МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по выполнению лабораторных работ
по дисциплине «Информационные технологии и программирование»
для студентов направления 09.03.04 Программная инженерия
(направленность(профиль) «Разработка и сопровождение
программного обеспечения»)

Ставрополь
2026

Лабораторная работа 1. HTML, CSS

Цель работы: сформировать навыки работы с HTML, CSS.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 1, тема 1-2, уроки 1-24.

Изучите следующие разделы: Что делают разработчики? Базовые элементы HTML. Мысли о стилях CSS. Сценарное мастерство JavaScript. Трюки и финты. Нужен плейлист. Теги HTML. Заголовки. Абзац. Ссылка. Об атрибутах. Один тег в другом: Родители и дети. Изображения. Структура HTML-документа. Страница рассказывает о себе. Введение в стили. Правила CSS. Связь CSS и HTML: Тег <style>. Связь CSS и HTML: CSS-файл. Закрепляем навыки. Блоки. Отступы. Подпись к обложке. Playground.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте простейшую HTML-страницу. Внутри тела документа (<body>) используйте теги <h1>, <p> и <h2>. Напишите в <h1> свое имя, в <p> — несколько слов о себе (например, ваше хобби), а в <h2> — название вашего учебного заведения. Подсказка: Не забудьте про базовую структуру (<!DOCTYPE html>, <html>, <head>, <body>).

2	Создайте страницу о воображаемой или реальной поездке. Используйте один <h1> для названия страны, несколько <h2> для названий городов и теги <p> под каждым заголовком для описания впечатлений от каждого города.
3	Создайте страницу со списком ваших 5 любимых веб-сайтов (например, музыкальных групп, новостных порталов). Оформите каждый пункт списка как ссылку с помощью тега <a>. Обязательно используйте атрибут href для указания адреса и атрибут target="_blank", чтобы ссылка открывалась в новой вкладке.
4	Создайте структуру с помощью тега . Внутрь первого div поместите два других. Внутри каждого из этих двух — еще по одному. В каждый из самых внутренних div вложите по одному тегу <p> с текстом "Потомок". Прокомментируйте в коде, кто кому является родителем, братом или потомком.
5	Найдите в интернете 3 изображения (например, пейзаж, архитектура, животное), которые вас вдохновляют. Добавьте их на страницу с помощью тега <img>. Используйте атрибут width или height, чтобы задать всем изображениям одинаковую ширину (например, 300px). Не забудьте про атрибут alt для описания картинки!
6	Создайте список из 3-5 своих любимых музыкальных треков. Для каждого трека сделайте следующее: разместите его обложку (изображение), сделайте название трека кликабельной ссылкой на этот трек на YouTube или Spotify. Используйте теги <img>, <a> и параграфы.
7	Создайте простую HTML-страницу о вашем любимом времени года. Добавьте заголовок <h1> с названием времени года и два абзаца <p> с его описанием. В раздел <head> документа добавьте тег <style> и пропишите в нём правила: – Для всего тела страницы (body) задайте приятный фоновый цвет (например, lightcyan для зимы или honeydew для лета). – Для заголовка <h1> задайте цвет текста, который будет хорошо читаться на фоне (например, darkblue или darkgreen). – Для всех абзацев <p> задайте другой цвет текста (например, darkslategray).
8	Создайте HTML-код с заголовком и абзацем. В том же теге <style> измените внешний вид текста с помощью новых свойств: – Для заголовка <h1> задайте выравнивание по центру (text-align: center), измените шрифт на Arial или Georgia (font-family) и превратите текст в заглавные буквы (text-transform: uppercase). – Для всех абзацев задайте размер шрифта 18px (font-size) и увеличьте высоту строки для удобства чтения (line-height: 1.5).
9	Создайте три с разными текстами внутри (например, "Новость 1", "Новость 2", "Новость 3"). Задайте каждому div с помощью CSS: – Фон цвета на ваш выбор (background-color: lightblue). – Внутренний отступ (padding: 20px).

	– Внешний отступ снизу (margin-bottom: 15px), чтобы разделить блоки между собой.
10	– Создайте новую папку для проекта. Внутри создайте два файла: index.html и styles.css. – В файл index.html напишите базовую структуру HTML-документа. Добавьте в <body> заголовок <h2> с вашим именем и абзац <p> с вашей будущей профессией. – В файл styles.css перенесите все стили из предыдущих заданий или создайте новые (например, задайте цвет фона, стили для текста). – Самое главное: В разделе <head> файла index.html с помощью тега <link> подключите внешний файл стилей styles.css. Убедитесь, что стили применились к вашей странице.
11	С помощью 9 тегов создайте сетку 3x3 (три строки по три блока). Задайте каждому блоку одинаковые ширину, высоту и цвет фона. Используйте свойство margin, чтобы создать расстояние между блоками. Добейтесь того, чтобы блоки выглядели как упорядоченная сетка.
12	Добавьте изображение-обложку. С помощью CSS поместите поверх изображения текст (например, название альбома). Для этого поместите изображение и текст в один контейнер ( ), а тексту задайте абсолютное позиционирование (position: absolute) относительно этого контейнера.
13	Создайте новый чистый файл. Ваша задача — создать один и стилизовать его всеми известными вам способами. Используйте: width, height, background-color, margin, padding, border (сделайте рамку пунктирной или толстой), border-radius (скруглите углы). Цвета и размеры выбирайте любые. Главное — посмотреть, как каждое свойство меняет блок.
14	В комментариях внутри кода вашего HTML-документа (между <!-- и -->) "от лица" каждого основного тега напишите, какую роль он выполняет. Например, внутри тега <head> напишите комментарий: "Я — раздел head. Я храню служебную информацию для браузера, а не контент для пользователей." Сделайте для <html>, <head>, <title>, <body>, <h1>.
15	Создайте семь прямоугольных блоков ( ), каждый из которых будет представлять цвет радуги (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый). Требования: – Каждый блок должен иметь: – Высоту 50px. – Ширину 300px. – Внешний отступ снизу 5px (margin-bottom), чтобы блоки не слипались. – Используйте CSS-свойство background-color, чтобы задать каждому блоку свой цвет. Дополнительно: Добавьте всем блокам белую границу толщиной 2px (border: 2px solid white;), чтобы подчеркнуть переход цветов.

Лабораторная работа 2. CSS

Цель работы: сформировать навыки работы с CSS.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 1, темы 2-4. Что такое HTML и CSS – урок 20. Базовые CSS-свойства – уроки 1-11. Больше CSS – уроки 1-18.

Изучите следующие разделы: Еще одна победа. **Базовые CSS-свойства.** Собрать лендинг. Размеры в пикселях. Размеры в процентах и долях. Цвета в HTML. Фон элемента. Позиция, размер, повтор фона. Прозрачность. Коробка в коробке. Наследование. Типографика. Больше вёрстки. **Больше CSS.** Пик вёрстки. Новая секция. Классы. Приятное и полезное. Типографика под контролем. Поток и блочная модель. Внешние и внутренние отступы. Шорткат. Что за страшное слово? Границы. Внутрь или наружу? Как элемент считает свои размеры. Блочные+строчные. Расположение элементов по центру: margin: auto:. Несколько классов. Тени. Подвал сайта. Центрировать по вертикали. Один шаг. Вы всохитительны!

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте HTML-страницу с тремя блоками . Задайте каждому блоку разный цвет фона, используя названия цветов (например, coral, steelblue). Для каждого блока установите: – Внутренний отступ (padding) - 25 пикселей. – Внешний отступ (margin) - 15 пикселей. – Границу (border) толщиной 3 пикселя, сплошную (solid) и черного цвета. – Ширину (width) - 200 пикселей. – Напишите внутри каждого блока текст "Блок №...".
2	Создайте страницу с двумя контейнерами ( ). Первому контейнеру задайте ширину 80% от окна браузера и высоту 150px. Внутри него создайте два вложенных блока. Первому вложенному блоку задайте ширину 50% (от родителя), а второму - 200px. Залейте все блоки разными цветами, чтобы видеть их границы.
3	Найдите в интернете небольшое текстурированное изображение (например, ткань, бумага). Создайте блок с заданными шириной и высотой (например, 400x300px). Установите найденное изображение как фон этого блока с помощью background-image. Поэкспериментируйте со свойствами: – background-repeat: no-repeat; – background-position: center; – background-size: cover; – Напишите внутри блока заголовок <h2> с текстом и убедитесь, что он читается.
4	Создайте блок с фоновым изображением (пейзаж). Поверх этого блока, но внутри него, создайте другой полупрозрачный блок ( ) черного цвета. Для этого внутреннему блоку задайте background-color: rgba(0, 0, 0, 0.5); (где 0.5 - это прозрачность). Внутри полупрозрачного блока разместите белый заголовок <h1> с текстом. Добейтесь эффекта "текст на затемненном фоне".
5	Создайте HTML-страницу с базовой структурой. В теге <style> задайте для элемента body свойство font-family: Arial, sans-serif; и color: navy;. Затем внутри <body> создайте заголовок <h1>, абзац <p> и список с несколькими пунктами . Убедитесь, что шрифт и цвет унаследовались всеми элементами. Теперь для <h1> задайте свой color: darkred; и посмотрите, как переопределяется наследуемое свойство.
6	Оформите текстовую страницу о вашем хобби, используя следующие CSS-свойства для текста: – Для заголовка <h1>: text-align: center;, font-weight: 700;, letter-spacing: 2px;. – Для всех абзацев <p>: text-align: justify;, line-height: 1.6; (межстрочный интервал), text-indent: 20px; (красная строка). – Для выделения ключевых слов внутри текста используйте тег с классом, которому задайте font-style: italic; и text-decoration: underline;.
7	Создайте два идентичных блока . Первому блоку задайте отступы и границу с помощью отдельных свойств: – margin-top: 10px; margin-right: 20px; margin-bottom: 10px; margin-left: 20px; – border-width: 2px; border-style: dashed; border-color: gray;

	– padding: 15px 25px; Второму блоку задайте те же самые значения, но используя шорткаты: – margin: 10px 20px; (первое значение — верх/низ, второе — право/лево) – border: 2px dashed gray; – padding: 15px 25px; Убедитесь, что блоки выглядят абсолютно одинаково.
8	Создайте страницу, где продемонстрируете разницу между блочными и строчными элементами. – Добавьте два блока с текстом и залейте их разными цветами. Посмотрите, как они ведут себя. – Затем внутри одного из добавьте несколько строчных элементов () с текстом и также залейте их цветом. Обратите внимание на разницу в поведении. – Попробуйте для одного из задать display: block; и посмотрите, как он изменит свое поведение.
9	Создайте блок с шириной 600px и залейте его цветом. Задайте этому блоку свойство margin: 20px auto;. Объясните в комментарии к коду, что делает auto для левого и правого отступа и почему это центрирует блок. Добавьте внутрь блока заголовок <h2> и задайте ему text-align: center;.
10	Создайте три кнопки, используя теги или <button>. – Создайте базовый класс .btn, который задаст общий вид: padding: 10px 20px;, margin: 5px;, border: none;, border-radius: 5px;. – Создайте три дополнительных класса для цветов: .btn-blue (синий фон), .btn-green (зеленый фон), .btn-red (красный фон). – Назначьте каждой кнопке два класса: общий btn и один из цветных (например, Кнопка 1).
11	Создайте визитную карточку. Это должен быть блок с белым фоном, внутренними отступами, закругленными углами (border-radius) и тенью. Для тени используйте свойство box-shadow. Попробуйте разные значения, например: box-shadow: 5px 5px 15px rgba(0, 0, 0, 0.3);. Внутри блока разместите свое имя (заголовок) и контактную информацию (абзац).
12	Создайте простую структуру страницы с "подвалом" (футером). Подвал должен всегда быть внизу окна браузера, даже если контента на странице мало. Для этого: – Задайте для body высоту 100vh и margin: 0; (чтобы убрать отступы по умолчанию). – Создайте для основного контента с классом .content и задайте ему min-height: calc(100vh - 60px); (60px — это высота футера). – Создайте <footer> и задайте ему высоту 60px, темный цвет фона, белый текст и свойство text-align: center;. Разместите в нем текст, например, "© Мой сайт, 2024".
13	Вертикальное центрирование одной строки текста. Создайте блок-шапку сайта (<header>) высотой 80px и с темным фоном. Внутри него разместите заголовок <h1> с названием сайта. Отцентрируйте текст заголовка по вертикали внутри этого блока. Для этого заголовку задайте свойство line-height: 80px; (такую же, как высота родителя). Убедитесь, что текст также отцентрован по горизонтали (text-align: center;).
14	Создайте блок для цитаты. Для этого блока задайте:

	– Нестандартную рамку с помощью border: сделайте ее толстой, пунктирной (dashed) и цветной. – Сильно скругленные углы с помощью border-radius: 20px;. – Фоновый цвет, отличающийся от фона страницы. – Внутренние отступы. – Тень. – Внутри блока разместите цитату в кавычках.
15	Создайте разноцветный блок с текстом и кнопкой. – Создайте файл index.html – Карточка имеет голубой фон – Текст отцентрирован – Кнопка синего цвета с белым текстом – У карточки скругленные углы

Лабораторная работа 3. JavaScript

Цель работы: сформировать навыки работы с JavaScript.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 1, темы 5-9. JavaScript. Начало – уроки 1-9. JavaScript. Приземление в реальность – уроки 1-18. Вёрстка — продолжение дизайна – уроки 1-8. Готовимся писать код – уроки 1-7. Файлы в проекте – уроки 1-9.

Изучите следующие разделы: **JavaScript. Начало.** Новая цель. JavaScript? Незаменимый инструмент. Числа. Строки. Арифметика строк. Переменные. **JavaScript. Приземление в реальность.** Мастер прокрастинации. Подключите JavaScript к HTML. Массивы. Случайные числа. Математика ниже плинтуса. Функции. Функции с аргументами. Возвращаемое значение. Выбор и изменение элементов страницы. Реакция на действия пользователей. Булевы величины. Условия в теории. Условия на практике. Объекты. Подключение внешних библиотек. Разделение труда. Циклы. Релиз. **Вёрстка — продолжение дизайна.** Работа с макетом. Акт l, сцена 1. Создание цифрового продукта. Что такое макет и где его делают? Знакомство с Figma. Страница → Фрейм → Группа → Элемент. Работа с

отступами, текстом и цветами. Экспорт элементов и копирование кода. Заключение. **Готовимся писать код.** Что нужно для написания кода? Редактор кода. Автоматизируем это! Комментарии в коде. Инструменты разработчика. О браузерных стилях. **Подход Pixel Perfect.** Файлы в проекте. Структура файлов, пути к файлам. Введение. Файл. Линейная структура и иерархия. Разделение файлов по типам. Абсолютные и относительные пути к файлам. Абсолютный путь и тег <link>. Относительные пути к файлам. CSS-директивы. @import. Заключение. Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте HTML-страницу с подключенным JavaScript-файлом. В скрипте создайте три переменные: name (ваше имя), age (ваш возраст) и hobby (ваше хобби). Выведите в консоль браузера (console.log) фразу: "Меня зовут [имя], мне [возраст] лет, и я люблю [хобби]".
2	Создайте в HTML кнопку <button id="myButton">Нажми меня</button>. В JavaScript найдите эту кнопку и добавьте ей обработчик события click. При нажатии на кнопку должен выводиться alert с текстом "Кнопка нажата!".
3	Создайте в HTML кнопку и div-блок с текстом. При нажатии на кнопку меняйте цвет фона div-блока на синий, а цвет текста - на белый. Используйте element.style.backgroundColor и element.style.color.
4	Создайте в HTML абзац с текстом и кнопку "Увеличить шрифт". При нажатии на кнопку размер шрифта (fontSize) абзаца должен увеличиваться на 2 пикселя.
5	Создайте в HTML пустой . С помощью JavaScript получите текущую дату (new Date().toLocaleDateString()) и выведите ее в этот div.

