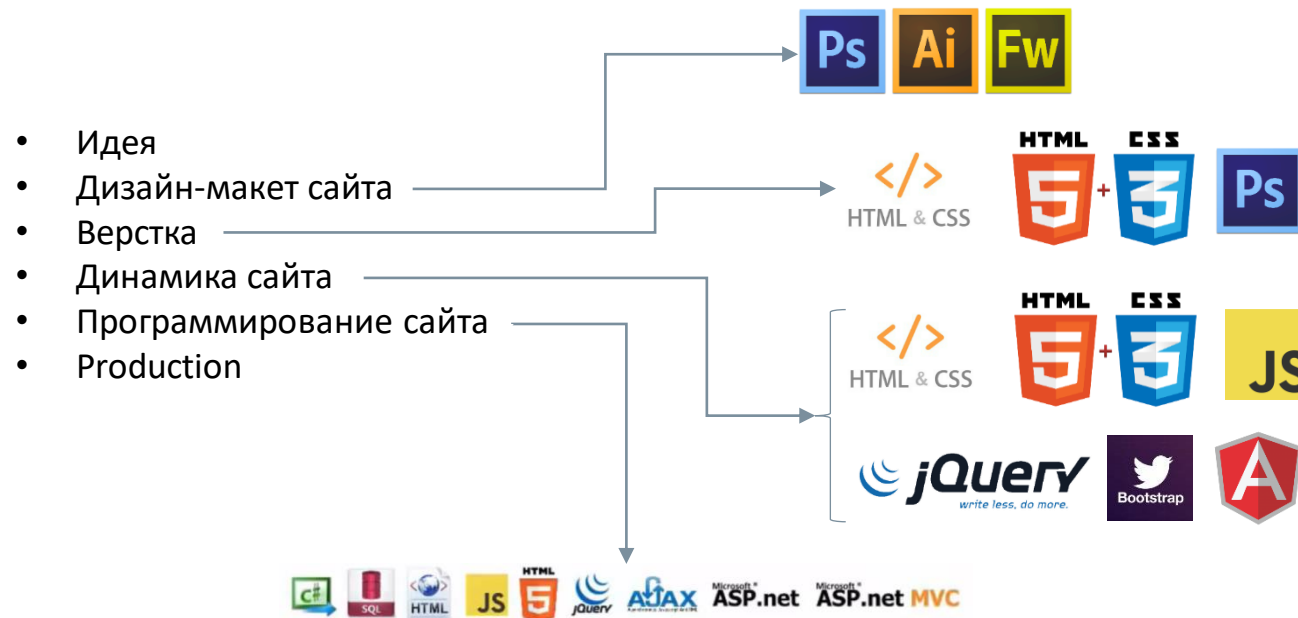


Знакомство с Web-приложениями.
Основные понятия о HTML, CSS.
Введение в Bootstrap.
Протокол HTTP.
Первое ASP.Net приложение.

Этапы создания Web-приложения



HTML

Hyper Text Markup Language (HTML, язык гипертекстовой разметки) — стандартизированный язык разметки документов во Всемирной паутине.

Я **очень** люблю:

- компьютеры
- кошек (*иногда*)

Я **очень** люблю:

компьютеры

кошек (иногда)



Проверить соответствие стандартам W3C:

<http://validator.w3.org>

Зачем так сложно?

Различное форматирование для различных устройств:

```
Я <strong>очень</strong> люблю:  
<ul>  
  <li>компьютеры</li>  
  <li>кошек (<em>иногда</em>)</li>  
</ul>
```



Я **очень** люблю:
• компьютеры
• кошек (*иногда*)



Я **ОЧЕНЬ** люблю:
* компьютеры
* кошек (иногда)

Некоторые термины

- **Элемент** (element) — конструкция языка HTML. Это контейнер, содержащий данные и позволяющий отформатировать их определенным образом.
- **Тег** (tag) — начальный или конечный маркеры элемента. Теги определяют границы действия элементов и отделяют элементы друг от друга. В тексте Web-страницы теги заключаются в угловые скобки, а конечный тег всегда снабжается косой чертой.
- **Атрибут** (attribute) — параметр или свойство элемента. Это переменная, которая имеет стандартное имя и которой может присваиваться определенный набор значений. Атрибуты располагаются внутри начального тега и отделяются друг от друга пробелами.
- **Гиперссылка** (hyperlink) — фрагмент текста, который является указателем на другой документ или файл/объект. Гиперссылки необходимы для того, чтобы обеспечить возможность перехода от одного документа к другому.
- **Web-страница** (web page) — документ (файл), подготовленный в формате гипертекста и размещенный в World Wide Web.
- **Сайт** (site) — набор Web-страниц, принадлежащих одному домену.

HTML: основные понятия

Я очень люблю:

 ком Одиночный тег
 кошек <i>иногда</i>

</br>
Тег

HTML: структура страницы

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>My Book Catalog</title>
```

```
  </head>
```

```
  <body>
```

```
    ...
```

```
  </body>
```

```
</html>
```

Заголовок
(описание)

Тело страницы

Каскадные таблицы стилей: основная идея

Стили – это гибкий способ переопределять форматирование элементов на странице.

Самое главное:

- HTML-разметка определяет содержимое (контент);
- CSS-разметка определяет внешний вид (дизайн).

Заменой стилей можно легко менять отображение одной и той же информации для различных устройств!

HTML + CSS

Page.html

