Стоимость разработки сайта

Начальный код html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<link rel="apple-touch-icon" sizes="57x57" href="favicon/apple-icon-57x57.png">

<link rel="apple-touch-icon" sizes="60x60" href="favicon/apple-icon-60x60.png">

<link rel="apple-touch-icon" sizes="72x72" href="favicon/apple-icon-72x72.png">

<link rel="apple-touch-icon" sizes="76x76" href="favicon/apple-icon-76x76.png">

<link rel="apple-touch-icon" sizes="114x114" href="favicon/apple-icon-114x114.png">

<link rel="apple-touch-icon" sizes="120x120" href="favicon/apple-icon-120x120.png">

<link rel="apple-touch-icon" sizes="144x144" href="favicon/apple-icon-144x144.png">

<link rel="apple-touch-icon" sizes="152x152" href="favicon/apple-icon-152x152.png">

<link rel="apple-touch-icon" sizes="180x180" href="favicon/apple-icon-180x180.png">

<link rel="icon" type="image/png" sizes="192x192" href="favicon/android-icon-192x192.png">

<link rel="icon" type="image/png" sizes="32x32" href="favicon/favicon-32x32.png">

<link rel="icon" type="image/png" sizes="96x96" href="favicon/favicon-96x96.png">

<link rel="icon" type="image/png" sizes="16x16" href="favicon/favicon-16x16.png">

<link rel="manifest" href="favicon/manifest.json">

<meta name="msapplication-TileColor" content="#ffffff">

<meta name="msapplication-TileImage" content="favicon/ms-icon-144x144.png">

<meta name="theme-color" content="#ffffff">

<link rel="stylesheet" href="css/style.min.css">

<script src="./script/script.js"></script>

<title>Калькулятор стоимости верстки</title>

</head>

<body>

<main class="card">

<h1 class="card-head">Стоимость разработки сайта</h1>

<div class="first-screen">

<p class="first-screen\_description">

Расскажите больше о сайте, который вам нужен, чтобы рассчитать примерную стоимость разработки. Если нужна помощь — напишите мне в телеграм или позвоните по телефону.

</p>

<button class="button start-button">Рассчитать...</button>

</div>

<section class="main-form" style="display: none;">

<form class="form-calculate" method="POST" action="server.php" enctype="multipart/form-data">

<fieldset class="first-fieldset">

<legend>Какой сайт вам нужен?</legend>

<p>

<input class="css-radio calc-handler" id="landing" type="radio" name="whichSite" value="landing" checked data-site="одностраничный сайт">

<label class="label-radio" for="landing">Одностраничный (лендинг)</label>

</p>

<p>

<input class="css-radio calc-handler" id="multiPage" type="radio" name="whichSite" value="multiPage" data-site="многостраничный сайт">

<label class="label-radio" for="multiPage">Многостраничный (корпоративный)</label>

</p>

<p>

<input class="css-radio calc-handler" id="onlineStore" type="radio" name="whichSite" value="onlineStore" data-site="интернет магазин">

<label class="label-radio" for="onlineStore">Интернет-магазин</label>

</p>

</fieldset>

<div class="total\_price">

<span class="total\_price\_\_wrap">

<span class="total\_price\_\_sum">4000</span>

<span class="total\_price\_\_val">₽</span>

</span>

</div>

<fieldset>

<legend>Нужно ли разрабатывать дизайн сайта?</legend>

<label class="switcher">

<input type="checkbox" class="calc-handler" name="option" id="desktopTemplates" value="desktopTemplates">

<span class="switcher\_\_indicator"></span>

<span class="checkbox-label desktopTemplates\_value">Нет</span>

</label>

</fieldset>

<fieldset>

<legend>Адаптировать сайт под смартфоны и&nbsp;планшеты?</legend>

<label class="switcher">

<input type="checkbox" class="calc-handler" name="option" id="adapt" value="adapt">

<span class="switcher\_\_indicator"></span>

<span class="checkbox-label adapt\_value">Нет</span>

</label>

</fieldset>

<fieldset>

<legend>Нужно ли разрабатывать дизайн сайта под мобильные устройства?</legend>

<label class="switcher">

<input type="checkbox" class="calc-handler" name="option" id="mobileTemplates" value="mobileTemplates" disabled>

<span class="switcher\_\_indicator"></span>

<span class="checkbox-label mobileTemplates\_value">Нет</span>

</label>

</fieldset>

<fieldset>

<legend>Вы хотите самостоятельно менять и добавлять текст на сайте без помощи разработчика?</legend>

<label class="switcher">

<input type="checkbox" class="calc-handler" name="option" id="editable" value="editable">

<span class="switcher\_\_indicator"></span>

<span class="checkbox-label editable\_value">Нет</span>

</label>

</fieldset>

<fieldset>

<legend>Дополнительный функционал для сайта</legend>

<p>

<input class="css-check calc-handler" id="metrikaYandex" type="checkbox" name="option" value="metrikaYandex">

<label class="label-check" for="metrikaYandex">Яндекс Метрика</label>

</p>

<p>

<input class="css-check calc-handler" id="analyticsGoogle" type="checkbox" name="option" value="analyticsGoogle">

<label class="label-check" for="analyticsGoogle">Гугл Аналитика</label>

</p>

<p>

<input class="css-check calc-handler" id="sendOrder" type="checkbox" name="option" value="sendOrder">

<label class="label-check" for="sendOrder">Отправка заявок на почту или в телеграм</label>

</p>

</fieldset>

<fieldset>

<legend>Время на разработку сайту</legend>

<p class="fast-description">Разрабатываю <span class="type-site">одностраничный сайт</span> от

<span class="max-deadline">7 дней</span>. Если вам нужно сделать сайт быстрее, поставьте галочку ниже и выберите подходящие сроки.

</p>

<label class="switcher">

<input type="checkbox" class="want-faster">

<span class="switcher\_\_indicator"></span><span class="checkbox-label">Хочу быстрее</span>

</label>

<p class="fast-range" style="display: none;">

<label><input type="range" class="range-deadline calc-handler" min="2" max="7" step="1" value="7" name="deadline"></label>

<span class="deadline"><span class="deadline-value">7 дней</span>\*</span>

<small class="deadline-small">\*точные сроки определяются при обсужении технического задания</small>

</p>

</fieldset>

<fieldset class="subtotal">

<h2>Примерная стоимость разработки сайта</h2>

<p class="calc-description">

Сделаем одностраничный сайт, адаптированный под мобильные

устройства и планшеты. Установим панель админстратора, чтобы вы

могли самостоятельно менять содержание на сайте без разработчика.

Подключим Яндекс Метрику, Гугл Аналитику и отправку заявок на почту.

</p>

<p class="calc-description">

Оставьте заявку, если хотите, чтобы я разработал сайт. Свяжусь с вами

в течение 2 часов, чтобы обсудить задачу.

</p>

<button class="button end-button" type="button">Отправить заявку</button>

</fieldset>

<fieldset class="total" style="display: none;">

<div class="total\_area">

<label for="task" class="total\_label">Расскажите о задаче</label>

<div class="total\_block">

<textarea class="total\_field" id="task" type="text" name="task" rows="7" maxlength="1000"></textarea>

<p><small class="total\_small\_\_gray">Что нужно сделать, что вы хотите получить</small></p>

</div>

</div>

<div class="total\_area">

<label for="communication" class="total\_label">Как с вами связаться</label>

<div class="total\_block">

<input id="communication" type="text" class="total\_field" name="communication">

<p><small class="total\_small\_\_gray">Телефон, телеграм, ватсапп, почта</small></p>

</div>

</div>

<div class="total\_area">

<label for="total\_field" class="total\_label">Имя</label>

<div class="total\_block">

<input id="total\_field" type="text" class="total\_field" name="nameCustomer">

</div>

</div>

<div class="total\_submit">

<button type="submit" class="button">Отправить</button>

<p class="total\_small">Свяжусь с вами в течение двух часов</p>

</div>

</fieldset>

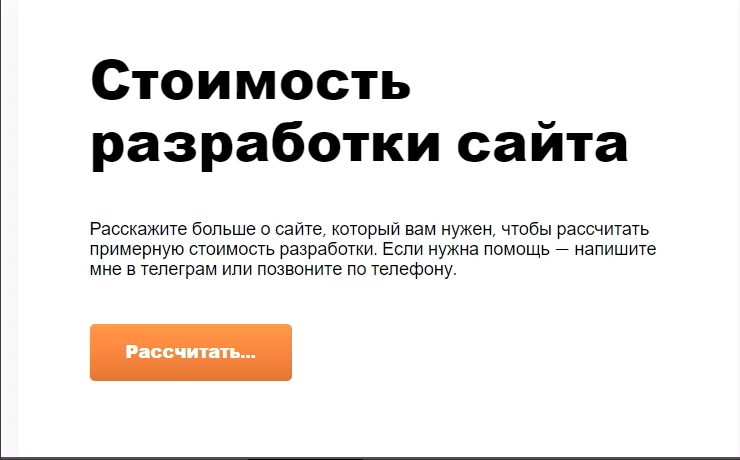
</form>

</section>

</main>

</body>

</html>



<script src="./script/script.js" defer></script>

Нам необходимо в нашей программе получить кнопку "рассчитать"

Для этого в файле js

'use strict';

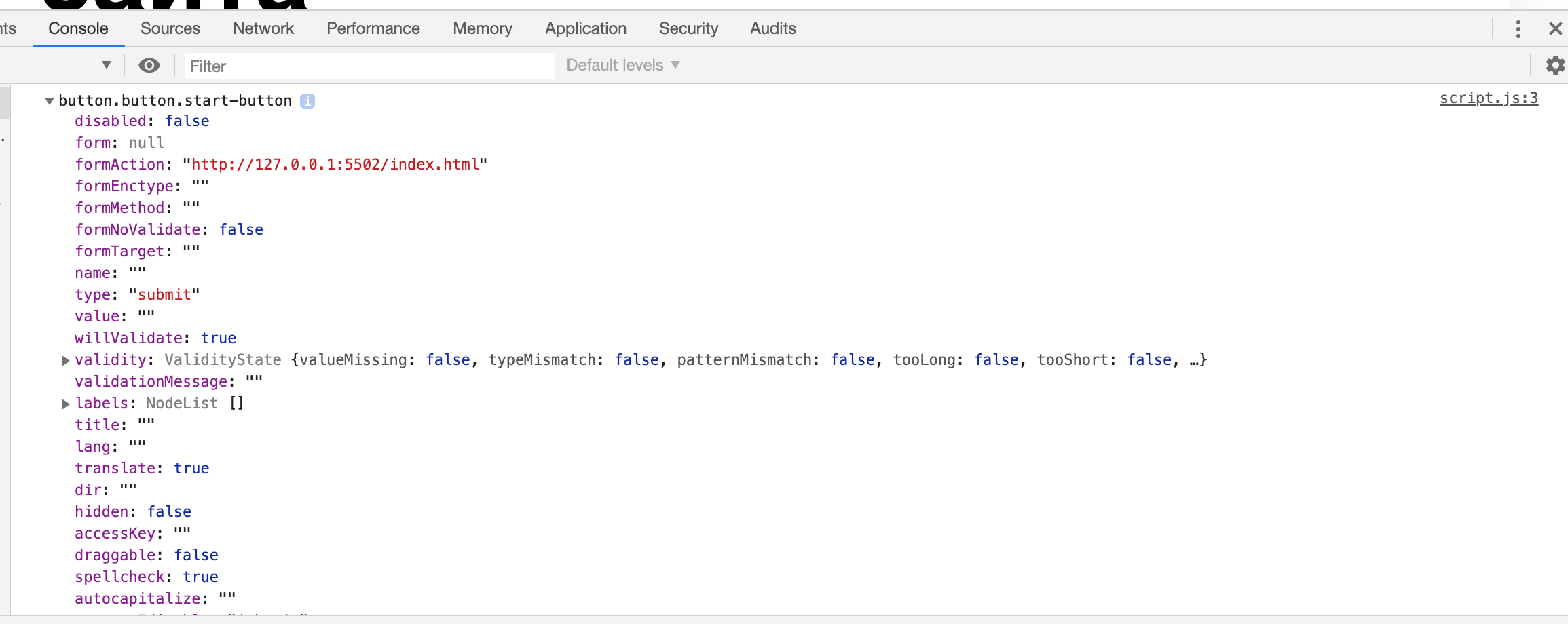
const startButton = document.querySelector('.start-button');

Навесим на нашу кнопку слушателя события. Чтобы узнать все свойства нашей кнопки, нужно вывести ее через console.dir

'use strict';

const startButton = document.querySelector('.start-button');

console.dir(startButton);



Через это свойство можно навешать на кнопку обработчик события

'use strict';

const startButton = document.querySelector('.start-button');