6	Создайте в HTML тег <img id="myImage" src="image1.jpg"> и кнопку "Сменить картинку". При нажатии на кнопку атрибут src изображения должен меняться на "image2.jpg".
7	Создайте в HTML пустой список <ul id="myList"> и поле ввода с кнопкой "Добавить". При нажатии на кнопку текст из поля ввода должен добавляться как новый пункт в конец списка.
8	Создайте в HTML , у которого в стилях задана граница (border: 2px solid black). Добавьте кнопку "Изменить рамку". При нажатии на кнопку цвет границы должен меняться на красный, а толщина — на 5px.
9	Создайте поле для ввода email и кнопку "Проверить". При нажатии проверяйте, содержит ли введенный текст символ "@". Если содержит - подсвечивайте поле зеленой рамкой, если нет - красной.
10	Создайте кнопку "Сменить тему". При нажатии на нее меняйте цвет фона страницы с белого на темно-синий, а цвет текста - с черного на белый, и наоборот при повторном нажатии.
11	Создайте массив phrases с 3-4 разными фразами (например, ["Отличный день!", "Вперед к успеху!", "Учим JavaScript!"]). Создайте кнопку, при нажатии на которую в заранее созданный выводится одна случайная фраза из массива.
12	Создайте в HTML div с фоновым цветом и кнопку "Сделать прозрачным". При нажатии на кнопку установите стиль opacity: 0.5 для этого div. Добавьте вторую кнопку "Вернуть непрозрачность", которая устанавливает opacity: 1.
13	Создайте поле ввода <input type="text" id="textInput"> и кнопку "Проверить". При нажатии на кнопку проверяйте: если поле пустое, устанавливайте ему красную границу (border: 2px solid red), если нет - зеленую (border: 2px solid green).
14	Создайте переменные userName, userCity и userAge. Присвойте им значения (ваше имя, город и возраст). Выведите в консоль строку: "Пользователь [userName] из города [userCity], возраст: [userAge] лет."
15	Создайте в HTML абзац Это секретный текст! и кнопку "Спрятать/Показать". Реализуйте функционал, чтобы при нажатии на кнопку абзац исчезал (стиль display: none), а при повторном нажатии — появлялся (стиль display: block).

Лабораторная работа 4. Вёрстка. Семантика и лейаут.

Цель работы: сформировать навыки вёрстки страниц, анализа макета.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 2, темы 1-6. Интро в спринт – урок 1. Семантика и глобальные атрибуты – уроки 1-6. Шрифты и типографика – уроки 1-5. Флексбокс-вёрстка – уроки 1-13. Готовый кейс –

уроки 1-2. Готовый кейс – уроки 1-2. Позиционирование элементов – уроки 1-9.

Изучите следующие разделы: **Интро в спринт.** Интро в спринт. **Семантика и глобальные атрибуты.** Разнообразие контейнеров HTML5. Семантика и выразительность HTML. Таблицы. Приведём в порядок код. Язык документа и его элементов. Идентификаторы. **Шрифты и типографика.** Подключение шрифтов к странице. Тонкости настройки шрифтов. Подключение внешних шрифтов. Оформление текста. Переполнение текстовых блоков. **Флексбокс-вёрстка.** Флексбокс-вёрстка. Введение. Преобразование во флекс-контейнер. Вложенные флекс-контейнеры. Направление внутри флекс-контейнера. Выравнивание по основной оси. Выравнивание по дополнительной оси. Ура! Инструменты разработчика. Перенос флекс-элементов. Свойство gap. Выравнивание флекс-элементов при переносе. Свойства флекс-элементов. Flex basis, flex-grow и flex-shrink. Заключение. **Готовый кейс.** Готовый кейс «С чистого листа». Видеоразбор готового кейса «Страх чистого листа». **Позиционирование элементов.** Позиционирование элементов. Введение. Понятие потока и статическое позиционирование. Относительное позиционирование. Абсолютное позиционирование. z-index. Фиксированное позиционирование. Липкое позиционирование. Свойство inset, упрощающее жизнь. Заключение.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Задание: Создайте HTML-страницу, используя семантические теги: <header>, <nav>, <main>, <section>, <article>, <footer>. В каждом разделе разместите соответствующий заголовок и краткий текст.
2	Создайте горизонтальное навигационное меню с 4-5 пунктами. Используйте Flexbox для выравнивания пунктов меню по центру контейнера с равными промежутками между ними.
3	Создайте карточку товара с изображением. Разместите в правом верхнем углу изображения бейдж "Акция" с помощью абсолютного позиционирования.
4	Создайте галерею из 6 изображений. Используйте Flexbox с переносом элементов, чтобы изображения располагались в 3 столбца с промежутками между ними.
5	Создайте длинную секцию с контентом. Заголовок секции должен быть зафиксирован вверху экрана при прокрутке, используя sticky-позиционирование.
6	Создайте блок профиля пользователя: аватар слева, имя и описание справа. Используйте Flexbox для выравнивания элементов по вертикали.
7	Создайте кнопку "Открыть". При клике на нее должно появляться модальное окно использованием фиксированного позиционирования.
8	Создайте Flex-контейнер с 3 элементами. Настройте разные пропорции роста элементов: 1:2:1 с помощью свойства flex-grow.
9	Создайте страницу блога с <main>, внутри которого 3 <article>. Каждая статья должна содержать <header> с заголовком и датой, и <footer> с автором.
10	Создайте блок фиксированной высоты с текстом. С помощью Flexbox отцентрируйте текст по вертикали и горизонтали.
11	Создайте подвал сайта с 3 колонками (о компании, контакты, соцсети). Используйте Flexbox для равномерного распределения колонок.
12	Создайте 3 разноцветных блока, частично перекрывающих друг друга. Управляйте порядком наложения с помощью z-index.
13	Создайте Flex-контейнер с элементами, которые переносятся на новую строку при уменьшении ширины экрана. Добавьте промежутки между элементами.
14	Создайте информационную карточку. В левом верхнем углу разместите иконку с помощью относительного/абсолютного позиционирования.
15	Создайте макет страницы: хедер, основная область с сайдбаром и контентом (Flexbox), футер. Реализуйте "липкий" футер, который всегда внизу при малом контенте.

Лабораторная работа 5. Grid Layout, Bash и Git. Основы.

Цель работы: сформировать навыки вёрстки страниц, анализа макета.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 2, темы 7-10. Grid Layout – урок 1-13. Кодстайл – уроки 1-5. Bash и Git. Основы – уроки 1-13. Проектная работа – уроки 1-7.

Изучите следующие разделы: **Grid Layout.** Введение. Что такое гриды. Столбцы и строки внутри грид-контейнера. Создание отступов. Функция repeat(). Единица измерения: фракция. Расположение элементов в грид-контейнере. Грид-области. Выравнивание сетки в грид-контейнере. Выравнивание содержимого грид-областей. Выравнивание содержимого внутри одной области. Наложение элементов. Заключение. **Кодстайл.** Что такое кодстайл. Общие принципы оформления HTML и CSS. Организация кода. Настройки IDE. Взгляд вперёд. **Bash и Git. Основы.** Введение. Установка Git. Ещё один инструмент — командная строка. Шпаргалка: командная строка. Настраиваем и подключаем Git к проекту. Делаем первый коммит. Командная работа. GitHub. Регистрация на GitHub. Генерируем SSH-ключи. Связываем локальный и удалённый репозитории. Синхронизация локального и удалённого репозитория. Шпаргалка: Git. Клонирование репозитория. **Проектная работа.** Что вас ждёт. Проектная работа «Оно тебе надо». Как сдавать проектные работы. Часто задаваемые вопросы (FAQ). Чек-лист. Сдача проектной работы «Оно тебе надо». Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Простая Grid-сетка. Создайте контейнер с display: grid и разместите в нем 6 элементов. Сделайте сетку 2x3 (2 строки, 3 столбца). Используйте свойство grid-template-columns с значениями repeat(3, 1fr). Добавьте отступы между элементами с помощью gap.
2	Фото-галерея на Grid. Создайте сетку для фото-галереи из 9 изображений. Сделайте 3 колонки равной ширины и автоматические строки. Задайте всем изображениям одинаковую высоту 150px и свойство object-fit: cover.
3	Макет страницы с Grid-областями. Создайте макет страницы с областями: header, sidebar, main, footer. Используйте grid-template-areas для размещения элементов. Сделайте header и footer на всю ширину, sidebar - 200px, main - оставшееся пространство.
4	Адаптивная Grid-сетка. Создайте сетку, которая на больших экранах показывает 4 колонки, на планшетах - 2 колонки, на телефонах - 1 колонку. Используйте медиа-запросы и изменение grid-template-columns.
5	Выравнивание в Grid. Создайте Grid-контейнер с 4 элементами разного размера. Поэкспериментируйте с свойствами justify-items, align-items, justify-content и align-content для разных типов выравнивания.
6	Наложение элементов в Grid. Создайте Grid с одним элементом, занимающим всю область. Разместите поверх него другой элемент с помощью grid-row и grid-column, используя z-index для управления порядком наложения.
7	Первый коммит в Git. Создайте новую папку для проекта, инициализируйте Git-репозиторий. Создайте файл index.html с базовой структурой, сделайте первый коммит с сообщением "Initial commit".
8	Работа с ветками. Создайте основную ветку main, затем создайте и переключитесь на ветку feature. Добавьте новый файл style.css, сделайте коммит. Вернитесь в main и слейте изменения из feature.
9	Подключение к GitHub. Создайте аккаунт на GitHub (если нет), создайте новый репозиторий. Свяжите локальный репозиторий с удаленным и выполните первый push.
10	Клонирование репозитория. Найдите любой открытый репозиторий на GitHub (например, с учебными примерами), склонируйте его себе на компьютер. Откройте файлы в редакторе кода.

11	Grid с именованными линиями. Создайте сетку с 3 колонками и 2 строками, используя именованные grid-line. Разместите элементы, явно указывая start-line и end-line.
12	Сложный Grid-макет. Создайте макет журнала с Grid. Один элемент должен занимать 2 колонки и 2 строки, остальные - стандартные ячейки. Используйте grid-template-rows и grid-template-columns с разными значениями.
13	Grid с автоматическим заполнением. Создайте сетку с автоматическим заполнением: grid-template-columns: repeat(auto-fill, minmax(150px, 1fr)). Добавьте 8 элементов и посмотрите, как сетка адаптируется под разную ширину экрана.
14	Карточка товара с Grid. Создайте карточку товара используя Grid. Разместите изображение вверху, название товара под ним, цену в левом нижнем углу, кнопку "Купить" в правом нижнем углу.
15	Адаптивное меню на Grid. Создайте навигационное меню, которое на компьютерах показывает 5 пунктов в ряд, а на мобильных устройствах - в один столбец. Используйте медиа-запросы и изменение grid-template-columns.

Лабораторная работа 6. Вёрстка. Доступность и подходы к организации стилей.

Цель работы: сформировать навыки вёрстки страниц, подходы к организации стилей.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 3, темы 1-8. Интро в спринт – урок 1. Настройка страницы и мета – уроки 1-5. Внешний встраиваемый контент: видео, iframe, API – уроки 1-7. Дополнение блочной модели – уроки 1-7. Псевдоклассы и псевдоэлементы – уроки 1-10. Готовый кейс – уроки 1-2. Доступность – уроки 1-10. Разметка форм – уроки 1-15.

Изучите следующие разделы: **Интро в спринт. Настройка страницы и мета.** Настройка страницы. Описание сайта для браузеров и поисковых машин. Фавикон. Разметка OpenGraph. Потренируемся. **Внешний встраиваемый контент: видео, iframe, API.** Введение. Страница в странице. API RuTube. Виджеты-информеры. Тег <video>. Тег <audio>.

Заключение. **Дополнение блочной модели.** Введение. Что ещё есть вокруг блока? Переполнение блока. Выпадение полей. Схлопывание полей. Относительные размеры всех полей. Заключение. **Псевдоклассы и псевдоэлементы.** Что это такое и в чём разница? LoVe HAte. Фокусы. Ведём счёт. Исключение. А если найду? Элементы, которых нет. ::marker. ::selection. О чём ещё не сказано. **Готовый кейс.** Готовый кейс «Надо сделать идеально». Видеоразбор кейса «Надо сделать идеально». **Доступность.** Введение. Доступность начинается с дизайна. Скринридеры, базовые правила a11y при вёрстке. Дерево доступности. Видимое и невидимое. Как скрывать элементы? Декоративные и контентные изображения. Поля форм: основы. ARIA. Настройки уменьшения движения и повышения контрастности. Заключение. **Разметка форм.** Введение. Что такое форма? Поля ввода с тегом <input>. Минимальные и максимальные значения. Поля загрузки, сброса и отправки данных. Поля ввода с другим синтаксисом. Ярлыки. Передаваемые значения. Поля множественного и единичного выбора. Подсказки в текстовых полях. Обязательные и заблокированные поля. Автозаполнение форм и полей. Режимы ввода для виртуальной клавиатуры. Отправка запроса атрибутом action. Заключение.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	**Задание**

1	Настройка мета-тегов и фавикона. Создайте HTML-страницу с правильными мета-тегами: title, description, keywords. Добавьте фавикон и разметку OpenGraph для социальных сетей (og:title, og:description, og:image).
2	Доступная навигация с ARIA-атрибутами. Создайте навигационное меню с использованием ARIA-ролей и атрибутов. Добавьте aria-label для навигации, aria-current="page" для активного пункта меню.
3	Форма с доступными полями ввода. Создайте форму обратной связи с правильно связанными labels и полями ввода. Добавьте обязательные поля, подсказки и сообщения об ошибках с aria-describedby.
4	Встраивание видео и аудио контента. Создайте страницу с встроенным видео через тег <video> и аудио через <audio>. Добавьте контролы, предзагрузку и альтернативный текст.
5	Псевдоклассы для интерактивных элементов. Создайте набор кнопок и ссылок. Стилизуйте состояния :hover, :focus, :active. Добавьте плавные переходы между состояниями.
6	Форма с различными типами полей. Создайте форму регистрации с разными типами input: text, email, password, tel, date, range, checkbox, radio. Добавьте правильную разметку для каждого поля.
7	Скрытие элементов для скринридеров. Создайте страницу с декоративными и контентными изображениями. Правильно скройте декоративные элементы используя aria-hidden и CSS-техники.
8	Псевдоэлементы для стилизации. Создайте стилизованный список используя ::before, ::after и ::marker. Добавьте кастомные маркеры и декоративные элементы.
9	Доступная таблица. Создайте таблицу с использованием scope для заголовков, caption для описания и правильной семантической разметкой для доступности.
10	Форма с валидацией и автозаполнением. Создайте форму заказа с автозаполнением (autocomplete), подсказками (placeholder) и валидацией через HTML5-атрибуты (required, pattern, min, max).
11	Аккордеон с семантической разметкой. Создайте аккордеон используя <details> и <summary>. Добавьте плавную анимацию раскрытия и обеспечьте доступность через правильные ARIA-роли.
12	Форма заказа с динамической валидацией. Создайте форму заказа товара с динамической проверкой наличия товара на складе. Добавьте live-region для объявления изменений состояния скринридерам.
13	Форма с зависимыми полями. Создайте форму где выбор в одном поле влияет на доступность и содержание других полей. Обеспечьте доступность изменений через aria-live.
14	Кастомный select с доступностью. Создайте кастомный выпадающий список с поддержкой клавиатурной навигации. Используйте ARIA-роли listbox, option и управляйте состоянием через JavaScript.
15	Доступная карусель изображений. Создайте карусель изображений с управлением через клавиатуру (Tab, Enter, стрелки). Добавьте ARIA-атрибуты для обозначения текущего слайда и общего количества. Реализуйте автопаузу при фокусе на карусели.

Лабораторная работа 7. Вёрстка. Доступность и подходы к организации стилей.

Цель работы: сформировать навыки вёрстки страниц, подходы к организации стилей.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 3, темы 9-13. Чудные селекторы – уроки 1-6. Стилизация форм – уроки 1-9. Продвинутый Git и Bash – уроки 1-13. Полезные материалы – уроки 1-4. Проектная работа – уроки 1- 4.