```
Я <b class="megared">очень</b>  
люблю:  
...
```

Styles.css

```
.megared {  
    font-size: 20pt;  
    color: red;  
}
```

Каскадные таблицы стилей: где размещать

Такие стили называются «каскадными», поскольку для любого стиля можно переопределять отдельные атрибуты.

Стили «наследуются» в порядке появления элементов на странице и могут объявляться:

- во внешних файлах CSS: `<link rel="stylesheet" type="text/css" href="..." />`
- внутри страницы (тег style): `<style>...</style>`
- внутри тега (атрибут style): ``

Возможно переопределять форматирование:

- С явным указанием класса: `.class { ... }`
- Для всех тегов: `h1 { color: red; }`
- Для заданного id тега: `#id { ... }`

Каскадные таблицы стилей: селекторы и объявления






- ✓ **CSS (Cascading Style Sheets)**, или каскадные таблицы стилей, описывают правила форматирования отдельного элемента веб-страницы.
- ✓ Создав стиль один раз, его можно применять к любым элементам страницы сколько угодно раз.
- ✓ Определение стиля состоит из двух основных частей: самого элемента веб-страницы – **селектора**, и команды форматирования – **блока объявления**.
- ✓ Селектор сообщает браузеру, какой именно элемент форматировать, в блоке объявления перечисляются формирующие команды.



Библиотека Bootstrap

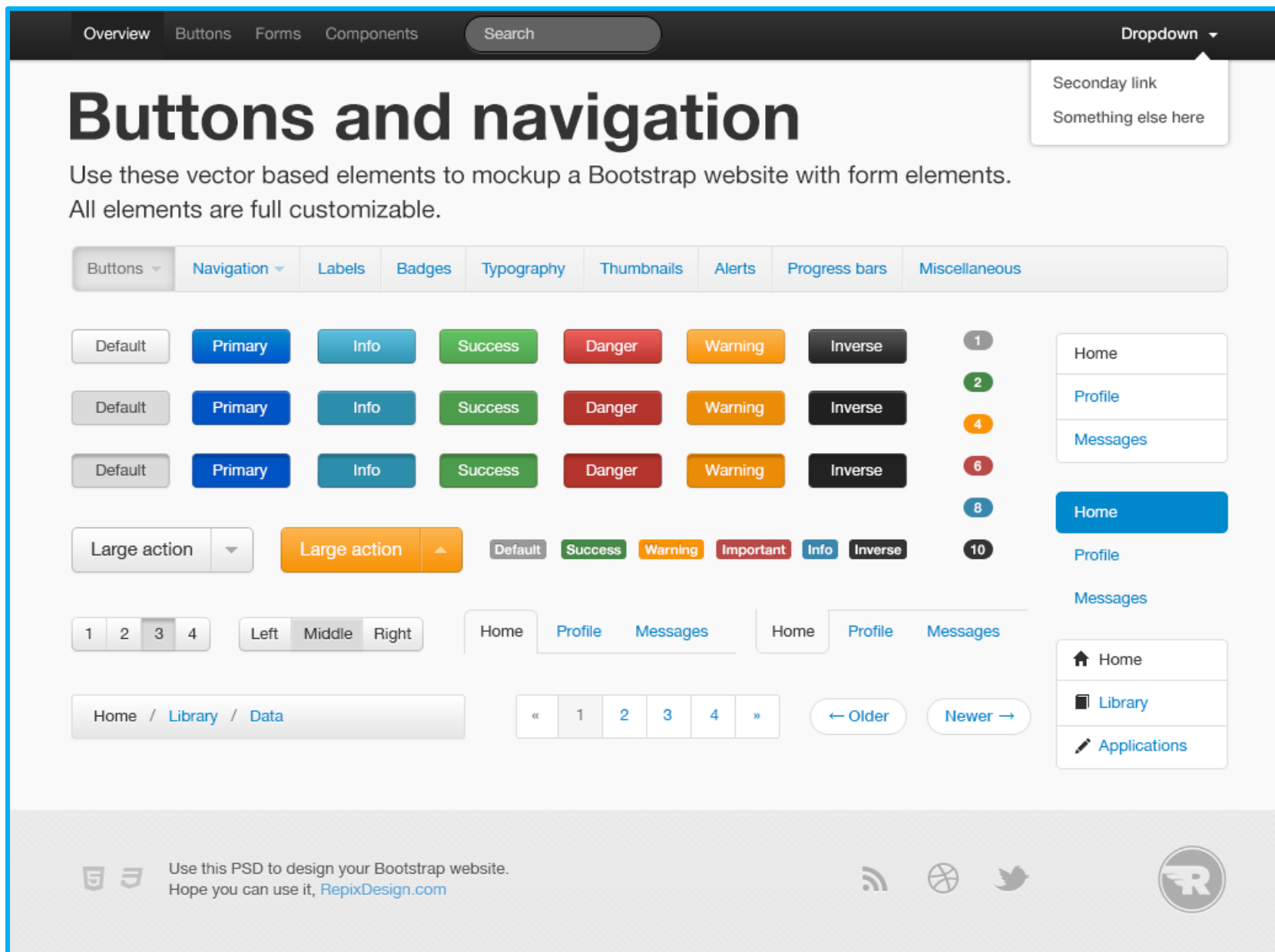


- **Bootstrap** — это открытая и бесплатная HTML, CSS и JS **библиотека** для быстрой вёрстки адаптивных сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса.
- Текущая версия – Bootstrap 5.1.3
- Сетки в версиях Bootstrap 4+ основаны на технологии flex.
- Официальная документация Bootstrap на английском языке <https://getbootstrap.com/>
- Документация на русском языке <http://bootstrap-4.ru/>

				
Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px



Bootstrap – библиотека визуальных компонентов



Bootstrap – библиотека визуальных компонентов



Код для них такой:

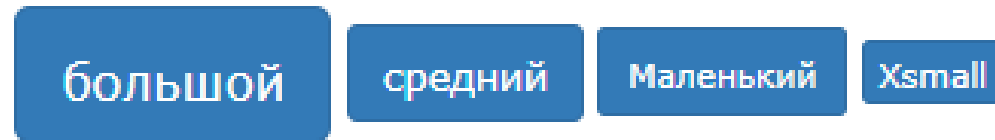
```
<button type="button" class="btn btn-default">Кнопка</button>  
<button type="button" class="btn btn-primary">Основная</button>  
<button type="button" class="btn btn-success">Успех</button>  
<button type="button" class="btn btn-info">Информация</button>  
<button type="button" class="btn btn-warning">Внимание</button>  
<button type="button" class="btn btn-danger">Ошибка</button>  
<button type="button" class="btn btn-link">Ссылка</button>
```

Библиотека построена на основе классов, добавляя которые к элементу вы задаёте ему стилевое оформление

Bootstrap – библиотека визуальных компонентов

Кнопка Размеры

Bootstrap обеспечивает четыре размера кнопки:



Классы, которые определяют различные размеры являются:

- `.btn-lg`
- `.btn-md`
- `.btn-sm`
- `.btn-xs`

Следующий пример показывает код для различных размеров кнопок:

Bootstrap – пример работы

Сделаем простые кнопки

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <!-- Required meta tags always come first -->
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
7     <meta http-equiv="x-ua-compatible" content="ie=edge">
8
9     <!-- Bootstrap CSS -->
10    <link rel="stylesheet" href="
11      https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.4/css/bootstrap.min.css">
12  </head>
13  <body>
14    <h1>Hello, world!</h1>
15
16    <button class="btn btn-success">Click me!</button>
17    <button class="btn btn-danger">Click and me!</button>
18
19    <!-- jQuery first, then Tether, then Bootstrap JS. -->
20    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.0.0/jquery.min.js"></script>
21    <script src="https://cdnjs.cloudflare.com/ajax/libs/tether/1.2.0/js/tether.min.js"></script>
22    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.4/js/bootstrap.min.js"></script>
23  </body>
24 </html>
```


Bootstrap – результат визуализации кнопок

Hello, world!

Click me!

Click and me!

```
<button class="btn btn-success">Click me!</button>  
<button class="btn btn-danger">Click and me!</button>
```

Добавив класс **btn** мы подключили к кнопке стилевое оформление, заложенное создателями Bootstrap. Задав класс **btn-success** – задали зеленый цвет кнопки, а классом **btn-danger** – красный цвет.

Bootstrap – набор компонентов

Text alignment

You can quickly change the text alignment of any card—in its entirety or specific parts—with our [text align](#) classes.

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

Special title treatment

With supporting text below as a natural lead-in to additional content.

[Go somewhere](#)

```
<div class="card card-block">  
  <h4 class="card-title">Special title treatment</h4>  
  <p class="card-text">With supporting text below as a natural lead-in to additional content.</p>  
  <a href="#" class="btn btn-primary">Go somewhere</a>  
</div>
```

[Copy](#)

Dapibus ac facilisis in

Cras sit amet nibh libero

Porta ac consectetur ac

Vestibulum at eros

```
<div class="list-group">  
  <a href="#" class="list-group-item list-group-item-action list-group-item-success">Dapibus ac fa  
  <a href="#" class="list-group-item list-group-item-action list-group-item-info">Cras sit amet nil  
  <a href="#" class="list-group-item list-group-item-action list-group-item-warning">Porta ac cons  
  <a href="#" class="list-group-item list-group-item-action list-group-item-danger">Vestibulum at  
</div>
```

[Copy](#)


В состав Bootstrap входит множество визуальных компонентов (из которых можно собрать аскетичный, но практичный интерфейс).

