**Руководство программиста**

**1.Верстка макета**

Для реализации верстки макетов использовался модуль flexbox.

В папке “макет” располагается главная веб-страница проекта, которая похожа на другие разделы сайта.

В теге <body> весь код заключен в контейнер:

<div class="wrapper">

...

</div>

Этот контейнер имеет flex-свойства позволяющие позиционировать все блоки на сайте в колонку:

.wrapper{

    display: flex;

    flex-direction:column;

    min-height: 100%;

    overflow: hidden;

}

Далее идет навигация по сайту:

<header class="header" id="header">

...

</header>

Навигация по сайту имеет ограничивающий контейнер:

<div class="content">

...

</div>

Для этого контейнера задана определенная ширина:

.content{

    max-width: 1105px;

    margin: 0 auto;

}

Сам контент навигации по сайту заключен в блок:

<div class="header\_inner">

...

</div>

Где располагаются ссылки на другие веб-страницы:

<div class="logo">

<img class="img" src="../макет/img/icons/figma.png" alt="">

</div>

<nav class="nav" id="nav">

<a class="nav\_\_link" href="../макет/index.html" data-scroll="figmaguid">FigmaGuid</a>

<a class="nav\_\_link" href="../Quiz/index.html" data-scroll="viktorina">Викторина</a>

<a class="nav\_\_link" href="../Видеоуроки/index.html" data-scroll="videouroki">Видеоуроки</a>

<a class="nav\_\_link" href="" data-scroll="site">Создание сайта с нуля</a>

<a class="nav\_\_link" href="" data-scroll="design">UX/UI-дизайн</a>

<a class="nav\_\_link" href="" data-scroll="Plagini">Плагины для Figma</a>

</nav>

Сам блок навигации по сайту имеет flex-свойства, выравнивающие элементы по центру и по всей ширине:

.header\_inner{

    display: flex;

    align-items: center;

    justify-content: space-between;

    padding-top: 25px;

}

Ссылки имеют свойства:

.nav{

    font-size: 16px;

    font-family: "Jost", serif;

    display: flex;

}

.nav\_\_link {

    margin-left: 84px;

    color:  white;

}

.nav a:first-child{

    margin-left: 0px;

    font-weight: 700;

}

И имеетcя hover-эффект изменяющий цвет ссылок с заданным промежутком времени:

.nav\_\_link:hover{

    color: #291c31bb;

    transition: .2s ;

}

Информация находящаяся на главном экране заключена в блок:

<main class="main" id="main">

...

</main>

Данный блок имеет свойства flexbox благодаря которым нижний колонтитул (футер) прижимается к низу:

.main{

    flex: 1 1 auto;

    font-family: 'Comfortaa', cursive;

    color: white;

    height: 753px;

    background: url("../макет/img/blurry-gradient.jpg") center /cover;

    padding-top: 100px;

    margin-bottom: 87px;

}

Где также имеется такой же ограничивающий контейнер как и в блоке навигации по сайту:

.content{

    max-width: 1105px;

    margin: 0 auto;

}

Контент на главном экране заключена в блок:

<div class="main\_body">

...

</div>

Блок с содержанием имеет вид:

<div class="anchor\_links">

<p class="title\_anchor\_links">Содержание</p>

<ul class="list">

<li><a href="#alink\_one" >Что такое Figma</a></li>

<li><a href="#alink\_two">Возможности и особенности Figma</a></li>

<li><a href="#alink\_three">Начало работы в Фигма</a></li>

<li><a href="#alink\_four">Интерфейс Фигмы</a></li>

<li><a href="#alink\_five">Как устроен редактор файлов Фигмы</a></li>

<li><a href="#alink\_six">Инструменты Фигмы</a></li>

</ul>

</div>

Для текста в содержании имеется иконки маркированного списка:

.list li{

    margin-top: 20px;

    background: url('../макет/img/icons/Ellipse\ 12.png') left no-repeat;

    padding-left: 15px;

    margin-left: 15px;

}

А также подчеркивание:

.list li a::before{

    content: '';

    bottom: -4px;

    left: 0;

    position: absolute;

    width: 100%;

    height: 1.5px;

    background-color: #ADFF00;

}

Весь контент теоретической информации заключен в блок:

<div class="content\_section">

...

