



HTML и CSS. Адаптивная вёрстка и автоматизация

Уровень 2, с 21 ноября 2022 по 30 января 2023

Меню курса

[Главная](#) /

Критерии

Подготовка и проверка личных проектов проводится по базовым и дополнительным критериям.

Базовые критерии охватывают наиболее важные требования к проекту и проверяют основные знания и навыки. Для успешной защиты личного проекта должны быть выполнены все базовые критерии.

Дополнительные критерии проверяют то, насколько студент внимателен к деталям, и оценивают проект с точки зрения шлифовки его качества и оптимизации. Выполнение этих критериев необходимо для защиты на 100%.

Во время финальной защиты баллы за выполнение дополнительных критериев добавляются только при выполнении всех базовых.

Для подачи проекта на защиту нужно сверстать мобильное, планшетное и десктопное состояния трёх страниц проекта.

Дополнительные



Разметка [Свернуть все](#)

- ▲ Д1. У всех векторных изображений размер прописан в теге ``, у встроенных SVG-изображений размер прописан в теге `<svg>`.

В HTML-коде должны быть указаны ширина и высота **векторных** картинок, так как векторные картинки могут растянуться на всю ширину родителя.

Важно: в атрибутах `width` и `height` нельзя использовать `px`, так как по спецификации размеры в пикселях задаются без единиц измерения. Проце

использовать можно, если картинка должна тянуться. Размеры должны быть указаны

не использовать floats, если картинка должна быть встроена, а размеры должны быть указаны в виде целого числа — `100`, `213`, а не дробного — `100.5`, `213.25`.

Для адаптивных изображений, у которых есть версии с разными размерами и пропорциями (например, заданными через тег `sources` в элементе `picture`), размеры в `img` в HTML можно не задавать.

Верно: картинке заданы целочисленные размеры.

```
<div class="page-header__logo">
  
</div>
```

Неверно: картинке не заданы размеры.

```
<div class="page-header__logo">
  
</div>
```

Неверно: картинке заданы размеры в виде дробного числа.

```
<div class="page-header__logo">
  
</div>
```

Неверно: картинке заданы размеры с использованием `px`.

```
<div class="page-header__logo">
  
</div>
```

Верно: тегу `svg` заданы размеры.

```
<svg width="200" height="200">
  <rect x="0" y="0" width="200" height="200" fill="#FAFAA2" stroke="#000"/>
</svg>
```

Верно: адаптивному изображению может быть не задан размер в HTML

— Сервис адаптирует изображение под размер экрана, не создавая растровый HTML-код.

```
<picture>
  <source media="(min-width: 1150px)" srcset="img/logo-desktop.svg">
  <source media="(min-width: 768px)" srcset="img/logo-tablet.svg">
  
</picture>
```

- ^ Д2. Использовано минимально возможное количество HTML-элементов (нет лишних элементов).

В разметке должно быть использовано минимально возможное количество элементов. Не должно быть лишних обёрток и блоков, которые используются для оформления и могут быть заменены на псевдоэлементы.

Стилизация [Свернуть все](#)

- ^ Д4. Для стилизации не использованы `#id`.

Для стилизации объектов лучше использовать селекторы по классам или тегам. Использовать `id` для стилизации недопустимо.

Важно: использовать атрибут `id` в HTML-разметке можно и нужно. Например, для привязки полей к подписям в формах.

Верно: стилизация элементов через классы.

```
.main-nav {}
.page-footer {}
```

Неверно: стилизация элементов строится через идентификаторы.

```
#main-nav {}  
#page-footer {}
```