Bootstrap – набор компонентов


<https://getbootstrap.com/docs/5.0/examples/>

Framework

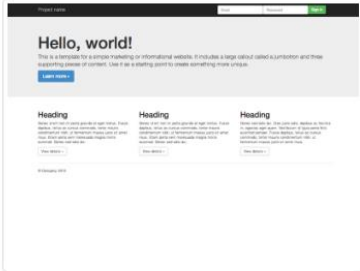
Examples that focus on implementing uses of built-in components provided by Bootstrap.



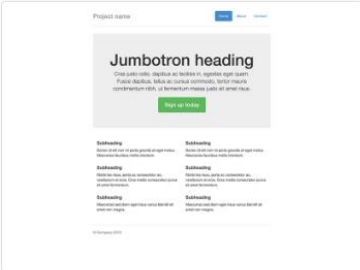
Starter template
Nothing but the basics: compiled CSS and JavaScript.



Grids
Multiple examples of grid layouts with all four tiers, nesting, and more.



Jumbotron
Build around the jumbotron with a navbar and some basic grid columns.



Narrow jumbotron
Build a more custom page by narrowing the default container and jumbotron.

А также готовые шаблоны, которые можно скачать и переработать под конкретную задачу.

Подключение Bootstrap

- 1 вариант: подключение через CDN
- 2 вариант: подключение скачанных CSS и JS файлов

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">

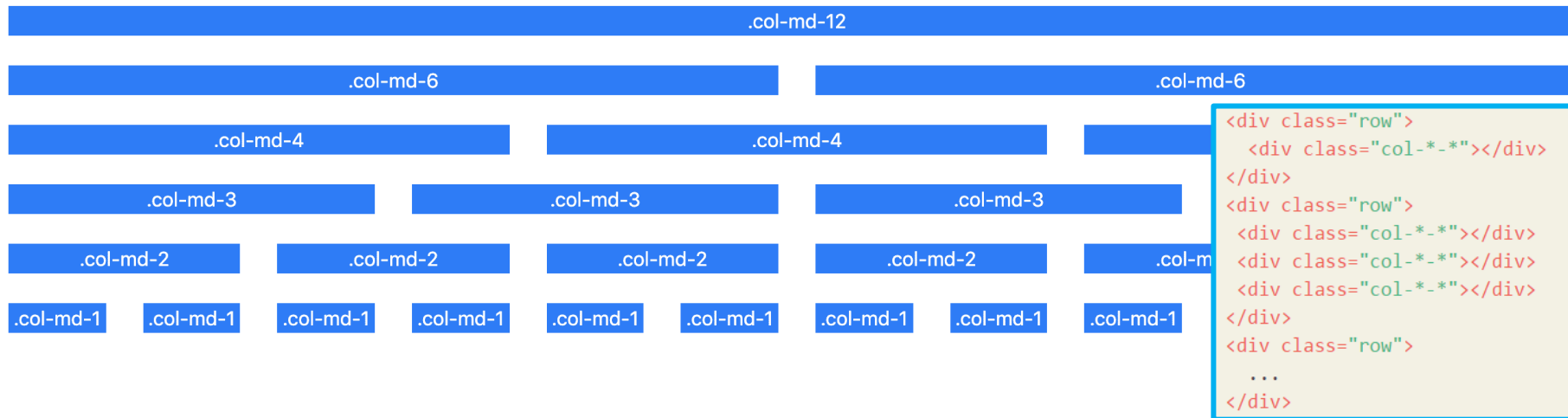
  <title>Hello, world!</title>
</head>
<body>
  <h1>Hello, world!</h1>

  <!-- Option 1: Bootstrap Bundle with Popper -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-b5kHyXgcpbZJ0/tY9U17kGkf1S0CWuKcCD38l8YkeH8z8QJE0GmW1gYU5S9F0nJ0"
  crossorigin="anonymous"></script>

  <!-- Option 2: Separate Popper and Bootstrap JS -->
  <!--
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.6.0/dist/umd/popper.min.js" integrity="sha384-
  xxx" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.min.js" integrity="sha384-
  xxx" crossorigin="anonymous"></script>
  -->
</body>
</html>
```

Основа Bootstrap – сетка

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											



Опора Bootstrap – «сетка» размещения элементов, позволяющая размещать элементы в несколько столбцов, размер (и количество) которых будет адаптироваться под размеры экрана.

Принятая в Bootstrap градация устройств

В таблице показано, как максимальная ширина **max-width** каждого контейнера **.container** и **.container-fluid** сравнивается с исходными в каждой точке останова (breakpoint).

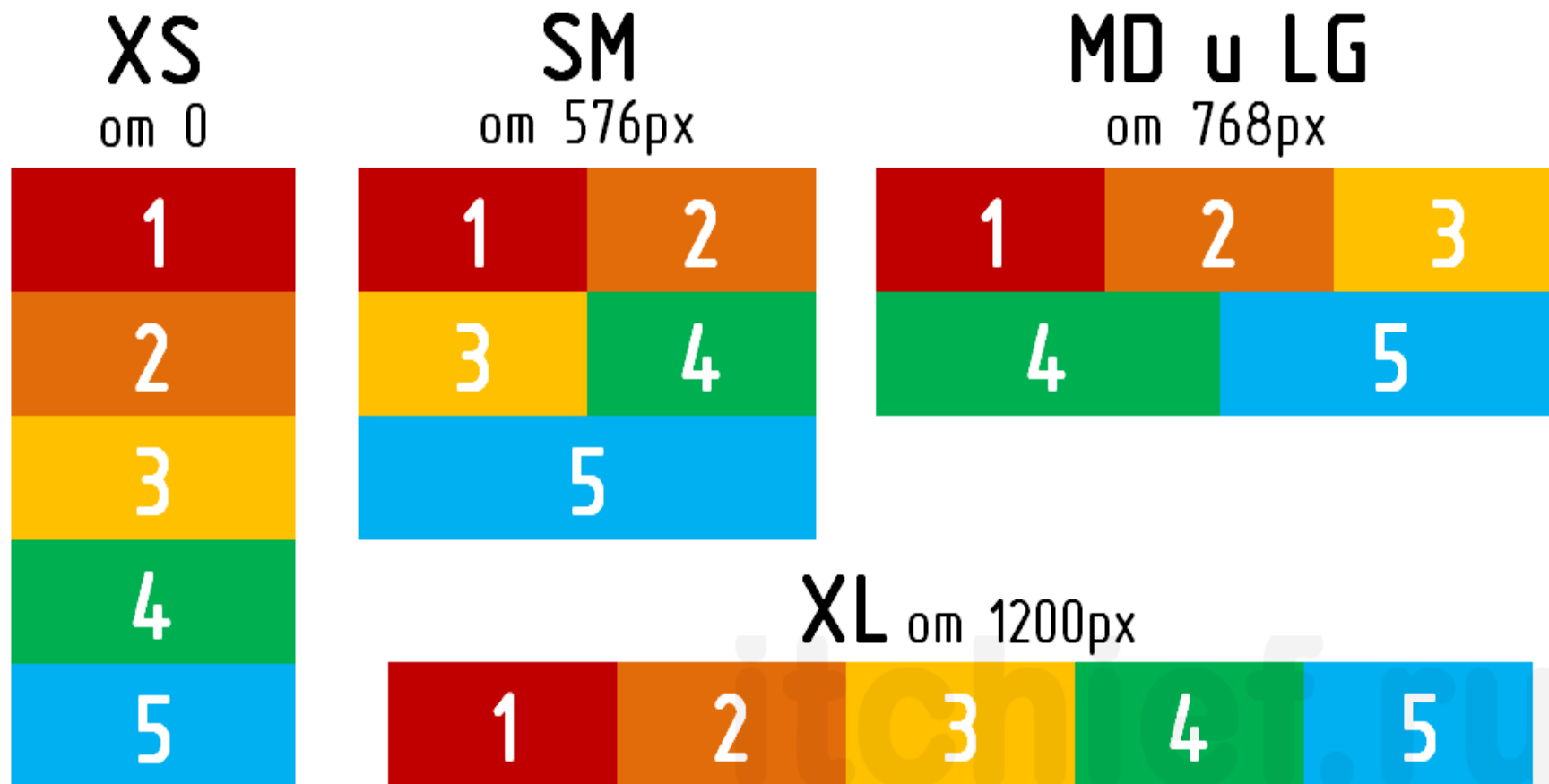
	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

Строки и столбцы в Bootstrap

