СОДЕРЖАНИЕ

[1 jQuery 3](#_Toc501531314)

[1.1 Введение 3](#_Toc501531314)

[1.2 Селекторы 4](#_Toc501531314)

[1.3 События 7](#_Toc501531314)

[1.4 Эффекты 9](#_Toc501531314)

[1.5 DOM](#_Toc501531314) 11

[1.6 CSS 13](#_Toc501531314)

[1.7 AJAX](#_Toc501531314) 14

[1.8 Плагины 17](#_Toc501531314)

[2 Bootstrap 2](#_Toc501531315)0

[2.1 Что такое Bootstrap и зачем он нужен? 20](#_Toc501531314)

2.2 Подключение к сайту 22

2.3 Адаптивный дизайн сайта 22

[2.4 Элементы сетки и как их использовать для создания макета 24](#_Toc501531314)

[2.5 Сетка Bootstrap 4 26](#_Toc501531314)

[2.6 Классы помощники 27](#_Toc501531314)

3 Задания по jQuery 29

4 Задания по Bootstrap 31

ТЕМА 1 - jQuery

* 1. **Краткое введение jQuery**

**Что такое jQuery?**

**jQuery** - это библиотека, которая значительно упрощает и ускоряет написание JavaScript кода.

Девиз jQuery *"write less, do more"* (пиши меньше, делай больше) отражает ее главное предназначение.

jQuery позволяет создавать анимацию, обработчики событий, значительно облегчает выбор элементов в DOM и создание AJAX запросов.

Данная библиотека работает со всеми браузерами (*IE 6.0+, FF 2.0+, Safari 3.0+, Opera 9.0+, Chrome*). Это значит, что больше не нужно будет беспокоиться о кроссбраузерной совместимости JavaScript кода.

Для jQuery написано огромное количество плагинов, которые позволяют расширить ее возможности еще больше.

**Пример использования jQuery**

****

**Рисунок 1.1 –** Пример использования jQuery

**Добавление jQuery на страницы**

Для того, чтобы начать использовать jQuery необходимо:

1. **Скачать ее с официального сайта**. Существуют две версии jQuery: для использования в готовых приложениях (production) и для разработки (development). Версия для разработки содержит комментарии и структурированный код. В сокращенной версии нет комментариев и код в ней не структурирован зато она занимает меньше места и поэтому страницы с ней будут загружаться быстрее. После того, как Вы выберите подходящую версию нажмите на кнопку "Download ( jQuery )"
2. **Добавить ее на страницу**. Для этого следующий код должен быть добавлен на страницу в секцию head:

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

</script>

Для тех кто по каким-либо причинам не может (или не хочет) скачивать jQuery предусмотрен альтернативный способ удаленного использования библиотеки предоставленный компанией Google.

Для того, чтобы использовать jQuery удаленно просто добавьте на страницу в секцию head следующий код:

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.5/jquery.min.js"></script>

* 1. **Селекторы jQuery**

**Команды jQuery**

Код jQuery как и код JavaScript состоит из последовательно идущих **команд**. Команды являются основной структурной единицей jQuery.

**Стандартный синтаксис jQuery команд**:

$(селектор).метод();

1. **Знак $** сообщает, что символы, идущие после него, являются jQuery кодом;
2. **Селектор** позволяет выбрать элемент на странице;
3. **Метод** задает действие, которое необходимо совершить над выбранным элементом. Методы в jQuery разделяются на следующие группы:
   * Методы для манипулирования DOM;
   * Методы для оформления элементов;
   * Методы для создания AJAX запросов;
   * Методы для создания эффектов;
   * Методы для привязки обработчиков событий.

**Символ $**

jQuery можно комбинировать с обычным JavaScript.

Если строка начинается с $ - это jQuery, если $ вначале строки отсутствует - это строка JavaScript кода.

function start() {

//Скроем абзац с id="hid" с помощью jQuery

$("p#hid").hide();

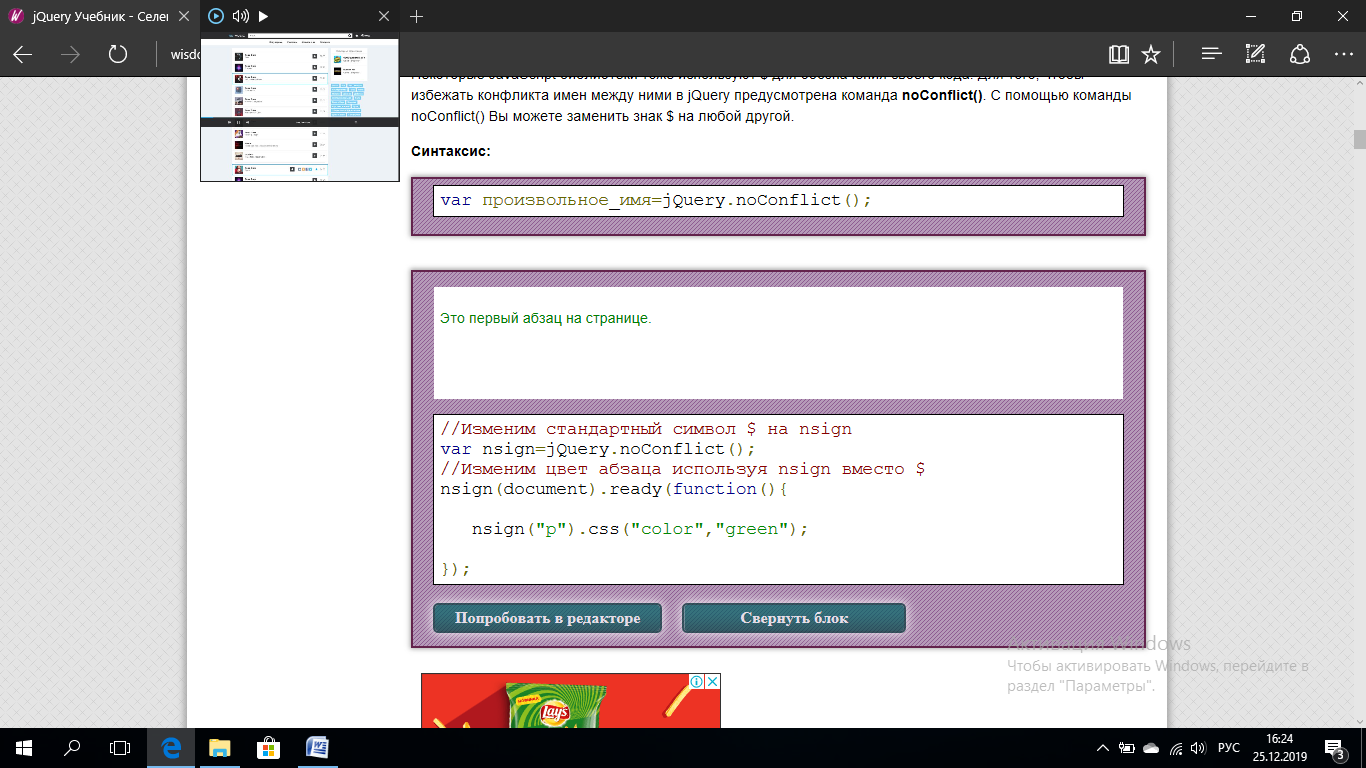
//Изменим содержимое абзаца с id="change"

document.getElementById("change").innerHTML="Данное содержимое было изменено с помощью JavaScript."; }

Некоторые JavaScript библиотеки тоже используют $ для обозначения своего кода. Для того, чтобы избежать конфликта имен между ними в jQuery предусмотрена команда **noConflict()**. С помощью команды noConflict() Вы можете заменить знак $ на любой другой.

