Проектная работа «Сложно сосредоточиться»

https://www.figma.com/design/jCCpF6ClxzMLF5rdpSKx9G/3-спринт.-Проектная-работа?node-id=0-1&t=E4aWbOlCKd0Z5Ewa-0 - ссылка на

проект в figma

В этом проекте перед вами амбициозная задача: сверстать адаптивный сайт в светлой и тёмной темах. Вы не будете начинать с чистого листа. В заготовке проекта уже есть нужная структура со всеми файлами, шрифтами, картинками и скриптом переключения темы.
Вам нужно написать разметку, перенести контент из макета и всё это стилизовать при помощи CSS.
В этом уроке вы ознакомитесь с описанием проекта, в следующем — с чек-листом. Стартовый репозиторий, как обычно, скопируется в ваш GitHub в последнем уроке темы, «Сдача проектной работы». Когда склонируете заготовку на свой компьютер, возвращайтесь в этот урок и приступайте к выполнению проекта.

Шаг 1. Определение подхода и изучение стартового кода

Внимательно изучите макет «Сложно сосредоточиться» .
В начале работы над проектом вам нужно принять решение, какие подходы применять при вёрстке:
Mobile first или Desktop first. В этой работе вы будете использовать Mobile first.

Чёрное или белое. Решите, какую тему верстать

как основную. Это не повлияет на то, какая тема будет включена у пользователя при входе на сайт. Наша рекомендация — начинать с тёмной темы. Можете ориентироваться на тему в вашей операционной системе.

Изучите стартовый код. Стили нужно писать в разных файлах, поэтому посмотрите на имена файлов и почитайте комментарии внутри. Так будет понятно, где какие стили писать.

В HTML уже есть блок с кнопками, которые переключают тему принудительно, вне зависимости от того, какая тема выбрана на компьютере пользователя. Оставьте разметку этого блока как есть, не меняйте классы. Иначе JavaScript-код сломается, и темы не будут переключаться.

В файле `globals.css` сброшены стандартные отступы элементов и задано несколько важных свойств. Вы можете дополнять или менять эти стили по необходимости.

Общие рекомендации перед стартом

Стили для основной темы и основной ширины экрана пишите в файле `style.css`. Ниже в этом же файле вы напишете медиавыражения для адаптива под разные экраны.

При вёрстке старайтесь использовать логические свойства там, где это возможно. Например, `margin-block-start` вместо `margin-top` или `border-inline-end` вместо `border-right` .

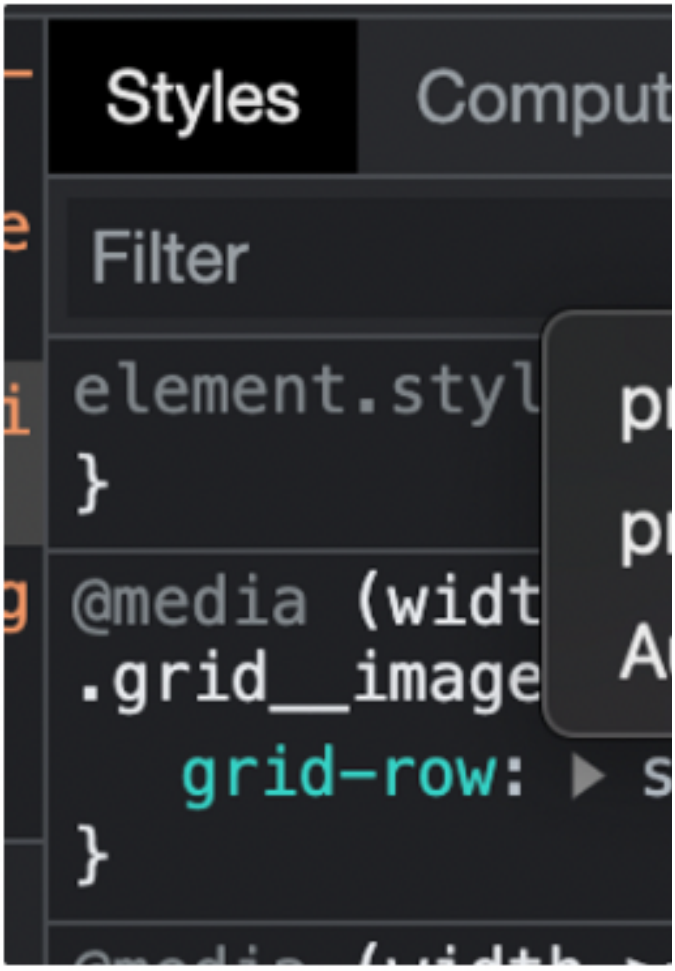
Практически во всей работе для расстановки элементов подойдут гриды. С их помощью

можно адаптировать сайт под разные экраны, практически не меняя стили. Советуем использовать их максимально.

