednost vsebine spremenljivke kopiranja podvoji in ponovno vstavi na spletno stran s spremenjenim atributom `id` elementa.

Uporabnik ima tudi možnost dodajanja vnaprej določenega elementa. Ob

kliku na gumb dodajanja preberemo vrednost trenutno izbranega elementa v seznamu vseh možnih elementov ter dinamično zgeneriramo element s poljubno barvo ter vnaprej določenimi vrednostmi gradnika. Gradniku, ki ga želimo dodati na spletno stran, podamo vrednost kot argument funkcije za dodajanje elementa na spletno stran, s katero bo lahko naše orodje določilo, kateri je želeni atribut, ki ga želimo vključiti na spletno stran. Želeno vrednost lahko vključimo na spletno stran z metodo `.append()`, ki jo pokličemo nad objektom vsebine spletne strani. Kot parameter podamo metodi vrednost v obliki HTML, ki jo želimo dodati na spletno stran. Ob uspešno dodanem elementu v vsebino spletne strani pokličemo še namensko funkcijo, ki poskrbi, da se dodanemu elementu dodajo funkcionalnosti premikanja in spreminjanja velikosti elementa.

Na podoben način kot smo dodali funkcionalnosti dodajanja, brisanja ter kopiranja elementa, lahko dodamo tudi funkcionalnost spreminjanja višinske pozicije elementa. Ob kliku na ustrezen gumb se pokliče funkcija, ki ji kot argument podamo smer premika elementa. V funkciji nato pregledamo, v katero smer želi uporabnik spremeniti višinsko pozicijo ter na podlagi teh podatkov ustrezno popravimo elementov atribut `z-index`. V kolikor želi uporabnik postaviti gradnik na dno vsebine spletne strani, v funkciji poiščemo element z najnižjo vrednostjo atributa `z-index` ter dobljeni vrednosti odštejemo 5 točk. Podobno storimo, če želi uporabnik postaviti gradnik na vrh vsebine spletne strani. V tem primeru poiščemo element z najvišjo vrednostjo ter mu prištejemo 5 točk. Če želi uporabnik spremeniti višinsko pozicijo za eno vrednost višje ali nižje, elementu odštejemo ali prištejemo 5 točk glede na trenutno vrednost atributa `z-index`.

Ob koncu razvoja privzetih funkcionalnosti vseh gradnikov preverimo, ali funkcionalnosti delujejo v skladu z načrtom orodja ter odpravimo napake, če so prisotne.

Nadaljnjega razvoja orodja se bomo lotili v implementaciji funkcionalnosti prikaza možnih urejanih atributov, ko uporabnik izbere gradnik v polju vrednosti. Ob kliku na gumb se uporabniku prikaže okvir s trenutnimi vre-

dnostmi gradnika. Slednje implementiramo tako, da si pripravimo namensko funkcijo, katere implementacijo bomo dokončali ob razvoju implementacije vsebine urejane spletne strani. V namenski funkciji lahko ročno ustvarimo klic funkcije s poljubnim parametrom, da bomo lahko našo funkcionalnost implementirali v celoti. V funkciji, ki bo procesirala prikazovanje in formatiranje podatkov v dinamičnem polju, preko parametra posredujemo objekt izbranega elementa. Sprva preverimo, ali morebiti izbrani element ni prazen, ter, v kolikor je pogoj ustrezen, nadaljujemo z izvajanjem procesiranja orodja. V prvem delu skrijemo vse okvirje za posamezne gradnike ter na podlagi atributa `id` izbranega elementa ugotovimo tip objekta, ki je izbran v polju vsebine. Na podlagi te informacije prikažemo polje vrednosti izbranega atributa ter nato dinamično za vse vnaprej definirane atribute preberemo vrednosti iz izbranega objekta in jih vključimo v vnaprej definirana polja, ki smo jih vključili na spletno stran v eni od prejšnjih iteracij razvoja spletne strani. V kolikor smo implementirali vse veljavne gradnike, lahko slednjo funkcionalnost preverimo tako, da ročno nastavljamo objekte posameznih tipov ter preverimo pravilnost prikaza okvirjev. Ob koncu implementacije funkcionalnosti okvirjev dodamo vsakemu okvirju še možnost posodabljanja spremenjenih vrednosti, kar pa storimo tako, da vse trenutne vrednosti iz vnaprej definiranih polj v okvirju preberemo ter jih ob klicu na ustrezno namensko funkcijo dodamo v objekt. Spremembe bodo v vsebini spletnega orodja vidne v trenutku, tako da jih bo lahko uporabnik spremljal v realnem času.