**Синтаксис:**

var произвольное\_имя=jQuery.noConflict();

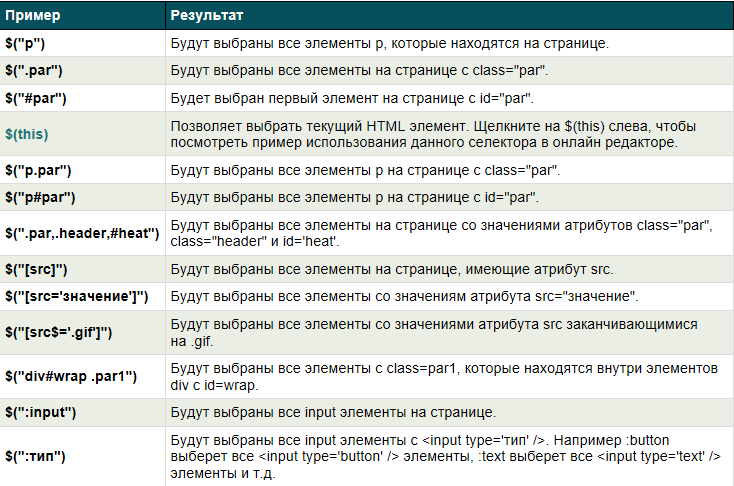
****

**Рисунок 1.2 –** Пример использования команды noConflict()

**Селекторы jQuery**

С помощью селекторов Вы можете выбирать элементы на странице для применения к ним определенных действий.

Ниже располагается таблица с различными примерами использования селекторов для выбора желаемых элементов:



**Рисунок 1.3 –** Пример селекторов

**Предотвращение преждевременного выполнения кода**

Из учебника по JavaScript Вы наверно помните, что выполнение кода до полной загрузки документа часто приводит к ошибкам.

Дело в том, что скрипт может обратиться к еще не загруженному содержимому, а это всегда приводит к ошибкам или неожиданным результатам.

Для того, чтобы избежать этого мы часто помещали код в функцию, которая начинала выполнение только после полной загрузки документа.

**Предотвращение преждевременного выполнения кода в JavaScript:**

<html>

<head>

<script type="text/javascript">

function start()

{ Код который будет выполнен после полной загрузки документа}

</script></head>

<body onload="start()"></body>

</html>

В jQuery можно избавиться от преждевременного выполнения кода следующими способами:

**Предотвращение преждевременного выполнения кода в jQuery:**

<script type='text/javascript'>

//Первый способ:

$(document).ready(function(){

Код который будет выполнен после полной загрузки документа});

//Второй способ:

$().ready(function(){

Код который будет выполнен после полной загрузки документа});

//Третий способ:

$(function(){

Код который будет выполнен после полной загрузки документа});

</script>

Существует еще один альтернативный способ также помогающий избежать преждевременное выполнение JavaScript и jQuery кода и позволяющий также ускорить загрузку страниц.

Необходимо помещать код в самый конец тела документа (т.е. перед </body>) в данном случае интерпретатор JavaScript встроенный в браузер начнет разбирать код только **после загрузки** документа. В предыдущем же способе загрузка скриптов происходит **одновременно с загрузкой** документа, а выполняется этот код после загрузки документа.

<body>

<p>Содержимое тела документа</p>

<div>Содержимое тела документа</div>

<script type='text/javascript'>

Код который будет выполнен после полной загрузки документа</script>

</body>

**Цепочки команд в jQuery**

Для того, чтобы сократить размер кода Вы можете соединять команды jQuery в цепочки.

Команды в цепочке будут выполняться поочередно слева направо.

<script type='text/javascript'>

//Код без сокращения

$("p").css("color","green");

$("p").css("font-size","30px");

//Сокращенный код

$("p").css("color","green").css("font-size","30px");

</script>

* 1. **События jQuery**

**Обработчики событий jQuery**

**Обработчики событий** - это функции, код которых исполняется только после совершения определенных действий.

Обработчики событий присутствовали и в JavaScript, но jQuery облегчает их использование и расширяет их функциональность.

Примеры действий, после которых выполняются обработчики:

* Курсор мыши наведен на элемент;
* Веб-страница или картинка полностью загружена;
* Изменено содержимое поля формы;
* HTML форма отправлена;
* Нажата клавиша на клавиатуре;

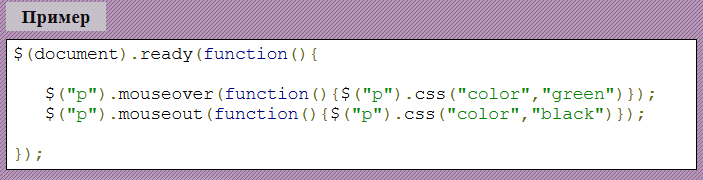
**Общий вид определения обработчиков jQuery:**

$(селектор).обработчик\_события(function(){код\_обработчика\_события}

**Обработчики событий в действии**

Код обработчика **mouseover** будет выполнен, когда курсор мыши будет наведен на элемент.

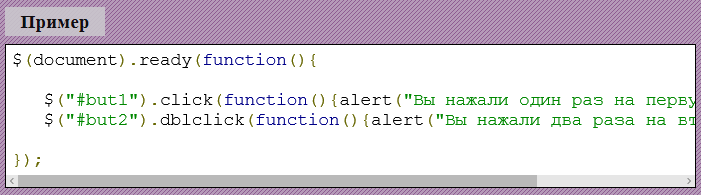
Код обработчика **mouseout** будет выполнен, когда курсор мыши будет выведен из границ элемента.



**Рисунок 1.4 –** Пример

Код обработчика **click** будет выполнен после одинарного щелчка мыши на элементе.

Код обработчика **dblclick** будет выполнен после двойного щелчка мыши на элементе.

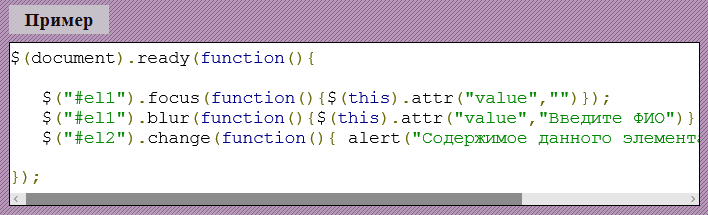


**Рисунок 1.5 –** Пример

Код обработчика **focus()** будет выполнен, когда элемент станет активным.

Код обработчика **blur()** будет выполнен, когда элемент перестанет быть активным.

Код обработчика **change()** будет выполнен, при изменении содержимого элемента.



**Рисунок 1.6 –** Пример

С помощью jQuery Вы можете также вызывать обработчики событий привязанные к элементу.

Например: **$('#test').blur()** вызовет обработчик blur у элемента с id='test'.

$(document).ready(function(){

// Зададим обработчик события, который будет выводить сообщение при нажатии на кнопку с id=but1

$("#but1").click(function(){alert("Вы нажали на кнопку с id=but1")});

// Вызовем обработчик click у элемента с id='but1'

$("#but1").click();});

**Объект event**

**Объект event** содержит информацию о произошедшем событии.

Объект event создается для каждого произошедшего события, но для того, чтобы иметь возможность обращаться к его свойствам и методам его необходимо явно передать в обработчик события.

