Школа:		
Дата:	**ФИО учителя:Джакипбаев Абай Казбекович**	
Класс:	**Участвовали:**	**Не участволвали:**
Тема урока: Скрипты. Подключение скриптов		
Цели обучения, которые достигаются на данном уроке	развития у школьников умения излагать мысли, моделировать ситуацию; повторение и закрепление основного материала, выраженного в неординарных ситуациях; .	
Цели урока	**Все учащиеся смогут**: • Что такое скрипты и для чего они нужны **Большинство учащихся смогут**: • Составить правильно скрипт <script src="scripts.js"></script> **Некоторые учащиеся смогут:** • Поймут отличие атрибутов async и defer друг от друга и использовать	
Критерии оценивания		
Воспитание ценностей	воспитания уважения к окружающим, умения достойно вести спор, воли к познанию нового, находчивости, умения работать в команде	
Предварительные знания		
Межпредметные связи		

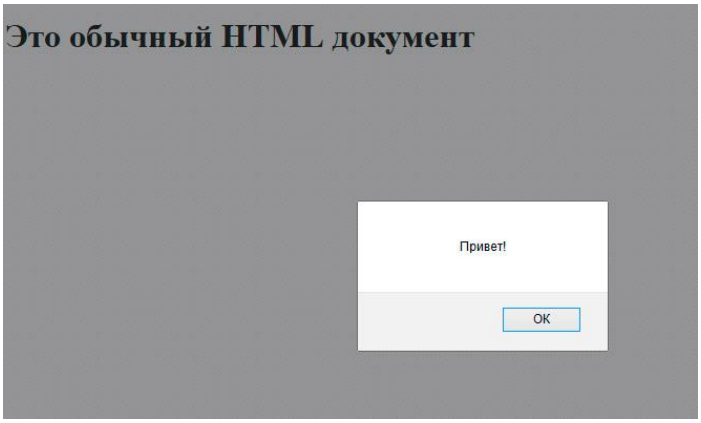
Запланированные этапы урока	**Запланированная деятельность на уроке** **1.Повторение пройденного (игра горячий стол)**	**Ресурсы**
Начало урока **_10__ мин**	Построение, конспектировать, приветствие, сообщение темы и целей обучения. Напоминание о технике безопасности, объяснить критерии успеха. Создать программму с переменными. Робота со скриптами. Отличие атрибутов async и defer друг от друга.	Доска компьютер Проектор интернет

Середина урока **_25__ мин**	Представьте, что ваш скрипт занимает десятки или сотни строк кода. Или даже больше. И, конечно же, этот скрипт нам, скорее всего, потребуется на каждой странице нашего сайта. Согласитесь, будет совсем нехорошо в этом случае дублировать в каждом файле эти сотни строк кода. Да и просто наличие не HTML кода в документе HTML будет смотреться не очень правильно и не очень красиво. Именно поэтому код JavaScript принято выносить в отдельный файл, который и подключается к страничке. Собственно, все так же, как и в случае с файлами стилей. Как же подключить скрипт JavaScript к основному файлу? Очень просто. Для этого используются уже знакомый нам тег <script>, к которому добавляется атрибут src, точно так же, как и в случае с картинками. Ну и, как вы уже догадались, в атрибуте src указывается путь к подключаемому скрипту JavaScript. Давайте попробуем перенести нашу программу, состоящую из одной строки кода, во внешний файл и подключим этот файл. Назовем этот файл, к примеру, scripts.js: Обратите внимание, внутри подключаемого файла нам нужно писать теги script. Мало того, если мы их напишем в файле .js, то наш код JavaScript перестанет работать, и мы получим ошибку JavaScript. Ну и само подключение внешнего файла JavaScript: 1 <script src="scripts.js"></script>	

