МИНИСТЕРСТВО НАУКИ и высшего образования РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ»

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

филиал «РКТ» МАИ в г. Химки Московской области

**Специальность 09.02.03 «Программирование в компьютерных системах»**

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

**ПМ.02«Разработка и администрирование баз данных**»

**Студент**

**Группы МП-32\_\_\_\_\_\_\_\_\_Дударев Д.А.\_\_ / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**Руководитель**

**практики от организации \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)**

**Руководитель**

**практики от филиала Шумаев А.Ю. / ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )**

**2020г.**

**ПРОГРАММА ПРОИЗВОДСТВЕННОЙ ПРАКТИКИ**

По специальности 09.02.03 «Программирование в компьютерных системах»

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Количество дней практики |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. | 4 |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |
| 06.07.20 | Составление отчёта по практике | 6 |
| 07.07.20 | Подготовка к защите отчёта по практике. | 6 |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт | 6 |

Руководитель практики от филиала «РКТ» МАИ преподаватель

Шумаев А.Ю.

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*(подпись)*

**АТТЕСТАЦИОННЫЙ ЛИСТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Дударев Дмитрий Александрович\_\_\_\_\_\_\_\_\_\_\_

*ФИО*

обучающийся (аяся) на 3-ем курсе по специальности СПО 09.02.03 «Программирование в компьютерных системах»

успешно прошел(ла) производственную практику по профессиональному модулю   
**ПМ.02 «Разработка и администрирование баз данных»**

в объеме 144 часа с 11.06.2020г. по 08.07.2020г.

в организации филиала «РКТ» МАИ

Виды и качество выполнения работ

|  |  |  |
| --- | --- | --- |
| Виды работ, выполненных обучающимся(ейся) во время практики | Объем работ | Качество выполнения работ в соответствии с технологией и (или) требованиями организации, в которой проходила практика |
| Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. | 4 |  |
| Описание структуры сети, в которой может функционировать разработанная база данных. | 4 |  |
| Описание сетевого оборудования необходимого для работы с базой данных. | 4 |  |
| Описание программного обеспечения необходимого для работы с базой данных по сети. | 4 |  |
| Разработка клиентских интерфейсов для клиент-серверных приложений. | 16 |  |
| Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. | 4 |  |
| Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. | 16 |  |
| Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. | 50 |  |
| Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. | 12 |  |
| Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. | 12 |  |
| Составление отчёта по практике | 6 |  |
| Подготовка к защите отчёта по практике. | 6 |  |
| Итоговая аттестация по производственной практике - зачёт | 6 |  |

Руководитель практики от филиала «РКТ» МАИ Шумаев А.Ю.

|  |  |  |
| --- | --- | --- |
| Дата\_\_\_\_\_\_\_\_\_\_\_\_\_ | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *подпись* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*  *Расшифровка подписи* |

**ДНЕВНИК** **ПРОХОЖДЕНИЯ ПРАКТИКИ**

|  |  |  |
| --- | --- | --- |
| Дата | Наименование выполняемых работ | Подпись  руководителя |
| 11.06.20 | Ознакомление с программой практики. Прохождение инструктажа по технике безопасности и охране труда, изучение внутреннего распорядка организации и правил работы. Разработка задания на производственную практику. |  |
| 12.06.20 | Описание структуры сети, в которой может функционировать разработанная база данных. |  |
| 13.06.20 | Описание сетевого оборудования необходимого для работы с базой данных. |  |
| 15.06.20 | Описание программного обеспечения необходимого для работы с базой данных по сети. |  |
| 16.06.20-18.06.20 | Разработка клиентских интерфейсов для клиент-серверных приложений. |  |
| 20.06.20 | Создание инфокоммуникационной системы.  Выполнение сетевых настроек для взаимодействия с СУБД. |  |
| 22.06.20-24.06.20 | Создание концептуальной модели данных.  Создание логической модели данных.  Создание физической модели данных.  Описание механизмов обеспечения целостности базы данных. |  |
| 25.06.20-01.07.20 | Описание СУБД представленной базы данных и ее возможностей. Внесение различных данных в базу данных.  Структурирование запросов базы данных.  Определение методов создания хранимых процедур и триггеров. Создание хранимых процедур и триггеров базы данных. |  |
| 02.07.20-03.07.20 | Определение способов управления правами пользователей.  Описание распределения прав пользователей и управления ими в базе данных. |  |
| 03.07.20-04.07.20 | Определение методов создания и синхронизации реплик базы данных. Описание существующих механизмов репликации в базе данных. Определение методов создания резервных копий базы данных. Описание существующих механизмов резервного копирования в базе данных. |  |
| 06.07.20 | Составление отчёта по практике |  |
| 07.07.20 | Подготовка к защите отчёта по практике. |  |
| 08.07.20 | Итоговая аттестация по производственной практике - зачёт |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 7](#_Toc54110279)