Na dnu vsebine stranske plošče implementiramo še funkcionalnost, ki bo uporabniku omogočala animiranje stranske plošče. Klic funkcije pokličemo v trenutku, ko uporabnik spremeni vrednost izključujočega stikala ter na podlagi podatka, ali je bilo stikalo vključeno ali izključeno, dodamo možnost animiranja spletne strani ali jo odstranimo. Sama implementacija animacije stranske plošče je implementirana s pomočjo knjižnice jQuery, v kateri s pomočjo funkcije `.animate()` podamo atribut polja plošče, ki ga želimo animirati. V našem primeru bomo animirali vrednost atributa `x`, kjer mu bomo

spreminjali njegovo vrednost. V atributu funkcije `.animate()` podamo še čas ter interval, od kjer želimo animirati vrednost atributa `x`. Pred začetkom izvajanja animacije preverimo še pogoj, ali ima uporabnik animacijo vključeno. Če ima animacijo onemogočeno, nastavimo stranski plošči statično postavitev ter prilagodimo velikost polja vsebine urejane spletne strani. Če ima uporabnik animacijo vključeno, poskrbimo, da se polje vsebine spletne strani prilagodi širini delovne površine spletnega brskalnika z definiranim odmikom levega roba strani.

Implementacija funkcionalnosti stranske plošče je zaključena, nakar se lotimo razvoja podpore za omogočanje grafičnega urejanja postavitve in spreminjanja velikosti elementa v vsebini urejane spletne strani. Na podlagi načrta spletnega orodja moramo ob vsakem dodajanju ali odstranjevanju gradnika iz vsebine spletne strani poskrbeti, da imajo atributi pripravljene ustrezne metode, s katerimi bo uporabniku omogočeno grafično urejanje spletne strani. Tako moramo ob vsakem uvozu ali izvozu spletne strani poskrbeti, da se nad vsemi elementi v vsebini spletne strani izvede klic funkcij `.draggable()` in `.resizable()`, ki bosta poskrbeli, da bodo imeli gradniki možnost spreminjanja postavitve in razširjanja njihove velikosti. Omenjeni funkciji sta implementaciji funkcionalnosti v knjižnici jQuery in jih lahko kličemo nad vsemi objekti, ki so predstavljeni kot objekt tipa jQuery. Ko poskrbimo, da orodje ustrezno doda omenjene funkcionalnosti vsebini urejane spletne strani, implementiramo funkcije, ki smo jih definirali v eni od prejšnjih iteracij. V klicu funkcije odstranjevanja vseh funkcionalnosti gradnikov vsebine spletne strani poskrbimo, da se vse spremembe, ki so bile dodane spletnim gradnikom ob uvozu ali urejanju spletne strani in ki niso del vsebine predstavitvene spletne strani, odstranijo. Prav tako poskrbimo, da se vse enake vrednosti, ki so bile definirane nad gradniki, ponovno povrnejo v stanje, kot je bilo pred samim urejanjem spletne strani. V implementaciji funkcionalnosti spletne strani moramo tudi poskrbeti, da se bo vsakemu gradniku ob kliku nanj izvedla določena akcija. V našem primeru moramo trenutno izbranemu elementu dodati sence. V ta namen v naši stil-

ski datoteki dodamo selektor, ki bo dodal sence izbranemu elementu. Ob kliku na gumb nato s pomočjo klica funkcije `.css()` nad trenutno izbranim objektom dodamo vrednost selektorja trenutno izbranemu elementu ter po potrebi odstranimo vse prejšnje označene elemente. Ob kliku na funkcijo izbiranja poskrbimo tudi, da se izvede funkcija, ki smo jo definirali ob razvijanju funkcionalnosti implementacije stranske plošče, ki bo kot argument podala trenutno izbrani gradnik v vsebini spletne strani ter v dinamičnem delu spletne strani pravilno pokazala ustrezno vsebino atributov gradnika.