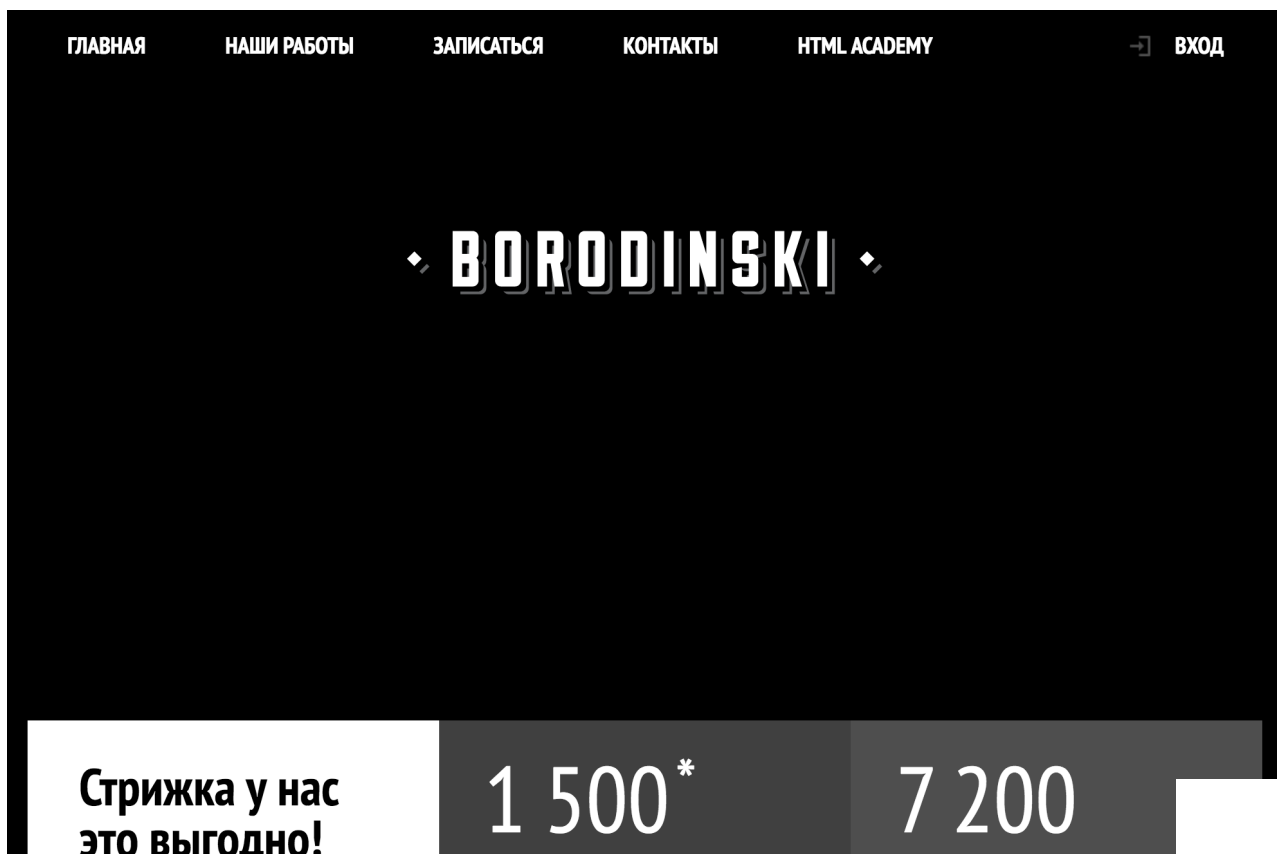
- ^ Д6. Для блока, у которого есть фоновое изображение, прописан фоновый цвет, который соответствует преобладающему цвету изображения (пока изображение не загружено, страница выглядит похоже на макет).

Такой метод использует для перестраховки, как и в случае со шрифтом. Только в этом случае, если фоновое изображение не загрузится, на заднем фоне останется преобладающий цвет.

Верно: указан цвет фона перед основным изображением.

```
body {  
  background: #000000 url("../img/background-body.jpg") no-repeat center  
  top;  
}
```

Результат: картинка не загрузилась, но при этом цвет фона остался тёмным.



Неверно: указано только основное изображение

...вернуть, указав только основные изображения.

```
body {  
  background: url("../img/background-body.jpg") no-repeat center top;  
}
```

Результат: картинка не загрузилась, а цвет фона стал белым.

