Университет по библиотекознание и информационни технологии

Факултет „Информационни науки“

Катедра „Информационни системи и технологии“

МАГИСТЪРСКА ТЕЗА

на тема

Създаване на сайт за търсене на работа

Дипломант: Анжело Димитров

Фак.№ 472

Специалност: ИСТ

Форма на обучение: задочно

Научен ръководител:

Проф. дн Иван Гарванов

София

2018

УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ

ДЕКЛАРАЦИЯ

От **Анжело Емилов Димитров**

(име, презиме и фамилия на студента)

С настоящата отстъпвам ⊗ / не отстъпвам □ безвъзмездно право на УНИБИТ да публикува представената моя авторска разработка (курсова работа, дипломна работа, магистърска теза) като общодостъпен ресурс за безплатен публичен достъп чрез информационните системи на УНИБИТ.

Дата:..... **Подпис**.....

(дипломант)

Съгласен съм авторската разработка (курсова работа, дипломна работа, магистърска теза) да бъде публикувана като общодостъпен ресурс за безплатен публичен достъп чрез информационните системи на УНИБИТ.

Дата:..... **Научен ръководител** .....

……

(подпис)

Резюме:

Дипломант: Анжело Димитров;

Тема: Създаване на сайт за търсене на работа;

Научен ръководител: проф. дн Иван Гарванов;

Година: 2018;

град: София;

Катедра „Информационни системи и технологии”;

Магистърска програма: ”Информационни системи и технологии”;

Университет: Университет по библиотекознание и информационни технологии;

Брой страници: 103 стр.;

Брой източници: 27;

Брой приложения: 1;

Цел на магистърската теза: да се разработи уеб сайт за търсене на работа;

Ключови думи: html, javascript, работа, уеб сайт.

Съдържание:

Увод

Целта на магистърската теза е да се създаде сайт за търсене на работа чрез редактора за програмен код **Visual Studio Code**. Това е първият редактор на Microsoft, който може да се ползва под Linux и OS X. Поддържа богат набор от инструменти за разработване като дебъгване, вграден Git Control, IntelliSense, "Side-by-Side Editing" (позволява работа едновременно върху 2 файла отворени един до друг) и др. Той също така дава възможност за персонализиране, което означава, че потребителите могат да променят темата на редактора, клавишните комбинации, настройките и др.

Магистърската теза се състой от три глави: **Въведение –** като първа глава, описваща история и развитие на интернета, уебсайтовете и видове уебсайтове. **Езици за програмиране, браузъри, бази данни –** като втора глава, описва видовете езици за програмиране, запознаване и примери с тях; видове бази данни и системи за управление на бази данни; и видове браузъри.

Разработка на уеб сайт – това е третата глава, в нея съм качил скрийншотове на сайта както и част от кода на страниците /HTML, CSS & JS/.

I. Въведение

1.Интернет - кратка история

1.1. История и развитие на Интернет

През 1957 г. Съветският извежда в орбита първия сателит “Sputnik”. Този факт е голяма изненада за света и най-вече за американските учени. В резултат на това САЩ формират Агенция за високотехнологични проекти (ARPA - Advanced Research Project Agency) под покровителството на Министерството на отбраната. Целта на създаването на ARPA е била да поддържа лидерството в технологиите на САЩ, в частност с особено внимание към отбраната. (10)

В началото на 60-те години paul baron в rand Corporation работи по решаване на проблема за изпращане на информация между компютри в мрежа чрез разделянето й на пакети, които съдържат освен данните, които ще се прехвърлят, още и информация за маршрута, т.е. компютъра, от който се изпраща и местоназначението на тази информация.

През 1968 година Националната лаборатория по физика създава първата мрежа, прехвърляща пакети информация.

Историята на Интернет започва когато в резултат на предложението rand Corporation за свързване в мрежа всички компютри от Министерството на отбраната, ARPA създава през 1969г. ARPANET - мрежа, при която предаваните информационни пакети могат да следват различни възможни маршрути, което позволява функционирането на мрежата, дори когато отделни сегменти от нея бъдат разрушени. Тази мрежа е послужила като прототип, върху който са тествани теоретичните постановки и софтуерът,

залегнали в основата на Интернет. След като стигат до извода, че разработената технология може да се приложи успешно и в други области, проектантите на ARPA включват към нея и университетите. Така мрежата постепенно се разширява и през 1971 г. в нея участват 15 опорни точки (суперкомпютри), а през 1972 г. – над 30.

Най-големият принос на проекта ARPANET е въвеждането на универсален комуникационен протокол за обмен на данни между компютрите в мрежата. През 1982г. ARPANET преминава на TCP/IP мрежов протокол (контролно-предавателен протокол/интернет протокол - Transmission Control Protocol/Internet Protocol). Чрез въвеждането на този протокол става възможен обмена на информация между отделни мрежи, използващи различни протоколи. През 1983 г. ARPANET се разделя на военна мрежа (MILnet) и гражданска мрежа (SCNET – Computer Science Network) и се появява по-краткото наименование на тази мрежа – "Интернет".(10)

През 1986 г. National Science Foundation (Националния научен фонд на САЩ) създава мрежата NSFNET, която осигурява някои големи технологични предимства – по-висока скорост на трансфер и по-мощни суперкомпютри за съхранение на информацията. В NSFNET участват огромен брой университети и научно-изследователски институти на САЩ. В последствие NSFNET става най-бързата и надеждна мрежа в състава на Интернет.

На 01.06.1990 г. първоначалната ARPANET се разпада, но преди това нейните потребители са преминали към мрежата NSFNET.

През 1989 г. Tim Bernes - Lee - физик от CERN, Швейцария, изобретил *WWW* (*World Wide Web)* като начин да се организира информацията като модел, наподобяващ повече човешки мозък. Неговата идея била да позволи на

хората да правят многократни връзки с различни парчета информация. Web сега замества всички други услуги, като FTP, gopher ,Wais и др. Web е толкова популярен в момента, че много хора си мислят за него, когато се спомене за Интернет. Степента на растеж на Web през 1993 година е била над 340%.

Растежа на Интернет е експлозивен. През 1995 г. имало само около 2000 хост-компютри и 10 милиона потребители в Интернет. През същата година, така както ARPANET преди това, NSFNET отмира, след като е изпълнила предназначението си за ускоряване развитието на Интернет и разширяване на неговия обхват. През този финален етап на формирането си Интернет обединява всички алтернативни мрежи CompuServe, MSN, Prodigy и AOL , които повече не могат да съществуват самостоятелно. Докато първоначалният Интернет бил създаден с изследователски цели, сега той се използва предимно с комерсиална цел.

Само за справка, според NEC Recesearch Institude от месец януари 2000 година всяка секунда към съществуващите 1,4 милиарда Web страници в света се добавят още по 25 нови.

В края на 2002 г. по света е имало 655млн потребители на Интернет, което е с 30% повече от края на 2001-ва.

През 2006 г. 18% от всички покупки от фирми и частни лица по света ще се извършват в мрежата. Популярността на он-лайн банкирането става все по-голяма в САЩ, където 37 милиона американци (32% от Интернет потребителите) извършват банкови транзакции в мрежата. Мобилните комуникации предлагат огромни възможности за развитие на търговията и обещават да променят изцяло начина по който пазаруваме сега. Мобилната търговия си пробива стабилно път най-вече в Северна Америка, Западна

Европа и Япония. През 2002 г. т.н. e-commerce е генерирала приходи от почти $50 млрд., като на първо място тук са SMS, следвани от микроразплащанията, финансовите и информационни услуги, мобилното управление на взаимоотношенията с клиентите.

1.2. Структура на Интернет

В прекия смисъл на думата Internet означава мрежа, състояща се от 2 и повече взаимно свързани мрежи. Думата Интернет обаче има и друго значение. Сега в света съществуват стотици хиляди големи и малки мрежи, като повечето от тях са свързани. Така се е образувало единно информационно поле, наречено Интернет.

Най-простото определение за Интернет - мрежа от мрежи или световна мрежа за обмен на информация.Интернет има йерархична структура на 4 нива, като може да се оприличи на сложна паяжина.

1. В нейният център (т.е. първо равнище) се намира т. нар. Опорна мрежа (backbone) от основни възли. Те са реализирани на базата на свръхмощни компютри (суперкомпютри), свързани помежду си с магистрални линии с максимален пропускателен капацитет.
2. По радиуса на "паяжината" магистрални линии свързват към опорната мрежа глобални компютърни мрежи (WAN) с регионален, национален и международен мащаб, като образуват второто равнище. Измежду международните мрежи в състава на Internet се открояват BITNET, EUNET, EARN и др., свързващи компютри съответно в северноамерикански и западноевропейски университети.(20)
3. Трето равнище представляват влизащите в състава на глобалните мрежи локални мрежи (LAN).

4. На последно IV равнище са отделни компютри в състава на локални мрежи или свързани пряко към глобалната мрежа. Компютрите, които участват в обмена на информация в Интернет и се подчиняват на нейните комуникационни правила се наричат възли. Тези от тях, които са включени постоянно и предоставят една или няколко услуги на отдалечените потребители на Интернет се наричат сървъри. Компютрите, които са включени в Интернет и се свързват със сървърите, за да получат достъп до услугите, които те предлагат, се наричат клиенти. Интернет включва и т.н. маршрутизатори (routers) - това са специализирани компютри, следящи и управляващи трафика на данни в мрежата. Те насочват предаваните данни по най-подходящ свободен в момента маршрут, така че да се постига по-висока скорост и се сведат до минимум задръстванията.(20)

Интернет обединява над 100 хиляди компютърни мрежи с над 200 милиона компютри от всякакъв вид - настолни и преносими персонални компютри, работни станции, миникомпютри, големи машини.

1.3. Софтуер

Приложният софтуер за ползване на Интернет се основава на модела "клиент / сървър" за организация на обслужването на потребителите. Съобразно този модел той е разделен на 2 части - за клиента (компютърът, искащ информация) и за сървъра, т.е. обслужващ компютър (наричан често хост).

Чрез клиентския софтуер или накратко "клиентът" потребителите задават своите заявки за определени услуги, а тя от своя страна определя в каква форма да представи данните на потребителя и по какъв начин да осъществи търсенето, като взаимодейства с един или няколко еднотипни сървъра, на които е разположена информацията. Клиентските програми

подържат потребителския интерфейс и другите детайли на заявките и резултатите, зависещи от вида на съответните потребителски компютри.

Сървърният софтуер, управлява информационните ресурси на сървъра и отговаря на отправените към него запитвания. Сървърите обслужват едновременно много клиенти, като подреждат постъпилите заявки по определени правила.

Независимо от това, че много от сървърите са програмно несъвместими, цялата система функционира надеждно благодарение на това, че всеки сървър използва стандартен протокол за предаване на данни - TCP/IP (Transmission Control Protocol / Internet Protocol). Този комуникационен протокол осигурява обмена на данни между всякакви компютри, влизащи в състава на Интернет чрез метода на пакетната комутация. Протоколът TCP/IP всъщност не е един протокол, а са два. Първият протокол (TCP) определя начина, по който информацията, която трябва да се изпрати от един компютър към друг, се разбива на пакети, изпраща ги към тяхното направление. След това отговаря за пристигането им в правилен ред и събирането им в цял документ. Вторият протокол (IP) отговаря за начина, по който тези пакети се предават в мрежата и достигат до получателя.

1.4. Адресиране

За да може всеки възел в Интернет да доставя или ползва услуги, той се идентифицира с уникален адрес. Протоколът TCP/IP определя я начина за съставяне на тези адреси.

Всеки сървър от мрежата има постоянен уникален цифров (IP) адрес. (Компютър, включен към сървър с помощта на отдалечена комутируема връзка,

получава по време на работата временен IP адрес. Докато връзката съществува, автономният компютър става участник в мрежата и може да обменя информация с всеки друг компютър в нея.)

Цифровият (IP) адрес на възела съдържа четири елемента под формата на числа от 0 до 255. Например: 193.68.254.193 . Тази форма на записване е удобна за компютрите, но не и за хората, защото такива адреси се помнят трудно.

Затова съществува и друга, по-удобна, форма на записване, използваща системата от имена на домейни (DNS - Domain Name System) - йерархична система от имена на области (домейни), позволяваща еднозначно да се идентифицира възелът. Този адрес включва 4 елемента (представляващи означения с малки или големи букви на латиница), разделени с точка и подредени в следната последователност:

1. Означение на възела;
2. Подобласт на ниско йерархично равнище (катедра, отдел);
3. Подобласт на средно равнище (организация, университет);
4. Област (домейн) - буквен код на страната, в която се намира сървърът, а за САЩ - трибуквен код за вида на организацията, в която е инсталиран той.

За САЩ - net - организация към мрежата, edu - образователна (университет), com – фирмен или изобщо комерсиален сайт, mil – военен сайт, gov – правителствен орган, org – нестопанска организация, net – Интернет шлюз или административен хост; ac -академичен във Великобритания, co - фирмен или изобщо комерсиален сайт във Великобритания и Нова Зеландия.

Съществуват специални DNS сървъри, които автоматично преобразуват буквения DNS адрес в съответния IP адрес, чрез който се осъществява връзката.

1.5. Основни услуги в Интернет

Накратко основните услуги, предлагани в Интернет могат да се разделят на две основни категории – комуникационни и информационни. Комуникационните услуги са предназначени за предоставяне на различни средства за общуване. Такива са електронната поща, новинарски групи, дискусионни списъци, чат, телефонни разговори и видео конференции. Информационните услуги са свързани с обмен на файлове (текстови, графични, музикални, мултимедийни) и търсене на информационни ресурси . Такива са WWW, FTP, Telnet, Gopher, Archie, WAIS. Ще разгледаме накратко най-популярните услуги на Интернет :

1.5.1. Електронна поща

Електронната поща е една от първите и най-популярни услуги в Интернет. Тя служи за обмен на съобщения и прикрепени към тях файлове чрез компютри и мрежа за пренос на данни. Възможно е да се изпращат и факсове по Интернет.

