

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет информационной безопасности

Кафедра инфокоммуникационных технологий

ВЕБ-ТЕХНОЛОГИИ В ИНФОКОММУНИКАЦИЯХ

Разработка структуры веб-страниц



Минск 2022

Содержание

Лабораторная работа №7 Разработка структуры веб-страниц.....	3
Что такое верстка сайта	3
Что включает в себя верстка сайта	3
3 основных вида верстки веб-сайтов	3
Какой должна быть верстка.....	7
Проверка верстки.....	7
jQuery	7
Правила работы с библиотекой jQuery.....	8
Методы jQuery.....	9
Манипуляции HTML-элементами.....	10
Чтение и изменение CSS-свойств, классов и атрибутов.....	14
Обход DOM и выборка html-элементов	17
jQuery анимация.....	20
Методы объекта window	27
События jQuery	27
Селекторы jQuery	29
Правила веб-тиографики.....	32
Шрифт Awesome.....	35
Favicon сайта	37
Сервисы подбора цвета для веб-дизайна	38
Адаптивная верстка сайта	39
Загрузка и подключение слайдера	49
Figma для верстальщика	53
Анализ ключевых слов	54
Яндекс.Вордстат	54
Задание к лабораторной работе №7	57

Лабораторная работа №7

Разработка структуры веб-страниц

Что такое верстка сайта

Чтобы макет сделать рабочим инструментом, а не просто изображением, необходимо написать код, который будут распознавать браузеры. Этот код отображается визуально так же, как выглядит макет сайта.

Дизайн и верстка сайта – всегда находятся вместе. Перед дизайнером стоит задача – разработать стильный макет. Задача верстальщика – воплотить планы дизайнера в реальность и технически, сделать сайт рабочим и удобным для пользователя.

Что значит верстать? Создавать структуру всех элементов на сайте согласно приготовленному заранее макету. Правильная верстка сайта включает множество нюансов, необходимо подобрать инструменты, проверять валидность верстки и хорошо разбираться в коде. Малейшие ошибки приводят к нестабильной работе веб-сайта, или вообще к полному прекращению функционирования.

Что включает в себя верстка сайта

Верстка сайтов – это целый комплекс процессов, вот лишь некоторые из них:

- вырезание из макета изображений, иконок и прочей графики, компоновка и сбор в отдельную папку;
- сбор шрифтов для корректного отображения;
- разработка страниц сайта, опираясь на дизайн-макет;
- верстка страниц в HTML и CSS;
- подключение JS-библиотек, создание динамики элементов;
- тестирование и проверка валидности верстки.

3 основных вида верстки веб-сайтов

Существует довольно много видов верстки сайтов, но специалисты выделяют 3 основных вида: табличная, блочная и адаптивная.

Адаптивная верстка

Подразумевает корректное отображение на любых видах устройств. На сегодняшний день она наиболее популярная, вряд ли найдется человек, который захочет, чтобы его сайт хорошо отображался только на компьютерах или только на смартфонах.



Тем не менее, адаптивная верстка – неотъемлемая часть двух других видов. Поэтому ее гораздо реже выделяют, как отдельный вид.

Табличная верстка

Безвозвратно устаревший метод. Его используют, но крайне редко. Сейчас такую верстку можно встретить разве что в HTML-письмах в email-рассылках или на очень старых сайтах. Он подразумевает собой огромную таблицу, в ячейках которых размещаются другие таблицы и в каждой таблице указывается какая-то информация. Код такой верстки очень сложный для восприятия, его тяжело понимать, править – еще сложнее.

Гибкая верстка

Это современный универсальный вид верстки сайтов, который активно используется для создания качественных веб-страниц. Принцип его заключается в том, что все элементы располагаются в блоках, или контейнерах. Они содержат необходимую информацию и сами по себе являются регулируемыми. Можно задавать их размер, цвет и прочие параметры.

Гибкая верстка дизайн макета проще, имеет больше возможностей и способна реализовать любые идеи дизайнера. Код получается чище и легче, он удобно читается браузерами, что влияет на ранжирование сайта в поисковых системах. Кроме того, блоки гарантируют адаптивный дизайн, поэтому сайт корректно отображается на различных устройствах.

Можно создавать гибкие компоненты, которые будут по-разному подстраиваться под пространство на экране без медиа-запросов, используя встроенную гибкость методов разметки. Например, можно отображать медиа объект в виде колонки, когда пространство ограничено, и как строка, когда пространства достаточно. Сделать это можно парой строк CSS.

`flex-basis` для флекс элементов – 250 пикселей. Если для двух элементов размером 250 пикселей места не хватает, они отображаются друг под другом. Элементы могут расти (`flex-grow` – положительное значение), поэтому они будут расти и заполнять строки.

```
.media {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.media > * {  
  flex: 1 1 250px;  
}
```



The first ascent, made in June from the Paris Observatory, though a lofty one, was attended with so much danger and confusion as to be barren of results. The departure, owing to stormy weather, was hurried and illordered, so that the velocity in rising was excessive, the net constricted the rapidly-swelling globe, and the volumes of out-rushing gas half-suffocated the voyagers. Then a large rent occurred, which caused an alarmingly rapid fall, and the two philosophers were reduced to the necessity of flinging away all they possessed, their instruments only excepted.



The first ascent, made in June from the Paris Observatory, though a lofty one, was attended with so much danger and confusion as to be barren of results. The departure, owing to stormy weather, was hurried and illordered, so that the velocity in rising was excessive, the net constricted the rapidly-swelling globe, and the volumes of out-rushing gas half-suffocated the voyagers. Then a large rent occurred, which caused an alarmingly rapid fall, and the two philosophers were reduced to the necessity of flinging away all they possessed, their instruments only excepted.

Замечательное свойство Flexbox заключается в том, что компонент ведет себя так в случае, если пространство ограничено размером экрана или компонент помещен в контекст, где у него меньше пространства в контейнере. Медиа-запросы не могут решить эту проблему, так как они смотрят на функции

всего экрана. Таким образом, новый макет дает нам то, чего не могут медиа-запросы.

В примере ниже компонент ограничен видимой частью окна браузера (измените размер окна, чтобы оценить гибкость) и контейнером.

[Смотреть пример.](#)

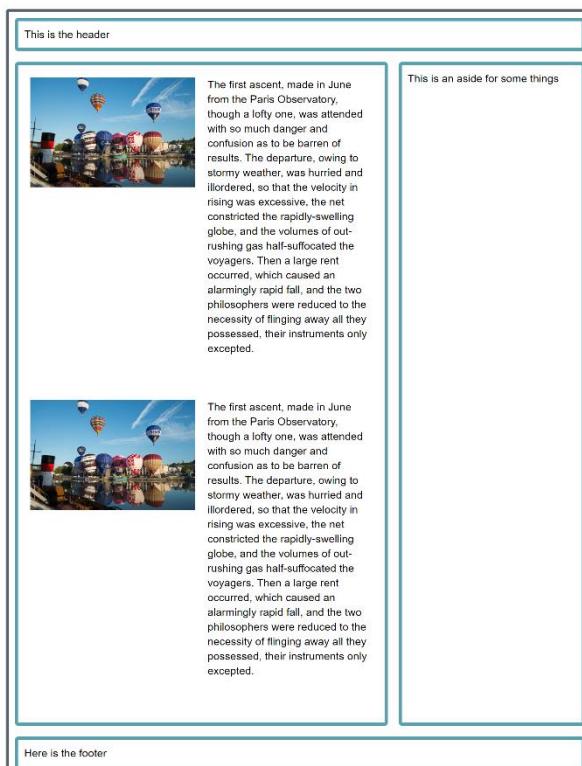
Если необходимо что-то большее, то придется подключать медиа-запросы. Медиа-запросы превосходно работают с Grid – можно полностью переопределить сетку одной строкой CSS или изменить расположение элементов. Для этого не понадобится править разметку. Сначала создам сетку в одну колонку для узких экранов.

```
.grid {  
  display: grid;  
  grid-gap: 1em;  
  grid-template-columns: 1fr;  
}
```

Далее с помощью медиа-запросов переопределим количество колонок, а также зададим элементы, которые растягиваются на несколько колонок в сетке на больших экранах.

```
@media (min-width: 40em) {  
  .grid {  
    grid-template-columns: 2fr 1fr;  
  }  
  
  header,  
  footer {  
    grid-column: 1 / 3;  
  }  
}
```

Тщательно подобранные медиа-запросы и наследственная адаптивность этих новых методов макетирования дает возможность использовать все пространство экрана и предоставлять лучше UX пользователям на любых устройствах. В следующем примере представлено использование Grid совместно с медиа-запросами.



Заметьте, как макет трансформируется с помощью медиа-запроса по мере уменьшения пространства на экране. Когда область контента становится

слишком узкой, компонент схлопывается до одной колонки – без отсылки к медиа-запросу.

[Смотреть пример.](#)

Преимущество Grid Layout по отношению к Flexbox заключается в возможности изменить порядок элементов в макете на разных переломных/контрольные точках. Для пользователей мобильных устройств и десктоп пользователей с клавиатурой и мышью это обеспечит отличный UX.

Как правило, используются медиа-запросы под ширину устройства, проверяя, достаточно ли места для отображения колонок с контентом. Медиа-запросы можно также использовать для проверки доступной высоты.

Один из примеров использования вертикальных медиа-запросов или медиа-запросов по высоте – проверка того, что высоты экрана достаточно для отображения многоколоночного макета без необходимости скролить окно вниз. В CSS коде ниже многоколоночный макет создаст колонки только, если места хватает под 2 колонки размером 15em. Поэтому здесь не нужен медиа-запрос по ширине. Добавим медиа-запрос `min-height`, чтобы текст перепрыгивал в колонки только при достижении разумной высоты. Если экран маленький и в альбомном режиме, будет показан только один столбец, и пользователь будет скролить вниз для чтения.

```
@media (min-height: 500px) {  
    section {  
        column-width: 15em;  
    }  
}
```

Использование медиа-запросов

Медиа-запросы позволяют задать стили для определенного размера экрана, типа устройства на основе характеристик устройства и т.д. Все медиа-запросы начинаются с `@media`, а далее следует условие.

С помощью медиа-запросов можно задать стили для следующих типов устройств:

- `all` – все типы (значение используется по умолчанию);
- `braille` – устройства, основанные на системе Брайля, которые предназначены для чтения слепыми людьми;
- `embossed` – принтеры, использующие для печати систему Брайля;
- `handheld` – смартфоны и аналогичные им аппараты;
- `print` – принтеры и другие печатающие устройства;
- `projection` – проекторы;
- `screen` – экран монитора;
- `speech` – речевые синтезаторы, а также программы для воспроизведения текста вслух;
- `tty` – устройства с фиксированным размером символов;
- `tv` – телевизоры.

Пример задания ширины для тега `div`, на экранах мониторов, смартфонах и т.п.

```
@media all and (max-width: 1200px) {  
    div {  
        width: 960px;  
    }  
  
@media all and (max-width: 480px) {  
    div {  
        width: 320px;  
    }  
}
```

Какой должна быть верстка

Верстка сайта подчиняется строгим законам. В противном случае код будет неправильным, а работоспособность сайта снижена.

Признаки правильной верстки:

- верстка обязательно должна быть кроссбраузерной и корректно отображаться в разных браузерах, независимо от разрешения экрана;
- блочная верстка считается наиболее качественной, при этом она должна быть адаптивна ко всем устройствам;
- валидная верстка сайта – в коде не должно быть ошибок;
- стили выносятся в отдельный файл, код страницы максимально короткий и легкий;
- содержимое HTML и CSS прописано только строчными буквами;
- CSS имеет преимущества перед JS, все, что можно сделать при помощи стилей, делается только через них;
- правильно структурированы заголовки, абзацы, продуманы их стили;
- код понятен, имеет четкую структуру, видны открывающие и закрывающие теги;
- все изображения имеют прописанные размеры;
- сохранены размеры первоначального макета.

При соблюдении этих правил сайт будет работать правильно, быстро, а другие специалисты при необходимости смогут разобраться в коде.

Проверка верстки

После того, как верстка дизайна сайта окончена, необходимо запустить проверку и убедиться, что все элементы правильно сделаны. Базовое тестирование подразумевает использование нескольких сценариев, где сайт проверяется на различных браузерах и устройствах.

Для более точной оценки работоспособности используются различные инструменты и сервисы. Некоторые сервисы позволяют сравнивать готовую верстку с первоначальным макетом сайта на соответствие.

В ходе проверки верстки изучается код на наличие незакрытых тегов. Проверяется правильность семантической разметки для поисковых систем, кроссбраузерность, а также функциональность страницы, даже если отключены изображения и коды JavaScript.

jQuery

jQuery – библиотека JavaScript, содержащая в себе готовые функции языка JavaScript, все операции jQuery выполняются из кода JavaScript.

Библиотека **jQuery** производит манипуляции с html-элементами, управляя их поведением и используя DOM для изменения структуры веб-страницы. При этом исходные файлы HTML и CSS не меняются, изменения вносятся лишь в отображение страницы для пользователя.

Для выбора элементов используются селекторы CSS. Выбор осуществляется с помощью функции `$()`. При вызове функция `$()` возвращает новый экземпляр объекта JQuery, который оборачивает ноль или более элементов DOM и позволяет взаимодействовать с ними различными способами.

Выполнение различных сценариев возможно только после окончания загрузки структуры документа `document`, когда браузер преобразует html-код страницы в дерево DOM. Управление процессом загрузки обеспечивает конструкция

```
jQuery(document).ready(function() {  
});
```

Сначала производится обертывание экземпляра `document` в функцию `jQuery()`, после применяется метод `ready()`, которому передается функция `function() {...}`, исполняемая после загрузки документа.

На практике обычно используется сокращенная форма такой записи `jQuery(function() {...});`, или `$(function() {...});`.

Для хранения информации при работе с библиотекой `jQuery` используются переменные `JavaScript`. В переменных могут храниться элементы. Имена переменных, предназначенных для хранения возвращаемых элементов, начинаются со знака `$`, например:

```
$h = $(".list").parent().parent().detach();
```

Для хранения нескольких элементов используются массивы `JavaScript`:

```
$k[3] = 15;
```

Правила работы с библиотекой `jQuery`

Как добавить `jQuery` на веб-страницу

Добавить библиотеку `jQuery` на свою веб-страницу можно двумя способами:

Использовать версию файла `jQuery`, размещенную на ресурсах `Google`, `Microsoft` или `jQuery.com`.

Данный метод использует «Сеть дистрибуции контента» (CDN, `content distribution network`), т.е. файл `jQuery` расположен на другом веб-сайте, который при запросе пользователем отправляет данный файл на его компьютер. Очевидные преимущества данного способа – снижение нагрузки на собственный веб-сервер и ускорение загрузки файла в силу разветвленности сети серверов дистрибутора.

Чтобы воспользоваться данным способом, необходимо перейти по одной из ссылок:

[Google CDN](#)

[Microsoft CDN](#)

[CDNJS CDN](#)

После перехода на сайт ресурса потребуется всего лишь скопировать ссылку на `jQuery`-файл и добавить ее на свою веб-страницу между тегами `<script>...</script>`. В результате должно получиться, например,

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
```

Загрузить последнюю версию библиотеки `jQuery` можно с официального сайта. Для этого необходимо перейти по адресу [jQuery.com](#) и выбрать интересующую версию библиотеки. Для загрузки предлагается две версии `jQuery`-файла – минимизированный и несжатый. Размер несжатого файла около 300 Кб, он содержит комментарии, поэтому его лучше использовать с целью разработки и отладки кода.

Минимизированная версия файла весит около 100Кб, в ней удалены все комментарии и ненужные пробелы, что ускоряет загрузку файла браузером.

Для загрузки нужно перейти по ссылке, и в открывшемся окне щелкнуть правой кнопкой мыши и выбрать «Сохранить как ...». После этого поместить файл

в нужную папку (обычно для этого используется папка «scripts») и добавить его на страницу:

```
<script src="scripts/jquery-3.6.0.min.js"></script>
```

Правила добавления jQuery на страницу

Размещать ссылки на jQuery-файл нужно внутри тега `<head>`.

Размещать ссылки на jQuery-файл нужно после ссылок на стили CSS, так как зачастую библиотека jQuery производит манипуляции со стилями элементов веб-страницы.

Размещать другие теги `<script>...</script>` нужно только после ссылки на файл jQuery, если данный скрипт использует библиотеку jQuery.

Как создать новый html-элемент

Создать пустой html-элемент, например, блок, можно несколькими способами:

1. С помощью краткой формы записи `$("<div>")`.
2. С помощью инструкции `$("<div></div>")`.
3. С помощью инструкции `$("<div/>")`.

Все три способа рабочие, но, тем не менее, рекомендуется включать открывающие и закрывающие теги, чтобы показать, что данный элемент может содержать другие элементы.

При создании нового элемента методу `$(())` можно передать второй параметр в виде объекта JavaScript, определяющий дополнительные атрибуты элемента:

```
$("<img/>",  
{src: "images/flower.jpg",  
title: "Rose_in_garden",  
click: function() {...}  
}).appendTo("body");
```

Методы jQuery

Методы jQuery позволяют манипулировать содержимым веб-страницы. Методы присваивают заданные действия отобранным в jQuery-объект элементам, в результате чего происходит динамическое изменение элементов и их содержимого.

Каждый метод либо сам что-то возвращает, либо получает параметр и выполняет, что указано в параметре.

В общем виде синтаксис для вызова метода jQuery имеет следующий вид: `$(“селектор”).имяМетода(параметры);`.

Динамическое изменение элементов веб-страниц

Библиотека jQuery упрощает процесс отбора элементов HTML-страниц. С помощью методов jQuery производятся манипуляции с объектной моделью документа DOM. Чтобы отобрать группу элементов, нужно передать селектор функции jQuery. В качестве селектора элемента может выступать сам элемент, его идентификатор или класс, а также комбинация селекторов:

```
$("a")  
$("#some-id")  
$(".someclass")  
$("header > ul:has(a)")
```

Функция `$(())` возвращает объект jQuery, содержащий массив элементов DOM – так называемый обернутый набор, соответствующих указанному селектору. Большинство методов возвращает по завершении действий первоначальный набор элементов.

Манипуляции HTML-элементами

Манипуляции HTML-элементами позволяют уточнять, расширять обернутый набор элементов путем добавления нового содержимого или удалять элементы из набора. Чтобы манипулировать элементами, их нужно отобрать с помощью селекторов или методов отбора.

Добавление содержимого на страницу

- Метод `.html()`

Возвращает HTML-содержимое первого элемента обернутого набора или добавляет HTML-содержимое в каждый элемент набора.

`.html()` – метод указывается без параметров. Возвращает содержимое первого элемента в соответствующем наборе в виде HTML-разметки.

`.html(фрагмент разметки)` – `фрагмент разметки` – добавляет фрагмент HTML-разметки к содержимому всех элементов соответствующего набора.

`.html(функция)` – `функция` – принимает два аргумента: индекс элемента и текущее содержимое элемента. Возвращаемое значение добавляется в качестве нового содержимого.

- Метод `.text()`

Возвращает объединенное текстовое содержимое всех элементов обернутого набора, включая их потомков или добавляет новое текстовое содержимое.

`.text()` – метод указывается без параметров. Объединяет путем конкатенации текстовое содержимое всех обернутых элементов и возвращает полученный текст в качестве результата, который можно записать в переменную.

`.text(строка)` – `строка` – устанавливает содержимое параметра как новое текстовое содержимое всех обернутых элементов, при этом старое содержимое удаляется. Если строка содержит угловые скобки, они замещаются эквивалентными HTML-элементами.

`.text(функция)` – `функция` – вызывается для каждого элемента обернутого набора. Функция принимает два аргумента – индекс элемента и текущее содержимое элемента. Возвращаемое значение будет добавлено в качестве нового содержимого.

Добавление элементов

- Метод `.append()`

Добавляет содержимое, определенное параметром, в конец каждого элемента обернутого набора, после его содержимого. Новый набор будет содержать первоначальное содержимое и добавленное.

`.append(содержимое1, содержимое2)` – `содержимое1` – в качестве содержимого добавляется HTML-элемент, массив, строка или объект jQuery. `содержимое2` – необязательный параметр, определяет дополнительное содержимое, добавляется один или несколько HTML-элементов, массивов, строк или объектов jQuery.

`.append(функция)` – `функция` – вызывается для каждого элемента набора, функции передаются два аргумента – индекс элемента и текущее содержимое элемента. Функция возвращает строку, DOM элемент или объект jQuery. Возвращаемое значение будет использовано в качестве содержимого, дополняющего имеющееся содержимое элемента.

- Метод .appendTo()

Метод является аналогом .append() с разницей в том, что функции jQuery передается добавляемый элемент, а не обернутый набор. Добавляет элементы обернутого набора после содержимого элементов, заданных параметром.

.appendTo(элемент, к которому добавляется содержимое) – элемент, к которому добавляется содержимое – к элементу добавляется селектор, HTML-элемент, массив элементов, строка или объект jQuery.

- Метод .prepend()

Работает аналогично с .append(), только новое содержимое добавляется в начало, сразу после открывающегося тега элемента, перед содержимым элемента.

.prepend(содержимое1, содержимое2) – содержимое1 – в качестве содержимого добавляется HTML-элемент, массив, строка или объект jQuery. содержимое2 – необязательный параметр, определяет дополнительное содержимое, добавляется один или несколько HTML-элементов, массивов, строк или объектов jQuery.

.prepend(функция) – функция – вызывается для каждого элемента набора, функции передаются два аргумента – индекс элемента и текущее содержимое элемента. Функция возвращает строку, DOM элемент или объект jQuery. Возвращаемое значение будет использовано в качестве содержимого, дополняющего имеющееся содержимое элемента.

- Метод .prependTo()

Метод является аналогом .prepend() с разницей в том, что функции jQuery передается добавляемый элемент, а не обернутый набор. Добавляет элементы обернутого набора в начало содержимого элементов, заданных параметром.

.prependTo(элемент, к которому добавляется содержимое) – элемент, к которому добавляется содержимое – к элементу добавляется селектор, HTML-элемент, массив элементов, строка или объект jQuery.

- Метод .before()

Добавляет HTML-фрагмент или элемент, указанные в параметре метода, в дерево DOM, перед каждым элементом обернутого набора. Функция вызывается для каждого элемента набора, ей передается сам элемент и два аргумента – индекс элемента и текущее содержимое элемента. Возвращаемое значение будет использовано в качестве содержимого, дополняющего имеющееся содержимое элемента.

.before(содержимое1, содержимое2) – содержимое1 – в качестве содержимого добавляется HTML-элемент, массив, строка или объект jQuery. содержимое2 – необязательный параметр, определяет дополнительное содержимое, добавляется один или несколько HTML-элементов, массивов, строк или объектов jQuery.

.before(функция) – функция – в качестве аргумента функции передается индекс элемента в наборе и его текущее значение. Функция возвращает строку, DOM элемент или объект jQuery. Возвращаемое значение будет использовано в качестве добавляемого содержимого.

- Метод .insertBefore()

Вставляет содержимое, переданное функции jQuery, перед каждым элементом, указанным в качестве аргумента данного метода.

`.insertBefore(элемент, к которому добавляется содержимое)` – элемент, к которому добавляется содержимое – к элементу добавляется селектор, HTML-элемент, массив элементов, строка или объект jQuery.

- Метод `.after()`

Добавляет содержимое, указанное в параметре метода, в дерево DOM, после каждого элемента обернутого набора.

`.after(содержимое1, содержимое2)` – `содержимое1` – в качестве содержимого добавляется HTML-элемент, массив, строка или объект jQuery. `содержимое2` – необязательный параметр, определяет дополнительное содержимое, добавляется один или несколько HTML-элементов, массивов, строк или объектов jQuery.

`.after(функция)` – функция – в качестве аргумента функции передается индекс элемента в наборе и его текущее содержимое. Функция возвращает строку, DOM элемент или объект jQuery. Возвращаемое значение будет использовано в качестве добавляемого содержимого.

- Метод `.insertAfter()`

Вставляет содержимое, переданное функции jQuery, после каждого элемента, указанного в качестве параметра данного метода.

`.insertAfter(элемент, к которому добавляется содержимое)` – элемент, к которому добавляется содержимое – к элементу добавляется селектор, HTML-элемент, массив элементов, строка или объект jQuery.

- Метод `.wrap()`

Метод обертывает элемент или группу элементов HTML-разметкой. Если в наборе содержится несколько элементов, будет обернут каждый из них.

`.wrap(элемент-обертка)` – элемент-обертка – строка, представляющая фрагмент HTML-разметки, селектор, элемент с заданным классом или функция jQuery, которыми будет обернут набор элементов.

`.wrap(функция обратного вызова)` – функция обратного вызова – вызывается один раз для каждого элемента набора. В качестве аргумента получает индекс элемента в наборе. Внутри функции this ссылается на текущий элемент в наборе. Возвращает элемент DOM, объект JQuery, или HTML-фрагмент, в который будет обернут соответствующий элемент.

- Метод `.wrapAll()`

Оборачивает все элементы набора как единое целое в указанную HTML-разметку.

`.wrapAll(элемент-обертка)` – элемент-обертка – строка, представляющая фрагмент HTML-разметки, селектор, элемент с заданным классом или функция jQuery, которыми будет обернут набор элементов.

`.wrapAll(функция обратного вызова)` – функция обратного вызова – вызывается один раз для каждого элемента набора. В качестве аргумента получает индекс элемента в наборе. Внутри функции this ссылается на текущий элемент в наборе. Возвращает элемент DOM, объект JQuery, или HTML-фрагмент, в который будет обернут соответствующий элемент.

- Метод `.wrapInner()`

Оборачивает содержимое элементов набора, включая текстовое содержимое, в указанную HTML-разметку.

`.wrapInner(элемент-обертка)` – элемент-обертка – строка, представляющая фрагмент HTML-разметки, селектор, элемент с заданным классом или функция jQuery, которыми будет обернуто содержимое набора элементов.

`.wrapInner(функция обратного вызова)` – функция обратного вызова – вызывается один раз для каждого элемента набора. В качестве аргумента получает индекс элемента в наборе. Внутри функции this ссылается на текущий элемент в наборе. Возвращает элемент DOM, объект JQuery, или HTML-фрагмент, в который будет обернуто содержимое соответствующего элемента.

Замена и удаление элементов

- Метод `.unwrap()`

Удаляет элемент, оберывающий набор.

`.unwrap()` – метод вызывается без параметров.

- Метод `.detach()`

Удаляет все элементы обернутого набора из DOM. Сохраняет нетронутыми данные и события, связанные с элементами. Все дочерние элементы также исключаются из DOM. Исключенные элементы и связанные с ними данные позднее можно вернуть в дерево DOM.

`.detach(селектор)` – селектор – необязательный параметр, уточняет, какие именно элементы подлежат удалению.

- Метод `.remove()`

Полностью удаляет все элементы обернутого набора из DOM. При этом можно удалить несколько элементов с одинаковым классом. Все дочерние элементы также исключаются из DOM. Одновременно с элементами удаляются все данные и события, связанные с ними.

`.remove(селектор)` – селектор – необязательный параметр, уточняет, какие именно элементы подлежат удалению.

- Метод `.empty()`

Удаляет содержимое всех элементов набора, удаляя все содержимое, а также все дочерние элементы, находящиеся внутри него.

`.empty()` – методу не передаются аргументы.

- Метод `.replaceWith()`

Заменяет каждый элемент в обернутом наборе содержимым, переданным методу в качестве аргумента.

`.replaceWith(новое содержимое)` – новое содержимое – определяет вставляемое содержимое. Может быть фрагмент HTML-разметки с содержимым или без него, селектор, массив элементов или объект JQuery.

`.replaceWith(функция)` – функция – возвращаемое значение заменит каждый элемент обернутого набора.

- Метод `.replaceAll()`

Заменяет каждый элемент, соответствующий переданному методу селектору, обернутым набором элементов. Возвращает обернутый набор, содержащий заменившие элементы. Замещенные элементы удаляются и не могут участвовать в последующих операциях.

`.replaceAll(целевой элемент)` – целевой элемент – селектор, объект JQuery, HTML-элемент или массив элементов, которые нужно заменить.

- Метод `.clone()`

Создает копии элементов в обернутом наборе. Элементы копируются вместе со вложенными элементами. Полученные копии элементов можно далее присоединить в другое место DOM для дальнейшей обработки.

`.clone(логическое значение)` – `логическое значение` – указывает, должны ли вместе с элементами копироваться их обработчики событий. Значение по умолчанию `false`. Если установлено `true`, то вместе с элементами будут скопированы обработчики событий.

Чтение и изменение CSS-свойств, классов и атрибутов

Библиотека jQuery позволяет управлять свойствами и атрибутами элементов обернутого набора, изменения первоначальные значения. Можно устанавливать новые свойства, получать и изменять значения первоначальных свойств. Удаляя или добавляя классы, можно динамически изменять стиль отображения элементов.

Добавление и удаление класса

- Метод `.addClass()`

Добавляет указанный класс (или несколько классов) к каждому элементу обернутого набора. Чтобы данный метод сработал, необходимо заранее создать стиль для добавляемого класса. Метод не удаляет старый класс, а просто добавляет новый.

`.addClass(имя класса)` – `имя класса` – одно или больше имен класса, разделенных друг от друга пробелами.

`.addClass(функция)` – `функция` – возвращает одно или более имен класса, разделенных пробелом, которые будут добавлены к существующим. В качестве аргумента принимает индекс элемента в наборе и существующее имя класса(ов).

- Метод `.removeClass()`

Удаляет указанное имя класса(ов) у всех элементов обернутого набора.

`.removeClass(имя класса)` – `имя класса` – необязательный параметр, одно или более имен класса, разделенных пробелом. Если имя класса не указано, метод удаляет все существующие классы у элементов набора. Если имя класса задано – удаляет только указанный класс.

`.removeClass(функция)` – `функция` – возвращает одно или более имен класса, разделенных пробелом, которые будут удалены из существующих. В качестве аргумента принимает индекс элемента в наборе и старое имя класса(ов).

- Метод `.toggleClass()`

Добавляет или удаляет один или более классов из каждого элемента в наборе. Каждый элемент обернутого набора проверяется отдельно. Метод добавляет указанное имя класса, если оно отсутствует в элементе, и удаляет у тех элементов, где оно присутствует. Используется для переключения визуального представления элементов.

`.toggleClass(имя класса)` – `имя класса` – одно или более имен класса, разделенных пробелами, которые будут переключаться для каждого элемента набора.

`.toggleClass(имя класса, логическое значение)` – `имя класса` – одно или более имен класса, разделенных пробелами, которые будут переключаться для каждого элемента набора. `логическое значение` – устанавливает, добавить или удалить указанный класс. Значение `true` добавляет класс, `false` – удаляет.

`.toggleClass(логическое значение)` – логическое значение – необязательный параметр, устанавливает, будут ли переключаться классы каждого элемента набора.

`.toggleClass(функция, логическое значение)` – функция – возвращает имя класса, которое будет переключаться для каждого элемента набора. В качестве аргументов получает индекс элемента в наборе и старое значение класса. логическое значение – необязательный параметр, устанавливает, будут ли переключаться классы каждого элемента набора.

- Метод `.hasClass()`

Проверяет наличие указанного имени класса хотя бы у одного элемента в соответствующем наборе. Возвращает `true`, если хотя бы один из элементов в наборе имеет проверяемое имя класса, в противном случае – `false`.

`.hasClass(имя класса)` – имя класса – строка с именем класса для поиска.

Изменение атрибутов элементов

Метод получает значение атрибута первого элемента набора или устанавливает одно или более значений атрибутов для элементов набора.

- Метод `.attr()`

`.attr(имя атрибута)` – имя атрибута – возвращает значение атрибута первого элемента в обернутом наборе. Если атрибут отсутствует, возвращает `undefined`.

`.attr(имя атрибута, значение)` – имя атрибута – задает имя атрибута, для которого будет установлено указанное значение. значение – строка или число, которое будет добавлено как значение атрибута для всех элементов обернутого набора.

`.attr(атрибуты)` – атрибуты – значения, которые копируются из свойств объекта, будут установлены для всех элементов обернутого набора.

`.attr(имя атрибута, функция)` – имя атрибута – задает имя атрибута, для которого будет установлено указанное значение. функция – в качестве аргументов принимает индекс элемента в наборе и старое значение атрибута. Возвращаемое значение будет установлено в качестве значения атрибута.

- Метод `.removeAttr()`

Удаляет указанный атрибут у каждого элемента обернутого набора.

`.removeAttr(имя атрибута)` – имя атрибута – строка, определяющая атрибут для удаления.

Изменение свойств элемента

- Метод `.css()`

Возвращает вычисляемое значение свойства стиля для первого элемента в обернутом наборе или устанавливает одно или несколько CSS-свойств для каждого элемента набора.

`.css(имя свойства)` – имя свойства – строка с именем свойства, возвращает его вычисляемое значение для первого элемента набора.

`.css(имена свойств)` – имена свойств – массив свойств, возвращает их вычисляемые значения для первого элемента набора.

`.css(имя свойства, значение)` – имя свойства – строка с именем свойства. значение – строка или число, которые будут установлены в качестве значения указанного свойства для всех элементов обернутого набора.

`.css(имя свойства, функция)` – имя свойства – строка с именем свойства. функция – в качестве аргументов функции передается индекс элемента в наборе

и старое значение свойства. Возвращаемое значение будет установлено для всех элементов набора.

`.css(объект свойств)` – `объект свойств` – добавляет CSS-свойства, имена которых определены как ключи в переданном объекте, в связанные с ними значения для всех элементов в соответствующем наборе.

Получение и изменение размеров и координат элемента

- Метод `.width()`

Возвращает текущее значение ширины для первого элемента в наборе или устанавливает ширину для каждого элемента набора. Единица измерения по умолчанию `px`. Метод можно использовать в случае, если полученное значение будет использоваться в математических расчетах. Размеры вычисляются без учета отступов и толщины рамки, без указания единицы измерения. При изменении размеров окна браузера размеры элемента могут изменяться.

`.width()` – метод вызывается без параметров. Возвращает текущее значение ширины для первого элемента в наборе без указания единицы измерения.

`.width(значение)` – `значение` – целое числовое значение или строка-значение ширины, которое будет установлено для каждого элемента набора.

`.width(функция)` – `функция` – принимает в качестве аргумента индекс элемента и старое значение свойства, возвращаемое значение будет установлено как ширина для всех элементов.

- Метод `.height()`

Возвращает текущее значение высоты для первого элемента в наборе или устанавливает высоту для каждого элемента набора.

`.height()` – метод вызывается без параметров. Возвращает текущее значение высоты для первого элемента в наборе.

`.height(значение)` – `значение` – целое числовое значение или строка-значение высоты, которое будет установлено для каждого элемента набора.

`.height(функция)` – `функция` – принимает в качестве аргумента индекс элемента и старое значение свойства, возвращаемое значение будет установлено как высота для всех элементов.

- Метод `.innerWidth()`

Возвращает ширину первого элемента в обернутом наборе с учетом отступов `padding` или устанавливает ее для каждого элемента обернутого набора.

`.innerWidth()` – метод вызывается без параметров. Возвращает текущее значение внутренней ширины для первого элемента в наборе.

`.innerWidth(значение)` – `значение` – целое числовое значение, которое будет установлено для каждого элемента набора.

`.innerWidth(функция)` – `функция` – принимает в качестве аргумента индекс элемента и старое значение свойства, возвращаемое значение будет установлено как внутренняя ширина для всех элементов набора.

- Метод `.innerHeight()`

Возвращает высоту первого элемента в обернутом наборе с учетом отступов `padding`.

`.innerHeight()` – метод вызывается без параметров. Возвращает текущее значение внутренней высоты для первого элемента в наборе.

`.innerHeight(значение)` – `значение` – целое числовое значение, которое будет установлено для каждого элемента набора.

`.innerHeight(функция)` – функция – принимает в качестве аргумента индекс элемента и старое значение свойства, возвращаемое значение будет установлено как внутренняя ширина для всех элементов набора.

- Метод `.outerWidth()`

Возвращают ширину первого элемента в обернутом наборе. В эти размеры входят толщина рамки и ширина отступа.

`.outerWidth(логическое значение)` – логическое значение – необязательное значение, если установлено `true`, значение `margin` учитывается, в противном случае нет.

- Метод `.outerHeight()`

Возвращают высоту первого элемента в обернутом наборе. В эти размеры входят толщина рамки и ширина отступа.

`.outerHeight(логическое значение)` – логическое значение – необязательное значение, если установлено `true`, значение `margin` учитывается, в противном случае нет.

- Метод `.offset()`

Получает текущие координаты первого элемента или устанавливает координаты для каждого элемента. Возвращает объект JavaScript со свойствами `left` и `top`, содержащими координаты первого элемента в px обернутого набора относительно начала документа. Метод применяется только к видимым элементам.

`.offset()` – метод вызывается без параметров.

- Метод `.position()`

Возвращает объект JavaScript со свойствами `left` и `top`, содержащими координаты первого элемента в px обернутого набора относительно ближайшего родительского элемента. Метод применяется только к видимым элементам.

`.position()` – метод вызывается без параметров.

Обход DOM и выборка html-элементов

Выборка элементов

`.parent()`

Возвращает родительские элементы всех элементов первоначального обернутого набора, расположенные на один уровень выше.

Параметры: селектор.

`.parents()`

Возвращает новый набор, содержащий родительские элементы всех элементов первоначального обернутого набора. В их число входят прямые предки, а также все остальные родительские элементы, за исключением корневого элемента документа.

Параметры: селектор.

`.children()`

Возвращает новый набор, содержащий уникальные дочерние элементы (находящиеся на первом уровне вложенности) обернутых элементов.

Параметры: селектор.

`.closest()`

Возвращает обернутый набор, содержащий единственный элемент ближайшего предка, соответствующий указанному селектору. Переходит по цепочке родительских элементов вверх и останавливается при первом совпадении с селектором, указанным в круглых скобках.

Параметры: селектор.

`.prev()`

Переходит к следующему элементу с тем же родителем, находящемся слева. Возвращает обернутый набор, состоящий из уникальных соседних элементов, предшествующих элементам первоначального обернутого набора.

Параметры: селектор.

`.prevAll()`

Возвращает обернутый набор, содержащий все соседние элементы, предшествующие элементам первоначального обернутого набора.

Параметры: селектор.

`.prevUntil()`

Возвращает обернутый набор, содержащий все соседние элементы, предшествующие элементам первоначального обернутого набора, за исключением элемента, соответствующего селектору.

Параметры: селектор.

`.next()`

Переходит к следующему элементу с тем же родителем, находящемся справа. Возвращает обернутый набор, состоящий из уникальных соседних элементов, следующих за элементами первоначального обернутого набора.

Параметры: селектор.

`.nextAll()`

Возвращает обернутый набор, содержащий все соседние элементы, следующие за элементами первоначального обернутого набора.

Параметры: селектор.

`.nextUntil()`

Возвращает обернутый набор, содержащий все соседние элементы, следующие за элементами первоначального обернутого набора, за исключением элемента, соответствующего селектору.

Параметры: селектор.

`.siblings()`

Обходит все элементы одного уровня с выбранным элементом. Возвращает обернутый набор, содержащий уникальные соседние элементы, находящиеся на одном уровне вложенности с элементами первоначального обернутого набора.

Параметры: селектор.

`.is()`

Проверяет обернутый набор на наличие, по крайней мере, одного элемента, соответствующего заданному селектору. Возвращает `true`, если заданному селектору соответствует хотя бы один элемент, в противном случае `false`.

Параметры: селектор.

`.find()`

Возвращает новый обернутый набор, содержащий все элементы, дочерние по отношению к элементам из первоначального набора, соответствующие заданному селектору.

Параметры: селектор.

`.size()`

Возвращает количество элементов в обернутом наборе.

Параметры: нет

`.get()`

Возвращает один элемент в соответствии с заданным индексом или массив элементов, если параметр не указан. Отрицательный индекс ведет отсчет с конца обернутого набора элементов.

Параметры: число, задающее индекс возвращаемого элемента.

`.toArray()`

Возвращает массив элементов, содержащихся в обернутом наборе.

Параметры: нет.

`.index()`

Возвращает порядковый номер заданного элемента обернутого набора. Если указанный элемент отсутствует, возвращается значение -1. Если параметр не задан, возвращает индекс первого элемента среди элементов одного уровня вложенности.

Параметры: элемент или селектор, определяющие или идентифицирующие ссылку на элемент.

`.add()`

Возвращает новый набор, содержащий копию элементов первоначального набора, в который добавлены элементы, определенные параметром.

Параметры: элемент, селектор, массив элементов, фрагмент html-разметки.

`.each()`

Позволяет перебрать все элементы выборки страницы и произвести серию действий над каждым из них. Вызываемой функции передаются два параметра – индекс элемента набора и сам элемент, который устанавливается в качестве контекста функции `this`.

Параметры: функция.

`.map()`

Вызывает функцию для каждого элемента в обернутом наборе и возвращает полученные значения в виде массива JavaScript. Функции передаются два параметра – индекс элемента внутри набора и сам элемент.

Параметры: функция.

`.end()`

Используется внутри цепочки методов jQuery, возвращает предыдущий обернутый набор, чтобы применить к нему последующие операции.

Параметры: нет

`.andSelf()`

Объединяет два самых верхних обернутых набора в единый набор.

Параметры: нет.

Фильтрующие методы

.first()

Возвращает из обернутого элемента только первый элемент.

Параметры: нет.

.eq()

Возвращает из обернутого набора только один элемент. Отрицательное значение индекса ведет отсчет с конца набора.

Параметры: число, соответствующее индексу возвращаемого элемента.

.last()

Возвращает из обернутого элемента только последний элемент.

Параметры: нет.

.slice()

Возвращает набор элементов, содержащий элемент с начальным индексом и элементы до конечного индекса. Если конечный индекс отсутствует, то элементы от начального и до конца набора. Отрицательный индекс ведет отсчет элементов с конца набора, -1 вернет последний элемент.

Параметры: начальный индекс, конечный индекс (не обязательно).

.filter()

Удаляет из обернутого набора элементы, не соответствующие правилу отбора. Возвращает копию первоначального обернутого набора, из которого удалены указанные элементы. Если параметром является функция, она будет вызвана для каждого элемента в наборе, чтобы проверить, соответствует ли элемент заданному критерию отбора. Если для элемента будет возвращено значение `false`, то он будет удален из набора.

Параметры: элемент, селектор, массив или функция.

.not()

Удаляет из выбранного множества все элементы, соответствующие заданному выражению селектора. Возвращает копию первоначального обернутого набора, из которого удалены указанные элементы. Если параметром является функция, она будет вызвана для каждого элемента в наборе, чтобы проверить, соответствует ли элемент заданному критерию отбора. Если для элемента будет возвращено значение `false`, то он будет удален из набора.

Параметры: элемент, селектор, массив или функция.

.has()

Возвращает новый обернутый набор, содержащий элементы из первоначального набора, имеющие вложенные элементы, соответствующие заданному параметру.

Параметры: элемент или селектор.

jQuery анимация

Библиотека jQuery содержит несколько кросс-браузерных методов для анимации элементов, например, скольжение и плавное исчезновение, без привлечения дополнительных библиотек или плагинов. Для расширения возможностей работы с анимацией можно использовать библиотеку jQuery UI

(<https://jqueryui.com>), которая содержит набор интерфейсных взаимодействий, эффекты, виджеты и темы.

CSS-стили придают элементам страницы визуальные свойства, которые описывают их внешний вид. jQuery анимация представляет собой интерактивный процесс изменения свойств html-элементов от одного значения к другому.

Создание собственных эффектов с помощью метода .animate()

Эффекты, которых нет в библиотеке jQuery, можно создавать с помощью метода `.animate()`. Интерпретатор браузера динамически, без перезагрузки страницы, изменяет выбранные свойства на указанные значения. Анимация происходит для всех элементов обернутого набора. Чтобы добавить эффекты для конкретного элемента, нужно воспользоваться фильтрами jQuery для отбора.

Метод позволяет анимировать любое css-свойство, имеющее числовое значение, например, `font-size`, `opacity`, `border-width`, `margin`, `padding`, `height`, `width`, `background-position` и т.д. При этом имена свойств должны быть указаны слитно – `fontSize`, `paddingLeft`, или должен использоваться css-эквивалент свойства – "font-size". Числовые значения свойств не заключаются в кавычки.

Для любого свойства предварительно должно быть установлено начальное значение, а в css-объявлении должна использоваться полная запись каждого свойства, т.е., вместо свойства `border` должно быть заданы значения для `border-style`, `border-width` и т.д.

Функция обратного вызова вызывается один раз после завершения анимации. Функции не передается никаких аргументов, но анимации выполняется для элемента, переданного свойству `this` в качестве контекста.

Значениями свойств могут также выступать `hide`, `show` или `toggle`, в результате чего к элементу применится вычисляемое значение – отображение, скрытие или переключение исходных состояний свойств.

Метод `.animate()` позволяет изменять css-свойства выбранных элементов с возможностью одновременной анимации нескольких свойств, задавая продолжительность анимации в миллисекундах.