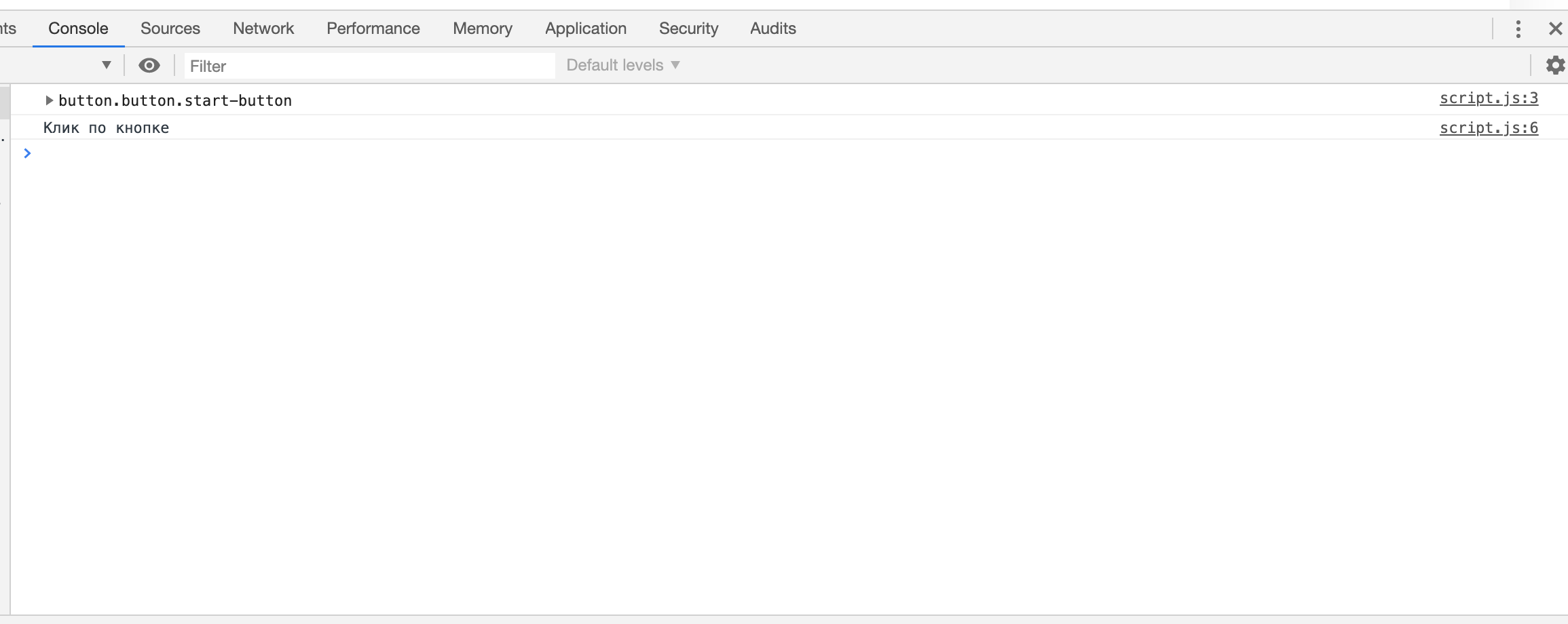
console.dir(startButton);

startButton.onclick = function () {

console.log('Клик по кнопке');

};

И если мы кликнем по кнопке, то получим

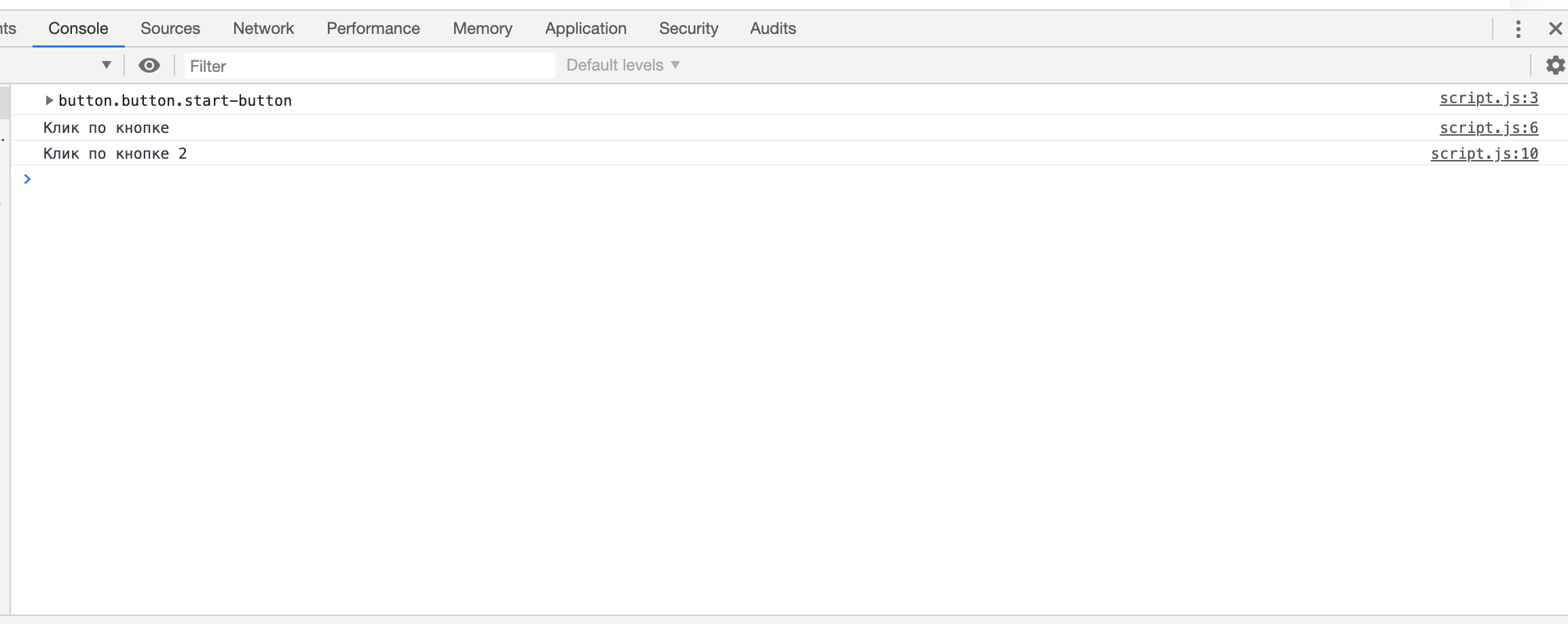


Но лучше этот метод не использовать. Современный метод:

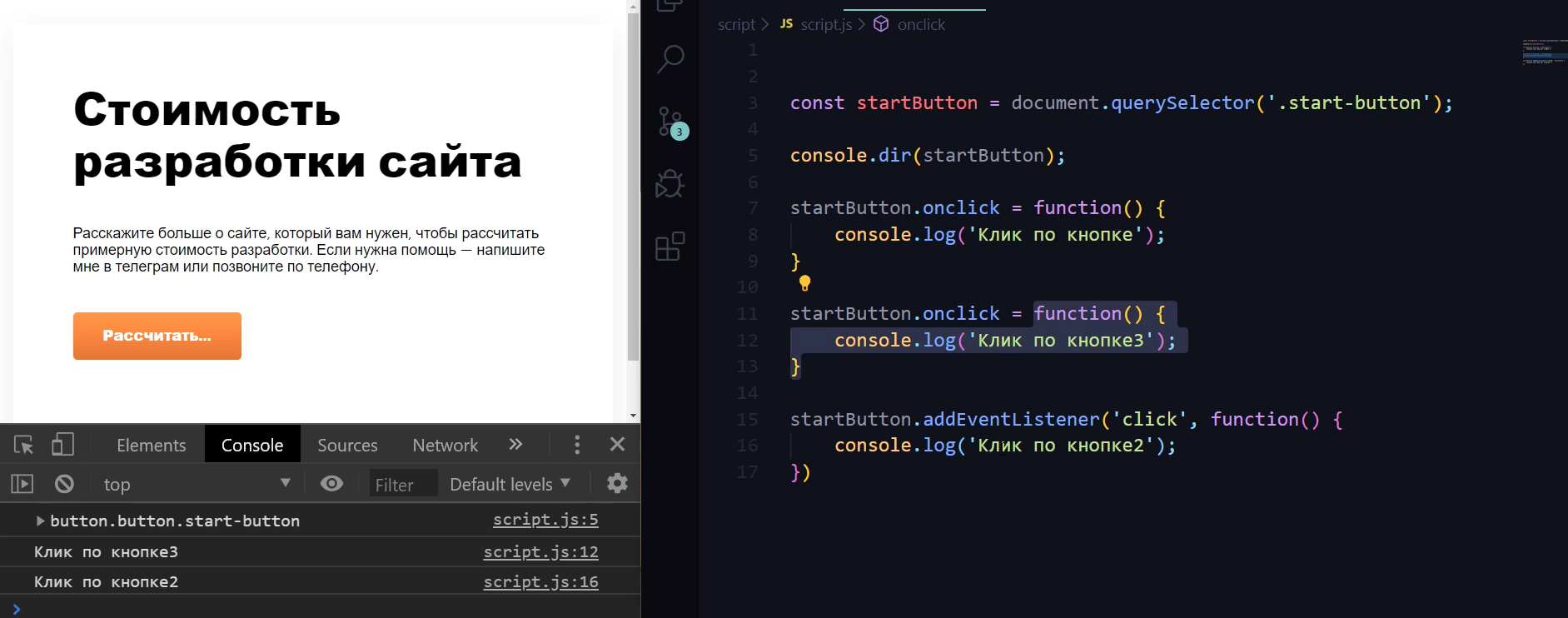
startButton.addEventListener('click', function(){

console.log('Клик по кнопке 2');

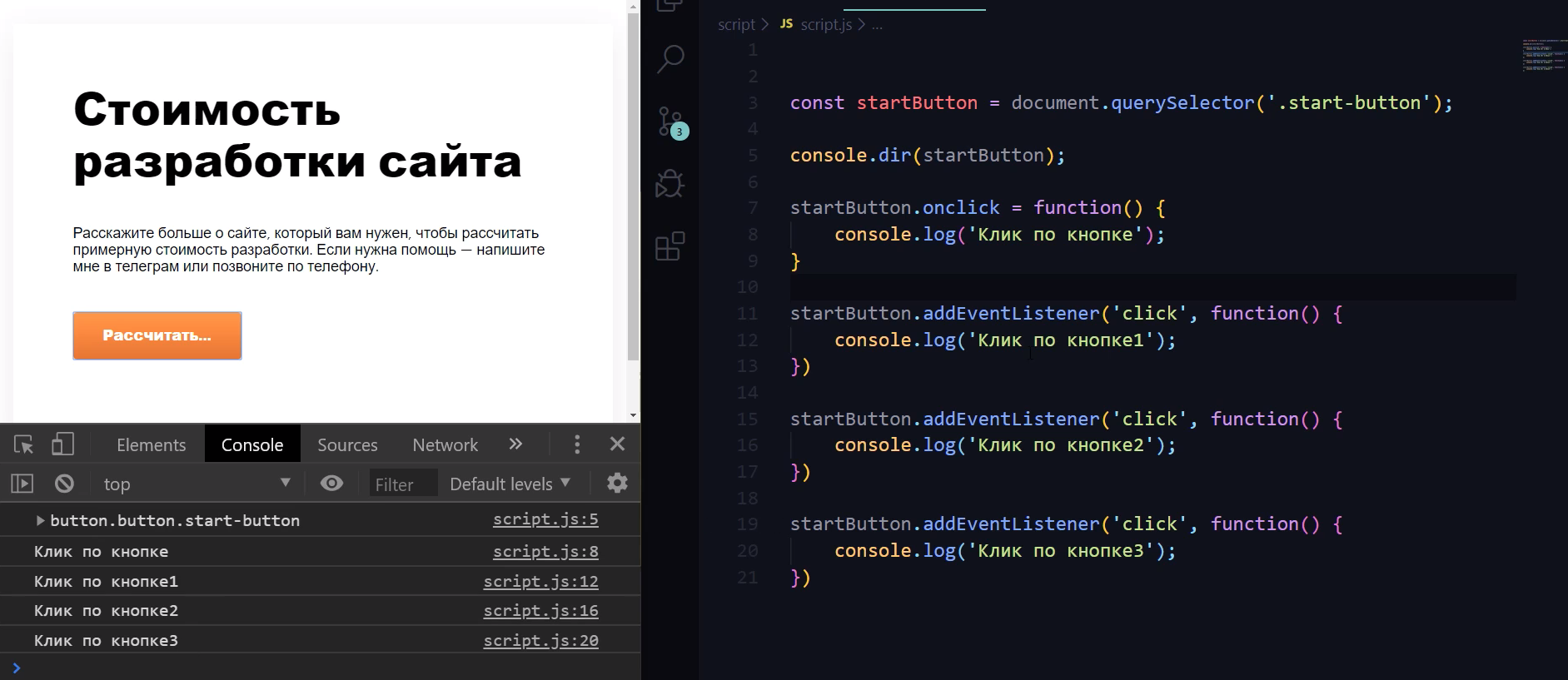
});



Второй метод лучше, так как первый перезаписывается, если нам на клик нужно повесить две функции:



Тогда как с eventListener такого не происходит:



При нажатии на кнопку у нас должно убираться с экрана все, что там есть, и показывать форму с классом "form-calculate" из верстки , у которой сейчас style="display: none"

Напишем ф-цию, которая принимает элемент, меняет его style и показывает

function showElem(elem) {

elem.style.display = 'block';

}

Вторая ф-ция должна скрывать элемент

function hideElem(elem) {

elem.style.display = 'none';

}

Сейчас весь код js выглядит так:

'use strict';

const startButton = document.querySelector('.start-button');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

startButton.addEventListener('click', function(){

console.log('Клик по кнопке 2');

});

=============================================

Наша кнопка "рассчитать" должна во-первых прятать то, что есть на экране. У него есть класс "first-screen"

Создадим константу, которая получает этот блок

const firstScreen = document.querySelector('.first-screen');

У секции, которая должна появиться на экране взамен есть класс "main-form"

Создадим константу, которая

const mainForm = document.querySelector('.main-form');

Тепеть, через слушателя по клику выполним ф-ции showElem и hideElem

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

});

Весь код выглядить так:

'use strict';