**Синтаксис:**

$(селектор).событие(function(event){код\_обработчика\_события});

После того, как объект event был передан обработчику события, Вы можете обращаться к его свойствам и методам.

В примере ниже с помощью объекта event мы будем выводить координаты, на которых находился курсор мыши во время того, как произошло событие.

$(document).ready(function(){

$("#square").click(function(event){

$("#coord").css("display","block");

$("#x").html(event.pageX);

$("#y").html(event.pageY);

}); });

**Управление обработчиками событий**

Метод jQuery **one()** позволяет создавать обработчики, которые могут быть вызваны лишь один раз.

$(document).ready(function(){

$("#but1").one("click",function(){alert("Вы нажали на кнопку с id=but1") }); });

Метод jQuery **toggle()** позволяет переключаться между различными обработчиками событий по щелчку мыши.

$(document).ready(function(){

$("#but1").click(function(){

$("#par1").css("color","black");

$("#par1").css("font-size","1em"); });

$("#par1").toggle(function(){

$("#par1").css("color","green");

$("#par1").css("font-size","10px");},function(){

$("#par1").css("color","red");

$("#par1").css("font-size","20px");},function(){

$("#par1").css("color","blue");

$("#par1").css("font-size","30px");},function(){

$("#par1").css("color","black");

$("#par1").css("font-size","1em"); });

});

* 1. **Эффекты jQuery**

С помощью jQuery методов **fadeOut()**, **fadeIn()** и **fadeTo()** Вы можете постепенно скрывать и отображать элементы анимировано.

**Синтаксис:**

//Позволяет постепенно скрыть выбранный элемент

$("селектор").fadeOut(скорость,функция);

//Позволяет постепенно отобразить выбранный элемент

$("селектор").fadeIn(скорость,функция);

//Позволяет постепенно скрыть/отобразить элемент до указанного значения прозрачности

$("селектор").fadeTo(скорость,прозрачность,функция);

**Пример**

$(document).ready(function(){

$("#but1").click(function(){$("#par1").fadeOut(3000)});

$("#but2").click(function(){$("#par1").fadeIn(3000)});

$("#but3").click(function(){$("#par1").fadeTo(3000,0.3)});

$("#but4").click(function(){$("#par1").fadeTo(3000,0.8)});

$("#but5").click(function(){$("#par1").fadeOut(3000,function(){

alert("Абзац был полностью скрыт.");

});

});

});

С помощью jQuery методов **slideUp**, **slideDown** и **slideToggle** Вы можете плавно изменять высоту выбранных элементов.

**Синтаксис:**

//Позволяет изменяет высоту элемента до 0

$("селектор").slideUp(скорость,функция);

//Позволяет плавно возвратить элементу его изначальную высоту

$("селектор").slideDown(скорость,функция);

//При первом вызове будет действовать как slideUp, а при втором как slideDown

$("селектор").slideToggle(скорость,функция);

**Пример**

$(document).ready(function(){

$("#but1").click(function(){$("#square").slideUp(3000)});

$("#but2").click(function(){$("#square").slideDown(3000)});

$("#but3").click(function(){$("#square").slideToggle(3000)});

$("#but4").click(function(){$("#square").slideUp(3000,function(){

alert("Текст был скрыт");

});

});

});

С помощью метода slideToggle Вы можете создавать на Ваших страницах удобные выпадающие меню.

**Пример**

$(document).ready(function(){

$("#menu").click(function(){$("#list").slideToggle(2000)});

$("#menu").toggle(function(){

$("#img").attr("src","menudown.gif")},function(){

$("#img").attr("src","menuup.gif")

});

$("#menu").mouseover(function(){$("#menu").css("background-color","#01939a")});

$("#menu").mouseout(function(){$("#menu").css("background-color","#006064")});

});

**Анимация в jQuery**

С помощью метода **animate()** Вы можете создавать на Ваших страницах полноценную анимацию.

**Синтаксис:**

$("селектор").animate({стили},скорость,функция\_смягчения,функция);

**стили** задает CSS стили для анимации (к элементу одновременно может быть применено несколько стилей).

**скорость** задает скорость анимации. Вы можете указать скорость используя предопределенные свойства: "slow", "fast", "normal" (медленно, быстро, нормально) или указать скорость в миллисекундах (1000 миллисекунд = 1 секунда).

**функция\_смягчения** задает функцию, которая будет отвечать за плавность выполнения анимации.

**функция** указывает имя функции, код которой будет выполнен после завершения анимации.

**Пример**

$(document).ready(function(){

$("#but1").click(function(){

$("p").animate({fontSize:30},2000);

$("p").animate({top:220},2000);

$("p").animate({fontSize:"1em"},2000);

$("p").animate({left:320},2000);

$("p").animate({top:0,left:0},2000);

});

});

**Обратите внимание:** для того, чтобы воспользоваться свойствами позиционирования элемента необходимо вначале установить свойству position значение отличное от static.

* 1. **DOM jQuery**

jQuery содержит набор методов значительно упрощающих взаимодействие с DOM.

Методы, которые будут рассмотрены в данной главе могут применяться и для HTML и для XML документов

**Изменение содержимого элементов с помощью jQuery**

С помощью метода **html()** Вы можете изменить или узнать внутреннее содержимое выбранного элемента.

**Синтаксис:**

//Узнаем содержимое выбранного элемента

$("селектор").html();

//Изменим содержимое выбранного элемента

$("селектор").html("новое содержимое");

**Пример**

$(document).ready(function(){

$("#but1").click(function(){

$("#par1").html("<b>jQuery</b> - это JavaScript библиотека, значительно упрощающая написание кода."); });

$("#but2").click(function(){

$("#par2").html("jQuery значительно облегчает взаимодействие с DOM."); });

$("#but3").click(function(){ alert($("#par1").html()); });

$("#but4").click(function(){ alert($("#par2").html()); });

});

С помощью метода **append()** Вы можете вставить произвольный текст после внутреннего содержимого выбранного элемента.

С помощью метода **prepend()** Вы можете вставить произвольный текст перед внутренним содержимым выбранного элемента.

**Синтаксис:**

//Добавим текст после внутреннего содержимого элемента

$("селектор").append("произвольный текст");

//Добавим текст перед внутренним содержимым элемента

$("селектор").prepend("произвольный текст");

**Пример**

$(document).ready(function(){

$("#but1").click(function(){

$("#par1").prepend("<b>jQuery</b> - это "); });

$("#but2").click(function(){

$("#par1").append(" значительно упрощающая написание кода.");

});

$("#but3").click(function(){

$("body").append("<p>Я добавленный абзац.</p>"); });

});

С помощью метода **attr()** Вы можете узнать или изменить содержимое указанного атрибута у выбранного элемента.

С помощью метода **removeAttr()** Вы можете удалить указанный атрибут у выбранного элемента.

//Узнаем значение произвольного атрибута

$("селектор").attr("атрибут");

//Установим новое значение произвольному атрибуту

$("селектор").attr("атрибут", "новое значение");

//Удалим атрибут

$("селектор").removeAttr("атрибут");

**Пример**

$(document).ready(function(){

$("#but1").click(function(){ alert($("#anchor1").attr("href")); });

$("#but2").click(function(){ $("#anchor1").attr("href","http://"); });

$("#but3").click(function(){ $("#anchor1").removeAttr("href"); });

});

Метод **wrap** позволяет "обернуть" выбранный элемент указанными тэгами.