[1 JQUERY 8](#_Toc54110280)

[1.1. Введение в Jquery 8](#_Toc54110281)

[1.2. Основные понятия 10](#_Toc54110282)

[1.3. Интеграция с другими продуктами 11](#_Toc54110283)

[2 JQUERY AJAX 12](#_Toc54110284)

[2.1 Пример работы jQuery.ajax 12](#_Toc54110285)

[2.2 Пример работы jQuery.getJSON 15](#_Toc54110286)

[2.3 Пример работы jQuery.getScript 15](#_Toc54110287)

[Заключение 17](#_Toc54110288)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 18](#_Toc54110289)

[ПРИЛОЖЕНИЯ 19](#_Toc54110290)

# ВВЕДЕНИЕ

Практическая выполнена в соответствии с рабочей программой ПМ.02 «Разработка и администрирование баз данных». В работе представлены основные этапы анализа и разработки предметной области «Разработка и администрирование базы данных», навыки работы с GitHub и Jquery.

База данных играет важную роль в большинстве предметных областей. Благодаря динамической природе автоматизированных отраслей сейчас приложения требуют некоторых механизмов хранения, доступа и изменения данных. Поскольку важность баз данных стремительно растёт, реляционные системы управления базами данных набирают свою популярность.

В качестве программного обеспечения для создания интерфейса выбран редактор Notepad++. А для написания кода выбраны: язык гипертекстовой разметки HTML, язык таблиц стилей CSS и для работы с базами данных PHP. Использован веб-интерфейс phpmyadmin и скрипты Jquery.

В процессе выполнения проекта должны быть получены знания и опыт в области в технологии Jquery. Также мною будут решаться вопросы администрирования базы данных и реализовываться методы и технологии её защиты.

# 1 JQUERY

jQuery — это замечательный JavaScript Framework, который подкупает своей простотой в понимании и удобством в использовании.  Он содержит [набор функций JavaScript](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующийся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript" \o "JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Библиотека jQuery помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими. Также библиотека jQuery предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX). Разработка jQuery ведётся командой добровольцев на пожертвования.

DOM (Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

API - Интерфейс прикладного программирования (иногда интерфейс программирования приложений) (Application Programming Interface, API) — набор готовых классов, функций, структур и констант, предоставляемых приложением (библиотекой) для использования во внешних программных продуктах   
Подключение библиотеки jQuery происходит так же, как и подключение любой другой.

# Введение в Jquery

JavaScript - фреймворки. Библиотеки этого типа призваны упростить использование языка JavaScript и сделать его более доступным как для новичков, так и для разработчиков со стажем за счет предоставления простых в применении функций, облегчающих решение повседневных задач.

Существует довольно много библиотек JavaScript. Самыми популярными из тех, которые используются разработчиками в настоящее время, являются библиотеки Prototype, MooTools, а также jQuery, которая в настоящее время является самой популярной.

Основная часть jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

Основная цель создания jQuery - возможность закодировать многоразовые куски кода, которые позволят упростить JavaScript и использовать их так, чтобы не беспокоиться о кросс-браузерных вопросах.

Преимущества:

1.получение доступа к любому элементу

2.обращение к атрибутам и содержимому элементов DOM и их манипуляция

3.синтаксис селекотора элементов схож с CSS

4.предоставление удобного API для работы с AJAX

5.возможность анимации полученных элементов на странице

6.небольшой размер файла

7.чрезвычайно простой синтаксис

8.возможность объединения последовательно вызываемых методов в цепочки

9.простая архитектура подключаемых модулей, расширяющих базовые возможности библиотеки

10. огромное сетевое сообщество пользователей

11. полезные расширения, такие как jQuery UI, предоставляющие дополнительную функциональность.