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

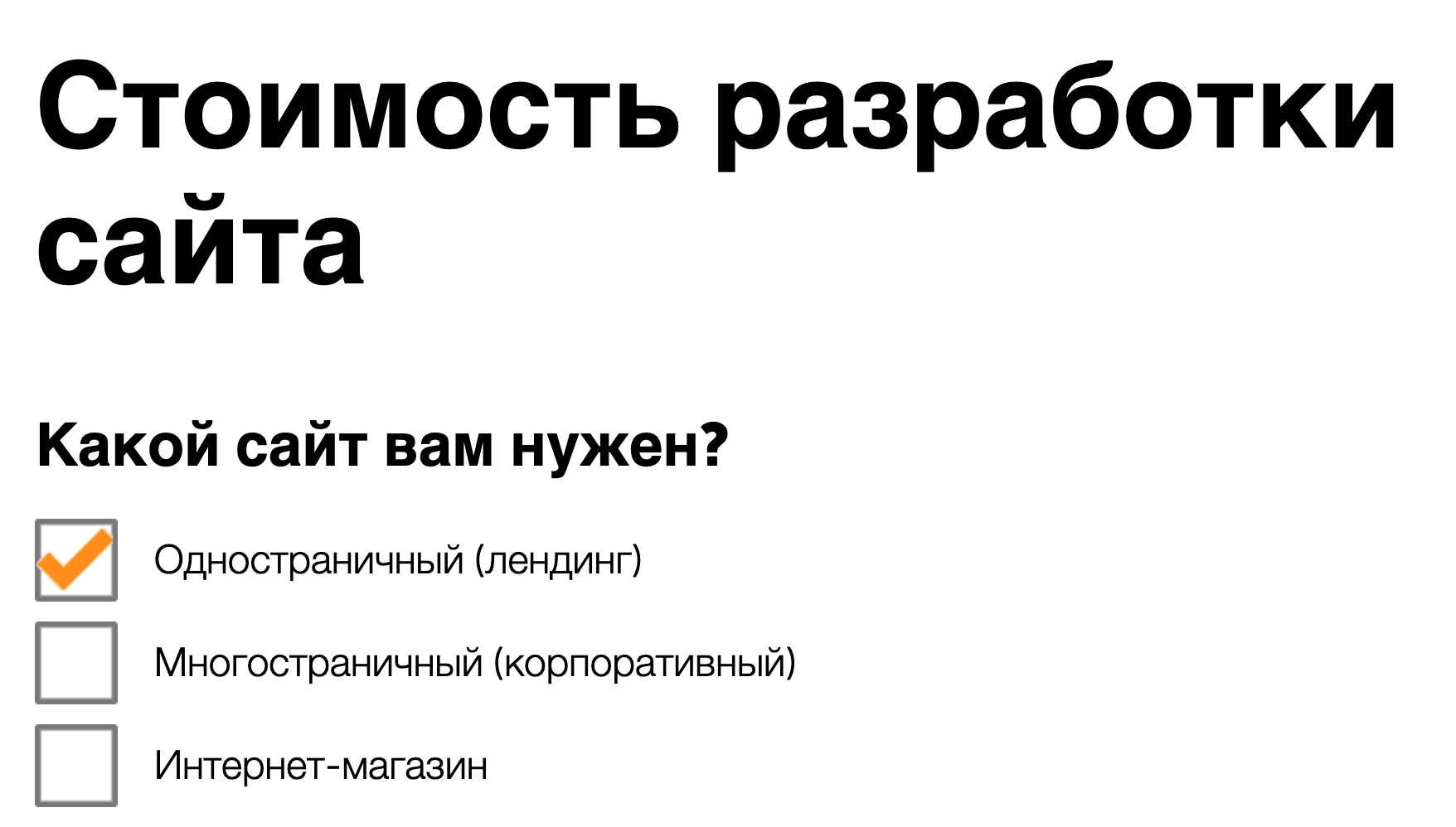
});

=================================

До нажатия на кнопку:



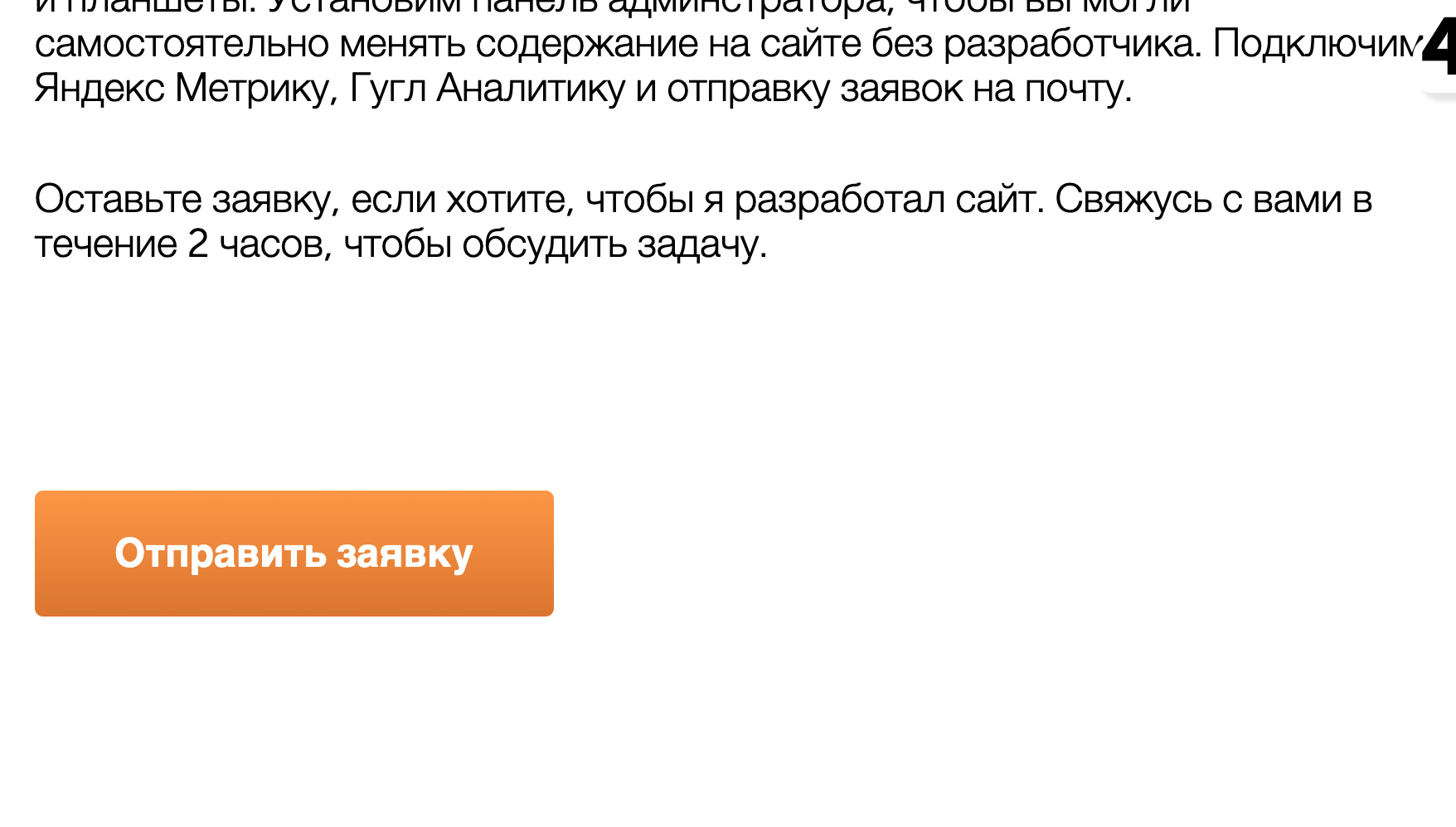
После нажатия на кнопку



Получилось!!!

==============================================

В конце нашей новой верстки есть кнопка:



После ее нажатия, все, что выше, должно быть скрыто и появится новая форма для отправления заявки. У нее есть класс "total"

Но в верстке эта форма прописана внутри контейнера, так что мы е можем просто крыть все. Нам нужно изменить для всех других fildset значение display: none

А для формы сделать display: block

=======================================

получим форму с классом form-calculate

const formCalculate = document.querySelector('.form-calculate');

Получим кнопку "отправить заявку"

const endButton = document.querySelector('.end-button');

Получим блок со всеми fieldset с классом total

const total = document.querySelector('.total');

===========================================

Весь код сейчас выглядит так:

'use strict';

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

const formCalculate = document.querySelector('.form-calculate');

const endButton = document.querySelector('.end-button');

const total = document.querySelector('.total');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

})

===================================

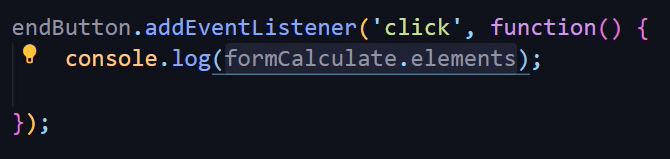
Теперь при клике на endButton снова нужно одно спрятать, а второе показать

Пишем слушателя на нажатие на кнопку

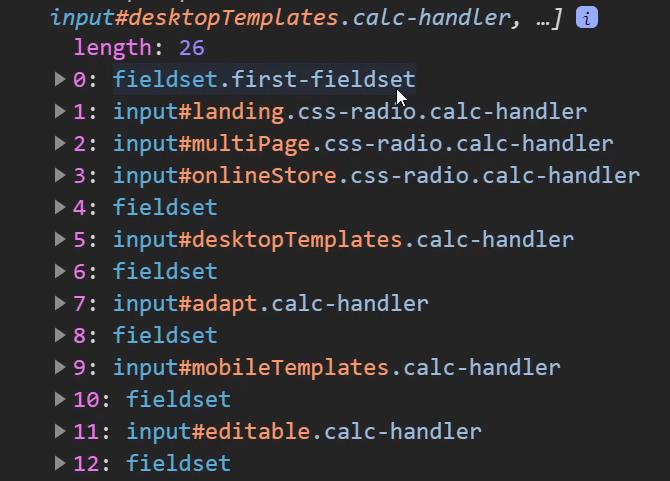
endButton.addEventListener('click', function(){

});

Если мы выведем нашу formCalculate в консоль



То увидим, что в ней 26 элементов!



Это инпуты, кнопки, филдсеты... Это все элементы, которые относятся к форме

Мы можем выбрать из них только fieldset, чтобы потом скрыть

Для этого воспользуемся циклом for , перебирая все элементы, которые есть в псевдомассиве

*for* (const elem of formCalculate.elements) {

}

То есть цикл будет крутиться ктолько раз, сколько у нас есть элементов. Мы с помощью условия найдем среди них fieldset

*if* (elem.tagName === 'FIELDSET')

И мы будем скрывать каждый такой элемент

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

То есть, после нажатия на кнопку, все элементы fitldset скроются:

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

}

});

Но еще нам нужно показать последнюю форму total :

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

}

showElem(total);

});

============================================

Весь код сейчас выглядит так:

'use strict';

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

const formCalculate = document.querySelector('.form-calculate');

const endButton = document.querySelector('.end-button');

const total = document.querySelector('.total');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

});

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

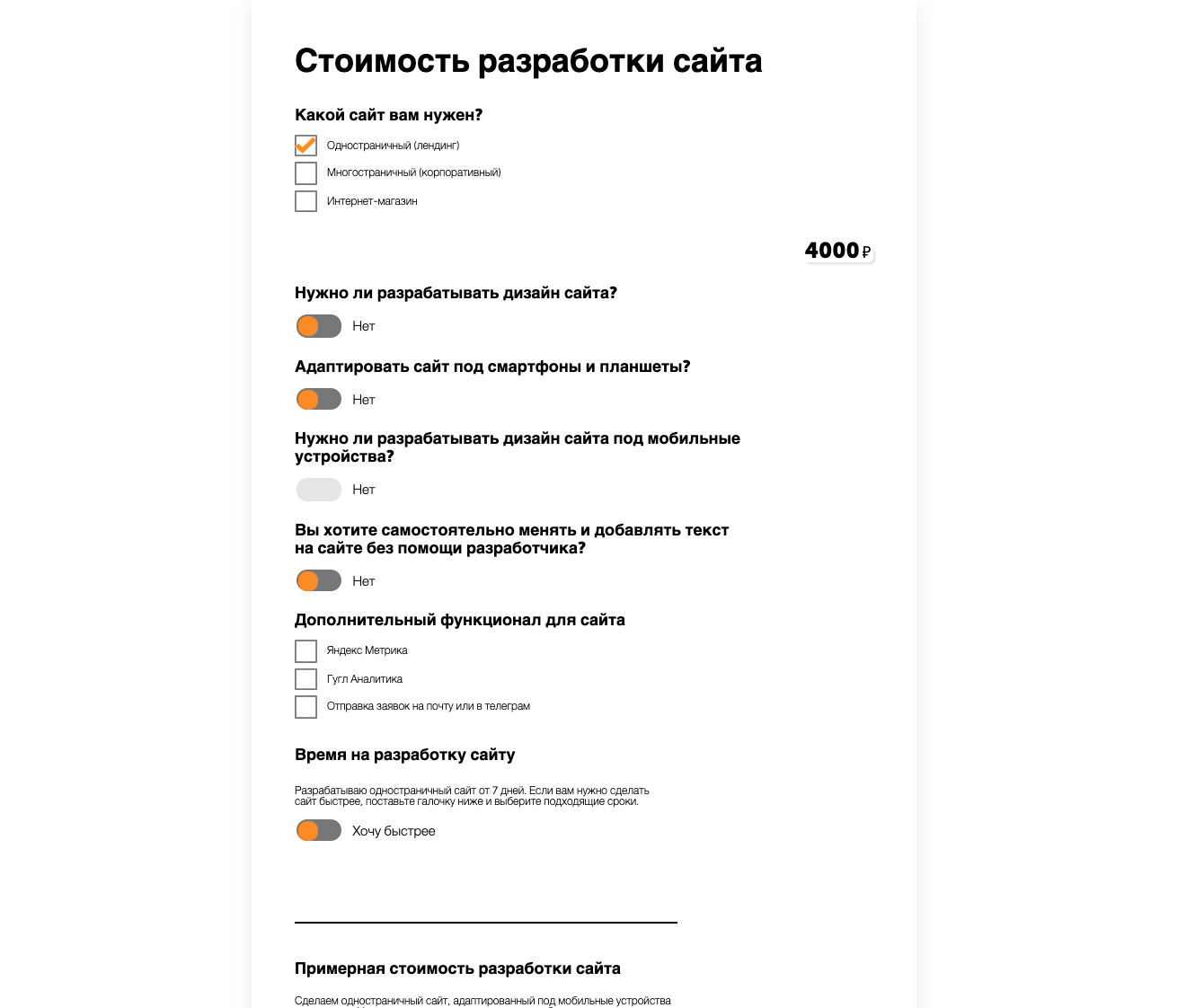
}

showElem(total);