$("селектор").wrap("<нач\_тэг><кон\_тэг>");

**Пример**

$(document).ready(function(){

$("#but1").click(function(){ $("#par1").wrap("<i></i>"); });

$("#but2").click(function(){ $("#par2").wrap("<b></b>"); });

$("#but3").click(function(){ $("#par3").wrap("<div id='wrap1'></div>");

});});

* 1. **CSS jQuery**

**Управление стилями в jQuery**

jQuery имеет группу различных методов значительно упрощающих оформление элементов.

Одним из самых важных методов в этой группе является метод **css()**.

С помощью метода css Вы можете узнавать текущие или устанавливать новые значения свойств оформления элементов.

**Синтаксис:**

//Узнаем значение указанного CSS свойства выбранного элемента

$("селектор").css("свойство");

//Установим новое значение указанному CSS свойству выбранного элемента

$("селектор").css("свойство","значение");

//Установим произвольные значения нескольким CSS свойствам выбранного элемента

$("селектор").css({свойство1:значение1, свойствоN:значениеN});

**Пример**

$(document).ready(function(){

$("#but1").click(function(){ alert($("#par1").css("color")); });

$("#but2").click(function(){ $("#par2").css("color","red"); });

$("#but3").click(function(){

$("#par3").css({"font-size":"27px","color":"red","font-family":"Arial"});

});

$("#text1").focus(function(){ $("#text1").val(""); });

$("#but4").click(function(){ ("#par4").css("font-size",$("#text1").val()+"px");

});

});

С помощью метода **addClass** Вы можете добавить указанный класс выбранному элементу (данный метод, добавляя новый класс, не удаляет и не заменяет существующие).

С помощью метода **removeClass** Вы можете удалить указанный класс у выбранного элемента.

Метод **toggleClass** позволяет переключаться между удалением и добавлением класса выбранного элемента.

**Синтаксис:**

//Добавим указанный класс выбранному элементу

$("селектор").addClass("имя\_класса");

//Удалим указанный класс у выбранного элемента

$("селектор").removeClass('имя\_класса');

//При первом вызове указанный класс будет добавлен, а при втором удален.

$("селектор").toggleClass('имя\_класса');

**Пример**

$(document).ready(function(){

$("#but1").click(function(){ $("#par1").addClass("blp"); });

$("#but2").click(function(){ $("#par1").removeClass("blp"); });

$("#but3").click(function(){ $("#par1").toggleClass("blp"); });

});

* 1. **AJAX jQuery**

**Создание AJAX в jQuery**

С помощью JavaScript Вы можете создавать асинхронные запросы и отправлять их на сервер.

Использование асинхронных запросов позволяет значительно ускорить загрузку страниц, так как в этом случае обновляться будет только та часть страницы, которая содержит новые данные, а не страница целиком.

Техника использования асинхронных запросов называется **AJAX** - **A**synchronous **J**avaScript **A**nd **X**ML (Асинхронный JavaScript и XML).

Создание AJAX запросов на "чистом" JavaScript имеет несколько недостатков:

* Код даже самого простого AJAX запроса получается достаточно громоздким и сложным для понимания без специального ознакомления.
* Необходимо добавлять дополнительный код для поддержки старых версий браузеров.

AJAX запрос созданный с помощью jQuery занимает всего одну строчку кода, и уже оптимизирован для использования и с новыми и старыми версиями браузеров.

**Синтаксис:**

$("селектор").load(url,данные,функция)

* **селектор** выбирает элемент, в котором будет отображен результат запроса.
* **url** адрес файла, который будет запрошен у сервера с помощью AJAX.
* **данные** которые будут переданы запрошенному файлу. Если Вы хотите запросить файл, при этом не передавать никаких данных, оставьте данный параметр пустым.
* **функция** имя функции, которая будет вызвана после выполнения запроса.

**Пример**

$(document).ready(function(){

$("#but1").click(function(){

$("#par1").load("testfile.txt");

})});

Теперь приведем пример запроса, который пересылает данные.

В примере ниже мы пересылаем скрипту add.php переменную x со значением 4 и переменную y со значением 5 скрипт в свою очередь принимает эти переменные, производит их сложение и отправляет результат обратно.

**Пример**

$(document).ready(function(){

$("#but1").click(function(){

$("#par1").load("add.php","x=4&y=5");

}) });

**Сопровождающие функции**

Сопровождающие функции позволяют выполнять различный код в зависимости от состояния выполнения AJAX запроса.

Метод **ajaxSend()** выполняет переданный в него код во время отправления AJAX запроса.

Метод **ajaxComplete** выполняет переданный в него код после завершения выполнения AJAX запроса (*при этом неважно произошла ошибка или нет*).

**Пример**

$(document).ready(function(){

$("#par1").ajaxSend(function(){ $("#img1").css("display","block"); })

$("#par1").ajaxComplete(function(){$("#img1").css("display","none"); })

$("#but1").click(function(){ $("#par1").load("add.php","x=10&y=100");

}) });

Метод **ajaxSuccess** выполняет переданный в него код, если выполнение AJAX запроса завершается успешно.

Метод **ajaxError** выполняет переданный в него код, если выполнение AJAX запроса завершается с ошибкой.

**Пример**

$(document).ready(function(){

$("#par1").ajaxSend(function(){ $("#img1").css("display","block"); });

$("#par1").ajaxComplete(function(){$("#img1").css("display","none"); });

$("#par1").ajaxError(function(){

alert("Выполнение AJAX запроса завершено с ошибкой.\nПроверьте Ваш запрос на наличие ошибок и попробуйте отправить его еще раз."); });

$("#par1").ajaxSuccess(function(){alert("AJAX запрос успешно выполнен!");

});

$("#but1").click(function(){$("#par1").load("add.php","x=10&y=100"); });

$("#but2").click(function(){$("#par1").load("abb.php","x=10&y=100"); });

});

Методы **ajaxStart** и **ajaxStop** используются для сопровождения группы AJAX запросов.

Метод **ajaxStart** выполняет переданный в него код при отправлении первого запроса из группы.

Метод **ajaxStop** выполняет переданный в него код при завершении последнего запроса из группы.

**Пример**

$(document).ready(function(){

$("#par1").ajaxStart(function(){ $("#wrap").css("display","block"); });

$("#par1").ajaxStop(function(){ $("#wrap").css("display","none"); });

$("#but1").click(function(){

$("#par1").load("add.php","x=5&=10");

$("#par1").load("add.php","x=10&y=100");

$("#par1").load("testfile.txt");

$("#par1").load("add.php","x=45&y=32"); });});

**Низкоуровневые AJAX запросы**

Высокоуровневые методы создания AJAX запросов (т.е. методы которые были разобраны ранее в данной главе) подходят для создания большей части запросов, но иногда их функциональности становится недостаточно.

Низкоуровневые AJAX запросы предоставляют более широкую функциональность, но более сложны в использовании.

**Синтаксис высокоуровневого AJAX запроса:**

$("#par1").load("add.php","x=4&y=5")

**Синтаксис аналогичного низкоуровневого AJAX запроса:**

**Пример**

$("#but1").click(function(){

$.ajax({url:"add.php", data:"x=4&y=5",

success:function(result){ $("#par1").html(result)} }); });

* 1. **Плагины jQuery**

Реализация одних и тех же функций в различных приложениях побуждает разработчиков заново писать один и тот же код несколько раз лишь незначительно изменяя его под конкретное приложение.

Плагины jQuery позволяют забыть разработчикам о данной проблеме. Разработчик может один раз написать плагин, который позволяет реализовать определенную функцию и затем использовать его в необходимых приложениях написав только одну строчку кода.

