Муниципальное бюджетное общеобразовательное учреждение
средняя общеобразовательная школа № 1 р.п.Лунино
имени Артамонова Н.С.

Районная научно-практическая конференция школьников
«Старт в науку»

Секция «Информатика»

Ускорение загрузки web-страницы

Подготовил:
Пурыга Никита Андреевич,
учащийся 10 класса
МБОУ СОШ № 1 р.п.Лунино
имени Артамонова Н.С.
Адрес: 442730, Пензенская область,
р.п.Лунино, ул.Мясникова, 42
Телефон: 8-841-61-3-13-09
e-mail: luninoschool1@yandex.ru

Руководитель:
Митрофанова Светлана Юрьевна
Учитель информатики
МБОУ СОШ № 1 р.п.Лунино
имени Артамонова Н.С.
Адрес: 442730, Пензенская область,
р.п.Лунино, ул.Мясникова, 42
Телефон: 8-841-61-3-13-09
e-mail: luninoschool1@yandex.ru

Оглавление

стр.

1. **Введение.**

Интернет плотно вошел в жизнь почти каждого человека или организации. Глобальная сеть является популярным источником информации, инструментом быстрой связи и надежного хранения данных с доступом из любой точки планеты.

В настоящее время доступ к интернету есть у большинства населения планеты, а главной единицей мировой сети является сайт.

Мировое сообщество веб-разработчиков постоянно совершенствует технологии веб-разработки, добавляя в языки программирования все больше новых возможностей. Наравне с совершенствованием языков программирования стоит задача оптимизации production файлов сайта до максимальной рациональности, для повышения скорости загрузки веб-страниц в браузере. Именно задача оптимизации production файлов стала предметом исследования данного проекта.

Объект исследования: веб-сайт

Предмет исследования: внутренние файлы современной веб-страницы

Цель: создать инструмент, позволяющий сократить время загрузки веб-страницы

Гипотеза: оптимизация production файлов уменьшит время загрузки веб-страницы

Актуальность: инструмент может быть применен разработчиками чтобы ускорить загрузку веб-страницы не замедляя процесс разработки

Задачи:

1. Изучить процесс загрузки веб-страниц в браузере пользователя
2. Проанализировать, какие языки программирования являются самыми популярными для создания современных веб-страниц
3. Определить популярный развивающийся язык программирования, файлы которого занимают большее время загрузки веб-страницы с сервера в браузер клиента
4. Создать инструмент веб-разработки оптимизирующий скорость загрузки веб-страницы

2. Теоретическая часть.

2.1. Процесс загрузки веб-страницы в браузер пользователя.

Процесс загрузки можно разделить на несколько последовательных этапов:

1. **Запрос к хостингу.** На этом этапе происходит обработка DNS запросов. Браузер передает URL страницы, которую в дальнейшем необходимо отобразить у пользователя.
2. **Редиректы.** Принудительное перенаправление с одного URL на другой. Редирект является необязательным этапом загрузки страницы
3. **Подключение к HTTP серверу.** Браузер пользователя подключается к серверу и сообщает ему IP компьютера, которому необходимо вернуть ответ сервера.
4. **Ответ сервера.** Браузер пользователя получает ответ в виде пакета данных, содержащего файл гипертекстовой разметки и дополнительные инструкции.
5. **Обработка HTML файла.** Браузер обрабатывает код html файла, определяет дополнительные файлы, необходимые для окончательной отрисовки веб-страницы.
6. **Повторение пунктов 1-4.** В случае, если дополнительные файлы необходимы, браузер повторяет пункты 1-4 для каждого файла.
7. **Обработка дополнительных файлов.**
8. **Отрисовка веб-страницы.** Браузер отрисовывает веб-страницу на основе тех правил, которые он получил от сервера

2.2. Популярные языки программирования, используемые для создания веб-страниц

Разработку программной части сайта можно разделить на 2 основных этапа:

1. **Frontend-разработка.** Разработка клиентской части веб-страницы. Осуществляет взаимодействие с пользователем на уровне браузера. Именно frontend файлы отправляются с сервера в браузер.
2. **Backend-разработка.** Разработка серверной части. На данном этапе создаются правила для сервера, на основе которых им будут сгенерированы HTML файлы для отправки их браузеру пользователя.