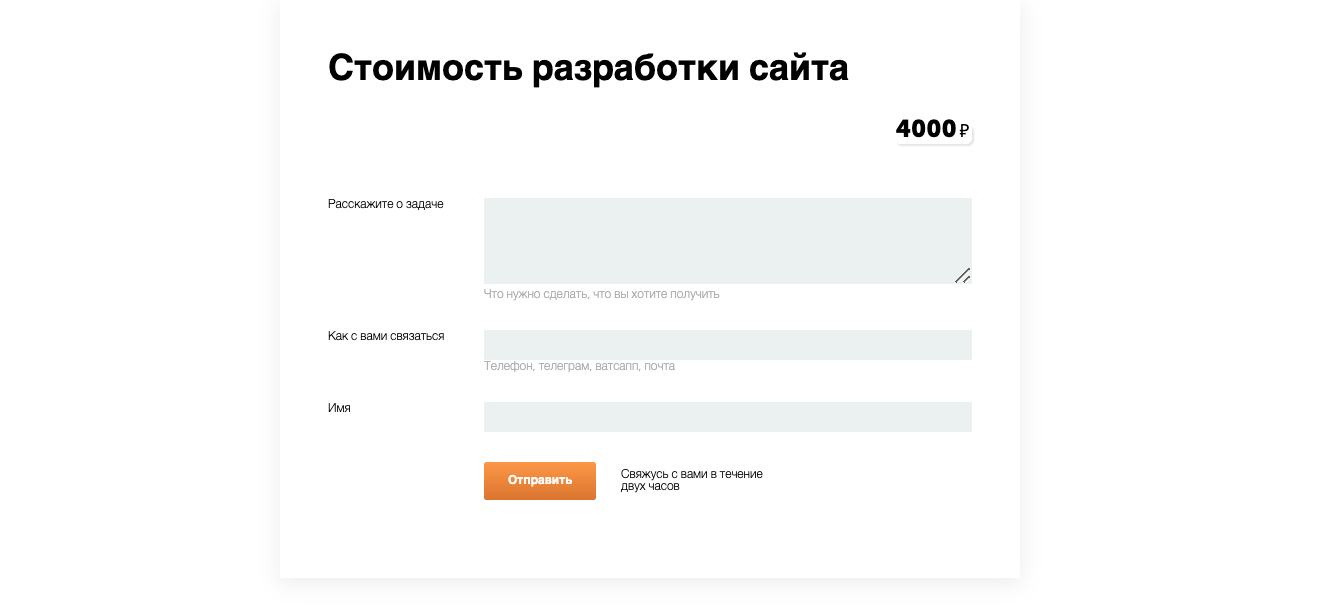
});

===================================

До нажатия на кнопку:



После нажатия:



Ура! Работает!!!

=========================================

Теперь на нашу вторую форму "formCalculate" навешаем событие "change" ==> если в форме изменится состояние любого импута, чекбокса, радиокнопки... то сработает это событие

formCalculate.addEventListener('change', handlerCallBackForm);

Где, handlerCallBackForm - это имя функции. Она будет запускать калькуляцию.

Пропишем ее в начале кода (чтобы можно было запускать везде)

function handlerCallBackForm() {

}

======================================

Весь код сейчас выглядит так:

'use strict';

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

const formCalculate = document.querySelector('.form-calculate');

const endButton = document.querySelector('.end-button');

const total = document.querySelector('.total');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

function handlerCallBackForm() {

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

});

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

}

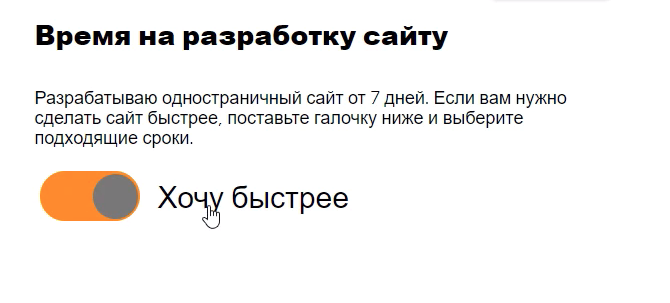
showElem(total);

});

formCalculate.addEventListener('change', handlerCallBackForm);

==================================

У нас в верстке есть кнопка "Хочу быстрее"



При переключении, внизу должен показываться скрытый элемент.

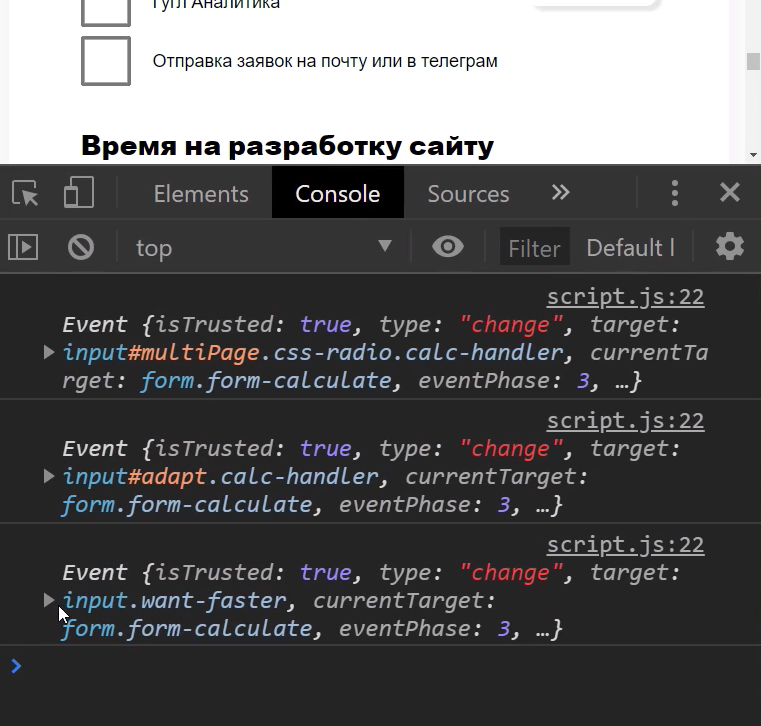
Для этого в ф-ции handlerCallBackForm нужно определять где именно произошло событие. Для этого существует объект event

function handlerCallBackForm(event) {

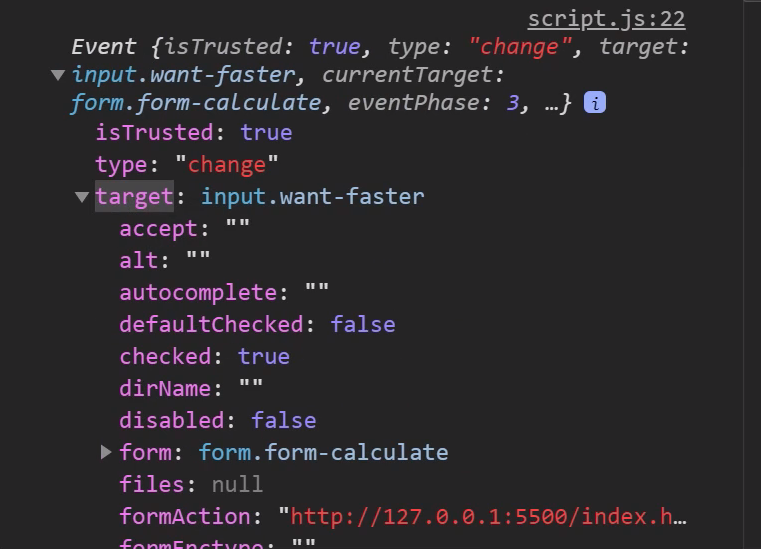
console.log('event: ', event);

}

Если вывести ее в консоль и создать какие-то изменения в верстке второго экрана, то получим свой event на каждое изменение



Нам нужно открыть событие изменения кнопки "хочу быстрее" и узнать значение target



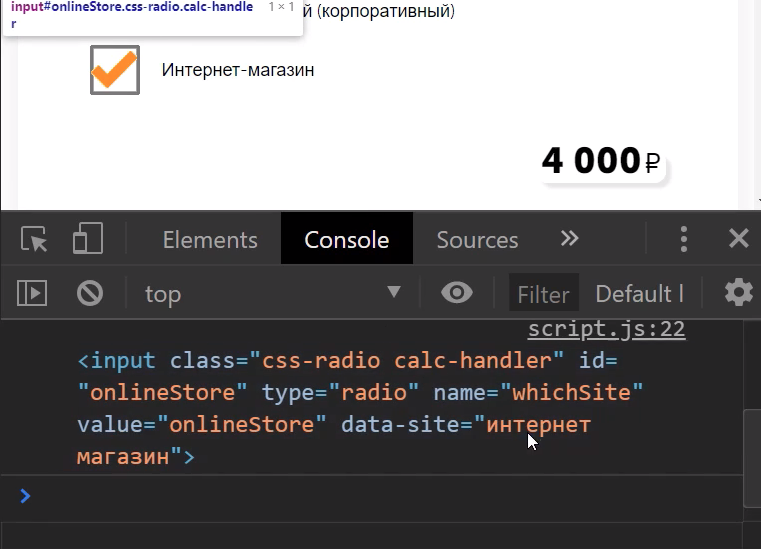
Выведем его:

function handlerCallBackForm(event) {

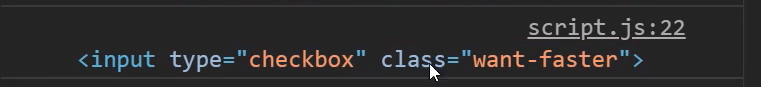
console.log(event.target);

}

Тогда при любом изменение на втором экране, мы будем получать значение target



Теперь, если мы нажмем "хочу быстрее", то получим:



В нашей ф-ции handlerCallBackForm получим константу, которая будет означать event.target - просто чтобы не писать это каждый раз

function handlerCallBackForm(event) {

const target = event.target;

}

Проверяем, произошло ли событие в нашей кнопке "хочу быстре" или где-то еще

*if* (target.classList.contains('want-faster')){

}

Проверим, стоит ли кнопка в позиции cheked (да, хочу быстрее)

*if* (target.checked){

}

Тогда напишем получение скрытого блока через showElem()

Класс блока, который нам нужно показать = fast-range . Чтобы использовать, получим его:

const fastRange = document.querySelector('.fast-range');

Тогда функция будет выглядеть так:

function handlerCallBackForm(event) {

const target = event.target;

*if* (target.classList.contains('want-faster')){

*if* (target.checked){

showElem(fastRange)

}

}

}

Но, если кнопка в позиции unchecked (нет, не хочу быстрее), то пишем else и скрываем этот элемент

*else* {

hideElem(fastRange)

}

=======================

вся функция будет иметь вид

function handlerCallBackForm(event) {

const target = event.target;

*if* (target.classList.contains('want-faster')){

*if* (target.checked){

showElem(fastRange)

} *else* {

hideElem(fastRange)

}

}

}

=======================

Весь код выглядит:

'use strict';

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

const formCalculate = document.querySelector('.form-calculate');

const endButton = document.querySelector('.end-button');

const total = document.querySelector('.total');

const fastRange = document.querySelector('.fast-range');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

function handlerCallBackForm(event) {

const target = event.target;

*if* (target.classList.contains('want-faster')){

*if* (target.checked){

showElem(fastRange)

} *else* {

hideElem(fastRange)

}

}

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

});

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

}

showElem(total);

});

formCalculate.addEventListener('change', handlerCallBackForm);

=================================

Итак вся функция имеет вид

function handlerCallBackForm(event) {

const target = event.target;

*if* (target.classList.contains('want-faster')){

*if* (target.checked){

showElem(fastRange)

} *else* {

hideElem(fastRange)

}

}

}

Такая запись понятна, но очень длинная. Для сокращения можно использовать тернарный оператор

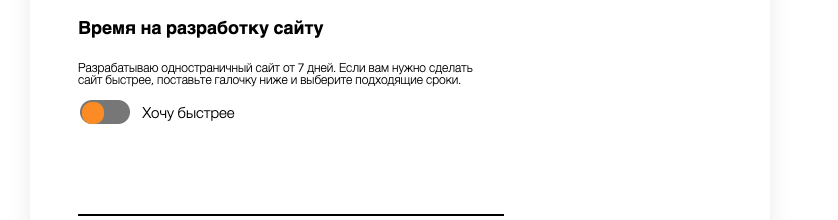
*if* (target.classList.contains('want-faster')){

target.checked ? showElem(fastRange) : hideElem(fastRange)

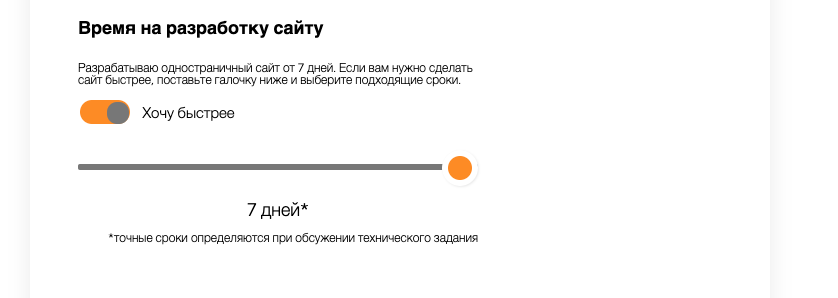
}

==================

До нажатия кнопки "хочу быстрее"



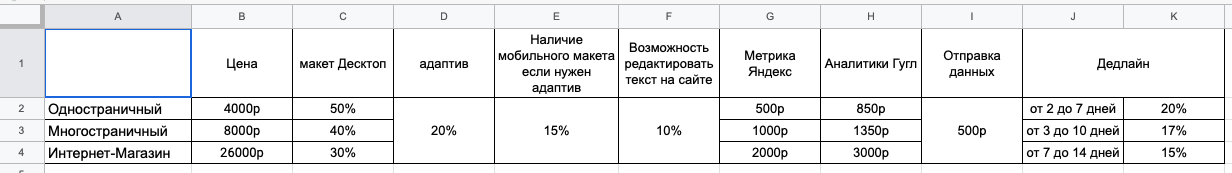
После нажатия



Ура! Работает :)

============================================

У нас есть таблица:



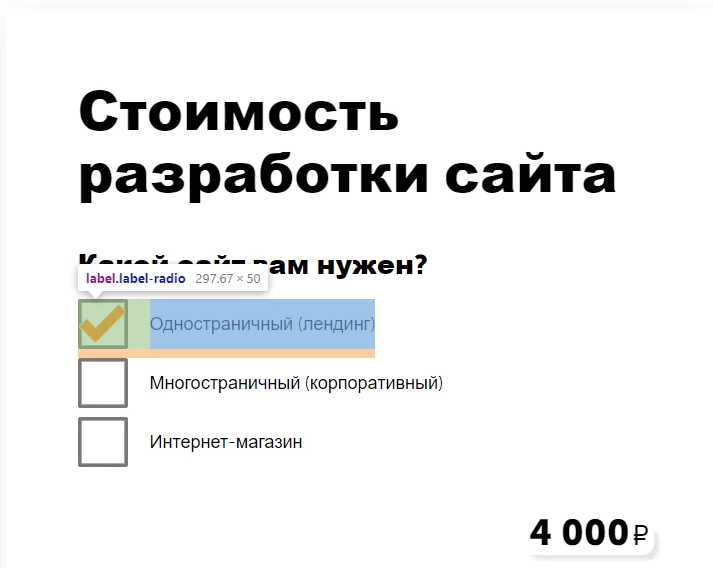
Создадим объект DATA, где она будет храниться.

const DATA = {

};

Первое, что мы в этом объекте сохраним, это тип сайта из колонки А

Эти элементы уже есть в верстке не втором экране



Находим нужный input

У него есть name = "whichSite"

Это имя общее для всех трех input

Так и запишем

const DATA = {

whichSite:

};

Это будет массив на 3 элемента. Первое значение берем из value нашего input это landing

const DATA = {

whichSite: ['landing',]

};

Второй input - это многостраничный сайт. У него значение multiPage

const DATA = {

whichSite: ['landing', 'multiPage',]

};

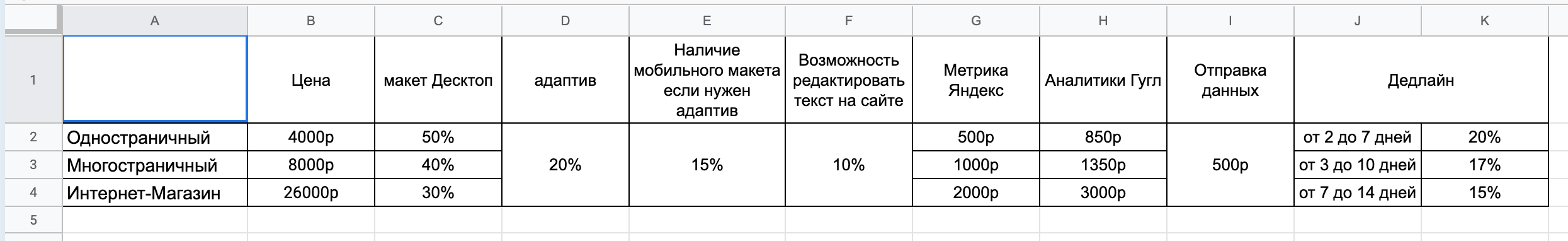
Следующая - интернет магазин с value = onlineStore

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

};

========================================



Колонка B в нашей таблице - это цена prise

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise:

};

Переносим данные из таблицы

const DATA = {

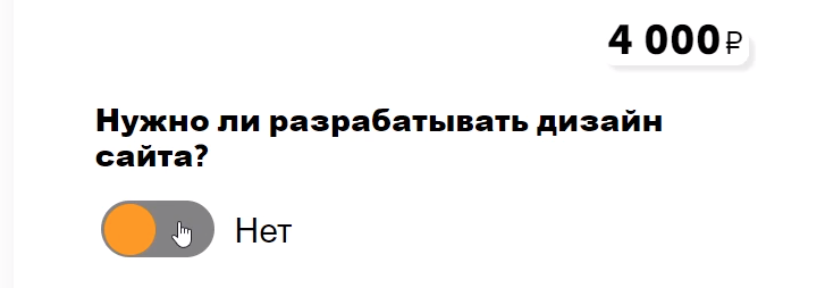
whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

};

================================

Дальше у нас по макету идет :



Значение этого input

value="desktopTemplates"

Создаем ключ с таким именем

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates:

};

Это также колонка С в нашей таблице - переносим значения

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

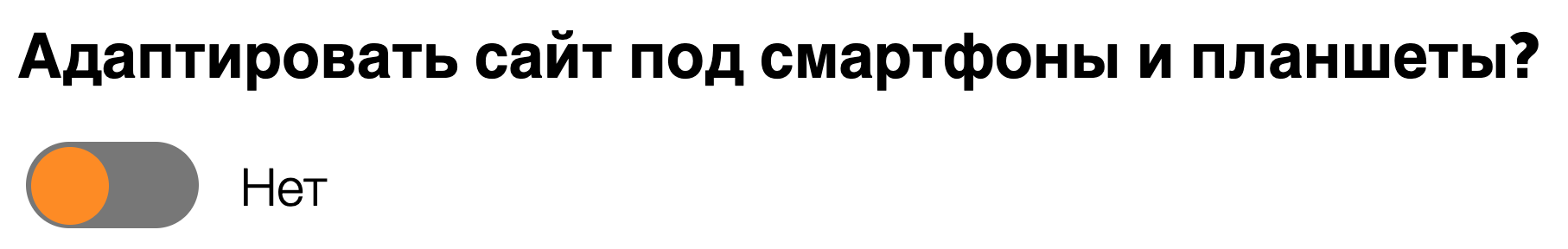
prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

};

==============================

Дальше идет:



В верстке value для этого input = adapt

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

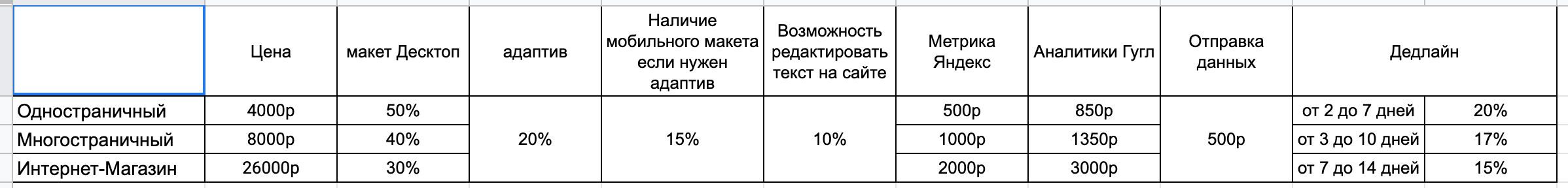
prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt:

};

В нашей такблице это значения из колонки D



const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

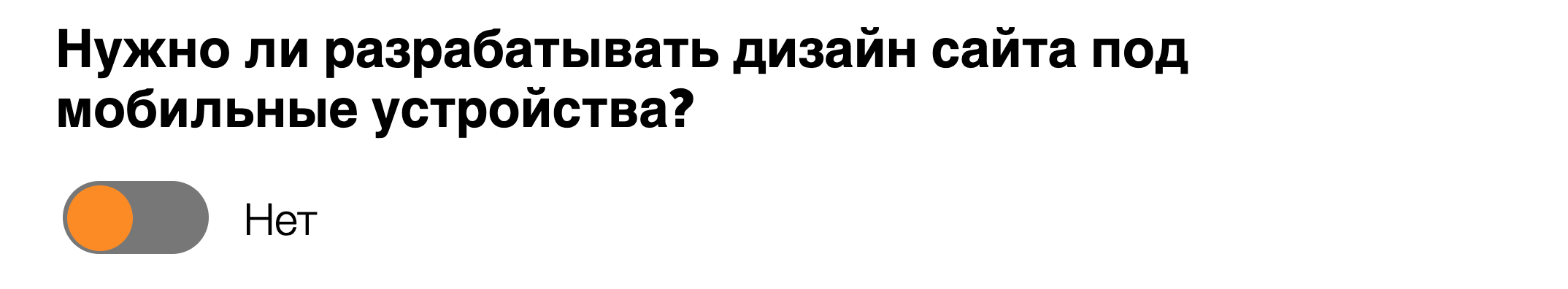
desktopTemplates: [50, 40, 30],

adapt: 20,

};

===========================

Дальше в верстке наличие мобильного макета



value="mobileTemplates"

Код будет выглядеть:

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

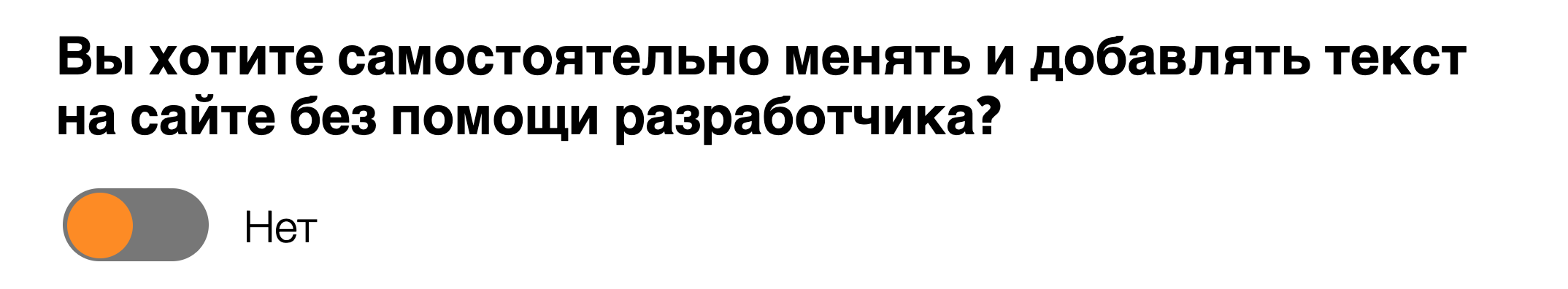
desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

};

Следующий пункт:



value = "editable"

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

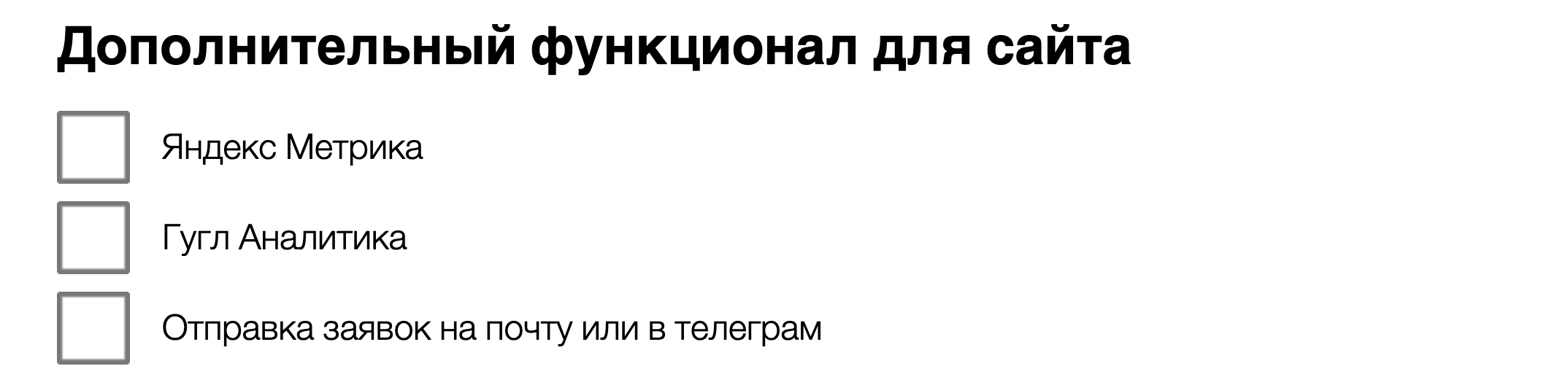
adapt: 20,

mobileTemplates: 15,

editable: 10,

};

Дальше идут:





const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

editable: 10,

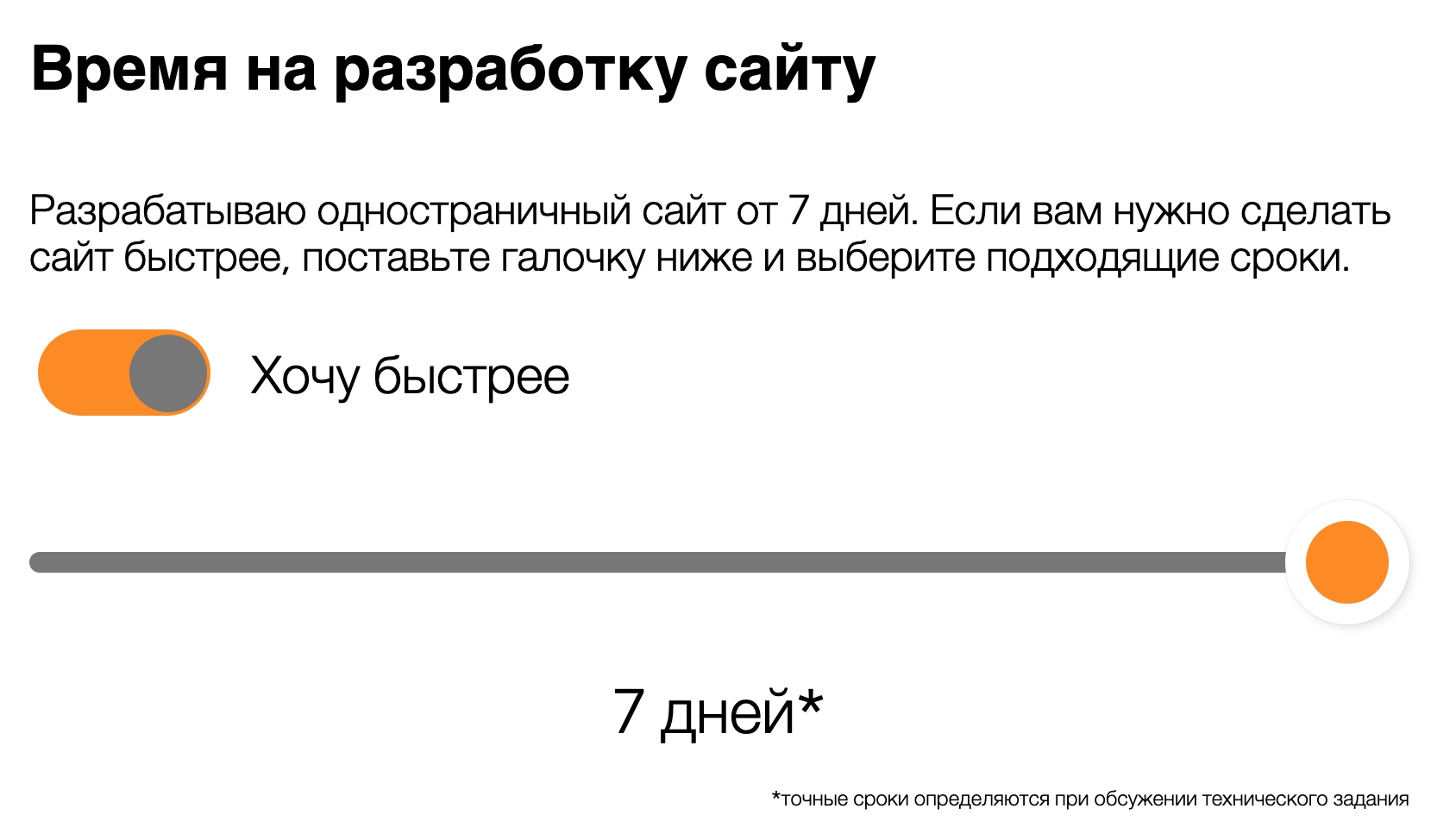
metrikaYandex: [500, 1000, 2000],

analyticsGoogle: [850, 1350, 3000],

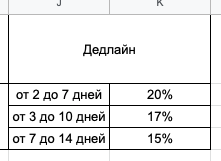
sendOrder: 500,

};