**Создание плагина**

Для того, чтобы создать плагин необходимо добавить к объекту jQuery.fn свойство, имя которого будет является именем плагина:

**Синтаксис:**

//Определяем код плагина

(function($){

$.fn.имяПлагина = function() {

// Код плагина

};

})(jQuery);

//Вызываем плагин

$(селектор).имяПлагина();

**Обратите внимание:** обертка *(function($){ }) (jQuery)* используется здесь для того, чтобы внутри кода плагина можно было использовать знак $ не боясь при этом, что он будет конфликтовать с другими библиотеками JavaScript.

**Ключевое слово this в плагинах**

При написании плагинов следует обратить внимание на то, что в коде плагина ключевое слово this будет относиться к jQuery объекту, который вызвал этот плагин, а не к DOM объекту как в функциях обратного вызова.

Это значит, что в коде плагина для того, чтобы обратится к jQuery объекту необходимо использовать this, а в коде функций обратного вызова использовать $(this).

**Пример**

(function($){

/\* Создаем плагин с именем plugin, который будет находить сумму содержимого всех выбранных абзацев на странице \*/

$.fn.plugin=function(){

var sum=0;

/\* Для того, чтобы сослаться на выбранный элемент в коде плагина мы используем this, но в функции обратного вызова мы уже должны использовать $(this) для этих же целей \*/

this.each(function(){

sum=sum+Number($(this).html()); });

// Возвращаем полученную сумму

return sum; } })(jQuery)

**Поддержка цепочки методов**

Плагин в предыдущем примере возвращал целое значение, но часто плагины используются для модификации группы элементов и передачи их следующему методу в цепочке.

**Пример цепочки методов:**

//Пример цепочки из двух методов css и html

$("p").css("color","red").html("Новое содержимое");

Для того, чтобы Ваш плагин не "разрывал" цепочку методов необходимо, чтобы он возвращал ключевое слово this.

**Пример**

(function($){

/\* Создаем плагин с именем adjust, который будет устанавливать размер текста выбранных элементов равным 1.4em и передавать их дальше по цепочки методов. \*/

$.fn.adjust=function(){

/\* Для того, чтобы поддержать цепочку методов нужно вернуть из плагина

преобразованную группу выбранных элементов \*/

return this.each(function(){ $(this).css("fontSize","1.4em"); });

}})(jQuery)

**Опции плагинов**

Многие плагины позволяют пользователям настраивать их поведение с помощью передаваемых им опций.

**Пример**

(function($){

/\* Создаем плагин с именем adjust, который будет устанавливать указанный размер и цвет выбранным элементам. \*/

$.fn.adjust=function(option){

/\* Установим значения опций по умолчанию. Они будут использованы если пользователь при вызове плагина не передаст значений. \*/

var setting={size:'1.4em',color:'red'};

return this.each(function(){

// Если пользователь передал опции соединить их с опциями по умолчанию

if (option){$.extend(setting,option);}

$(this).css("fontSize",setting.size);

$(this).css("color",setting.color);

}); } })(jQuery)

**Вынесение кода плагина во внешний файл**

Заключительным шагом в создании плагинов должно стать вынесение кода плагина в отдельный файл. И написания инструкций, которые должны разъяснить потенциальным пользователям, как и зачем можно использовать Ваш плагин.

**Пример**

<script src="http://www.wisdomweb.ru/editor/plugin1.js"></script>

$(document).ready(function(){

$("#but1").click(function(){ $("#demo p").adjust(); });

$("#but2").click(function(){ $("#demo p").adjust({size:"2em"}); });

$("#but3").click(function(){ $("#demo p").adjust({size:"1.7em",color:"green"});

}); });

ТЕМА 2 – BOOTSTRAP

**2.1 Что такое Bootstrap и зачем он нужен?**

**Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.**

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями. На Bootstrap создано очень много различных сайтов.

Основная область его применения – это разработка frontend сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным.

**Почему Bootstrap так популярен?**

Это связано с тем, что он **позволяет верстать сайты в несколько раз быстрее**, чем это можно выполнить на «чистом» CSS и JavaScript. А в нашем мире время – это самый дорогой ресурс. Также его популярность связана с доступностью. Она заключается в том, что на нём даже начинающий разработчик может верстать достаточно качественные макеты, которые трудно было бы выполнить без глубоких знаний веб-технологий и достаточной практики.

Фреймворк Bootstrap представляет собой набор CSS и JavaScript файлов. Чтобы его использовать на странице эти файлы нужно просто подключить к ней. После подключения вам станут доступны инструменты данного фреймворка: колоночная система (сетка Bootstrap), классы и компоненты.

Как это работает? Например, чтобы создать кнопку на странице с помощью Bootstrap, достаточно к ссылке или элементу button добавить всего несколько классов.

**Bootstrap состоит из:**

* сетки;
* классов для стилизации текста, изображений, таблиц и другого контента;
* компонентов предназначенных для создания кнопок, различных форм на странице, горизонтальных и вертикальных навигационных меню, слайдеров, выпадающих списков, аккордеонов, модальных окон, всплывающих подсказок и других элементов интерфейса;
* классов для решения вспомогательных задач наиболее часто возникающими перед веб-разработчиками (выравнивание текста, скрытие или отображение элемента, задания цвета и фона элементу, задание margin и padding отступов, и др.).

Преимущества, которые даёт фреймворк Bootstrap при разработке на его основе frontend части сайтов и интерфейсов админок:

* высокая скорость создания качественной адаптивной вёрстки даже начинающими веб-разработчиками (достигается это благодаря использованию готовых компонентов, созданных профессионалами);
* кроссбраузерность и кроссплатформеннось (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);
* наличие большого количество готовых хорошо продуманных компонентов, протестированных огромным сообществом веб-разработчиков на различных устройствах;
* возможность настройки под свой проект, достигается это посредством изменения SCSS переменных и использования Bootstrap миксинов (можно изменить количество колонок, цвета, радиуса скруглений углов элементов, отступы между колонками и многое другое);
* низкий порог вхождения; для работы с фреймворком не обязательно иметь глубокие знания по HTML, CSS, JavaScript и jQuery (достаточно знать только основы вышеперечисленных технологий);
* наличие хорошо продуманного дизайна компонентов и согласованности (в Bootstrap все компоненты выполнены в едином стиле);
* наличие огромного сообщества, большого количества статей, рецептов и видеоматериалов; всё это при желании поможет не только хорошо разобраться в фреймворке, но и найти ответы практически на любые вопросы.

Фреймворк Bootstrap – это проект с открытым исходным кодом, доступным на Github. Он имеет лицензию MIT. Это означает, что его можно использовать бесплатно как в личных, так и в коммерческих проектах.

На Bootstrap, конечно, верстают далеко не всё. Если проект большой или имеет уникальный дизайн, или если вы работаете на заказ и клиент готов платить за проект на «чистом» CSS и JavaScript достаточную сумму, то в этом случае Bootstrap конечно не имеет смысла использовать. Но такая разработка в основном осуществляется в команде, в которой каждый её участник выполняет какой-то свой определенный набор функций.

Тоже касается и личных проектов, если у вас есть достаточно количество времени и ваш уровень знания технологий HTML, CSS и JavaScript достаточный, чтобы это осуществить, то тогда Bootstrap вам не нужен.

Bootstrap, как и большинство подобных фреймворков, имеет недостатки. Один из них заключается в том, что его нет смысла использовать для вёрстки сайтов, когда это требует переписывания большого количества CSS кода и простой настройкой переменных Bootstrap, например, тут не обойтись.