Работу с jQuery можно разделить на 2 типа:

Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих подкритерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции текучего интерфейса.

Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.  
Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора CSS, что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery.

Например:  
$("div.test").add("p.quote").addClass("blue").slideDown("slow");  
находит все элементы div с классом test, а также все элементы p с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Здесь методы add, addClass и slideDown возвращают ссылку на исходный объект $("div.test"), поэтому

возможно вести такую цепочку.

Методы, начинающиеся с $., удобно применять для обработки глобальных объектов.

Например:

$.each([1,2,3], function() {

document.write (this + 1);

});

добавит на страницу 234.

Подключение библиотеки jQuery происходит так же, как и подключение любой другой JavaScript библиотеки:

<head>

<script type="text/javascript” src="путь\_к/jquery.js"></script>

</head>

Вызывается работа библиотеки символом $:

$(function() {

// Тут код скрипта

})

В редких случаях вызывается командой jQuery, это бывает в тех случаях, когда браузер не может обработать или понять символ $:

jQuery(function($) {

// Тут код скрипта, где в $ будет jQuery

})

Пример работы jQuery с Ajax:

$.ajax({

dataType : "json",

url: "some.php",

data: {name: 'John'},

complete:cbComplete,  
error: cbError,

success: cbSuccess

});

error — если произошла ошибка

success — если ошибок невозникло

complete — срабатывает по окончанию запроса

* 1. Основные понятия

DOM (Document Object Model — «объектная модель документа») — это не зависящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML, XHTML и XML-документов, а также изменять содержимое, структуру и оформление таких документов.  
API - Интерфейс прикладного программирования (иногда интерфейс программирования приложений) (Application Programming Interface, API) — набор готовых классов, функций, структур и констант, предоставляемых приложением (библиотекой) для использования во внешних программных продуктах

# Интеграция с другими продуктами

28 сентября 2008 года на официальном блоге jQuery сообщили о том, что компании Microsoft и Nokia собираются сотрудничать с группой разработчиков. Компания Microsoft собирается интегрировать в свой продукт ASP.NET листинги кода и примеры jQuery, а компания Nokia собирается интегрировать jQuery для своих мобильных виджетов.

# 2 JQUERY AJAX

jQuery функция $.ajax() позволяет выполнить асинхронный AJAX запрос. AJAX (от англ. Asynchronous Javascript and XML — "асинхронный JavaScript и XML") — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в "фоновом" обмене данными браузера с веб-сервером.

Начнем с самого простого — загрузка HTML кода в необходимый нам DOM элемент на странице. Для этой цели нам подойдет метод load. Данный метод может принимать следующие параметры:

url запрашиваемой страницы

передаваемые данные (необязательный параметр)

функция которой будет скормлен результат (необязательный параметр)

Приведу пример JavaScript кода:

// по окончанию загрузки страницы

$(document).ready(function(){

// вешаем на клик по элементу с id = example-1

$('#example-1').click(function(){

// загрузку HTML кода из файла example.html

$(this).load('ajax/example.html');

})

});

Пример подгружаемых данных (содержимое файла example.html):

Example<br/>

Data Loaded By AJAX<br/>

Bye-Bye

# 2.1 Пример работы jQuery.ajax

Это самый основной метод, а все последующие методы лишь обертки для метода jQuery.ajax. У данного метода лишь один входной параметр — объект включающий в себя все настройки (выделены параметры которые стоит запомнить):

async — асинхронность запроса, по умолчанию true

cache — вкл/выкл кэширование данных браузером, по умолчанию true

contentType — по умолчанию «application/x-www-form-urlencoded»

data — передаваемые данные — строка иль объект

dataFilter — фильтр для входных данных

dataType — тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)

global — тригер — отвечает за использование глобальных AJAX Event'ов, по умолчанию true