4.4.5 Hranjenje podatkov v spletni shrambi

Na mestu implementacije funkcionalnosti v vsebini urejane spletne strani se razvoj naših funkcionalnosti bliža koncu. Orodje mora omogočati ustrezno shranjevanje podatkov v spletno shrambo. Poskrbeti moramo, da se ustrezno shranjuje vrednost vključene ali izključene animacije ter da se vsakih petnajst sekund samodejno shrani vrednost vsebine spletne strani. Implementacijo funkcionalnosti shranjevanja nastavitve lahko implementiramo na način, da v spletno shrambo shranimo pod ključem `animation` vrednost vključenega ali izključenega stikala animacije. V orodju se lahko nato enostavno sklicujemo in preverjamo, kakšno je stanje implementacije animacije. V kolikor želimo implementirati funkcionalnost shranjevanja vsebine urejane spletne strani, lahko s pomočjo klica funkcije `setInterval()` poskrbimo, da se bo vsakih petnajst sekund izvedla funkcija, ki nam bo vsebino urejane spletne strani shranila v spletno shrambo. Slednje lahko implementiramo tako, da v metodi `setInterval(funkcija_shranjevanja(), 15000);` podamo dva argumenta. Prvi predstavlja funkcijo, ki se bo izvedla vsakih petnajst sekund. Vrednost petnajstih sekund je definirana kot drugi parameter in predstavlja vrednost petnajstih sekund, zapisanih v milisekundah. Poskrbeti moramo še, da se klic funkcije za shranjevanje izvede vedno na začetku zagona orodja za urejanje spletne strani ter da se funkcija preverja in shranjuje le v primeru, ko vsebina urejane spletne strani ni prazna.

4.4.6 Izvoz in uvoz vsebine generirane spletne strani

Implementacija uvoza in izvoza spletne strani je predstavljena kot ločena funkcionalnost in je deloma implementirana v zaglavju orodja. Poskrbeti moramo še, da bo naše orodje ponujalo ustrezen izvoz in uvoz podatkov.

V eni od prejšnjih iteracij razvoja spletne strani smo pripravili namensko funkcijo, v kateri bomo implementirali uvoz in izvoz spletne strani v vsebino orodja. V prvem delu se bomo lotili implementacije branja obstoječe datoteke zapisa označevalnega jezika v vsebino spletnega brskalnika. S pomočjo podprte knjižnice `FileReader` s strani spletnega brskalnika Mozilla Firefox lahko datoteko preberemo in jo uvozimo v spletni brskalnik. Iz vrednosti značke, kjer orodje hrani datoteko, ki jo je uporabnik izbral, podamo kot argument objektu `FileReader` ter s pomočjo klica funkcije `.readAsText()` preberemo znakovno vrednost datoteke. V funkciji poskrbimo tudi za vso potrebno preverjanje robnih pogojev ter ob uspešni vključitvi vrednosti prebrane datoteke izvedemo namensko funkcijo, ki bo poskrbela, da bo pripravila vse vrednosti v polju vsebine, da bo lahko uporabnik grafično spreminjal postavitve in velikost elementov. Klic začetka branja nove datoteke izvedemo v trenutku, ko uporabnik izbere novo različico datoteke z lokalnega diska. Funkcionalnost stestiramo ter preverimo, ali se vsebina spletne strani pravilno prikaže v polju vsebine spletne strani.

Ko imamo implementiran uvoz spletne strani, se lahko lotimo implementacije izvoza spletne strani. V samem izvozu spletne strani moramo poskrbeti, da v kolikor je vsebina polja urejane spletne strani prazna, da se omenjena funkcija ne izvede. Poiščemo namensko izbrano funkcijo za izvoz spletne strani ter pričnemo z implementacijo. V prvem delu implementacije zapišemo funkcijo, ki trenutno ne naredi nič, v bodoče pa bo poskrbela, da bo vse potrebne podatke pripravila za izvoz, kar pomeni, da bo odstranila vso nepotrebno vsebino iz polja vsebine spletne strani. Prav tako vključimo funkcijo, ki bo ob zaključku izvoza spletne strani poskrbela, da bo podatke pripravila v stanje, kot je bilo pred izvozom spletne strani. Nato pričnemo z implementacijo funkcionalnosti izvoza spletne strani. Uporabili