Другой недостаток связан с тем, что код CSS и JavaScript вашего проекта будет значительно больше по размеру, чем если бы вы его самостоятельно написали конкретно под этот проект. Ну, это понятно, т.к. стили, да и код JavaScript, заточены в Bootstrap под различные сценарии, которые в вашем проекте могут просто не использоваться. Уменьшить код фреймворка Bootstrap можно выполнив самостоятельную сборку проекта из исходных кодов и включить в неё только те компоненты, которые нужны.

**2.2 Подключение к сайту**

## **Набор инструментов для изучения Bootstrap**

Минимальный набор инструментов (программ) для создания веб-проектов на фреймворке Bootstrap:

* текстовый редактор для работы с кодом ("Блокнот", "Brackets", "Notepad++" или др.);
* браузер (Google Chrome, Mozilla Firefox, Opera, Safari, Internet Explorer или др.).

## **Загрузка фреймворка Bootstrap.**

Создание веб-проекта, в основу дизайна которого положен фреймворк Bootstrap всегда начинают с его загрузки. Загрузить фреймворк Bootstrap можно **различными способами**. Например, посредством ссылки, расположенной на сайте getbootstrap.com или с помощью пакетного менеджера npm, Composer, Bower или др. Способ как это осуществить зависит **от вашего опыта или конкретной ситуации**.

После скачивания архива (с готовыми к применению CSS и JavaScript файлами), его необходимо распаковать в каталог вашего веб-проекта.

В каталоге css находятся стили фреймворка Bootstrap, а в js - плагины для обеспечения работы некоторых компонентов. Плагины написаны с использованием функций библиотеки jQuery. Поэтому перед Bootstrap JS необходимо подключить библиотеку jQuery.

## **Подключение Bootstrap к HTML странице**

Процесс установки фреймворка Bootstrap 3 состоит из подключения следующих файлов к HTML 5 странице:

1. Bootstrap CSS (bootstrap.min.css);
2. Последней версии библиотеки jQuery (необходима для работы JS плагинов Bootstrap);
3. Bootstrap JavaScript (bootstrap.min.js).

**2.3 Адаптивный дизайн сайта**

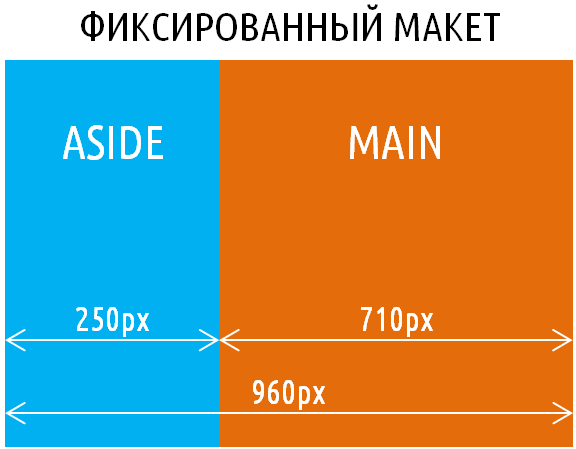
## **Основные виды макетов сайтов**

Различают следующие основные типы макетов сайтов, связанных с шириной:

* фиксированный;
* резиновый (гибкий);
* адаптивный.

## **Фиксированный макет сайта**

Фиксированный макет сайта - это макет, который имеет строго определённую ширину (в пикселях). Такой вид дизайна широко использовался для разработки сайтов в прошлом, когда интернет только начал внедряться в нашу жизнь. В это время "господствовали" компьютеры с мониторами, имеющими диагональ 14" или 15", в редких случаях встречались "монстры", имеющие диагональ 17". Эти мониторы не сильно отличались по горизонтальному разрешению. Таким образом, выбрав для разметки какую-то определённую ширину (в основном останавливались на 960px), можно было разработать сайт, который оптимально бы отображался на всех мониторах этого времени.



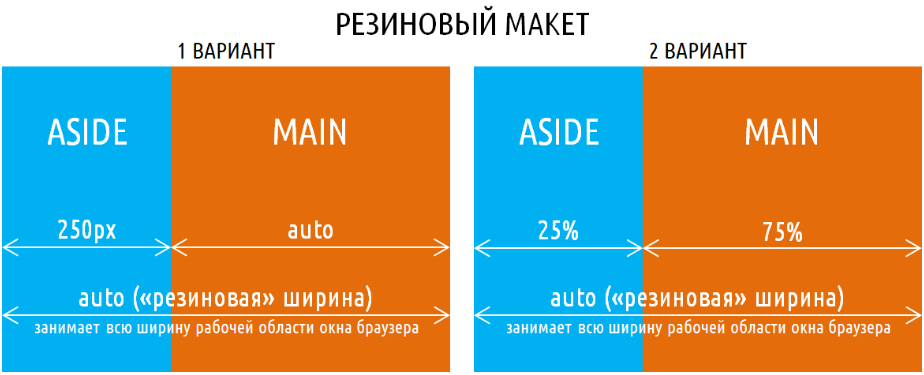
**Рисунок 2.1** – Фиксированный макет

## **Резиновый (гибкий) макет сайта**

Резиновый (гибкий) макет сайта - это макет, который изменяется в зависимости от ширины рабочей области окна (вкладки) браузера. Потребность в такой разметке возникла, когда у многих пользователей появились мониторы, имеющие диагональ 19", 21" и больше.

В этот период времени сложилась такая ситуация, когда у одних пользователей были маленькие мониторы, а у других средние и большие. Разрешения, которые были установлены на этих мониторах стали сильно отличаться, и уже трудно было выбрать для макета какую-то определённую ширину.

Выходом из этой ситуации стало создание разметки, в которой блоки имеют не строго фиксированную ширину, а резиновую (обычно заданную в процентах). Такой макет в отличие от фиксированного уже мог «подстраиваться» под размеры монитора, т.е. изменять свои размеры.



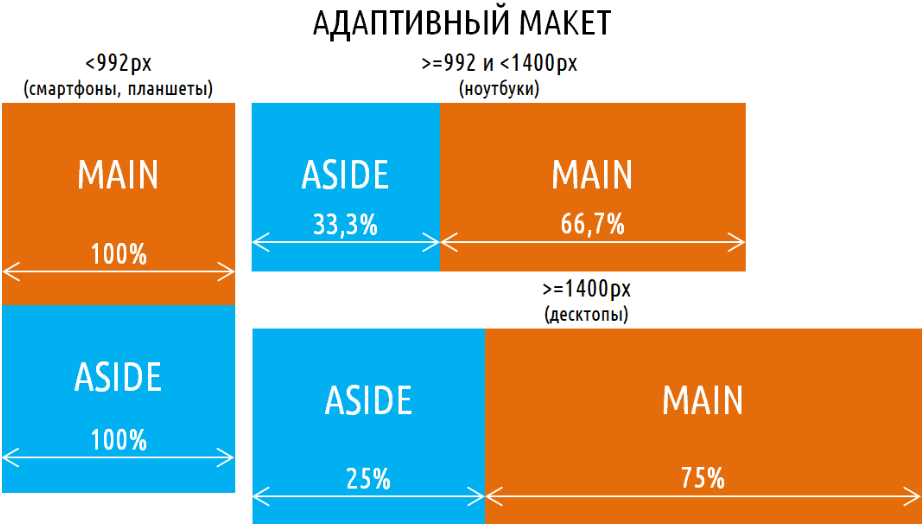
**Рисунок 2.2** – Резиновый макет

## **Адаптивный макет сайта**

Адаптивный макет сайта - это макет, который может «приспосабливаться» под различные устройства (ширину рабочей области окна браузера). Т.е. на одних устройствах он может иметь одну структуру, а на других - другую.