ifModified — тригер — проверяет были ли изменения в ответе сервера, дабы не слать еще запрос, по умолчанию false

jsonp — переустановить имя callback функции для работы с JSONP (по умолчанию генерируется на лету)

processData — по умолчанию отправляемые данный заворачиваются в объект, и отправляются как «application/x-www-form-urlencoded», если надо иначе — отключаем

scriptCharset — кодировочка — актуально для JSONP и подгрузки JavaScript'ов

timeout — время таймаут в миллисекундах

type — GET либо POST

url — url запрашиваемой страницы

Локальные AJAX Event'ы:

beforeSend — срабатывает перед отправкой запроса

error — если произошла ошибка

success — если ошибок не возникло

complete — срабатывает по окончанию запроса

Для организации HTTP авторизации (О\_о):

username — логин

password — пароль

Пример javaScript'а:

$.ajax({

url: '/ajax/example.html', // указываем URL и

dataType : "json", // тип загружаемых данных

success: function (data, textStatus) { // вешаем свой обработчик на функцию success

$.each(data, function(i, val) { // обрабатываем полученные данные

/\* ... \*/

});

}

});

jQuery.get-Загружает страницу, используя для передачи данных GET запрос. Может принимать следующие параметры:

url запрашиваемой страницы

передаваемые данные (необязательный параметр)

callback функция, которой будет скормлен результат (необязательный параметр)

тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)

jQuery.post-Данный метод аналогичен предыдущему, лишь передаваемые данные уйдут на сервер посредством POST'а. Может принимать следующие параметры:

url запрашиваемой страницы

передаваемые данные (необязательный параметр)

callback функция, которой будет скормлен результат (необязательный параметр)

тип данных возвращаемых в callback функцию (xml, html, script, json, text, \_default)

JavaScript:

$(document).ready(function(){ // по завершению загрузки страницы

$('#example-3').click(function(){ // вешаем на клик по элементу с id = example-3

$.post('ajax/example.xml', {}, function(xml){ // загрузку XML из файла example.xml

$('#example-3').html('');

$(xml).find('note').each(function(){ // заполняем DOM элемент данными из XML

$('#example-3').append('To: ' + $(this).find('to').text() + '<br/>')

.append('From: ' + $(this).find('from').text() + '<br/>')

.append('<b>' + $(this).find('heading').text() + '</b><br/>')

.append( $(this).find('body').text() + '<br/>');

});

}, 'xml'); // указываем явно тип данных

})

});

Файл example.xml:

<?xml version="1.0" encoding="UTF-8"?>

<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Don't forget me this weekend!</body>

</note>

# 2.2 Пример работы jQuery.getJSON

Загружает данные в формате JSON (удобней и быстрее нежели XML). Может принимать следующие параметры:

url запрашиваемой страницы

передаваемые данные (необязательный параметр)

callback функция, которой будет скормлен результат (необязательный параметр)

JavaScript:

$(document).ready(function(){ // по завершению загрузки страницы

$('#example-4').click(function(){ // вешаем на клик по элементу с id = example-4

$.getJSON('ajax/example.json', {}, function(json){ // загрузку JSON данных из файла example.json

$('#example-4').html('');

// заполняем DOM элемент данными из JSON объекта

$('#example-4').append('To: ' + json.note.to + '<br/>')

.append('From: ' + json.note.from + '<br/>')

.append('<b>' + json.note.heading + '</b><br/>')

.append( json.note.body + '<br/>');

});

})

});

Файл example.json:

{

note:{

to:'Tove',

from:'Jani',

heading:'Reminder',

body:'Don\'t forget me this weekend!'