[ГЛАВНАЯ](#)[НАШИ РАБОТЫ](#)[ЗАПИСАТЬСЯ](#)[КОНТАКТЫ](#)[HTML ACADEMY](#)[→ ВХОД](#)

· ВОРОТНИКИ ·

**Стрижка у нас
это выгодно!**

1 500*

7 200

^ Д7. Все состояния элементов (смотрите стайлгайд) прописаны в стилевом файле.

В соответствии со стайлгайдом, который присутствует в каждом проекте, все указанные в нём элементы должны иметь соответствующие эффекты при наведении и нажатии. Должны присутствовать все активные состояния, а для чекбоксов и радиокнопок прописаны состояния `disabled`. Если в стайлгайде не предусмотрено какое-то состояние, то его реализация остаётся на усмотрение студента.

^ Д16. Нет глобальных стилей тегов.

Для задания стилей используются только селекторы по классам или вложенные селекторы.

Исключения:

- Normalize.css, который исправляет браузерные умолчания
- Уникальные теги документа: `<html>`, `<body>`
- Дополнительная нормализация: `<a>` и ``

Верно:

```
body {  
  margin: 0  
}  
  
img {  
  max-width: 100%;  
  height: auto;  
  object-fit: contain;  
}  
  
a {  
  text-decoration: none  
}  
  
.feedback { color: black }  
.feedback ul { list-style: none }  
.feedback p { margin-bottom: 0 }  
  
*, *::before, *::after {  
  box-sizing: border-box;  
}
```

Неверно:

```
ul { list-style: none }  
p { margin: 0 }
```

```
div { font-size: 80% }
```

```
... { font-size: 100%; }  
* { box-sizing: border-box }
```

CSS-препроцессор [Свернуть все](#)

- ^ **Д8.** Запрещено использовать цветовые функции для изменения цветовых значений в коде.

Если в макете указаны конкретные цвета, нужно задавать конкретные цветовые значения в коде. Не нужно подбирать эти значения с помощью цветовых функций CSS-препроцессоров (изменение цвета, осветление, затемнение и так далее).

Исключение составляют функции изменения альфа-канала (прозрачности цвета) — их использовать можно. Это функция `rgba()` в Less и функции `opacity()` и `transparentize()` в Sass.

- ^ **Д9.** Примеси не используются для генерации правил с вендорными префиксами.

Этот критерий запрещает использовать миксины для генерации правил с вендорными префиксами. Например:

```
.box-sizing {  
  -webkit-box-sizing: border-box;  
  -moz-box-sizing: border-box;  
  box-sizing: border-box;  
}  
  
.class {  
  .box-sizing();  
}
```

Использование таких миксинов нецелесообразно, так как в проекте используется Автопрефиксер, который расставляет вендорные префиксы самостоятельно.

- ^ **Д10.** Вложенность селекторов не больше двух уровней.

Вложенность усложняет читаемость кода и делает код хрупким. Хорошим подходом считается вложенность не больше двух уровней. Вложенность считается по CSS на выходе.

Плохо: вложенность на три уровня.

```
.block {  
  .title {  
    span {  
      font-weight: bold;  
    }  
  }  
}
```



```
.block .title span {  
  font-weight: bold;  
}
```

Комбинаторы `+` и `~` вложенность не увеличивают.

Хорошо: вложенность на два уровня с комбинатором.

```
.block .one + .two {  
  font-weight: bold;  
}
```

Вложенные селекторы в препроцессорах вложенность не увеличивают, если в результирующем CSS они остаются простыми.

Хорошо: результирующий селектор остаётся простым.

```
.block {  
  &--mod {  
    &::before {  
      font-weight: bold;  
    }  
  }  
}
```



```
.block--mod::before {
```



```
font-weight: bold;
}
```

- ^ Д11. Родительский селектор `&` используется только для псевдоэлементов, псевдоклассов и модификаторов.

Использование `&` для комбинации селекторов не допускается в именах блоков и элементов. Комбинировать можно только псевдоэлементы, псевдоклассы и модификаторы блоков и элементов.