Последний пункт:



Там у нас две колонки



const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

editable: 10,

metrikaYandex: [500, 1000, 2000],

analyticsGoogle: [850, 1350, 3000],

sendOrder: 500,

deadlineDay: [[], [], []],

};

Получился массив, элементами котрого являются массивы

Заполним его для дней

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

editable: 10,

metrikaYandex: [500, 1000, 2000],

analyticsGoogle: [850, 1350, 3000],

sendOrder: 500,

deadlineDay: [[2, 7], [3, 10], [7, 14]],

};

для процентов

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

editable: 10,

metrikaYandex: [500, 1000, 2000],

analyticsGoogle: [850, 1350, 3000],

sendOrder: 500,

deadlineDay: [[2, 7], [3, 10], [7, 14]],

deadlinePercent: [20, 17, 15],

};

===================================

Весь код сейчас выглядит так

'use strict';

const DATA = {

whichSite: ['landing', 'multiPage', 'onlineStore'],

prise: [4000, 8000, 26000],

desktopTemplates: [50, 40, 30],

adapt: 20,

mobileTemplates: 15,

editable: 10,

metrikaYandex: [500, 1000, 2000],

analyticsGoogle: [850, 1350, 3000],

sendOrder: 500,

deadlineDay: [[2, 7], [3, 10], [7, 14]],

deadlinePercent: [20, 17, 15],

};

const startButton = document.querySelector('.start-button');

const firstScreen = document.querySelector('.first-screen');

const mainForm = document.querySelector('.main-form');

const formCalculate = document.querySelector('.form-calculate');

const endButton = document.querySelector('.end-button');

const total = document.querySelector('.total');

const fastRange = document.querySelector('.fast-range');

function showElem(elem) {

elem.style.display = 'block';

}

function hideElem(elem) {

elem.style.display = 'none';

}

function handlerCallBackForm(event) {

const target = event.target;

*if* (target.classList.contains('want-faster')){

target.checked ? showElem(fastRange) : hideElem(fastRange);

}

}

startButton.addEventListener('click', function(){

hideElem(firstScreen);

showElem(mainForm);

});

endButton.addEventListener('click', function(){

*for* (const elem of formCalculate.elements) {

*if* (elem.tagName === 'FIELDSET') {

hideElem(elem);

}

}

showElem(total);

});

formCalculate.addEventListener('change', handlerCallBackForm);

================================

В нашей верстке у всех элементов, которые имеют влияние на конечную цену, есть класс "calc-handler"

И мы сейчас в нашей ф-ции handlerCallBackForm будем отлавливать изменения в этих элементах

И запускать ф-цию, которая будет нам все это подсчитывать

target - это элемент в котром произошло событие

*if* (target.classList.contains('calc-handler')){

priceCalculation(target);

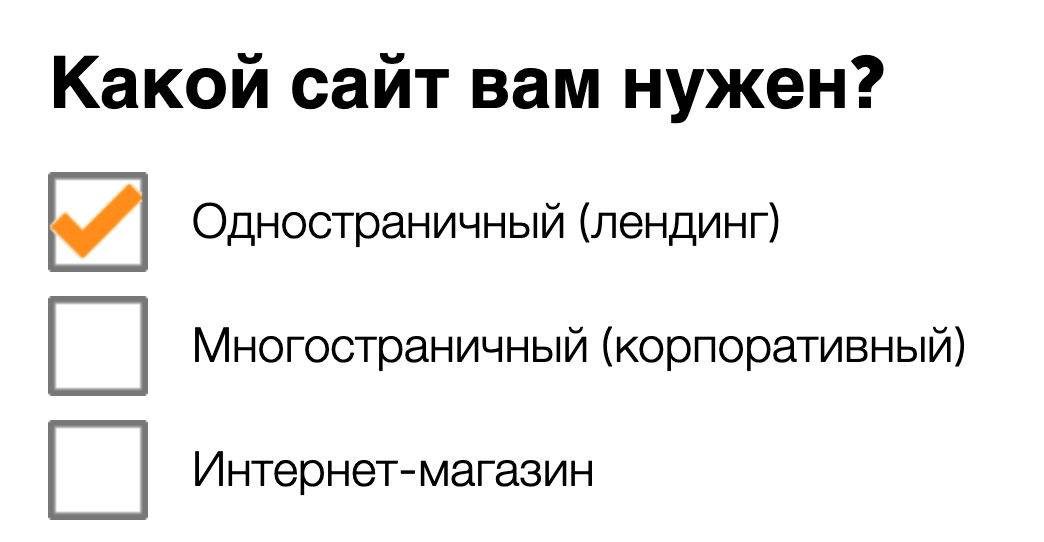
}

где priceCalculation - функция, которую мы напишем позже

function priceCalculation(elem) {

}

1) мы пропишем, чтобы когда пользователь выбирает сайт здесь



Все остальные галочки и выборы сбрасывались и калькулятор показывал начальную стоимость

Для этого все полученные элементы target мы проверим на то являются ли они input с name="whichSite" и если да, циклом переберем все остальные элементы поменяем в значение false

Условие и перебор элементов:

function priceCalculation(elem) {

*if* (elem.name === 'whichSite') {

*for* (const item of formCalculate.elements) {

}

}

}

Теперь возвращаем в false если это чекбокс

function priceCalculation(elem) {

*if* (elem.name === 'whichSite') {

*for* (const item of formCalculate.elements) {

*if* (item.type === 'checkbox') {

item.checked = false;

}

}

}

}

Также после этого выбора, нужнос крыть элемент выбора количества дней на выполнение задания

function priceCalculation(elem) {

*if* (elem.name === 'whichSite') {

*for* (const item of formCalculate.elements) {

*if* (item.type === 'checkbox') {

item.checked = false;

}

}

hideElem(fastRange);

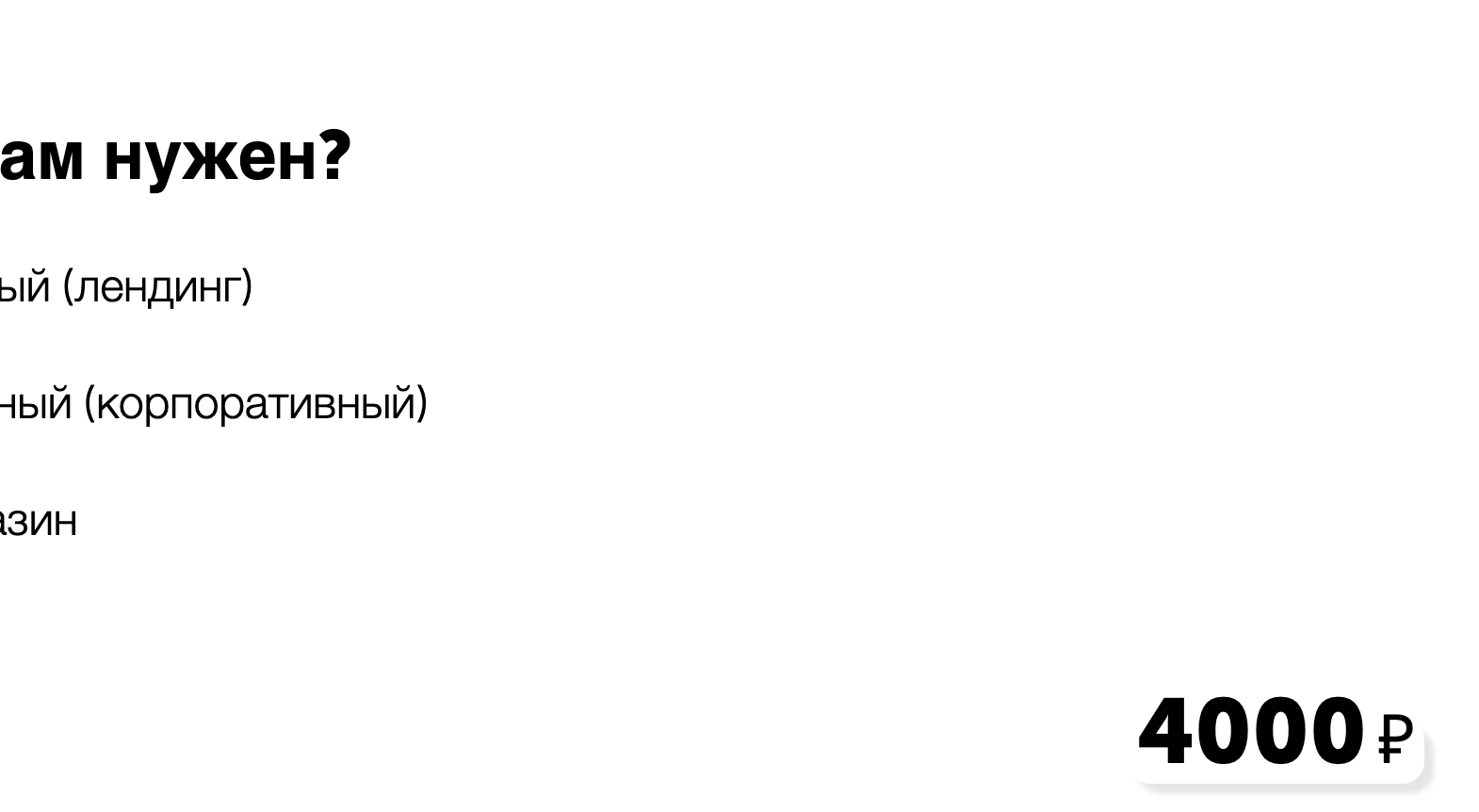
}

}

Проверяем... Все работает! Ура)))

========================================

Теперь необходимо на экран выводить актуальную стоимость сайта. Сюда:



Для этого первым делом получим этот элемент

У него есть класс total-prise-sum

const totalPriceSum = document.querySelector('.total\_price\_\_sum');

Теперь в функции priceCalculation создадим переменную result

И теперь эта ф-ция будет в нашу новую переменную totalPriceSum водить значение result

А чтобы вывести значение в какой-то эллемент, мы после его имени ставим точку пишем textContent

function priceCalculation(elem) {

let result = 0;

*if* (elem.name === 'whichSite') {

*for* (const item of formCalculate.elements) {

*if* (item.type === 'checkbox') {

item.checked = false;

}

}

hideElem(fastRange);

}

totalPriceSum.textContent = result;

}

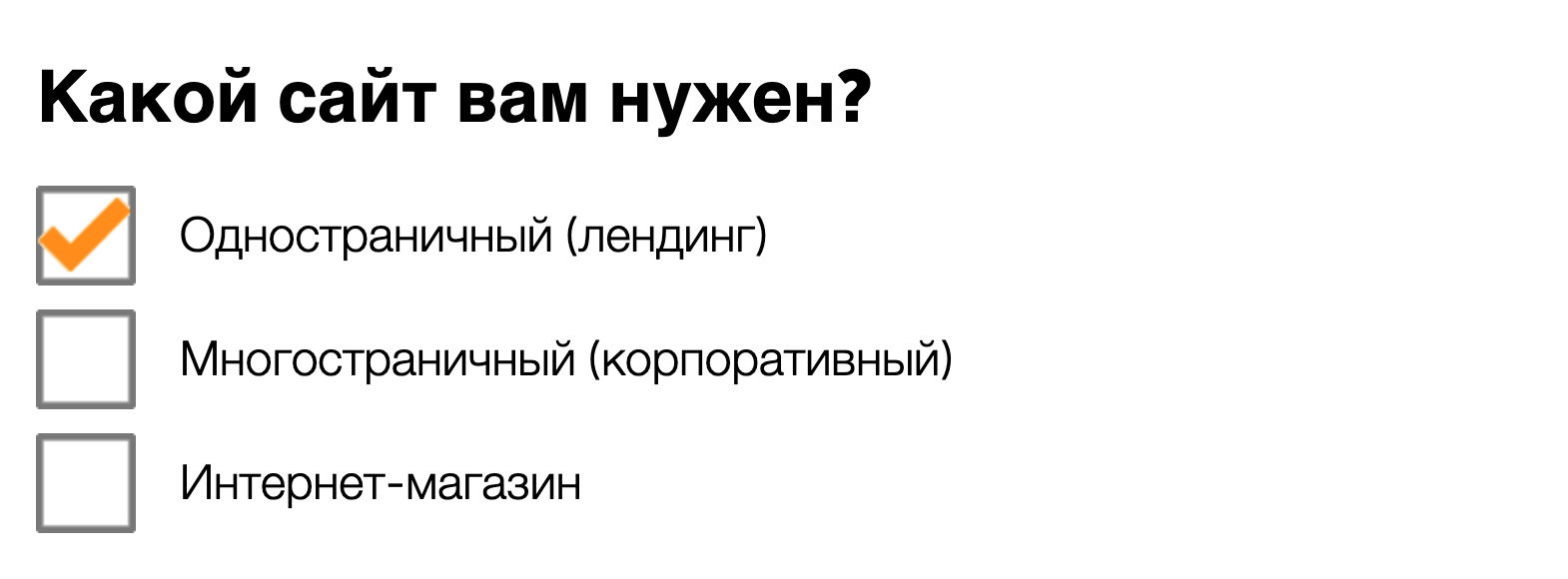
==============================

теперь нам нужно этот самый result посчитать. Для этого перебираем все элементы формы