```
<div class="container">
  <div class="row">
    <div class="col">
      1 из 2
    </div>
    <div class="col">
      2 из 2
    </div>
  </div>
  <div class="row">
    <div class="col">
      1 из 3
    </div>
    <div class="col">
      2 из 3
    </div>
    <div class="col">
      3 из 3
    </div>
  </div>
</div>
```

- Сначала задается контейнер.
- Далее задается горизонтальный ряд или строка. Строку всегда следует помещать в контейнер.
- В рядах могут быть расположены только колонки.
- Контент должен быть расположен в колонках
- Сетка Bootstrap состоит из 12 колонок.
- Цифры в наименовании классов колонок показывают, сколько колонок из 12-ти возможных нужно использовать.

Адаптивная верстка в Bootstrap



Bootstrap – размещение элементов на странице

```
<div class="container">

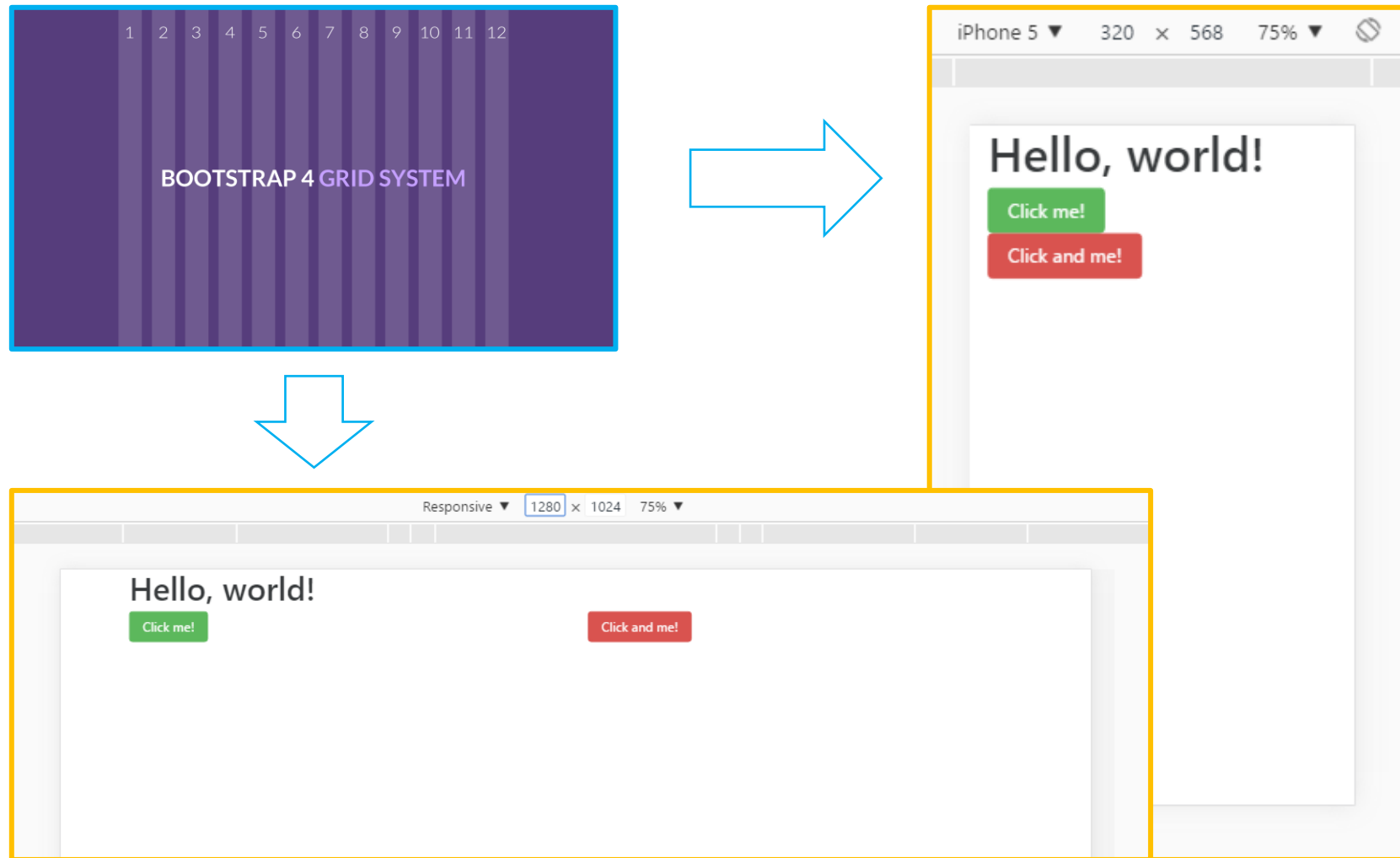
  <div class="row">
    <div class="col-md-12">
      <h1>Hello, world!</h1>
    </div>
  </div>

  <div class="row">
    <div class="col-md-6">
      <button class="btn btn-success">Click me!</button>
    </div>
    <div class="col-md-6">
      <button class="btn btn-danger">Click and me!</button>
    </div>
  </div>