1.5.2. Новинарски групи

Тази интересна услуга е своеобразно разширение на електронната поща. Новинарските групи се наричат още дискусионни групи или групи по интереси, бюлетин-бордове(електронни табла за обяви).

1.5.3. Дискусионни (пощенски) списъци

Те са друг начин на комуникиране между потребителите на Интернет. Докато Usenet съобщенията се събират на едно и също централно място в хост системата, съобщенията в дискусионните списъци се доставят директно в пощенската кутия на потребителя, който се е абонирал за тях.

1.5.4. Разговори по интернет (Chat) , телефонни разговори и видеоконференции.

За разлика от UseNet и електронната поща, при Интернет разговорите, телефонните разговори и видеоконференциите се провеждат на в реално време (on-line). Най-простата услуга от този тип е разговор по Интернет (Chat), чрез който двама абонати общуват, като всеки от тях разполага на дисплея на своя компютър с два прозореца, като в единия може да въвежда своите съобщения, а в другия – да чете съобщенията на своя събеседник. По-сложен начин за on-line общуване е Интернет-телефония, т.е. възможността да се пренася по мрежата човешки глас. Най – съвършеният начин за връзка в реално време е видеоконференцията, при която освен звук се предава и видео по мрежата.

1.5.5. Трансфер на файлове (FTP).

FTP (File Transfer Protocol) е една от най-старите услуги на Интернет. FTP е услуга за трансфер на файлове между компютри, свързани в мрежа и основният метод за прехвърляне на файлове по Интернет.

1.5.6. Telnet

Услугата Telnet позволява достъп до отдалечен компютър и ползване на неговите ресурси (процесор, дисково пространство, принтер и др.). Така получаваме достъп до ресурси, с каквито не разполагаме: изчислителни и запаметяващи, бази от данни и софтуер.

1.5.7. World Wide Web (WWW)

WWW е най-известната и най-популярната услуга на Интернет. Предназначена е за интерактивна работа. Тя предоставя възможност за търсене и събиране на информация не само текстова, но и мултимедийна, чрез "паяжина" от влизащи в състава на Internet компютри, наричани Web-сървъри. Много от потребителите на Интернет ползват предимно WWW, понеже повечето от услугите в Интернет са достъпни и чрез Web.

2.Уебсайт

2.1 Какво е уебсайт?

Уебсайт е съвкупност от уеб страници, които се адресират на общ URL, който често се състои само от името на домейна или IP адреса и пътя до основната директория („/“) в мрежа базирана на протокола IP. Всеки уебсайт се хоства на компютър (компютри), наречен уеб сървър, достъпен през мрежа като например Интернет или частни локални мрежи. За разглеждането на уебсайтове са създадени специални програми, наречени уеб браузъри. Съществуват различни видове уебсайтове в зависимост от тяхното предназначение, аудитория и съдържание (11).

Сайтовете се съхраняват на уеб сървъра под формата на файлове. Всеки файл съдържа информация, която директно или след някаква обработка се изпраща на уеб браузъра, когато той поиска съдържанието на дадена страница. Сайтовете, които правят обработка на информацията, се наричат динамични, а тези, които не правят такава обработка – статични. Статичните сайтове се създават по-лесно и работят по-бързо, но не позволяват промяна на съдържанието. Почти всички съвременни сайтове са динамични.

2.2 Видове уебсайтове

- Блог – сайт, който представлява личен дневник и може да съдържа дискусионен форум
- Електронен магазин – сайт, който служи за продажба на стоки по интернет
- Корпоративен сайт – сайт, който осигурява основна информация за фирма, организация или услуга
- Личен сайт – сайт, който се поддържа от частно лице или малка група хора
- Новинарски сайт – сайт, който предоставя главно новини и репортажи
- Форумен сайт – сайт, в който хора дискутират на различни теми
- Портален сайт – сайт, който служи за отправна точка към различни други сайтове в интернет или интранет
- Сайт с база данни – сайт, който служи за търсене и намиране на информация в специфична база данни
- Сайт за обяви – уеб базирана платформа за публикуване, четене и отговор на обяви в различни рубрики
- Сайт за запознанства – сайт, в който самотните хора си уговарят срещи
- Огледален сайт – сайт, който е точно копие на друг сайт, но функционира под различен домейн
- Продуктов или промоционален сайт
- Сайт на общности или сайт на социална мрежа
- Търсачка – сайт, който осигурява обща информация и служи за намиране на други сайтове

- Уикисайт – сайт, в който посетителите могат съвместно да редактират информацията (като Уикипедия)
- Фенсайт – сайт, който се поддържа от почитателите на някоя знаменитост
- Приспособим (адаптивен) уеб сайт – сайт, който отговоря на (приспособява се към) характеристиките на устройството на потребителя, съобразявайки размера и ориентацията на екрана, платформата.

II. Езици за програмиране, браузъри, бази данни

1. Въведение

Езикът за програмиране е изкуствен език, предназначен за задаване на команди, които искаме компютъра да прочете, обработи и изпълни. Чрез езиците за програмиране пишем поредици от команди, които задават какво да прави компютъра. Изпълнението на компютърните програми може да се реализира с компилатор или с интерпретатор (9).

Компилаторът превежда кода от програмен език на машинен код, като за всяка от командите в кода избира подходящ, предварително подготвен фрагмент от машинен код, като междувременно проверява за грешки текста на програмата. Заедно компилираните фрагменти съставят програмата в машинен код, както я очаква микропроцесорът на компютъра. След като е компилирана програмата, тя може да бъде директно изпълнена от микропроесора в кооперация с операционната система. При компилируемите езици за програмиране компилирането на програмата се извършва задължително преди нейното изпълнение и по време на компилация се откриват синтактични

грешки /грешно зададени команди/. С компилатор работят езици като C++, C#, Swift и Java.

Някои езици за програмиране не използват компилатор, а се интерпретират директно от специализиран софтуер, наречен интерпретатор. Интерпретаторът е програма за изпълняване на програми, написани на някакъв програмен език. Той изпълнява командите на програмата една след друга, като разбира не само от единични команди и поредици от команди, но и от другите езикови конструкции (проверки, повторения, функции и т.н.). Езици като PHP, Python и JavaScript работят с интерпретатор и се изпълняват без да се компилират. Поради липса на предварителна компилация, при интерпретируемите езици грешките се откриват по време на изпълнение, след като програмата започне да работи, а не предварително (9).

Средата за програмиране (Integrated Development Environment - IDE, интегрирана среда за разработка) е съвкупност от традиционни инструменти за разработване на софтуерни приложения. В средата за разработка пишем код, компилираме и изпълняваме програмите. Средите за разработка интегрират в себе си текстов редактор за писане на кода, език за програмиране, компилатор или интерпретатор и среда за изпълнение на програмите, дебъгер за проследяване на програмата и търсене на грешки, инструменти за дизайн на потребителски интерфейс и други инструменти и добавки.

Средите за програмиране са удобни, защото интегрират всичко необходимо за разработката на програмата, без да се напуска средата. Ако не ползваме среда за разработка, ще трябва да пишем кода в текстов редактор, да го компилираме с команда от конзолата, да го изпълняваме с друга команда от конзолата и да пишем още допълнителни команди, когато се налага, и това ще

ни губи време. Затова повечето програмисти ползват IDE в ежедневната си работа.

За програмиране на езика Javascript се използват средите за разработка Visual Studio или Visual Studio Code, като втората е по-малко натоварваща и може да тръгва, без да затормозява работата на компютри с по-слаби параметри.

Програмата в своята същност е набор от инструкции, които карат компютъра да свърши определена задача. Те се въвеждат от програмиста и се изпълняват безусловно от машината.

Съществуват различни видове езици за програмиране. С езиците от най-ниско ниво могат да бъдат написани самите инструкции, които управляват процесора, например с езика "assembler". С езици от малко по-високо ниво като C и C++ могат да бъдат създадени операционна система, драйвери за управление на хардуера (например драйвер за видеокарта), уеб браузъри, компилатори, двигатели за графика и игри (game engines) и други системни компоненти и програми. С езици от още по-високо ниво като C#, Python и JavaScript се създават приложни програми, например програма за четене на поща или чат програма.

Езиците от ниско ниво управляват директно хардуера и изискват много усилия и огромен брой команди, за да свършат единица работа. Езиците от по-високо ниво изискват по-малко код за единица работа, но нямат директен достъп до хардуера. На тях се разработва приложен софтуер, например уеб приложения и мобилни приложения.

Както вече споменахме, програмата е последователност от команди, иначе казано тя описва поредица от пресмятания, проверки, повторения и всякакви подобни операции, които целят постигане на някакъв резултат.

Програмата се пише в текстов формат, а самият текст на програмата се нарича сорс код (source code). Процесът на компилация на кода преди изпълнение се използва само при компилируеми езици като Java, C# и C++. При скриптови и интерпретеруеми езици, като JavaScript, Python и PHP, сорс кодът се изпълнява постъпково от интерпретатор.

2.Видове езици за програмиране

2.1. Java

Обектно-ориентиран и основна концепция са класовете, които се дефинират чрез свойствата (атрибутите) и поведението (методите) на обектите. Няма глобални променливи и функции – всичко се дефинира в класове (18).

Java е език от високо ниво, който прилича на C# и C++ и донякъде на езици като Delphi, VB.NET и C. Синтаксисът на Java е сходен до този на C++ и C# и преминаването от тези езици към Java е лесно. Java програмите са обектно-ориентирани. Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика.

На Java се разработва разнообразен софтуер: офис приложения, уеб приложения, настолни приложения, приложения за мобилни телефони, игри и много други.

В днешно време Java е един от най-популярните езици за програмиране. На него пишат милиони разработчици по цял свят. Най-големите световни софтуерни корпорации като IBM, Oracle, Google и SAP базират своите решения на Java платформата и използват Java като основен език за разработка на своите продукти. Сред „големите“, Java не се ползва единствено от Microsoft, тъй като те разработват и поддържат собствена платформа, подобна на Java платформата, наречена .NET Framework и език, подобен на Java, наречен C#.

Езикът Java е първоначално разработен и поддържан от Sun Microsystems, но през 2006 г. Java платформата става проект с отворен код и се поддържа и развива от световната Java общност съвместно със Sun. Благодарение на отворения код популярността на Java постоянно се увеличава и броят Java разработчици непрекъснато расте (18).

2.2. C# (C Sharp)

Обектно-ориентиран език за програмиране, разработен от Microsoft, като част от софтуерната платформа .NET. Стремежът още при създаването на C# езика е бил да се създаде един прост, модерен, обектно-ориентиран език с общо предназначение. Основа за C# са C++, Java и донякъде езици като Delphi, VB.NET и C. Той е проектиран да балансира мощност (C++) с възможност за бързо разработване (Visual Basic и Java). Те представляват съвкупност от дефиниции на класове, които съдържат в себе си методи, а в методите е разположена програмната логика – инструкциите, които компютърът изпълнява. Програмите на C# представляват един или няколко файла с разширение .cs., в които се съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# (csc) до изпълним код и в

резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll) (12).

2.3. PHP

PHP е скриптов език със синтаксис базиран на C и Perl. Използва се предимно в Web-среда за реализиране на широк кръг от услуги. Той е един от най-популярните езици за програмиране в Интернет и популярността му расте непрекъснато (22).

PHP се разпространява под отворен лиценз (PHP License), който по своята същност е BSD лицензa и който позволява безплатно разпространяване на програмния код на интерпретатора на езика, както и създаването на производни интерпретатори под други лицензи с уговорката, че тези интерпретатори не могат да включват PHP в името си. Фактът, че PHP се разпространява свободно, го прави удачен избор за изграждане на Web-сървър базиран изцяло на свободни продукти – GNU/Linux, Apache, MySQL/PostgreSQL и др.

При поискване, кодът, който е написан на PHP се интерпретира от уеб сървъра на който е качен, и резултатът се връща на уеб браузърът. Потребителят не може да види чистият PHP код без да има достъп до самият файл в който той е записан. По този начин е помислено за сигурността. PHP файловете могат да съдържат текст, HTML, CSS, JavaScript и PHP код. PHP файловете имат разширение .php.

2.4. Python

Интерпретируем, интерактивен, обектно-ориентиран език за програмиране, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC „Monty Python’s Flying Circus“.

„Python“ предлага добра структура и поддръжка за разработка на големи приложения. Той притежава вградени сложни типове данни като гъвкави масиви и речници, за които биха били необходими дни, за да се напишат ефикасно на C (23).

Python позволява разделянето на една програма на модули, които могат да се използват отново в други програми. Също така притежава голям набор от стандартни модули, които да се използват като основа на програмите. Съществуват и вградени модули, които обезпечават такива неща като файлов вход/изход (I/O), различни системни функции, сокети (sockets), програмни интерфейси към GUI-библиотеки като Tk, както и много други.

Тъй като Python е език, който се интерпретира, се спестява значително време за разработка, тъй като не са необходими компилиране и свързване (linking) за тестването на дадено приложение. Освен това, бидейки интерпретируем език с идеология сходна с тази на Java, приложение, написано на него, е сравнително лесно преносимо на множеството от останали платформи (или операционни системи).

2.5. Ruby

Интерпретируем, интерактивен, обектно-ориентиран език за програмиране. Комбинира черти от много други езици, сред които Smalltalk, Perl, Lisp и Python. Основната му имплементация е безплатна и с отворен код (24).

2.6. Swift

Много-парадигмен компилируем програмен език, създаден от „Apple“ за разработване на приложения за iOS и Mac. Представен за първи път на Световната Конференция за Софтуерни Разработчици (WWDC Worldwide

Developers Conference) през април 2014, Swift е разработен за Cocoa и Cocoa Touch. Проектиран е по такъв начин, че да помага на разработчиците при създаването на по-безопасен и надежден код като отстранява и избягва цели категории разпространени програмни грешки. Swift може да се интегрира във вече създадени приложения благодарение на възможността за съвместното му използване с код, написан на Objective-C (26).