```
$("div").animate({left: "200px", top: "200px"}, 500);
```

Данная анимация одновременно применяется к свойству `left`, для которого задано значение `200px`, и к свойству `top` со значением `200px`, продолжительность анимации задана `500ms`.

Для элемента можно задавать относительное перемещение при каждом вызове анимации с помощью операторов `+=`, `-=`, `*=`, `/=`, например,

```
$("div").animate({left: "+=200", top: "-=200"}, 500);
```

Метод `.animate()` имеет две формы записи. В первой методу передаются четыре аргумента:

`.animate({свойство1: «значение1», свойство2: «значение2»}, продолжительность, функция перехода, function() {...});` – `{свойство1: «значение1»}` – объект свойств, которые собираемся анимировать в формате свойство: «значение», перечисленные в фигурных скобках через запятую. `продолжительность` – необязательный параметр продолжительности анимации, задающий время в миллисекундах или с помощью ключевых слов `slow`, `fast`, `normal`. `функция перехода` – необязательное имя функции перехода. `function() {...}` – необязательная функция обратного вызова.

Вторая форма принимает два аргумента: объект свойств и объект дополнительных функциональных возможностей.

`.animate({свойство1: «значение1», свойство2: значение2}, {duration: «значение», easing: «значение», specialEasing: {свойство1: «easing1», свойство2: «easing2»}, complete: function() {...}, queue: true, step: callback});` – {свойство1: «значение1»} – объект свойств, которые собираемся анимировать в формате свойство: «значение», перечисленные в фигурных скобках через запятую. duration – число или ключевое слово, значение по умолчанию 400. Устанавливает продолжительность анимации. easing – имя функции перехода. Значение по умолчанию swing – колебательный переход, второе доступное значение – linear – линейный переход. Расширенные возможности реализуются с помощью плагинов, можно посмотреть [jQuery UI](#). queue – помещает анимацию в очередь эффектов, по умолчанию true, значение false – немедленно воспроизводит анимацию. specialEasing – объект, содержащий одно и более свойств и их значений, описывающих функции перехода. step – функция, которая будет вызываться по окончании каждого этапа анимации и будет применяться для каждого элемента обернутого набора. Функции передается порядковый номер этапа и внутренний объект, описывающий эффект. progress – функция, которая будет вызываться после каждого шага анимации, только один раз за анимацию независимо от количества анимационных свойств. complete – функция, которая должна быть вызвана по окончании воспроизведения анимации. start – функция, которая будет вызвана в начале анимации. done – функция, вызванная по завершении анимации. fail – функция, вызываемая в случае невозможности завершения анимации. always – функция, которая будет вызвана в случае остановки или неполного завершения анимации.

Анимационные эффекты jQuery

- Метод `.fadeIn()`

Управляет прозрачностью, показывая скрытый элемент, при этом свойство opacity выбранного элемента изменяется от 0 до 1. Для этого на странице появляется необходимое пространство для элемента, при этом остальные элементы могут сдвинуться с места.

`.fadeIn(длительность, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта с помощью ключевых слов "fast", "normal", "slow" или числовых значений. По умолчанию используется значение "normal", равное 400 миллисекундам. функция по завершении анимации – необязательный параметр, задает функцию, которая будет вызвана после проявления элемента.

`.fadeIn(объект свойств)` – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

`.fadeIn(длительность, функция перехода, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. Значение по умолчанию swing – колебательный переход, второе доступное значение – linear – линейный переход. функция по завершении анимации – необязательный параметр, задает функцию, которая будет вызвана после проявления элемента.

- Метод `.fadeOut()`

Заставляет элемент исчезнуть, сделав его прозрачным, сохраняя на странице место, занимаемое элементом. Свойство opacity выбранного элемента изменяется от 1 до 0.

`.fadeOut(длительность, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта с помощью ключевых слов "fast", "normal", "slow" или числовых значений. Значение по умолчанию – 400 миллисекунд. функция по завершении анимации –

необязательный параметр, задает функцию, которая будет вызвана после проявления элемента.

.fadeOut(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

.fadeOut(длительность, функция перехода, функция по завершении анимации) – длительность – необязательный параметр, задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, задает функцию, которая будет вызвана после проявления элемента.

- Метод .fadeTo()

Позволяет изменить степень прозрачности до заданного значения, например, \$("element").fadeTo("normal",.50);

.fadeTo(длительность, прозрачность, функция по завершении анимации) – длительность – задает скорость проявления эффекта. прозрачность – число от 0 до 1, задающеее прозрачность элемента. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.fadeTo(длительность, прозрачность, функция перехода, функция по завершении анимации) – длительность – задает скорость проявления эффекта. прозрачность – число от 0 до 1, задающеее прозрачность элемента. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

- Метод .fadeToggle()

Если элемент скрыт, то он появляется на экране, если виден, то исчезает.

.fadeToggle(длительность, функция перехода, функция по завершении анимации) – длительность – задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.fadeToggle(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

- Метод .hide()

Скрывает видимый элемент. При установлении скорости элемент исчезает, как бы сужаясь. Чтобы замедлить действие, нужно передать значение продолжительности.

.hide() – метод указывается без параметров.

.hide(длительность, функция по завершении анимации) – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.hide(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

.hide(длительность, функция перехода, функция по завершении анимации) – длительность – задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

- Метод .show()

Показывает ранее скрытый элемент. Если не задано значение скорости, то элемент появляется моментально, если скорость задана, то элемент появляется от верхнего левого к нижнему левому углу.

.show() – метод указывается без параметров.

.show(длительность, функция по завершении анимации) – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.show(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

.show(длительность, функция перехода, функция по завершении анимации) – длительность – задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

- Метод .toggle()

Одиночный метод, переключает выбранный элемент из одного состояния в другое, в зависимости от его текущего состояния, скрывая или отображая его.

.toggle(длительность, функция по завершении анимации) – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.toggle(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

.toggle(длительность, функция перехода, функция по завершении анимации) – длительность – задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.toggle(display) – display – если параметр display установлен, то элемент будет показан.

- Метод .slideDown()

Заставляет скрытый элемент появиться на веб-странице. Элемент проявляется постепенно – сначала его верхняя часть, и по мере проявления остальной части то, что находилось под элементом, сдвигается вниз. Поэтому для того, чтобы контент не перемещался по странице, можно использовать абсолютное позиционирование элемента {position:absolute;}. Если необходимо разместить данный элемент относительно другого, то задайте относительное позиционирование {position:relative;} для элемента, который окружает абсолютно позиционированный элемент.

.slideDown(длительность, функция по завершении анимации) – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

.slideDown(объект свойств) – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

.slideDown(длительность, функция перехода, функция по завершении анимации) – длительность – необязательный параметр, задает скорость

проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

- Метод `.slideUp()`

Изменяет свойство `height` элемента, пока она не станет равной 0, после скрывает элемент `display: none;`. При этом удаление начинается снизу, и если для элемента не задано позиционирование, то контент, находившийся ниже, перемещается вверх.

`.slideUp(длительность, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

`.slideUp(объект свойств)` – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

`.slideUp(длительность, функция перехода, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

- Метод `.slideToggle()`

Скрывает видимый элемент и показывает скрытый, т.е. может использоваться в качестве переключателя, позволяющего как отображать, так и скрывать элемент. При этом элемент проявляется постепенно, сверху вниз.

`.slideToggle(длительность, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

`.slideToggle(объект свойств)` – объект свойств – представляет собой пары дополнительных опций, указанных в формате свойство: «значение».

`.slideToggle(длительность, функция перехода, функция по завершении анимации)` – длительность – необязательный параметр, задает скорость проявления эффекта. функция перехода – необязательный параметр, задает функцию перехода. функция по завершении анимации – необязательный параметр, описывает функцию, которая будет вызвана после проявления элемента.

Управление очередью анимации

Анимация элемента реализуется только с помощью методов, создающих анимационные эффекты. Если анимация реализована цепочкой методов, каждый эффект выполняется последовательно, друг за другом. В примере сначала начнется скольжение блока вниз, а выцветание станет в очередь "fx" и будет вызвано только при завершении скольжения.

```
$( "div" ).slideDown().fadeOut();
```

Чтобы добавить пользовательские эффекты, создаются функции, которые также добавляются в очередь "fx". Очередь представляет массив функций, существующих на уровне элемента и хранящихся в `jQuery.data`. Каждый элемент может иметь одну или несколько очередей функций, но обычно используется только одна очередь по умолчанию "fx".

Функции включаются в очередь с помощью метода `.queue()`, без него функции выполнятся не будут. Каждая функция по своему завершению должна вызвать метод `.dequeue()`, чтобы передать управление следующей функции в очереди.

```
$( "div" ).slideUp();
$( "div" ).queue(function() {
  document.write("Hello");
  $(this).dequeue();
});();
```

- Метод `.queue()`

Метод показывает текущую очередь анимации или манипулирует очередью функций, создающих анимационные эффекты, для элемента/элементов набора.

`.queue(имяОчереди)` – `имяОчереди` – строка, содержащая имя очереди. По умолчанию текущая очередь имеет зарезервированное имя `"fx"`. Возвращает массив функций, находящихся в очереди.

`.queue(имяОчереди, функция)` – `имяОчереди` – строка, содержащая имя очереди. `функция` – функция обратного вызова, которая добавляется в конец очереди с именем `имяОчереди` для всех элементов обернутого набора.

`.queue(имяОчереди, имяНовойОчереди)` – `имяОчереди` – строка, содержащая имя очереди. `имяНовойОчереди` – заменяет текущую очередь всех элементов набора на `имяНовойОчереди`, состоящую из массива функций.

- Метод `.dequeue()`

Ставит следующую анимацию в очередь. Это означает, что если выполняется анимация, когда этот метод вызывается, то новый анимация будет начата сразу после того, как текущая закончит свое выполнение. Выполняет первую функцию очереди для всех элементов набора, после выполнения функция удаляется из очереди. Обработчики события будут выполнять и удалять из очереди поочередно последующие функции. Если метод вызывается внутри функции, добавляемой в очередь, это вызовет поочередное исполнение всех функций, входящих в очередь.

`.dequeue(имяОчереди)` – `имяОчереди` – строка, содержащая имя очереди.

- Метод `.clearQueue()`

Удаляет все функции анимации из очереди, не только текущую, но и все последующие, без их выполнения. Может использоваться для удаления любых очередей.

`.clearQueue(имяОчереди)` – `имяОчереди` – строка, содержащая имя очереди.

- Метод `.delay()`

Позволяет установить отсрочку для запуска эффектов, ожидая определенное количество миллисекунд, прежде чем запустить следующий эффект в очереди.

`.delay(число или строка, имяОчереди)` – `число или строка` – устанавливает продолжительность задержки в миллисекундах или с помощью ключевых слов – `fast` (эквивалентно 200ms) и `slow` (эквивалентно 600ms). `имяОчереди` – необязательный параметр, указывает на очередь, для которой устанавливается задержка, по умолчанию – это очередь `"fx"`.

- Метод `.stop()`

Останавливает текущую анимацию в очереди сразу после его запуска для элементов набора. Если при вызове метода передаются некоторые аргументы, то вы также можете очистить очередь и определить, должны ли элементы при

остановке анимации остаться на месте, или они должны вернуться к первоначальному состоянию. По умолчанию оба параметра имеют значение `false`.

`.stop(логическое значение1, логическое значение2)` – логическое значение1 – значение `true` останавливает все анимационные эффекты, находящиеся в очереди. логическое значение2 – значение `true` останавливает все эффекты, кроме текущего.

- Метод `.finish()`

Останавливает текущую анимацию, удаляет все очереди анимации и завершает все анимации для соответствующих элементов.

`.finish(имяОчереди)` – имяОчереди – задает имя очереди анимации, которую нужно остановить. По умолчанию – очередь "fx".

Управление анимацией через свойства объекта jQuery

Свойство `jQuery.fx.interval` позволяет изменить скорость выполнения анимации. В качестве значения указывается количество миллисекунд. Значение по умолчанию `13ms`. Уменьшая скорость, можно добиться более плавного выполнения эффектов.