}

}

# 2.3 Пример работы jQuery.getScript

данная функция загружает и выполняет локальный JavaScript. Может принимать следующие параметры:

url запрашиваемого скрипта

callback функция, которой будет скормлен результат (необязательный параметр)

JavaScript:

$(document).ready(function(){ // по завершению загрузки страницы

$('#example-5').click(function(){ // вешаем на клик по элементу с id = example-5

$.getScript('ajax/example.js', function(){ // загрузку JavaScript'а из файла example.js

testAjax(); // выполняем загруженный JavaScript

});

})

});

Файл example.js:

function testAjax() {

$('#example-5').html('Test completed'); // изменяем элемент с id = example-5

}

# Заключение

Библиотека jQuery, детище Джона Резига, была впервые представлена общественности на компьютерной конференции BarCamp в Нью-Йорке в начале 2006 года. Как отметил на своем сайте сам Резиг, к созданию jQuery его подтолкнули неудовлетворенность существовавшими на то время библиотеками и осознание того факта, что их можно значительно улучшить, уменьшив объем "синтаксической шелухи" и введя специфические элементы управления для часто выполняемых операций.  
Библиотека jQuery сразу же завоевала признание сообщества разработчиков и быстро распространилась среди его членов. В процесс усовершенствования библиотеки включились другие разработчики, итогом чего стал выпуск первой устойчивой версии 1.0, состоявшийся 26 августа 2006 года.

С тех пор библиотека jQuery непрерывно улучшается (на данный момент уже выпущена версия jQuery 2.0), и в настоящее время она снабжена множеством дополнений (подключаемых модулей), предложенных сообществом разработчиков.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Github: [Электронный ресурс]/https://tproger.ru/translations/difference-between-git-and-github - статья в интернете.
2. Jquery: [Электронный ресурс]/ https://jquery.com/ - статья в интернете.
3. Руководство AJAX: [Электронный ресурс]/https://basicweb.ru/jquery/jquery\_ajax.php - статья в интернете.

# ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А

( function( global, factory ) {

"use strict";

if ( typeof module === "object" && typeof module.exports === "object" ) {

module.exports = global.document ?

factory( global, true ) :

function( w ) {

if ( !w.document ) {

throw new Error( "jQuery requires a window with a document" );

}

return factory( w );

};

} else {

factory( global );

}

} )( typeof window !== "undefined" ? window : this, function( window, noGlobal ) {

var arr = [];

var getProto = Object.getPrototypeOf;

var slice = arr.slice;

var flat = arr.flat ? function( array ) {

return arr.flat.call( array );

} : function( array ) {

return arr.concat.apply( [], array );

};

var push = arr.push;

var indexOf = arr.indexOf;

var class2type = {};

var toString = class2type.toString;

var hasOwn = class2type.hasOwnProperty;

var fnToString = hasOwn.toString;

var ObjectFunctionString = fnToString.call( Object );

var support = {};

var isFunction = function isFunction( obj ) {

return typeof obj === "function" && typeof obj.nodeType !== "number";

};

var isWindow = function isWindow( obj ) {

return obj != null && obj === obj.window;

};

var document = window.document;

var preservedScriptAttributes = {

type: true,

src: true,

nonce: true,

noModule: true

};

function DOMEval( code, node, doc ) {

doc = doc || document;

var i, val,

script = doc.createElement( "script" );

script.text = code;

if ( node ) {

for ( i in preservedScriptAttributes ) {

val = node[ i ] || node.getAttribute && node.getAttribute( i );

if ( val ) {

script.setAttribute( i, val );

}

}

}

doc.head.appendChild( script ).parentNode.removeChild( script );

}

function toType( obj ) {

if ( obj == null ) {

return obj + "";

}

return typeof obj === "object" || typeof obj === "function" ?

class2type[ toString.call( obj ) ] || "object" :

typeof obj;

}

var

version = "3.5.1",

jQuery = function( selector, context ) {

return new jQuery.fn.init( selector, context );

};

jQuery.fn = jQuery.prototype = {

jquery: version,

constructor: jQuery,

length: 0,

toArray: function() {

return slice.call( this );

},

get: function( num ) {

if ( num == null ) {

return slice.call( this );

}

return num < 0 ? this[ num + this.length ] : this[ num ];

},

pushStack: function( elems ) {

var ret = jQuery.merge( this.constructor(), elems );

ret.prevObject = this;

return ret;

},

each: function( callback ) {

return jQuery.each( this, callback );

},

map: function( callback ) {

return this.pushStack( jQuery.map( this, function( elem, i ) {

return callback.call( elem, i, elem );

} ) );

},

slice: function() {

return this.pushStack( slice.apply( this, arguments ) );

},

first: function() {

return this.eq( 0 );