2.7. HTML

Основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в Интернет, а правилата се определят от международния консорциум W3C (16).

Описанието на документа става чрез специални елементи, наречени HTML елементи или маркери, които се състоят от етикети или тагове (HTML tags) и ъглови скоби (като например елемента <html>). HTML елементите са основната градивна единица на уеб страниците. Чрез тях се оформят отделните части от текста на една уеб страница, като заглавия, цитати, раздели, хипертекстови препратки и т.н. Най-често HTML елементите са групирани по двойки <h1> и </h1> (2).

В повечето случаи HTML кодът е написан в текстови файлове и се хоства на сървъри, свързани към Интернет. Тези файлове съдържат текстово съдържание с маркери – инструкции за браузъра за това как да се показва текстът. Например <маркер> Някакъв текст. </край на маркера>. Предназначението на уеб браузърите е да могат да прочетат HTML документите и да ги превърнат в уеб страници. Браузърите не показват HTML таговете, а ги използват, за да интерпретират съдържанието на страницата.

Основното предимство на HTML е, че документите, оформени по този начин, могат да се разглеждат на различни устройства, а не само на екрана. Документът може да бъде правилно оформен и върху монитора на персонален компютър, и върху миниатюрния дисплей на пейджър или мобилен телефон.

HTML може да прикрепя скриптове писани на езици като JavaScript, което променя поведението на уеб страницата. Може да се използва Cascading Style Sheets (CSS), който определя изгледа и оформлението на текста и други материали. World Wide Web Consortium (W3C) поддържа и двете CSS и HTML и насърчава използването на CSS в HTML страниците от 1997. Това допринася за разделяне съдържанието и структурата на уеб страниците от тяхното визуално представяне (2).

Създаването на HTML-базирана уеб страница може да се извърши с помощта на обикновен текстов редактор. Този начин изисква познаване на HTML тагове, така че те да бъдат интегрирани в текста, който ще се показва на страницата. Също така, често срещани са по-приятелски настроените инструменти, които не изискват от потребителя да притежава познания по HTML, което му позволява да създаде страница по метода WYSIWYG. Основен инструмент за тази цел е текстообработваща програма Word, която позволява да запазите документ като HTML и да го редактирате. Специализирани инструменти за създаване на HTML страници са Microsoft, FrontPage, Macromedia Dreamweaver, Notepad, Notepad++,Sublime Text и други.

2.7.1. Структура и основни елементи

HTML таговете са най-малката съставна част на един HTML документ. Те, заедно със своите атрибути (като цвят, размер и т.н), съставят HTML елементите. Таговете са ключови думи, заградени в ъглови скоби, като обикновено са по двойки: таг за начало, който маркира началото на елемента

и таг за край, който маркира края на елемента, крайният има наклонена надясно чера пред името си (5).

- <!DOCTYPE html> - Декларира се първи, още преди <html> тага. Валидира документа. <!DOCTYPE> не е HTML таг. Той е инструкция за уеб браузъра – указва HTML версията, на която е написана страницата.
- Html елемент – указва на браузъра, че това е HTML документ. Отбелязва началото и края на всички други негови елементи.

```
<html>

</html>
```

- Head елемент - съдържа заглавието на документа, и може да съдържа стилове, скриптове и енкодинг.

```
<html>
    <head>
        <title>
            Заглавие
        </title>
    </head>

</html>
```

- Body елемент - Съдържа форматирането, което е видимо за потребителя – текст, хиперлинк, картинки, таблици, бутони, параграфи.

```
<html>
    <head>
        <title>
```

```
            Заглавие
        </title>
    </head>

    <body>
        Тук се пише всичко, което потребителя ще вижда
    </body>
</html>
```

2.7.2. Основни тагове

- <b> - използва се за удебеляване на текста, но в днешно време се препоръчва това да се случва чрез css “**font-weight: bold**” или с тага , тъй като дава по-голяма важност на текста;
- <i> - използва се за задаване на наклон на текста, това също се препоръчва да се прави чрез css “**font-style: italic**” или с тага , тъй като дава по-голяма важност на документа;
- <u> - използва се за подчертаване на текст, препоръчва се да се прави с css, “**text-decoration: underline**”;
- <sub> - дефинира текст под черта;
- <sup> - дефинира текст над черта;
- <del> - указва изтрит текст;
-
 - единичен таг, указва нов ред;
- <blockquote> - указва част от текста, който е цитат;
- <a> - указва линк към друга страница, най-важния му атрибут е **href**, той посочва URL адреса, към който сочи линка;

- <img> - дефинира картинка в страницата, има два задължителни атрибута, **src** – указва URL адреса на картинката и **alt** – указва алтернативен текст на картинката, ако тя не може да се покаже;

2.7.3. Заглавия и параграфи

- заглавия – дефинират се с тагове от <h1> до <h6>, h1 дефинира най-важното заглавие, съответно h6 дефинира най-маловажното заглавие;
- параграфи – указват се с тага <p>, браузърите автоматично добавят по един празен ред преди и след текста, който е маркиран като параграф;
- списъци
    - подредени – започват с тага <ol> /ordered list/ и всеки елемент на списъка започва с тага <li>;
    - неподредени – започват с тага <ul> /unordered list/ и всеки елемент на списъка започва с тага <li>;
    - описателни - това са списъци от термини/имена с описание на всеки термин/име. Указват се с тага <dl> в съчетание с таговете <dl> и <dt>.
- таблици – дефинират се с тага <table>. Разделени са на редове, с тага <tr>, а всеки ред е разделен на клетки с данни чрез тага <td>. Всяка клетка може да съдържа текст, линкове, картинки, списъци, форми, други таблици;

2.7.4. Атрибути

Повечето атрибути на елементи са двойки име – стойност, разделени със знак за равенство и записвани в рамките на отварящият таг след името на елемента. Името може да е с единични или двойни кавички. Оставянето на стойности на атрибути без кавички се смята за несигурно(2).

- Id - предоставя уникален идентификатор за елемента в документа. Използва се за идентифициране на елемента, така, че стиловете да могат да променят свойствата им;
- Class - начин за класифициране на подобни елементи. Може да бъде използван за семантични или презентационни цели;
- Title - използва за прикачване на подтекстово обяснение на елемент. В повечето браузъри този атрибут се показва на екрана като съвет;
- Lang - идентифицира естествения език на съдържанието в елемента, който може да бъде различен от този в останалата част на документа;

2.7.5. HTML5

HTML5 е последният голям проект от HTML стандарта. HTML5 е смесица от много функции, въведени от различни спецификации, заедно с тези, въведени от софтуерни продукти, като уеб браузърите (2).

По-специално, HTML5 добавя много нови синтаксови функции. Те включват <video>, <audio> и <canvas> елементи, също така и интеграция на SVG съдържание, което е създадено да подобрява работата с мултимедийно и графично съдържание в уеб пространството без плъгини и техните APIs. Други нови елементи, като <section>, <article>, <header> и <nav> са създадени

да подобрят семантичното богатство на документите. Други елементи са премахнати. Също така са въведени и нови атрибути.

HTML 5 е поддържан от по-стари браузъри, тъй като е направен така, че новите му функции просто да се игнорират от тях.

2.7.6. Валидиране на HTML

Валидацията е един от критериите за качествена уеб страница, но има ощемного други. Валидна уеб страница означава, че създателя ѝ следва и прилага успешно установените стандарти. Понякога страниците, които създаваш няма да бъдат сто процента валидни. Голяма част от световно популярните уеб сайтове имат грешки в своя код, но това не е пречка на милиони потребители да ги използват ежедневно.

Предимства на валидирането

- Лесен за поддръжка код;
- Добра съвместимост с бъдещи технологии (нови стандарти/браузъри/програми);
- Признак на професионализъм;
- Достъпност от голям кръг от потребители;
- Страницата зарежда по-бързо;
- Инструмент за откриване на грешки;

W3C(World Wide Web Consortium) е основната организация по стандартизация, установяваща международни стандарти за технологии като HTML, xHTML, CSS и много други.

Валидатора е онлайн инструмент (програма), който ни служи за да проверим дали кода на нашата страница е написан според установените стандарти.

http://validator.w3.org/

Валидацията е процес, през който преминаваме, за да проверим дали страницата е написана според установените стандарти за езика, който използваме. Когато кода на нашата страница "премине" успешно валидацията казваме, че той е валиден. И обратно, когато кода не премине валидацията успешно казваме, че той не е написан според стандартите.

Има три начина за валидиране на страницата:

- Валидиране чрез URI;
- Валидиране чрез качване на файл;
- Валидиране чрез директно въвеждане;

Версии на HTML:

- 24 ноември, 1995 г. – HTML 2.0 е представен като RFC 1866.
  Впоследствие са добавени:
- 25 ноември, 1995 г. – качване на файлове с формуляри;
- Май 1996 г. – таблици;
- Август 1996 г. – клиентски картови изображения;
- През януари 1997 г. е представен HTML 3.2. Това е първата версия, разработена и стандартизирана от World Wide Web Consortium.

- През декември 1997 г. е представен HTML 4.0, отново от W3C (World Wide Web Consortium) в три вариации:
    1. Строг, отпадналите елементи са забранени
    2. Междинен, отпадналите елементи са разрешени
    3. Фреймов, най-често рамкови елементи са позволени
- април 1998 г. – версия 4.0 претърпява леки промени, без смяна на номера;
- декември 1999 г. – версия 4.01 е налице;
- януари 2008 г. – HTML5 е представен като работен проект от W3C;
- май 2011 г. – версия 5 е в процес на развитие на техническите спецификации. Пълната спецификация се очаква до 2014 г.
- октомври 2014 – HTML5 е публикувана като препоръка на W3C.

2.8. CSS

Език за описание на стилове (език за стилови листове, style sheet language) – използва се основно за описване на представянето на документ, написан на език за маркиране. Най-често се използва заедно с HTML, но може да се приложи върху произволен XML документ. Официално спецификацията на CSS се поддържа от W3C (13).

CSS е създаден с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Преди стандартите за CSS, установени от W3C през 1995 г., съдържанието на сайтовете и стила на техния дизайн са писани в една и съща HTML страницата. В резултат на това HTML кода се превръща в сложен и нечетлив, а всяка промяна в проекта на даден сайт изисквала корекцията да бъде нанасяна в целия сайт страница по

страница. Използвайки CSS, настройките за форматиране могат да бъдат поставени в един единствен файл, и тогава промяната ще бъде отразена едновременно на всички страници, които използват този CSS файл.

CSS позволява да се определя как да изглеждат елементите на една HTML страница – шрифтове, размери, цветове, фонове, и др. CSS кодът се състои от последователност от стилови правила, всяко от които представлява селектор, последван от свойства и стойности. Например в следния CSS код:

p {font-size: 14pt;} има едно правило. То се състои от селектора p и свойството font-size, на което е зададена стойност 14pt. Това правило ще направи размера на шрифта във всички параграфи 14 точки (2).

2.8.1. Задаване на стилове в HTML документ

Има три начина за задаване на стиловете:

- Вграден стил на елемента - Зададеният по този начин стил се нарича inline. Представлява записване на стилова информация в атрибута style на даден таг. Полученият стил е валиден директно само за елемента, на който е зададен (въпреки това негови поделементи могат да го наследят при определени условия). Този метод има редица ограничения, тъй като не позволява използването на CSS селектори (5).
- Блок със стилове в документа - Зададеният по този начин стил се нарича internal. Представлява блок със стилове, затворен в таг <style>. Този таг трябва да е поставен в заглавната част на HTML документа (тага <head>) (5).
- Файл със стилове - Зададеният по този начин стил се нарича external. Представлява самостоятелен файл, който се състои от стилове и към него е направено обръщение в HTML документа. Това е единственият

способ, който отговаря на идеята за отделяне на съдържание от оформление. Могат да бъдат вкарани неограничен брой файла, но е препоръчително броят им да е максимално ограничен, заради бавното зареждане на уеб страницата (5).

Указването на такъв файл се случва между <head> таговете и има следният синтаксис:

```
<link rel="stylesheet" href="../styles/styles.css">
```

2.8.2. Версии на CSS

Съществуват 3 основни версии на езика CSS + една, която още не е поддържана от основните браузъри. За всяка от тях е разработен специален тест, чрез който може да се провери дали даден web браузър поддържа съответната версия на езика. Всяка следваща версия разширява възможностите на езика. Могат да се използват и разширения на CSS (Sass, Less) или фреймуърци(Foundation Zurb), които улесняват работата и позволяват допълнителна функционалност като наследяване, задаване на променливи и др.

2.9. Javascript

JavaScript прави Web страницата по-динамичнаи привлекателна. Позволява на страницата да реагира на действията на посетителите, дава възможност за използване на специални ефекти (визуални и др.). JavaScript не се нуждае от компилатор (за разлика от Java) и е по- снизходителен в някои области като синтаксиса (19).

JavaScript е обектно-базиран скриптов език от страна на клиента, който се използва, за да се направят по-динамични Web страници. Обектно-базиран

означава, че могат да се използват елементи като обекти; от страна на клиента означава, че JavaScript се изпълнява от софтуера, който използва посетителя, а не Web сървъра на сайта, обслужващ тази страница.

2.9.1. Вмъкване на javascript

- **В същата страница**

JavaScript кодът може да се вмъква в HTML документа между двойката елементи <script> и</script>. Когато срещне тага <script>, браузърът разбира, че трябва да спре интерпретирането на HTML кода и да започне да обработва скрипта, намиращ се между <script> и </script>. Този скрипт не е задължително да бъде написан на JavaScript. Има и други езици за писане на скриптове, например VBScript. Но езикът по подразбиране е JavaScript (3).