</div>
```

Вставьте в стартовый шаблон bootstrap'а приведенный код.

Bootstrap – размещение элементов на странице в несколько колонок



*Страница в зависимости от размера экрана
элементы на странице перестраиваются*

Bootstrap Grid – сетка Bootstrap

Bootstrap разделяет все экраны на такие группы, разделяя их на

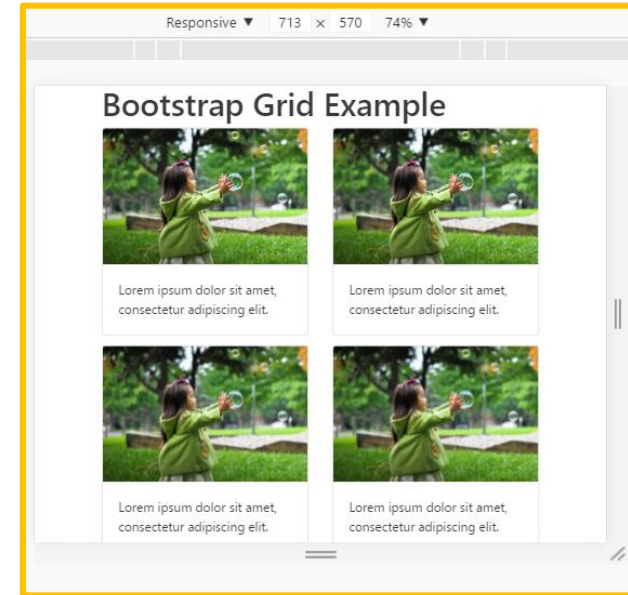
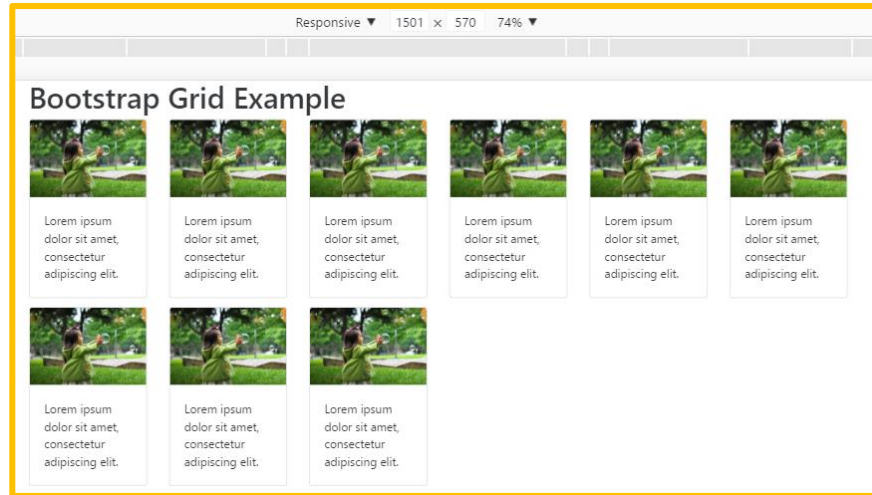
12 столбцов:

	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container <small>max-width</small>	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	<small>.col-</small>	<small>.col-sm-</small>	<small>.col-md-</small>	<small>.col-lg-</small>	<small>.col-xl-</small>	<small>.col-xxl-</small>
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

При помощи классов **col-md-6** (**col-sm-8** и т.п.), которые указываются для элементов, можно указать сколько именно столбцов сетки будет выделено под элемент.

Для того, чтобы сетка работала корректно все элементы должны располагаться в рамках тега с классом **row**, каждая такая строка содержит набор элементов которые могут расположиться или в один ряд, или перенестись на другие строки. Все «строки» (теги с классом **row**) должны располагаться в теге-«контейнере» который должен быть отмечен классом **container**.

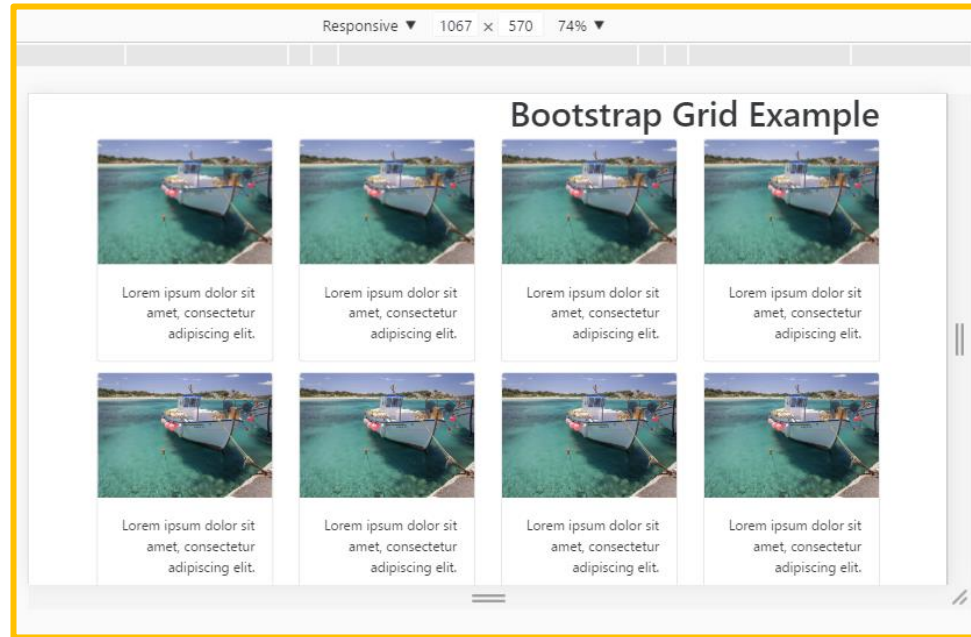
Работа сетки Bootstrap на примере



```
<div class="row ">  
  <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-xs-12">  
    <div class="card">  
        
      <div class="card-block">  
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>  
      </div>  
    </div>  
  </div>
```

При помощи классов `col-**-**` мы можем задавать сколько столбцов будет выделено под помеченный тег. Такими классами необходимо пометить каждый тег, который входит в строку (`row`)

Работа Bootstrap на примере



```
<div class="container">
  <div class="row ">
    <div class="col-xl-2 col-lg-3 col-md-4 col-sm-6 col-xs-12 text-sm-left text-md-center text-lg-right">
      <div class="card">
        
```

При помощи классов **text-**-left** (**-center**, **-right**, **-justify**) можно задавать выравнивание содержимого внутри блока (причём для различных разрешений выравнивание может быть разное). Эффект подобен CSS-свойству *text-align*.

Сетка Bootstrap

Сетка Bootstrap состоит из следующих элементов:

- Обёрточные контейнеры `container`, `container-fluid`, `container-{breakpoint}`;
- Горизонтальные ряды (строки) `row`;
- Вертикальные колонки или столбцы `col`.