Плохо: комбинируется имя элемента.

```
.block {
  &__element {
    color: tomato;
  }
}
```

Хорошо: комбинируется псевдоэлемент и модификатор элемента.

```
.block__element {
  &::before {
    color: tomato;
  }
  &--mod {
    color: tomato;
  }

  /* для классического стиля */
  &_mod_value {
    color: tomato;
  }
}
```

- ^ Д12. Не используются расширения (`extend`).

Функции `&:extend` в Less и `@extend` в Sass запрещены.

Использование расширений приводит к неочевидной трансформации кода в стилях на выходе. Это усложняет отладку и может привести к генерации неоптимальных групп селекторов.

Невинная задача по абстрагированию размера шрифта в расширение:

```
%basic-font-size {
  font-size: 16px;
}

.block-1 {
  @extend %basic-font-size;
}
...
.block-9999 {
  @extend %basic-font-size;
}
```

...может привести к генерации мегабайтов селекторов в выходном файле:

```
.block-1, .block-2, .block-3, .block-4, ..., .block-9999 {
  font-size: 16px;
}
```

Тестирование [Свернуть все](#)

^ Д13. Вёрстка проходит тест на переполнение контентом.

— Не ломается при добавлении в элементы большего количества текста.

Верно: текст растягивает блок.



БЫСТРО

Мы делаем свою работу быстро! Два часа пролетят незаметно и вы — счастливый обладатель стильной стрижки-минутки. Мы делаем свою работу быстро! Два часа пролетят незаметно и вы — счастливый обладатель стильной стрижки-минутки. Мы делаем свою работу быстро! Два часа пролетят незаметно и вы — счастливый обладатель стильной стрижки-минутки.



КРУТО

Забудьте, как вы стриглись раньше. Мы сделаем из вас звезду футбола или кино. Во всяком случае внешне.



ДОРОГО

Наши мастера — профессионалы своего дела и не могут стоять дёшево. К тому же, разве цена не даёт определённый статус?

НОВОСТИ И АКЦИИ

[ПОКАЗАТЬ ВСЕ](#)

Неверно: текст выпадает за пределы блока с тёмным фоном.



БЫСТРО

Мы делаем свою работу быстро! Два часа пролетят незаметно и вы — счастливый обладатель стильной стрижки-минутки. Мы делаем свою работу быстро! Два часа пролетят незаметно и вы — счастливый обладатель стильной

стрижки-минутки. Мы делаем свою работу быстро! Два часа пролетят незаметно и вы —



КРУТО

Забудьте, как вы стриглись раньше. Мы сделаем из вас звезду футбола или кино. Во всяком случае внешне.



ДОРОГО




Наши мастера — профессионалы своего дела и не могут стоять дёшево. К тому же, разве цена не даёт определённый статус?

НОВОСТИ И АКЦИИ

[ПОКАЗАТЬ ВСЕ](#)

— Не ломается при использовании картинок с неподходящими размерами.

Верно: добавление картинки не сломало сетку, она вписалась в соответствующий размер.

 <p>Саша Мальцев</p> <p>Попросил омолодить и омолодили! Кто теперь скажет, ...</p>	 <p>Владимир Иванович</p> <p>К зимнему сезону – готов!</p>	 <p>Винстон Синий</p> <p>Как только заростаю и волосы начинают мешать видеть – бегом ...</p>
---	---	---

Неверно: добавление большей картинки сломало сетку.

- Не ломается при изменении количества потоковых блоков: текст не выпадает из блоков, нижерасположенные блоки не скрываются, смещение блоков в потоке сохраняет логику потока (не приводит к нарушению сетки).

Верно: переполнение верхнего блока растягивает его и двигает нижний.

Неверно: переполнение верхнего блока приводит к тому, что блоки скрываются под следующими элементами.

^ Д14. Критическая функциональность сайта работоспособна без JavaScript (использовано прогрессивное улучшение).