Пример:

```
<!Doctype HTML>
<html>
<head>
<title>Вмъкване на javascript</title>
</head>
<body>
<script>
document.write("Hello World!");
</script>
```

</**body**>

</**html**>

От отделна страница

Другият начин да се зареди Javascript страница е като се постави в отделен файл. Първо създаваме файла Hi.js

document.write(“Hello World!”);

След което го вкарваме в html страницата по следния начин:

<script src="Hello.js"></script>

2.9.2.Стойности, типове и оператори

Във вътрешния свят на компютъра има само данни. Можете да четете данни, да променяте и създавате данни, но всичко, което не е данни просто не съществува. Всички тези данни се съхраняват като дълги поредици от битове и всичко е основано на това (3).

Бит е двустойностно нещо което обикновено се описва, като нули и единици. Във вътрешността на компютъра, те са под формата на по-висок или нисък електрически заряд, силен или слаб сигнал, лъскаво или матово място по повърхността на CD. Всяко парче от обособена информация може да се намали до последователност от нули и единици представени в битове.

Например, как може да се представи числото 13 в битове. То работи по същия начин, като десетичните числа, но вместо 10 различни цифри имаме само 2, а степента на всяка се увеличава с коефициент 2 от дясно на ляво. Това

са битовете, които съхраняват числото 13 със степента на цифрите показани по-долу от тях:

0	0	0	0	1	1	0	1
128	64	32	16	8	4	2	1

Това е бинарен номер 00001101 или 8+4+1, който се равнява на числото 13.

2.9.2.1.Променливи

Променливата представлява или съдържа стойност. Декларират се по следния начин: var име_на_променлива;

Присвояване на стойности на промеливи – извършва се със знака „=“, като може да се декларира променлива и да й се зададе стойност на един ред или да се декларира на един и по-надолу да й се зададе стойността, на друг ред. Всяко ново присвояване на стойност на променлива се записва на нов ред. В Javascript променливите са case sensitive, което значи че има значение дали се пишат главни или малки букви, например abc, ABC, Abc, AbC са различни променливи. Името на променливата може да съдържа латинска буква, цифра или долно тире, но не може да започва с цифра. Името не може да съвпада със запазена дума в Javascript (6).

2.9.2.2.Запазени думи

Abstract, arguments, await, Boolean, break, byte, case, catch, char, class, const, continue, debugger, default, delete, do, double, else, enum, eval, export, extends, false, final, finally, float, for, function, goto, if, implements, import, in, instanceof, int, interface, let, long, native, new, null, package, private, protected,

public, return, short, static, super, switch, synchronized, this, throw, throws, transient, true, try, typeof, var, void, volatile, while, with, yield, Array, Date, eval, function, hasOwnProperty, Infinity, isFinite, isNaN, isPrototypeOf, length, Math, NaN, name, Number, Object, prototype, String, toString, undefined, valueOf, getClass, java, JavaArray, javaClass, JavaObject, JavaPackage, alert, all, anchor, anchors, area, assign, blur, button, checkbox, clearInterval, clearTimeout, clientInformation, close, closed, confirm, constructor, crypto, decodeURI, decodeURIComponent, defaultStatus, document, element, elements, embed, embeds, encodeURI, encodeURIComponent, escape, event, fileUpload, focus, form, forms, frame, innerHeight, innerWidth, layer, layers, link, location, mimeTypes, navigate, navigator, frames, frameRate, hidden, history, image, images, offscreenBuffering, open, opener, option, outerHeight, outerWidth, packages, pageXOffset, pageYOffset , parent, parseFloat, parseInt, password, pkcs11, plugin, prompt, propertyIsEnum, radio, reset, screenX, screenY, scroll, secure, select, self, setInterval, setTimeout, status, submit, taint, text, textarea, top, unescape, untaint, window, onblur, onclick, onerror, onfocus, onkeydown, onkeypress, onkeyup, onmouseover, onload, onmouseup, onmousedown, onsubmit.

2.9.2.3.Типове променливи

Има шест основни типа променливи в JavaScript: numbers, strings, Booleans, objects, functions и undefined (4).

2.9.2.3.1.Numbers

Стойносттите на вида number са цифрови стойности. В програмите на JavaScript, се записват както следва:

13

Използвайки тази стойност в програма, ще доведе до моделът на битовете за числото 13, вътре в компютърната памет.

JavaScript използва фиксиран брой битове, а именно 64 от тях за съхранение на една цифрова стойност. Това са толкова модели колкото могат да се направят с 64 бита, което означава, че количеството на различните номера, които могат да бъдат представени е ограничен. За N десетични числа, сумата от числата, които могат да бъдат представени е 10N. По същия начин имайки в предвид 64-те бинарни цифри, можем да представим 264 различни номера, което е около 18 квинталиона (18 с 18 нули след него), а това е много.

Компютърната памет използва много по-малко и хората са склонни да използват групи от 8 и 16 бита, когато представят техните номера. Преди беше лесно случайно да се препълнят такива малки номера, които да не могат да се впишат в даден размер на бита. Днес, дори и персоналните компютри имат достатъчно памет, за да може свободно да се използват 64 битови парчета, което означава, че трябва да се притесняваме от препълване на паметта само, когато се занимаваме с наистина астрономически цифри.

Не всички цели числа под 18 квинталиона се вписват в номер на JavaScript, все пак. В битовите също се съхраняват и отрицателни числа, така че един бит се използва за да показва знака на числото. По-голям проблем обаче е, че и не-цели числа същото трябва да бъдат представени. За да се направи това някои от битовете се използват за съхранение на позицията на десетичната запетая. Действителното максимално цяло число, което може да се съхранява е от порядъка на 9 квадрилиона (15 нули), което все още е огромно.

Дробни числа се изписват с помощта на точка:

9.81

За много големи или много малки номера, може да използвате и научна нотация, чрез добавяне на “e” (за “exponent”) следвана от експонентата на броя:

2.998e8

Това е 2.998 × 108 = 299,800,000.

Изчисления с цели числа (наричани integers) по-малки от посочените по-горе 9 квадрилиона са гарантирани да бъдат винаги точни. За съжаление изчисленията с дробни числа обикновено не са. Точно, както π (пи) не може да бъде точно изразено от краен брой десетични цифри, така и много номера губят част от своята прецизност, когато само 64 бита са на разположение да ги съхраняват. Това е жалко, но причинява практически проблеми само в определени случаи. Важно е да сме наясно с това и да мислим за дробните числа като цифрови приближения, а не като точни стойности.

Специални номера

Има три специални стойности в JavaScript, които се смятат за числа, но не се държат като нормални такива.

Първите две са Infinity и -Infinity, които представляват положителна и отрицателна безкрайност. Infinity - 1 все още е Infinity и т.н. NaN означава “not a number” макар и да е стойност от типа номер. NaN се получава, например, когато се опитаме да разделим 0 / 0 (нула делено на нула), Infinity - Infinity или произволен брой цифрови операции, които не дават точен, смислен резултат.

2.9.2.3.2.Strings

Следващият основен тип данни е string. Той се използва за представяне на текст, който е написан в кавички.

“Hello World!”

‘Hello All!’

И двете единични и двойни кавички могат да се използват за отбелязване на *strings* , но не и двата вида едновременно.Текстовете могат да бъдат толкова дълги, колкото са отбелязани с кавички от началото до края на *string*.

Почти всичко може да се постави между кавичките и JavaScript ще направи *string* стойност от него. Но няколко характера са по-трудни. Може да си представите поставянето на кавички между кавичките, колко може да бъде трудно. Новите редове (характерите, които се получават, когато натиснете *Enter*) също не могат да се поставят между кавички. *String* трябва да остане на един ред (4).

За да може да се включат такива характери в *string*, се използва следната нотация: когато една обратно наклонена черта (\) се намира вътре в цитирания текст, това показва, че характера след нея има специално значение. Това се нарича escaping (бягство) на характера. Цитат, който се предхожда от обратно наклонена черта \ няма да сложи край на *string*, а е част от него. Когато се постави n характера след обратно наклонената черта \, това се тълкува като

знак за нов ред \n. По същия начин, t след обратно наклонена черта \ означава табулация \t.

String не могат да бъдат разделяни, умножавани или изваждани, но +оператора може да се използва върху тях. Той не събира, но конкатенира - слепва два *strings* заедно. Следващия ред ще произведе *string* - "concatenate":

"con" + "cat" + "e" + "nate"

2.9.2.3.3.Boolean values

Често, ще е необходима стойност, която да прави разграничение между две възможности, като “yes” и “no” или “on” и “off”. За това в JavaScript има Булев тип, който има само две стойности: вярно и невярно (които са написани просто като думи *true* и *false*).

Това е един от начините за произвеждане на булеви стойности:

```
console.log(5 > 1)
// → true
console.log(5 < 1)
// → false
```

> и < са символите за “по-голямо” и ”по-малко ”. Те са двукомпонентни оператори. Прилагането им води до булева стойност, която показва дали е верен този случай.

Strings може да се сравняват и по следния начин:

```
console.log("Ahtopol" < "Sozopol")
```

// → true

Начинът на подредба на strings е повече или по-малко азбучен: главните букви са винаги “по-малки”от малките такива, така че "Z" < "a" е вярно, други характери като (!, -, и т.н.) също са включени в подредбата. Действието на сравнението се основава на стандарта Unicode. Този стандарт определя номер за почти всеки характер, от който ще имате някога нужда, включително характери от Гръцки, Арабски, Японски и т.н. Такива номера са полезни за съхраняване на strings вътре в компютъра, защото така ще бъде възможно да се представят, като последоватленост от цифри. При сравняване на strings, JavaScript ги проверява от ляво на дясно, като се сравняват цифровите кодове на характерите един по един (4).

Други подобни: >= (по-голямо или равно), <= (по-малко или равно), == (равно), и != (различно).

Има само една стойност в JavaScript, която не е равна на себе си, това е NaN, което означава “not a number”.

```
console.log(NaN == NaN)
// → false
```

С NaN се означава резултата от безсмислено изчисление и като такова не е равно на резултата от други безсмислени изчисления.

2.9.2.3.4. Undefined values

Има две специални стойности, null и undefined, които се използват за обозначаване на липсата на смислена стойност. Те самите са стойности, но не носят информация.

Много операции в езика, които не произвеждат смислена стойност дават undefined просто, защото те трябва да върнат някаква стойност.

Разликата в значението между undefined и null е инцидент в дизайна на JavaScript, но това няма значение през по-голямата част от времето (3).

Автоматично преобразуване на типа

```
console.log(8 * null)
// → 0
console.log("5" - 1)
// → 4
console.log("5" + 1)
// → 51
console.log("five" * 2)
// → NaN
console.log(false == 0)
// → true
```

Когато се прилага оператор върху “неправилен” тип стойност, JavaScript тихо ще преобразува тази стойност до типа, който иска с помощта на набор от правила, които често не са това, което искате или очаквате. Това се нарича

корекция на тип. Така че null в първия израз става 0, а "5" във втория става 5 от string в номер. И все пак в третия израз, + опитва да конкатенира номер към string, така че 1 се превръща в "1" (от номер в string).

Когато нещо, което не се вписва в картата на номерата по очевиден начин (като "five" или undefined) се преобразува в номер и се произвежда NaN. Други аритметични операции върху NaN пазят резултата NaN, така че ако откриете един от тези изрази на неочаквано място, погледнете за случайно преобразуване на типа.

При сравняване на стойности от един и същи тип с използване на ==, резултатът е лесно предвидим: трябва да е вярно, когато и двете стойности са едни и същи, с изключение на случаите с NaN. Но когато типовете се различават, JavaScript използва сложни и объркващи набори от правила, за да реши, какво да прави. В повечето случаи просто се опитва да преобразува една от стойностите в типа на другата стойност. Въпреки това, когато null или undefined са от двете страни на оператора, той произвежда вярно само ако и двете стани са от един и същи тип null или undefined.

```
console.log(null == undefined);
// → true
console.log(null == 0);
// → false
```

Поведението в последния код често е полезно. Когато трябва да се провери дали дадена стойност е реална, вместо null или undefined, може просто да се сравни null с == (или с !=) оператора.

Правилата за преобразуване на *string* и числа до Булеви стойности посочват, че 0, NaN и празен *string* ("") се отчитат, като false, а всички останали стойности като true. Поради това, изрази като 0 == false и "" == false също са верни. За случаи, като този, когато не искаме автоматично преобразуване на типа, има два допълнителни оператора: === и !==. Първият тества дали дадена стойност е точно равна на другата стойност, а другия дали е точно различна. Така "" === false е false, както се очаква. Препоръчително е използването на три знаковите оператори за сравнение, като защита за предотвратяване на нежелано преобразуване на типа. Но ако е сигурно че типовете от двете страни са еднакви, няма проблеми да се ползват по-късите оператори.

2.9.2.3.5. Switch

Конструкцията switch се използва за извършване на различни действия въз основа на различни условия.