```
<div class="container">
  <!-- Content here -->
</div>
```

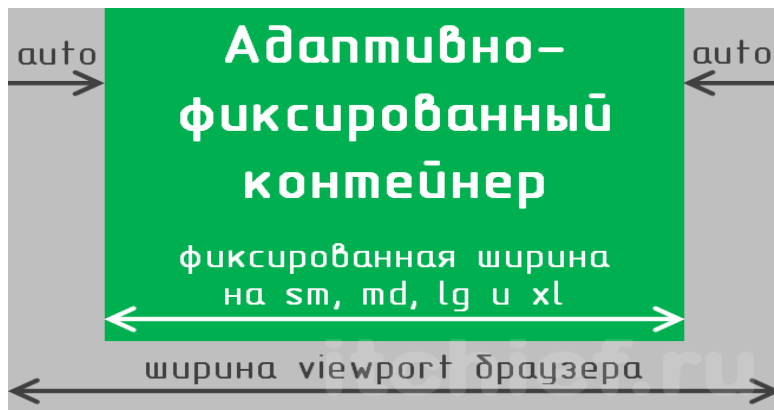
```
<div class="container">
  <div class="row">
    <div class="col-sm">
      Одна из трёх колонок
    </div>
    <div class="col-sm">
      Одна из трёх колонок
    </div>
    <div class="col-sm">
      Одна из трёх колонок
    </div>
  </div>
</div>
```

Адаптивные контейнеры Bootstrap

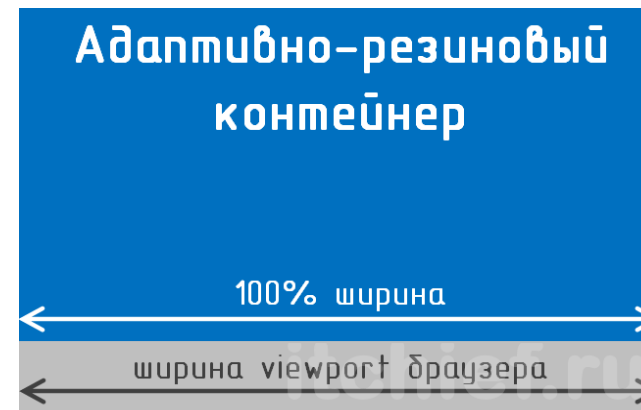
Контейнеры необходимы при использовании стандартной сетки.

Контейнеры используются для размещения в них содержимого и дополнений.

- **container** - устанавливает максимальную ширину **max-width** в каждой отзывчивой точке останова
- **container-fluid** – **width: 100%** во всех точках останова
- **container-{breakpoint}**, то есть ширина **width: 100%** до указанной точки останова



container



container-fluid

Элементы ввода и формы

Получение данных от пользователя.

Регистрация пользователя

Имя:

Email:

Отправить

<form> ... </form>

Элементы ввода

<input ... >

type="text"

*Для размещения элементов ввода на форме применяется тег **input** с различными значениями атрибута **type**.*

<input type="...">

Тип	Описание	Вид
button	Кнопка.	<input type="button" value="Кнопка"/>
checkbox	Флажки. Позволяют выбрать более одного варианта из предложенных.	<input type="checkbox"/> Пиво <input type="checkbox"/> Чай <input type="checkbox"/> Кофе
file	Поле для ввода имени файла, который пересылается на сервер.	<input type="button" value="Выбрати..."/> Файл не выбрано.
hidden	Скрытое поле. Оно никак не отображается на веб-странице.	
image	Поле с изображением. При нажатии на рисунок данные формы отправляются на сервер.	<input type="image" value="Отправить"/>
password	Обычное текстовое поле, но отличается от него тем, что все символы показываются звездочками. Предназначено для того, чтобы никто не подглядел вводимый пароль.	<input type="password"/>
radio	Переключатели. Используются, когда следует выбрать один вариант из нескольких предложенных.	<input type="radio"/> Пиво <input type="radio"/> Чай <input type="radio"/> Кофе
reset	Кнопка для возвращения данных формы в первоначальное значение.	<input type="button" value="Скинути"/>
submit	Кнопка для отправки данных формы на сервер.	<input type="button" value="Надіслати"/>
text	Текстовое поле. Предназначено для ввода символов с помощью клавиатуры.	<input type="text"/>

Типовой набор элементов ввода

Параметры в URL

```
/file.html?userName=Ivan&userEmail=ivan%40mail.com
```

Передача параметров файлу, через URL

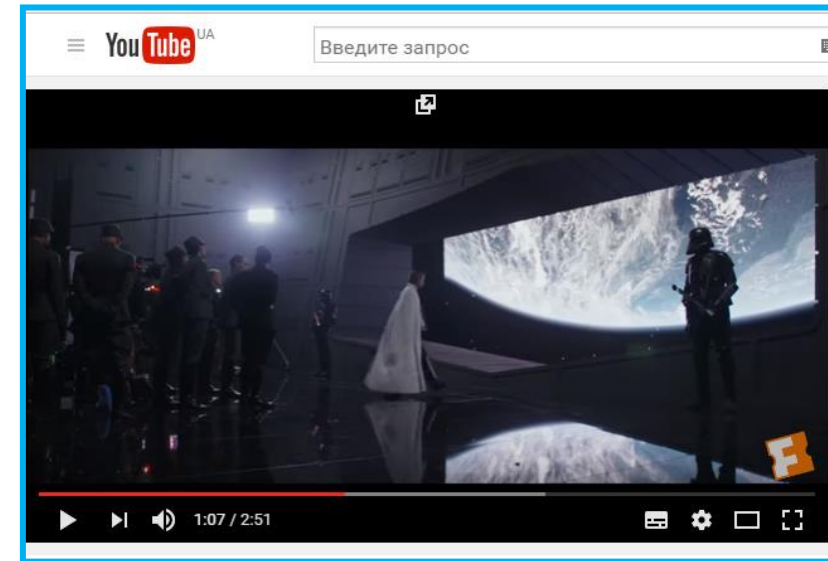
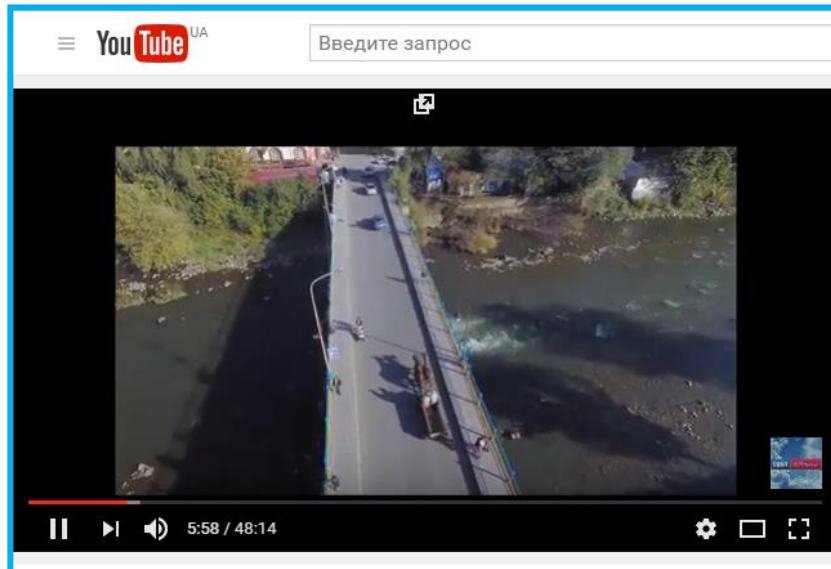
- ? – спецсимвол, говорящий, что в этом месте заканчивается адрес файла, и начинается перечень параметров.
- = – спецсимвол разделяющий имя параметра и его значение.
- & – спецсимвол разделяющий пары (имя параметра и его значение) друг от друга.

Параметры передаваемые из формы добавляются в URL адрес, эти параметры есть возможность обработать на стороне сервера

Параметры в URL

<https://www.youtube.com/watch?v=IMhJgaqPepo>

https://www.youtube.com/watch?v=4B6_y1s-Hco



Страница одна – параметры разные, как результат и содержимое разное.

Ручное оформление параметров ввода

```
6
7
8
9
10
11
12
13
14 <head>
```

```
<style>
  input {
    border: 2px solid green;
    border-radius: 15px 40px;
    background: lightblue;
    font-size: 20pt;
  }
</style>
```

1

Form Example

Имя:

Email:

Send data

Оформление элементов ввода в Bootstrap

```
14 <div class="container">
15
16 <h1> Bootstrap From Example</h1>
17
18 <form>
19   <div class="form-group">
20     <label for="user_name_id" class="form-control-label">User name: </label>
21     <input type="text" name="user_name" class="form-control" id="user_name_id" placeholder="Enter user name, ex: Ivan">
22     <small class="form-text text-muted">Fill required!</small>
23   </div>
24   <div class="form-group">
25     <label for="user_email_id" class="form-control-label">Email address</label>
26     <input type="email" name="user_email" class="form-control" id="user_email_id" placeholder="Enter email, ex: ivan@mail.com">
27     <small id="emailHelp" class="form-text text-muted">Fill required correct email!</small>
28   </div>
29   <button type="submit" class="btn btn-primary">Send data</button>
30 </form>
31
32 </div>
```

Bootstrap From Example

User name:

Fill required!

Email address


Fill required correct email!


Bootstrap содержит в себе набор стилей для оформления элементов ввода (пример на основе стартового шаблона bootstrap)

Оформление элементов ввода в Bootstrap