Для каждого этапа предназначены собственные языки программирования. По данным исследования компании TIOBE Software на февраль 2020 года, популярными языками frontend разработки являются:

- **HTML** - язык гипертекстовой разметки, отвечающий за наличие элементов страницы, таких как кнопка, ссылка, список, изображение и т.п.
- **CSS** - язык таблиц стилей. С его помощью задаются такие параметры, как размер элементов, их цвет, положение на странице и многие другие.
- **JavaScript** - полноценный язык программирования, имеющий, в отличие от предыдущих двух, функционал, позволяющий создавать набор сложных правил поведения элементов или страницы, в том числе при воздействия на них пользователем.

На втором этапе, по данным того же исследования, разработчик чаще всего использует следующие языки программирования:

1. **PHP** - полноценный язык программирования, позволяющий создавать html файлы в зависимости от различных факторов, таких как дата или время суток, IP пользователя, геолокация и другие.
2. **SQL** - язык структурированных запросов. С помощью него, разработчик осуществляет запросы к базе данных.

Вывод: среди мировых языков, JavaScript, PHP и SQL заняли 7, 8 и 9 места соответственно. HTML и CSS не вошли рейтинг, потому что не являются полноценными языками. Но их популярность неопровержима. Единственным конкурентом HTML является язык XHTML, группа разработки которого XHTML 2, прекратила свою работу 2 июля 2009 года. А CSS является единственным языком таблиц стилей.

2.3. **Язык веб-разработки, файлы которого имеют самый большой вес.**

Информационный вес файла прямо-пропорционален количеству содержащейся в нем информации.

Для того, чтобы определить тип файлов, на загрузку которого уходит больше всего времени, проведем анализ файлов популярных веб-страниц на количество информации с помощью сервиса https://text.ru/seo. Стоит отметить, что анализирование файлов предназначенных для backend не имеет смысла, так как они не загружаются в браузер, а осуществляют свою работу на сервере, не покидая его границы.

Результаты исследования представлены формате: язык программирования - количество символов.

Память народа (pamyat-naroda.ru).

- HTML - 73 128 символов
- CSS - 365 636 символов
- JavaScript - 186 580 символов

Wikipedia (ru.wikipedia.org)

- HTML - 111 223 символов
- CSS - 153 633
- JavaScript - 122 017 символов

Microsoft (www.microsoft.com)

- HTML - 256 866 символов
- CSS - 468 748 символов
- JavaScript - 139 934 символов

Анализ файлов показал, что наибольшее количество информации приходится на CSS файлы.

Вывод: CSS файлы занимают больше всего времени при загрузке веб-страницы.

3. Практическая часть.

3.1. Создание инструмента ускорения загрузки веб-страницы.

```
img {
    width: 120px
    height: 80px
    border: 2px solid red
}
```

Синтаксис CSS представляет собой набор стилей для каждого селектора на странице. На первом изображении цифрой 1 отмечен селектор, а цифрой 2 набор стилей.

```
img, menu {
    width: 120px
    height: 80px
    border: 2px solid red
}
```

В языке также присутствует возможность задать одинаковые стили для различных селекторов, перечислив их через запятую. Так, на втором изображении мы объединили стили для изображений (img) и меню (menu).

Но во время разработки большого проекта, с сотнями селекторов, разработчик не способен объединить стили всех селекторов без ущерба скорости разработки проекта. Однако, если создать инструмент, то объединение стилей будет проходить в автоматическом режиме, не нарушая скорость разработки веб-страницы.

Итог: уже сейчас можно создать техническое задание для будущего инструмента:

1. Получение исходных CSS стилей
2. Синтаксис исходных стилей должен быть максимально приближен к синтаксису CSS
3. Сортировка стилей
4. Генерация исходного CSS кода для вывода на страницу

3.1.1. **Получение исходных CSS стилей**

Для решения задачи объединения селекторов потребуется написать программу на JavaScript. Это единственный язык frontend-разработки, функционал которого способен обрабатывать сложно-логические правила.

JavaScript имеет особый тип данных - объекты, синтаксис которых схож с синтаксисом CSS, а в самом языке предусмотрены возможности сортировки объектов.