</div>

Для которой задано свойство стиля шрифта:

.content\_section{

    font-family: 'Nunito', sans-serif;

}

Информационный блок имеет ограничивающий контейнер:

<div class="text\_content">

...

</div>

Который имеет заданную ширину:

.text\_content{

    max-width: 920px;

    margin: 0 auto;

}

Сама текстовая информация заключена в блок

<div class="text\_body">

...

<div class="text\_body">

В текстовом блоке темы имею в основном данную структуру:

 <div class="paragraph">

                <div class="title\_text">

                    <p class="text\_section" id="alink\_one">...</p>

                </div>

                <div class="paragraph\_text">

                    <p class="paragraph\_content">

                        ...

                    </p>

                </div>

                <div class="paragraph\_img">

   <img  src="..." alt="">

                </div>

            </div>

Футер заключен в блок:

<footer class="footer">

...

</footer>

Который имеет ограничивающий контейнер:

<div class="footer\_content">

...

</div>

С заданной шириной:

.footer\_content{

    max-width: 920px;

    margin: 0 auto;

}

Вся контентная часть заключена в блок:

<div class="footer\_body">

...

</div>

Данный блок имеет flex-свойства, благодаря которым элементы равномерно распределяются по всей ширине и центрируются:

.footer\_body{

 display: flex;

 justify-content: space-between;

 align-items: center;

}

В папке “Видеоуроки” находится веб-страница содержащая видео.

Главная контентная часть находятся в блоке:

<main class="main">

...

</main>

Данный блок имеет flex-свойства благодаря которым нижний колонтитул (футер) прижимается к низу:

.main{

    flex: 1 1 auto;

    font-family: 'Comfortaa', cursive;

    color: white;

    height: 753px;

    background: url("../макет/img/blurry-gradient.jpg") center /cover;

    padding-top: 100px;

    margin-bottom: 87px;

}

Структура блока с видео выглядит так:

 <div class="video\_content">

                <div class="video">

                <iframe width="470" height="315" src="..." title="..." frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>

                <div class="title\_video">...</div>

            </div>

  </div>

Блок с видео контентом имеет flex-свойства, с центрированием и переносом видео:

.video\_content{

    padding: 50px 0px;

    display: flex;

    justify-content: center;

    flex-wrap: wrap;

}

**2. Front-end часть сайта**

В проекте использовались такие библиотеки как: jquery и ajax.

Имеется кнопка “бургер меню” которая появляется вместо обычной навигации по сайту, при уменьшении окна браузера:

<button class="burger" type="button" id="navToggle">

                        <span class="burger\_\_item">Menu</span>

                    </button>

Возможность отобразить ссылки на веб-страницы при клике на “бургер меню”, реализовано с помощью добавления нового класса “show”:

    .nav.show {

        display: flex;

    }

Который применяется к переменной “nav” при клике на “бургер меню”:

let nav = $("#nav");

let navToggle = $("#navToggle");

/\* Nav Toggle \*/

navToggle.on("click", function(event) {

    event.preventDefault();

    nav.toggleClass("show");

});

Для реализации фиксированной навигации по сайту использовался класс “fixed”:

.header.fixed {

    height: auto;

    position: fixed;

    background-color: #A100F4;

}

.header.fixed .header\_inner{

  margin-top: -15px;

    padding-bottom: 10px;

}

Который применяется если позиция скролла больше чем высота главного экрана:

let header = $("#header");

let main = $("#main");

let introH = main.innerHeight();

let scrollPos = $(window).scrollTop();

function checkScroll(scrollPos, introH) {

    if( scrollPos > introH ) {

        header.addClass("fixed");

    } else {

        header.removeClass("fixed");

    }

}

Для того чтобы фиксированная навигация по сайту не пропала при перезагрузке сайта, используется данное событие:

$(window).on("scroll resize", function() {

    introH = main.innerHeight();

    scrollPos = $(this).scrollTop();

    checkScroll(scrollPos, introH);

});

Кнопка “Наверх”

<button class="scroll-to-top"><i class="fa fa-arrow-up"></i></button>

В данном скрипте идет проверка пикселей, прокрученных от верха элемента.