bomo namenski knjižnici BlobBuilder ter FileSaver. S pomočjo knjižnice `BlobBuilder` zgeneriramo objekt tipa blob, ki v praksi predstavlja skladišče znakov, ki jih želimo zapisati v določeno datoteko. Pred izvozom spletne strani poskrbimo, da v zaglavje dodamo potrebne značke HTML za pravilno formatiranje naše spletne strani. Ko na spletno stran vključimo značko `<body>`, lahko preberemo vsebino urejane spletne strani ter jo dodamo na koncu objekta blob. Ob koncu še zaključimo vse značke označevalnega jezika HTML, ki smo jih dodali pred začetkom branja vsebine. Objekt je pripravljen za izvoz, zato lahko nato s pomočjo integrirane funkcije brskalnika `saveAs` shranimo vrednost objekta blob. Pred tem uporabnika povprašamo še o želenem imenu datoteke, ki ga nato kot drugi argument podamo funkciji za shranitev podatkov. Podatki se bodo potem pretočili do uporabnika, kjer bo lahko uporabnik lokalno shranil dokument. Funkcionalnosti ob koncu implementacije testiramo, da preverimo, ali funkcionalnosti dosegajo zapisane točke v načrtu orodja WebsiteEdit.

4.4.7 Izdelava uporabniškega vmesnika orodja

Z implementacijo uvoza in izvoza vsebine spletne strani smo zaključili z implementacijo funkcionalnosti v spletnem orodju. Preostala nam je samo ureditev grafičnega izgleda orodja, kjer moramo poskrbeti, da se nad določenimi elementi uporabijo isti slogi ter zagotoviti kriterije načrta orodja. Vse selektorje, ki jih želimo uporabiti v našem orodju, bomo vključili v namensko datoteko, ki nam služi za opis izgleda uporabniškega vmesnika orodja.

Po načrtu orodja dodamo zaglavju ustrezen prehod barv ozadja, kjer definiramo prehod iz sive barve v belo. Podobno ustvarimo tudi selektor, ki bo vseboval senčenje ter rdečo barvo obrobe. Nato dodamo črtam rdeče obrobe, ki so definirane v načrtu orodja. Barvo besedila nastavimo na privzeto rdečo barvo, ki se bo kot privzeta vrednost uporabila na vseh mestih v brskalniku. Po končani implementaciji barv v zaglavje spletnega orodja WebsiteEdit dobimo uporabniški vmesnik, kot je prikazan na Sliki 4.12.

Ustvarimo selektor, ki nam bo oblikoval vse gumbe na spletni strani. To

Slika 4.12: Končana izdelava uporabniškega vmesnika zaglavja orodja.

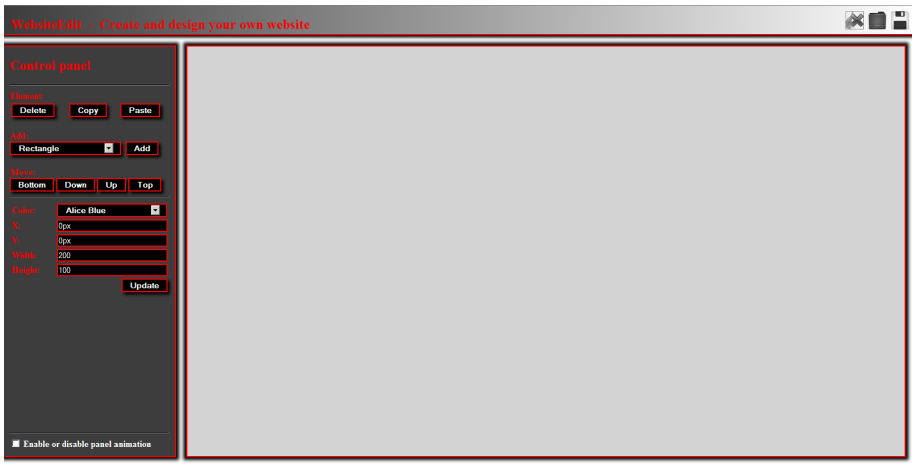
lahko dosežemo tako, da selektor definiramo kot razred, ki ga nato dodamo vsakemu gumbu. Podamo tudi obliko za senčenje ter obliko, ko uporabnik klikne na izbrani gumb. Isto obliko lahko uporabimo tudi za formatiranje seznamov v našem orodju, kjer mu podobno kot gumbu podamo v razredu ime selektorja, ki ga želimo vključiti. Grafična oblika spletnega orodja WebsiteEdit je prikazana na Sliki 4.13.