На первом изображении (слева на право) - синтаксис объектов JavaScript, на втором - синтаксис CSS.

```
img {
    width: 120px;
    height: 80px;
}
menu {
    width: 120px;
    height: 20px;
}
```

```
'img': {
    width: '120px',
    height: '80px'
},
'menu': {
    width: '120px',
    height: '20px'
},
```

Различия в синтаксисе минимальны, что означает полную пригодность объектов для принятия исходных стилей.

В качестве исходных стилей, оптимизация которых будет проведена в результате, возьмем следующий набор правил и селекторов.

```
'body,html': {
    'margin': '0',
    'padding': '0'
},
'img': {
    width: '120px',
    height: '80px'
},
'menu': {
    width: '120px',
    height: '20px'
},
main: {
    'position': 'relative',
    'background': '#000',
    'color': '#fff',
    'padding': '2%',
    'display': 'flex',
    'margin': 0
},
```

```
header: {
    'position': 'relative',
    'background': 'grey',
    'color': '#fff',
    'padding': '2%'
},
'menu a': {
    'padding': '2%',
    'color': '#fff',
    'margin': '10px 20px',
    'text-decoration': 'none'
},
```

3.1.2. **Сортировка стилей**

На данном этапе, требуется определить, какие CSS стили присутствуют в правилах нескольких селекторов, объединить их и исключить селекторы правила которых не присутствуют в других. Для этого реализуется функция сортировки повторяющихся правил, параметрами которой является селектор и его правило.

```
// sorting selectors by properties
function sorterSelectors (property, selector) {
    if (cssProperty.hasOwnProperty(property)) {
        cssOutputPropertPush(property, selector);
    } else {
        cssProperty[property] = property;
        cssOutput[property] = new Array;
        cssOutputPropertPush(property, selector);
    }
    return true;
}
```

Данная функция вызывается для каждого элемента исходного объекта стилей при его итерации.

```
// sort properties
for (let property in obj[selector]) {
    if (obj[selector].hasOwnProperty(property)) {
        sorterSelectors(property + ': ' + obj[selector][property], selector);
    }
}
```

Также реализуем код склеивания свойств при полном совпадении массивов селекторов

```
//combining css properties of the same selectors
function equal (a, b) {
    if (a === b) return true;
    const aProps = Object.getOwnPropertyNames(a);
    const bProps = Object.getOwnPropertyNames(b);
    if (aProps.length != bProps.length) {
        return false;
    }
    for (let i = 0; i < aProps.length; i++) {
        const propName = aProps[i];
        if (!equal(a[propName], b[propName])) {
            return false;
        }
    }
    return true;
}

function combiningDuplicateProperties (obj) {
    const result = [];
    for (let key in obj) {
        const idx = result.findIndex(([prop, val]) => equal(obj[key], val));
        if (idx == -1) {
            result.push([key, obj[key]]);
        } else {
            const [prop, val] = result[idx];
            result[idx] = [prop + ';' + NEWLINE + key, val];
        }
    }
    return Object.fromEntries(result);
}
```

И вызываем склеивание для output массива

```
cssOutput = combiningDuplicateProperties(cssOutput);
```

в результате мы получим объект отсортированных стилей в формате: правило: [массив селекторов, использующих это правило]

```
'margin: 0': [ 'body,html', 'main' ],
'width: 120px': [ 'img', 'menu' ],
'position: relative': [ 'main', 'header' ],
'color: #fff;\npadding: 2%': [ 'main', 'header', 'menu a' ]
```

Также реализуется функция поиска правил, встречающихся при описании только одного селектора,

```
// selecting selectors that have a single css property
for (let key of Object.keys (cssOutput)) {
    if (cssOutput[key].length == 1) {
        if (cssOutput[key][0] in oneSelector) {
            oneSelectorKeyPush(key);
        } else {
            oneSelector[cssOutput[key]] = new Array;
            oneSelectorKeyPush(key);
        }
        delete cssOutput[key];
    }
}
```

в результате исполнения которой создается новый объект в формате:
селектор: [массив правил, не используемых при описании других селекторов]

```
'*': [ 'box-sizing: content-box;\n' ],
'body,html': [ 'padding: 0;\n' ],
img: [ 'height: 80px;\n' ],
menu: [ 'height: 20px;\n' ],
main: [ 'background: #000;\n', 'display: flex;\n' ],
header: [ 'background: grey;\n' ],
'menu a': [ 'margin: 10px 20px;\n', 'text-decoration: none;\n' ]
```