Изучите следующие разделы: **Чудные селекторы.** Введение. Каскад. «Сколько вешать в граммах?». Нарушитель правил. Разные типы селекторов. Фолбэки. **Стилизация форм.** Введение. Стилизуем поля ввода, кнопки и ярлыки. Псевдоклассы валидации. Выходим за пределы. Создаём красивые флажки. Стилизуем выпадающие списки. accent-color. Стилизация ренджа. Заключение. **Продвинутый Git и Bash.** Как работать в командной строке быстро. Копирование и перемещение файлов. Просмотр и редактирование файлов в командной строке. Шпаргалка: bash. Ещё раз про git add и git commit. Логирование и хеширование. Просмотр изменений: git diff. Что делать, если закоммитили что-то не то. Шпаргалка: git. Оформление репозитория. README.md. Знакомство с ZSH. Платформа SourceCraft. **Полезные материалы.** Компонентный подход. БЭМ — Блок, Элемент, Модификатор. Введение в автоматизацию: линтеры и форматтеры. Настраиваем автоматизацию форматирования и линтинга. **Проектная работа.** Проектная работа «Посмотри в окно». Чек-лист. Сдача проектной работы «Посмотри в окно». Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Сложные CSS-селекторы для таблицы. Создайте таблицу 5x5 и стилизуйте ее используя различные селекторы: • :nth-child() для создания зебры • :first-child и :last-child для крайних ячеек • :not() для исключения заголовков • Селекторы атрибутов для ячеек с определенными данными
2	Стилизация полей ввода с псевдоклассами валидации. Создайте форму с полями разных типов (email, tel, number). Добавьте стилизацию для состояний :valid, :invalid, :required, :optional. Используйте :focus для активного поля.
3	Кастомные чекбоксы и радиокнопки. Создайте стилизованные чекбоксы и радиокнопки без использования JavaScript. Скрывайте нативные элементы и создавайте кастомные через ::before и ::after.
4	Стилизация range-слайдера. Создайте кастомный range-слайдер с изменяющимся цветом трека и стилизованным ползунком. Используйте псевдоэлементы для браузерных префиксов.
5	Стилизация select с кастомной стрелкой. Создайте кастомный выпадающий список с стилизованной стрелкой через appearance: none и псевдоэлементы. Добавьте градиентный фон для options.
6	Каскад и специфичность селекторов. Создайте HTML со вложенными элементами. Напишите CSS-правила с разной специфичностью (классы, id, вложенные селекторы). Продемонстрируйте работу !important.
7	Стилизация полей с accent-color. Создайте форму с элементами, поддерживающими accent-color: чекбоксы, радиокнопки, range, progress. Покажите разницу между кастомной стилизацией и accent-color.
8	Стилизация формы с кастомными toggle-переключателями. Создайте toggle-переключатели (on/off) на основе чекбоксов. Используйте псевдоэлементы для создания слайдера и анимации переключения. Добавьте разные состояния: disabled, checked.

9	Git: работа с submodules и worktrees. Создайте основой репозиторий и подключите submodule с общими компонентами. Используйте git worktree для работы с несколькими ветками одновременно.
10	Комплексные селекторы для таблицы с группировкой. Создайте таблицу с группами строк (<tbody>) и примените: • :nth-of-type() для чередования групп • :first-of-type и :last-of-type для крайних групп • :has() для выделения групп с определенным контентом
11	Стилизация формы с кастомными radio-кнопками в виде карточек. Создайте форму выбора способа оплаты где radio-кнопки выглядят как карточки с иконками. Добавьте анимацию переключения и состояния hover/focus.
12	Git: работа с bisect для поиска багов. Создайте серию коммитов с преднамеренно добавленным багом. Используйте git bisect для автоматического поиска коммита, в котором был introduced баг.
13	Стилизация элементов с backdrop-filter. Создайте модальные окна и всплывающие подсказки с эффектом размытия фона используя backdrop-filter: blur(). Добавьте градиентные фоны с прозрачностью.
14	БЭМ: компонент рейтинга со звездами. Создайте компонент рейтинга используя методологию БЭМ: • .rating--interactive (для выставления оценки) • .rating--readonly (только отображение) • .rating--small, .rating--large (размеры)
15	Контейнерные запросы для адаптивных компонентов. Создайте карточку товара, которая меняет layout в зависимости от размера контейнера используя @container queries.

Лабораторная работа 8. Вёрстка. Адаптивность и графика

Цель работы: сформировать навыки вёрстки страниц и адаптивность.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 4, темы 1-6. Интро в спринт – уроки 1-2. Подходы – уроки 1-4. Растровая графика – уроки 1-6. CSS-переменные – уроки 1-6. Единицы измерения и функции – уроки 1-8. Готовый кейс – уроки 1-2.

Изучите следующие разделы: **Интро в спринт.** Интро в спринт. Видеоразбор. Проектные работы прошлых спринтов. **Подходы.** Введение. Разница между резиновой и адаптивной вёрсткой. Мобильный или десктоп: с

чего начать вёрстку? Адаптивный макет. **Растровая графика.** Растровая графика: форматы. Оптимизация изображений? Плотность пикселей. Картинки на выбор (браузера). image-set(). loading=”lazy”. **CSS-переменные.** Что такое переменные? Как пишется. Переопределение переменных. Запасное значение. Переменная внутри другой функции. Тёмная тема на CSS-переменных. **Единицы измерения и функции.** Введение. Проценты. Размеры от единиц окна просмотра. Единицы, связанные со шрифтами. Минимум и максимум. Функция calc(). Функции расчётов min(), max(), clamp(). Заключение. **Готовый кейс.** Готовый кейс: «Карты подскажут». Видеоразбор кейса «Карты подскажут».

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с выполненными заданиями.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Responsive images с srcset. Создайте <img> с атрибутом srcset, который предоставляет браузеру разные версии изображения для разных разрешений экрана.
2	Добавьте на страницу 10 изображений с атрибутом loading="lazy". Прокрутите страницу и наблюдайте за постепенной загрузкой.
3	Создайте набор CSS-переменных для цветов, шрифтов и отступов. Используйте их для стилизации нескольких элементов.
4	Реализуйте переключение между светлой и темной темой используя CSS-переменные и JavaScript.
5	Создайте макет где ширина элемента вычисляется как calc(100% - 40px), а высота как calc(50vh - 20px).
6	Создайте контейнер с шириной min(100%, 1200px) чтобы он не превышал 1200px даже на широких экранах.

7	Создайте кнопку с шириной max(200px, 50%) чтобы она была не менее 200px даже на узких экранах.
8	Используйте clamp(1rem, 2.5vw, 2rem) для создания responsive заголовков, которые масштабируются с размером экрана.
9	Создайте адаптивный iframe для видео, который сохраняет соотношение сторон 16:9 на всех экранах.
10	Создайте сетку с grid-template-columns: repeat(auto-fit, minmax(250px, 1fr)). Добавьте 8 карточек и наблюдайте, как они адаптируются под разную ширину экрана.
11	Используйте тег <picture> с несколькими <source> для предоставления разных форматов изображений (WebP, JPEG) и разных размеров для различных разрешений экрана.
12	Создайте компонент с отступами, которые плавно меняются от мобильных к десктопу: padding: clamp(1rem, 5vw, 3rem).
13	Создайте таблицу, которая на мобильных устройствах получает горизонтальный скролл, а на десктопе отображается полностью.
14	Создайте сложную форму, которая меняет layout с одноколоночного на многоколоночный при увеличении ширины экрана.
15	Создайте систему теней и градиентов используя CSS-переменные для легкого переиспользования.

Лабораторная работа 9. Вёрстка. Адаптивность и графика

Цель работы: сформировать навыки вёрстки страниц и адаптивность.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 4, темы 7-12. Grid Layout, часть 2 – уроки 1-7. Разработка интерфейса для разных устройств – уроки 1-10. Кроссбраузерность, вендорные префиксы – уроки 1-2. Подходы к написанию вёрстки – уроки 1-6. Git для взрослых – уроки 1-5. Проектная работа 1-4.

Изучите следующие разделы: **Grid Layout, часть 2.** Введение. Вспомнить всё. Размеры неявных строк и колонок. Свойство grid-auto-flow. Функции minmax() и fit-content(). Параметры auto-fill и auto-fit. Заключение. **Разработка интерфейса для разных устройств.** Введение. Адаптив без медиазапросов. Синтаксис медиазапросов. Характеристики устройств. Синтаксис диапазонов. Пользовательские предпочтения. Взаимодействие. Выражения от контейнера., Синтаксис. Выражения от контейнера., Новые

возможности. Заключение. **Кроссбраузерность, вендорные префиксы.** Вендорные префиксы и флаги. Кросс-браузерность. **Подходы к написанию вёрстки.** Введение. Desktop First и Mobile First. Выбор брейкпоинта. Где писать медиазапросы. Постепенная деградация и Прогрессивное улучшение. Заключение. **Git для взрослых.** Введение и повторение. Ветки. Слияние. Шпаргалка., Ветки в Git. Заключение. **Проектная работа.** Проектная работа «Сложно сосредоточиться». Чек-лист. Сдача проектной работы «Сложно сосредоточиться». Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте контейнер display: grid с 3 колонками шириной 150px. Добавьте в него 5 квадратных блоков (div) с картинками или просто цветным фоном. С помощью свойства grid-auto-rows задайте высоту всех неявно созданных строк (которые появятся, так как элементов больше, чем вмещает 3 колонки в одну строку) равной 200px. Используйте grid-auto-rows: 200px;. Обратите внимание, как 4-й и 5-й элементы автоматически перешли на новую строку.
2	Есть контейнер с фиксированной шириной 400px и высотой 600px. Сделайте его grid-контейнером. Создайте 3 колонки по 100px и 2 строки по 250px. Добавьте 6 книг (цветных блоков). Поменяйте значение свойства grid-auto-flow с row (по умолчанию) на column. Объясните в результате, почему книги теперь выстроились колонкой, а не строкой.
3	Создайте макет профиля пользователя. Сетка из 2 колонок. Левая колонка (аватар) должна иметь минимальную ширину 80px и максимальную

	120px. Правая колонка (имя и описание) должна занять всё оставшееся пространство. Используйте для этого функцию minmax() при определении grid-template-columns.
4	Создайте контейнер для тегов (например, «Дизайн», «Вёрстка», «JavaScript»). Используйте свойство grid-template-columns вместе с repeat(), auto-fit и minmax(). Сделайте так, чтобы колонки автоматически подстраивались под ширину контейнера, но не были уже 100px и не шире 200px.
5	Сверстайте карточку товара (изображение, заголовок, кнопка). Сначала (без медиазапросов) задайте стили для мобильных: ширина карточки 100%, отступы, крупный шрифт. Затем, внутри медиазапроса min-width: 768px, пропишите стили для планшетов: сделай карточку шириной 50% и размести две карточки рядом с помощью Flexbox или Grid.
6	Создайте блок с основной информацией (<p>) и блок с дополнительной, детальной информацией (<p class="details">). На маленьком экране (шириной меньше 600px) скройте блок .details с помощью display: none. На большом экране (600px и больше) покажите его.
7	Сделайте кнопку, у которой при наведении (:hover) есть анимация изменения масштаба (transform: scale(1.1)). Оберните это правило в медиазапрос @media (prefers-reduced-motion: no-preference). Это значит, что анимация будет работать только у тех пользователей, кто не включил настройку уменьшения движения в системе.
8	Задайте для страницы светлый фон и темный текст по умолчанию. Затем, с помощью медиазапроса @media (prefers-color-scheme: dark), переопределите цвета на тёмный фон и светлый текст.
9	Создайте красивую кнопку. Добавьте ей тень с помощью box-shadow. Для максимальной совместимости со старыми браузерами, продублируйте это свойство с префиксами -webkit-box-shadow и -moz-box-shadow. Значения у всех трёх свойств должны быть одинаковыми. Современные браузеры игнорируют непонятные им префиксы и используют последнее, понятное им свойство без префикса.
10	Создайте простой макет из трёх блоков (шапка, основное содержание, боковая панель) с помощью display: grid (grid-template-areas). Затем, до этого CSS-кода, пропишите стили с помощью display: block или flex, которые расположат эти блоки приемлемо для браузеров, которые не поддерживают Grid. Более современные браузера применят Grid и "улучшат" вёрстку.
11	Создайте форму из 6 полей ввода (input). Задайте сетку с 3 колонками (grid-template-columns: repeat(3, 1fr)). Первому и последнему полю дайте класс .wide и сделайте его на 2 колонки (grid-column: span 2). Теперь добавьте контейнеру свойство grid-auto-flow: row dense. Наблюдайте, как пустоты заполняются.
12	Создайте боковую панель (aside). Задайте ей ширину через fit-content(300px). Внутри добавьте текст разной длины. Наблюдайте: панель будет растягиваться под контент, но никогда не превысит 300px. Для сравнения, рядом создайте блок с width: 300px (фиксированная ширина) и блок с max-width: 300px (только ограничение максимума).
13	Создайте грид-контейнер с правилом: grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));. Добавьте в него 5 элементов (цветных div). Первым

	трём задайте высоту 100px, а последним двум — 200px. Добавьте grid-auto-rows: 100px;. Наблюдайте, как сетка создаёт строки высотой 100px, но для элементов высотой 200px она автоматически растягивает их на 2 строки.
14	Создайте заголовок <h1>. Вместо фиксированного font-size, задайте динамический размер: font-size: clamp(24px, 5vw, 48px);. Это означает: минимальный размер 24px, идеальный — 5% от ширины окна (viewport width), максимальный — 48px. Меняйте ширину окна браузера и наблюдай, как плавно меняется размер текста, не выходя за границы.
15	Создайте грид-контейнер с 2 явными колонками: grid-template-columns: 100px 200px;. Затем добавьте свойство grid-auto-columns: 50px;. Теперь добавьте в сетку 4 элемента. Наблюдайте: первые две колонки будут 100px и 200px (явные), а для 3-го и 4-го элемента, которые не влезли в явную сетку, Grid создаст неявные колонки шириной 50px каждая (как указано в grid-auto-columns).

Лабораторная работа № 10. Вёрстка: декорирование, подходы и инструменты

Цель работы: сформировать навыки декорирование, подходы и инструменты в вёрстке.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 5, темы 1-5. Интро в спринт – урок 1. Про SVG – уроки 1-8. Переходы и 2D-трансформации – уроки 1-7. Анимации – уроки 1-11. Вариативные шрифты – уроки 1-7.

Изучите следующие разделы: **Интро в спринт.** Интро в спринт. **Про SVG.** Введение в векторную графику. Экспорт SVG. Оптимизация SVG. Как вставить SVG на страницу. Управляем SVG. SVG-маски. Стили внутри SVG. Заключение. **Переходы и 2D-трансформации.** Добавляем плавности. Функция времени и задержка. Трансформации. Применение нескольких трансформаций. Новые свойства трансформаций. Свойство transform-origin. Матрица трансформаций. **Анимации.** Вау! Анимации! Из чего состоит анимация. Директива для анимаций — @keyframes. Задержка, пауза и стили после анимации. Функции времени. Опять. Уважаем пользователя. Отладка

анимаций. Дешёвые свойства для анимации. Свойство will-change. Анимация svg. Заключение. **Вариативные шрифты.** Вариативные шрифты: что такое и зачем. Узнаём больше о шрифте. Подключаем шрифт. Немного о font-variation-settings. Основные оси. Пользовательские оси. Заключение.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте файл heart.svg. Внутри напишите код SVG с базовой структурой. Добавьте красное сердце, используя тег <path> с этим значением: d="M12 21.35l-1.45-1.32C5.4 15.36 2 12.28 2 8.5 2 5.42 4.42 3 7.5 3c1.74 0 3.41.81 4.5 2.09C13.09 3.81 14.76 3 16.5 3 19.58 3 22 5.42 22 8.5c0 3.78-3.4 6.86-8.55 11.54L12 21.35z". Не забудьте про атрибуты viewBox, width и height у корневого тега <svg>. Чтобы сердце было красным, используй атрибут fill.
2	Вставьте SVG-круг (<circle>) прямо в HTML. Дайте ему класс, например, .pulsing-dot. В CSS для этого класса напишите анимацию пульсации: чередуй свойства transform: scale(1) и transform: scale(1.5) с помощью @keyframes. Добавьте бесконечное повторение и плавность. У круга должны быть атрибуты cx, cy (координаты центра) и r (радиус). Анимируйте его через CSS, как обычный HTML-элемент.
3	Сделайте полосу загрузки: длинный узкий прямоугольник. Внутри него создайте второй прямоугольник (индикатор). Напишите для индикатора анимацию @keyframes loading, которая меняет его ширину от 0% до 100%. Настройте анимацию так, чтобы она длилась 2 секунды, повторялась бесконечно и имела линейную функцию времени. Используйте overflow: hidden для внешнего блока, чтобы индикатор не вылезал за его границы. Для реалистичности попробуйте функцию времени ease-in-out.