При истинности условия кнопка удаляется, при ложном условии кнопка отображается:

const scrollBtn=document.querySelector(".scroll-to-top");

const refreshButtonVisibility=()=>{

    if (document.documentElement.scrollTop<=800){

        scrollBtn.style.display="none";

    } else {

        scrollBtn.style.display="block";

    }

};

Реализация теста выглядит следующим образом:

//выбираем все необходимые элементы

const start\_btn = document.querySelector(".start\_btn button");

const info\_box = document.querySelector(".info\_box");

const exit\_btn = info\_box.querySelector(".buttons .quit");

const continue\_btn = info\_box.querySelector(".buttons .restart");

const quiz\_box = document.querySelector(".quiz\_box");

const result\_box = document.querySelector(".result\_box");

const option\_list = document.querySelector(".option\_list");

const time\_line = document.querySelector(".top .time\_line");

const timeText = document.querySelector(".timer .time\_left\_txt");

const timeCount = document.querySelector(".timer .timer\_sec");

//если нажата кнопка startQuiz

start\_btn.onclick = ()=>{

    info\_box.classList.add("activeInfo"); //показать информационное окно

}

//если нажата кнопка exitQuiz

exit\_btn.onclick = ()=>{

    info\_box.classList.remove("activeInfo"); //скрыть информационное окно

}

// if continueQuiz button clicked

continue\_btn.onclick = ()=>{

    info\_box.classList.remove("activeInfo"); //скрыть информационное окно

    quiz\_box.classList.add("activeQuiz"); //показать окно викторины

    showQuetions(0); //вызов функции showQestions

    queCounter(1); //передача 1 параметра queCounter

    startTimer(15); //вызов функции startTimer

    startTimerLine(0); //вызов функции startTimerLine

}

let timeValue =  15;

let que\_count = 0;

let que\_numb = 1;

let userScore = 0;

let counter;

let counterLine;

let widthValue = 0;

const restart\_quiz = result\_box.querySelector(".buttons .restart");

const quit\_quiz = result\_box.querySelector(".buttons .quit");

// если нажата кнопка restartQuiz

restart\_quiz.onclick = ()=>{

    quiz\_box.classList.add("activeQuiz"); //показать окно викторины

    result\_box.classList.remove("activeResult"); //скрыть окно результата

    timeValue = 15;

    que\_count = 0;

    que\_numb = 1;

    userScore = 0;

    widthValue = 0;

    showQuetions(que\_count); //вызов функции showQestions

    queCounter(que\_numb); //передача значения que\_numb в queCounter

    clearInterval(counter); //очистить счетчик

    clearInterval(counterLine); //очищаем counterLine

    startTimer(timeValue); //вызов функции startTimer

    startTimerLine(widthValue); //вызов функции startTimerLine

    timeText.textContent = "Time Left"; //меняем текст timeText на Time Left

    next\_btn.classList.remove("show"); //скрываем следующую кнопку

}

// если нажата кнопка quitQuiz

quit\_quiz.onclick = ()=>{

    window.location.reload(); //перезагружаем текущее окно

}

const next\_btn = document.querySelector(".bottom .next\_btn");

const bottom\_ques\_counter = document.querySelector(".bottom .total\_que");

// если нажата кнопка Next Que

next\_btn.onclick = ()=>{

    if(que\_count < questions.length - 1){ //если количество вопросов меньше общей длины вопроса

        que\_count++; //увеличиваем значение que\_count

        que\_numb++; //увеличиваем значение que\_numb на единицу

        showQuetions(que\_count); //вызов функции showQestions

        queCounter(que\_numb); //передача значения que\_numb в queCounter

        clearInterval(counter); //очистить счетчик

        clearInterval(counterLine); //очищаем counterLine

        startTimer(timeValue); //вызов функции startTimer

        startTimerLine(widthValue); //вызов функции startTimerLine

        timeText.textContent = "Время"; //меняем timeText на Time Left

        next\_btn.classList.remove("show"); //скрываем следующую кнопку

    }else{

        clearInterval(counter); //очистить счетчик

        clearInterval(counterLine); //очищаем counterLine

        showResult(); //вызов функции showResult

    }

}

// получение вопросов и вариантов из массива

function showQuetions(index){

    const que\_text = document.querySelector(".que\_text");