Обязательно убедитесь, что сайт совпадает с макетом на основных разрешениях, а в промежуточных состояниях не ломается и блоки выглядят хорошо. Вспомните про «резиновую» вёрстку.

Отдельно обратите внимание на горизонтальный скролл. Он не должен появляться ни на каких разрешениях и ни при каких условиях.

Держите инструменты разработчика открытыми. В них удобно переключать автоматические темы.



Шаг 2. Подключение файлов и мета страницы

Подключите все нужные файлы стилей в HTML-разметке. Внимательно выбирайте порядок подключения.
Помните про каскадную сущность CSS: чем ниже в коде, тем приоритетнее. Поэтому важно, чтобы более общие стили были подключены раньше менее общих.
Например, при подключении файлов для цветовых тем советуем такой порядок:

Скопировать код

```
<link rel="stylesheet" href="./styles/variables.css">
<link rel="stylesheet" href="./styles/style.css">
<link rel="stylesheet" href="./styles/dark.css">
<link rel="stylesheet" href="./styles/light.css">
```

Для остальных файлов стилей выберите порядок самостоятельно.
Чтобы браузер понял, что у нашего сайта есть тёмная и светлая темы внутри `<head>`, важно прописать строку с метатегом для `color-scheme`. На первом месте укажите ту тему, которую верстаете как основную. Если это тёмная тема, то запись будет такой:

Скопировать код

```
<meta name="color-scheme" content="dark light">
```

После этих действий браузер научится отображать страницу в той теме, которая выбрана в операционной системе. Например, на скринкасте видно, что страница не белая, как обычно, а с тёмным фоном.

После этого подключите фавиконку для страницы, чтобы вкладка выглядела аккуратно.

Шаг 3. Создание статичной версии

Поработайте над шапкой, основным контентом, галереей и футером.

Работа над шапкой

Для расположения элементов в шапке сайта отлично подходят гриды и свойства индивидуального выравнивания грид-элементов.
Если присмотреться к макетам в разных темах, то можно заметить, что в тёмной теме в правом верхнем углу есть небольшая надпись REC. В светлой теме этого декоративного элемента нет.
Пожалуй, это единственный пустой тег в разметке и единственный элемент, к которому уместно применить абсолютное позиционирование.
Точку справа от надписи реализуйте через псевдоэлемент — одного пустого тега в HTML достаточно.
Не забудьте о скринридерах. Поскольку это декоративный элемент, его нужно скрыть при помощи aria-атрибута от читалок.
В правом верхнем и нижнем левом углах есть декоративные уголки. Их можно реализовать при помощи псевдоэлементов и указания границ для двух сторон.
Чтобы заголовок в шапке красиво адаптировался и при этом хорошо выглядел на больших и маленьких экранах, используйте `clamp()`. Ниже вы найдёте формулы расчёта размера текста. Подумайте, почему используются именно такие значения и для какого разрешения применить формулу:

Скопировать код

```
/* Для мобильных */
clamp(7.25rem, 7.0115rem + 1.0178vw, 7.5rem)

/* Для планшета и десктопа */
clamp(7.5rem, 0.5625rem + 14.4531vw, 9.8125rem)
```

Формулы сгенерированы с помощью сайта Font-size Clamp Generator. Он пригодится вам, чтобы адаптировать текст на других разрешениях.
У текста в заголовке задана тень. Вот стили для неё: `text-shadow: 4px 4px 0 var(--accent-color);`.
Параграфу в шапке задан фоновый цвет. Такой же декоративный приём встречается в основном блоке сайта. Используйте отдельный класс `title-decor`, чтобы удобно задавать фон и цвет текста всем схожим элементам.