4	Создайте 3 звезды (например, через псевдоэлементы ::before или отдельные div). Настройте для всех одну и ту же анимацию мерцания (меняй opacity). Задайте им одинаковую длительность анимации, но разную задержку (animation-delay): 0s, 0.5s, 1s.
5	Зайдите на сайт fonts.google.com. В фильтрах выберите «Variable fonts». Выберите понравившийся шрифт (например, «Roboto Flex») и подключите его к своей странице через @import или link. Напишите заголовок этим шрифтом. Google Fonts автоматически сгенерирует код для подключения. Ваша задача вставить его в <head> и применить шрифт к элементу через font-family.
6	Создайте простой SVG-элемент размером 20x20 пикселей (например, маленький круг или звездочку). Сохраните его как pattern.svg. Теперь создайте блок div размером 300x300px и задайте ему фон: css background-image: url('pattern.svg'); background-size: 20px 20px;
7	Создайте файл sprite.svg. Внутри разместите 3 разные иконки (например, дом, сердце, звезда), каждая в своем теге <symbol> с уникальным id. Затем на HTML-странице вставьте иконки через <use>: <svg><use xlink:href="sprite.svg#home"></use></svg>
8	Возьмите любую анимацию (например, движение блока слева направо). Зайди на сайт cubic-bezier.com. Передвигайте точки кривой Безье и смотри, как меняется характер движения блока на сайте в реальном времени. Скопируйте значения (например, cubic-bezier(.17,.67,.83,.67)) и вставьте в своё CSS-правило вместо linear.
9	Скачайте вариативный шрифт в формате .ttf или .woff2 (например, RobotoFlex). Зайдите на сайт www.axis-praxis.org и загрузи шрифт. Понадвигайте ползунки всех доступных осей (wght, wdth, opsz, GRAD, slnt, XTRA и т.д.). Напишите текст и понаблюдай, как каждая ось меняет отображение.
10	Создайте два параграфа. Первому задайте обычный шрифт и стиль font-weight: bold. Второму – вариативный шрифт и font-variation-settings: "wght" 700;. Сравните визуально. Теперь попробуйте установить значение "wght" 800 или 900. Увидите, что вариативный шрифт даёт гораздо больше контроля над насыщенностью. Обычный font-weight обычно дает доступ только к 2-3 начертаниям (400, 700, maybe 900). Вариативный шрифт дает непрерывный спектр от 100 до 900, а часто и за этими пределами.
11	Создайте страницу с крупным заголовком. Под ним разместите 3 ползунка input type="range" для управления осями вариативного шрифта: wght (толщина), wdth (ширина), slnt (наклон). С помощью JavaScript (простого oninput) свяжите значение каждого ползунка с соответствующим значением в font-variation-settings заголовка.
12	Преобразуйте короткое слово (например, "Hello") в SVG-пути с помощью любого онлайн-конвертора. Каждую букву поместите в отдельный <path>. Задайте всем путям stroke и fill="none", а также свойства stroke-dasharray и stroke-dashoffset для эффекта рисования. Настройте анимацию так, чтобы буквы рисовались последовательно, с небольшой задержкой одна за другой.

13	Создайте длинную горизонтальную спрайт-ленту с 5 кадрами человечка или простой фигурки. Поместите её в контейнер с overflow: hidden. Анимируй фоновую позицию (background-position) контейнера, чтобы лента сдвигалась от 0 0 до -100% 0. Но задайте анимации функцию времени steps(5, end). Наблюдайте, как плавное движение превратилось в чёткое переключение между 5 кадрами.
14	Создайте нарочно "тяжёлую" анимацию: например, анимируйте box-shadow с большим размытием или border-radius у большого блока. Откройте вкладку Performance в DevTools, нажмите "Запись", выполните действие, запускающее анимацию, останови запись. Проанализируйте шкалу "FPS": если она падает ниже 60, анимация дёргается. Найдите на временной шкале длинные жёлтые блоки (Layout, Paint) — они указывают на дорогие операции. Замените анимируемое свойство на transform или opacity и повтори запись.
15	Создайте набор из 5 точек (для индикатора загрузки). Назначьте им одну общую анимацию @keyframes pulse с изменением opacity и scale. Но для каждой точки задай разный animation-delay (0s, 0.1s, 0.2s...). Теперь измените один @keyframes — например, добавьте смену цвета. Все точки синхронно изменят своё поведение, но сохранят волновой эффект.

Лабораторная работа 11. Вёрстка: декорирование, подходы и инструменты

Цель работы: сформировать навыки декорирование, подходы и инструменты в вёрстке.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 5, темы 6-9. Декорирование – уроки 1-15. 3D трансформации – уроки 1-10. Современные интерактивные элементы – уроки 1-5. Проектная работа – уроки 1-4.

Изучите следующие разделы: **Декорирование.** Введение. border-image. Тени. Тень текста и множественные тени. Фильтры. Линейный градиент. Радиальный градиент. **3D трансформации.** Введение. От планиметрии к стереометрии. Свойства 3D-трансформаций. Перспектива. transform-origin в 3D. Обратная сторона. Вглубь, а не наружу. Сокращенные свойства. Матрица трансформаций в 3D. Заключение. Что было и что будет? **Современные**

интерактивные элементы. Введение. Тег <dialog>. Тег <details>. Атрибут popover. Заключение. **Проектная работа.** Проектная работа «Закрывающий тег». Чек-лист проектной работы «Закрывающий тег. Финал». Сдача проектной работы «Закрывающий тег. Финал». Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте кнопку. Вместо обычной рамки (border) задайте ей рамку из изображения. Найдите в интернете маленькое (30x30 пикселей) тайлуемое (безшовное) изображение для рамки (например, узор или пунктир). Используйте свойство: `border: 15px solid transparent;` `border-image: url('frame.png') 30 round;` Поэкспериментируйте с последним значением: замените round на stretch и repeat. Посмотрите, как меняется отображение углов.
2	Создайте карточку (div с белым фоном и скруглёнными углами). Добавьте ей тень box-shadow. Сделайте её не чёрной и размытой, а цветной и сдвинутой, чтобы создать эффект «парящей» карточки: `box-shadow: 0 10px 30px rgba(0, 100, 200, 0.2);`
3	Создайте заголовок с тёмным фоном. Добавьте ему неоновое свечение с помощью множественных теней. Попробуйте создать эффект «огненного» текста, используя градиент от жёлтого к красному в тенях. Тени применяются в порядке от первой (самой нижней) к последней (самой верхней). Размытие (15px, 20px) создаёт эффект свечения.
4	Создайте кнопку. Задайте ей линейный градиент от синего к фиолетовому под углом 45 градусов. Сделайте так, чтобы при наведении градиент

	менял направление на -45 градусов и цвета на противоположные. Добавьте плавный переход с помощью transition.
5	Создайте квадратный элемент, это будет "дверь". Поместите его в контейнер с perspective. При наведении на контейнер дверь должна "открываться", поворачиваясь на 80 градусов вокруг своей левой границы. Ключ – свойство transform-origin: left center; для самой двери. Анимация: transform: rotateY(-80deg);. transform-origin задаёт точку, вокруг которой происходит вращение. По умолчанию это center. Поэкспериментируй с top, bottom, right. Добавь плавности с transition.
6	Создайте контейнер для карточки. Внутри два div: лицевая сторона (front) и обратная (back). Расположите их друг над другом с помощью абсолютного позиционирования. Обратной стороне сразу задайте transform: rotateY(180deg);. Контейнеру задайте transform-style: preserve-3d;. При наведении на контейнер он должен поворачиваться на 180 градусов: transform: rotateY(180deg);. Стороны будут сменять друг друга! Чтобы обратная сторона не была видна в начале, используйте backface-visibility: hidden; для обеих сторон. Это скроет "изнанку" элемента при повороте.
7	Используйте тег <details>. Внутри него поместите <summary>, это будет кликабельный заголовок. А после любой контент (текст, картинки). Нажмите на заголовок, контент появится/скроется. Добавьте несколько таких блоков для FAQ-страницы. Можно стилизовать маркер (треугольник) через summary::marker или summary::-webkit-details-marker. Добавьте плавного раскрытия через CSS: details[open] summary ~ * { animation: sweep .5s ease-in-out; }.
8	Создайте ряд миниатюр (превью фотографий). Над ними разместите 3-4 кнопки-фильтра ("Сепия", "Ч/Б", "Яркость"). При клике на кнопку все миниатюры должны окрашиваться в выбранный фильтр. Активный фильтр можно подсветить с помощью box-shadow. Реализуйте через добавление/удаление CSS-класса миниатюрам по клику на кнопку. Класс будет содержать правило filter.
9	Создайте квадратный блок. Вместо обычной рамки, задайте ему градиентную рамку с помощью border-image и linear-gradient. Измените угол градиента на 90deg и 135deg. Добавьте больше цветов: linear-gradient(45deg, red, yellow, blue, purple). Попробуйте радиальный градиент: radial-gradient(circle, red, blue). Значение 1 в конце это border-image-slice. При значении 1 градиент равномерно растягивается по всей рамке. Попробуйте значение 30, чтобы увидеть разницу.
10	Создайте 5 карточек, расположенных по кругу. Каждой карточке задайте: `transform: rotateY(calc(var(--i) * 72deg)) translateZ(200px);` где --i — индекс карточки (0, 1, 2, 3, 4). Контейнеру задайте transform-style: preserve-3d; и анимируйте его вращение по оси Y. Получится 3D-карусель.
11	Создайте переключатель тем (светлая/темная). Для всех изображений на странице задайте CSS-фильтры, которые меняются при смене темы:

	`` /* Светлая тема (по умолчанию) */ img { filter: brightness(1) contrast(1); } /* Темная тема */ [data-theme="dark"] img { filter: brightness(0.8) contrast(1.2) saturate(0.9); } ``` Добавьте плавный переход: transition: filter 0.3s ease;.
12	Создайте шестиугольную кнопку (как у сот). Используйте clip-path: polygon(...) для создания формы. Теперь добавьте градиентную рамку. Проблема: border-image не работает с clip-path. Решение: создайте псевдоэлемент ::before с той же clip-path, но чуть большего размера, и примените border-image к нему. Основной элемент будет фоном.
13	Создайте текст или иконку. Задайте им text-shadow или box-shadow с голубым/розовым цветом и большим размытием. Добавьте CSS-фильтр filter: drop-shadow(...) с теми же параметрами для усиления эффекта. Теперь создайте анимацию @keyframes, которая плавно меняет: 1. Цвет тени (от голубого к фиолетовому и обратно) 2. Размер размытия 3. Яркость (filter: brightness()) самого элемента Используйте animation-iteration-count: infinite; и animation-direction: alternate;. Для большей достоверности, изучите палитры неоновых вывесок.
14	Создайте круговую диаграмму (pie chart), отображающую, например, распределение времени: Работа — 40% (синий), Отдых — 25% (зеленый), Учеба — 35% (красный). Добавьте белую окружность по центру с помощью радиального градиента поверх, чтобы получилось кольцо (donut chart). Рассчитывайте градусы по формуле: процент * 3.6. Используйте промежуточные значения в conic-gradient для точного контроля.
15	Создайте кнопку-триггер и popover-элемент. Используйте новый JavaScript API anchor() (или пока эмулируй через вычисления) для позиционирования popover не просто поверх, а привязанного к краю триггера (например, снизу по центру). Стилизуйте popover со стрелкой, указывающей на триггер.

Контрольные вопросы:

1. Какое свойство CSS отвечает за рамку-картинку?
2. Какое свойство создаёт тень у блока?
3. Какой тег создаёт раскрывающийся блок?
4. Какой фильтр делает фото чёрно-белым?
5. Зачем нужен clip-path?
6. Для чего нужны @keyframes?
7. Как сделать анимацию бесконечной?

8. Какой API помогает привязать всплывашку к кнопке?
9. Какое свойство делает углы блока скруглёнными?
10. С помощью какого свойства можно плавно изменить цвет кнопки при наведении?
11. Какое значение свойства filter увеличивает яркость изображения?
12. Какой псевдокласс срабатывает при наведении мыши на элемент?
13. Какое значение animation-timing-function делает анимацию равномерной?

Лабораторная работа № 12. Основы JavaScript, базовые типы, работа с DOM

Цель работы: сформировать навыки работы с JavaScript

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 6, темы 1-3. Интро в спринт – уроки 1-2. Вспомнить всё – уроки 1-11. Как находить решения проблем? – уроки 1-2.

Изучите следующие разделы: **Интро в спринт.** Интро в спринт. Видеоразбор: Проектные работы прошлых спринтов. **Вспомнить всё.** Вспомнить всё., Введение., От вёрстки к программированию. Вывести на страницу и в консоль, подключить JS к странице. Числа, операции с числами. Строки, конкатенация, приведение числа к строке. Переменные, типы переменных. Условия, больше-меньше, булевы операции. Массивы, создание, доступ по индексу, длина массива. Циклы for, while, do...while. Функции: определение и вызов. Объекты. Заключение. **Как находить решения проблем?** Поиск решения. Как работать с документацией.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте пустой HTML-файл и подключите к нему файл script.js. Внутри script.js напишите одну строку кода, которая выводит в консоль браузера ваше имя и фамилию. Запустите HTML-файл в браузере и проверьте вкладку "Console" (F12).
2	Даны исходная цена товара (1500 рублей) и размер скидки (15 процентов). Рассчитайте и выведите в консоль финальную цену товара после применения скидки. Используйте переменные для хранения чисел.
3	У вас есть два слова: "ivan" и "company". Создайте переменную email, которая с помощью конкатенации и символа @ соберёт строку "ivan@company.com". Выведите результат в консоль.
4	Создайте переменную stringNumber со значением "100" (это строка). Преобразуйте её в число с помощью Number() и умножьте на 2. Выведите результат в консоль.
5	Создайте две переменные: a = "кофе" и b = "чай". Напишите код, который меняет их значения местами (чтобы в a был "чай", а в b – "кофе"), используя третью временную переменную. Выведите новые значения.
6	Создайте переменную userPassword со строковым значением (например, "qwerty123"). Напишите условие: если длина пароля больше или равна 8 символам, выведите в консоль "Надёжный пароль", иначе – "Пароль слишком короткий".
7	Попросите пользователя ввести текущий час (число от 0 до 23) с помощью prompt(). Если час меньше 12, выведите alert("Доброе утро!"), если от 12 до 18 – alert("Добрый день!"), иначе – alert("Добрый вечер!").
8	Создайте массив colors, который содержит названия трёх ваших любимых цветов в виде строк. Выведите весь массив и его длину в консоль.
9	Объявите функцию greet(). Эта функция должна выводить в консоль одно и то же сообщение "Добро пожаловать!" при каждом вызове. Вызовите функцию дважды.

10	Напишите функцию convertToDollars(rubles), которая принимает количество рублей и возвращает эквивалент в долларах (курс: 1 доллар = 90 рублей). Вызовите функцию с аргументом 2700 и выведите результат в консоль.
11	Создайте функцию isEven(num), которая принимает число и возвращает строку "Чётное", если число чётное, и "Нечётное" – если нет. Протестируйте функцию на числах 10 и 7, выводя результат в консоль.
12	Создайте объект book со свойствами: title ("Гарри Поттер"), author ("Дж. Роулинг"), pages (400). Выведите в консоль строку по шаблону: "Название книги: [title], Автор: [author]", обращаясь к свойствам объекта.
13	Создайте объект calculator с одним методом add. Этот метод должен принимать два числа и возвращать их сумму. Вызовите метод add для чисел 33 и 12 и выведите результат.
14	Используя встроенный объект Math и его метод Math.random(), сгенерируйте случайное целое число от 1 до 10 включительно и выведите его в консоль с надписью "Ваше случайное число: X".
15	Создайте переменную rawTitle со значением "главные новости дня" (все буквы строчные). Используя строковые методы, преобразуйте эту строку так, чтобы каждое слово начиналось с заглавной буквы. Выведите результат в консоль: "Главные Новости Дня".

Контрольные вопросы

1. Чтобы подключить внешний JavaScript-файл к HTML-странице, какой используется тег?
2. Переменная в JavaScript объявляется с помощью какого ключевого слова?
3. Какой оператор проверяет равенство значений в JavaScript?
4. Что делает Math.random()?
5. Каким образом можно преобразовать строку "100" в число для выполнения математических операций?
6. Какой цикл лучше всего подходит для перебора всех элементов массива?
7. Что такое комментарий в JavaScript?
8. Как вывести всплывающее окно с сообщением "Привет!"?
9. Какое ключевое слово используется для объявления функции?
10. Что произойдет, если функция не содержит инструкцию return?
11. Какой метод массива объединяет все его элементы в строку?

12. Чем отличаются операторы == и === в JavaScript?

Лабораторная работа № 13. Основы JavaScript, базовые типы, работа с DOM

Цель работы: сформировать навыки работы с JavaScript

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 6, темы 4-5. Примитивы. Начало – уроки 1-6. Знакомство с DOM – уроки 1-10.