Slika 4.13: Končana izdelava uporabniškega vmesnika stranske plošče orodja.

Definirani razred z rdečo barvo in senčenjem dodamo tudi črtam na stranski plošči ter dodamo barvo ozadja. Zaradi definicije stilov za gumb je grafični izgled stranske plošče dokončan in polno funkcionalen. Podobno kot stranske plošče se lotimo tudi dodajanja barve obrobe in senčenja okvirja vsebine

spletne strani ter dodamo svetlo sivo barvo ozadja.

S tem korakom smo zaključili grafično urejanje orodja. Naše izdelano orodje predstavlja zaključeno, polno funkcionalno celoto z izdelanim grafičnim uporabniškim vmesnikom orodja WebsiteEdit, kot je prikazan na Sliki 4.14.



Slika 4.14: Končana izdelava uporabniškega vmesnika orodja.

4.5 Testiranje orodja

Zaključek razvoja spletnega orodja WebsiteEdit še vedno ne predstavlja točke, na kateri bi lahko zagotovili, da je razvoj končan. Korak, ki je bistvenega pomena pri razvoju vsake aplikacije, je njeno testiranje ter pregled funkcionalnosti, da le-te zadostujejo specifikaciji, ki je bila zapisana v okviru načrta vsake aplikacije [3].

Trenutno obstajajo številne tehnike in načini testiranja aplikacij. Med najpomembnejšimi in najbolj razširjenimi načini testiranja aplikacij je seveda samodejno testiranje, kjer s pomočjo različnih orodij in prijemov izdelamo in vzporedno z aplikacijo razvijamo testno okolje, ki nam lahko v življenjski dobi aplikacije zagotovi njeno pravilno funkcionalno delovanje [3]. Zaradi zahtevnosti in narave razvoja orodja WebsiteEdit se za ta pristop testiranja

ne bomo odločili, ampak se bomo oprijeli enega prvih načinov testiranja aplikacij.

Zaradi sodelovanja pri načrtovanju specifikacije orodja WebsiteEdit vemo, kakšne so njene funkcionalnosti in katere točke razvoja je moralo orodje podpreti. V ta namen bomo izdelano orodje preverili na preprost način: začeli ga bomo uporabljati ter skušali preveriti delovanje vseh funkcionalnosti. Ob testiranju bomo pozorni, da je rezultat vsake posamezne akcije uporabnika skladen z zapisom v načrtu orodja. Tako lahko zagotovimo, da naše orodje deluje pravilno in zadostuje kriterijem implementiranih funkcionalnosti.

Če ugotovimo, da določena funkcionalnost ne deluje, skušamo napako odpraviti ter nato ponovimo postopek testiranja celotnega orodja. Ko smo prepričani, da je naše orodje polno delujoča aplikacija, smo lahko prepričani, da smo implementacijo funkcionalnosti orodja zaključili z dobro oceno. Na podlagi najdenih napak v izvajanju orodja lahko ocenimo sposobnost razvijanja aplikacij ter na podlagi izkušenj v prihodnje ocenimo, koliko napora bo potrebnega za razvoj nove aplikacije [3]. Prav tako lahko s pravilnim načinom pristopa testiranja pridobimo ogromno novih in dragocenih izkušenj, ki nam lahko pridejo še kako prav pri razvoju ali vzdrževanju trenutne ali katere druge aplikacije.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo se podrobneje seznanili z implementacijo najnovejših tehnologij in orodij ter jih praktično uporabili ob razvoju orodja za grafično upravljanje s spletnimi stranmi. Ob samem razvoju orodja bi lahko naleteli na ogromno težav, ki bi upočasnile razvoj ali nas celo prisilile v spremembo razvijajočega koncepta ali zasnove tehnologije.

V izogib težavam smo spoznali, kako pomembno je poznavanje najnovejših tehnologij ter implementacij že obstoječih orodij ali knjižnic, ki nam vsakodnevno lajšajo razvoj produkta. Zaradi podrobnejše raziskave tehnologij, uporabe dobrih praks razvoja aplikacij in uporabe konceptov snovanja arhitekture smo si prihranili ogromno težav ter razvili orodje, ki je modularno, preprosto ter omogoča uporabniku enostavno nadaljnje posodabljanje, urejanje ter razvoj.