query.sql | index.html | scripts.js

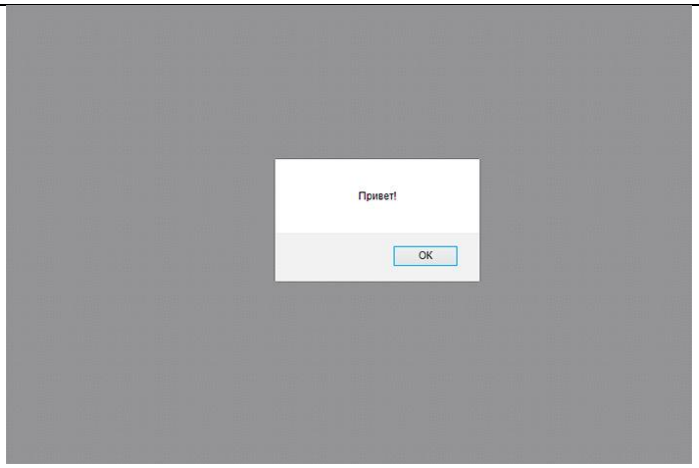
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript</title>
</head>
<body>

    <h1>Это обычный HTML документ</h1>

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

</body>
</html>
```

Если теперь обновить страничку, то ничего не изменится, наш скрипт работает и, по-прежнему, выводит модальное окно с приветствием.



На что стоит обратить внимание при подключении скриптов? Мы подключили скрипт в конце документа, перед закрывающим тегом body. Ранее практиковалось подключение скриптов в начале документа, в тегах head. Однако сегодня так делать не рекомендуется и скрипты рекомендуют подключать именно в конце документа. Почему так?

Давайте попробуем перенести подключения между тегами head:

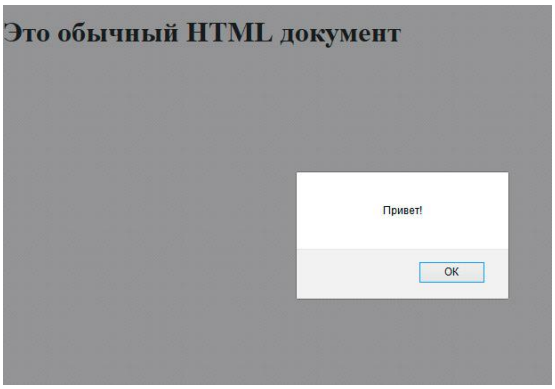


Что мы видим? Ничего, кроме модального окна. Никакого контента нет. В этом и суть. Когда внешний скрипт подключается в начале документа, браузер начинает загружать его и пытается выполнить. И пока загрузка и выполнение скрипта не будут завершены, браузер не покажет часть документа, следующую после подключаемого файла.

А теперь представьте, что этот файл по каким-то причинам загружается крайне медленно. В результате пользователю придется ждать, и порой он может просто не дождаться. Именно поэтому скрипты рекомендуется подключать в конце документа, перед закрывающим тегом body.

Но что делать, если некая библиотека требует подключения именно в начале документа? Как быть в этом случае? В этом случае нас выручат атрибуты async и defer, которые позволяют браузеру загружать скрипты асинхронно, т.е. браузер начнет загружать скрипт, но при этом не остановит показ документа. Попробуем поочередно использовать данные атрибуты:

```
1 <!-- вариант 1 -->
2 <script src="scripts.js" async></script>
```

	3 <!-- вариант 2 --> 4 <script src="scripts.js" defer></script> В обоих случаях мы получим одинаковый результат, скрипт подключается, не прерывая показ документа:  В чем же отличие атрибутов async и defer друг от друга? Атрибут defer сохраняет последовательность подключения внешних скриптов, т.е. первым всегда выполнится тот скрипт, который подключается выше. Это важно в том случае, когда мы подключаем несколько скриптов и какой-то из них может зависеть от другого. В этом случае основной скрипт должен подключаться раньше зависимого. Атрибут defer гарантирует соблюдение порядка. Атрибут async же обеспечит выполнение скрипта сразу после его загрузки. Поэтому такой вариант не всегда подойдет, поскольку зависимый скрипт может загрузиться раньше основного.	
Конец урока **__5_ мин**		
Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными	**Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?**	**Охрана здоровья и соблюдение техники безопасност и**

учащимися?		
Дома пройдите по ссылке и изучите 2-3 урок https://fructcode.com/ru/courses/javascript-and-jquery/interactive-first-javascript/		Тех безопасность Урок подвижный практический упражнение для глаз