},

last: function() {

return this.eq( -1 );

},

even: function() {

return this.pushStack( jQuery.grep( this, function( \_elem, i ) {

return ( i + 1 ) % 2;

} ) );

},

odd: function() {

return this.pushStack( jQuery.grep( this, function( \_elem, i ) {

return i % 2;

} ) );

},

eq: function( i ) {

var len = this.length,

j = +i + ( i < 0 ? len : 0 );

return this.pushStack( j >= 0 && j < len ? [ this[ j ] ] : [] );

},

end: function() {

return this.prevObject || this.constructor();

},

push: push,

sort: arr.sort,

splice: arr.splice

};

jQuery.extend = jQuery.fn.extend = function() {

var options, name, src, copy, copyIsArray, clone,

target = arguments[ 0 ] || {},

i = 1,

length = arguments.length,

deep = false;

if ( typeof target === "boolean" ) {

deep = target;

target = arguments[ i ] || {};

i++;

}

if ( typeof target !== "object" && !isFunction( target ) ) {

target = {};

}

if ( i === length ) {

target = this;

i--;

}

for ( ; i < length; i++ ) {

if ( ( options = arguments[ i ] ) != null ) {

for ( name in options ) {

copy = options[ name ];

if ( name === "\_\_proto\_\_" || target === copy ) {

continue;

}

if ( deep && copy && ( jQuery.isPlainObject( copy ) ||

( copyIsArray = Array.isArray( copy ) ) ) ) {

src = target[ name ];

if ( copyIsArray && !Array.isArray( src ) ) {

clone = [];

} else if ( !copyIsArray && !jQuery.isPlainObject( src ) ) {

clone = {};

} else {

clone = src;

}

copyIsArray = false;

target[ name ] = jQuery.extend( deep, clone, copy );

} else if ( copy !== undefined ) {

target[ name ] = copy;

}

}

}

}

return target;

};

jQuery.extend( {

expando: "jQuery" + ( version + Math.random() ).replace( /\D/g, "" ),

isReady: true,

error: function( msg ) {

throw new Error( msg );

},

noop: function() {},

isPlainObject: function( obj ) {

var proto, Ctor;

if ( !obj || toString.call( obj ) !== "[object Object]" ) {

return false;

}

proto = getProto( obj );

if ( !proto ) {

return true;

}

Ctor = hasOwn.call( proto, "constructor" ) && proto.constructor;

return typeof Ctor === "function" && fnToString.call( Ctor ) === ObjectFunctionString;

},

isEmptyObject: function( obj ) {

var name;

for ( name in obj ) {

return false;

}

return true;

},

globalEval: function( code, options, doc ) {

DOMEval( code, { nonce: options && options.nonce }, doc );

},

each: function( obj, callback ) {

var length, i = 0;

if ( isArrayLike( obj ) ) {

length = obj.length;

for ( ; i < length; i++ ) {

if ( callback.call( obj[ i ], i, obj[ i ] ) === false ) {

break;

}

}

} else {

for ( i in obj ) {

if ( callback.call( obj[ i ], i, obj[ i ] ) === false ) {

break;

}

}

}

return obj;

},

makeArray: function( arr, results ) {

var ret = results || [];

if ( arr != null ) {

if ( isArrayLike( Object( arr ) ) ) {

jQuery.merge( ret,

typeof arr === "string" ?

[ arr ] : arr

);

} else {

push.call( ret, arr );

}

}

return ret;

},

inArray: function( elem, arr, i ) {

return arr == null ? -1 : indexOf.call( arr, elem, i );

},

merge: function( first, second ) {

var len = +second.length,

j = 0,

i = first.length;

for ( ; j < len; j++ ) {

first[ i++ ] = second[ j ];

}

first.length = i;

return first;

},

grep: function( elems, callback, invert ) {

var callbackInverse,

matches = [],

i = 0,

length = elems.length,

callbackExpect = !invert;

for ( ; i < length; i++ ) {

callbackInverse = !callback( elems[ i ], i );

if ( callbackInverse !== callbackExpect ) {

matches.push( elems[ i ] );

}

}

return matches;