```
14 <div class="container">
15
16 <h1> Bootstrap From Example</h1>
17
18 <form>
19   <div class="form-group has-warning">
20     <label for="user_name_id" class="form-control-label form-control-warning">User name: </label>
21     <input type="text" name="user_name" class="form-control form-control-warning"
22       id="user_name_id" placeholder="Enter user name, ex: Ivan">
23     <small class="form-text text-muted">Fill required!</small>
24   </div>
25   <div class="form-group has-danger">
26     <label for="user_email_id" class="form-control-label">Email address</label>
27     <input type="email" name="user_email" class="form-control form-control-danger"
28       id="user_email_id" placeholder="Enter email, ex: ivan@mail.com">
29     <small id="emailHelp" class="form-text text-muted">Fill required correct email!</small>
30   </div>
31   <button type="submit" class="btn btn-primary">Send data</button>
32 </form>
33
34 </div>
```


Bootstrap From Example

User name: 



Fill required!

Email address



Fill required correct email!

Send data

Bootstrap содержит в себе стили классы для обозначения неверно заполненных полей.

HTML, CSS, Bootstrap. Как отладить и разобраться?

Chrome DevTools

The screenshot displays the Chrome DevTools interface with the Bootstrap v5.0 documentation open in the browser. The documentation page shows the `.form-select` class, which is used to style custom `<select>` menus. The page includes a visual example of the select menu and the corresponding HTML code.

HTML Code:

```
<select class="form-select" aria-label="Default select example">
  <option selected>Open this select menu</option>
  <option value="1">One</option>
  <option value="2">Two</option>
  <option value="3">Three</option>
</select>
```

CSS Code:

```
element.style {
}

*, ::after, ::before {
  box-sizing: border-box;
}

option {
  font-weight: normal;
  display: block;
  white-space: nowrap;
  min-height: 1.2em;
  padding: 0px 2px 1px;
}
```

The Chrome DevTools interface shows the **Elements** panel on the left, displaying the DOM tree with the `<select>` element selected. The **Styles** panel on the right shows the default styles for the `option` element, including `font-weight: normal`, `display: block`, `white-space: nowrap`, `min-height: 1.2em`, and `padding: 0px 2px 1px`.

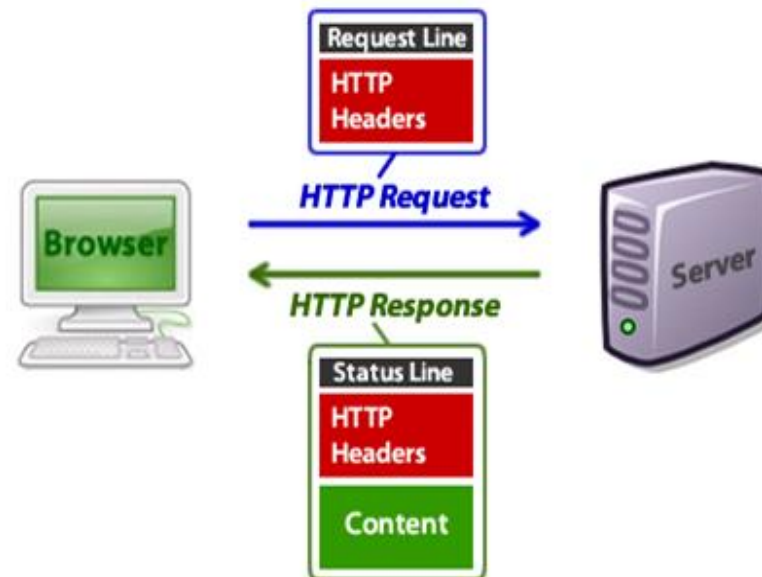
HTTP – протокол передачи гипертекста

HyperText Transfer Protocol — «протокол передачи гипертекста» — протокол прикладного уровня передачи данных (изначально — в виде гипертекстовых документов в формате HTML).

Текущая версия протокола HTTP/2.

В настоящий момент используется для передачи произвольных данных.

Основой HTTP является технология «клиент-сервер».



HTTP. Структура сообщения

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

- **Стартовая строка** (англ. Starting line) — определяет тип сообщения;
- **Заголовки** (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
- **Тело сообщения** (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

Заголовки и тело сообщения могут отсутствовать, но стартовая строка является обязательным элементом, так как указывает на тип запроса/ответа. Исключением является версия 0.9 протокола, у которой сообщение запроса содержит только стартовую строку, а сообщения ответа только тело сообщения.

Для версии протокола 1.1 сообщение запроса обязательно должно содержать заголовок Host.