3.1.3. **Генерация production CSS кода**

В качестве output CSS кода используется многострочная переменная. Производится перебор всех выходных массивов и последующая запись их содержимого, согласно синтаксису CSS.

```
// creating an output styles
for (let el of cssImport) {
    cssOut += el;
}
for (let key of Object.keys(oneSelector)) {
    let result = '';
    if (!(oneSelector[key] === undefined)) {
        cssOut += key;
        for (let el of oneSelector[key]) {
            result += PROPERTYSPACE + el;
        }
    }
    cssOut += SELECTORSPACE + `{` + NEWLINE + result +`}` + NEWLINE;
}
for (let key of Object.keys(cssOutput)) {
    if (!(cssOutput[key] === undefined)) cssOut += cssOutput[key] + `,`;
    cssOut = cssOut.replace(/,\s*$/, '');
    cssOut += SELECTORSPACE + `{` + NEWLINE + PROPERTYSPACE + key + NEWLINE + `}` + NEWLINE;
}
```

Результат генерации стилей

```
header {
    background: grey;
}
menu a {
    margin: 10px 20px;
    text-decoration: none;
}
body,html,main {
    margin: 0
}
img,menu {
    width: 120px
}
main,header {
    position: relative
}
main,header,menu a {
    color: #fff;
    padding: 2%
}
```

```
* {
    box-sizing: content-box;
}
body,html {
    padding: 0;
}
img {
    height: 80px;
}
menu {
    height: 20px;
}
main {
    background: #000;
    display: flex;
}
```

3.2. **Применение программы оптимизации на практике**

Для проверки практической значимости написанного алгоритма, проведем оптимизацию CSS кода сайтов, используемых нами во время поиска языка программирования, файлы которого занимают наибольшее время загрузки. Результаты оптимизации представлены в таблице.

Название сайта	**Кол-во символов до оптимизации**	**Кол-во символов после оптимизации**	**Уменьшение веса файлов в процентах**
Память народа	365 636	208 413	43
Wikipedia	153 633	78 353	49
Microsoft	468 748	295 312	37

4. Заключение

В результате проведенного исследования, удалось создать инструмент, после использования которого, вес файлов, в среднем, уменьшился на 43%. Следовательно, скорость загрузки CSS файлов также уменьшилась на 43%, **что подтверждает гипотезу исследования** и покрывает цель работы.

5. Список литературы

- Рейтинг популярности языков программирования во всем мире https://www.tiobe.com/tiobe-index/
- Рабочая группа XHTML 2 прекращает свою работу https://habr.com/ru/post/63470/
- Справочный материал по языку JavaScript https://learn.javascript.ru/, https://learn.javascript.ru/iterable

6. Приложения

***Сайт**, или **веб-сайт**, - одна или несколько логически связанных между собой веб-страниц*

***DNS** (англ. Domain Name System — система доменных имён) - компьютерная распределённая система для получения информации о доменах.*

***URL** - единый указатель ресурса (англ. Uniform Resource Locator) - уникальный адрес веб-страницы.*

***Пакет данных** - это определенным образом оформленный блок данных, передаваемый по сети с одного компьютера на другой.*

***HTML** (Hyper Text Markup Language) - язык разметки гипертекста - предназначен для написания гипертекстовых документов, публикуемых в интернете. Гипертекстовый документ - это текстовый файл, имеющий специальные метки, называемые тегами, которые впоследствии опознаются браузером и используются им для отображения содержимого файла на экране компьютера.*

***HTML файлы** - файлы языка гипертекстовой разметки HTML, имеющие расширение html.*

***CSS файлы** - файлы языка таблиц стилей CSS, имеющие расширение css.*

***Синтаксис** - это совокупность формальных правил написания программ на данном языке.*

***CSS стили** - правила, указывающие браузеру, как требуется отобразить html элемент на странице.*

***Селектор CSS** - имя HTML элемента, по которому CSS сообщает браузеру, к какому элементу необходимо применить стилевые правила. В роли селектора может выступать класс элемента, его ID, имя тега или любой другой атрибут.*

***Сложно-логические правила** - возможности языка, работать с булевым типом данных, поддерживающие логические операции и вложенности.*