Изучите следующие разделы: **Примитивы. Начало.** Введение в примитивы. Примитивные типы данных., Оператор typeof. Два одиноких типа данных: undefined и null. Строки. Числа и специальные числовые значения. Заключение. **Знакомство с DOM.** Знакомство с DOM., Введение. DOM: выбор элементов. Атрибуты и их методы. Работа с атрибутами через встроенные свойства. Манипуляции с классами CSS. Управление содержимым: свойства .innerHTML и .textContent. Реакция на действия пользователя., События. Гибкая вставка: методы insertAdjacentHTML и insertAdjacentText. Другие полезные свойства элементов. Заключение.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте три переменные: одну со строковым значением, одну с числовым и одну с булевым. Используйте оператор typeof для каждой переменной и выведите в консоль результаты в формате: "Переменная со значением 'Привет' имеет тип: string".
2	Создайте переменную userData и явно присвойте ей значение null. Выведите в консоль её тип с помощью typeof. На следующей строке проверьте, равна ли переменная null (используйте ===), и выведите результат.
3	Создайте в HTML несколько элементов с классом "card". В JavaScript выберите первый элемент с этим классом, используя document.querySelector(). Выведите его в консоль.
4	Создайте кнопку <button id="click-me">Нажми меня</button>. Назначьте ей обработчик события click с помощью .addEventListener(). При клике должен выводиться alert с текстом "Кнопка была нажата!".
5	Создайте переменную mixedCase = "JaVaScRiPt". Выведите в консоль эту строку полностью в верхнем регистре, а затем полностью в нижнем регистре, используя соответствующие методы строк.
6	Создайте в HTML элемент с текстом и задайте в CSS класс .highlight (например, с жёлтым фоном). В JavaScript добавьте этот класс к элементу с помощью .classList.add(). Через 3 секунды (используйте setTimeout) удалите этот класс с помощью .classList.remove().
7	Имеется строка greeting = "Добро пожаловать на курс по фронтенду!". Используйте метод .includes(), чтобы проверить, содержится ли в этой строке слово "курс". Выведите в консоль true или false в зависимости от результата.
8	Создайте в HTML несколько элементов с классом "card". В JavaScript выберите первый элемент с этим классом, используя document.querySelector(). Выведите его в консоль.
9	Введите любое слово через prompt(). Сохраните ответ в переменную и выведите в консоль длину введённой строки с помощью свойства .length в формате: "В слове 'JavaScript' 10 символов".
10	Добавьте в HTML несколько тегов внутри списка . В JavaScript выберите все элементы на странице с помощью document.querySelectorAll(). Выведите в консоль количество найденных элементов (свойство .length).
11	Создайте кнопку в HTML с кастомным атрибутом: <button data-action="save">Сохранить</button>. В JavaScript получите значение этого атрибута двумя способами: через .getAttribute('data-action') и через свойство .dataset.action. Сравните результаты в консоли.
12	Создайте строку alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ". Выведите в консоль первый символ строки и последний символ строки, используя квадратные скобки и свойство .length.
13	Создайте кнопку в HTML. При клике на эту кнопку должен переключаться (добавляться/удаляться) класс .active у самой кнопки. В CSS задайте для

	этого класса заметное изменение (например, другой цвет фона). Используйте .classList.toggle().
14	Создайте в HTML пустой элемент <h2 id="subtitle"></h2>. В JavaScript выберите этот элемент и с помощью свойства .textContent установите его текстовое содержимое: "Добро пожаловать в наше приложение!".
15	Создайте переменную itemsCount = 5 (число) и переменную itemName = "яблок" (строка). С помощью конкатенации создайте строку: "У меня есть 5 яблок" и выведите её в консоль. Обратите внимание на автоматическое преобразование типов.

Контрольные вопросы:

1. Что вернет оператор typeof для переменной, которой явно присвоено значение null?
2. Какой метод используется для добавления класса CSS к элементу?
3. Какое свойство элемента HTML используется для получения или установки ТОЛЬКО текстового содержимого, без HTML-тегов?
4. Какой тип данных вернет prompt()?
5. Для сего используется метод insertAdjacentHTML?
6. Какое свойство позволяет получить или изменить текстовое содержимое элемента, игнорируя HTML-теги?
7. Какой метод используется для установки задержки перед выполнением кода?
8. Какой атрибут используется для хранения пользовательских данных в HTML5?
9. Какой метод превращает строку в нижний регистр?
10. Какой синтаксис используется для создания однострочного комментария в JavaScript?
11. Какой метод DOM используется для создания нового элемента (например,)?
12. Что выведет console.log(5 == "5");?

Лабораторная работа № 14. Основы JavaScript, базовые типы, работа с DOM

Цель работы: сформировать навыки работы с JavaScript

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 6, темы 6-7. Создание, добавление и удаление элементов в DOM – уроки 1-10. Методы работы с данными, условия, циклы – уроки 1-16.

Изучите следующие разделы: **Создание, добавление и удаление элементов в DOM.** Введение. Неприятный минус innerHTML. Создание элементов: createElement и createTextNode. Добавление элементов на страницу. Удаление и перемещение элементов. Клонирование элементов. template-элементы. Объект event. Родственные связи в DOM. Заключение. **Методы работы с данными, условия, циклы.** Введение., Второе знакомство с JavaScript. let и const против var. Методы поиска в строке. Методы преобразования строк. Методы для работы с числами. Неявное преобразование типов. Явное преобразование типов. Логические операторы. Оператор логическое НЕ (!). Оператор логическое ИЛИ (||). Оператор логическое И (&&). Конструкция switch-case. Тернарный оператор. Циклы., Директивы break и continue. Заключение. Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода).

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте новый элемент с помощью document.createElement(). Добавьте ему текстовое содержимое "Этот абзац создан через JavaScript" с помощью textContent. Добавьте созданный элемент в конец тега <body> с помощью appendChild(). Проверьте результат в браузере.
2	Дано число num = 7.89. Используя методы Math.round(), Math.floor() и Math.ceil(), выведите в консоль три результата округления: до ближайшего целого, в меньшую сторону и в большую сторону.
3	Создайте кнопку <button id="click-info">Кликни меня</button>. Добавьте обработчик события click, который принимает параметр event. В обработчике выведите в консоль свойства event.target (элемент, на который кликнули) и event.type (тип события). Проанализируйте, что выводится.
4	Введите своё имя через prompt(). Примените к введённой строке методы .trim() (убирает пробелы по краям) и .toUpperCase() (переводит в верхний регистр). Выведите результат в консоль в формате: "Привет, ИВАН!".
5	Создайте новый элемент . Затем создайте три элемента с текстами "Первый пункт", "Второй пункт", "Третий пункт". Добавьте все в , а — в конец document.body. Используйте только createElement, textContent и appendChild.
6	Создайте переменную userInput = "123abc". Попробуйте преобразовать её в число с помощью Number(). Затем проверьте результат с помощью isNaN() и выведите в консоль понятное сообщение: "Значение '123abc' не является корректным числом" или "Это число: ...".
7	Создайте две переменные: hasTicket = true и hasTime = false. Напишите условие, которое проверяет обе переменные с помощью оператора &&, и выводит в консоль "Можете идти в кино" только если обе истинны. Иначе выводите "Не можете идти в кино".
8	Создайте две строковые переменные: a = "10" и b = "5". Преобразуйте их в числа (используйте Number() или унарный плюс +) и выведите в консоль результат их сложения (должно получиться 15, а не "105").
9	Введите возраст. Сохраните значение в переменную. Используйте тернарный оператор (? :), чтобы создать переменную message, которая будет содержать "Доступ разрешён" если возраст >= 18, иначе "Доступ запрещён". Выведите message в консоль.
10	Создайте два элемента: Старый текст и новый элемент Новый текст (создайте его через JavaScript). Используйте метод .replaceWith(), чтобы заменить элемент с id="old" на созданный вами новый элемент.
11	Создайте структуру: <button id="child">Я внутри</button> . При клике на кнопку найдите её родительский элемент (parentNode) и выведите в консоль его id. Также измените цвет фона родительского элемента на светло-голубой.
12	Напишите код, который генерирует случайное целое число от 1 до 100 включительно. Используйте Math.random(), Math.floor() и арифметические операции. Выведите результат в консоль с пояснением: "Случайное число: 42".
13	Создайте цикл while, который перебирает числа от 1 до 10. Используйте директиву continue, чтобы пропустить вывод чётных чисел (то есть

	выводить только нечётные). В консоли должны появиться числа: 1, 3, 5, 7, 9.
14	Создайте переменную day = "суббота". Напишите условие с использованием ||, которое проверяет, является ли день выходным ("суббота" ИЛИ "воскресенье"). Выводите соответствующее сообщение в консоль.
15	Создайте строку с перечислением: "яблоки,бананы,апельсины,киви". Используйте метод .split(','), чтобы разбить эту строку на массив фруктов. Выведите в консоль полученный массив и его длину.

Контрольные вопросы:

1. Какой метод используется для создания нового HTML-элемента в JavaScript?
2. Какой метод добавляет созданный элемент в конец указанного родительского элемента?
3. Какой метод следует использовать для удаления дочернего элемента child у родителя parent?
4. Какое свойство позволяет получить текстовое содержимое элемента без учёта HTML-тегов?
5. Какой метод строки удаляет пробелы в начале и в конце строки?
6. Какое значение вернет Math.round(7.49)?
7. Какой метод разбивает строку на массив по разделителю?
8. Что означает аббревиатура DOM?
9. Какое ключевое слово используется для объявления функции в JavaScript?
10. Какой метод выводит диалоговое окно с сообщением и полем для ввода текста?
11. Как объявить константу в JavaScript?
12. Какое значение имеет переменная, которой присвоено null?

Лабораторная работа № 15. Продолжение JavaScript

Цель работы: приобрести первичные навыки работы с фреймверком React

Методические указания к практическому выполнению

Цель работы: продолжение изучения работы с JavaScript

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 7, темы 1-3. Массивы – уроки 1-15. Функции – уроки 1-11. Объекты – уроки 1-12.

Изучите следующие разделы: **Массивы.** Введение. Зачем нужны массивы. Объединение и преобразование в строку. Добавление и удаление последнего элемента. Добавление и удаление первого элемента. Управление элементами на любых позициях. Очень похожи на массивы, но не массивы. Коллекции в DOM. Перебор массива. Методы forEach и map. Функции обратного вызова и их аргументы. Отбор элементов массива: метод filter. Методы some, every, find. Сведение массива. Метод reduce. Сортировка массива. Дополнительно: императивное и декларативное программирование. Шпаргалка по методам массивов. Заключение. **Функции.** Функции. Повторение. Зачем нужны функции. Область видимости функции. Затенение идентификаторов. Способы создания функции. Функции — это значения. Стрелочные функции. Аргументы по умолчанию. Функции с неопределённым числом аргументов. Дополнительно: поднятие переменных и функций. Заключение. **Объекты.** Объекты. Введение. Создание объектов и запись свойств. Обращение к свойству. Операторы: delete, in. Перебор свойств. Передача по ссылке. Сравнение объектов. Поверхностное копирование объектов. Глубокое копирование объектов. Массивы — это объекты. Функции — это объекты. Объекты. Резюме.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте массив fruits, который содержит названия трёх фруктов, например: "яблоко", "банан", "апельсин". 1. Выведите в консоль первый элемент массива. 2. Используя свойство length, выведите последний элемент массива. 3. Выведите длину массива. 4. Добавьте ещё один фрукт в конец массива с помощью индекса. Проверьте, что длина стала равна 4, и выведите весь массив.
2	Дан массив чисел [1, 2, 3, 4, 5]. 1. С помощью метода push добавьте число 6 в конец массива. Выведите массив. 2. С помощью метода pop удалите последний элемент и сохраните его в переменную. Выведите удалённый элемент и получившийся массив. 3. С помощью метода unshift добавьте число 0 в начало массива. Выведите массив. 4. С помощью метода shift удалите первый элемент и сохраните его в переменную. Выведите удалённый элемент и итоговый массив.
3	Дан массив цветов: colors = ['красный', 'зеленый', 'синий', 'желтый']. 1. Используя splice, удалите элемент 'зеленый' (он находится под индексом 1). Выведите изменённый массив. 2. Теперь с помощью splice вставьте на место удалённого элемента два цвета: 'фиолетовый' и 'оранжевый'. Выведите массив. 3. Используя splice, замените 'синий' на 'голубой' (одновременно удалите один элемент и вставьте новый). Выведите итоговый массив.
4	Преобразование массива в строку и обратно (split и join). 1. Дана строка: "яблоко,груша,банан,апельсин". Преобразуйте её в массив фруктов, используя метод split с разделителем ",". 2. Из полученного массива создайте новую строку, где фрукты разделены пробелом (метод join). 3. Выведите в консоль исходную строку, полученный массив и новую строку.
5	Дан массив чисел: nums = [10, 20, 30, 40, 50].

	1. Используя метод forEach, выведите для каждого числа строку вида: "Элемент с индексом i имеет значение x", где i — индекс элемента, а x — его значение. 2. Дополнительно: используйте forEach, чтобы создать новый массив, содержащий эти же числа, умноженные на 2. Для этого сначала создайте пустой массив doubled, а внутри forEach добавляйте в него удвоенные значения. Выведите doubled.
6	Дан массив чисел: [1, 2, 3, 4, 5]. 1. С помощью метода map создайте новый массив, содержащий квадраты этих чисел. Выведите результат. 2. С помощью map создайте массив строковых представлений этих чисел (например, "1", "2", …). Выведите результат.
7	Дан массив чисел: [12, 5, 8, 130, 44, 3, 17]. Используя метод filter, создайте три новых массива: • только чётные числа; • только числа больше 10; • только числа, которые делятся на 3 без остатка. • Выведите каждый полученный массив в консоль.
8	Дан массив возрастов: ages = [18, 22, 15, 30, 25]. 1. С помощью some проверьте, есть ли в массиве хотя бы один элемент меньше 18. Выведите true или false. 2. С помощью every проверьте, все ли элементы больше 10. Выведите результат. 3. Используя find, найдите первый элемент, который больше 20. Выведите его. 4. Что вернёт метод find, если ни один элемент не удовлетворяет условию? Проверьте это на массиве [1, 2, 3] с условием >10. Выведите результат.
9	Дан массив чисел: [5, 10, 15, 20, 25]. 1. С помощью reduce вычислите сумму всех элементов. Начальное значение суммы возьмите 0. Выведите результат. 2. С помощью reduce найдите максимальное число в массиве. Подсказка: в функции-редьюсере сравнивайте текущее значение с накопленным. Выведите максимальное число.
10	Дан массив чисел: [3, 30, 10, 2, 22, 5]. 1. Отсортируйте его по умолчанию (без функции сравнения) и выведите результат. Объясните (в комментарии или устно), почему порядок может быть неожиданным. 2. Отсортируйте числа по возрастанию, передав в sort функцию сравнения для чисел. Выведите массив. 3. Отсортируйте по убыванию. Выведите массив. 4. Дан массив строк: ['яблоко', 'Банан', 'апельсин', 'груша']. Отсортируйте его без учёта регистра (подсказка: в функции сравнения приведите оба элемента к одному регистру через toLowerCase()). Выведите результат.
11	1. Напишите обычную функцию (function declaration) sum(a, b), которая возвращает сумму двух чисел. Вызовите её с числами 5 и 7, результат выведите в консоль. 2. Перепишите эту функцию как стрелочную (arrow function) и сохраните в переменную sumArrow. Вызовите её с числами 3 и 8, выведите результат.

	3. Напишите функцию greet(name = "гость"), которая возвращает строку "Привет, <имя>!". Если аргумент не передан, используется значение "гость". Проверьте работу с аргументом и без. 4. Напишите функцию sumAll, которая принимает любое количество чисел (используйте оператор ... в параметрах) и возвращает их сумму. Проверьте на наборах: sumAll(1, 2, 3) и sumAll(10, 20, 30, 40).
12	Дан массив чисел: [10, 20, 30, 40, 50, 30]. 1. Найдите индекс первого вхождения числа 30 с помощью метода indexOf. Выведите результат. 2. Проверьте, есть ли в массиве число 100, используя метод includes. Выведите true или false. 3. С помощью indexOf найдите индекс числа 30, начиная поиск с индекса 4 (второй аргумент метода). Выведите результат.
13	Даны два массива: group1 = ["Анна", "Иван", "Мария"] group2 = ["Петр", "Елена", "Дмитрий"] 1. Объедините их в новый массив allStudents с помощью метода concat. Выведите результат. 2. Объедините те же массивы с помощью оператора расширения ... (создайте новый массив). Выведите результат. 3. Добавьте в начало объединённого массива с помощью оператора расширения ещё одного студента, например, "Сергей". Выведите новый массив.
14	Дан массив букв: letters = ["а", "б", "в", "г", "д"]. 1. Создайте копию массива lettersCopy, используя метод slice(). 2. Примените метод reverse к копии. Выведите исходный массив и перевёрнутую копию. Убедитесь, что исходный массив не изменился. 3. Напишите, почему важно было создать копию перед вызовом reverse?
15	Дан массив с координатами точки: point = [15, 25, 35, 45]. 1. Используя деструктуризацию, извлеките первые два элемента в переменные x и y. Выведите их. 2. Извлеките первый элемент в переменную first, а остальные элементы соберите в переменную rest с помощью оператора .... Выведите first и rest. 3. Пропустите второй элемент при деструктуризации (например, [a, , b] = point). Выведите a и b.

