**2 ПРОЕКТУВАННЯ СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМІВ ФУНКЦІОНУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ**

**2.1 Вибір програмних засобів для реалізації проекту**

Перед початком розробки даного проекту потрібно вибрати ті методи і засоби, які дозволять реалізувати задумане.

## Для реалізації даної програми було використано програмове середовище WebStorm 6 та операційна система Ubuntu 12.04.

Основними засобами для реалізації веб-проекту на стороні клієнта є HTML (мова розмітки гіпертексту), CSS (каскадна таблиця стилів), JavaScript (прототипно-орієнтована скриптова мова програмування). Та для зручної взаємодії JavaScript і HTML найчастіше використовується jQuery (бібліотека JavaScript), яка дозволяє легко створювати візуальні ефекти або опрацьовувати дані.

## 2.1.1 Опис мови розмітки HTML.

HTML (англ. *HyperText Markup Language* — Мова розмітки гіпертексту) — стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді.

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту.

HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.[1]

HTML впроваджує засоби для:[1]

* створення структурованого документу шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
* отримання інформації із Всесвітньої мережі через гіперпосилання;
* створення інтерактивних форм;
* включення зображень, звуку, відео, та інших об'єктів до тексту.

Розмітка в HTML складається з чотирьох основних компонентів: елементів (та їхніх атрибутів), базових типів даних, символьних мнемонік та декларації типу документа.

#### Загальна структура

Документ HTML 4.01 складається з трьох частин:

1. Декларація типу документу (англ. *Document type declaration*, Doctype), на початку документа, в якій визначається тип документа (DTD).
2. Шапка документу (знаходиться в межах елементу head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;
3. Тіло документу (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

Нижче наведено приклад загальної структури HTML-документу:

<!doctype HTML public "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

<**html**>

<**head**>

<**title**>Мій перший HTML-документ</**title**>

</**head**>

<**body**>

Hello world!

</**body**>

</**html**>

## 2.1.2 Каскадні таблиці стилів – CSS.

Каскадні таблиці стилів (англ. Cascading Style Sheets або скорочено CSS) — спеціальна мова, що використовується для відображення сторінок, написаних мовами розмітки даних. Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже існуючи функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їх візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок для того, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Таке розділення може покращити сприйняття та доступність контенту, забезпечити більшу гнучкість та контроль за відображенням контенту в різних умовах, зробити контент більш структурованим та простим, прибрати повтори та ін. CSS також дозволяє адаптувати контент до різних умов відображення (на екрані монітора, мобільного пристрою (КПК), у роздрукованому вигляді, на екрані телевізора, пристроях з підтримкою шрифту Брайля або голосових броузерах та ін.)

Один і той самий HTML або XML документ може бути відображенний по-різному в залежності від використаного CSS. Стилі для відображення сторінки можуть бути:

* Стилі автора (інформація надана автором сторінки):

1. зовнішні таблиці стилів (англ. stylesheet), частіше за все окремий файл або файли .css
2. внутрішні таблиці стилів, включені як частина документу або блоку
3. стилі для окремого елементу

* Стилі користувача

1. локальний .css-файл, вказаний користувачем для використання на сторінках і вказаний в налаштуваннях браузера (наприклад Opera)

* Стилі переглядача (браузера)

1. стандартний стиль переглядача, наприклад стандартні стилі для елементів, визначені браузером, використовуються коли немає інформації про стиль елемента або вона неповна.

Стандарт CSS визначає порядок та діапазон застосування стилів, те, в якій послідовності і для яких елементів застосовуються стилі. Таким чином, використовується принцип каскадності, коли для елементів вказується лише та інформація про стилі, що змінилася або не визначена загальнішими стилями.

Переваги

* Інформація про стиль для цілого сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок;
* Різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв;
* Сторінки зменшуються в об'ємі та стають більш структурованими, за рахунок того що інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з їх урахуванням;
* Прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

## 

## 2.1.3 Опис мови програмування JavaScript.

JavaScript — назва реалізації стандарту мови програмування ECMAScript компанії Netscape, базується на принципах прототипного програмування. Найпоширеніше і найвідоміше застосування мови — написання сценаріїв для веб-сторінок, але, також, використовується для впровадження сценаріїв керування об’єктами вбудованими в інші програми.

Незважаючи на схожість назв, JavaScript та мова програмування Java мають дуже мало спільного. Права на назву Java належать компанії Sun Microsystems.

JavaScript розроблений у компанії Netscape. На сьогоднішній день підтримується більшістю браузерів. Текст програми включається безпосередньо в HTML-документ і інтерпретується самим браузером (точніше, вбудованим у браузер рушієм JavaScript). Застосовується в основному для часткової автоматизації обробки і маніпуляції даними, які використовує сторінка.

JavaScript має низку властивостей об’єктно-орієнтованої мови, але завдяки концепції прототипів підтримка об’єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, властивих функціональним мовам, — функції як об’єкти першого рівня, об’єкти як списки, каррінг (currying), анонімні функції, замикання(closures) — що додає мові додаткову гнучкість.