А именно:

* на смартфонах и планшетах (устройствах с очень маленьким размером экрана) блоки должны располагаться вертикально, т.е. один под другим;
* на ноутбуках (устройствах со средним размером экрана) блоки должны располагаться горизонтально (1 блок - 33.3%, 2 блок - 66.7%);
* на десктопах (устройствах с большим размером экрана) тоже горизонтально, но с другими размерами (1 блок - 25%, 2 блок - 75%)



**Рисунок 2.3** – Адаптивный макет

**2.4 Элементы сетки и как их использовать для создания макета**

## **Элементы сетки фреймворков Bootstrap 3 и 4**

Основными элементами сетки Bootstrap 3 и 4 являются:

* обёрточные контейнеры - элементы с классом container или .container-fluid;
* ряды - элемент с классом row;
* адаптивные блоки - элементы, с одним или несколькими классами col.

Обёрточный контейнер - это первый элемент, с которого начинается создание макета страницы или некоторой его самостоятельной части. Его основное назначение - это **установить ширину разрабатываемого макета**. В Bootstrap 3 и 4 обёрточные контейнеры бывают 2 типов. Первый (container) предназначен для создания адаптивно-фиксированного макета, а второй (container-fluid) - для адаптивно-резинового (адаптивно-гибкого) макета.

Адаптивно-фиксированный макет характеризуется тем, что он имеет условно постоянную ширину, которая на одних диазонах viewport браузера имеет одно значение, а на дугих - другое.

Обёрточный контейнер (container) устанавливает макету:

* на дипазоне xs ширину, равную ширине viewport браузера;
* на дипазоне sm, ширину равную 750px;
* на дипазоне md, ширину равную 970px;
* на дипазоне lg, ширину равную 1170px.

Ширина же адаптивно-резинового макета не имеет фиксированного значения, она всегда равна ширине vieport браузера.

Обёрточный контейнер кроме установления ширины макету ещё выравнивает его по центру страницы и задаёт внутренние поля (padding) слева и справа по 15px.

Ряд - это тоже контейнер, но для адаптивных блоков сетки Bootstrap.

В Bootstrap 3 его основная роль - это создать отрицательные отступы (margin) слева и справа по 15px.

В Boostrap 4 он не только задаёт отрицательные отступы, но и выполняет ещё функцию flex-контейнера. Т.е. если данный элемент не установить, то адаптивные блоки вообще не будут иметь свойственного им поведения.

Принцип использования элемента "ряд" очень прост, он всегда **должен выступать родителем для адаптивных блоков**. Т.е. если какой-то элемент (обёрточный контейнер или адаптивный блок) необходимо разметить с помощью адаптивных блоков, то перед тем как их создать сначала установите ряд, а уже в нём эти блоки.

Адаптивный блок - это элемент, который имеют адаптивную ширину. Т.е. его ширина на одном диапазоне viewport может иметь одно значение, а на другом - другое.

Установка поведения адаптивного блока осуществляется с помощью одного или нескольких классов col.

Синтаксис класса col:

*col-{breakpoint}-{number\_columns}*

{breakpoint} - это контрольная точка, которая определяет минимальную ширину viewport, начиная с которой этот класс будет действовать.

В Bootstrap 3 для использования доступно по умолчанию четыре контрольные точки (xs, sm, md и lg), а в Bootstrap 4 - пять контрольных точек (без обозначения, sm, md, lg и xl). Контрольные точки приведены в порядке возрастания ширины viewport, начиная с которой они начинают действовать.

{number\_columns} - это ширина адаптивного блока, которую он будет иметь, начиная с этой контрольной точки. Указывается ширина адаптивного блока с помощью колонок Bootstrap (целого числа) по умолчанию от 1 до 12. Данное число определяет, **какую часть ширины он будет иметь относительно содержащего его блока** (элемента «ряд»). Минимальная ширина адаптивного блока - это 1/12 (8.3%), а максимальная - 12/12 (100%).

## **Основные правила создания макета с помощью элементов сетки Bootstrap**

Основные этапы создания макета веб-страницы:

1. создать основные секции (например: header, main, footer);
2. создать внутри каждой секции обёрточный контейнер;
3. поместить внутрь каждого обёрточного контейнера, разметку которых необходимо произвести с помощью адаптивных блоков, элемент «ряд»;
4. создать внутри каждого ряда необходимую структуру с помощью адаптивных блоков;
5. поместить внутрь необходимых адаптивных блоков, разметку которых необходимо произвести с помощью адаптивных блоков, элемент «ряд»;
6. выполнять пункты 5 и 6 до тех пор, пока не будет достигнута необходимая структура создаваемого макета.

**2.5 Сетка Bootstrap 4**

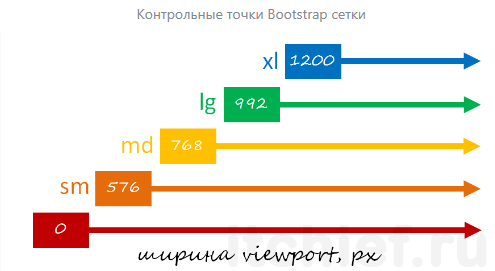
Сетку Bootstrap 4 можно использовать отдельно (без других компонентов). Этот вариант применения фреймворка тоже используется веб-разработчиками, т.к. не всегда готовые Bootstrap элементы, такие как кнопки, навигационные панели, таблицы и прочие компоненты можно использовать в рамках того или иного проекта. Зачастую это касается проектов с уникальным дизайном. В этом случае нужные элементы проще написать с нуля самостоятельно, вместо того чтобы переписывать готовые компоненты Bootstrap.

## **Назначение Bootstrap сетки. Контрольные точки**

**Сетка Bootstrap 4 – это основная часть фреймворка. Она предназначена для создания адаптивных шаблонов.**

Сетка Bootstrap 4 основывается на [CSS Flexbox](https://itchief.ru/html-and-css/flexbox) и [медиа-запросах](https://itchief.ru/lessons/html-and-css/css-media-queries) (CSS media queries).

Если кратко, адаптивный макет – это такой макет, вид которого может изменяться в зависимости от того, какую ширину [viewport](https://itchief.ru/lessons/html-and-css/meta-viewport-how-it-works) имеет браузер. Это означает, что при одних значениях ширины viewport адаптивный макет может выглядеть одним образом, а при иных – совершенно по-другому.



**Рисунок 2.4** – Контрольные точки сетки

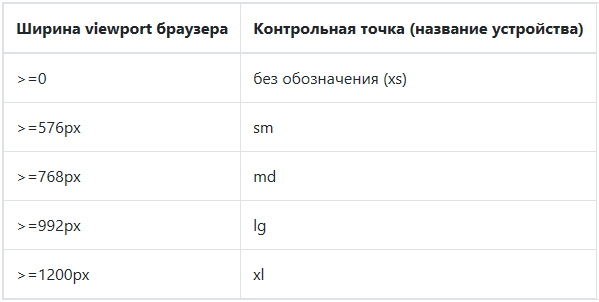
Это означает что до 576px макет сайта может отображаться одним образом, от 576px до 768px – другим образом и т.д. Таким образом, можно создать макет, который на каждом из этих участков может выглядеть по-разномуv

Контрольные точки имеют обозначение. Первый breakpoint не имеет обозначения (xs), второй называется – sm, третий – md, четвёртый – lg и пятый – xl. Эти обозначения необходимо запомнить, т.к. они используются в классах, которые мы будем добавлять к элементам. Эти обозначения в имени класса будут указывать на то с какой ширины viewport стили, определённые в нём, будут применяться к элементу

**Сетка Bootstrap 4 состоит из:**

* обёрточных контейнеров (элементов с классом container и container-fluid);
* рядов (элементов с классом row):
* адаптивных блоков (элементов, имеющих один или несколько классов, равных col или начинающихся с col).