Контрольные вопросы:

1. Что вернет typeof null?
2. Как проверить, есть ли у объекта obj ключ age?
3. Что будет в консоли: console.log(typeof \1,2,31,2,3)?
4. Можно ли вызвать функцию, объявленную как const foo = function() {}, до строки её объявления?

5. Какой метод массива создает новый массив, преобразуя каждый элемент с помощью функции?
6. Что вернет Array.isArray(\1,2,31,2,3)?
7. Как объявить функцию с параметрами a и b по умолчанию равными 0?
8. Что выведет console.log(typeof function(){})?
9. Для чего используется метод filter() у массива?
10. Как получить массив всех ключей объекта user?
11. Какой оператор используется для удаления свойства из объекта?

Лабораторная работа № 16. Продолжение JavaScript. Проектная работа. Сервис To-do.

Цель работы: продолжение изучения работы с JavaScript, выполнение проектной работы.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 7, темы 4-7. Обработка событий – уроки 1-6. Работа с формами – уроки 1-8. Хранилище. localStorage и sessionStorage – уроки 1-5. Проектная работа. Сервис To-do – уроки 1-3.

Изучите следующие разделы: **Обработка событий.** Обработка событий. Введение. События клавиатуры, обработчики событий и объект event. События мыши и объект event. Метод addEventListener. Как работают события. Заключение. **Работа с формами.** Работа с формами. Введение. Доступ к форме из JavaScript. Отправка формы. Событие submit. Получение элементов форм. Доступ к значениям элементов форм. События change и input. Методы reset и submit. Работа с формами. Заключение. **Хранилище. localStorage и sessionStorage.** Хранилище. Введение. Что такое хранилище в браузере. Какие бывают виды и чем отличаются. Локальное хранилище. localStorage. Сессионное хранилище. sessionStorage. Хранилище. Заключение.

Проектная работа. Сервис To-do. Проектная работа. Сервис To-do. Чеклист к проекту Сервис To-Do. Сдача проектной работы.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте пустой блок на странице. Добавьте на него обработчик события mousedown. В этом обработчике с помощью объекта event определите, какая кнопка мыши была нажата (левая, средняя или правая). Выводите название кнопки в этом блоке (например, "Нажата левая кнопка").
2	Создайте на странице счетчик, который показывает, сколько раз вы открывали эту вкладку в рамках текущей сессии (пока не закрыли браузер). Используйте sessionStorage. При каждом обновлении страницы счетчик должен увеличиваться на 1. При закрытии и открытии новой вкладки счет должен начаться заново.
3	Создайте форму с одним обязательным полем "Название задачи" и кнопкой "Добавить". При отправке формы (submit) проверяйте, не пустое ли поле. Если поле пустое, выводите предупреждение (alert) "Поле не может быть пустым!" и отменяйте отправку. Если поле заполнено, просто выводите в консоль текст "Успешно" и чистите форму (метод reset).
4	Создайте кнопку "Очистить всё". При клике на неё должен полностью очищаться localStorage. Также добавьте на страницу какое-нибудь сообщение, которое показывает, что в хранилище сейчас пусто (проверяйте при загрузке и после клика).
5	Создайте блок, в котором отображается текущее время в формате ЧЧ:ММ:СС. Используйте setInterval, чтобы обновлять время каждую секунду. Добавьте кнопку «Стоп», которая останавливает обновление (очищает интервал).

6	Создайте страницу с полем ввода и кнопкой "Сохранить". Когда пользователь вводит имя и нажимает кнопку, сохраняйте это имя в localStorage. При перезагрузке страницы проверяйте: если в хранилище уже есть имя, выводите его где-нибудь на странице (например, "С возвращением, {ваше ФИО}!").
7	Создайте несколько полей ввода (<input>). Используя события focus и blur, сделайте так, чтобы поле, в котором в данный момент находится курсор, меняло цвет фона на светло-желтый, а когда пользователь уходит (фокус теряется) — возвращало белый цвет.
8	Создайте поле <input type="text">. Снизу от него создайте пустой параграф. Как только пользователь нажимает клавишу на клавиатуре (событие keydown), в этом параграфе должен появляться текст: "Была нажата клавиша: [Symbol]". Вместо [Symbol] подставьте реальную клавишу (используйте event.key).
9	Создайте форму с полем для email и кнопкой "Проверить". При клике на кнопку проверьте, введен ли символ "@" в значении поля. Если нет — выведите предупреждение (alert) "Некорректный email". Если есть — выведите сообщение "Email корректен".
10	Создайте поле ввода с типом password и чекбокс с подписью «Показать пароль». При установке флажка тип поля должен меняться на text, чтобы пароль стал видимым, а при снятии — обратно на password.
11	Создайте интерфейс: поле ввода и кнопка «Добавить». При клике товар добавляется в маркированный список ( ), а также в массив, который хранится в localStorage. При перезагрузке страницы список восстанавливается из хранилища. Добавьте кнопку «Очистить всё», которая удаляет все элементы из DOM и из localStorage.
12	Создайте интерфейс: поле ввода для секунд, кнопка «Установить таймер» и параграф для отображения оставшегося времени. При клике на кнопку начинайте обратный отсчёт. Когда таймер достигает 0, покажите всплывающее сообщение (alert) и выведите на страницу текст «Время вышло!». Добавьте кнопку «Сброс», которая останавливает текущий отсчёт и очищает поле ввода.
13	Создайте три круга (красный, жёлтый, зелёный), расположенных вертикально. Изначально все круги серые. Под ними разместите кнопку «Следующий». При каждом клике загорается следующий цвет по правилам светофора: красный → жёлтый → зелёный → красный. Активный круг должен быть ярким, остальные — бледными.
14	Добавьте на страницу поле ввода типа date (дата рождения) и кнопку «Рассчитать возраст». При клике вычислите, сколько полных лет прошло от этой даты до сегодняшнего дня. Выведите результат в параграф. Если дата из будущего — покажите сообщение об ошибке.
15	Создайте HTML-страницу с кнопкой. При клике на кнопку в консоль должно выводиться сообщение «Привет, мир!». 1. Создайте файл index.html с базовой структурой.

	2. Добавьте кнопку с текстом «Нажми меня» и присвойте ей id="helloBtn". 3. В теге <script> найдите кнопку по id с помощью document.getElementById('helloBtn'). 4. Используйте метод addEventListener, чтобы повесить обработчик события click. 5. В обработчике напишите console.log("Привет, мир!").

Контрольные вопросы:

1. В каком свойстве объекта event хранится название нажатой клавиши (с учетом раскладки)?
2. Какое свойство объекта event у событий мыши содержит координаты курсора относительно окна браузера?
3. Какой параметр нужно передать в addEventListener, чтобы обработчик сработал только один раз?
4. Что такое всплытие события (event bubbling)?
5. Какое свойство объекта event указывает на элемент, на котором произошло событие?
6. Для чего используется метод preventDefault?
7. Какое преимущество дает делегирование событий?
8. Какое событие срабатывает у текстового поля только после потери фокуса, если значение изменилось?
9. В каком формате хранятся данные в localStorage и sessionStorage?
10. Как превратить объект JavaScript в строку для сохранения в хранилище?

Лабораторная работа № 17. Сборка кода и модульность. Валидация форм. Проектная работа «Mesto. Валидация форм и сборка кода».

Цель работы: изучения работы сборки кода и модульность, выполнение проектной работы «Mesto. Валидация форм и сборка кода».

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). Модуль 8, темы 1-6. Дебаггинг JavaScript – уроки 1-7. Модули в JS – уроки 1-8. Сборка кода на Vite – уроки 1-6. Валидация форм – уроки 1-8. Регулярные выражения – уроки 1-2. Проектная работа «Mesto. Валидация форм и сборка кода» – уроки 1-4.

Изучите следующие разделы: **Дебаггинг JavaScript.** Дебаггинг JavaScript. Введение**.** Как читать ошибки**.** Типы ошибок**.** Логические ошибки и console.log**.** Поиск документации**.** Отладка через debugger**.** Дебаггинг JavaScript. Заключение. **Модули в JS.** Введение**.** IIFE. Инкапсуляция и модули. Что такое модуль и как его использовать**.** Директивы export и import**.** Особенности работы модулей в браузере**.** Локальный сервер**.** Модули в JS. Заключение. **Сборка кода на Vite.** Введение**.** Библиотека пакетов NPM**.** Что такое сборка**.** Подключение сборщика Vite к проекту**.** Как подключать ресурсы в проекте с Vite и конфигурировать сборку**.** Заключение. **Валидация форм.** Введение**.** Встроенная браузерная валидация форм**.** Валидация с JavaScript**.** Связываем JS-методы валидации с DOM**.** Валидация нескольких полей и форм**.** Взаимодействие с другими элементами DOM**.** Введение в регулярные выражения и кастомные сообщения об ошибках**.** Валидация форм. Заключение. **Регулярные выражения.** Регулярные выражения**.** Применение регулярных выражений во фронтенде. **Проектная работа «Mesto. Валидация форм и сборка кода».** Проектная работа «Mesto. Валидация форм и сборка кода»**.** Чек-лист проектной работы «Mesto. Валидация форм и сборка кода»**.** Сдача проектной работы**.** Обратная связь о курсе.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с

выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Напишите функцию divide(a, b), которая возвращает результат деления a на b. Если b равно 0, функция должна выбрасывать ошибку throw new Error('Деление на ноль!'). Вызовите эту функцию внутри конструкции try...catch и выведите сообщение ошибки в консоль (через console.error).
2	В коде ниже допущена ошибка. Скопируйте его в консоль, прочитайте сообщение об ошибке и исправьте её. const greeting = "Привет, мир! " console.log(greeting);
3	Создайте HTML-страницу с полем ввода пароля и кнопкой. При нажатии на кнопку проверяйте, что длина пароля не менее 6 символов. Если пароль слишком короткий, выводите под полем сообщение красным цветом. Если подходит — зелёное «Пароль принят».
4	Создайте два файла: greeting.js и main.js. В greeting.js объявите функцию sayHello(name), которая возвращает строку Привет, {name}!. Экспортируйте её. В main.js импортируйте функцию и вызовите её с любым именем, результат выведите в консоль. Подключите main.js к HTML как модуль (type="module").
5	В проект Vite добавьте в папку public любую картинку (например, cat.jpg). В файле main.js напишите код, который создаст элемент <img> и установит ему путь так, чтобы картинка отобразилась на странице.
6	Программа должна посчитать, сколько будет 10 + 5, но выводит неправильный ответ. Найдите ошибку и исправьте её. const a = 10; const b = 5; const result = a * b; // Здесь ошибка! console.log('Результат сложения:', result);
7	Напишите регулярное выражение для проверки номера телефона в формате +7-999-123-45-67, где после +7 идут группы цифр. Используйте \d и {...}. Проверьте его на своей строке.
8	Напишите функцию isHexColor(color), которая проверяет, является ли переданная строка валидным HEX-кодом цвета (например, #FFF, #fff, #A1B2C3). Допускаются как трёхзначные, так и шестизначные варианты, символ # обязателен. Используйте регулярное выражение с учётом регистра. Протестируйте на строках "#FFF", "#123abc", "#GHI", "123456".
9	В проекте Vite создайте файл style.css в папке src. Напишите в нём правило, которое делает цвет фона всей страницы светло-серым (background-color: lightgray;). Импортируйте этот файл в main.js (просто import './style.css';) и запустите проект. Убедитесь, что стили применились.

10	У вас есть код. С помощью console.log() выясните, какое значение хранится в переменной userAge и какого оно типа (typeof). let userAge = "25"; console.log(typeof userAge); // <- добавьте эту строку console.log(userAge); // <- добавьте эту строку if (userAge === 18) { console.log("Доступ разрешен"); } else { console.log("Во сколько же вам лет?"); }
11	Создайте форму с полем для ввода номера телефона. Используя Constraint Validation API, напишите валидацию, которая проверяет, что номер содержит ровно 11 цифр (можно с пробелами и дефисами). Если номер не соответствует формату, установите своё сообщение об ошибке с помощью setCustomValidity(). При попытке отправки формы с неверным номером должно появляться это сообщение.
12	Создайте файл formatters.js, который экспортирует две функции: formatPrice (добавляет знак рубля) и formatDate (приводит дату к формату ДД.ММ.ГГГГ). В файле main.js импортируйте эти функции, но дайте им более короткие имена: price и date. Используйте синтаксис import { formatPrice as price, formatDate as date }. Вызовите обе функции с тестовыми данными.
13	Напишите регулярное выражение, которое найдет все гласные буквы (а, е, ё, и, о, у, ы, э, ю, я) в строке "Привет, мир!" (используйте метод match).
14	Создайте простую HTML-форму с одним полем ввода и кнопкой "Отправить". Сделайте это поле обязательным, добавив атрибут required. Попробуйте отправить форму, не заполнив поле. Что произошло?
15	Создайте два HTML-файла. В каждом подключите один и тот же JS-файл как модуль (<script type="module" src="script.js">). Внутри script.js просто напишите console.log('Модуль загружен');. Откройте оба HTML-файла. Сколько раз выполнится код из script.js? (Проверьте в консоли каждого файла).

Контрольные вопросы:

1. Что такое IIFE (Immediately Invoked Function Expression) в JavaScript?
2. Как называется процесс удаления из кода всех пробелов, комментариев и лишних символов для уменьшения размера файла?
3. Свойство validity.valid вернёт true, если поле...
4. Какой метод объекта event используется для отмены действия браузера по умолчанию (например, перезагрузки страницы при отправке формы)?
5. Что произойдет при выполнении кода console.log(typeof null);?

6. Какое значение нужно вернуть из функции валидации, чтобы кнопка отправки формы стала активной (разблокированной)?
7. Какой атрибут нужно добавить к тегу <script>, чтобы браузер воспринимал его как модуль?
8. Какой HTML-атрибут делает поле обязательным для заполнения?
9. Какое событие срабатывает при каждом изменении значения в текстовом поле (ввод символа, вставка)?
10. Для чего нужен файл package.json в проекте?

Лабораторная работа № 18. Работа с API в JS. Финальный проект.

Цель работы: изучения работы с API в JS, выполнение финального проекта.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). **Модуль 9**, темы 1-2. Асинхронность – уроки 1-9. Работа с API – уроки 1-10. **Модуль 10**, темы 1-2. Проектная работа Mesto. Интеграция с API и деплой проекта – уроки 1-3. Пути дальнейшего развития фронтенд-разработчика – урок 1.

Изучите следующие разделы:

Модуль 9: **Асинхронность.** Асинхронные операции**.** Колбэки**.** Асинхронные колбэки**.** Таймеры**.** Event Loop**.** Promise**.** Продвинутый JavaScript. Асинхронность. Заключение**.** Дополнительные материалы**.** Обратная связь о курсе. **Работа с API.** Работа с API. Введение**.** Протокол HTTP**.** Запросы из JavaScript**.** Формат JSON**.** HTTP-запрос**.** Ответ**.** Инструменты: вкладка Network**.** Дополнительно: стандартная отправка формы**.** Работа с API. Заключение**.** Дополнительные материалы темы.