},

map: function( elems, callback, arg ) {

var length, value,

i = 0,

ret = [];

if ( isArrayLike( elems ) ) {

length = elems.length;

for ( ; i < length; i++ ) {

value = callback( elems[ i ], i, arg );

if ( value != null ) {

ret.push( value );

}

}

} else {

for ( i in elems ) {

value = callback( elems[ i ], i, arg );

if ( value != null ) {

ret.push( value );

}

}

}

return flat( ret );

},

guid: 1,

support: support

} );

if ( typeof Symbol === "function" ) {

jQuery.fn[ Symbol.iterator ] = arr[ Symbol.iterator ];

}

jQuery.each( "Boolean Number String Function Array Date RegExp Object Error Symbol".split( " " ),

function( \_i, name ) {

class2type[ "[object " + name + "]" ] = name.toLowerCase();

} );

function isArrayLike( obj ) {

var length = !!obj && "length" in obj && obj.length,

type = toType( obj );

if ( isFunction( obj ) || isWindow( obj ) ) {

return false;

}

return type === "array" || length === 0 ||

typeof length === "number" && length > 0 && ( length - 1 ) in obj;

}

var Sizzle =

( function( window ) {

var i,

support,

Expr,

getText,

isXML,

tokenize,

compile,

select,

outermostContext,

sortInput,

hasDuplicate,

setDocument,

document,

docElem,

documentIsHTML,

rbuggyQSA,

rbuggyMatches,

matches,

contains,

expando = "sizzle" + 1 \* new Date(),

preferredDoc = window.document,

dirruns = 0,

done = 0,

classCache = createCache(),

tokenCache = createCache(),

compilerCache = createCache(),

nonnativeSelectorCache = createCache(),

sortOrder = function( a, b ) {

if ( a === b ) {

hasDuplicate = true;

}

return 0;

},

hasOwn = ( {} ).hasOwnProperty,

arr = [],

pop = arr.pop,

pushNative = arr.push,

push = arr.push,

slice = arr.slice,

indexOf = function( list, elem ) {

var i = 0,

len = list.length;

for ( ; i < len; i++ ) {

if ( list[ i ] === elem ) {

return i;

}

}

return -1;

},

booleans = "checked|selected|async|autofocus|autoplay|controls|defer|disabled|hidden|" +

"ismap|loop|multiple|open|readonly|required|scoped",

whitespace = "[\\x20\\t\\r\\n\\f]",

identifier = "(?:\\\\[\\da-fA-F]{1,6}" + whitespace +

"?|\\\\[^\\r\\n\\f]|[\\w-]|[^\0-\\x7f])+",

attributes = "\\[" + whitespace + "\*(" + identifier + ")(?:" + whitespace +

"\*([\*^$|!~]?=)" + whitespace +

"\*(?:'((?:\\\\.|[^\\\\'])\*)'|\"((?:\\\\.|[^\\\\\"])\*)\"|(" + identifier + "))|)" +

whitespace + "\*\\]",

pseudos = ":(" + identifier + ")(?:\\((" +

"('((?:\\\\.|[^\\\\'])\*)'|\"((?:\\\\.|[^\\\\\"])\*)\")|" +

"((?:\\\\.|[^\\\\()[\\]]|" + attributes + ")\*)|" +

".\*" +

")\\)|)",

rwhitespace = new RegExp( whitespace + "+", "g" ),

rtrim = new RegExp( "^" + whitespace + "+|((?:^|[^\\\\])(?:\\\\.)\*)" +

whitespace + "+$", "g" ),

rcomma = new RegExp( "^" + whitespace + "\*," + whitespace + "\*" ),

rcombinators = new RegExp( "^" + whitespace + "\*([>+~]|" + whitespace + ")" + whitespace +