```
generic-message =   start-line
                   *(message-header CRLF)
                   CRLF [ message-body ]
```

HTTP. Стартовая строка запроса

Метод URI HTTP/Версия

Метод (англ. Method) — название запроса, одно слово заглавными буквами

URI — путь к запрашиваемому документу

Версия (англ. Version) — версия протокола HTTP, например, 1.1

Примеры стартовых строк запроса:

POST / HTTP/1.1

GET /background.png HTTP/1.0

HEAD /test.html?query=alibaba HTTP/1.1

OPTIONS /anypage.html HTTP/1.0

HTTP. Стартовая строка ответа

HTTP/Версия Код-Состояния Пояснение

Версия — версия протокола HTTP (как и в запросе).

Код состояния (англ. Status Code) — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента.

Пояснение (англ. Reason Phrase) — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Примеры стартовых строк ответов:

HTTP/1.1 200 OK

HTTP/1.1 201 Created

HTTP/1.1 204 No Content

HTTP/1.1 304 Not Modified

HTTP/1.1 400 Bad Request

HTTP/1.1 404 Not Found

HTTP/1.1 503 Service Unavailable

HTTP. Пример запроса и ответа

GET / HTTP/1.1
Host: xbb.uz
User-Agent: Mozilla/5.0 ...
Accept: text/html,application ...
Accept-Language: ru-ru,r ...
Accept-Encoding: gzip,de ...
...



Запрос



Ответ

HTTP/1.0 200 OK
Server: nginx/0.7.67
Date: Tue, 08 Feb 2011 08: ...
Content-Type: text/html; c ...
X-Powered-By: PHP/5.2.12
Expires: Thu, 19 Nov 1981 ...
...

HTTP. Методы

GET

Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.

HEAD

HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.

POST

POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.

PUT

PUT заменяет все текущие представления ресурса данными запроса.

DELETE

DELETE удаляет указанный ресурс.

CONNECT

CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.

OPTIONS

OPTIONS используется для описания параметров соединения с ресурсом.

TRACE

TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.

PATCH

PATCH используется для частичного изменения ресурса.

Каждый сервер обязан поддерживать как минимум методы GET и HEAD. Кроме методов GET и HEAD, часто применяется метод POST.

HTTP. Коды ответов сервера

1xx Informational («Информационный»)

В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.

2xx Success («Успех»)

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

3xx Redirection («Перенаправление»)

Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4xx Client Error («Ошибка клиента»)

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

5xx Server Error («Ошибка сервера»)

Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

HTTP. Коды ответов сервера

1xx Informational («Информационный»)

В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-серверы подобные сообщения должны отправлять дальше от сервера к клиенту.

2xx Success («Успех»)

Сообщения данного класса информируют о случаях успешного принятия и обработки запроса клиента. В зависимости от статуса сервер может ещё передать заголовки и тело сообщения.

3xx Redirection («Перенаправление»)

Коды класса 3xx сообщают клиенту что для успешного выполнения операции необходимо сделать другой запрос (как правило по другому URI). Из данного класса пять кодов 301, 302, 303, 305 и 307 относятся непосредственно к перенаправлениям (редирект). Адрес, по которому клиенту следует произвести запрос, сервер указывает в заголовке Location. При этом допускается использование фрагментов в целевом URI.

4xx Client Error («Ошибка клиента»)

Класс кодов 4xx предназначен для указания ошибок со стороны клиента. При использовании всех методов, кроме HEAD, сервер должен вернуть в теле сообщения гипертекстовое пояснение для пользователя.

5xx Server Error («Ошибка сервера»)

Коды 5xx выделены под случаи неудачного выполнения операции по вине сервера. Для всех ситуаций, кроме использования метода HEAD, сервер должен включать в тело сообщения объяснение, которое клиент отобразит пользователю.

HTTP. Заголовки

HTTP Headers — это строки в HTTP-сообщении, содержащие разделенную двоеточием пару параметр-значение. Формат заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (RFC 822). Заголовки должны отделяться от тела сообщения хотя бы одной пустой строкой!

Все заголовки разделяются на четыре основных группы:

- **General Headers («Основные заголовки»)** — могут включаться в любое сообщение клиента и сервера.
- **Request Headers («Заголовки запроса»)** — используются только в запросах клиента.
- **Response Headers («Заголовки ответа»)** — только для ответов от сервера.
- **Entity Headers («Заголовки сущности»)** — сопровождают каждую сущность сообщения.

Заголовок	Описание	Пример
Accept	Список допустимых форматов ресурса	Accept: text/plain
Accept-Charset	Перечень поддерживаемых кодировок для предоставления пользователю	Accept-Charset: utf-8
Allow	Список поддерживаемых методов	Allow: OPTIONS, GET, HEAD
Referer	URI ресурса, после которого клиент сделал текущий запрос	Referer: http://en.wikipedia.org/wiki/Main_Page
User-Agent	Список названий и версий клиента и его компонентов с комментариями	User-Agent: Mozilla/5.0 (X11; Linux i686; rv:2.0.1) Gecko/20100101 Firefox/4.0.1

HTTP. Заголовки. Примеры

Пример заголовков из запроса:

Host: www.kns.ru

Connection: keep-alive

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36

Accept:

text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng, */*;q=0.8,application/signed-exchange;v=b3;q=0.9

Accept-Encoding: gzip, deflate, br

Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: _ym_uid=1500910643685716520; ***

Пример заголовков из ответа:

Server: nginx/1.19.0

Date: Fri, 12 Mar 2021 13:51:08 GMT

Content-Type: text/html; charset=utf-8

Content-Length: 30604

Connection: keep-alive

Cache-Control: private

Content-Encoding: gzip

X-Powered-By: ASP.NET

HTTP. Как отладить и разобраться?

The screenshot displays the Progress Telerik Fiddler Web Debugger interface. The main window is divided into two panes. The left pane shows a list of intercepted requests, and the right pane shows the details of the selected request.

Request List (Left Pane):

#	Server_ThinkTime	Result	Protocol	Host	URL
92	149,55	200	HTTPS	t.co	/adsc?type=java
93	164,08	200	HTTPS	analytics.twitter.com	/adsc?type=java
94	88,18	200	HTTPS	www.google.com	/complete/search?clie
95		200	HTTP	Tunnel to	www.google.com:443
96	50,98	200	HTTPS	www.google.com	/complete/search?clie
97		200	HTTP	Tunnel to	www.google.com:443
98	60,02	200	HTTPS	www.google.com	/complete/search?clie
99	66,61	200	HTTPS	www.google.com	/complete/search?clie
100	121,87	200	HTTPS	www.google.com	/complete/search?clie
101	49,54	200	HTTPS	www.google.com	/complete/search?clie
102		200	HTTP	Tunnel to	www.kns.ru:443
103		200	HTTP	Tunnel to	www.kns.ru:443
104		200	HTTP	Tunnel to	cdn.kns.ru:443
105	21,99	200	HTTP	tile-service.weather...	/ru-RU/livatile/preinst
106	22,99	301	HTTP	cdn.content.prod.c...	/singletile/summary/al
107		200	HTTP	Tunnel to	assets.msn.com:443
108	192,89	200	HTTPS	www.kns.ru	/
109		200	HTTP	Tunnel to	bat.bing.com:443
110	5,00	304	HTTPS	cdn.kns.ru	/lib/bootstrap/css/boc
111		200	HTTP	Tunnel to	cdn.kns.ru:443
112		200	HTTP	Tunnel to	cdn.kns.ru:443
113	145,92	200	HTTPS	www.kns.ru	/ajaxdata.aspx?utm_
114	7,99	304	HTTPS	cdn.kns.ru	/lib/lightslider/css/light
115		200	HTTP	Tunnel to	cdn.kns.ru:443
116		200	HTTP	Tunnel to	cdn.kns.ru:443
117		200	HTTP	Tunnel to	cdn.kns.ru:443
118		200	HTTP	Tunnel to	cdn.kns.ru:443
119		200	HTTP	Tunnel to	cdn.kns.ru:443
120	8,00	304	HTTPS	cdn.kns.ru	/lib/font/fonts/icomoo
121		200	HTTP	Tunnel to	cdn.kns.ru:443
122	5,99	200	HTTPS	cdn.kns.ru	/lib/stvle-all.min.css

Request Details (Right Pane):

GET https://www.kns.ru/ HTTP/1.1

Host: www.kns.ru

Connection: keep-alive

Sec-ch-ua: "Google Chrome";v="89", "Chromium";v="89", ";Not A Brand";v="99"

Sec-ch-ua-mobile: ?0

Upgrade-Insecure-Requests: 1

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.114 Safari/537.36

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/svg+xml,*/*;q=0.8

Sec-Fetch-Site: none

Sec-Fetch-Mode: navigate

Sec-Fetch-User: ?1

Sec-Fetch-Dest: document

Accept-Encoding: gzip, deflate, br

Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7

Cookie: _ym_uid=1500910643685716520; caltat=45fd457556e347b09c0f6dc4c5cf44d4; top100_i

HTTP/1.1 200 OK

Server: nginx/1.19.0

Date: Fri, 12 Mar 2021 14:11:29 GMT

Content-Type: text/html; charset=utf-8

Content-Length: 127734

Connection: keep-alive

Cache-Control: private

Vary: Accept-Encoding

X-AspNet-Version: 4.0.30319

X-Powered-By: ASP.NET

<!DOCTYPE html>

<html lang="ru">

<https://www.telerik.com/fiddler>

HTTPS и безопасность

HyperText Transfer Protocol Secure — расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS.

В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443 (для незащищённого HTTP — 80).

HTTPS обеспечивает защиту от атак, основанных на прослушивании сетевого соединения — от sniffерских атак и атак типа man-in-the-middle, при условии, что будут использоваться шифрующие средства, сертификат сервера проверен и ему доверяют.

Где взять сертификат:

- **VeriSign**;
- **GlobalSign**;
- **Let's Encrypt** - бесплатный, автоматизированный и открытый ЦС (Центр Сертификации).