Модуль 10: **Проектная работа Mesto. Интеграция с API и деплой проекта.** Проектная работа Mesto. Работа с сервером**.** Чек-лист для самопроверки. Проектная работа Mesto. Работа с сервером**.** Обратная связь о курсе. **Пути**

дальнейшего развития фронтенд-разработчика. Пути дальнейшего развития фронтенд-разработчика.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Создайте на странице заголовок <h1>. Напишите код, который будет менять его цвет с черного на красный и обратно каждые 2 секунды.
2	оздайте функцию fetchData, которая принимает колбэк. Внутри функции через setTimeout (1 секунда) сымитируйте получение данных (например, объект {name: "Иван"}) и вызовите колбэк, передав в него эти данные. В колбэке выведите данные в консоль.
3	Создайте промис, который через 1 секунду с вероятностью 50% завершится успехом ("Повезло"), а с вероятностью 50% — ошибкой ("Не повезло"). Обработайте и тот, и другой случай (используйте .then() и .catch()).
4	Создайте объект с данными новой карточки (например, { name: 'Моё любимое место', link: 'https://ссылка.jpg' }). Отправьте POST-запрос на сервер, чтобы сохранить эту карточку.
5	Напишите функцию filterArray(arr, callback). Она принимает массив чисел и функцию-условие. Функция должна создавать новый массив, в который попадают только те элементы, для которых колбэк вернул true. Проверьте её, например, с колбэком (num) => num % 2 === 0 (оставить только чётные числа).
6	Динамическая загрузка скрипта (Promise). Напишите функцию loadScript(src), которая создает тег <script>, добавляет его на страницу и возвращает промис. Промис должен выполняться при успешной загрузке скрипта и отклоняться при ошибке (например, если скрипт не найден). Проверьте, загрузив какой-нибудь внешний скрипт

	(например, https://code.jquery.com/jquery-3.6.0.min.js), и в then выведите сообщение "Скрипт загружен".
7	Сделайте поле ввода (input type="number") и кнопку «Запустить отсчёт». Пользователь вводит число секунд, и на странице начинается обратный отсчёт (каждую секунду число уменьшается на 1), когда достигает 0 — показывается сообщение «Время вышло!».
8	Создайте промис, который через 2 секунды успешно завершается со значением "Данные получены". Обработайте его .then().
9	Создайте простую HTML-карточку: картинка (можно заглушка), название и кнопка «❤️ Лайк». При первом клике на кнопку она должна менять цвет на красный и отправлять PUT-запрос (можно использовать https://jsonplaceholder.typicode.com/posts/1 как заглушку, но сам запрос не изменит данные). При втором клике цвет возвращается к исходному и отправляется DELETE-запрос. В консоль выводите, какой метод был бы отправлен на реальном сервере.
10	Напишите код для отправки PATCH-запроса на https://mesto.mockedapi.com/users/me (или любой другой тестовый эндпоинт, поддерживающий PATCH). В теле запроса передайте новый аватар: { avatar: 'https://новая-ссылка.jpg' }.
11	Создайте HTML-страницу с элементом 0 и двумя кнопками: «Старт» и «Стоп». При нажатии «Старт» число в div должно увеличиваться на 1 каждую секунду. При нажатии «Стоп» увеличение должно прекращаться.
12	Создайте два промиса: первый резолвится через 2 секунды со значением «Черепаха», второй — через 1 секунду со значением «Заяц». Используйте Promise.race, чтобы определить, кто быстрее, и выведите результат в консоль.
13	Создайте функцию transformArray(arr, callback), которая возвращает новый массив, где каждый элемент — результат вызова колбэка для исходного элемента. Например, с колбэком (x) => x * x получим массив квадратов чисел.
14	Напишите функцию processNumbers(a, b, callback). Функция должна складывать числа, а затем передавать результат в колбэк. Вызовите эту функцию, а в колбэке умножьте полученный результат на 2 и выведите в консоль.
15	Напишите в консоли браузера следующий код и объясните (устно или в комментарии), в каком порядке появятся цифры и почему. console.log(1); setTimeout(() => console.log(2), 0); console.log(3); setTimeout(() => console.log(4), 100); console.log(5);

Контрольные вопросы:

1. Какое устройство в браузере отвечает за то, чтобы задачи из очереди колбэков попадали в стек вызовов (call stack), когда он освобождается?
2. Объект, который является «обещанием» (аналогом бипера в кафе) выполнить какой-либо асинхронный код в будущем, в JavaScript называется...
3. Какие два основных колбэка (состояния) принимает функция-исполнитель при создании нового Promise?
4. Какой метод промиса срабатывает, когда промис переходит в состояние "исполнен" (fulfilled/resolved)?
5. Какой тип таймера позволяет вызвать функцию один раз через заданный промежуток времени?
6. Какой тип таймера позволяет вызывать функцию многократно с заданным интервалом?
7. Какой метод используется для отмены выполнения таймера, созданного с помощью setInterval?
8. Что возвращает метод fetch()?
9. Где в инструментах разработчика браузера можно посмотреть список всех сетевых запросов, которые делает страница?
10. Какой формат является самым популярным для обмена данными между клиентом и сервером в текстовом виде?

Лабораторная работа № 19. Использование ИИ в разработке.

Цель работы: изучения использования ИИ в разработке.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). **Модуль 11**, темы 1-3. Введение в искусственный интеллект – уроки 1-5. ИИ для обучения и исследований – уроки 1-3. Базовые кейсы в разработке – уроки 1-4.

Изучите следующие разделы: **Введение в искусственный интеллект.** Эволюция ИИ**.** Как работают нейросети. Безопасность и этика**.** Выбор инструментов**.** Основы промт-инжиниринга. **ИИ для обучения и исследований.** Поиск информации и объяснение теории Генерация учебных задач для практики**.** Планирование работы по проекту. **Базовые кейсы в разработке.** Генерация фрагментов кода**.** Отладка кода**.** Документирование**.** Генерация тестовых данных и автотестов.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	**Задание**
1	Проведите мини-исследование: какие инструменты на базе ИИ существуют для помощи фронтенд-разработчику? Составьте таблицу из 5–7 инструментов (например, ChatGPT, GitHub Copilot, Tabnine, Codeium, Midjourney, Uizard). Для каждого укажите: назначение, основные функции, наличие бесплатной версии. Сделайте вывод, какой инструмент вам хотелось бы попробовать в первую очередь и почему.
2	Вы хотите добавить на свой сайт уникальные изображения, но не умеете рисовать. Используйте любой бесплатный ИИ-генератор картинок (например, DALL-E через Bing, Kandinsky, Stable Diffusion через онлайн-демо). Придумайте промпт для генерации изображения "кота-программиста за ноутбуком в стиле киберпанк". Сохраните полученное изображение. Затем попросите текстового ИИ-ассистента написать HTML-код для вставки этого изображения на страницу с подписью и адаптивными размерами. Объясните, как можно оптимизировать загрузку изображения (например, использовать атрибуты srcset или форматы WebP).

3	Найдите видео-лекцию по фронтенду (например, "JavaScript Fetch API") длительностью 10–15 минут. Скопируйте ссылку и попросите ИИ-ассистента с функцией работы с контекстом (например, NotebookLM от Google или загрузив субтитры в Gemini) составить краткий конспект лекции, выделив ключевые понятия, примеры кода и вопросы для самопроверки. Оцените, насколько конспект отражает суть видео.
4	Вы не дизайнер, но вам нужна простая иконка для вашего учебного проекта (например, иконка домика, шестерёнки, корзины). С помощью промпта опишите ИИ (тому, который понимает код), как должна выглядеть эта иконка, чтобы он сгенерировал её на чистом HTML и CSS (например, с использованием псевдоэлементов). Попросите объяснить, как работает код.
5	Вы разрабатываете сайт для кофейни. Попросите ИИ подобрать гармоничную цветовую палитру из 4–5 цветов на основе базового цвета (#6F4E37 — "кофейный") или просто описания ("тёплые, уютные тона, напоминающие кофе и карамель"). Попросите выдать цвета в HEX-формате и кратко объяснить, почему они подходят друг другу. Затем используйте эту палитру, чтобы стилизовать простую карточку товара (можно взять из Задания 16) — замените цвета в CSS на новые. Оцените результат.
6	Попросите ИИ объяснить, что такое CSS-селекторы, но каждый раз меняйте роль: • как для пятилетнего ребёнка; • как для коллеги-разработчика; • как для студента, который готовится к экзамену. Сравните три объяснения. Какое вам понравилось больше всего и почему? Какие элементы роли повлияли на стиль и глубину ответа?
7	Найдите в интернете (например, на CodePen) интересный, но сложный для вас фрагмент кода (анимация на CSS, нестандартный JS-эффект). Скопируйте код и отправьте его ИИ с запросом: "Объясни построчно, что делает этот код. Особое внимание удели строкам [укажите строки, которые непонятны]". Запишите объяснение ИИ. Позволило ли оно вам лучше понять код?
8	Попросите ИИ создать шаблоны для GitHub Issues и Pull Requests (файлы .github/ISSUE_TEMPLATE.md и .github/PULL_REQUEST_TEMPLATE.md). Шаблоны должны включать разделы: описание проблемы/изменения, шаги для воспроизведения (для багов), среда, предлагаемое решение, чеклист. Это познакомит с практиками open source разработки.
9	Составьте запрос, чтобы ИИ создал простое приложение «Список дел» (To-Do list) на HTML, CSS и JavaScript с условиями: • код должен быть в одном файле (inline-стили и скрипт); • минималистичный дизайн (белый фон, чёрный текст); • не использовать сторонние библиотеки. Проверьте, выполнил ли ИИ все ограничения. Если возникли ошибки, попробуйте исправить их с помощью уточняющих запросов. В отчёте опишите, какие проблемы возникли и как вы их решили.
10	Попросите ИИ создать простой список задач (To-Do list), в котором можно менять порядок задач перетаскиванием (drag-and-drop). Используйте HTML5 Drag and Drop API или, как альтернативу, события

	мыши. Код должен позволять перетаскивать элемент и вставлять его на новое место. Объясните, как работает механизм. Проверьте в разных браузерах. Есть ли особенности?
11	Опишите ИИ идею лендинга (например, «Кофейня "Усатый бармен" предлагает свежую выпечку и авторский кофе»). Попросите ИИ составить поэтапный план разработки, включающий: • Какие разделы должны быть на странице (шапка, о нас, меню, контакты и т.д.). • Какие технологии использовать (HTML, CSS, возможно, немного JS). • В какой последовательности вёрстывать (сначала структура, потом стили, потом адаптивность). • Какие инструменты ИИ могут помочь на каждом этапе. Полученный план выпишите и отметьте, какие шаги вы уже умеете выполнять
12	Представьте, что вы хотите через полгода начать изучать React (библиотеку для интерфейсов). Попросите ИИ составить для вас 4-недельный план подготовки, учитывая, что вы знаете HTML, CSS и основы JavaScript. План должен включать конкретные темы, практические задания и ссылки на ресурсы (или названия). Оцените реалистичность плана.
13	Вы создали сайт. Попросите ИИ объяснить, какие форматы и размеры иконок нужны для корректного отображения favicon в браузерах и на мобильных устройствах (при добавлении на домашний экран). Затем попросите ИИ сгенерировать HTML-код для подключения всех этих иконок (link rel="icon", link rel="apple-touch-icon" и т.д.). Если у вас есть изображение (например, логотип), можно использовать онлайн-генератор favicon, но здесь задача — получить правильную разметку от ИИ. Вставьте код в свой проект.
14	Вы хотите добавить на страницу интерактивный рисунок. Попросите ИИ написать код на JavaScript с использованием Canvas API, который рисует простую сцену: например, домик, солнце и дерево. Код должен быть прокомментирован, объясняя каждую команду (beginPath, fillStyle, fillRect и т.д.). Затем попросите ИИ добавить анимацию: например, облако, движущееся по небу. Запустите код и убедитесь, что всё работает. Подумайте, как можно было бы сделать рисунок параметризуемым (менять цвета через переменные).
15	Представьте, что вы решили создать сайт-портфолио. Попросите ИИ составить поэтапный план работ на неделю: какие разделы должны быть на сайте, какие технологии использовать, в какой последовательности делать вёрстку, стилизацию, добавление интерактива. Полученный план выпишите. Затем задайте уточняющие вопросы, чтобы адаптировать план под свои предпочтения (например, «я хочу, чтобы был тёмный фон»). Запишите улучшенный план.

Контрольные вопросы:

1. Что такое «локальное развёртывание» ИИ-моделей?

2. Какой тип ИИ-инструментов представляет собой виртуального члена команды, способного выполнять сложные задачи «под ключ»?
3. Кто несёт ответственность за ошибки в коде, сгенерированном ИИ?
4. Какое правило помогает предотвратить утечку конфиденциальных данных при использовании онлайн-ИИ?
5. Что такое XSS-уязвимость?
6. Какое ограничение ИИ связано с тем, что модель может «забыть» начало длинного диалога?
7. Что такое «галлюцинация» применительно к работе языковых моделей?
8. Какие два основных типа ИИ выделяют в зависимости от широты решаемых задач?
9. Для чего используется механизм внимания (self-attention) в трансформерах?

Какая архитектура нейросетей стала основой для современных больших языковых моделей (GPT)?

Лабораторная работа № 20. Typescript.

Цель работы: изучение Typescript.

Теоретическое обоснование

Для изучения теоретической части перейдите на платформу Яндекс по ссылке Яндекс Практикум (yandex.ru). **Модуль 12**, темы 1-2. Введение в TypeScript – уроки 1-9. Основы TypeScript – уроки 1-8.

Изучите следующие разделы: **Введение в TypeScript.** TypeScript: не путать с JavaScript**.** Динамическая типизация**.** Самодокументированный код через JSDoc**.** Статическая типизация и TypeScript**.** Проверка типов во время исполнения**.** Полнота программного интерфейса**.** Добавляем TypeScript в проект**.** Настройка TypeScript**.** Инструментарий TS**. Основы TypeScript.** Введение**.** Приведение типов**.** Дженерики**.** Типизация DOM-элементов и их

событий. Типизация стандартных объектов JS. Файлы деклараций d.ts. Использование и создание библиотек. Заключение.

Методические указания к практическому выполнению

1. Выполните практические задания к каждой теме на платформе Яндекс.

2. Составьте отчет о выполненной работе. В отчете отразите основные теоретические аспекты и скриншоты с Яндекс Практикума и с выполненным индивидуальным заданием (листинг код, результат листинга кода), ответы на контрольные вопросы.

3. Выполните индивидуальные задания по вариантам. Итоги проделанной работы пришлите архивом вашему преподавателю.

Индивидуальные задания

№ Варианта	Задание
1	Добавляем TypeScript в проект, Инструментарий TS. Цель: Убедиться, что на компьютере установлены Node.js и npm, инициализировать пустой проект. 1. Откройте терминал (командную строку). 2. Проверьте, установлен ли Node.js: выполните node -v. Если нет — скачайте и установите с официального сайта. 3. Проверьте npm: npm -v. 4. Создайте новую папку для проекта, например ts-intro. 5. Перейдите в неё и выполните npm init -y (создастся package.json). 6. Установите TypeScript глобально или локально в проект: npm install --save-dev typescript. 7. Убедитесь, что TypeScript установлен: npx tsc --version. Результат: В папке проекта появился package.json, TypeScript доступен для компиляции.
2	Самодокументированный код через JSDoc. Цель: Улучшить читаемость JS-кода и получить подсказки в редакторе с помощью комментариев JSDoc. 1. Создайте файл jsdoc.js. 2. Скопируйте функцию add из предыдущего задания. 3. Добавьте над функцией комментарий JSDoc: /** * Складывает два числа. * @param {number} a - Первое число * @param {number} b - Второе число * @returns {number} Сумма чисел

	*/ function add(a, b) { return a + b; } 4. Попробуйте вызвать функцию с разными аргументами. Если ваш редактор (VS Code) поддерживает TypeScript, он будет подсказывать ожидаемые типы. 5. Обратите внимание, что код всё ещё JS, и ошибки типов не возникают при запуске. Результат: Вы научились документировать типы в JS, но проверка остаётся только на уровне подсказок.
3	Основы TypeScript. Цель: Использовать опциональные поля (?) и объединение типов (union). 1. Расширьте интерфейс User, добавив опциональное поле phone?: string. 2. Создайте переменную status типа "active" | "inactive" | "banned" (union строк). 3. Добавьте поле status в интерфейс User. 4. Создайте несколько объектов User с разными статусами и без телефона. Результат: Вы понимаете, как сделать поле необязательным и как ограничить значение строки несколькими вариантами.
4	Типизация стандартных объектов JS. Цель: Использовать встроенные объекты JavaScript с типами TypeScript. 1. Создайте файл standard.ts. 2. Создайте переменную currentDate типа Date и присвойте new Date(). 3. Получите текущий год, вызвав метод getFullYear(). 4. Создайте функцию formatDate(date: Date): string, которая возвращает строку вида "ДД.ММ.ГГГГ". 5. Вызовите её с currentDate. Результат: Вы поработали с объектом Date, который автоматически типизирован.
5	Основы TypeScript. Цель: Научиться задавать необязательные параметры и значения по умолчанию. 1. В файле functions.ts создайте функцию greet: function greet(name: string, greeting: string = "Привет"): string { return `${greeting}, ${name}!`; } 2. Вызовите её с одним и двумя аргументами. 3. Добавьте опциональный параметр age?: number и выводите возраст, если он передан. Результат: Параметры по умолчанию и опциональные поля упрощают использование функций.
6	Проверка типов во время исполнения. Цель: Сравнить статическую проверку TypeScript с runtime-проверками через typeof. 1. Создайте файл runtime.ts. 2. Напишите функцию processInput(input: number | string): string: Если input — число, верните его удвоенное значение как строку. Если строка — верните её в верхнем регистре.