```
jQuery.fx.interval = 500;
```

Свойство `jQuery.fx.off` используется для того, чтобы глобально отключить или включить анимацию. Значение по умолчанию `false`, что соответствует выполнению анимации. Если задано значение `true`, то все методы анимации будут отключены, а элементы вернутся к своему первоначальному состоянию.

Методы объекта window

`.close()` – закрывает указанное окно.

`.blur()` – выводит окно из фокуса.

`.focus()` – помещает окно поверх всех других окон.

`.moveBy()` – перемещает окно на заданное число пикселей вправо и вниз.

`.moveTo()` – перемещает окно в заданное место на экране.

`.resizeBy()` – изменяет ширину и высоту окна. Первый аргумент определяет, на сколько пикселей должна увеличиться ширина окна, а второй – на сколько пикселей больше должна стать высота: `resizeBy(100,200)`. Для уменьшения размера окна нужно использовать отрицательные значения.

`.resizeTo()` – изменяет размеры окна на заданные.

`.scrollBy()` – прокручивает документ внутри окна на заданное количество пикселей вправо и вниз.

`.scrollTo()` – прокручивает документ внутри окна на заданную позицию по отношению к левому и верхнему краям.

`.scrollTop(), .scrollLeft()` – позволяют определить, сколько пикселей документа расположено сверху и слева от окна браузера, помогая предотвратить позиционирование за пределами окна браузера. Возвращает значение прокрутки по вертикали и горизонтали соответственно для первого элемента в наборе. Методы работают как с видимыми, так и с невидимыми элементами. Параметров нет.

События jQuery

События jQuery помогают сделать веб-страницы интерактивными, реагируя на простейшие действия пользователя. События представляют собой точный момент, в который что-либо происходит, например щелчок кнопки мыши.

Момент, в который произошло событие, называется запуском события. События могут срабатывать при выполнении различных операций с веб-страницей. Помимо этого, и сам браузер может стать источником событий.

События мыши

`.click()` – это событие запускается, когда нажимаете и отпускаете кнопку мыши. Применяется к ссылкам, картинкам, кнопкам, абзацам, блокам и т.д.

`.dblclick()` – событие запускается, когда дважды нажимаете и отпускаете кнопку мыши, например, открываете какую-нибудь папку на рабочем столе.

`.mousedown()` – событие происходит во время нажатия кнопки мыши, например, при перетаскивании элементов.

`.mousemove()` – событие запускается, когда передвигаете указатель мыши по странице.

`.mouseout()` – событие запускается, когда уводите указатель мыши с элемента.

`.mouseover()` – событие запускается, когда проводите указателем мыши по элементу (аналогично использованию псевдокласса `:hover`).

`.mouseup()` – событие активизируется, когда отпускаете кнопку мыши.

События документа/окна

`.load()` – событие запускается, когда браузер загрузит все файлы веб-страницы: html-файлы, внешние css и javascript файлы, медиа-файлы. Это может быть неудобно в случае если веб-страница содержит большое количество файлов. Для решения данной проблемы можно воспользоваться функцией `ready()`, которая запускает скрипты сразу после загрузки html-кода.

`.resize()` – событие запускается, когда изменяете размер окна браузера.

`.scroll()` – событие запускается, когда используете полосы прокрутки, либо прокручиваете веб-страницу с помощью колесика мыши, или же используете для этих целей клавиши клавиатуры (pg up, pg dn, home, end).

`.unload()` – событие запускается, когда собираетесь покинуть страницу, щелкая по ссылке для перехода на другую страницу, закрываете вкладку страницы или же окно браузера.

События форм

`.blur()` – событие запускается, когда поле формы выводится из фокуса, т.е. ввели данные в поле формы и перешли на другое.

`.change()` – событие запускается при изменении статуса поля формы, например при выбора пункта из выпадающего меню.

`.focus()` – событие запускается при переходе в поле формы, щелкая в нем кнопкой мыши или клавишей табуляции.

`.reset()` – событие позволяет вернуть форму в первоначальное состояние, отменив сделанные изменения.

`.select()` – событие запускается, когда вы выделяете текст внутри текстового поля формы.

`.submit()` – событие запускается, когда отправляете заполненную форму с помощью щелчка по кнопке «Отправить» или нажатия клавиши «Enter», когда курсор помещен в текстовом поле.

События клавиатуры

.keydown() – событие запускается при нажатии клавиши, перед событием keypress.

.keypress() – событие запускается, когда нажимаете на клавишу, до тех пор, пока не отпустите клавишу.

.keyup() – событие запускается, когда отпускаете клавишу.

События jQuery

.hover() – данная функция работает как событие, позволяя одновременно решить две задачи, связанные с событием наведения указателя мыши и событием снятия указателя мыши в отношении выбранного объекта. Основная структура функции:

```
$( '#селектор' ).hover(функция1, функция2);
```

.toggle() – событие работает аналогично событию hover(), с разницей в том, что событие запускается от щелчка кнопкой мыши. Например, можно открыть выпадающее меню одним щелчком и скрыть вторым.

Объект события

При запуске события браузер сохраняет информацию о нем в объекте события. Объект события содержит данные, собранные в момент, когда событие произошло. Обработка события происходит с помощью функции, при этом объект передается функции как аргумент – переменная evt (в качестве имени переменной можно использовать также event или просто e). Чтобы получить доступ к аргументу, необходимо добавить к функции имя параметра. В пределах данной функции можно получить доступ к различным свойствам, используя точечный синтаксис. Когда функция вызывается, то объект события сохраняется в переменной evt.

Объект события имеет различные свойства, наиболее распространенные из них описаны в нижеприведенной таблице.

Свойство	Описание
pageX	Расстояние (px) от указателя мыши до левого края окна браузера.
pageY	Расстояние (px) от указателя мыши до верхнего края окна браузера.
screenX	Расстояние (px) от указателя мыши до левого края монитора.
screenY	Расстояние (px) от указателя мыши до верхнего края монитора.
shiftKey	true, если, когда происходило событие, была нажата клавиша shift.
which	Используется для определения числового кода нажатой клавиши (вместе с shiftKey).
target	Означает, что по объекту события щелкнули кнопкой мыши (например, для события click()).
data	Объект, использованный с функцией bind() для передачи данных функции, управляющей событием.

Селекторы jQuery

Селекторы jQuery выбирают элементы веб-страницы, а методы выполняют операции с этими элементами.

Чтобы выбрать элементы, нужно передать селектор функции \$(), например, \$("img:odd"). Данный селектор будет применен ко всему дереву DOM, и чтобы

ограничить процедуру отбора элементов, можно указать определенный фрагмент дерева DOM – `$("img:odd", "div#slideshow")`. Таким образом будут выбраны все четные картинки из блока с `id=slideshow`.

Для более точного выбора элементов селекторы можно комбинировать, например, следующая запись позволит выбрать все флагки полей формы, которые были выделены пользователем – `$("#input[type=checkbox][checked]")`.

А с помощью этой комбинации селекторов `$("#input:checkbox:checked:enabled")` можно выбрать только активные и отмеченные флагки полей формы.

Также, допускается объединять несколько селекторов в одно выражение, разделяя селекторы запятой – `$("#p,span")`, что позволит отобрать все элементы `<p>` и ``.

Селектор	Описание, пример
элемент	Выбирает все элементы данного типа на странице, например, <code>\$("#div")</code> .
элемент1 элемент2	Выбирает все элементы2, вложенные непосредственно в элемент1, например, <code>\$("#p img")</code> .
класс	Выбирает все элементы заданного класса, например, <code>\$(".sidebar")</code> .
идентификатор	Выбирает элемент с указанным идентификатором, например, <code>\$("#main")</code> .
элемент класс	Выбирает из элементов данного типа только те элементы, которым назначен указанный класс, например, <code>\$("#p.first")</code> .
потомок	Выбирает все указанные элементы выбранного селектора, например, <code>\$(".sidebar a")</code> .
дочерний	Выбирает элементы, соответствующие второму селектору, которые содержатся непосредственно внутри первого селектора, являющиеся дочерними по отношению к нему, например, <code>\$("#body > p")</code> .
сестринский	Выбирает элементы, соответствующие второму селектору, идущие непосредственно за первым элементом, являющимся для него сестринским, например, <code>\$("#h2 + p")</code> . Выбирает элементы, соответствующие второму селектору, являющиеся сестринскими по отношению к первому элементу и расположенные после него, например, <code>\$("#h2 ~ p")</code> .
атрибут	Выбирает все элементы, которые содержат данный атрибут или указанно значение атрибута, например, <code>\$("#img[alt]")</code> , <code>\$("#a[href]")</code> , <code>(\$("#input[type='text'])")</code> . Выбирает все элементы, начинающиеся с определенного значения, например, <code>(\$("#a[href^='http://'])")</code> . Выбирает все элементы, заканчивающиеся на определенное значение, например, <code>(\$("#a[href\$='.pdf'])")</code> . Выбирает все элементы, содержащие в любом месте определенное значение, например, <code>(\$("#a[target*='blank'])")</code> .
:even	Выбирает элементы по четным значениям индекса 0, 2, 4..., т.е. выбирает 1, 3, 5... элементы, например, <code>(\$("#li:even")</code> .
:odd	Выбирает элементы по нечетным значениям индекса, т.е. выбирает 0, 2, 4... элементы.

:first	Выбирает только один элемент, первый из подходящих, например, <code>\$("p:first")</code> .
:last	Выбирает только один элемент, последний из подходящих.
:first-child	Выбирает элементы, которые являются первыми дочерними элементами своих родителей.
:last-child	Выбирает элементы, которые являются последними дочерними элементами своих родителей.
:only-child	Выбирает элементы, являющиеся единственными дочерними элементами своих родителей.
:nth-child(n)	Выбирает элементы, которые являются n-дочерними элементами своих родителей.
:nth-child(Xn+Y)	Выбирает n-элемент, порядковый номер которого рассчитывается по формуле в круглых скобках.
:nth-of-type(n)	Выбирает элементы, являющиеся n-ми дочерними элементами данного типа для своих родителей.
:parent	Выбирает непустые элементы, которые имеют вложенные (дочерние) элементы, а также содержащие текст.
:eq(n)	Выбирает элементы с индексом n, при этом индексы отсчитываются от нуля.
:gt(n)	Выбирает все элементы, индекс которых больше n, индексы отсчитываются от нуля.
:lt(n)	Выбирает все элементы, расположенные перед n-элементом, не включая n-элемент.
:not(селектор)	Позволяет выбрать элемент, не соответствующий данному типу селектора, например, <code>\$("a:not(.link)")</code> , <code>\$("a:not([href\$='.pdf'])")</code> .
:has(селектор)	Выбирает элементы, которые содержат внутри себя указанный селектор, например, элементы списка, содержащие внутри себя ссылки: <code>\$("li:has(a)")</code> .
:contains(текст)	Выбирает элементы, которые содержат указанный в скобках текст, например, <code>\$("a:contains(Скачать)")</code> .
:hidden	Выбирает скрытые элементы, для которых установлено значение <code>display: none;</code> , а также элементы форм со значением <code>type="hidden"</code> например, <code>\$("ul:hidden").show()</code> - делает скрытые элементы видимыми.
:visible	Выбирает видимые элементы, к видимым элементам относятся элементы, размеры которых больше нуля, а также элементы со значением <code>visibility: hidden</code> и <code>opacity: 0</code> .
:active	Выбирает элемент, который активизирован пользователем, например, с помощью щелчка мыши.
:checked	Выбирает только отмеченные флагки или радиокнопки.
:focus	Выбирает элемент, находящийся в фокусе.
:hover	Выбирает элемент, на который наведен указатель мыши.
:disabled	Выбирает неактивные элементы (форм).
:enabled	Выбирает активные элементы (форм).

:empty	Выбирает элементы, не содержащие дочерних элементов.
:target	Выбирает элементы, на которые ссылается идентификатор фрагмента в url-адресе.
:animated	Выбирает все анимируемые в данный момент элементы.
:button	Выбирает все кнопки <code>input[type=submit]</code> , <code>input[type=reset]</code> , <code>input[type=button]</code> , <code>button</code> .
:checkbox	Выбирает элементы-флажки <code>input[type=checkbox]</code> .
:file	Выбирает элементы типа <code>file</code> , <code>input[type=file]</code> .
:header	Выбирает элементы-заголовки от <code>h1</code> до <code>h6</code> .
:image	Выбирает изображения в элементах форм <code>input[type=image]</code> .
:input	Выбирает элементы форм <code>input</code> , <code>select</code> , <code>textarea</code> , <code>button</code> .
:password	Выбирает элементы ввода пароля <code>input[type=password]</code> .
:radio	Выбирает радиокнопки <code>input[type=radio]</code> .
:reset	Выбирает кнопки сброса <code>input[type=reset]</code> , <code>button[type=reset]</code> .
:selected	Выбирает выделенные элементы <code>option</code> .
:submit	Выбирает кнопки отправки формы <code>input[type=submit]</code> , <code>button[type=submit]</code> .
:text	Выбирает элементы ввода текста <code>input[type=text]</code> .

Правила веб-типоврафики

Типографика играет ключевую роль в веб-дизайне, по статистике 95% содержимого сайтов составляет текстовый контент. Шрифтовое оформление управляет настроением и создает определенную атмосферу при прочтении текстового содержимого веб-страниц.

Современная веб-типоврафика базируется на CSS-стилях. Меняя значения стилей браузера по умолчанию, можно сделать текстовое содержимое более привлекательным.

Семейство шрифтов (свойство `font-family`)

Подбор шрифта начинается с выбора гарнитуры шрифта. **Гарнитура** – шрифт или несколько шрифтов, имеющих стилистическое единство начертания. Состоит из набора знаков (обычно – цифры, буквы, знаки пунктуации, спецсимволы, но может состоять так же исключительно из неалфавитных знаков). Например, гарнитура «Times New Roman» состоит из обычного, курсивного, полужирного и множества других шрифтов этого семейства.

Гарнитуры можно разделить на две основные категории: с засечками (антиква и брусковые шрифты) и без засечек (готески).

Шрифты без засечек имеют простой и четкий внешний вид. Шрифты с засечками, напротив, придают более серьезный и официальный тон.

В процессе чтения глаза привыкают к основному шрифту, и они утомляются, если заголовки, оглавление и второстепенный текст набраны шрифтами разной гарнитуры, не гармонирующими с основным шрифтом. Поэтому, при подборе шрифтов достаточно остановиться на одном-двух шрифтах, а акценты расставлять за счет размера, цвета, начертания и т.п.

В каких случаях можно использовать несколько шрифтов:

- в гарнитуре нет специальных начертаний (жирного, полужирного, курсивного);
- необходимо добиться сходства с определенной эпохой;
- для визуального разделения текстов нескольких типов (например, комментарии в тексте, строки кода, формулы, тексты на другом языке);
- для эстетических целей.

[Сервис TypeTester для подбора шрифта.](#)

Сочетание шрифтов с засечками и без засечек

В рекомендациях по сочетанию шрифтов преобладает негласное правило – для заголовков выбирается шрифт без засечек, а основной текст формируется шрифтом с засечками. Однако такой подход не настолько популярен, как может показаться.

На самом деле это правило касается в первую очередь печатных изданий, книгопечатания. Связано это с тем, что шрифт с засечками (небольшими росчерками на концах основных штрихов) плавно выстраивается в одну линию, облегчая восприятие текста и делая его более читабельным.

Иначе обстоят дела с отображением текста дисплеями различных устройств. Возникает проблема сглаживания и неровности засечек. Поэтому для основного текста лучше подходит шрифт без засечек с несколько увеличенной высотой строчных знаков.

Цвет шрифта (свойство color)

Цвет придает тексту четкость и выразительность. Цветные заголовки и небольшие акценты в тексте способны привлечь больше внимания, чем текст черного цвета.

Тем не менее, не стоит забывать, что любой цвет несет с собой собственное настроение, а у каждого человека есть личные ощущения, которые вызывает у них тот или иной цвет.

Теплый цвет активно привлекает внимание к тексту, делая его визуально большим по размеру, чем шрифт аналогичного размера холодных оттенков. Для небольших по размеру элементов текста подходят более яркие цвета, боковое содержимое страницы можно выделить при помощи цвета, который на 20–30% светлее цвета текста основного содержимого веб-страницы.

При выборе количества цветов текста предпочтительно ограничиться двумя достаточно контрастными цветами (не считая черного и белого цвета). Черный текст на белом фоне – это классика, достаточно контрастная.

Размер шрифта (свойство font-size)

Размер базового шрифта в браузере равен 16px, а размер заголовков устанавливается пропорционально размеру базового шрифта (`h1` – 2em, `h2` – 1.5em, `h3` – 1.17em и т.д.).

Варьируя размером шрифта, можно придать тексту визуальную значимость и привлечь внимание посетителей к наиболее важным фрагментам текста. Как правило, чем крупнее элемент, тем он важнее.

На размер текста в окне браузера оказывает влияние разрешение монитора пользователя: текст одного и того же размера на мониторе с большим разрешением кажется меньше, чем текст такого же размера на мониторе с более низким разрешением.

Шрифты различных семейств одинакового размера также могут иметь разный фактический размер.

С ростом разрешения экранов и размеров мониторов необходимо пересмотреть привычный размер текста в 12–14px. Для обычного текста уже повсеместно применяется шрифт размером 14–18px. Задавая размер шрифта, нужно не забывать про адаптивность, т.е. размер шрифта должен изменяться в зависимости от размера экрана.

Выравнивание текста (свойство `text-align`)

Выровненный по ширине текст хорошо выглядит на печатной странице за счет равномерного распределения слов по строкам. Подобное форматирование веб-страниц средствами CSS невозможно и выравнивание текста по ширине образует большие неприятные промежутки между словами. Поэтому, в веб-типоврафике обычно используется выравнивание по левой стороне.

Межбуквенный и межсимвольный интервал (свойства `word-spacing` и `letter-spacing`)

Межбуквенный интервал оказывает влияние на читабельность текста. Варьирование плотности текста (уплотненный и разряженный текст) позволит разнообразить темп восприятия, добившись баланса текстовой композиции сайта.

Шрифты с засечками смотрятся более выразительно при уменьшенном значении `letter-spacing`.

Чем меньше шрифт, тем дальше друг от друга должны стоять буквы, чем крупнее – тем плотнее должны быть расположены буквы.

Межстрочный интервал (свойство `line-height`)

Оптимальное значение межстрочного интервала в 1,4–1,6 раза больше размера шрифта.

Также, не делайте больших отступов между заголовками и связанным с ним абзацем.

Длина строки

В строке для сплошного чтения должно помещаться от 30 до 75 знаков (приблизительно 7–10 слов на одной строке). Чем шире строка текста, тем больше должен быть межстрочный интервал. Межстрочное расстояние не должно быть меньше пробела между словами.

Начертание шрифта (свойство `font-style`)

Можно создавать красивую типографику, управляя начертанием шрифта. Например, курсив придает тексту некую торжественность. Жирное начертание плюс увеличенный размер шрифта позволяет выделить нужное содержимое, но только в том случае, если такой текст будет выделяться на фоне расположенных рядом объектов.

Свойство `font-variant: small-caps;` придает тексту типографскую изысканность, преобразуя текст таким образом, что все буквы отражаются малыми прописными. Данный прием подходит для небольших фрагментов текста.

Структура текста и визуальная иерархия

В структуре веб-страницы выделяют следующие объекты:

- Логотип / название сайта;
- Названия / заголовки;
- Заголовки второго плана;
- Основной текст;
- Навигация;
- Гипертекстовые ссылки;

- Длинные цитаты;
- Боковые панели;
- Подписи / надписи в таблицах, рисунках.

Каждый элемент списка представляет собой фундаментальную часть сайта и это оправдывает необходимость его выделения.

В HTML существует шесть уровней заголовков, начиная с более важного `<h1>` и заканчивая менее значимым `<h6>`. Заголовок `<h1>` должен быть первым в структуре документа, а заголовки низших уровней должны идти за ним и детализировать информацию. Для выделения заголовков можно воспользоваться приемом выделения цветом части заголовка.

Для более легкого усвоения текст нужно разбивать на части и для каждого раздела выбрать заголовок соответствующего уровня. Чем выше уровень заголовка, тем значимее по содержанию должен быть раздел.

В общей структуре текста особое внимание нужно уделять ссылкам, они должны с легкостью выделяться из окружающего их контента.

Визуальная иерархия устанавливает на странице правильную структуру, благодаря чему текст становится прост для восприятия и чтения. В большинстве текстов присутствует деление текста по смыслу. Существует несколько способов, самый простой – разбивка текста на абзацы.

- Капитель;
- Курсив;
- Полужирное/жирное начертание;
- Размер;
- Цвет;
- Смена гарнитуры;
- Изменение положения знаков на полосе текста (верхний и нижний регистр, отступы);
- Выделение знаков при помощи графических элементов – указателей, рамок, иконок и т.п.

Слишком сильное выделение не только чрезмерно акцентирует внимание на каком-либо фрагменте, но и затрудняет чтение.

Шрифт Awesome

Шрифт Awesome – это коллекция масштабируемых векторных иконок. Иконки можно форматировать с помощью CSS-свойств, устанавливать для них цвет, размер, тень и многое другое.

Как установить шрифт Awesome

- Способ 1

Использовать версию файла `font-awesome.css`, размещенную на ресурсе CDNJS. Для этого нужно добавить следующий код в раздел `<head>`:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/fontawesome.min.css">
```