- Синтаксис

```
switch(expression) {
case n:
    code block
    break;
case n:
    code block
```

```
        break;
    default:
        default code block
    }
```

2.9.2.3.6. Примери

```
<!DOCTYPE html>
<html>
<body>

<p id="test"></p>

<script>

var day;
switch (new Date().getDay()) {
    case 0:
        day = "Неделя";
    break;
    case 1:
        day = "Понеделник";
    break;
    case 2:
        day = "Вторник";
    break;
    case 3:
        day = "Сряда";
    break;
    case 4:
        day = "Четвъртък";
    break;
```

```
    case 5:
        day = "Петък";
    break;
    case 6:
        day = "Събота";
}
document.getElementById("test").innerHTML = "Днес е " + day;

</script>

</body>
</html>
```

Да се създадат 3 променливи A, B и C с числови стойности. Да се провери дали C е между A и B. Да се изведат числата заедно с подходящо съобщение за това дали C е между A и B.

```
var A = Math.random()*100;
var B = Math.random()*100;
var C = Math.random()*100;
var mejduAiB = ((A<C) && (B>C)) || ((A>C) && (B<C));
console.log(A);
console.log(B);
console.log(C);
console.log("C е между A и B  "+mejduAiB);
```

```
68.3401854163457
39.390496192974766
57.43560622834176
C е между A и B  true
```

Да се напише програма, която на random показва дата и да се изчисли коя дата е следващия ден.

```
var d = 0;
var m = Math.round(Math.random() * 11 + 1);
var y = Math.round(Math.random() * 2017);
var nextD = 0;
var nextM = 0;
var nextY = 0;

if (m == 2) {
    if((y % 4 == 0) && (y % 400 == 0) || (y % 100 !== 0)){
    d = Math.round(Math.random()*28+1);
    if (d==29){
        nextD = 1;
        nextM = m + 1;
        nextY = y;
    }else{
        nextD = d + 1;
        nextM = m;
        nextY = y;
    }
}
}else{
    if (y % 100 == 0){
    d = Math.round(Math.random()*27+1);
    if (d == 28){
        nextD = 1;
        nextM = m + 1;
        nextY = y;

    }
    }
}
if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m == 10 || m == 12) {
```

```
    d = Math.round(Math.random()*30+1);
    if (d == 31){
        nextD = 1;
        nextM = m + 1;
        nextY = y;

        if (m = 12) {
            nextD = 1;
            nextM = 1;
            nextY = y + 1;
        }
    } else {
        nextD = d + 1;
        nextM = m;
        nextY = y;
    }
}
if (m == 4 || m == 6 || m == 9 || m == 11){
    d = Math.round(Math.random()*29+1);
    if (d == 30){
        nextD = 1;
        nextM = m + 1;
        nextY = y;

    } else {
        nextD = d + 1;
        nextM = m;
        nextY = y;
    }
}

console.log("Днес е " +d+"." +m+ "." +y);
console.log("Утре ще е " +nextD+ "." +nextM+ "." +nextY);
```

В първото условие проверяваме ако месеца е февруари и ако годината е високосна т.е /дели се на 4 или 400 и не се дели на 100/, и ако деня е 29, за следващ ден да се смята първия ден от месец март. Годината ако не е високосна /т.е има 28 дни/ за следващ се смята първия от месец март.

Във второто условие проверяваме първо месеците, ако са януари, март, май, юли, август, октомври и ноември и денят ако е 31, следващия ден ще бъде 1ви от следващия месец. Ако пък месеца е 12 и деня е 31, увеличаваме годината с 1 и следващия ден ще е от януари.

С третото последно условие проверяваме четните месеци /април, юни, септември, ноември/ ако денят е 30, за да получим следващия ден, увеличаваме месеца с 1 и получаваме първия ден от следващия месец.

Накрая изкарваме на екрана кой е днешния ден /изчислено е на random и кой ще бъде следващия ден/.

2.9.2.4. Цикли

Чрез цикли, може да изпълним даден блок от код няколко пъти без да повтаряме кода. Те са едно от основните неща в програмирането (4).

2.9.2.4.1. While и Do цикли

```
var num = 0;

while (num <= 10) {

    console.log(num);

    num = num + 2;
```

}

В изявлението се започва с ключовата дума while, която създава цикъл. Думата while е последвана от израз, подобен на if в скоби и след това от изявление в фигурни скоби. Цикълът изпълнява това изявление докато израза произвежда стойност, която е *true*, когато се конвентира в булев тип.

В този цикъл искаме да отпечатаме и двете, текущия номер и добавянето на две към нашата променлива. Всеки път, когато трябва да изпълним няколко изявления вътре в цикъл, ние ги увиваме във фигурни скоби {}. Фигурните скоби направляват изявленията, както иска изразът в обикновените скоби: те ги групират заедно, правейки ги, като едно изявление. Поредицата от изявления поставени във фигурни скоби се нарича блок.

Променливата num показва начина, по който с дадена променлива може да следите напредъка на програмата. Всеки път когато цикълът се повтаря, num се увеличава с 2. След това, в началото на всяко повторение, тя се сравнява с броя 10 за да реши дали програмата е изпълнила цялата работа, която е възнамерявала да направи.

Пример:

```
var count = 10;
while (count>=1){
process.stdout.write(" "+count);
    count--;
}
```

Извеждаме числата в намаляващ вид, от 10 до 1

```
10 9 8 7 6 5 4 3 2 1
```

2.9.2.4.2. Цикъла For

Скобите, след ключовата дума for, трябва да съдържат две точки и запетя. Частта преди първата запетая инициализира цикъла, обикновено чрез дефиниране на променлива. Втората част е израз, който проверява дали цикъла трябва да продължи. В последната част се актуализира състоянието на цикъла след всяка итерация (6).

Пример1:

```
for (var x=1; x<=100; x++){
    process.stdout.write(" "+x);
}
```

Изкарва на екрана всички числа от 1 до 100:

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100
```

Пример2:

```
for (var x=-10; x<=10; x++) {
    if ((x%2 == 1) || (x%2 == -1)) {
    process.stdout.write(" "+x);
    }
}
```

Изкарва на екрана всички нечетни числа между -10 и +10

```
-9 -7 -5 -3 -1 1 3 5 7 9
```

2.9.2.5. Обекти и масиви

2.9.2.5.1. Масиви

JavaScript осигурява начин специално за съхранение на поредици от стойности. Той се нарича масив (array) и е написан, като списък от стойности между квадратни скоби, разделени със запетаи (4).

```
var listOfNumbers = [2, 3, 5, 7, 11];

console.log(listOfNumbers[1]);

// → 3

console.log(listOfNumbers[1 - 1]);

// → 2
```

Означението за получаване на елементите вътре в масива също използва квадратни скоби. Чифт квадратни скоби веднага след израз с друг израз вътре в тях, вижда елементите в израза от лявата страна, които съответстват на индекса даден от израза в скобите.

Първият индекс в масива е нула, не едно. Така че, първият елемент може да се прочете с listOfNumbers[0]

- .push(); - добавя елемент най-отзад в масива и преиндексира,
- .pop(); - премахва последния елемент от масива и преиндексира,
- .shift(); - премахва първия елемент от масива и преидексира,
- .unshift(); - вмъква елементи в началото на масива и преиндексира,
- .slice(); - прави копие на масива, като оригиналния се запазва цял,
- .splice(); - може да трие, заменя и вмъква елементи в масива,

- indexOf(); - показва на кое място се намира символа в масива, като ако няма такъв символ връща -1, а ако има няколко, показва само първото място, на което го открива,
- lastIndexOf(); - показва само последното срещане на елемента от масива,
- .sort(); - сортира азбучно, използва се само за стрингове,
- .toString(); - преобразува масива в string,
- .join(); - също преобразува масива в стринг, но тук при него може да се сложи какъв символ искаме за разделител,
- .concat(); - обединява два масива в един.

Пример 1:

```
var arr = [1, 2, 3, 4, 5];


var otricatelni = false;


for (var index = 0; index < arr.length; index++) {
        if (arr[index] < 0) {
            otricatelni = true;
            break;
        }
    }


otricatelni ? console.log("Има отрицателни") : console.log("Всички са
положителни");
```

Проверяваме дали в масива има отрицателни числа

```
Всички са положителни
```

Пример 2:

```
var arr = [4, 1, 1, 4, 2, 3, 4, 4, 1, 2, 4, 9, 3];

arr.sort();

var currLength = 1;

var maxLength = 1;

var element;

var count = 1;


for (var index = 1; index < arr.length; index++) {

    if (arr[index] == arr[index - 1]) {

        currLength++;


        if (currLength > maxLength) {

            maxLength = currLength;

            element = arr[index];

            count++;

        }

    } else {

        currLength = 1;

    }

}

console.log("Елемент " + element + " се среща " + count + " пъти.");
```

Задачата е да проверим кой елемент се среща най-много пъти в масива и колко точно пъти се среща.

```
Елемент 4 се среща 5 пъти.
```

Пример 3:

```
var masiv = [[1, 2, 3, 4], [11, 5, 3, 1]];
var sum = 0;
var countOfElements = 0;
for (var row = 0; row < masiv.length; row++) {
    for (var col = 0; col < masiv[row].length; col++){
        sum += masiv[row][col]
        countOfElements++
    }
}
 var srednoaritmetichno = sum/countOfElements
    console.log('Сумата на всички числа в масива е ' + sum);
    console.log('Средноаритметичната стойност на числата е ' +
srednoaritmetichno);
```

```
Сумата на всички числа в масива е 30
Средноаритметичната стойност на числата е 3.75
```

2.9.2.5.2. Обекти

Стойности на типа обект са произволни колекции от свойства и ние можем да добавяме или премахваме тези свойства, както преценим. Един от начините за създаване на обект е с помощта на фигурни скоби (4).

```
var Human = new Object();
    firstName:"Иван",
    lastName:"Петров",
    age:50,
    city:"София"
};
```

Вътре във фигурните скоби, може да дадем списък от свойства, разделени със запетаи. Всяко свойство е написано като име, следвано от двуточие, следвано от израз, който осигурява стойност на свойството. Интервали и нови редове не са от значение. Четене на свойство, което не съществува генерира стойност undefined.

Възможно е да се присвои стойност на свойство с оператора =. Това ще замени стойността на свойството, ако тя вече съществува или ще създаде ново свойство в обекта, ако той не го е направил.

Оператора == в JavaScript се използва, когато се сравняват обекти и ще върне true само ако и двата обекта са точно с една и съща стойност. Сравнявайки различни обекти, ще върне false, дори ако те имат идентично съдържание. Няма оператор за "дълбоко" сравнение, вграден в JavaScript, който разглежда съдържанието на обекта, но е възможно да го напишем сами.

Обекти и масиви (които са специфични видове обекти) осигуряват начини за групиране на няколко стойности в една единствена стойност. Концептуално, това ни позволява да сложим един куп свързани неща в един

пакет, вместо да се опитваме да приключим всички отделни неща и да ги ползваме по отделно.

Обектите могат да служат и като карти, за асоцииране на стойностти с имена. Оператора in може да се използва , за да се разбере, дали даден обект съдържа свойство с дадено име. Същата ключова дума може да се използва и за цикъл (for (var name in object)), който минава над свойствата на даден обект.

2.9.2.6. Функции

Дефинирането на функция, е като дефиниране на променлива, където стойността дадена на тази променлива се случи да е функция. Например, следния код дефинира променливата square, която се отнася към функция, която произвежда квадрата на дадено число:

```
var square = function(x) {
  return x * x;
};
console.log(square(12));
// → 144
```

Функцията се създава с израз, който започва с ключовата дума function. Функциите имат набор от параметри (в този случай само x) и тяло, което съдържа изявления, които трябва да се изпълнят, когато функцията се извика. Тялото на функцията трябва винаги да е увито в скоби, дори ако то се състои само от едно изявление (както в предишния пример).

Всеки път, когато дадена функция се извиква, специална променлива наречена arguments се добавя към средата, в която тялото на функцията се изпълнява. Тази променлива се отнася към обект, който притежава всички аргументи подадени на функцията. В JavaScript е позволено да се подават повече (или по-малко) аргументи към функция от броя на параметрите, които функцията обявява.

Обекта arguments има свойство length, което ни казва броят на аргументите, които се подават на функцията. Тя също има свойство за всеки аргумент, 0, 1, 2 и т.н.

Обектите много приличат на масивите, но нямат никакви методи като масивите /като slice или indexOf/, така че е малко по-труден за използване от истински масив.

Пример 1:

```
function arrayGenerator(length) {
    var arrGen = new Array(length);
    for (var index = 0; index < length; index++) {
        arrGen[index] = Math.floor(Math.random() * 50);
    }
    return arrGen;
}
console.log(arrayGenerator(10));
```

С примера за тази функция генерираме масив, който е на random принцип като максималната възможна стойност на число от масива е 50.

```
[ 29, 11, 6, 14, 25, 24, 11, 25, 45, 12 ]
```

Пример 2:

```
function howManyTimes(arr, num) {
    var count = 0;
    for (var index = 0; index < arr.length; index++) {
        if (arr[index] === num) {
            count++;
        }
    }
    return count;
}
var arr = [1, 2, 2, 5, 2, 6, 8, 2, 4, 2];
console.log(howManyTimes(arr, 2));
```

Тази функция показва, кое число колко пъти се среща в масив, който ние въвеждаме. В случая това ни е масива :

```
var arr = [1, 2, 2, 5, 2, 6, 8, 2, 4, 2];
```

а числото, което искаме да видим колко пъти се среща е: 2, следователно

отговора е: 5

Пример 3:

```
var arr = [1, 20, 3, 44, 5, 17, 2, 64, 11, 29];
function sortBy(x, y) {
    return x - y;
}
console.log(arr.sort(sortBy));
```

Подреждаме числата в нарастващ ред