	3. Используйте typeof внутри функции для определения типа во время выполнения. 4. Вызовите функцию с разными аргументами. Результат: Даже с TypeScript иногда нужны проверки в рантайме, например, для union-типов.
7	Использование и создание библиотек, Файлы деклараций. Цель: Установить популярную библиотеку (например, lodash) и использовать её с типами. 1. Установите lodash и его типы: npm install lodash @types/lodash. 2. В TS-файле импортируйте _ из lodash. 3. Используйте метод _.capitalize для строки "hello world" и выведите результат. 4. Убедитесь, что TypeScript подсказывает методы и их типы. Результат: Вы подключили библиотеку и воспользовались готовыми типами.
8	Инструментарий TS. Цель: Скомпилировать весь проект в JS и запустить итоговый скрипт. 1. Убедитесь, что в tsconfig.json указаны "outDir": "./dist" и "rootDir": "./src". 2. Запустите компиляцию всего проекта: npx tsc. 3. В папке dist появятся все скомпилированные JS-файлы. 4. Выберите любой из них (например, dist/index.js) и запустите через Node: node dist/index.js. 5. Если использовались DOM-элементы, создайте HTML-файл, подключающий скомпилированный JS, и откройте в браузере. Результат: Весь проект собран, вы увидели результат работы TypeScript в действии.
9	Типизация функций с несколькими параметрами. Цель: Научиться явно указывать типы для нескольких параметров и возвращаемого значения функции. 1. Создайте файл functionParams.ts. 2. Напишите функцию greetUser, которая принимает два параметра: firstName: string и age: number. Функция должна возвращать строку вида: "Привет, {firstName}! Тебе {age} лет.". 3. Укажите тип возвращаемого значения после списка параметров: : string. 4. Вызовите функцию с корректными аргументами и выведите результат. 5. Попробуйте передать возраст строкой — убедитесь, что TypeScript укажет на ошибку. Результат: Функция строго проверяет типы входных данных и гарантирует возврат строки.
10	Кортежи (Tuples). Цель: Познакомиться с кортежами — массивами фиксированной длины и типов элементов. 1. Создайте файл tuples.ts. 2. Объявите тип Point как кортеж [number, number], представляющий координаты (x, y). 3. Создайте переменную start: Point = [0, 0]. 4. Напишите функцию calcDistance, которая принимает два кортежа Point и возвращает расстояние между ними (евклидово). Для вычисления можно использовать формулу: Math.sqrt((x2-x1)**2 + (y2-y1)**2).

	5. Проверьте функцию на двух точках. 6. Попробуйте добавить третий элемент в кортеж или изменить тип элемента — TypeScript выдаст ошибку. Результат: Вы научились использовать кортежи для хранения строго типизированных наборов значений фиксированной длины.
11	Readonly свойства. Цель: Использовать модификатор readonly для защиты полей объекта от изменений. 1. Создайте файл readonly.ts. 2. Определите интерфейс Book с полями: title: string author: string isbn: string 3. Поле year сделайте доступным только для чтения: readonly year: number. 4. Создайте объект книги, заполнив все поля. 5. Попробуйте изменить значение year — TypeScript выдаст ошибку. 6. Измените другие поля — это должно быть разрешено. Результат: Модификатор readonly защищает данные от случайного изменения.
12	Type Guards: typeof и in. Цель: Научиться сужать объединённые типы с помощью проверок typeof и оператора in. 1. Создайте файл typeGuards.ts. 2. Напишите функцию processValue(value: string | number): string: Если typeof value === "string", верните строку в верхнем регистре. Если typeof value === "number", верните строку "Число: X", где X — удвоенное число. 3. Создайте интерфейсы Cat (поле meow: boolean) и Dog (поле bark: boolean). 4. Создайте объединённый тип Pet = Cat | Dog. 5. Напишите функцию makeSound(pet: Pet): string: Если "meow" in pet, верните "Мяу!". Если "bark" in pet, верните "Гав!". 6. Проверьте обе функции. Результат: Type Guards позволяют безопасно работать с объединёнными типами.
13	Расширение интерфейсов (extends). Цель: Создать иерархию интерфейсов с помощью наследования. 1. Создайте файл extends.ts. 2. Определите базовый интерфейс Person: interface Person { name: string; age: number; } 3. Создайте интерфейс Student, расширяющий Person и добавляющий поле course: string. 4. Создайте интерфейс Teacher, расширяющий Person и добавляющий поле subject: string. 5. Создайте объекты студента и преподавателя, удовлетворяющие своим интерфейсам. 6. Напишите функцию, которая принимает Person и выводит имя. Убедитесь, что в неё можно передать и студента, и преподавателя.

	Результат: Наследование интерфейсов позволяет строить иерархии типов.
14	Типизация деструктуризации объекта. Цель: Научиться указывать типы при деструктуризации параметров функции. 1. Создайте файл destructuring.ts. 2. Определите интерфейс Product с полями: id: number, title: string, price: number. 3. Напишите функцию printProduct, которая принимает объект Product и деструктурирует его прямо в параметрах: `function printProduct({ id, title, price }: Product): void {` `console.log(`${id}: ${title} стоит ${price} руб.`);` `}` 4. Создайте объект продукта и передайте его в функцию. 5. Добавьте ещё один параметр — скидку discount?: number и обработайте её. Результат: Вы научились типизировать деструктурированные параметры, делая код чище.
15	Типизация функций обратного вызова (callback). Цель: Создать функцию, которая принимает другую функцию (колбэк) с заданной сигнатурой. 1. Создайте файл callback.ts. 2. Определите тип Operation как функцию, принимающую два числа и возвращающую число: `type Operation = (a: number, b: number) => number;` 3. Напишите функцию calculate, которая принимает три аргумента: два числа и колбэк типа Operation. Функция должна возвращать результат вызова колбэка. 4. Создайте несколько колбэков: сложение, вычитание, умножение. 5. Проверьте функцию calculate с разными операциями. 6. Попробуйте передать колбэк с неверной сигнатурой (например, принимающий строку) — TypeScript укажет на ошибку. Результат: Вы строго типизировали колбэки, что гарантирует правильное использование.

Контрольные вопросы:

1. Какой встроенный объект JavaScript представляет коллекцию ключ-значение с любыми типами ключей?
2. Какой встроенный объект JavaScript представляет коллекцию уникальных значений?
3. Для чего используется оператор instanceof?
4. Для чего применяются защитники типов (type guards)?
5. Какой синтаксис используется для объявления дженерик-функции?
6. Что такое дженерики в TypeScript?

7. Для чего используются файлы с расширением .d.ts?
8. Что такое JSDoc?
9. Какой параметр tsconfig.json задаёт директорию для скомпилированных файлов?
10. Для чего нужен параметр strict в tsconfig.json?

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

для выполнения самостоятельной работы
по дисциплине «Информационные технологии и программирование»
для студентов направления 09.03.04 Программная инженерия
(направленность(профиль) «Разработка и сопровождение
программного обеспечения»)

СОДЕРЖАНИЕ

Введение

1. Цель и задачи освоения дисциплины

Цель дисциплины являются формирование у студентов профессиональных компетенций, основанных на овладении фроненд-технологий, интернет-программирования, изучение принципов обмена данными в сети интернет.

Задачи освоения дисциплины:

1. изучение технологий, методов разработки web-приложений;
2. приобретение навыков разработки frontend.

2. Место дисциплины в структуре ОП ВО бакалавриата

Дисциплина «Информационные технологии и программирование» относится к дисциплинам (модулям) Б1, обязательная часть, формируемая участниками образовательных отношений и изучается в 1, 2 семестрах.

3. Связь с предшествующими дисциплинами

4. Связь с последующими дисциплинами

Освоение данной дисциплины является необходимой основой для изучения дисциплин:

- Программная инженерия
- Архитектура и конструирование программного обеспечения
- Web-технологии

В результате освоения дисциплины обучающийся должен:

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
ОПК-2 Способен использовать современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности	ИД2 $_{ПК-2}$ Использует современные информационные технологии и программные средства, в том числе отечественного производства, при решении задач профессиональной деятельности.	Классифицирует программное обеспечение (системное, прикладное, инструментальное) и объясняет назначение основных компонентов вычислительной системы
ОПК-5 Способен инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем	ИД-1$_{ОПК-5}$ Инсталлирует программное обеспечение для информационных систем	Выполняет установку и первичную настройку интегрированных сред разработки (IDE), компиляторов и интерпретаторов
ПК-1 Способен осуществлять управление программными проектами: планировать,	ИД1$_{ПК-1}$ Применяет методы декомпозиции работ (WBS), оценки трудоёмкости (PERT, Planning Poker) и календарного планирования (диаграмма Ганта, критический путь) для формирования плана проекта	Демонстрирует базовые навыки работы с Git: инициализирует репозиторий, выполняет коммиты, создает и сливает ветки, публикует код в удаленном репозитории

контролировать сроки и ресурсы, применять системы контроля версий и инструменты управления задачами		
УК-6 Способен управлять своим временем, выстраивать и реализовывать траекторию саморазвития на основе принципов образования в течение всей жизни	ИД1$_{УК-6}$ Устанавливает личные и профессиональные цели в соответствии с уровнем своих ресурсов и приоритетов действий, для успешного развития в избранной сфере профессиональной деятельности	Планирует последовательность выполнения заданий по дисциплине, распределяет время на изучение теоретического материала, подготовку к лабораторным работам и экзамену

5. Организационно-методические рекомендации по освоению дисциплины

Самостоятельная работа студентов является важнейшим условием формирования научного способа познания. Она проводится накануне каждого лабораторного занятия и включает изучение необходимого для выполнения лабораторной работы теоретического материала, а также подготовку отчета по выполненным на лабораторном занятии заданиям.

Подготовленный материал оформляется в виде отчета.

Самостоятельные занятия (СЗ) являются одной из активных форм обучения.

Самостоятельные занятия по дисциплине «Информационные технологии и программирование» имеют целью:

– закрепить и углубить знания, полученные студентами на лекциях и в процессе лабораторных занятий;

– привить практические навыки при разработке, тестировании Web-приложений.

Самостоятельные занятия проводятся в специализированных аудиториях, оборудованных СВТ и возможностью пользования Интерент-ресурсами, в библиотеке, а также дома.

Предлагаемые методические рекомендации содержат информацию для студентов, необходимую при подготовке и проведении лабораторных занятий по дисциплине «Web-технологии».

6. Методические указания по выполнению самостоятельной работы студентов

Методические указания включают следующие разделы:

– цель работы;

– задание, которое должно быть выполнено студентом в результате проведения самостоятельной работы;

– варианты индивидуальных заданий;

– основные теоретические положения, необходимые для выполнения задания, они должны быть краткими и содержать ссылки на литературу, в которой эти положения изложены в объеме, достаточном для выполнения самостоятельной работы;

– этапы выполнения задания с указанием конкретных сроков выполнения каждого из этапов и всего задания в целом;

– требования к оформлению графической и текстовой части самостоятельной работы;

– пример выполнения одного из вариантов задания и оформления отчета;

– библиографический список использованных источников

7. Содержание самостоятельной работы

Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы представлено в таблице 2.

Таблица 2 – Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы

№	Наименование разделов и тем дисциплины, их краткое содержание; вид самостоятельной работы	Форма контроля
	2 семестр	
1	подготовка к лабораторным занятиям	защита
2	Самостоятельная разработка веб-сайта (веб-приложения)	Защита проекта на зачете

Наименование тем дисциплины, цель, форма контроля, задания, требования к оформлению , перечень литературных источников представлен в таблице3.

Таблица 3 – Наименование тем дисциплины, цель, форма контроля, задания и требования к оформлению самостоятельной работы

Название темы	Цель	Форма контроля	Задания для СРС
Подготовка к лекциям и лабораторным занятиям	изучить основные технологии разработки веб-приложения для сервер-клиентской архитектуры	индивидуальное собеседование	Сформулировать ответы на контрольные вопросы
Разработка собственного веб-сайта	Получить навыки разработки веб-приложения со стороны frontend		сформулировать и ответить на вопросы для контроля владения компетенциями данного раздела программы

8. Примерная тематика заданий для самостоятельной работы студентов

Примерные варианты веб-сайта (веб-приложения) по дисциплине «Информационные технологии и программирование»:

1. Разработка новостного блога
2. Разработка интернет-магазина (любая тематика)
3. Разработка личного кабинета специалиста (любая тематика)

Любая тематика интересная студенту

9. Организация контроля знаний студентов

9. 1 Формы контроля знаний студентов

Контроль и оценка знаний, умений и навыков студентов осуществляется на лабораторных занятиях, консультациях, получении зачета с оценкой. В ходе контроля знаний преподаватель оценивает понимание студентом содержания дисциплины «Информационные технологии и программирование», его способность разрабатывать и тестировать веб-приложение, применять веб-технологии в разработке.

Контроль знаний студентов может осуществляться в следующих формах:

– текущий контроль знаний;

– итоговый контроль знаний.

Текущий контроль знаний студентов имеет целью: дать оценку работы каждого студента по усвоению им учебного материала, выявить недостатки в его подготовке и оказать практическую помощь в их устранении;

Основными формами текущего контроля знаний студентов являются:

– устный контрольный опрос;
– защита лабораторной работы.

Устный контрольный опрос студентов проводится на лекциях (и лабораторных занятиях). По его результатам преподаватель оценивает качество подготовки студента к занятию.

На лабораторных занятиях знания и практические навыки студентов оцениваются по 5-балльной системе.

9.2. Рекомендации по подготовке к экзамену

Подготовка к экзамену начинается с начала изучения дисциплины.

Необходимо посещать все лекции и лабораторные занятия.

Экзамен проводится в 1, 2 семестрах после защиты всех лабораторных работ в объеме учебной программы и выполненной самостоятельной работы.

10 Перечень рекомендуемой литературы

1.1. Основная литература

1. Диков А.В. Клиентские технологии веб-дизайна. HTML5 и CSS3: учебное пособие / А.В. Диков. – Санкт-Петербург: Лань, 2022. – 188 с. – Книга находится в ЭБС ЛАНЬ. – ISBN 978-5-8114-3822-8
2. Садыков А.М. Методы разработки веб-приложений: Учеб.-метод. Пособие /ФГБОУ «Ивановский государственный энергетический университет имени В.И. Ленина. – Иваново, 2019.- 72 с. – Книга находится в ЭБС ЛАНЬ

Дополнительная литература:

1. Заяц А.М. Основы WEB -технологий. Разработка WEB-приложений современными инструментальными средствами: учебно-методическое пособие для бакалавров по направлению подготовки 09.03.03 «Информационных систем и технологий»/ А.М. Заяц, Л.Г. Пушкарева.- Санкт-Петербург: СПбГЛТУб 2021. – 116 с. – Книга находится в ЭБС ЛАНЬ. – ISBN 978-5-9239-1269-2

2. Тузовский А.Ф. Проектирование и разработка web-приложений: учебное пособие / А.Ф. Тузовский: Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2014. – 219 с. – Книга находится в ЭБС ЛАНЬ.

10.3. Методическая литература

– УМК по дисциплине «Информационные технологии и программирование» (электронных вариант).

10.4. Интернет-ресурсы

1	https://metanit.com/python/django/
2	https://django.fun/ru/docs/

3	https://webformyself.com/minikurs/django/

11. Материально-техническое обеспечение дисциплины

– индивидуальное взаимодействие со студентами по электронной почте для предварительного ознакомления с их разработками при подготовке к аудиторным занятиям;

– использование на лекциях и лабораторных занятиях мультимедийного оборудования для демонстрации листингов программного кода, скриншотов выполненных запросов, работы программ и пр.;

– включение в лабораторные работы индивидуального поиска, систематизации и анализа информации через Интернет;

– авторские презентации к лекциям;

– лекционная аудитория должна быть оборудована мультимедиа проектором;

– Лабораторные работы рассчитаны на компьютеры, имеющие доступ к Internet.

12.1. Программное обеспечение:

1	Notepad++

12.2. Материально-техническое обеспечение дисциплины (модуля)

Мультимедиа аудитория: проектор, акустическая система, аудио-усилитель, ноутбук, Internet.

Компьютерный класс. Высокопроизводительная ЭВМ, локальная вычислительная сеть.