После того, как шрифт подключен, можно использовать иконки на своем сайте.

- Способ 2

Скачать шрифт с сайта <https://fontawesome.com>.

Можно использовать полную или минимизированную версию файла:

```
<link rel="stylesheet" href="http://ваш_сайт/css/font-awesome.css">
<link rel="stylesheet" href="http://ваш_сайт/css/font-awesome.min.css">
```

Как использовать иконки Font Awesome

Иконки можно добавить на веб-страницу двумя способами: задать соответствующие классы для элементов `<i>` и `` или добавить к нужному элементу с помощью псевдоэлементов `::before`, `::after` и соответствующего значения свойства `content`.

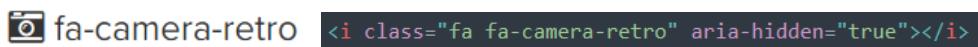
```
<i class="fa fa-home" aria-hidden="true"></i>

li::before {
  content: "\f015"; /* добавляем иконку дом */
  font-family: FontAwesome;
  color: #aaaaaa;
  margin-right: 10px;
}
```

Стандартные иконки

Font Awesome предназначен для использования со строчными элементами. Для того, чтобы добавить иконки, сначала нужно задать класс `fa` для элемента `<i>` или ``.

Чтобы добавить выбранную иконку перед элементом или после него, к элементу добавляется пустой элемент `<i></i>` или ``, которому назначен класс иконки, а также дополнительный класс, расширяющий стилевое оформление.



Большие иконки

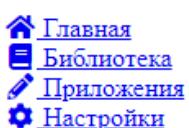
Чтобы увеличить размер иконки относительно ее контейнера, используются классы `fa-lg` (33% увеличение), `fa-2x`, `fa-3x`, `fa-4x` или `fa-5x`.



```
<i class="fa fa-camera-retro fa-lg"></i>
<i class="fa fa-camera-retro fa-2x"></i>
<i class="fa fa-camera-retro fa-3x"></i>
<i class="fa fa-camera-retro fa-4x"></i>
<i class="fa fa-camera-retro fa-5x"></i>
```

Иконки с фиксированной шириной

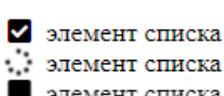
Используя класс `fa-fw`, можно зафиксировать ширину иконки. Это может пригодиться для оформления навигации или списков на сайте.



```
<ul>
  <li><a href="#"><i class="fa fa-home fa-fw"></i> Главная</a></li>
  <li><a href="#"><i class="fa fa-book fa-fw"></i> Библиотека</a></li>
  <li><a href="#"><i class="fa fa-pencil fa-fw"></i> Приложения</a></li>
  <li><a href="#"><i class="fa fa-cog fa-fw"></i> Настройки</a></li>
</ul>
```

Иконки для маркированного списка

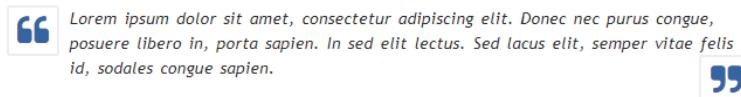
Используя классы `fa-ul` и `fa-li`, можно заменить маркеры по умолчанию в маркированном списке `...`.



```
<ul class="fa-ul">
  <li><i class="fa-li fa fa-check-square"></i>элемент списка</li>
  <li><i class="fa-li fa fa-spinner fa-spin"></i>элемент списка</li>
  <li><i class="fa-li fa fa-square"></i>элемент списка</li>
</ul>
```

Иконки в рамке и кавычки

Можно воспользоваться классом `fa-border`, чтобы установить рамку для иконки. Классы `pull-right` и `pull-left` добавят кавычки для текста.



```
<p><i class="fa fa-quote-left fa-3x pull-left fa-border"></i>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec nec purus congue, posuere libero in, porta sapien. In sed elit lectus. Sed lacus elit, semper vitae felis id, sodales congue sapien.</p>
```

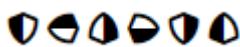
Анимированные иконки

Классы `fa-spin`, `fa-pulse`, `fa-spinner`, `fa-refresh` или `fa-cog`, предоставляют установку вращающихся иконок.

```
<i class="fa fa-spinner fa-spin"></i>
<i class="fa fa-circle-o-notch fa-spin"></i>
<i class="fa fa-refresh fa-spin"></i>
<i class="fa fa-cog fa-spin"></i>
<i class="fa fa-spinner fa-pulse"></i>
```

Трансформированные иконки

Чтобы повернуть иконки или отобразить их зеркально, можно использовать классы `fa-rotate-*` и `fa-flip-*`.



```
<i class="fa fa-shield"></i>
<i class="fa fa-shield fa-rotate-90"></i>
<i class="fa fa-shield fa-rotate-180"></i>
<i class="fa fa-shield fa-rotate-270"></i>
<i class="fa fa-shield fa-flip-horizontal"></i>
<i class="fa fa-shield fa-flip-vertical"></i>
```

Комбинированные иконки

Можно комбинировать иконки, накладывая одну на другую. Класс `fa-stack` используется для родительской иконки, класс `fa-stack-1x` – для стандартного размера и `fa-stack-2x` для увеличенного размера.



```
<span class="fa-stack fa-lg">
  <i class="fa fa-square-o fa-stack-2x"></i>
  <i class="fa fa-twitter fa-stack-1x"></i>
</span>

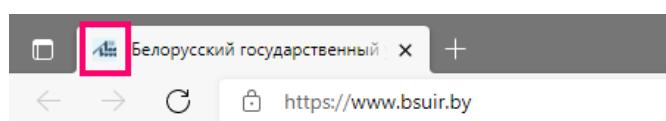
<span class="fa-stack fa-lg">
  <i class="fa fa-circle fa-stack-2x"></i>
  <i class="fa fa-flag fa-stack-1x fa-inverse"></i>
</span>

<span class="fa-stack fa-lg">
  <i class="fa fa-square fa-stack-2x"></i>
  <i class="fa fa-terminal fa-stack-1x fa-inverse"></i>
</span>

<span class="fa-stack fa-lg">
  <i class="fa fa-camera fa-stack-1x"></i>
  <i class="fa fa-ban fa-stack-2x text-danger"></i>
</span>
```

Favicon сайта

Фавикон (Favicon) – это иконка, которая отображается рядом с названием страницы во вкладке браузера. Также данный значок отображается в сниппете сайта на странице результатов поиска в некоторых поисковых системах, например Яндекс.



Фавикон повышает узнаваемость сайта во вкладке браузера, повышает кликабельность в результатах выдачи Яндекса, также он может способствовать уровню запоминаемости ресурса. В сети Интернет достаточно сервисов, предназначенных для генерации фавикон.

Как установить фавикон

Для установки favicon нужно иметь подходящее изображение. Оно должно быть размером 16×16 пикселей и иметь формат .ico. Фавикон можно создать самостоятельно или выбрать из существующих на специальных сайтах.

Данную иконку необходимо загрузить на сайт. После чего указать на нее ссылку между тегами <head>...</head>.

```
<link type="image/x-icon" href="/favicon.ico" rel="shortcut icon">
<link type="Image/x-icon" href="/favicon.ico" rel="icon">
```

Для добавления favicon необходимо разместить следующий html код:

В атрибуте href указывается адрес соответствующего файла.

После выполнения данных действий, если все сделано правильно, фавикон должен появиться во вкладке браузера на сайте.

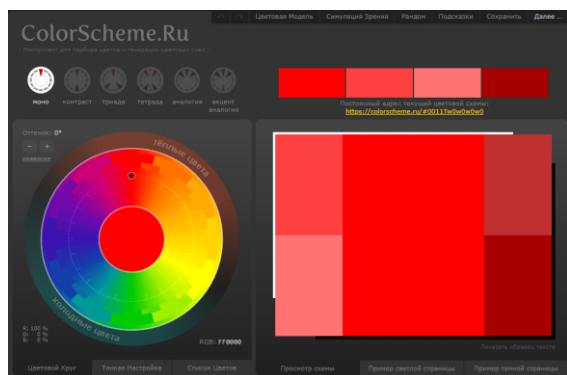
Сервисы подбора цвета для веб-дизайна

Цвета для веб-дизайна играют огромную роль. Чтобы грамотно подобрать цветовую схему для сайта, существуют специальные сервисы.

colorscheme.ru

Удобный инструмент для подбора цветов. У него много дополнительных возможностей. Например, можно посмотреть пример светлой и темной страницы с выбранными цветами.

Есть возможность оценить, как будут видеть цветовую гамму люди с дальтонизмом и другими особенностями зрения. Можно выбрать безопасные веб-цвета.



color.adobe.com

Тут можно не только создавать цветовые схемы, но и посмотреть и использовать схемы, созданные другими людьми.

Для этого надо нажать кнопку в верхнем меню «Смотреть».



[Color Palette Generator](http://color.adobe.com)

Сайт генерирует палитру из выбранного изображения. Необходимо указать ссылку на изображение.

Color Palette Generator

The screenshot shows a landscape image of a bridge over water with mountains in the background. To the right, a vertical color palette is displayed with six colors: dark grey (#222111), medium grey (#555555), light grey (#4499dd), white (#bbbbbb), dark blue (#666677), and light blue (#556688). Below the palette, the words "dull" and "vibrant" are written.

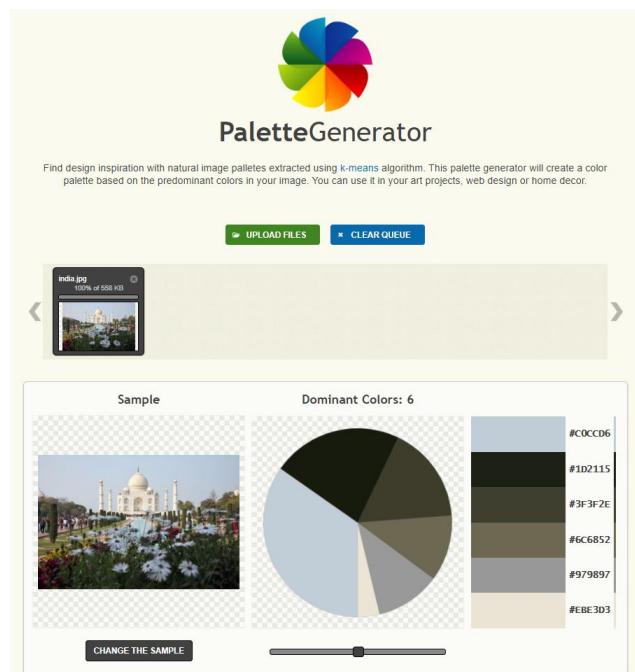
URL of image:
https://cdn.fine.ws/userfiles/102444.jpg
Color-Palette-ify!

Make color schemes. Enter the URL of an image to get a color palette that matches the image. This is useful for coming up with a website color scheme that matches a stock photo a client wants to work with. If you like this color palette generator, you might like [ColorHunter.com](#)

If you like this you might also like my [logo maker](#)

palettegenerator.com

Сайт генерирует палитру из выбранного изображения. Необходимо загрузить изображение со своего компьютера.



flatcolors.net

Как видно из названия, здесь можно подобрать цвета для плоского дизайна.



Адаптивная верстка сайта

Адаптивная верстка сайта позволяет веб-страницам автоматически подстраиваться под экраны планшетов и смартфонов. Мобильный интернет-трафик растет с каждым годом, и чтобы эффективно обрабатывать этот трафик, нужно предлагать пользователям адаптивные сайты с удобным интерфейсом.

Поисковые системы используют ряд критериев для оценки адаптивности сайта при просмотре на мобильных устройствах.

Google старается упростить пользование Интернетом для владельцев смартфонов и планшетов, отмечая в мобильной выдаче адаптированные под мобильные устройства сайты специальной пометкой **mobile-friendly**.

В Яндексе также работает алгоритм, который отдает предпочтение сайтам с мобильной/адаптивной версией для пользователей в мобильном поиске.

Верстка главной страницы

Страница состоит из трех основных блоков: верхний колонтитул (шапка), контейнер-обертка для основного содержимого и сайдбара, и нижний колонтитул (футер). В качестве переломных точек дизайна будем использовать 768px и 480px.

В первой точке скроем верхнее меню и переместим сайдбар под контейнер с постами. Во второй точке изменим расположение элементов шапки, отменим позиционирование кнопок социальных сетей в постах и отменим обтекание столбцов подвала страницы.

The screenshot shows a mobile-optimized travel blog homepage. At the top, there's a header with 'M E' initials, a search bar, and navigation links for 'БЛОГ', 'ПОРТФОЛИО', and 'ОБ АВТОРЕ'. Below the header is a large image of a traditional Chinese bridge over water with snow-capped mountains in the background. To the right of the image are two sidebar boxes: 'Категории' (Categories) listing 'Путешествия (1)' and 'Стажировки (1)', and 'Последние записи' (Recent Posts) showing thumbnails for 'Китай' and 'Индия'. Further down is a 'Подписка на рассылку' (Newsletter Subscription) box with an input field for 'Ваш email' and a red 'ПОДПИСАТЬСЯ' (Subscribe) button. The main content area features a large image of the Taj Mahal in India, with a 'ПРОДОЛЖИТЬ ЧТЕНИЕ' (Read More) button to its left. Below this image is a section titled 'СТАЖИРОВКИ' (Internships) with a thumbnail for 'Индия' and a brief description of India's geographical neighbors and historical significance. At the very bottom, there's a footer with social media icons for Facebook, VK, and Instagram, and a link to 'Написать письмо' (Write a message).

Метатеги и раздел `<head>`

Добавим в раздел `<head>` необходимые файлы – ссылку на используемые шрифты, библиотеку jQuery:

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="author" content="MEG">
  <title>Мой блог</title>
  <link rel="stylesheet" type="text/css" href="https://fonts.googleapis.com/css?family=Open+Sans:400,400italic,600,600italic|Playfair+Display:400,700&subset=latin,cyrillic">
  <link rel="stylesheet" type="text/css" href="style.css">
  <link type="Image/x-icon" href="favicon.ico" rel="shortcut icon">
  <link type="Image/x-icon" href="favicon.ico" rel="icon">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
  <script src="https://kit.fontawesome.com/bd671bd5e8.js" crossorigin="anonymous"></script>
</head>
```

Шапка страницы

В шапке страницы `<header>` поместим следующие элементы-контейнеры:

- логотип ``;
- кнопку для показа/скрытия главного меню `<div class="nav-toggle">`;
- главное меню `<ul id="menu">`;
- форму поиска по сайту `<form id="searchform">`.