JavaScript має C-подібний синтаксис, але в порівнянні з мовою Сі має такі корінні відмінності:

* об’єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів
* функції як об’єкти першого класу
* обробка винятків
* автоматичне приведення типів
* автоматичне прибирання сміття
* анонімні функції

JavaScript містить декілька вбудованих об’єктів: Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp. Крім того, JavaScript містить набір вбудованих операцій, які, строго кажучи, не обов’язково є функціями або методами, а також набір вбудованих операторів, що управляють логікою виконання програм. Синтаксис JavaScript в основному відповідає синтаксису мови Java (тобто, зрештою, успадкований від C), але спрощений порівняно з ним, щоб зробити мову сценаріїв легкою для вивчення. Так, приміром, декларація змінної не містить її типу, властивості також не мають типів, а декларація функції може стояти в тексті програми після неї.

Приклад оголошення і використання класу в JavaScript (клас є одночасно функцією, оскільки функції — це об’єкти першого рівня):

**function** MyClass()

{

**this**.myValue1 = 1;

**this**.myValue2 = 2;

}

**var** mc = **new** MyClass();

mc.myValue1 = mc.myValue2 \* 2;

Одна з популярних технологій, що дозволила зробити сторінки динамічнішими і забезпечити нові можливості — це динамічне завантаження і вставка даних в документ, що отримала назву AJAX.

## 

## 2.1.4 Опис JavaScript-бібліотеки Jquery.

jQuery — популярна JavaScript-бібліотека з відкритим вихідним кодом. Вона була представлена у січні 2006 року у BarCamp NYC Джоном Ресіґом (John Resig). Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

jQuery є вільним відкритим програмним забезпеченням під ліцензією MIT (до вересня 2012 було подвійне ліцензування під MIT таGNU General Public License другої версії).

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

Основне завдання jQuery — це надавати розробнику легкий та гнучкий інструментарій кросбраузерної адресації DOM об'єктів за допомогою CSS та XPath селекторів. Також даний фреймворк надає інтерфейси для Ajax-застосунків, обробників подій і простої анімації.

Принцип роботи jQuery полягає в використанні класу (функції), який при звертанні до нього повертає сам себе. Таким чином, це дозволяє будувати послідовний ланцюг методів

$('#test') *//знаходимо елемент з id="test"*

.text('Клікни по мені') *//встановлюємо текст елемента рівним "Клікни по мені"*

.addClass('myAlert') *//додаємо клас "myAlert"*

.css('color','red') *//встановлюємо колір тексту червоним*

.attr('alert','Привіт, світе!') *// додаємо атрибут "alert" із значенням "Привіт, світе!"*

.bind( *// додаємо в обробник події click функцію, яка відкриє модальне*

'click', *// вікно із текстом, що вказаний в атрибуті "alert" ("Привіт, світе!")*

**function**(){alert($(**this**).attr('alert'))}

);

Бібліотека jQuery є JavaScript файлом, яка включає всю його DOM, події(events), ефекти(effects), і Ajax функції. Вона може бути додана до web-сторінки посиланням на локальну копію, або на одну з копій доступних на публічному сервері (наприклад Google або Microsoft CDN).

<script type="text/javascript" src="jquery.js"> </script>

*// Виберемо всі парні елементи "tr", і застосуємо для них css клас "odd"*

$("tr:nth-child(odd)").addClass("odd");

*// Відправлення асинхронного POST запиту на адресу '/ajaxtest.php'*

$.post(

'/ajaxtest.php',

{

type: "test-request",

param1: "param1",

param2: 2

},

onAjaxSuccess

);

**function** onAjaxSuccess(data){ *// Тут ми отримуєм відповідь, і опрацьовуєм результат*

alert(data);

}

**2.2 Програмна структура системи**

**2.2.1 Принципи роботи системи**

При кліку курсором миші по формі для вводу тексту, відкривається вікно програми, яке дає можливість малювати символи, та керувати процесом розпізнавання.

Розпізнавання символів реалізується з використання шаблонного алгоритму. Перед початком розпізнавання, намальоване зображення потрібно обробити. Спершу відбувається процес обрізання зображення від непотрібної рамки, далі одержане зображення масштабується і перетворюється у бінарну матрицю. Після вище обробки зображення відбувається сам процес розпізнавання символу шаблонним методом.

Після того як символ було розпізнано, можна продовжити процес малювання і розпізнавання символів. Коли усі потрібні символи були розпізнані, за допомогою панелі керування, увесь отриманий текст у ході розпізнавання можна занести у початкову форму, для якого і було викликано вікно програми.

**2.2.2 Розробка структури об’єкта розпізнавання**

Для забезпечення роботи даної системи, було розроблено клас який забезпечує увесь необхідний функціонал. Структура класу зображена на рисунку 2.1.

Клас містить властивості, для зберігання, як всіх основних параметрів, так і допоміжних.

Методи класу забезпечують сам процес розпізнавання символів, допоміжні операції потрібні для розпізнавання, та опрацювання подій миші.

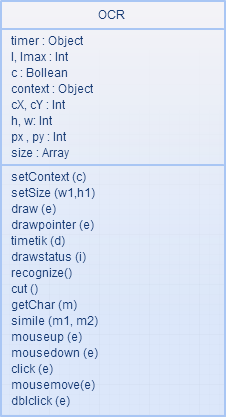


Рисунок 2.1 – Структура об’єкта розпізнавання символу

**2.2.3 Створення структури об’єкта для зберігання матриць**

Для роботи даної системи необхідно розробити структуру об’єктів, у яких буде зберігатися інформація про еталонні символи, з якими буде легко працювати. Була розроблена структура об’єкта яка містить:

* Сам символ
* Бінарну матрицю символа
* Ширину матриці
* Висоту матриці

Вигляд даного об’єкта зображено на рисунку 2.2



Рисунок 2.2 – Розроблена структура еталонного об’єкта