```
[ 1, 2, 3, 5, 11, 17, 20, 29, 44, 64 ]
```

3.Браузъри

Браузърите са приложен софтуер, предназначен за разглеждане на информационни ресурси в Уеб. Всеки информационен ресурс притежава стандартизиран адрес Унифициран локатор на ресурси (URI/URL) и може да бъде уеб сайт, изображение, видеоклип или друго съдържание. Процесът на придвижване от един ресурс към друг, обикновено следвайки хипервръзки, се нарича уеб навигация (8).

3.1. История

Първият браузър и редактор е разработен от Тим Бърнърс-Лий заедно с белгиеца Робер Кайо към края на 1990 г. по време на изпълнението на проекта за информационна система, съчетаваща хипертекстови документи с достъп до разпределени компютърни мрежи. Първоначално браузърът се нарича WorldWideWeb, но по-късно е преименуван на Nexus, за да не се обърква с World Wide Web (Уеб), както става известна информационната система. Нейните основните компоненти включват още: мрежов протокол HTTP, маркиращ език HTML,уеб страници и уеб сървър.

3.2. Протоколи и стандарти

Браузърът се свързва с уеб сървър основно използвайки HTTP протокол за разглеждане на уеб страници.

URL (uniform resource locator – универсален указател на ресурсите) е адресът на някоя страница започваща обикновено с http: за HTTP достъп (има също и криптирана връзка започваща с https). Различните браузъри поддържат и други URL протоколи, например ftp: за FTP (file transfer protocol), rtsp: за RTSP (real-time streaming protocol).

Програмният език за уебстраница е HTML (hyper-text markup language), който се съчетава с HTTP протокол, ползващ MIME стандарт. Най-общо браузърите поддържат изобразяването на изображения и текст. Основните файлови формати за картинки са JPEG, PNG и GIF. Поддръжката на други файлови формати може да се добави чрез плъгини (добавки) в браузърите. Възможно е уеб страницата да съдържа анимация, видео, звук и поточна информация.

Ранните браузъри поддържат само много проста версия на HTML. Съвременните браузъри де факто представляват съчетание на HTML и XHTML. Проблемът на съвременните браузъри е, че не спазват стандарта, разработен от работната група W3C и затова често се случва една и съща уебстраница да се показва по различен начин на различните браузъри. Много сайтове са проектирани чрез WYSIWYG HTML редактори, които правят уебстраниците, така че често не спазват стандарта HTML по подразбиране и това затруднява работата на работната група W3C в разработването на стандарти, спецификации с XHTML и CSS (cascading style sheets, използвани за теми на страницата).

Някои браузъри включват допълнителни компоненти за поддръжка на Usenet новини, IRC (Internet relay chat) и електронна поща. Протоколите могат да поддържат включително NNTP (network news transfer protocol), SMTP (simple mail transfer protocol), IMAP (Internet message access protocol) и POP (post office protocol пощенски офис протокол).

3.3. Видове браузъри

3.3.1.Internet Explorer

Уеббраузър на компанията Майкрософт, включен във всяка операционна система на Windows от 1995.След първото пускане на Windows 95, още няколко версии на браузъра били разработени за различните операционни системи: Internet Explorer за Mac и Internet Explorer за UNIX (17).

3.3.2.Mozilla Firefox

Браузър с отворен код, създаден от доброволни сътрудници на фондацията Mozilla, известен в тестовата си фаза като Финикс и Mozilla Firebird. Базиран е на кода на Mozilla Application Suite (14).

Идеята на създателите му е сложността на ползване и размерът на програмата да се намали до минимум, като се премахнат програмите за електронна поща, нюзгрупи и HTML редактора, които са част от Mozilla Application Suite. Поради това, както и поради ефективното и бързо възпроизвеждане на уебстраници и многото възможности за разширения на функциите (написани също от доброволци), Firefox придобива значителен успех и постепенно взема част от пазарния дял на Internet Explorer.

Съществуват версии за Windows, Mac OS X, GNU/Linux, OS/2, FreeBSD, Solaris и други операционни системи.

3.3.3.Opera

Най-голямата норвежка телекомуникационна компания, Telenor, поставя началото на развитието на програмата като изследователски проект през 1992 г. По-късно създадената за целта норвежка компания Opera Software, поема отговорността за разработката, маркетинга и продажбите на нови версии на програмата.

Opera е многоплатформен браузър – т.е. приложението може да работи под различни операционни системи, включително Microsoft Windows, Mac OS, Solaris, FreeBSD и GNU/Linux системи.

Освен за персонални компютри Opera се предлага и във версии за цифрови и мултимедийни устройства, снабдени с малки дисплеи, като мобилни телефони, смартфони, персонални асистенти и игрални конзоли като Wii и PlayStation 3 (21).

3.3.4.Safari

Уеб браузър, разработен от Apple Inc. и включен в нейните операционни системи Mac OS X и iPhone OS. Първата публична бета-версия е пусната на 7 януари 2003 г. за операционната система Mac OS X на Apple. Тя започва да се предлага, като стандартен браузър, в пакета от програми на Mac OS X след версия 10.3 („Panther“) и заменя преди това използвания Internet Explorer за Mac. Safari също е вграден браузър и в iOS. От 11 юни 2007 г. се предлага и за Windows XP, Windows Vista и Windows 7 (25).

3.3.5.Google Chrome

Безплатен уеб браузър на компанията Google. Софтуерът е създаден чрез продукта с отворен код Chromium и дава възможност за гледане на уебстраници като thumbnails („тъмбнейли“, миниатюрни изображения).

Според Google браузърът е създаден, за „да направи работата в интернет по-бърза, по-лесна и по-безопасна, с минимален дизайн, който не се натрапва“.

Google Chrome, както някои други съвременни браузъри, има възможност за т. нар. „инкогнито“ режим, при който на страниците, посетени от потребителя, бисквитките и всякаква друга информация, свързана с тях, не се запазва на компютъра на потребителя, а се изтрива след затварянето на програмата (15).

4.Бази данни

Базата данни представлява специализирана информационна среда за съхраняване и обработка на голямо количество данни в конкретна предметна област,които са структурирани(взаимно-свързани) по определен начин. Неразделна част от базата данни са програмните средства, които поддържат и управляват информационната среда. Тази организирана съвкупност от програми се нарича система за управление на база данни (СУБД) (7).

4.1.Основни обекти

4.1.1. Таблици

Информацията се съхранява в таблици, всеки ред на които представлява един запис, а колоните са негови полета.

Всички записи съдържат еднакъв брой полета с еднотипни данни.Тези данни могат да бъдат числови, текстови, дати , време и т.н , но не могат да се изчисляват на базата на други полета от таблицата.

Едно или няколко полета в таблицата се определят за ключови.Чрез ключовите полета се осъществяват връзки между различни таблици в една база от данни;

4.1.2.Форми

Това са формуляри, бланки за попълване на данни в таблицата.Чрез тях може да се извеждат само тези полета, в които ще се актуализира информацията във формат, удобен за потребителя;

4.1.3.Заявки

По зададени критерии може да се получи извадка, част от наличните данни в свързаните таблици от базата данни;

4.1.4.Отчети

Предназначени са за отпечатване на данни, които се съдържат в таблици или са подбрани чрез заявки. В отчетите могат да бадат включени и допълнителни обекти за по- добра илюстрация на материала. Включването на графични обекти и използването на редица форматиращи техники дава възможност за представяне на информацията в добър естетически вид;

4.1.5. Макроси

С тях се автоматизират повтарящи се операции;

4.1.6. Модули

За автоматично управление на определени операции могат да бъдат създадени програмни модули, програмирани на езици, специлизирани за работа с бази данни.

4.2.Видове СУБД

4.2.1. Microsoft SQL Server

Сървърна система за управление на бази от данни (и по-точно, на релационни бази от данни) на компанията Microsoft. Microsoft SQL Server е предназначена за управление на големи сървърно базирани БД за разлика от MS Access която е desktop базирана и не е предназначена за управление на големи корпоративни БД.(1)

4.2.2. Oracle Database

Представлява СУБД, основен продукт от фамилията, реализиран съгласно архитектурата клиент – сървър. Поддържа релационен и обектно-релационен модел на данните, като едно от важните му разширения е възможността за съхраняване и обработка на структурирани и неструктурирани XML данни. Oracle-DB е проектиран да работи с множество конкурентни потребители и да поддържа едновременен достъп до множество независими една от друга БД с висока производителност, богата функционалност, надеждност на съхранението и сигурност на данните.(1)

4.2.3.MySQL

Най-популярната система за управление на бази данни с отворен код. SQL е съкращение от Structured Query Language (Структуриран език за заявки), предназначен за създаване, обработка и четене на бази данни, които представляват пакети от свързана информация, съхранявана в таблици.

MySQL поддържа интерфейси за програмиране под множество езици - C/C++, Eiffel, Java, Perl, Python, но най-често се използва в комбинация с езика PHP (1).

4.2.4.IBM Informix

Един от лидерите в СУБД. Осигурява висока надеждност, способност за работа с голям брой потребители и огромно натоварване. Той е комерсиален продукт, работещ в повечето случаи на UNIX операционни системи.

IBM Informix Dynamic Server и IBM DB2 са базите данни на IBM, предназначени и оптимизирани за направата на приложения, работещи с база от данни и имащи високи изисквания към надеждността и мощността. Възможността за надграждане позволява инсталирането на тези бази като се започне от персоналните компютри и се стигне до широка и мощна многопроцесорна архитектура, от приложение за един потребител до терабайтови масиви от данни.(1)

4.2.5.Ingres

Система за управление на релационни бази данни, предназначена за подпомагане на големи търговски и правителствени приложения. Ingres предлага функционални възможности, изисквани от всяка една база от данни от корпоративен клас. Тя е лидер в поддръжката на критични бизнес приложения и подпомагането на управлението, дори и на най-взискателните

корпоративни приложения на повече от 500 компании. Фокусирана върху сигурността, надеждността, мащабируемостта и лекотата на използване, Ingres съдържа функции изисквани от всяко едно предприятие, като успява да ги съчетае с гъвкавостта на отворения код. Ingres е чудесен инструмент за намаляване на разходите, породени от инвестирането на пари в неподходящи софтуерни продукти и загубеното време в търсене на тяхна алтернатива, която да осигури желания ефект от фирмата.(1)

4.2.6.FireBird

Система за управление на бази от данни, която работи на Linux, Windows, както и на разновидности на Unix. Базата данни е разцепена от отворения код на Borland през 200г., но кода на Firebird 1.5 е до голяма степен пренаписан.(1)

4.2.7.MongoDB

Класифициран като No-SQL база данни, Mongo DB избягва от употребата на традиционната таблица, на която са базирани релационните бази данни. MongoDB е документно-ориентирана база данни, поддържаща JSON, динамични заявки, индекси, репликация, map/reduce, профилиране за оптимизация на заявките, по подобие на SQL. Разработена е на C++. Разпространява се под комбинация на GNU Affero General Public License и Apache License, MongoDB е свободен софтуер с отворен код.

Работи бързо и надеждно върху един или няколко сървъра /устойчивост на данните се постига при повече от един сървър/. В някои случаи MongoDB се справя по-добре върху единичен сървър, отколкото MySQL. Функционира в стабилна версия, за разлика от повечето от конкурентните бази данни.(1)

4.2.8.SQL Azure

Microsoft Windows Azure SQL Database е базирана на облачни технологии релационна база данни, изградена посредством технологиите на SQL Server. Използвайки Windows Azure SQL база данни, можем лесно да предоставим и разработим релационни решения за бази данни в облака и да се възползваме от организиран център за данни, който осигурява надеждност от корпоративен клас, мащабируемост и сигурност с предимствата на вградената защита на данните и self-healing.

Windows Azure SQL база данни е релационна база данни за платформата Windows Azure, която е отворена и гъвкава cloud платформа, осигуряваща възможност за бързо изграждане, внедряване и управление на приложения през глобалната мрежа от центрове за данни на Microsoft. Можем да създадем приложения, използвайки всеки език, инструмент или framework.(1)

III. Разработка на уеб сайт

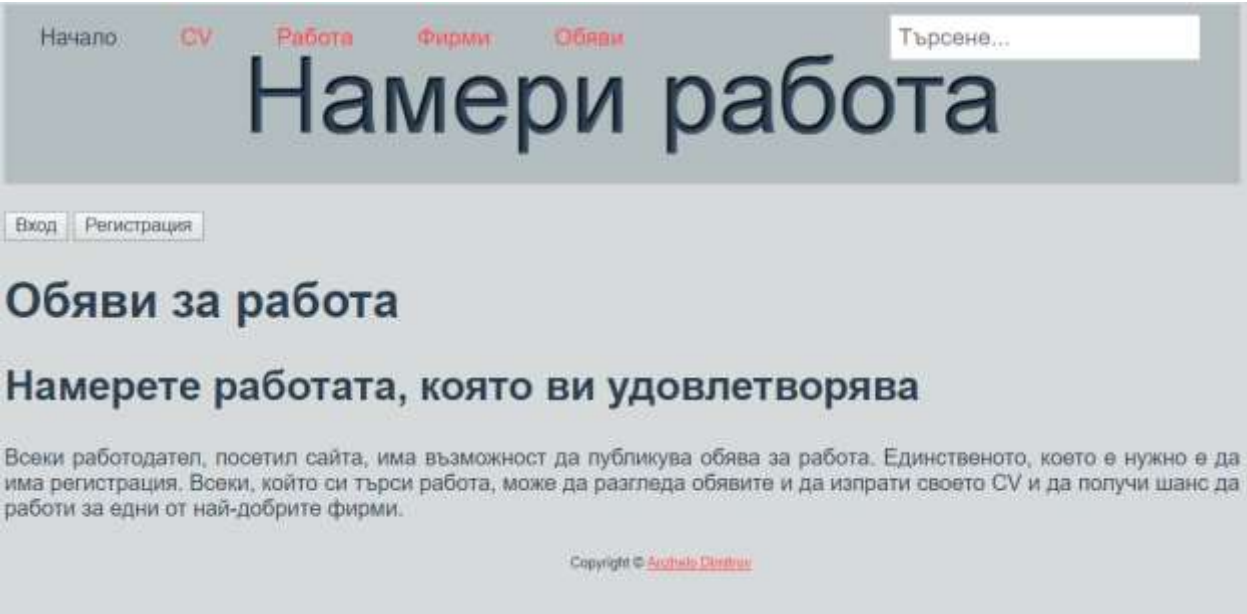
1.Начална страница

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Намери работа</title>
    <link rel="stylesheet" href="./styles/styles.css">
    <script src="javascript/javascript.js"></script>
</head>
<body>
    <header class="container">
        <form>
            <input id="search" type="text" placeholder="Търсене...">
        </form>
        <button class="menu-toggle" id="drop-down-btn">
            <i class="fa fa-bars" aria-hidden="true"></i>
        </button>
        <ul id="menu">
            <li>Начало</li>
            <li><a href="./cv.html">CV</a></li>
            <li><a href="./work.html">Работа</a></li>
            <li><a href="./firms.html">Фирми</a></li>
            <li><a href="./ad.html">Обяви</a></li>
        </ul>
    </header>

    <div class="modal-container">
```