"\*" ),

rdescend = new RegExp( whitespace + "|>" ),

rpseudo = new RegExp( pseudos ),

ridentifier = new RegExp( "^" + identifier + "$" ),

matchExpr = {

"ID": new RegExp( "^#(" + identifier + ")" ),

"CLASS": new RegExp( "^\\.(" + identifier + ")" ),

"TAG": new RegExp( "^(" + identifier + "|[\*])" ),

"ATTR": new RegExp( "^" + attributes ),

"PSEUDO": new RegExp( "^" + pseudos ),

"CHILD": new RegExp( "^:(only|first|last|nth|nth-last)-(child|of-type)(?:\\(" +

whitespace + "\*(even|odd|(([+-]|)(\\d\*)n|)" + whitespace + "\*(?:([+-]|)" +

whitespace + "\*(\\d+)|))" + whitespace + "\*\\)|)", "i" ),

"bool": new RegExp( "^(?:" + booleans + ")$", "i" ),

"needsContext": new RegExp( "^" + whitespace +

"\*[>+~]|:(even|odd|eq|gt|lt|nth|first|last)(?:\\(" + whitespace +

"\*((?:-\\d)?\\d\*)" + whitespace + "\*\\)|)(?=[^-]|$)", "i" )

},

rhtml = /HTML$/i,

rinputs = /^(?:input|select|textarea|button)$/i,

rheader = /^h\d$/i,

rnative = /^[^{]+\{\s\*\[native \w/,

rquickExpr = /^(?:#([\w-]+)|(\w+)|\.([\w-]+))$/,

rsibling = /[+~]/,

runescape = new RegExp( "\\\\[\\da-fA-F]{1,6}" + whitespace + "?|\\\\([^\\r\\n\\f])", "g" ),

funescape = function( escape, nonHex ) {

var high = "0x" + escape.slice( 1 ) - 0x10000;

return nonHex ?

nonHex :

high < 0 ?

String.fromCharCode( high + 0x10000 ) :

String.fromCharCode( high >> 10 | 0xD800, high & 0x3FF | 0xDC00 );

},

rcssescape = /([\0-\x1f\x7f]|^-?\d)|^-$|[^\0-\x1f\x7f-\uFFFF\w-]/g,

fcssescape = function( ch, asCodePoint ) {

if ( asCodePoint ) {

if ( ch === "\0" ) {

return "\uFFFD";

}

return ch.slice( 0, -1 ) + "\\" +

ch.charCodeAt( ch.length - 1 ).toString( 16 ) + " ";

}

return "\\" + ch;

},

unloadHandler = function() {

setDocument();

},

inDisabledFieldset = addCombinator(

function( elem ) {

return elem.disabled === true && elem.nodeName.toLowerCase() === "fieldset";

},

{ dir: "parentNode", next: "legend" }

);

try {

push.apply(

( arr = slice.call( preferredDoc.childNodes ) ),

preferredDoc.childNodes

);

arr[ preferredDoc.childNodes.length ].nodeType;

} catch ( e ) {

push = { apply: arr.length ?

function( target, els ) {

pushNative.apply( target, slice.call( els ) );

} :

function( target, els ) {

var j = target.length,

i = 0;

while ( ( target[ j++ ] = els[ i++ ] ) ) {}

target.length = j - 1;

}

};

}

function Sizzle( selector, context, results, seed ) {

var m, i, elem, nid, match, groups, newSelector,

newContext = context && context.ownerDocument,

nodeType = context ? context.nodeType : 9;

results = results || [];

if ( typeof selector !== "string" || !selector ||

nodeType !== 1 && nodeType !== 9 && nodeType !== 11 ) {

return results;

}

if ( !seed ) {

setDocument( context );

context = context || document;

if ( documentIsHTML ) {

if ( nodeType !== 11 && ( match = rquickExpr.exec( selector ) ) ) {

if ( ( m = match[ 1 ] ) ) {

if ( nodeType === 9 ) {

if ( ( elem = context.getElementById( m ) ) ) {

if ( elem.id === m ) {

results.push( elem );

return results;

}

} else {

return results;

}

} else {

if ( newContext && ( elem = newContext.getElementById( m ) ) &&

contains( context, elem ) &&

elem.id === m ) {

results.push( elem );

return results;

}

}

} else if ( match[ 2 ] ) {

push.apply( results, context.getElementsByTagName( selector ) );

return results;

} else if ( ( m = match[ 3 ] ) && support.getElementsByClassName &&

context.getElementsByClassName ) {

push.apply( results, context.getElementsByClassName( m ) );

return results;

}

}