Работа над основным контентом

Перед тем как верстать основной контент страницы:

Задайте для `.page` фоновое изображение. Путь до фоновой картинки так же можно вынести в переменную, которая будет переопределяться в зависимости от выбранной темы — светлой или тёмной.

Расположите фоновое изображение по центру. Пусть оно закрывает собой всего родителя.

После этого пропишите `background-attachment: fixed;`, чтобы фон оставался на месте при прокрутке страницы.

В основном контенте встречается обычный текст и заголовки разных уровней:

Разбейте весь текст на секции, определите, какого уровня нужны заголовки.

Просмотрите макеты для всех разрешений и примерно прикиньте, какие обёртки вам понадобятся для удобного адаптива этих блоков.

Переиспользуйте класс `title-decor` там, где текст находится на цветном фоне.

Старайтесь задавать похожим элементам одинаковые классы, чтобы писать стили только один раз. Повторений и так хватает.

В тексте встречаются ссылки. Им тоже задана тень. Возьмите значения для свойства `text-shadow` из макета и реализуйте этот стиль по аналогии с заголовком в шапке. Не забудьте про состояния ховера и фокуса для ссылок.

Расставьте цвета, согласно выбранной схеме. Для этого используйте CSS-переменные.

Работа над переменными

Большинство цветов, шрифт, размеры текста и отступы удобно хранить в переменных, потому что они часто используются в разных местах кода. А ещё так можно быстро менять темы простым переопределением переменных.

Все переменные должны храниться только в файле `variables.css`.

Задавая значения для переменных цветов, пишите только нужные для той темы, которую вы выбрали основной. Цвета для второй темы вы добавите позже.

Выбирайте для переменных имена, которые будут иметь смысл в обеих темах. Старайтесь без особой надобности не плодить сущности. Не нужно

создавать для одного цвета две переменные — для светлой и тёмной темы. Создайте одну переменную, а потом, если нужно, поменяйте её значение.

Не надо так:

Скопировать код

```
:root {
    --accent-color-dark-theme: #ff0070;
    --accent-color-light-theme: #ff8dcb;
}
```

Лучше так:

Скопировать код

```
/* Файл variables.css */
:root {
    --accent-color: #ff0070;
}
```

```
/* Файл light.css */
:root {
    --accent-color: #ff8dcb;
}
```

Работа над галереей

В галерее всем картинкам должен быть задан атрибут `alt` с описанием изображения. Поскольку галерея находится сильно ниже первого экрана, который увидит пользователь, вы можете сократить время загрузки страницы и задать для `<img>` атрибут `loading` со значением `lazy`, отвечающим за «ленивую» загрузку.

Сложная разметка здесь не понадобится — дополнительные обертки изображениям не нужны, адаптировать галерею под разные устройства удобно при помощи свойств гридов.

Работа над футером

Используйте гриды и установите отступы.

Расставьте цвета, согласно выбранной схеме. Для этого примените CSS-переменные.

Футер во многом повторяет шапку сайта. Тот же размер, те же декоративные элементы и стили заголовка. Но есть маленький подвох: стили заголовка отличаются от стилей в шапке, пусть и совсем немного. Будьте внимательны.

Помните, что хотя заголовки и выглядят одинаково, они не могут быть одного уровня. На странице должен быть только один заголовок `<h1>`, и его место в шапке, а не в подвале.

Декоративные уголки можно переиспользовать из шапки. Для этого пропишите стили отдельному, независимому от положения в документе классу.

Шаг 4. Добавление адаптивности

Реализуем адаптивность

Создайте медиазапрос для расширения больше 768 пикселей. Он будет определять стили для экранов с шириной больше 768 пикселей.

Создайте медиазапрос для расширения больше 1024 пикселей. Он будет определять стили для экранов с шириной больше 1024 пикселей.

Скорректируйте правила для нужных элементов согласно макету.

Добавляем вторую цветовую схему

В файле второй цветовой схемы установите CSS-переменные и цвета.

Скорректируйте отображение элементов, которые меняются в этой теме:

- фоновую картинку страницы;
- изображение блока «Запись»;
- стиль кнопки в шапке.

Шаг 5. Работа над кнопками переключения тем

Разметка этого блока уже готова. При нажатии кнопок запускается JavaScript-код, который принудительно меняет темы сайта. Как это работает?