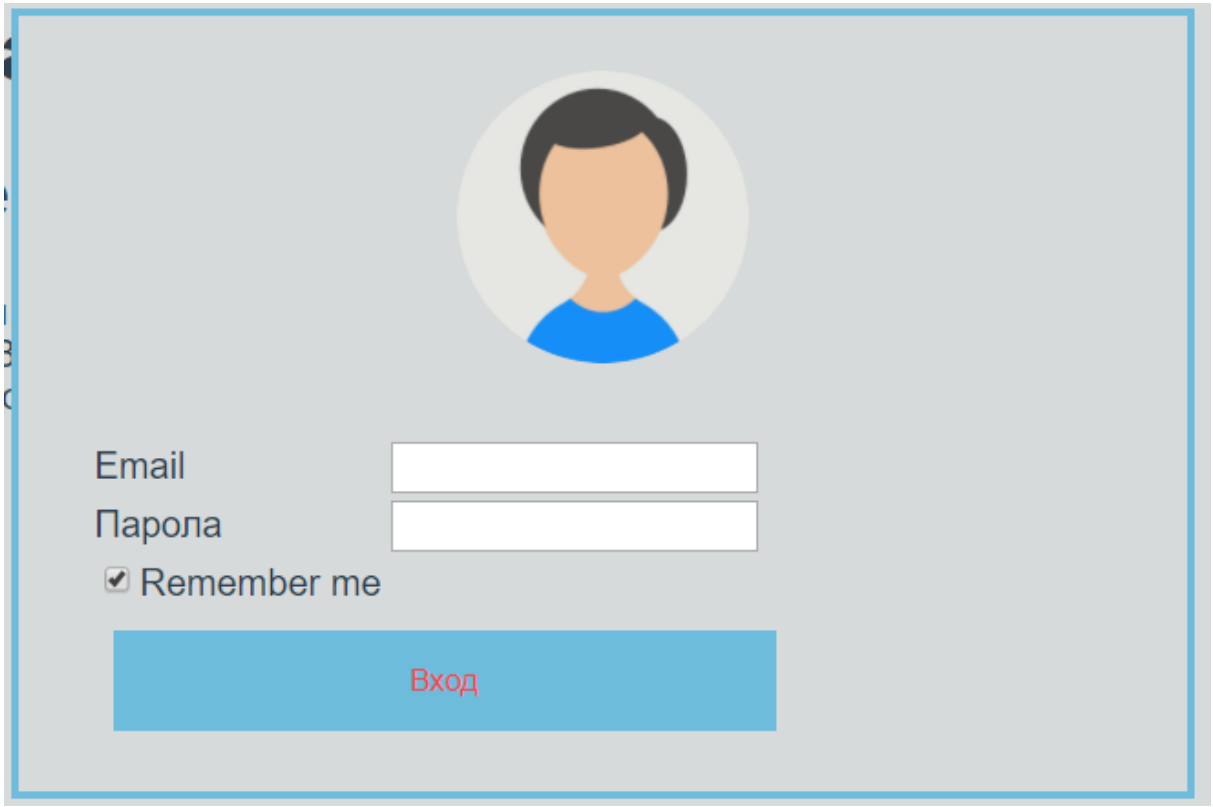
```
        <button
onclick="document.getElementById('enter').style.display='flex'">Вход</button>
        <div id="enter" class="modal">
            <div class="modal-content">
                <div class="modal-container">
                    <span
onclick="document.getElementById('enter').style.display='none'" class="display-
topright">&times;</span>
                    <form id="login-form">
                        <div class="imgcontainer">
                            <img src="./assets/avatar.png" alt="Avatar"
id="avatar">
                        </div>
                        <table class="login-container">
                            <tr>
                                <td><label for="email">Email</label></td>
                                <td><input type="text" id="email-login"
required="required"></td>
                            </tr>
                            <tr>
                                <td><label for="pass">Парола</label></td>
                                <td><input type="password" id="password-login"
name="pass"></td>
                            </tr>
                            <tr>
                                <td><input type="checkbox"
checked="checked">Remember me</td>
                            </tr>
                            <tr>
                                <td colspan="2"><input type="submit" id="login-
btn" value="Вход"></td>
                            </tr>
                        </table>
                    </form>
                </div>
```

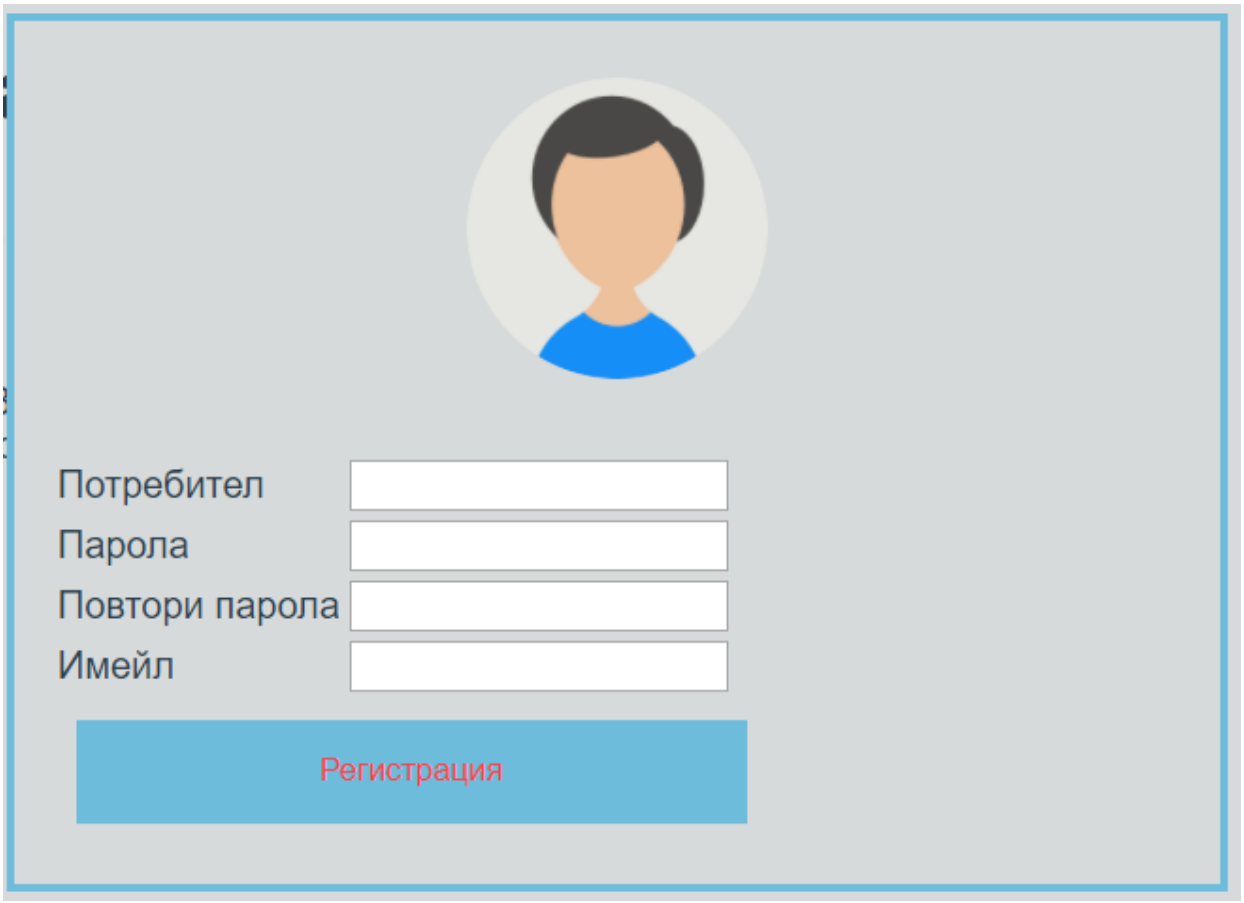
```
            </div>
        </div>
    <div class="modal-container">
            <button
onclick="document.getElementById('register').style.display='flex'">Регистрация</b
utton>
            <div id="register" class="modal">
                <div class="modal-content">
                    <div class="modal-container">
                        <span
onclick="document.getElementById('register').style.display='none'"
class="display-topright">&times;</span>

                        <form id="register-form">
                            <div class="imgcontainer">
                                <img src="./assets/avatar.png" alt="Avatar"
id="avatar">
                            </div>
                                <table class="register-container">
                                    <tr>
                                        <td><label
for="username">Потребител</label></td>
                                        <td><input type="text" id="user"
required="required"></td>
                                    </tr>
                                    <tr>
                                        <td><label
for="password">Парола</label></td>
                                        <td><input type="password" id="password-
register" name="pass"></td>
                                    </tr>
                                    <tr>
                                        <td><label for="repeatPassword">Повтори
парола</label></td>
```

```
                                        <td><input type="password"
id="repeatPassword" name="repeatPass"></td>
                                    </tr>
                                    <tr>
                                        <td><label for="email">Имейл</label></td>
                                        <td><input type="email" id="email-
register" name="mail"></td>
                                    </tr>
                                    <tr>
                                        <td colspan="2"><input type="submit"
id="register-btn" value="Регистрация"></td>
                                    </tr>
                                </table>
                            </form>
                        </div>
                    </div>
                </div>
            </div>

    <h1>Обяви за работа</h1>

    <h2>Намерете работата, която ви удовлетворява</h2>
        <p>
            Всеки работодател, посетил сайта, има възможност да публикува обява
за работа. Единственото, което е нужно е да има регистрация.
            Всеки, който си търси работа, може да разгледа обявите и да изпрати
своето CV и да получи шанс да работи за едни от най-добрите фирми.
        </p>
    <footer id="footer">Copyright &copy; <a
href="mailto:anzh.dimitrov@gmail.com">Anzhelo Dimitrov</a></footer>
</body>
</html>
```



На бутоните „Вход“ и „Регистрация“ е добавена финкционалността, да се отваря прозорец, съответно за да може потребителя да си влезе в профила или ако няма профил, да се регистрира.





2. Фирми

Сайта е настроен да се вижда нормално за всички устройства, като на страниците е намален. Добавен е бутон за менюто, чрез който при кликване се показва. При кликване на снимка на фирма, директно препраща към страницата на избраната фирма.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Намери работа</title>
    <link rel="stylesheet" href="./styles/styles.css">
    <script src="javascript/javascript.js"></script>
</head>
<body>
    <header class="container">
```

```
        <form>
            <input type="text" placeholder="Search..">
        </form>
        <button class="menu-toggle" id="drop-down-btn">
            <i class="fa fa-bars" aria-hidden="true"></i>
        </button>
        <ul id="menu">
            <li><a href="./index.html">Начало</a></li>
            <li><a href="./cv.html">CV</a></li>
            <li><a href="./work.html">Работа</a></li>
            <li>Фирми</li>
            <li><a href="./ad.html">Обяви</a></li>
        </ul>

    </header>

        <ul id="firms-logos">
            <li><a href="./firms/epam.html"><img
src="./firms/assets/epam.png"></a></li>
            <li><a href="./firms/happy.html"><img
src="./firms/assets/happy.jpg"></a></li>
            <li><a href="./firms/lufthansa.html"><img
src="./firms/assets/lufthansa.jpg"></a></li>
            <li><a href="./firms/raiffeisenbank.html"><img
src="./firms/assets/raiffeisenbank.jpg"></a></li>
            <li><a href="./firms/sutherland.html"><img
src="./firms/assets/sutherland.png"></a></li>
        </ul>

    <footer id="footer">Copyright &copy; <a
href="mailto:anzh.dimitrov@gmail.com">Anzhelo Dimitrov</a></footer>