```
<header>
  <nav class="container">
    <a class="logo" href="index.html" title="На главную">
      <span>M</span>
      <span>E</span>
    </a>
    <div class="nav-toggle"><span></span></div>
    <form action="" method="get" id="searchform">
      <input type="text" placeholder="Искать на сайте...">
      <button type="submit"><i class="fa fa-search"></i></button>
    </form>
    <ul id="menu">
      <li><a href="">Блог</a></li>
      <li><a href="">Портфолио</a></li>
      <li><a href="">06 авторе</a></li>
    </ul>
  </nav>
</header>
```

Блок с кратким содержанием статьи

Анонс статей обернем элементом `<article id="post-1" class="post">`:

```
<div class="container">
  <div class="posts-list">
    <article id="post-1" class="post">
      <div class="post-image"><a href=""></a></div>
      <div class="post-content">
        <div class="category"><a href="">Путешествия</a></div>
        <h2 class="post-title">Китай</h2>
        <p>Китайская Народная Республика, согласно конституции, - социалистическое государство. Является великой державой - потенциальной сверхдержавой, экономической сверхдержавой, постоянным членом Совета безопасности ООН. Одна из ведущих космических держав мира, обладает ядерным оружием и крупнейшей в мире армией по численности военнослужащих. Китайская экономика является второй экономикой мира по nominalному ВВП и первой экономикой мира по ВВП (ППС). КНР является мировым экспортёром (фабрика мира!). Располагает наибольшими в мире золотовалютными резервами.</p>
        <div class="post-footer">
          <a class="more-link" href="">Продолжить чтение</a>
          <div class="post-social">
            <a href="" target="_blank"><i class="fab fa-facebook"></i></a>
            <a href="" target="_blank"><i class="fab fa-vk"></i></a>
            <a href="" target="_blank"><i class="fab fa-instagram"></i></a>
          </div>
        </div>
      </div>
    </article>
    <article id="post-2" class="post">
      ...
    </article>
  </div> <!-- конец div class="posts-list"-->
```

Боковая колонка

В боковую колонку `<aside>` добавим список категорий, последние записи и форму подписки на рассылку:

```

<aside>
  <div class="widget">
    <h3 class="widget-title">Категории</h3>
    <ul class="widget-category-list">
      <li><a href="">Путешествия</a> (1)</li>
      <li><a href="">Стажировки</a> (1)</li>
    </ul>
  </div>
  <div class="widget">
    <h3 class="widget-title">Последние записи</h3>
    <ul class="widget-posts-list">
      <li>
        <div class="post-image-small"><a href=""></a></div>
        <h4 class="widget-post-title">Китай</h4>
      </li>
      <li>
        <div class="post-image-small"><a href=""></a></div>
        <h4 class="widget-post-title">Индия</h4>
      </li>
    </ul>
  </div>
  <div class="widget">
    <h3 class="widget-title">Подписка на рассылку</h3>
    <form action="" method="post" id="subscribe">
      <input type="email" name="email" placeholder="Ваш email" required>
      <button type="submit"><i class="fa fa-refresh fa-spin fa-fw"></i></button>
    </form>
  </div>
</aside>

```

Нижний колонтикул

В подвале страницы разместим информацию о копирайте, кнопки социальных сетей и ссылку на электронную почту:

```

<footer>
  <div class="container">
    <div class="footer-col"><span>Мой блог © 2022</span></div>
    <div class="footer-col">
      <div class="social-bar-wrap">
        <a title="Facebook" href="" target="_blank"><i class="fab fa-facebook"></i></a>
        <a title="VK" href="" target="_blank"><i class="fab fa-vk"></i></a>
        <a title="Instagram" href="" target="_blank"><i class="fab fa-instagram"></i></a>
      </div>
    </div>
    <div class="footer-col">
      <a href="mailto:makeichik@bsuir.by">Написать письмо</a>
    </div>
  </div>
</footer>

```

Общие CSS-стили

Общие стили, сброс стилей браузера по умолчанию:

```

*, *::after, *::before {
  box-sizing: border-box;
  padding: 0;
  margin: 0;
  transition: .5s ease-in-out;
  /* добавим плавность переходов для всех элементов страницы */
}

ul {
  list-style: none;
}

a {
  text-decoration: none;
  outline: none;
}

img {
  display: block;
  width: 100%;
}

h1, h2, h3, h4, h5, h6 {
  font-family: 'Playfair Display';
  font-weight: normal;
  letter-spacing: 1px;
}

body {
  font-family: 'Open Sans', arial, sans-serif;
  font-size: 14px;
  line-height: 1;
  color: #373737;
  background: #f7f7f7;
}

/* добавим очистку потока для всех контейнеров,
внутри которых задано обтекание дочерних элементов */

header::after, .container::after, footer::after, .widget-posts-list li::after, #subscribe::after {
  content: "";
  display: table;
  clear: both;
}

/* стилевой класс, который управляет шириной контейнера сетки*/
.container {
  margin: 0 auto;
  width: 100%;
  max-width: 960px;
  padding: 0 15px;
}

```

Стили для шапки и ее содержимого

```
header {  
    width: 100%;  
    background: white;  
    box-shadow: 3px 3px 1px rgba(0, 0, 0, .05);  
    padding: 15px 0;  
    margin-bottom: 30px;  
    position: relative;  
}  
  
/* логотип */  
.logo {  
    display: block;  
    float: left;  
}  
  
.logo span {  
    color: white;  
    display: inline-block;  
    width: 30px;  
    height: 30px;  
    line-height: 30px;  
    border-radius: 50%;  
    margin: 5px 0;  
    text-align: center;  
    text-shadow: 2px 2px 1px rgba(0, 0, 0, .4);  
}  
  
.logo span:nth-child(odd) {  
    background: #FF9F55;  
}  
  
.logo span:nth-child(even) {  
    background: #FF7055;  
}  
/* меню */  
.menu {  
    float: right;  
}  
  
.menu li {  
    display: inline-block;  
    margin-right: 30px;  
}  
  
.menu a {  
    color: #111;  
    text-transform: uppercase;  
    letter-spacing: 1px;  
    font-weight: 600;  
    display: block;  
    line-height: 40px;  
}  
  
.menu a:hover {  
    color: #EF5A42;  
}  
  
.menu li:last-child {  
    margin-right: 0;  
}  
  
#searchform {  
    float: right;  
    margin-left: 46px;  
    display: inline-block;  
    position: relative;  
} /* форма поиска */  
.searchform input {  
    width: 170px;  
    float: left;  
    border: none;  
    padding-left: 10px;  
    height: 40px;  
    overflow: hidden;  
    outline: none;  
    color: #9E9C9C;  
    font-style: italic;  
}  
  
.searchform button {  
    background: transparent;  
    height: 40px;  
    border: none;  
    position: absolute;  
    right: 10px;  
    color: #FF7055;  
    cursor: pointer;  
    font-size: 18px;  
}  
  
.searchform input:focus {  
    outline: 2px solid #EBEBEB;  
} /* кнопка переключения меню, появляющаяся при ширине 768px */  
.nav-toggle {  
    display: none;  
    position: relative;  
    float: right;  
    width: 40px;  
    height: 40px;  
    margin-left: 20px;  
    background: #EF5A42;  
    cursor: pointer;  
}  
  
.nav-toggle span {  
    display: block;  
    position: absolute;  
    top: 19px;  
    left: 8px;  
    right: 8px;  
    height: 2px;  
    background: white;  
}  
  
.nav-toggle span::before, .nav-toggle span::after {  
    position: absolute;  
    display: block;  
    left: 0;  
    width: 100%;  
    height: 2px;  
    background: white;  
    content: "";  
}  
  
.nav-toggle span::before {  
    top: -10px;  
}
```

Стили для блока с основным содержимым

```
/* левый контейнер */
.posts-list {
    margin-bottom: 30px;
    width: 64%;
    float: left;
}
/* блок для статьи */
.post {
    margin-bottom: 35px;
}
.post-content p {
    line-height: 1.5;
    padding-bottom: 1em;
}
.post-image {
    margin-bottom: 30px;
}
.category {
    margin-bottom: 15px;
}
.category a {
    color: #FF9F55;
    text-transform: uppercase;
}
.post-title {
    margin-bottom: 12px;
    font-size: 26px;
}
/* блок с кнопкой "продолжить чтение" и кнопками социальных сетей */
.post-footer {
    border-top: 1px solid #E8E8E8;
    border-bottom: 1px solid #E8E8E8;
    position: relative;
    margin-top: 15px;
}
.more-link {
    position: relative;
    display: inline-block;
    font-size: 10px;
    text-transform: uppercase;
    color: white;
    line-height: 44px;
    padding: 0 22px;
    background: #3C3D41;
    letter-spacing: 0.1em;
    white-space: nowrap;
}
.more-link::after {
    content: '';
    display: block;
    position: absolute;
    width: 0;
    height: 0;
    top: 0;
    right: 0;
    border: solid transparent;
    border-width: 22px 18px;
    border-left-color: #3C3D41;
    transform: translateY(100%);
}

```

```
.post-social {
    position: absolute;
    left: auto;
    top: 50%;
    right: 0;
    text-align: right;
    transform: translateY(-50%);
    padding: 0;
    font-size: 12px;
}
.post-social a {
    display: inline-block;
    margin-left: 8px;
    color: #FF7055;
    width: 25px;
    height: 25px;
    line-height: 23px;
    text-align: center;
    border-radius: 50%;
    border: 1px solid;
}
```

Стили для боковой колонки

```
/* правый контейнер */
aside {
    width: 33%;
    float: right;
}
/* блок для виджетов */
.widget {
    padding: 20px 15px;
    background: white;
    font-size: 13px;
    margin-bottom: 30px;
    box-shadow: 3px 3px 1px rgba(0, 0, 0, .05);
}
.widget-title {
    font-size: 18px;
    padding: 10px;
    margin-bottom: 20px;
    text-align: center;
    border: 2px solid #fadaa3;
    box-shadow: 3px 3px 0 0 #fadaa3;
}
.widget-category-list li {
    border-bottom: 1px solid #E8E8E8;
    padding: 10px 0;
    color: #c6c6c6;
    font-style: italic;
}
.widget-category-list li:last-child {
    border-bottom: none;
}
.widget-category-list li a {
    color: #626262;
    margin-right: 6px;
    font-style: normal;
}
.widget-category-list li a::before {
    content: "\f105";
    display: inline-block;
    font-family: 'FontAwesome';
    margin-right: 10px;
    color: #c6c6c6;
}
.widget-posts-list li {
    border-top: 1px solid #E8E8E8;
    padding: 15px 0;
}
.widget-posts-list li:nth-child(1) {
    border-top: none;
}
.post-image-small {
    width: 30%;
    float: left;
    margin-right: 15px;
}
.widget-post-title {
    float: left;
}
```

```
/* форма подписки */
#subscribe {
    position: relative;
    width: 100%;
    padding: 15px 0;
}

#subscribe input {
    width: 100%;
    display: block;
    float: left;
    border: 2px solid #E8E8E8;
    padding: 0 0 0 10px;
    height: 40px;
    position: relative;
    outline: none;
    color: #9E9C9C;
    font-style: italic;
}

#subscribe button {
    padding: 0 15px;
    background: transparent;
    height: 40px;
    border: none;
    position: absolute;
    right: 0;
    color: #FF7055;
    cursor: pointer;
    font-size: 18px;
}

#subscribe input:focus+button {
    background: #EF5A42;
    color: white;
}
```

Стили для нижнего колонитула

Подвал сайта разделим на три равных столбца:

```
.footer-col {  
    width: 33.333333333%;  
    float: left;  
}  
  
.footer-col a {  
    color: white;  
}  
  
.footer-col:last-child {  
    text-align: right;  
}  
  
.social-bar-wrap {  
    text-align: center;  
}  
  
.social-bar-wrap a {  
    padding: 0 7px;  
    font-size: 18px;  
}  
  
/* форма подписки */  
#subscribe {  
    position: relative;  
    width: 100%;  
    padding: 15px 0;  
}
```

Медиа-запросы

```
/* показываем кнопку для переключения верхней навигации */  
@media (max-width: 768px) {  
    .nav-toggle {  
        display: block;  
    }  
    header {  
        padding: 10px 0;  
    }  
    /* скрываем верхнее меню, отменяя обтекание, позиционируем его, сместив на высоту шапки сайта */  
    #menu {  
        max-height: 0;  
        background: white;  
        position: absolute;  
        overflow: hidden;  
        top: 63px;  
        right: 0;  
        left: 0;  
        margin: 0;  
        padding: 0;  
        float: none;  
        z-index: 3;  
    }  
    /* делаем элементы списка блочными, чтобы они располагались друг под другом */  
    #menu li {  
        display: block;  
        text-align: center;  
        border-bottom: 1px solid #E6E6E6;  
        margin-right: 0;  
    }  
    /* отменяем обтекание левой и правой колонок, устанавливаем им ширину 100% */  
    .posts-list, aside {  
        width: 100%;  
        float: none;  
    }  
    .widget-post-title {  
        font-size: 1.5em;  
    }  
}
```

```
/* отменяем обтекание для логотипа и выравниваем по центру */  
@media(max-width: 480px) {  
    .logo {  
        float: none;  
        margin: 0 auto 15px;  
        display: table;  
    }  
    .logo span {  
        margin: 0 2px;  
    }  
    /* позиционируем меню на увеличивающуюся высоту шапки */  
    #menu {  
        top: 120px;  
    }  
    /* позиционируем форму поиска по левому краю */  
    #searchform {  
        float: left;  
        margin-left: 0;  
    }  
    /* убираем верхнюю и нижнюю границы и выравниваем кнопку по центру */  
    .post-footer {  
        border-top: none;  
        border-bottom: none;  
        text-align: center;  
    }  
    /* отменяя позиционирование кнопок соцсетей */  
    .post-social {  
        position: static;  
        text-align: center;  
        transform: none;  
        margin-top: 20px;  
    }  
    .widget-post-title {  
        font-size: 1.2em;  
    }  
    /* отменяя обтекание для столбцов подвала страницы */  
    .footer-col {  
        float: none;  
        margin-bottom: 20px;  
        width: 100%;  
        text-align: center;  
    }  
    .footer-col:last-child {  
        text-align: center;  
        margin-bottom: 0;  
    }  
}
```

Скрипт для мобильного меню

За показ-скрытие верхнего меню при клике на кнопку (переключается высота меню – от нулевой до равной ее содержимому) отвечает код jQuery, который ранее добавили в разметку страницы перед закрывающим тегом </body>:

```
<script>  
    $('.nav-toggle').on('click', function () {  
        $('#menu').toggleClass('active');  
    });  
</script>
```

Верстка страницы

Для создания плиточной сетки макета страницы портфолио используются библиотеки [Masonry](#) и [imagesLoaded](#).



Copyright © 2022 Dobby

f vk

Masonry представляет собой сетку, базирующуюся на колонках. В отличие от сетки, созданной с помощью обтекания `float`, Masonry-сетка не имеет фиксированной высоты строк, что обеспечивает оптимальное использование пространства внутри веб-страницы, уменьшая любые ненужные пробелы. Такая сетка будет уместна на страницах-портфолио, страницах с галереями изображений, а также страницах с записями блога.

Метатеги и раздел `<head>`

Раздел `<head>` должен содержать ссылки на необходимые файлы – используемые шрифты и таблицу стилей.

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="author" content="MEG">
  <meta name="description" content="Photo Dobby">
  <title>Dobby the cat</title>
  <link href="https://fonts.googleapis.com/css?family=Muli:400,600,700|Radley" rel="stylesheet">
  <link rel="stylesheet" type="text/css" href="style.css">
  <script src="https://kit.fontawesome.com/bd671bd5e8.js" crossorigin="anonymous"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>
  <script src="https://unpkg.com/imagesloaded@4/imagesloaded.pkgd.min.js"></script>
  <script src="https://unpkg.com/masonry-layout@4/dist/masonry.pkgd.min.js"></script>
  <script src="main.js"></script>
</head>
```

Шапка страницы

Шапка страницы – раздел `<header>` содержит следующие элементы-контейнеры:

- логотип `<div class="logo">;`
- главное меню `<ul id="menu">;`
- кнопку для показа/скрытия главного меню `<div class="burger">.`

```
<header>
  <div class="container">
    <div class="header-row">
      <div class="logo">
        <a href="index.html">
          <span class="logo-name">Dobby</span>
          <span class="logo-description">
            <span>private photos</span>
          </span>
        </a>
      </div>
      <ul id="menu">
        <li>
          <a href="">Blog</a>
        </li>
        <li class="current">
          <a href="">Photo gallery</a>
        </li>
        <li>
          <a href="">About</a>
        </li>
      </ul>
      <div class="burger">
        <span></span>
      </div>
    </div>
  </div>
</header>
```

Сетка с фотографиями

Основная часть страницы состоит из блоков с фотографиями, для которых задан фильтр `grayscale`, придающий им эффект черно-белых фотографий. При наведении этот эффект исчезает.

```
<div class="main">
  <div class="container">
    <div class="row">
      <div class="grid">
        <div class="grid-item">
          <a class="grid-link" href="#">
            
          </a>
        </div>
        <div class="grid-item">
          <a class="grid-link" href="#">
            
          </a>
        </div>
      </div>
    </div>
  </div>
</div>
```

Подвал сайта

Нижний колонтикул содержит сведения о копирайте и ссылки на социальные сети.

```
<footer>
  <div class="container">
    <div class="footer-row">
      <div class="footer-col">
        <p>Copyright © 2022 Dobby</p>
      </div>
      <div class="footer-col">
        <div class="social-icons-wrapper">
          <a class="social-icon" target="_blank" href=""><i class="fa fa-facebook"></i></a>
          <a class="social-icon" target="_blank" href=""><i class="fa fa-vk"></i></a>
          <a class="social-icon" target="_blank" href=""><i class="fa fa-instagram"></i></a>
        </div>
      </div>
    </div>
  </div>
</footer>
```

Файл style.css

```
/*
 * box-sizing: border-box;
 * padding: 0;
 * margin: 0;
 */

body {
  font-family: 'Muli', sans-serif;
  font-size: 14px;
  line-height: 1.5;
  color: #19121e;
  background: white;
}

ul {
  list-style: none;
}

a {
  text-decoration: none;
  outline: none;
}

.container {
  width: 100%;
  max-width: 1130px;
  padding: 0 10px;
  margin: 0 auto;
}

.row {
  margin: 0 -10px;
}

/* ГАЛЕРЕЯ GRID */
.grid {
  margin-bottom: 20px;
}
.grid-item {
  width: 100%;
  padding: 10px;
}
.grid-item a {
  display: block;
}
.grid-item img {
  display: block;
  width: 100%;
  height: auto;
  -webkit-filter: grayscale(100%);
  filter: grayscale(100%);
  transition: .5s ease-in-out;
}
.grid-item a:hover img {
  -webkit-filter: grayscale(0%);
  filter: grayscale(0%);
}

/* ШАПКА */
.header {
  margin-bottom: 20px;
}
.header-row {
  display: flex;
  align-items: center;
  justify-content: space-between;
  padding: 20px 0;
  position: relative;
  border-bottom: 2px solid #d9d9d9;
}
.logo-name {
  display: block;
  font-size: 26px;
  line-height: 1;
  font-family: 'Radley', serif;
  text-transform: uppercase;
  color: #19121e;
}
.logo-description {
  position: relative;
  display: block;
  text-align: right;
  font-size: 10px;
  text-transform: uppercase;
  color: #a1787e;
}
.logo-description::before {
  content: "";
  position: absolute;
  top: 50%;
  left: 0;
  right: 0;
  height: 1px;
  background: #a1787e;
  z-index: -1;
}
.logo-description span {
  padding-left: 10px;
  background: white;
}
#menu {
  position: absolute;
  top: calc(100% + 2px);
  left: 0;
  right: 0;
  z-index: 10;
  visibility: hidden;
  opacity: 0;
  background: white;
  transition: .3s linear;
}

/* МЕНЮ */
.footer-row {
  padding: 20px 0;
  border-top: 2px solid #d9d9d9;
}
.footer-col {
  padding: 0 10px;
}
.footer-col:first-child {
  text-align: center;
}
.social-icons-wrapper {
  display: flex;
  justify-content: center;
  align-items: center;
  margin-top: 20px;
}
.social-icon {
  display: block;
  width: 24px;
  height: 24px;
  line-height: 24px;
  border-radius: 50%;
  margin-left: 12px;
  text-align: center;
  font-size: 15px;
  color: #a1787e;
  background: rgba(20, 20, 20, 0);
  transform: scale(1);
  transition: .3s linear;
}
.social-icon:hover {
  transform: scale(1.2);
  background: #a1787e;
  color: #fff;
}

/* МОБИЛЬНЫЙ МЕНЮ */
@media (min-width: 480px) {
  .header-row {
    display: flex;
  }
  .grid-item, .footer-col {
    width: 50%;
  }
  .social-icons-wrapper {
    justify-content: flex-end;
    margin-top: 0;
  }
  .header-col:first-child {
    text-align: left;
  }
}
@media (min-width: 768px) {
  .grid-item {
    width: 33.33333333333333px;
  }
  .burger {
    display: none;
  }
  .header-row {
    display: flex;
    align-items: center;
    justify-content: space-between;
  }
  #menu {
    display: flex;
    position: static;
    visibility: visible;
    opacity: 1;
  }
  #menu li {
    padding-left: 40px;
  }
  #menu li a {
    padding: 0;
  }
  #menu li a::before {
    content: '';
    position: absolute;
    transform: rotate(-45deg);
    right: 0;
    top: 0;
    width: 0;
    height: 1px;
    background: #19121e;
    transition: .25s cubic-bezier(.094, .648, .335, 1).15s;
  }
  #menu li a::before, #menu li.current a::before {
    width: 100px;
    left: 0;
  }
  #menu li.current a::before {
    background: #a1787e;
  }
}
```

Для полноценной работы нужно подключить следующие скрипты:

```
<script src="https://kit.fontawesome.com/bd671bd5e8.js" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/jquery@3.6.0/dist/jquery.min.js"></script>
<script src="https://unpkg.com/imagesloaded@4/imagesloaded.pkgd.min.js"></script>
<script src="https://unpkg.com/masonry-layout@4/dist/masonry.pkgd.min.js"></script>
<script src="main.js"></script>
```

Проблема с накладыванием изображений сетки решается с помощью плагина `imagesLoaded`. В `main.js` он определяет, когда изображение загружено и после этого строит сетку Masonry.

```
function ($) {
  $(window).on("load", function () {
    let masonryGrid = $(".grid");
    masonryGrid.imagesLoaded(function () {
      masonryGrid.masonry({
        itemSelector: ".grid-item",
        percentPosition: true
      });
    });
  });
  $(document).on("click", ".burger", function () {
    $(".burger").toggleClass("close");
    $("#menu").toggleClass("show");
  });
})(jQuery);
```

Загрузка и подключение слайдера

SimpleAdaptiveSlider – это простой адаптивный слайдер для сайта, написанный на чистом CSS и JavaScript.