    //создание нового тега span и div для вопроса и варианта и передача значения с использованием индекса массива

    let que\_tag = '<span>'+ questions[index].numb + ". " + questions[index].question +'</span>';

    let option\_tag = '<div class="option"><span>'+ questions[index].options[0] +'</span></div>'

    + '<div class="option"><span>'+ questions[index].options[1] +'</span></div>'

    + '<div class="option"><span>'+ questions[index].options[2] +'</span></div>'

    + '<div class="option"><span>'+ questions[index].options[3] +'</span></div>';

    que\_text.innerHTML = que\_tag; //добавляем новый тег span внутри que\_tag

    option\_list.innerHTML = option\_tag; //добавляем новый тег div внутри option\_tag

    const option = option\_list.querySelectorAll(".option");

    // устанавливаем атрибут onclick для всех доступных опций

    for(i=0; i < option.length; i++){

        option[i].setAttribute("onclick", "optionSelected(this)");

    }

}

// создание новых тегов div для значков

let tickIconTag = '<div class="icon tick"><i class="fas fa-check"></i></div>';

let crossIconTag = '<div class="icon cross"><i class="fas fa-times"></i></div>';

//если пользователь выбрал опцию

function optionSelected(answer){

    clearInterval(counter); //очистить счетчик

    clearInterval(counterLine); //очищаем counterLine

    let userAns = answer.textContent; //получение выбранной пользователем опции

    let correcAns = questions[que\_count].answer; //получение правильного ответа из массива

    const allOptions = option\_list.children.length; //получение всех пунктов опций

    if(userAns == correcAns){ //если выбранная пользователем опция равна правильному ответу массива

        userScore += 1; //увеличение значения оценки на 1

        answer.classList.add("correct"); //добавляем зеленый цвет, чтобы исправить выбранную опцию

        answer.insertAdjacentHTML("beforeend", tickIconTag); //добавляем значок галочки для исправления выбранной опции

        console.log("Correct Answer");

        console.log("Your correct answers = " + userScore);

    }else{

        answer.classList.add("incorrect"); //добавление красного цвета для исправления выбранной опции

        answer.insertAdjacentHTML("beforeend", crossIconTag); //добавляем крестик для исправления выбранной опции

        console.log("Wrong Answer");

        for(i=0; i < allOptions; i++){

            if(option\_list.children[i].textContent == correcAns){ //сли есть опция, соответствующая ответу массива

                option\_list.children[i].setAttribute("class", "option correct"); //добавляем зеленый цвет к согласованному варианту

                option\_list.children[i].insertAdjacentHTML("beforeend", tickIconTag); //добавляем галочку к выбранному варианту

                console.log("Auto selected correct answer.");

            }

        }

    }

    for(i=0; i < allOptions; i++){

        option\_list.children[i].classList.add("disabled"); //как только пользователь выберет опцию, отключит все опции

    }

    next\_btn.classList.add("show"); //показать следующую кнопку, если пользователь выбрал какую-либо опцию

}

function showResult(){

    info\_box.classList.remove("activeInfo"); //скрыть информационное окно

    quiz\_box.classList.remove("activeQuiz"); //скрыть окно викторины

    result\_box.classList.add("activeResult"); //показать окно результата

    const scoreText = result\_box.querySelector(".score\_text");

    if (userScore > 1){ // если пользователь набрал больше 3

        //создание нового тега span и передача номера оценки пользователя и общего количества вопросов

        let scoreTag = '<span> Вы получили <p>'+ userScore +'</p>баллов из <p>'+ questions.length +'</p></span>';

        scoreText.innerHTML = scoreTag;  //добавляем новый тег span внутри score\_Text

    }

    else{ // если пользователь набрал меньше 1

        let scoreTag = '<span>Вы получили <p>'+ userScore +'</p>баллов из <p>'+ questions.length +'</p></span>';

        scoreText.innerHTML = scoreTag;