Кнопка «День»

При нажатии на кнопку для корневого тега `<html>` добавляется класс `theme-light`. При этом должна отобразиться светлая тема.

Кнопка «Неон»

При нажатии на кнопку для корневого тега `<html>` добавляется класс `theme-dark`. При этом должна отобразиться тёмная тема.

Кнопка «Авто»

При нажатии на кнопку для корневого тега `<html>` добавляется класс `theme-auto`. При этом должна отобразиться та тема, которая выбрана у пользователя в системе. Сработает медиафича `prefers-color-scheme`.
Для нажатой кнопки устанавливается класс `.header__theme-menu-button_active`. Кнопка должна стать недоступна для клика. Поможет свойство `pointer-events: none`.
Нужно стилизовать состояния кнопок по ховеру и по фокусу. Для стилизации фокуса используйте `:focus-visible`. Стили всех состояний можно найти в макете.
Из-за того, что темы на сайте будут переключаться не только в зависимости от настроек пользователя, но и принудительно, по классу, при написании стилей придётся дублировать стили тем. Один раз для класса, а второй раз для медиафичи `prefers-color-scheme`.
Предположим, вы выбрали основной тёмную тему. Опишем структуру кода внутри файлов стилей.
Файл `variables.css`:

Скопировать код

```
:root {
  /* Переменные тёмной темы */
}
```

Файл `dark.css`:

Скопировать код

```
:root.theme-dark {
  /* Дублируем переменные для тёмной темы */
```

```
}

/* Остальные стили, только при необходимости */
```

Файл `light.css`:

Скопировать код

```
:root.theme-light {
  /* Переменные светлой темы */
}

/* Остальные стили, только при необходимости */

@media (prefers-color-scheme: light) {
  :root {
    /* Дублируем переменные светлой темы */
  }
}

/* Остальные стили, только при необходимости */
```

Для тёмной темы не нужно использовать правило `@media (prefers-color-scheme: dark) {}`, поскольку она выставлена по умолчанию, и её переменные уже заданы в `variables.css`. Они будут применены автоматически.

Помните о рефакторинге

Запланируйте время на рефакторинг. После того, как закончите, отложите код хотя бы на один день. Затем вернитесь и посмотрите, можно ли где-то сделать проще и изящнее. Может быть, вы увидите, что использовали лишние обёртки или слишком длинные записи для стилей.
Такой подход поможет выработать критический взгляд на свою работу — и сделать её лучше.

Чек-лист

В чек-листе собраны требования, которые нужно соблюсти, чтобы пройти автоматическое тестирование и профессиональное ревью.

Работа отклоняется от проверки ревьюером

Пока эти ошибки не будут исправлены, ревьюер не примет работу на проверку и не даст обратную связь:

Пять или больше элементов макета отсутствуют.

Не хватает секции.

Порядок блоков не соответствует брифу.

На повторных итерациях не исправлены критические замечания.

Работа содержит вопросы или просьбы о помощи к ревьюеру.

Критические требования

Без соблюдения этих требований ваш проект не пройдёт автоматическое тестирование. Если по какой-то причине автотесты пропустят работу (такое может случиться), эти пункты обязательно попросит исправить ревьюер.

Отображение в браузере

Есть все секции, блоки и элементы макета, и они

корректно отображаются в Firefox, Google Chrome или Yandex Browser.

Проект визуально соответствует макету на всех заданных размерах экрана:

- отличие не более 10% при проверке системой скриншот-проверки автотестов,
- отличие не более `5px` по вертикали и `3px` по горизонтали при проверке ревьюером.

Рекомендуем пользоваться плагином Pixel Perfect для доводки вёрстки.

Сетка макета не сбивается и вёрстка не «ломается» между брейкпоинтами, например:

- при сжатии страницы сохраняется расположение элементов относительно друг друга на странице;
- содержимое не выпадает за пределы своего блока;
- блоки не наезжают друг на друга;
- изображения не искажаются и выглядят так же, как в макете, на всех размерах экрана.

На всех размерах экрана, в том числе между брейкпоинтами, страница не прокручивается по горизонтали. Не появляется лишних полос прокрутки при ширине экрана, которая больше или равна минимально допустимой.