Для индексов создадим переменную, которая будет изначально равна 0

let index = 0;

Теперь, пербирая все элементы, проверяем, что нужный является галочкой при выборе сайта



И он выбран:

*for* (const item of formCalculate.elements) {

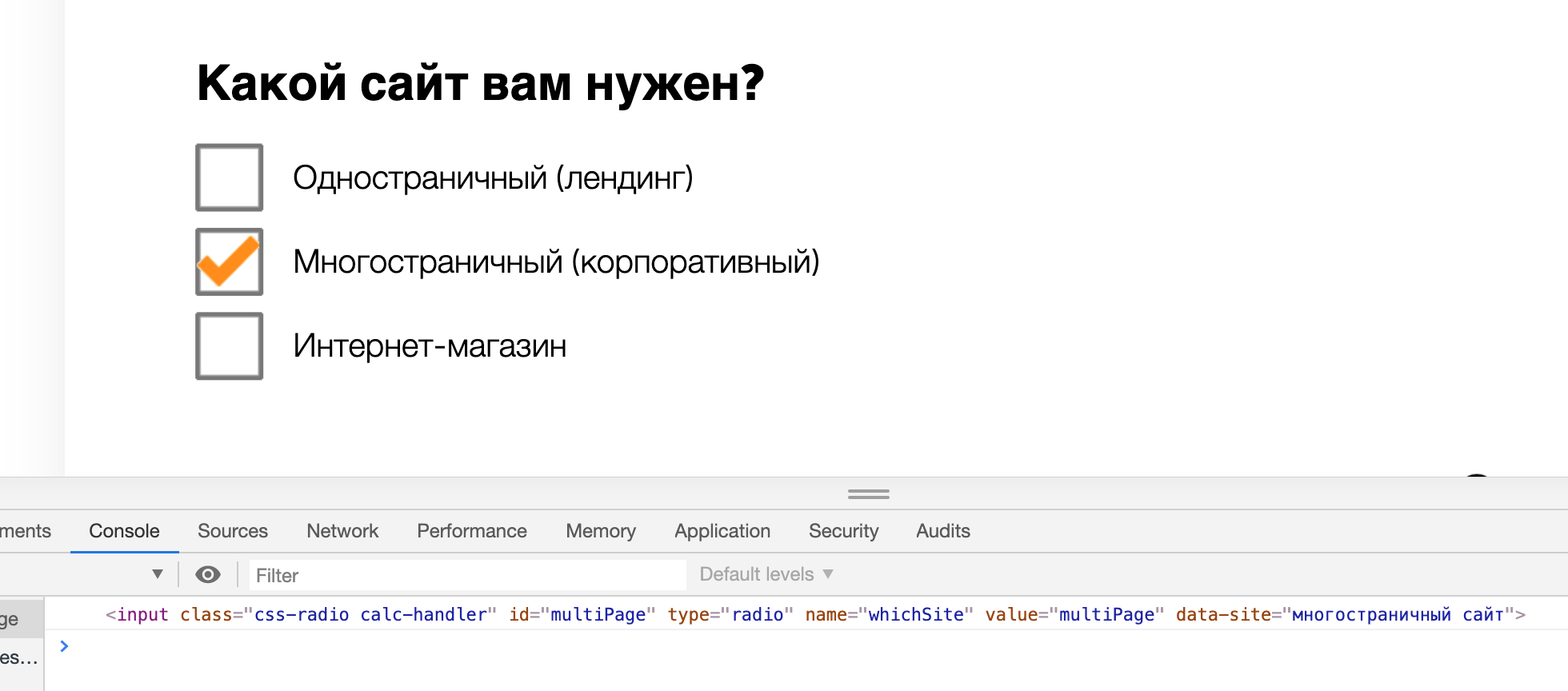
*if* (item.name === 'whichSite' && item.checked) {

console.log(item);

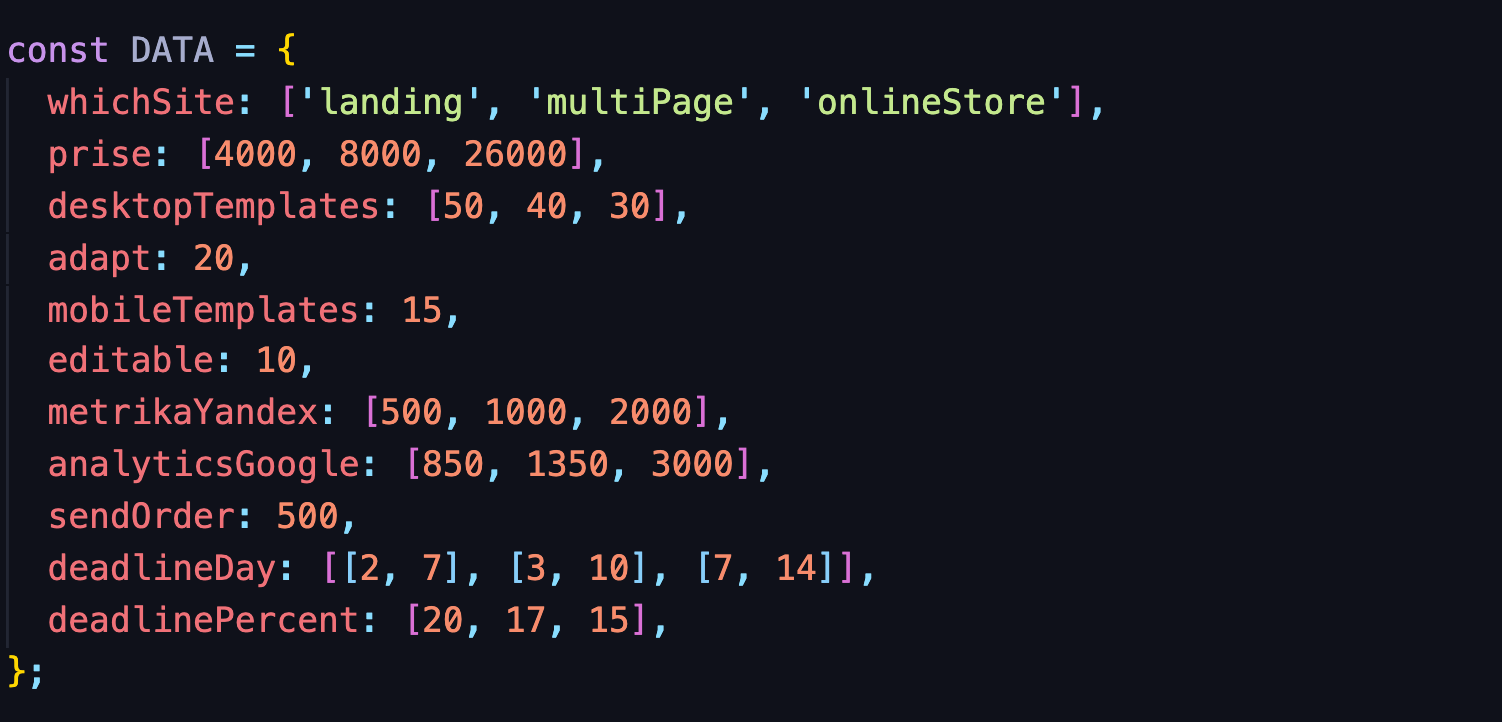
}

};

Теперь при нажатии на любую галочку в консоли должно появиться



То есть, какой элемент выбрали, тот мы и получили. Нам из него нужно value - его значение будет совпадать со значениями wichSite объекта DATA, который мы создавали ранее



Следовательно, будем брать наш объект DATA, его свойство wichSite и с помощью метода indexOf будем искать этот элемент по значению value в масиве

В итоге у нас сохранится значение его индекса в нашем DATA

*for* (const item of formCalculate.elements) {

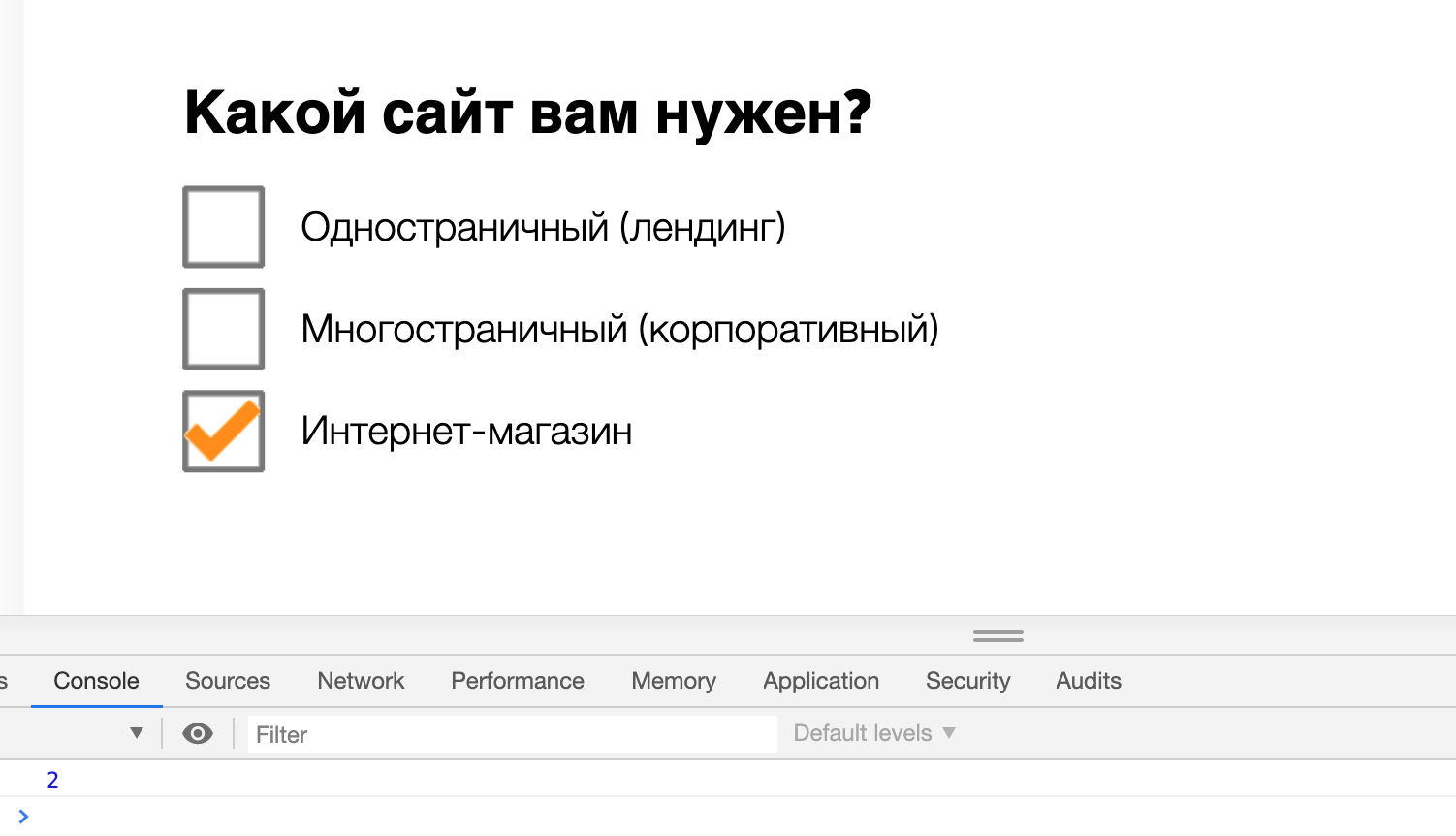
*if* (item.name === 'whichSite' && item.checked) {

console.log(DATA.whichSite.indexOf(item.value));

}

};

Получили:



Будем сохранять это значение в переменную index

*for* (const item of formCalculate.elements) {

*if* (item.name === 'whichSite' && item.checked) {

index = DATA.whichSite.indexOf(item.value);

}

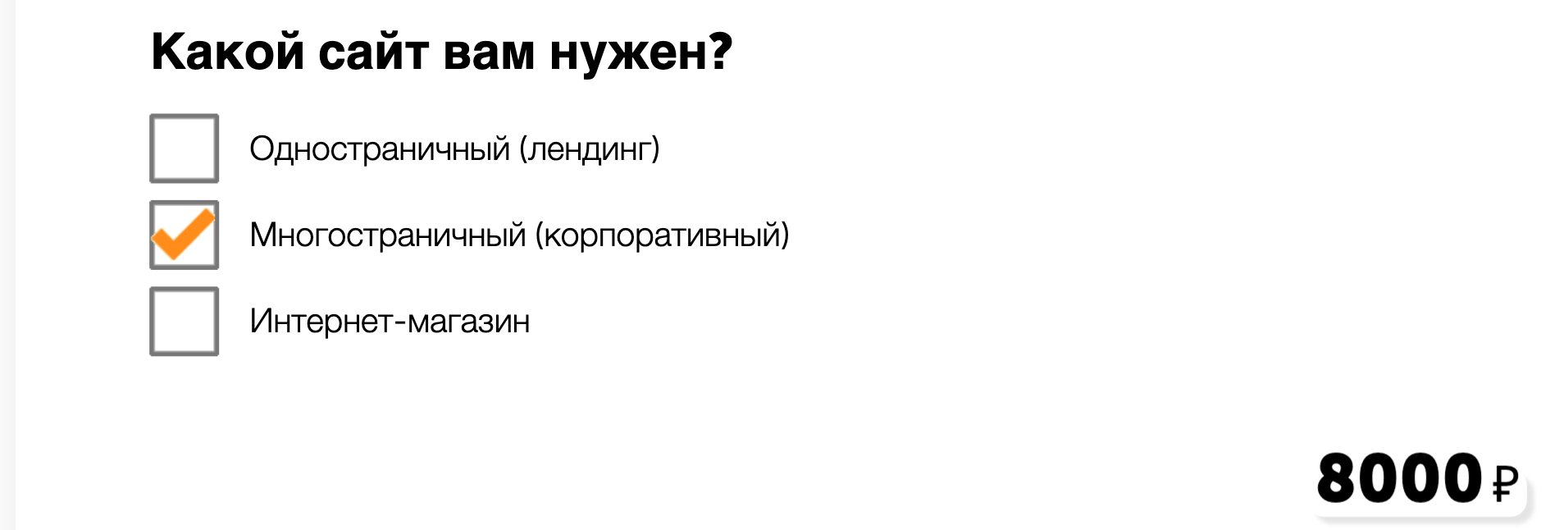
};