Проект слайдера расположен на GitHub. Данный слайдер распространяется под лицензией MIT. Следовательно, его можно использовать бесплатно как в личных, так и в коммерческих проектах.

SimpleAdaptiveSlider имеет следующие характеристики:

- легкий (без jQuery), минимизированный размер JS кода;
- адаптивный, т.е. динамически подстраивающийся под заданные размеры `viewport`;
- без зацикливания, так и с бесконечной прокруткой;
- с автоматической сменой слайдов через определенные интервалы времени;
- возможность перелистывания слайдов посредством свайпа;
- в контенте слайда можно размещать различную информацию (текстовую, изображения, отзывы, товары и т.д.).

Коды слайдера состоят из CSS и JavaScript файлов:

- [simple-adaptive-slider.css](#) и его минимизированная версия [simple-adaptive-slider.min.css](#).
- [simple-adaptive-slider.js](#) и его минимизированная версия [simple-adaptive-slider.min.js](#).

Подключаем CSS и JavaScript файлы к странице:

```
<!-- Подключаем CSS слайдера -->
<link rel="stylesheet" href="simple-adaptive-slider.min.css">
<!-- Подключаем JS слайдера -->
<script defer src="simple-adaptive-slider.js"></script>
```

Вставляем в необходимое место страницы следующую разметку слайдера:

```
<!-- Разметка слайдера (html код) -->
<div class="slider">
  <div class="slider_wrapper">
    <div class="slider_items">
      <div class="slider_item">
        |--- Контент 1 слайда --
      </div>
      <div class="slider_item">
        |--- Контент 2 слайда --
      </div>
      <div class="slider_item">
        |--- Контент 3 слайда --
      </div>
      <div class="slider_item">
        |--- Контент 4 слайда --
      </div>
    </div>
    <!-- Стрелки для перехода к предыдущему и следующему слайду -->
    <a class="slider_control slider_control_prev" href="#" role="button" data-slide="prev"></a>
    <a class="slider_control slider_control_next" href="#" role="button" data-slide="next"></a>
  </div>
</div>
```

В этой разметки карусель состоит из 4 слайдов. Контент слайдов необходимо помещать в элементы "slider__item".

После подключение стилей и скриптов слайдера, а также вставки в нужное место страницы его разметки необходимо выполнить его инициализацию и настройку.

Инициализация слайдера осуществляется посредством создания нового экземпляра объекта типа SimpleAdaptiveSlider:

```
document.addEventListener('DOMContentLoaded', function () {
  // инициализация слайдера
  new SimpleAdaptiveSlider('.slider', {
    loop: true,
    autoplay: false,
    interval: 5000,
    swipe: true,
  });
});
```

В качестве первого аргумента необходимо передать селектор корневого элемента слайдера. Для примера выше это можно выполнить посредством селектора .slider.

Если, например, нужно инициализировать несколько слайдеров на странице, то к каждому можно добавить id, а затем использовать указанный идентификатор для выбора определенного слайдера.

```
<script>
document.addEventListener('DOMContentLoaded', function () {
  // инициализация 1 слайдера с нужными настройками
  new SimpleAdaptiveSlider('#slider-1', {
    loop: true,
    autoplay: false,
    interval: 5000,
    swipe: true,
  });
  // инициализация 2 слайдера с нужными настройками
  new SimpleAdaptiveSlider('#slider-2', {
    loop: true,
    autoplay: false,
    interval: 5000,
    swipe: true,
  });
});
</script>
<!-- 1 слайдер -->
<div class="slider" id="slider-1">...</div>
<!-- 2 слайдер -->
<div class="slider" id="slider-2">...</div>
```

Настройка слайдера осуществляется с помощью второго аргумента. Его нужно передавать в формате объекта, и он имеет 4 ключа:

- `loop` – отвечает за зацикленность; по умолчанию этот ключ имеет значение `true`; если слайдер не должен быть бесконечным, т.е. быть без зацикливания `loop` необходимо установить значение `false`;
- `autoplay` – включает автоматическую смену слайдов; время, через которое это необходимо выполнять определяется ключом `interval`;
- `interval` – время в миллисекундах через которое нужно автоматически переключать слайды; по умолчанию – 5000, т.е. 5 секунд;
- `swipe` – определяет можно ли слайды менять свайпом; по умолчанию включено.

Значение ключей во втором аргументе по умолчанию:

```
new SimpleAdaptiveSlider('.slider', {
  loop: true,
  autoplay: false,
  interval: 5000,
  swipe: true,
});
```

Пример инициализация слайдера без зацикливания:

```
new SimpleAdaptiveSlider('.slider', {
  loop: false,
});
```

Пример инициализация слайдера с автоматической сменой слайдов через 10 секунд:

```
new SimpleAdaptiveSlider('.slider', {  
    autoplay: true,  
    interval: 10000,  
});
```

Методы слайдера

Данный слайдер имеет несколько методов, с помощью которых можно им управлять:

- `next()` – перейти к следующему слайду;
- `prev()` – перейти к предыдущему слайду;
- `autoplay()` – включение и выключение автоматической смены слайдов.

Чтобы эти методы были доступны нужно при инициализации слайдера сохранить ссылку на созданный экземпляр объекта `SimpleAdaptiveSlider` в некоторую переменную:

```
const slider = new SimpleAdaptiveSlider('.slider', {  
    loop: false,  
});
```

Пример использования методов:

```
slider.next(); // переход к следующему слайду  
slider.prev(); // переход к предыдущему слайду  
slider.autoplay(); // включить автоматическую смену слайдов  
slider.autoplay('stop'); // выключить автоматическую смену слайдов
```

Пример с использованием методов слайдера

Например, создадим слайдер, в котором будем использовать свои кнопки для перехода к предыдущему и следующему слайду. Исходные кнопки удалим из разметки.

```
<!-- Разметка слайдера -->  
<div class="slider">  
  <div class="slider__wrapper">  
    <div class="slider__items">...</div>  
  </div>  
</div>  
<!-- Новые кнопки -->  
<div class="btn-wrapper">  
  <button class="btn btn-prev">PREV</button>  
  <button class="btn btn-next">NEXT</button>  
</div>
```

Для отключения индикаторов добавим код в CSS:

```
.slider__indicators {  
  display: none;  
}
```

Напишем, скрипт, который будет при нажатии на кнопки выполнять переход на предыдущий или следующий item:

```
document.addEventListener('DOMContentLoaded', function () {  
  // инициализация слайдера  
  let slider = new SimpleAdaptiveSlider('.slider');  
  // назначим обработчик при нажатии на кнопку .btn-prev  
  document.querySelector('.btn-prev').onclick = function () {  
    // перейдём к предыдущему слайду  
    slider.prev();  
  }  
  // назначим обработчик при нажатии на кнопку .btn-next  
  document.querySelector('.btn-next').onclick = function () {  
    // перейдём к следующему слайду  
    slider.next();  
  }  
});
```

Примеры использования слайдера

1. Использование слайдера для чередований изображений
[Смотреть пример](#)
2. Применение слайдера для текстовой информации
[Смотреть пример](#)
3. Использование слайдера с изображением и текстом
[Смотреть пример](#)

Краткое описание исходных кодов и принципа работы

HTML структура слайдера:

```
<div class="slider">
  <div class="slider__wrapper">
    <div class="slider__items">
      <div class="slider__item">
        ...
      </div>
      <div class="slider__item">
        ...
      </div>
      <div class="slider__item">
        ...
      </div>
      <div class="slider__item">
        ...
      </div>
    </div>
    <a class="slider__control slider__control_prev" href="#" role="button"></a>
    <a class="slider__control slider__control_next" href="#" role="button"></a>
  </div>
```

В разметке корневым элементом является тег `<div>` с классом `slider`.

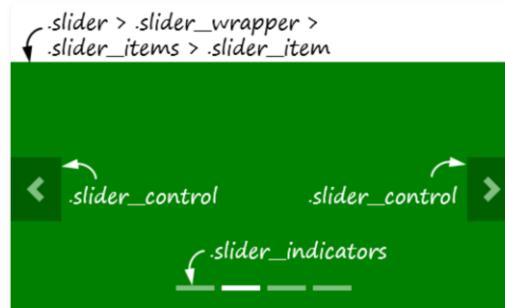
Внутри него находятся:

- `.slider__wrapper` – wrapper (обертка для слайдов);
- два элемента `.slider__control` – ссылки, оформленные в виде кнопок для перехода соответственно к предыдущему и следующему слайду;
- точки или другими словами индикаторы `.slider__indicators` – добавляются динамически посредством JavaScript кода.

Обертка `.slider__wrapper` содержит внутри `.slider__items`, а он в свою очередь непосредственно сами item (слайды). Это элементы – `.slider__item`.

Индикаторы (`.slider__indicators`) выполнены в виде нумерованного списка. Каждый элемент `li` внутри него содержит атрибутом `data-slide-to`. В нем содержится индекс слайда. Он используется в JavaScript коде и определяет слайд, на который нужно перейти в случае нажатия на него. Активный индикатор отмечается посредством класса `.slider__indicator_active`.

Кнопки «назад» и «вперед» размечены с помощью элемента `<a>`.

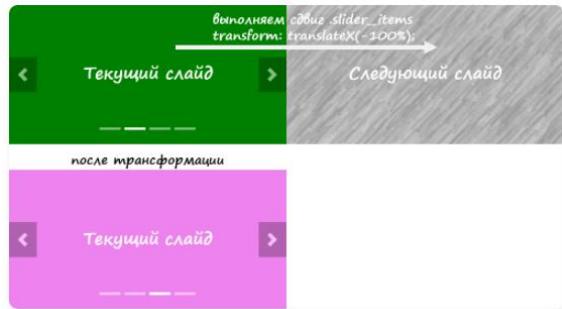


В определённый момент времени в слайдере отображается только один item, который не сдвинут относительно `.slider__wrapper`. Все другие слайды сдвинуты. Скрытие слайдов выходящих за пределы `.slider__wrapper` осуществляется с помощью свойства `overflow: hidden`.



Перемещение item выполняется с помощью CSS трансформации. Для этого к .slider__items и .slider__item в нужные моменты времени добавляется CSS-свойство transform: translateX(...) с нужным значением.

Например, когда к .slider__items добавляется transform: translateX(...) со значением -100%, то браузер осуществляет переход к следующему слайду, а если наоборот, 100% – то к предыдущему.



Анимация перехода осуществляется помошью CSS свойства transition со значение transform 0.5s ease;.

Скрипт слайдера имеет следующую укрупненную структуру:

```
// конструктор для создания экземпляров объектов типа SimpleAdaptiveSlider
function SimpleAdaptiveSlider(selector, config) { ... }
// добавляет класс к активным элементам и управляет видимостью кнопок управления
SimpleAdaptiveSlider.prototype._setActiveClass = function () { ... }
// выполняет смену слайдов
SimpleAdaptiveSlider.prototype._move = function () { ... }
// перемещает на слайд по его индексу
SimpleAdaptiveSlider.prototype._moveTo = function (index) { ... }
// включает таймер для автоматической смены слайдов и выключает его
SimpleAdaptiveSlider.prototype._autoplay = function (action) { ... }
// добавляет в разметку индикаторы
SimpleAdaptiveSlider.prototype._addIndicators = function () { ... }
// обновляет значения переменных, содержащих экстремальные значения слайдов
SimpleAdaptiveSlider.prototype._refreshExtremeValues = function () { ... }
// уравновешивает слайды (для защищенности)
SimpleAdaptiveSlider.prototype._balancingItems = function () { ... }
// назначаем обработчики для событий
SimpleAdaptiveSlider.prototype._addEventListener = function () { ... }

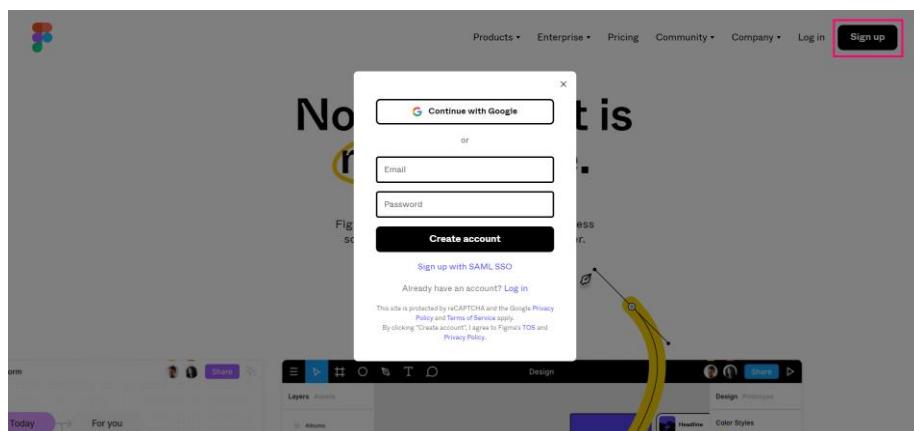
// публичные методы
SimpleAdaptiveSlider.prototype.next = function () { ... }
SimpleAdaptiveSlider.prototype.prev = function () { ... }
SimpleAdaptiveSlider.prototype.autoplay = function (action) { ... }
```

Figma для верстальщика

Figma – это графический онлайн-редактор для дизайнеров интерфейсов и веб-разработчиков. Сейчас это удобная, бесплатная альтернатива **Photoshop**. Большое преимущество платформы – возможность работать прямо в браузере. При этом есть и десктопная версия.

Регистрация и добавление нового макета

Чтобы начать пользоваться редактором, нужно зайти на [сайт](#) и зарегистрироваться. Можно войти через Google или просто создать аккаунт на сайте.



Затем вы попадете в менеджер проектов, где увидите образцы макетов. Их можно использовать для ознакомления с инструментами приложения. Здесь же можно добавить свой проект, нажав на соответствующую иконку **Import file**. Макет проекта должен иметь расширение **.fig** или **.sketch**.

После добавления макет появится в общем списке, и можно будет приступить к работе с ним.

Для выполнения задания к лабораторной работе необходимо изучить материал по [верстке сайта из Figma](#).

Анализ ключевых слов

Анализ ключевых слов – очень важная процедура для продвижения своего ресурса или своих услуг в Интернете, так как от выбора слов в значительной степени зависит и эффективность интернет-рекламы, и результативность оптимизации веб-ресурсов.

Существует несколько путей подбора ключевых слов.

- Во-первых, можно сформулировать ключевые запросы самостоятельно, представив, что именно и с какими формулировками будут искать пользователи. Но подобрать таким образом большой массив ключевых слов невозможно, как и быть уверенным в том, что они окажутся эффективными.
- Во-вторых, слова можно подобрать путем анализа ключевых слов сайта. Анализ может осуществляться либо визуально (просматривая контент сайта и выбирая наиболее часто встречающиеся слова), либо при помощи специальных инструментов, которые «просканируют» содержимое сайта и выдадут список релевантных для поисковой системы запросов.
- В-третьих, ключевые слова можно подобрать при помощи инструмента Яндекс.Вордстат (Yandex Wordstat).

Яндекс.Вордстат предоставляет обширную статистику поисковых запросов, с возможностью отслеживать результаты по регионам, по месяцам или по языку. В **Яндекс.Вордстат** помимо информации о количестве запросов, содержащих интересующие вас слова, имеется функция «подсказки». Под заголовком «Ищут также» отображаются похожие словосочетания, синонимы и близкие по темам запросы.

Яндекс.Вордстат

Главная страница сервиса [Яндекс.Вордстат](#) состоит из поисковой строки с небольшим набором настроек и кнопку «Подобрать».

The screenshot shows the Yandex.Вордстат search interface. At the top, there is a navigation bar with links: Директ, Справочник, Метрика, Рекламная сеть, Маркет, and ещё. Below the navigation bar is a search bar with three radio button options: По словам (selected), По регионам, and История запросов. To the right of the search bar is a 'Подобрать' button and a link 'Все регионы'. A large input field below the search bar contains placeholder text: 'Введите слово или словосочетание, обозначающее ваш товар или услугу, и нажмите кнопку "Подобрать".' Below this input field is a note: 'В результатах подбора будет приведена статистика запросов на Яндексе, включающая заданное вами слово или словосочетание (слева), и похожих запросов (справа).'. Another note explains: 'Цифры рядом с каждым запросом в результатах подбора слов дают предварительный прогноз числа показов в месяц, которое вы получите, выбрав этот запрос в качестве ключевого слова. Так, цифра рядом со словом "телефон" обозначает число показов по всем запросам со словом "телефон": "купить телефон", "сотовый телефон", "купить сотовый телефон", "купить новый сотовый телефон в кратинку" и т.п.'. At the bottom of the interface, there is a note: 'Если вы хотите узнать количество показов для пользователей из определенного региона, кликните по "Все регионы".'