Свёрстаны все состояния элементов страницы, описанные в рекомендациях и макете.

Рекомендации к текущему проекту

Разметка кнопок переключения цветовой схемы

взята из стартового кода без изменений.

Цветовая схема переключается.

- Переключение цветовых схем реализовано корректно:
  - поддерживаемые цветовые схемы описаны в метатеге, например `<meta name="color-scheme" content="dark light">`;
  - для смены цветовых схем используется подход через задание класса контейнеру страницы и медиавыражение с `prefers-color-scheme`;
  - цветовую схему меняют все нужные элементы;
  - фон соответствует выбранной цветовой схеме и задаётся через стили тега `body` всей странице, а не только контенту;
  - цвета, которые использованы в цветовых схемах, вынесены в CSS-переменные;
  - оформление цветовых схем описано в соответствующих файлах: `dark.css` и `light.css`.
- Блоки `header` и `footer` имеют высоту равную высоте экрана (`vh`), также им задана минимально допустимая высота (`px`).
- При отображении светлой цветовой схемы в блоке `header` не должен отображаться элемент REC.

Структура проекта

В проекте отсутствуют пустые файлы.

Есть файл `index.html`.

Файловая структура проекта единообразна.
Например: стили находятся в папке `styles`, картинки в папке `images`, шрифты в папке `fonts`, аналогично для других типов файлов.

HTML

Аккуратная разметка HTML. Предлагаем использовать Prettier для автоформатирования кода.

Валидный HTML, например:
нет незакрытых тегов, которые нужно закрывать;

списки свёрстаны правильно, внутри тегов списка находятся только пункты списка;

нет ошибок вложенности.
Валидность HTML можно проверить в этом валидаторе .
Стили подключены в правильном порядке: сначала шрифты, затем глобальные стили, затем собственные.

Язык страницы указан корректно.

Задано подходящее значение `title` .

Используется семантическая разметка:

теги `<header>` , `<main>` , `<footer>` , `<section>` , `<nav>` использованы по назначению. Ими выделены соответствующие секции страницы;

для вёрстки заголовков применяются теги от `<h1>` до `<h6>`, текстовые блоки размечены

тегами `<p>`, списки — `<ul>` и `<li>`;

есть многоуровневость заголовков;

есть единственный заголовок первого уровня;

для блоков, содержащих информацию, используются соответствующие ей смысловые семантические теги, например, `<article>` , `<address>` , `<time>` или другие при необходимости;

ссылки на номер телефона и почту, если они есть на странице, снабжены префиксами в значении атрибутов `href`.

Для обозначения абзаца не используется тег переноса строки.

У всех изображений задан атрибут `alt` с описанием на языке страницы.

CSS

Аккуратное форматирование кода:

между селекторами и открывающими скобками стоит пробел,

каждое правило начинается с новой строки,

стоят точки с запятой,

закрывающие скобки вынесены на отдельную строку.

Соблюдены требования к именованию CSS классов:

для CSS классов выбраны подходящие по смыслу имена;

отсутствует слитное написание слов в именах классов;

единообразное разделение слов в именах CSS-классов во всём проекте, например, используется только `kebab-case` — разделение слов знаком - или `camelCase` — разделение слов регистром символов;

в именах не используется транслитерация и сокращения, которые не являются общеупотребимыми.

Если в брифе к проекту описаны имена классов, то старайтесь использовать их.

У `body` и типовых элементов сбрасываются браузерные отступы.

Для всех элементов сайта корректно заданы:

цвета фона,

размеры,

межстрочные расстояния,

межбуквенные расстояния и расстояния между словами при необходимости.

Корректно используются шрифты:

разные форматы шрифтов подключаются в правильном порядке — от самых современных к более старым с указанием формата как свойства CSS;

указаны альтернативные шрифты;

семейство, вес, начертание и размеры шрифтов во всех элементах страницы соответствуют

заданным в макете.

Правильно организован лейаут страницы:

для организации лейаута ключевых блоков использован `flex` или `grid`;

корректно отцентрированы необходимые элементы на странице;

не задана фиксированная высота и ширина элементов там, где их можно не использовать или применены минимальные или максимальные значения. Блок растягивается, если в него вставлено в два-три раза больше контента;

абсолютное позиционирование используется только там, где нельзя применить статичное или относительное позиционирование;

контекст позиционирования указан корректно (например, `position: relative` у нужного элемента);

корректно задано свойство `z-index`, нет спрятавшихся элементов.