    }

}

function startTimer(time){

    counter = setInterval(timer, 1000);

    function timer(){

        timeCount.textContent = time; //изменение значения timeCount на значение времени

        time--; //уменьшаем значение времени

        if(time < 9){ //если таймер меньше 9

            let addZero = timeCount.textContent;

            timeCount.textContent = "0" + addZero; //добавляем 0 перед значением времени

        }

        if(time < 0){ //если таймер меньше 0

            clearInterval(counter); //очистить счетчик

            timeText.textContent = "Время вышло"; //меняем текст времени на время выключено

            const allOptions = option\_list.children.length; //получение всех пунктов опций

            let correcAns = questions[que\_count].answer; //получение правильного ответа из массива

            for(i=0; i < allOptions; i++){

                if(option\_list.children[i].textContent == correcAns){ //если есть вариант, соответствующий массиву answer

                    option\_list.children[i].setAttribute("class", "option correct"); //добавляем зеленый цвет к согласованному варианту

                    option\_list.children[i].insertAdjacentHTML("beforeend", tickIconTag); //добавляем галочку к выбранному варианту

                    console.log("Время вышло: Auto selected correct answer.");

                }

            }

            for(i=0; i < allOptions; i++){

                option\_list.children[i].classList.add("disabled");//как только пользователь выберет опцию, отключит все опции

             }

            next\_btn.classList.add("show"); //показать следующую кнопку, если пользователь выбрал какую-либо опцию

        }

    }

}

function startTimerLine(time){

    counterLine = setInterval(timer, 29);

    function timer(){

        time += 1; //обновление значения времени на 1

        time\_line.style.width = time + "px"; //увеличиваем ширину time\_line на px по значению времени

        if(time > 549){ //если значение времени больше 549

            clearInterval(counterLine); //очищаем counterLine

        }

    }

}

function queCounter(index){

    //создание нового тега span и передача номера вопроса и общего вопроса

    let totalQueCounTag = '<span><p>'+ index +'</p> из <p>'+ questions.length +'</p>Вопросов</span>';

    bottom\_ques\_counter.innerHTML = totalQueCounTag;  //добавляем новый тег span внутри bottom\_ques\_counter

}

Реализация функционала изменения и создания новых вопросов находится в questions.js.

// создание массива и передача числа, вопросов, вариантов и ответов

let questions = [

    {

    numb: 1,

    question: "Что такое Figma?", //вопрос

    answer: "Графический онлайн-редактор", //ответ

    options: [  //варианты ответов

      "Графический онлайн-редактор",

      "Текстовый редактор",

      "Видеохостинг",

      "Видеоредактор"

    ]

  },

];

**3. Спецификация проекта**

|  |  |  |
| --- | --- | --- |
| Имя папки | Имя файла | Назначение |
| Site | «макет» | Файл содержащий страницу сайта «Гид по Фигме» |
| «Дизайн» | Файл содержащий страницу сайта «UX/UI-дизайн» |
| «Создание\_сайта\_с\_нуля» | Файл содержащий страницу сайта «Создание сайта с нуля» |
| «Плагины» | Файл содержащий страницу сайта «Плагины для Figma» |
| «Видеоуроки» | Файл содержащий страницу сайта «Видеоуроки» |
| «Quiz» | Файл содержащий страницу сайта «Викторина» |
| макет | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «script.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| «scroll-to-top.js» | js-скрипт, добавляющий кнопку «Наверх» на сайте |
| Создание\_сайта\_с\_нуля | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «script.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| «scroll-to-top.js» | js-скрипт, добавляющий кнопку «Наверх» на сайте |
| Плагины | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «script.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| «scroll-to-top.js» | js-скрипт, добавляющий кнопку «Наверх» на сайте |
| Дизайн | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «script.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| «scroll-to-top.js» | js-скрипт, добавляющий кнопку «Наверх» на сайте |
| Видеоуроки | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «script.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| Quiz | «index.html» | веб-страница «Создание сайта с нуля» |
| «style.css» | стили для файла «index.html» |
| «img» | папка, содержащая изображения для сайта |
| «icon» | папка, содержащая иконки для сайта |
| «header.js» | js-скрипт, который делают навигационное меню сайта фиксированным, добавляет бургер меню при изменении окна браузера |
| «questions.js» | js-скрипт содержащий информацию по вопросам и ответам на эти вопросам |
| «script.js» | js-скрипт выполняющий функционал викторины |