Здесь же отображается краткая справка по работе с сервисом и в общих чертах описывается логика отображаемых данных.

Первое, что нужно сделать, – ввести фразу, которая будет использоваться в качестве ключевого запроса для поисковых служб. Например, «семиструнная электрогитара» или «смартфон iPhone». После этого на экране отобразится список запросов и их вариаций в левой колонке, а также схожие ключевые фразы в правой колонке.

Первый запрос в левой колонке содержит в себе все последующие. То есть расширенные ключевые слова под основным – это не дополнительные запросы, а вложенные. Это значит, что «семиструнная электрогитара» с 466 показами включает в себя «семиструнная гитара купить» с 154 показами из вышенназванных 466.

[Директ](#) [Справочник](#) [Метрика](#) [Рекламная сеть](#) [Маркет](#) [ещё](#)

семиструнная электрогитара

По словам По регионам История запросов

Подобрать Все регионы

Последнее обновление: 21.09.2021

Что искали со словом «семиструнная электрогитара» — 466 показов в месяц	
Статистика по словам	Показов в месяц
семиструнная электрогитара	466
семиструнная электрогитара купить	154
строй семиструнной электрогитары	34
струны +для семиструнной электрогитары	22
лучшая семиструнная электрогитара	15
семиструнная электрогитара купить +в екатеринбурге	9
чертежи семиструнной электрогитары	9
семиструнная электрогитара ibanez	8
семиструнная электрогитара shine	7
Как играть +на семиструнной электрогитаре	6
семиструнная электрогитара минск	6

Запросы, похожие на «семиструнная электрогитара»	
Статистика по словам	Показов в месяц
гитары гитар	4 491 435
+как настроить гитару 6 струнную новичку	1 182
гитара +с нуля уроки игры +на гитаре	4 630
мелодии +на гитаре +для начинающих 6 струн	610
комбик +для гитары	2 975
настройка гитары 6ти струнной через микрофон	663
кузнецик +на +одной струне +на гитаре	1 903
современные песни +на гитаре аккорды	1 097
валль бостон аккорды +для шестиструнной гитары	579
бой 8ка +на гитаре	278
кузнецик +на +одной струне +на гитаре	1 903
настройка 12 струнной гитары онлайн тюнер	154
+как настроить струны +на гитаре	681
vst гитара	2 115
12 струнная гитара купить	2 763
+для +чего коподстр +для гитары	555
евгений броневицкий поющие гитары биография	74
аккорды +на гитаре современных песен	1 097
freaks +на гитаре +на +одной струне	223

Не нужно складывать «ключи» друг с другом, так как получится некорректный расчет. Сами показатели условны. Это не точная статистика, а лишь прогноз на количество показов по выбранному ключевому слову.

В правой колонке отображаются отдельные запросы. Они независимы друг от друга и показывают количество вхождений только для самих себя. Без вложенных «ключей».

смартфон iphone

По словам По регионам История запросов

Подобрать Все регионы

Последнее обновление: 21.09.2021

Что искали со словом «смартфон iphone» — 46 355 показов в месяц	
Статистика по словам	Показов в месяц
смартфон iphone	46 355
смартфон apple iphone	39 177
смартфон iphone 12	13 794
смартфон iphone 11	13 063
смартфон apple iphone 12	12 622
смартфон iphone 11	12 064
смартфон apple iphone 128gb	10 451
купить смартфон iphone	8 073
смартфон apple iphone 64gb	7 794
купить смартфон apple iphone	7 419
смартфон iphone 12 pro	6 155
смартфон apple iphone 12 pro	5 685
смартфон iphone 12 128gb	4 810
смартфон apple iphone 12 128gb	4 662
смартфон apple iphone max	4 497
смартфон apple iphone 256gb	4 361
смартфон iphone 11 128gb	3 889
смартфон apple iphone 11 128gb	3 743
смартфон iphone 11 pro	3 674
смартфон iphone 12 pro max	3 632
смартфон iphone 11 64gb	3 410
смартфон iphone xr	3 376
смартфон apple iphone 11 64gb	3 317
смартфон apple iphone se	3 316
смартфон apple iphone xr	3 172
смартфон iphone 11 pro	3 130
смартфон iphone 12 pro max	3 032
купить смартфон iphone 11	2 904

Запросы, похожие на «смартфон iphone»	
Статистика по словам	Показов в месяц
купить смартфон телефон	37 431
купить смартфон телефон	37 431
айфон +на гарант	15 942
айфон less or	2 958
купить айфон 9	3 601
айфон se/2020	2 838
ростест айфон +что +это	2 471
ищу айфона	14 707
купить мобильный телефон	16 599
айфон ac в	1 590
айфон 5s	102 266
mobile phone	20 277
айфон 16	7 229
айфон 11	492
мобильный телефон	35 194
telephone	54 639
айфон 15	156
купить айфон телефон	15 103
переплата линий +с айфона +на айфон	1 499
недорогие смартфоны телефонны	7 140

Запросы в Яндекс.Вордстате можно дополнительно настроить с помощью специальных символов. Их ставят перед словом или фразой для выполнения какого-то условия. Соответственно, у каждого символа есть свое предназначение и выполняемое условие.

- **+** – этот символ обязывает систему Wordstat учитывать слово при поиске. По умолчанию некоторые слова русского языка игнорируются. Это касается предлогов и следующих союзов: в, на, от, для, как, из, и, от.
- **--** – этот символ позволяет более адекватно оценивать перспективность выбранной ниши за счет исключения из ключевой фразы всех лишних составляющих, способных повлиять на результат.
- **!** – этот символ запрещает Яндекс.Вордстату корректировать словоформу. То есть поиск будет учитывать исключительно ключевые слова с выбранным окончанием, числом, родом и т.п. Его рекомендуют использовать оптовикам, чтобы точно учитывать запросы на покупку большого количества товаров.
- **<>** – кавычки выводят статистику по выделенному слову или фразе в отдельное окно, чтобы можно было оценить количество запросов без вложенных «ключей». То есть увидеть запрос «купить электрогитару» без «купить электрогитару Ibanez» и других вариаций, которые учитываются в первом значении левого столбца Яндекс.Вордстата.
- **[]** – этим символом можно зафиксировать используемый в запросе порядок слов. Этим пользуются туристические фирмы и авиакомпании, чтобы предлагать клиентам билеты в точных (а не в похожих) направлениях.
- **(|)** – в скобки можно занести 2 или больше похожих слов. К примеру, если вы продаете товары сразу из двух стран, можно занести их в скобки и посмотреть статистику сразу по двум категориям запросов.

Задание к лабораторной работе №7

Для выполнения лабораторной работы необходимо установить и настроить редактор кода. Выполненные задачи необходимо разместить на GitHub в приватном репозитории и выслать ссылку преподавателю.

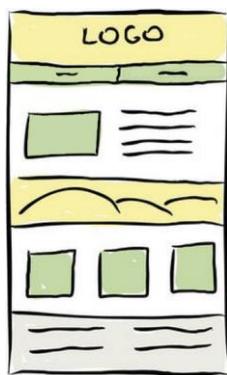
Задание к лабораторной работе №7 состоит из задач разного уровня сложности. **Выполнение задачи 1 оценивается максимально в 8 баллов, задач 1, 2 в 10 баллов.**

Задача 1

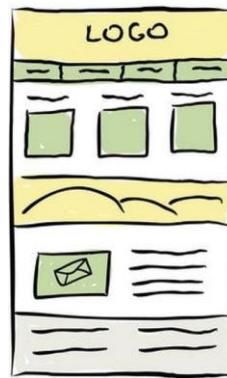
Условие: Необходимо сверстать посадочную страницу сайта.

1. Выбрать прототип согласно своему варианту (номер по списку в группе).
2. Осуществить выбор цветовой палитры для веб-дизайна используя сервисы подбора цветовой схемы сайта.
3. Подобрать иконочный шрифт используя Font Awesome.
4. Заполнить главную страницу своим уникальным контентом с общим смысловым содержанием.
5. Разработать и установить фавикон.
6. Подобрать и заполнить метатеги (description, keywords), используя Яндекс.Вордстат.
7. Выполнить адаптивную верстку под разные экраны устройств.
8. Осуществить проверку файлов на валидность.

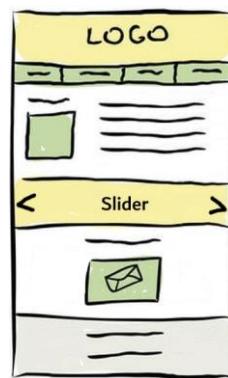
Вариант 1



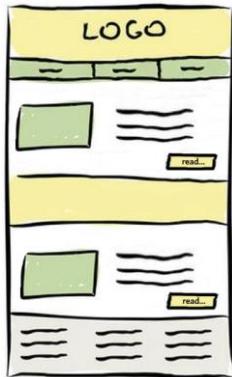
Вариант 2



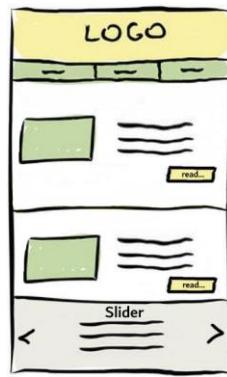
Вариант 3



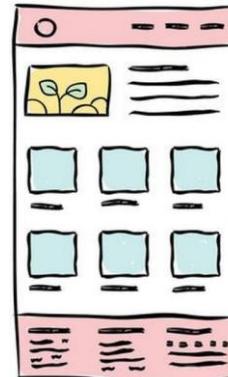
Вариант 4



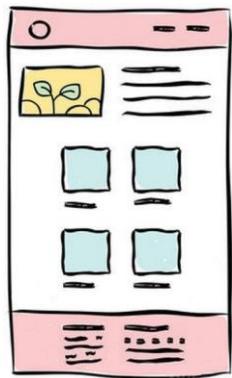
Вариант 5



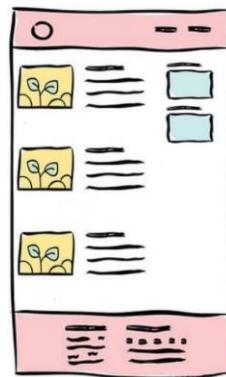
Вариант 6



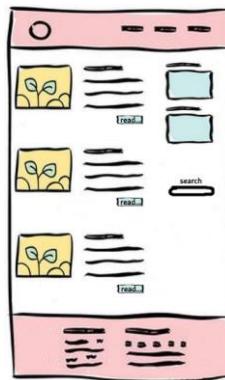
Вариант 7



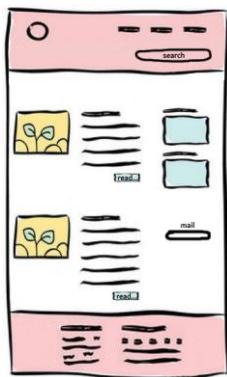
Вариант 8



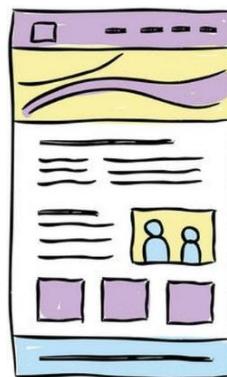
Вариант 9



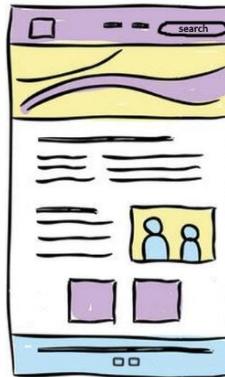
Вариант 10



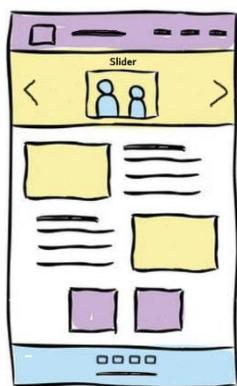
Вариант 11



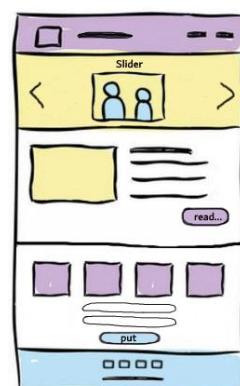
Вариант 12



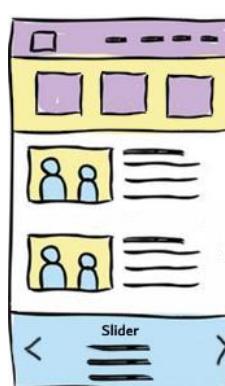
Вариант 13



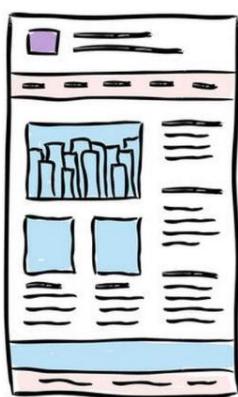
Вариант 14



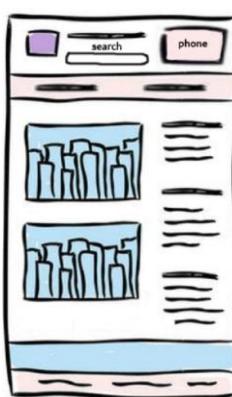
Вариант 15



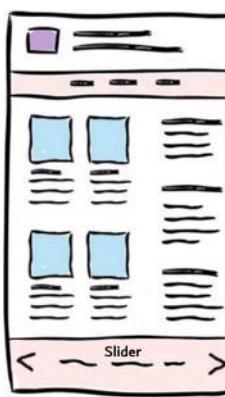
Вариант 16



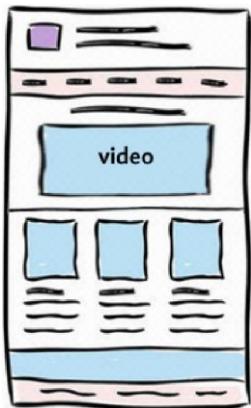
Вариант 17



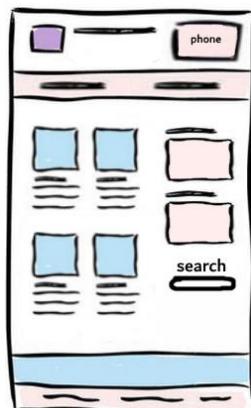
Вариант 18



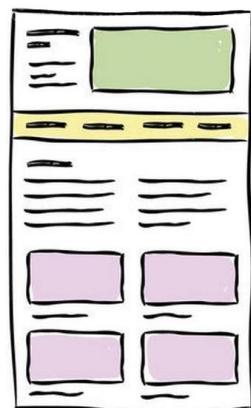
Вариант 19



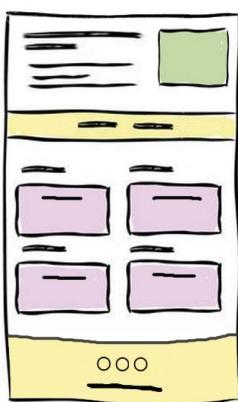
Вариант 20



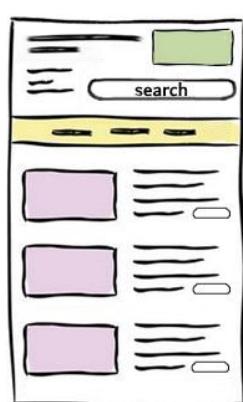
Вариант 21



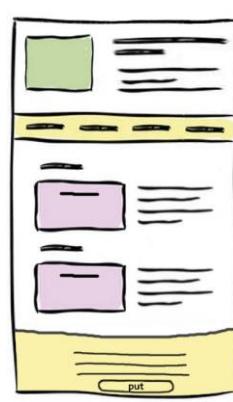
Вариант 22



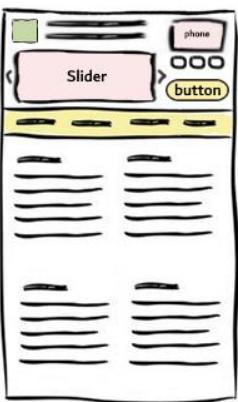
Вариант 23



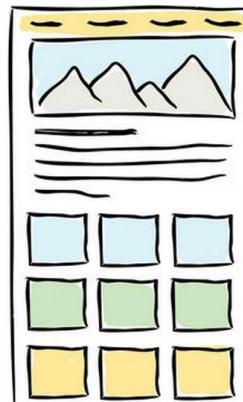
Вариант 24



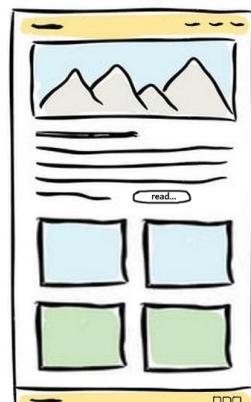
Вариант 25



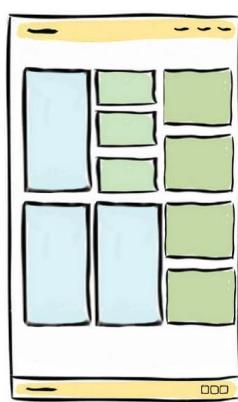
Вариант 26



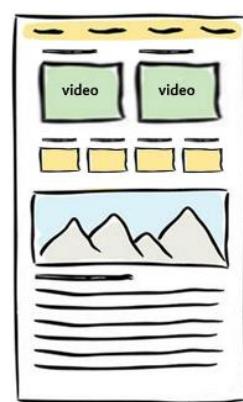
Вариант 27



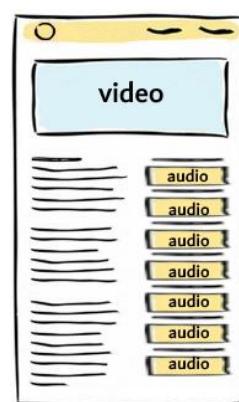
Вариант 28



Вариант 29



Вариант 30



Задача 2

Условие: Необходимо осуществить верстку страницы по макету в **Figma**. Макет выбирается согласно своему варианту (номер по списку в группе).

Вариант 1



[Shop](#)

Вариант 2



[Vso](#)

Вариант 3



[Supabase](#)

Вариант 4



[Qatalog](#)

Вариант 5



[Indicius](#)

Вариант 6



[Porten](#)

Вариант 7



[Beauty](#)

Вариант 8



[C20](#)

Вариант 9



[Yes.](#)

Вариант 10



[Acc.cloud](#)

Вариант 11



[Portfolio](#)

Вариант 12



[By humankind](#)

Вариант 13



[Plant shop](#)

Вариант 14



[Harbor](#)

Вариант 15



[Plain](#)

Вариант 16



[Locus](#)

Вариант 17



[Superbloom](#)

Вариант 18



[Upword](#)

Вариант 19



[Business](#)

Вариант 20



[Dance](#)

Вариант 21



[Melanish](#)

Вариант 22



[Altos ventures](#)

Вариант 23



[CitySpace](#)

Вариант 24



[Proton trade](#)

Вариант 25



[The glass hunts](#)

Вариант 26



[Toy stream](#)

Вариант 27



[Untitled](#)

Вариант 28



[Mafia](#)

Вариант 29



[SEO](#)

Вариант 30



[MYEL](#)