Корректно реализована адаптивная верстка:

установлена максимальная и минимальная ширина контента в соответствии с макетом;

брейкпоинты сгруппированы. Если два брейкпоинта имеют небольшую пиксельную разницу в медиаправиле, они объединены в один;

между брейкпоинтами используется «резиновая» вёрстка;

правильно определены «резиновые» и статические размеры;

одинаковые свойства в разных медиаправилах не дублируются.

Нет пустых CSS-правил.

Нет дублирующихся селекторов.

Нет дублирующихся свойств внутри CSS-правил.

CSS-переменные корректно заданы и переопределены, используются запасные значения и имеют информативные названия.

Стили одинаковых элементов не дублируются, а переиспользуются.

Не используются инлайновые стили в HTML.

Хорошие практики

В этом списке собраны приёмы, которые помогут сделать проект ещё лучше.
На данном этапе их использовать необязательно — автотесты и ревьюер примут работу без них.
Однако такие хорошие практики помогут вам сделать портфолио более профессиональным.
Опытные верстальщики обратят на них внимание.

В разметке нет лишних обёрток `<div>` у элементов.

В разметке у ссылок нет пустых атрибутов `href` .

Всем изображениям заданы ограничения размеров.

Для изображений настроена «ленивая» загрузка.

Элементам не добавлены классы, которые не используются.

В стилях не указаны неработающие свойства (например, не заданы размеры строчных элементов).

Для всех элементов на странице переопределено дефолтное значение свойства `box-sizing`.

При организации внутренней геометрии блоков внутренние отступы задаются через `padding`.

Нет лишних отступов у первых и последних элементов в списках и блоках с абзацами;

Раскладки `flex` или `grid` не применяются без необходимости там, где можно обойтись дефолтным отображением.

Для задания отступов между элементами `flex` и `grid` блоков использован `gap` вместо `margin`.

Нет чрезмерной стилизации. Например, одинаковые размеры прописаны и для обёртки, и для изображения внутри.

Интерлиньяж задан в относительных единицах измерения.

Если в проекте используется методология БЭМ, то нет нарушений соглашения по именованию классов и требований методологии. Например:

применено единообразное разделение имён блоков, элементов и модификаторов во всем проекте;

элемент не используется вне своего блока;

нет элементов элементов;

класс модификатор не используется без указания блока или элемента, который он модифицирует;

классы модификаторы описывают только модифицированное значение, остальные описаны в стилях блока;

внешняя геометрия блоков задана через миксы;
В элементах формы при необходимости используются `inherit` или `currentColor` .

Для предсказуемого поведения инпутов в разных браузерах использован `appearance: none;` .

Вместо `overflow: scroll` использован `overflow: auto`, чтобы не отображался ненужный скролл.

Вид курсора меняется при наведении на интерактивные элементы.

Для наилучшего позиционирования изображения при задании свойства `object-fit` задаётся также подходящее значение свойства `object-position` .

Для стилизации интерактивных элементов используется `outline`, а не `border` (`outline` не влияет на размеры элемента в потоке).

CSS-правила в коде расположены в примерном соответствии с позицией элементов в разметке.

Для стилизации используются только селекторы

классов, псевдоэлементов и псевдоклассов, если речь не идёт о сбросе браузерных дефолтных стилей.

Вместо физических CSS-свойств используются логические CSS-свойства; единообразие в типе свойств.

Для отправки декоративных линий и фонов назад используется `z-index: -1`.

В медиавыражениях под ширины экрана используется современный синтаксис задания диапазонов.

Для реализации лейаута максимально используется сетка, построенная на `grid`, и её возможности.

Для создания декоративных элементов используются псевдоэлементы, а не дополнительные теги в разметке.

Для элементов разметки, не несущих смысловой нагрузки, задаётся атрибут `aria-hidden="true"`.

Ссылки на внешние ресурсы открываются в новом окне.

У ссылок сброшен `outline` в `:focus` и заданы стили `:focus-visible`.

Используются разные форматы и размеры изображений в элементе `<picture>`.