Razvito orodje uporabljamo približno dva meseca in smo zadovoljni z njegovo uporabo. Glede na želje in cilje nam orodje ponuja funkcionalnosti, ki jih trenutno podpira zelo malo podobnih sistemov za upravljanje s spletnimi vsebinami.

Dejstvo je, da bi lahko v orodje vključili številne dodatne funkcionalnosti, v prihodnje implementirali za zdaj še neznano tehnologijo ali knjižnico in še veliko več. V kolikor želimo, da bo izdelano orodje uporabno še naprej, bomo morali vzporedno z razvojem novih tehnologij razvijati in dopolnjevati orodje

z novimi funkcionalnostmi ter po potrebi spremeniti arhitekturno zasnovo orodja.

Ob samem testiranju orodja smo izdelali veliko različic spletnih strani, s katerimi smo lahko pokazali in dokazali zmožnosti razvitega orodja. Na podlagi arhitekturne zasnove in same implementacije smo bili zelo presenečeni nad zmožnostmi orodja, katere si sprva niti nismo znali predstavljali. Izkazalo se je, da lahko tako enostavno orodje pokrije ogromno različnih načinov in prijemov izdelave spletne strani ter uporabniku ponudi v uporabo napredno spletno stran.

S ponosom lahko rečemo, da nam je bil razvoj orodja v veselje, pridobili smo ogromno novega in uporabnega znanja, ki nam bo odlično služilo pri nadaljnjem razvoju aplikacij. Pridobili smo si tudi večji obseg poznavanja najnovejših tehnologij, namenskih orodij, uporabnih knjižnic in vtičnikov, ki nam olajšajo izdelavo modernih aplikacij.

Cilj izdelave diplomskega dela je bil dosežen, saj smo razvili orodje, ki nam ponuja funkcionalnosti, ki smo jih najprej predstavili kot idejo, nato nadaljevali s snovanjem zahtev ter specifikacije in podrobno izdelanim načrtom orodja s samo implementacijo. Prepričani smo, da je lahko izdelano orodje v pomoč številnim razvijalcem in uporabnikom spletnih strani, saj se lahko z njegovo uporabnostjo, enostavnostjo in preprostostjo približamo zahtevnim sistemom za dosego podobnega rezultata.

Literatura

[1] D. Addey, *Content Management Systems (Tools of the Trade)*, Glasshaus, 2003.

[2] J. Jackson, *Web Technologies: A Computer Science Perspective*, Prentice Hall, 2007, pogl. 2–5.

[3] L. Shklar, R. Rosen, *Web application architecture: principles, protocols, and practices*, John Wiley, 2003.

[4] (2013) Eligrey, "eligrey/FileSaver.js · GitHub". Dostopno na: https://github.com/eligrey/FileSaver.js/

[5] (2013) J. Koetsier, "5,000 developers say HTML5 is real". Dostopno na: http://venturebeat.com/2013/02/26/5000-developers-say-html5-is-real-its-now-and-yeah-its-also-the-future/

[6] (2013) P. Lepage, "Pete Lepage on the Web". Dostopno na: https://petelepage.com/blog/2010/07/tips-tricks-cheat-sheets-for-web-people/

[7] (2013) Mozilla, "BlobBuilder - Web API interfaces". Dostopno na: https://developer.mozilla.org/en-US/docs/Web/API/BlobBuilder

[8] (2013) M. Pilgrim, "Local Storage - Dive Into HTML5". Dostopno na: http://diveintohtml5.info/storage.html

[9] (2013) N. Saumure, "Why You Should Get a CMS". Dostopno na: http://www.prosar.com/inbound_marketing_blog/bid/169118/4-Reasons-Why-You-Should-Get-a-CMS-for-Your-Business-Website

[10] (2013) Wikipedia, "ActiveState Komodo". Dostopno na: http://en.wikipedia.org/wiki/ActiveState_Komodo

[11] (2013) Wikipedia, "Firebug (software)". Dostopno na: http://en.wikipedia.org/wiki/Firebug_(software)

[12] (2013) Wikipedia, "Firefox". Dostopno na: https://en.wikipedia.org/wiki/Firefox