Например:

- Все формы являются работоспособными без JavaScript.
- Элементы, вызывающие появление попапов, являются ссылками, ведущими на отдельные страницы: достаточно указать адрес на страницу, на которую будет происходить переход в случае неработоспособности JavaScript, при этом верстать саму страницу необязательно. Пример:

```
<a class="btn btn-open-form" href="form.html">Открыть форму</a>
```

- Интерактивная карта без JavaScript показывает статичную картинку с картой.
- Мобильное меню по умолчанию открыто.

Доступность [Свернуть все](#)

- ^ **Д17.** У интерактивных элементов при нажатии или фокусе с клавиатуры есть активное состояние.

Активное состояние интерактивных элементов при нажатии или фокусе с клавиатуры должно оставаться либо встроенным браузерным, либо быть равноценно переназначено. В таком случае по интерактивным элементам сайта можно передвигаться с клавиатуры клавишей `Tab` и видеть каждый текущий элемент в активном состоянии.

Допускается делать недоступные для скринридеров элементы, если равносильная функциональность или контент уже доступны. Например, если есть ссылка на страницу авторизации, модальное окно можно спрятать с помощью `display: none` или если есть простое поле для ввода даты, вспомогательный попап с календарём тоже можно спрятать недоступно. Однако интерактивные элементы и контент по-прежнему должны быть доступны с клавиатуры.

- ^ **Д18.** Все интерактивные элементы имеют текстовое описание.

Интерактивные элементы, представленные на макете только графически, без текста, содержат текстовое описание. Это поможет пользователям понять, что именно произойдёт в результате взаимодействия, в случае, если они не видят элементы интерфейса. Это описание будет озвучено скринридером.

К интерактивным элементам относятся все элементы страницы, с которыми пользователь взаимодействует: кликом, наведением, фокусом, кнопками клавиатуры. Например: ссылки, кнопки, поля ввода и другие элементы форм.

Описание можно добавить с помощью скрытых HTML-элементов, либо с помощью специальных атрибутов `aria-label`. Обратите внимание: атрибут `placeholder` не предназначен для описания полей ввода, он приводит пример заполнения

ланных. В этом случае предпочтительнее будет добавить скрытый связанный

данных в `status` tag, предполагается, будет добавлен скрытый элемент `<label>` с описанием поля.

Форматирование и внешний вид [Свернуть все](#)

^ [Д19](#). Код соответствует правилам в EditorConfig

При выполнении автоматической проверки и консольной команды `npm run editorconfig` в корневой папке проекта не возникает ошибок.

^ [Д20](#). Код соответствует правилам в Stylelint.

При выполнении автоматической проверки и консольной команды `npm run stylelint` в корневой папке проекта не возникает ошибок.

Оптимизация [Свернуть все](#)

^ [Д21](#). Используются изображения в формате WebP.

Для браузеров, поддерживающих формат `WebP`, контентные изображения подключаются в этом формате. Для браузеров, не поддерживающих `WebP`, изображения подключаются в формате `jpeg` или `png`.

^ [Д22](#). Использован векторный спрайт.

Векторные изображения на странице объединены в спрайт.

^ [Д23](#). Произведена оптимизация загрузки шрифтов.

Шрифты предзагружаются через `link rel="preload"`.

Используется подходящее значение `font-display` в описании `@font-face`

используемый подпадающее под тип `code-block` в отладчике `code-view`.

Разное

[Д15](#). При взаимодействии с элементами (наведение, нажатие) ни сам элемент, ни окружающие его блоки не меняют своего положения (если иное не прописано в техническом задании или стайлгайде).

Поиск по материалам

Git

[Все материалы](#)

В самом начале



- ☐ [Пройдите опрос](#)
- ☐ [Укажите персональные данные](#)
- ☐ [Изучите регламент](#)
- ☐ [Прочитайте FAQ](#)
- ☐ [Добавьте свой Гитхаб](#)
- ☐ [Выберите наставника](#)
- ☐ [Создайте проект](#)

Мой наставник



[Выбрать наставника](#)

Работа с наставником

У вас осталось **10** из 10 консультаций.

[История](#)



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Информация

Об Академии

О центре карьеры

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Услуги

Работа наставником

Для учителей

Стать автором

Остальное

Написать нам

Мероприятия

Форум