Все эти части сетки – это обычные HTML элементы, к которым просто добавлены определённые классы.



**Рисунок 2.5** – Обозначения контрольных точек

**2.6 Классы помощники**

Основное назначение flex классов Bootstrap - это удобное и быстрое создание flex-контейнеров, а также адаптивное (на всех контрольных точках) управление раскладкой flex-элементов внутри них.

## **Создание flex-контейнера**

Для того чтобы в Bootstrap 4 некоторый элемент на странице сделать flex-контейнером к нему необходимо добавить класс d-flex или d-inline-flex. После этого все дочерние элементы flex-контейнера (дети) автоматически станут flex-элементами.

Первый класс (d-flex) используется, когда flex-контейнер необходимо представить по отношению к окружающим его элементам (neighbor, соседям) блочным (block). А второй класс (d-inline-flex) применяется тогда, когда его нужно представить строчным (inline).

*<!-- Блочный flex-контейнер -->*

*<div class="d-flex"></div>*

*<!-- Строчный flex-контейнер -->*

*<div class="d-inline-flex"></div>*

Если вам нужно сделать элемент flex-контейнером, только начиная с определённых контрольных точек, то используйте следующие классы

* d-sm-flex
* d-sm-inline-flex
* d-md-flex
* d-md-inline-flex
* d-lg-flex
* d-lg-inline-flex
* d-xl-flex
* d-xl-inline-flex

## **Расположение flex-элементов во flex-контейнере**

Выкладывание flex-элементов во flex-контейнере осуществляется линиями.

Направление расположения элементов в пределах линии определяется главной осью. По умолчанию она направлена горизонтально слева направо (flex-direction: row).

Тем не менее, бывают ситуации, когда вам необходимо установить это явно. Для этого к flex-контейнеру необходимо добавить класс flex-row.

Задание обратного порядка следования flex-элементов во flex-контейнере осуществляется посредством установления главной оси направления справа налево (flex-direction: row-reverse). В Bootstrap 4 это действие осуществляется с помощью класса flex-row-reverse.

Изменение направления основной оси с горизонтального на вертикальное во flex-контейнере осуществляется с помощью класса flex-column или flex-column-reverse. Последний класс отличается от первого тем, что ось будет направлена не сверху вниз (flex-direction: column), а снизу вверх (flex-direction: column-reverse).

ТЕМА 3 – ЗАДАНИЯ ПО jQuery

**Задание 1** Используя add-методы фреймворка jQuery добавить новые элементы / контент.

Add-методы jQuery:

append() - Вставляет содержимое в конце выбранных элементов

prepend() - Вставляет содержимое в начало выбранных элементов

after() - Вставляет содержимое после выбранного элемента

before() - Вставляет содержимое перед выбранным элементом

Использовать как минимум два из предложенных методов.

**Листинг 3.1** - prepend()

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn1").click(function(){

$("p").prepend("<b>Prepended text</b>. ");

});

$("#btn2").click(function(){

$("ol").prepend("<li>Prepended item</li>");

});

});

</script>

</head>

<body>

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

<ol>

<li>List item 1</li>

<li>List item 2</li>

<li>List item 3</li>

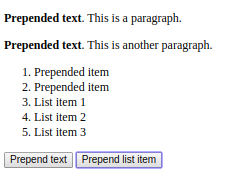
</ol>

<button id="btn1">Prepend text</button>

<button id="btn2">Prepend list item</button>

</body>

</html>

**Рисунок 3.1 – Результат использования prepend()**

**Листинг 3.2** - after()

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("#btn2").click(function(){

$("img").after("<i>After</i>");

});

});

</script>

</head>

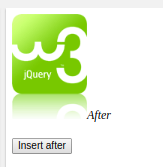
<body>

<img src="/images/w3jquery.gif" alt="jQuery" width="100" height="140"><br><br>

<button id="btn2">Insert after</button>

</body>

</html>



**Рисунок 3.2 – Результат использования after()**

ТЕМА 4 – ЗАДАНИЯ ПО BOOTSTRAP

**Задание 1 – разработка адаптивной разметки используя классы-контейнеры Bootstrap.**

В Bootstrap Вы можете использовать такие классы как .container-sm | md | lg | xl для создания адаптивных контейнеров. Применяя их максимальная ширина контейнера будет меняться на разных размерах экрана / окнах просмотра.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Класс** | **<576px** | **≥576px** | **≥768px** | **≥992px** | **≥1200px** |
| .container-sm | 100% | 540px | 720px | 960px | 1140px |
| .container-md | 100% | 100% | 720px | 960px | 1140px |
| .container-lg | 100% | 100% | 100% | 960px | 1140px |
| .container-xl | 100% | 100% | 100% | 100% | 1140px |

Пример использования классов:

<div class="container-sm">.container-sm</div>

<div class="container-md">.container-md</div>

<div class="container-lg">.container-lg</div>

<div class="container-xl">.container-xl</div>

Разметка адаптивного контейнера:

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>

</head>

<body>

<div class="container">

<h1>Responsive Containers</h1>

<p>Resize the browser window to see the effect.</p>

</div>

<div class="container-sm border">.container-sm</div>

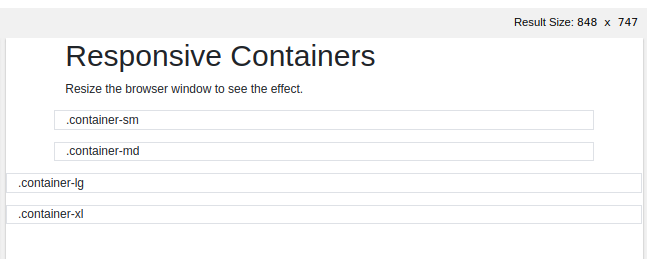
<div class="container-md mt-3 border">.container-md</div>

<div class="container-lg mt-3 border">.container-lg</div>

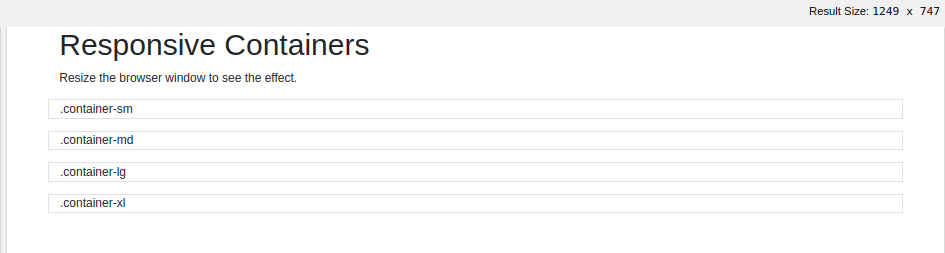
<div class="container-xl mt-3 border">.container-xl</div>

</body>

</html>



**Рисунок 4.1** – Результат выполнения задания 1: ширина области 848



**Рисунок 4.2** – Результат выполнения задания 1: ширина области 1249