</body>
</html>
```


Search..
Начало
CV
Работа
Фирми
Обяви
HP
epam
Happy
Bar & Grill
Lufthansa Technik
Sofia
More mobility for the world

3.Страница на една от фирмите

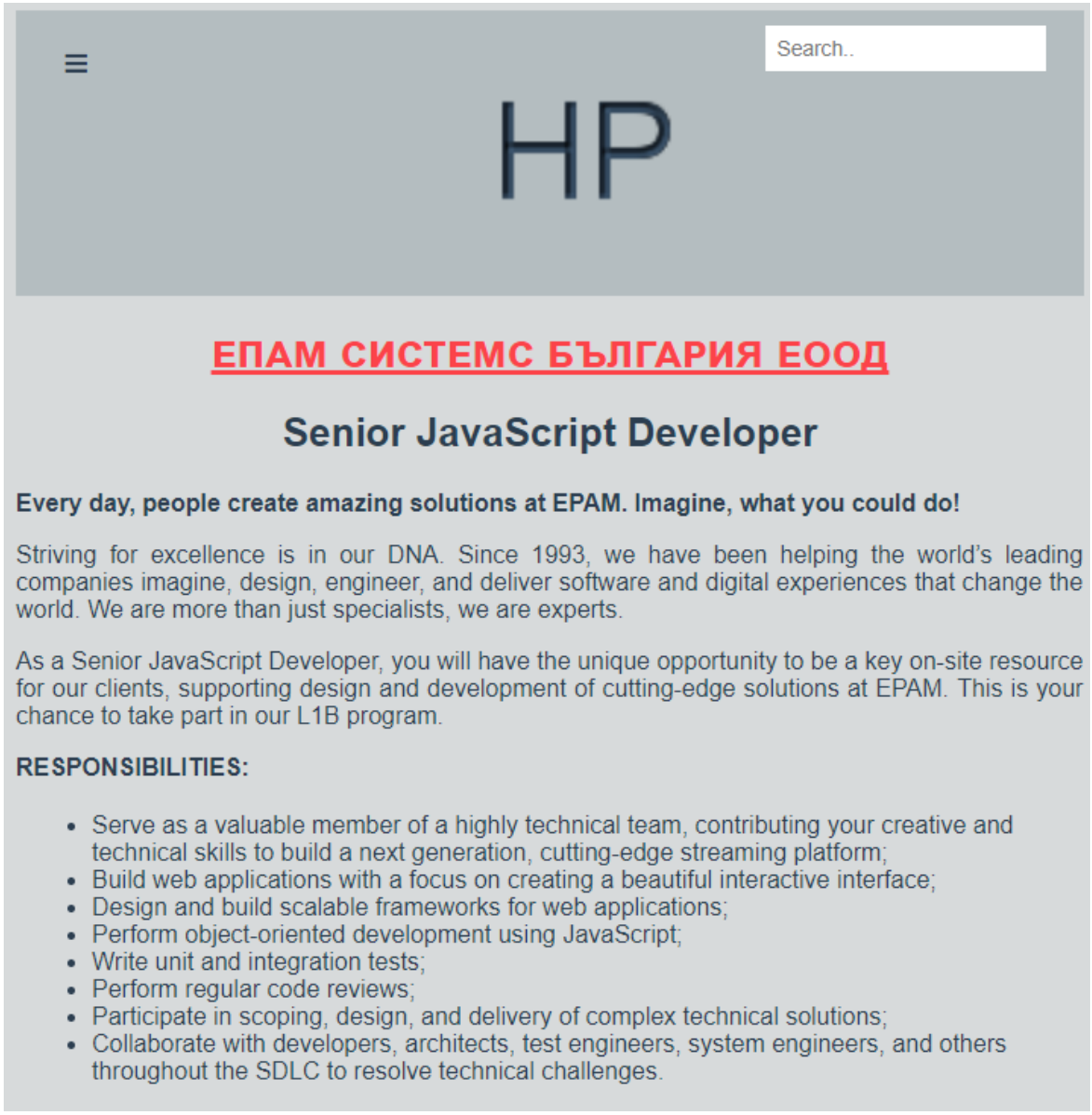
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Намери работа</title>
    <link rel="stylesheet" href="../styles/styles.css">
    <script src="../javascript/javascript.js"></script>
</head>
<body>
    <header class="container">
        <form>
            <input type="text" placeholder="Search..">
        </form>
        <button class="menu-toggle" id="drop-down-btn">
            <i class="fa fa-bars" aria-hidden="true"></i>
        </button>
        <ul id="menu">
            <li><a href="../index.html">Начало</a></li>
            <li><a href="../cv.html">CV</a></li>
            <li><a href="../work.html">Работа</a></li>
            <li><a href="../firms.html">Фирми</a></li>
            <li><a href="../ad.html">Обяви</a></li>
        </ul>
    </header>
    <h1 id="firm-name">Съдърланд Глобъл Сървисиз България ЕООД</h1>
    <p id="logo"><img src="./assets/sutherland.png" class="img" alt="Sutherland
Logo"></p>

    <h2>За компанията</h2>
    <p>
        As a process transformation company, <strong>Sutherland</strong> rethinks
and rebuilds processes for the digital age by combining the speed and insight of
```

```
design thinking with the scale and accuracy of data analytics. We have been
helping customers, across industries from financial services to health care,
achieve greater agility through transformed and automated customer experiences
for over 30 years.
    </p>
    <p>
        Headquartered in Rochester, N.Y., <strong>Sutherland</strong> employs
thousands of professionals spanning 19 countries around the world.
    </p>
    <h3>Контакти</h3>
    <address id="address">
        <b>Адрес:</b> 48, Sitnyakovo Blvd, 4th Floor 1505 Sofia, Bulgaria<br>
        <b>Телефон:</b> +359 2 892 68 69<br>
        <a href="http://www.sutherlandglobal.bg/">Sutherland</a><br>
    </address>
    <footer id="footer">Copyright &copy; <a
href="mailto:anzh.dimitrov@gmail.com">Anzhelo Dimitrov</a></footer>
</body>
</html>
```

4.Обяви

При кликване на някоя от фирмите, потребителя се препраща към страницата на самата фирма. При кликване на някоя от позициите, потребителя се препраща към самата обява, която го интересува.

5.Страница на една от обявите



Заключение

Разработената магистърска теза на тема: “Създаване на сайт за търсене на работа” е осъществена с помоща на HTML, CSS и Javascript.

Първият глава е уводна и представлява запознаване с история и възникване на интернет, основни услуги в интернет и видове уебсайтове.

Във втората глава е включено проучване за възможностите за изграждане на сайта, видовете браузъри, с които е възможно представянето му, както и бази данни, с които е възможно по-нататъшно усъвършенстване, за запазване на данните от потребителите.

В третата глава е показан част от кода, както и са показани екрани на някои от страниците на сайта.

В заключение може да се каже, че целта на магистърската теза е изпълнена. Крайният продукт е функционален уеб сайт, състоящ се от семпъл дизайн, подходящ за сайт за търсене на работа и работещ добре с всички браузъри.

Използвани източници

Печатни издания:

1.Бази от данни и приложения за работа с тях, Автор: Георги Димитров, 2015, Издател: Про Лангс, 2015г.

2.HTML 5 & CSS 3 практическо програмиране за начинаещи, Автор: Денис Николаевич Колисниченко, 2014. и колектив на издателство Асеневци, 2014, Издател: Асеневци трейд ЕООД, 2014г.

3.JavaScript & jQuery практическо програмиране, Автор: Денис Николаевич Колисниченко, 2014 и колектив на издателство Асеневци, 2014, Издател: Асеневци трейд ЕООД, 2014г.

4.Eloquent JavaScript, Written by Marijn Haverbeke, 2015.

5.HTML & CSS Design and Build Websites, Written by Jon Duckett, Published by John Wiley & Sons, Inc., 2011.

6.JavaScript for Web Developers, Third Edition, Written by Nicholas C. Zakas, Published by John Wiley & Sons, Inc., 2012.

Уеб ресурси:

7.Бази данни - https://bg.wikipedia.org/wiki/Бази_данни

8.Браузър - https://bg.wikipedia.org/wiki/Браузър

9.Език за програмиране - https://bg.wikipedia.org/wiki/Език_за_програмиране

10.Интернет - https://bg.wikipedia.org/wiki/Интернет

11.Уебсайт - https://bg.wikipedia.org/wiki/Уебсайт

12.C# - https://bg.wikipedia.org/wiki/C_Sharp

13.CSS - https://en.wikipedia.org/wiki/Cascading_Style_Sheets

14.Firefox - https://en.wikipedia.org/wiki/Firefox

15.Google Chrome - https://en.wikipedia.org/wiki/Google_Chrome

16.HTML - https://bg.wikipedia.org/wiki/HTML

17.Internet Explorer - https://bg.wikipedia.org/wiki/Internet_Explorer

18.Java - https://bg.wikipedia.org/wiki/Java

19.JavaScript - https://bg.wikipedia.org/wiki/JavaScript

20.LAN, MAN и WAN мрежи - http://techs-mobile.blogspot.bg/2011/03/lan-man-wan.html

21.Opera - https://bg.wikipedia.org/wiki/Opera_(браузър)

22.PHP - https://bg.wikipedia.org/wiki/PHP

23.Python - https://en.wikipedia.org/wiki/Python_(programming_language)

24.Ruby - https://bg.wikipedia.org/wiki/Ruby

25.Safari - https://en.wikipedia.org/wiki/Safari_(web_browser)

26.Swift - https://bg.wikipedia.org/wiki/SWIFT_(програмен_език)

27.Visual Studio Code - https://bg.wikipedia.org/wiki/Visual_Studio_Code

Приложения

CSS код за сайта – за изготвянето на сайта не са използвани рамки като Bootstrap.

```
:root{
    background-color: #D7DADB;
}
* p{
    text-align: justify;
}
*{
    font-family: Arial, Helvetica, sans-serif;
    color:#2C3E50;
}
a:link{
    color:#FC4349;
 }
a:active{
    color:#FFFFFF;
}
a:visited{
    color:#FC4349;
}
.menu-toggle {
    border-style: none;
    background-color: transparent;
}
#firms-logos{
    list-style: none;
}
#firms-logos li{
    margin-top: 10px;
}
#apply-on-job{
```

```
    color:#6DBCDB;
    margin-left: 45%;
}
#table-ads{
    margin-top: 2%;
    border-collapse: collapse;
    width: 100%;
    align-self: center;
}
#table-ads td, th{
    border: 1px solid #2C3E50;
    text-align: left;
    padding: 8px;
}
#table-ads tr:nth-child(even){
    background-color: #6DBCDB;
}
#table-ads tr td a{
    color:#2C3E50;
    text-decoration: none;
}
table{
    color:#FC4349 !important;
}
input:focus{
    background-color: #FC4349;
}
li a {
    text-decoration: none;
}
#firm-choose{
    list-style-type: none;
}
#choose{
    font-weight: 700;
```

```
}
.container{
    overflow: hidden;
    background-color: #6DBCDB;
}
.container{
    background-image: url(../assets/banner.png);
    background-repeat: no-repeat;
    background-size: cover;
    height: 130px;
}
.container a{
    text-decoration: none;
    font-size: 17px;
}
.container input[type=text] {
    float: right;
    padding: 6px;
    border: none;
    margin-top: 8px;
    margin-right: 30px;
    font-size: 17px;
}
.container a:hover {
    background-color: #FFFFFF;
}
.show{
    display: block !important;
    width: 30%;
    list-style: none;
    padding: 10px 5px;
    background-color: transparent;
    margin-top: 6px;
    margin-left: 19px;
}
```

```
#footer{
    position: absolute;
    font-size: 10px;
    margin-top: 1%;
    padding-left: 45%;
    color:#2C3E50;
}
#address{
    color:#6DBCDB;
    font-size: 13px;
}
#logo{
    padding-left: 40%;
}
#firm-name{
    text-align: center !important;
    font-variant-caps:all-petite-caps;
}
#position{
    text-align: center !important;
}
h1{
    font-size: 35px;
}
h2{
    font-size: 30px;
}
h3{
    font-size: 25px;
}
h4{
    font-size: 20px;
}
h5{
```

```
    font-size: 15px;
}
h6{
    font-size: 10px;
}
/* ЛОГИН */
#login-form, #register-form {
    border: 3px solid #6DBCDB;
}
.login-container{
    margin-left: 15px;
}
#login-btn, #register-btn {
    background-color: #6DBCDB;
    color: #FC4349;
    padding: 14px 20px;
    margin: 8px 8px;
    border: none;
    cursor: pointer;
    width: 100%;
}
#login-btn:hover, #register-btn:hover {
    opacity: 0.8;
}
.imgcontainer {
    text-align: center;
    margin: 24px 0 12px 0;
}
#avatar {
    width: 25%;
    border-radius: 50%;
}
.login-container, .register-container {
    padding: 16px;
}
```

```
.modal-container{
    margin-top: 10px;
    display: inline-block;
}
.modal{
    z-index:3;
    display:none;
    padding-top:5px;
    position:fixed;
    left:0;
    top:0;
    width:100%;
    height:100%;
    overflow:auto;
    background-color:transparent;
}
.modal-content{
    margin-top: 5px;
    margin:auto;
    background-color:#D7DADB;
    position:relative;
    padding:0;
    outline:0;
    width:600px;
}
.display-topright{
    position:absolute;
    right:0;
    top:0
}
/* ЛОГИН КРАЙ */
/* Media */
@media print{
    .img{
        display: none;
```

```
    }
}
@media (max-width: 600px) {
    .container input[type=text]{
        font-size: 11px;
        margin-right: 20px;
    }
    .modal-content{
        margin:0 10px;
        width:auto!important;
    }
    .modal{
        padding-top:200px;
    }
    #menu{
        display: none;
    }
    .container{
        display: block;
        background-image: url(../assets/banner2.png);
    }
    h1 {
        font-size: 1.5em;
    }
    h2 {
        font-size: 1.3em;
    }
    h3 {
        font-size: 1.1em;
    }
    h4,h5,h6 {
        font-size: 0.9em;
    }
    * p {
        font-size: 0.8em;
```

```
    }
    ul li {
        font-size: 0.8em;
    }
    .menu-toggle{
        margin: 20px 0 0 20px;
    }
    .img{
        display: none;
    }
}
@media (min-width: 601px) {
    #menu{
        list-style: none;
        width: 60%;
        float: left;
        padding: 0 0 0 10px;
        margin-top: 0;
    }
    #menu li {
        display: inline-flex;
        margin-right: 10px;
        padding: 14px 16px;
    }
    .menu-toggle {
      display: none;
    }
}
@media(min-width: 600px) and (max-width: 811px){
    #menu li {
        margin-right: 2px;
    }
}
@media(min-width: 600px) and (max-width: 704px){
    .container input[type=text]{
```

```
        font-size: 11px;
        margin-right: 20px;
    }
    #menu{
        width: 67%;
    }
    .container a{
        font-size: 10px;
    }
}
@media (min-width: 704px) and (max-width: 813px){
    .container a{
        font-size: 10px;
    }
}
@media (min-width: 814px) and (max-width: 900px){
    .container a{
        font-size: 13px;
    }
}
@media (max-width:768px){
    .modal-content{
        width:500px;
    }
    .container{
        height: 150px !important;
    }
}
@media (min-width: 1440px){
    h1 {
        font-size: 2.2em;
    }
    h2 {
        font-size: 2em;
    }
```

```
    h3 {
        font-size: 1.8em;
    }
    h4,h5,h6 {
        font-size: 1.6em;
    }
    * p {
        font-size: 1.4em;
    }
    ul li {
        font-size: 1.4em;
    }
    .menu-toggle{
        margin-top: 10px;
        margin-bottom: 10px;
    }
}
/* Banner */
@media(max-width: 400px){
    .container{
        height: 210px;
        background-image: url(../assets/banner3.png);
    }
}
@media(max-width: 600px){
    .container{
        height: 100px;
    }
}
@media(max-width: 820px){
    .container{
        height: 100px;
    }
}
@media(min-width: 1100px){
```

```
    .container{
        height: 210px;
    }
}
@media(min-width: 1630px){
    .container{
        height: 300px;
    }
}
/* End Banner */
```