===============================

Теперь по текущему индексу мы можем получить значение DATA.price

result = DATA.prise[index];

Должно получиться:



Для того, чтобы мы в будущем могли добавлять до этого значения суммы из чекбоксов:

result = result + DATA.prise[index];

В этом же переборе всех элементов, если это не wichSite (через else) а наш класс calc-handler (чекбоксовя ющих на цену)

*else* *if* (item.classList.contains('.calc-handler')) {

}

Весь перебор:

*for* (const item of formCalculate.elements) {

*if* (item.name === 'whichSite' && item.checked) {

index = DATA.whichSite.indexOf(item.value);

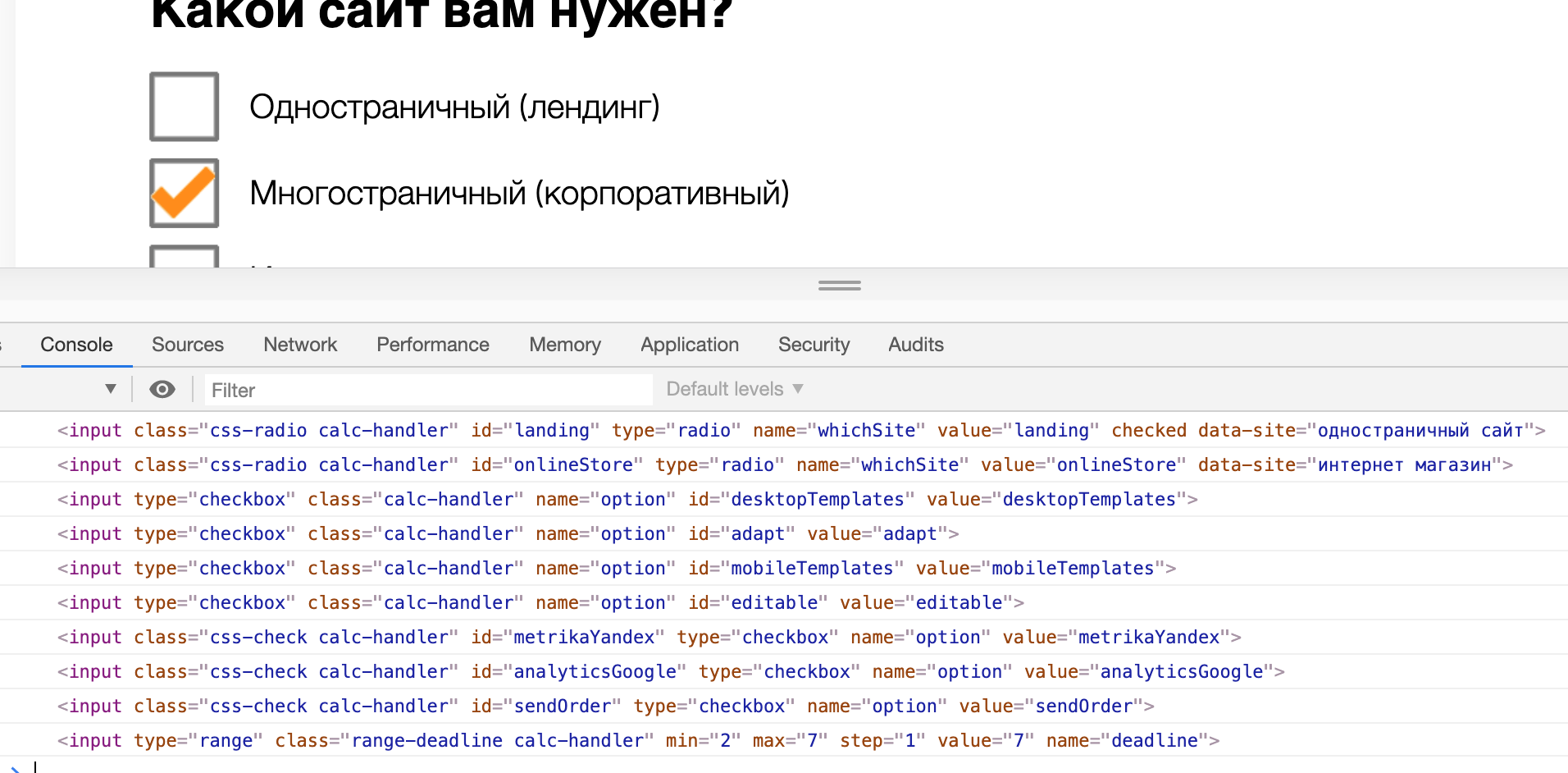
} *else* *if* (item.classList.contains('calc-handler')) {

console.log(item);

}

};

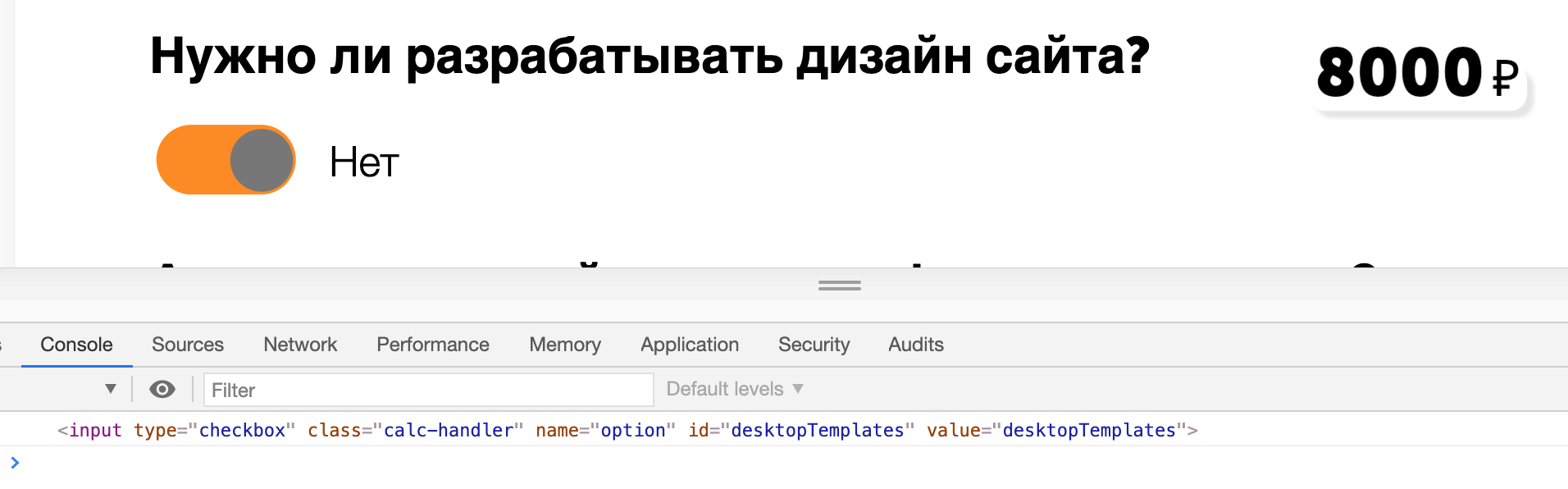
Теперь, выбирая любой сайт из wichSite мы будем получать все элементы, которые могут влиять на его цену:



Теперь выберем те елементы, которые имеют значение true (бфли выбраны)

Для этого в нашем последнем if через И добавим условие, что item.checked

Теперь будет появляться только тот элемент, который мы выбрали



Но нам нужен не весь элемент, а только его value

*for* (const item of formCalculate.elements) {

*if* (item.name === 'whichSite' && item.checked) {

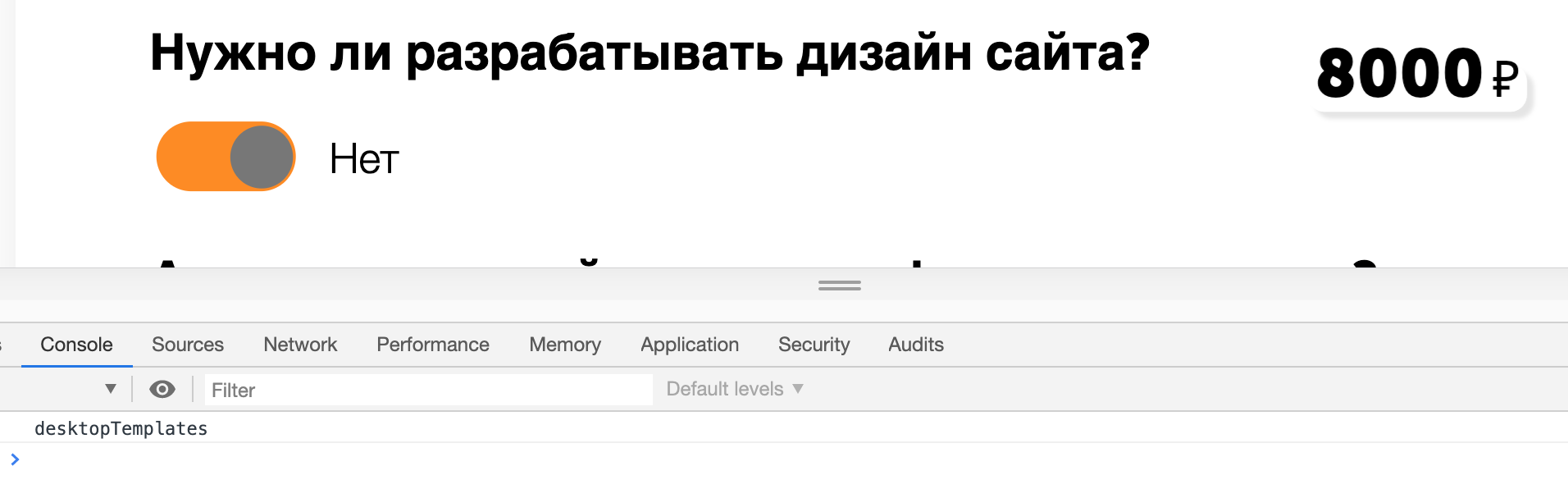
index = DATA.whichSite.indexOf(item.value);

} *else* *if* (item.classList.contains('calc-handler') && item.checked) {

console.log(item.value);

}

};



=====================

создадим пустой массив и будем в него добавлять те значение value, что были выбраны

let options = [];

будем их добавлять с помощью метода push

options.push(item.value);

Все вместе теперь выглядит:

*for* (const item of formCalculate.elements) {

*if* (item.name === 'whichSite' && item.checked) {

index = DATA.whichSite.indexOf(item.value);

} *else* *if* (item.classList.contains('calc-handler') && item.checked) {

options.push(item.value);

}

};

Вся ф-ция:

function priceCalculation(elem) {

let result = 0;

let index = 0;

let options = [];

*if* (elem.name === 'whichSite') {

*for* (const item of formCalculate.elements) {

*if* (item.type === 'checkbox') {

item.checked = false;

}

}

hideElem(fastRange);

}

*for* (const item of formCalculate.elements) {

*if* (item.name === 'whichSite' && item.checked) {

index = DATA.whichSite.indexOf(item.value);

} *else* *if* (item.classList.contains('calc-handler') && item.checked) {

options.push(item.value);

}

};

result = result + DATA.prise[index];

totalPriceSum.textContent = result;

}

Теперь, испльзуя метод forEach будем перебирать все элементы массива. И при каждом из них выполнять ф-цию

options.forEach(function(){

})

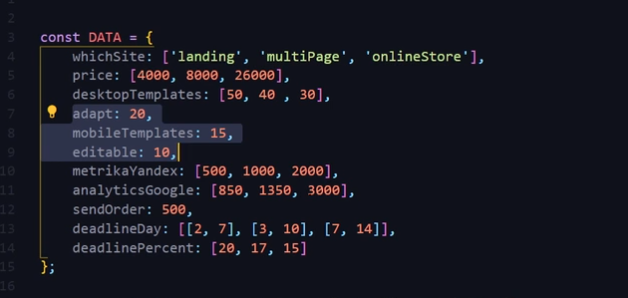
Получать мы будем ключи

options.forEach(function(key){

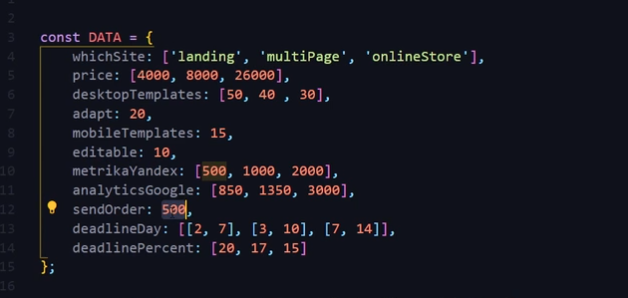
})

А в самой ф-ции будем проверять, какой у нас тип данных

Если тип данных в нашем объекте по этому ключу будет число, (то есть вот это

 )

то нам нужно умножать на проценты.

Если это будет sendOrder (), то просто плюсуем

options.forEach(function(key){

*if* (typeof(DATA[key]) === 'number') {

*if* (key === 'sendOrder') {

result += DATA[key]

} *else* {

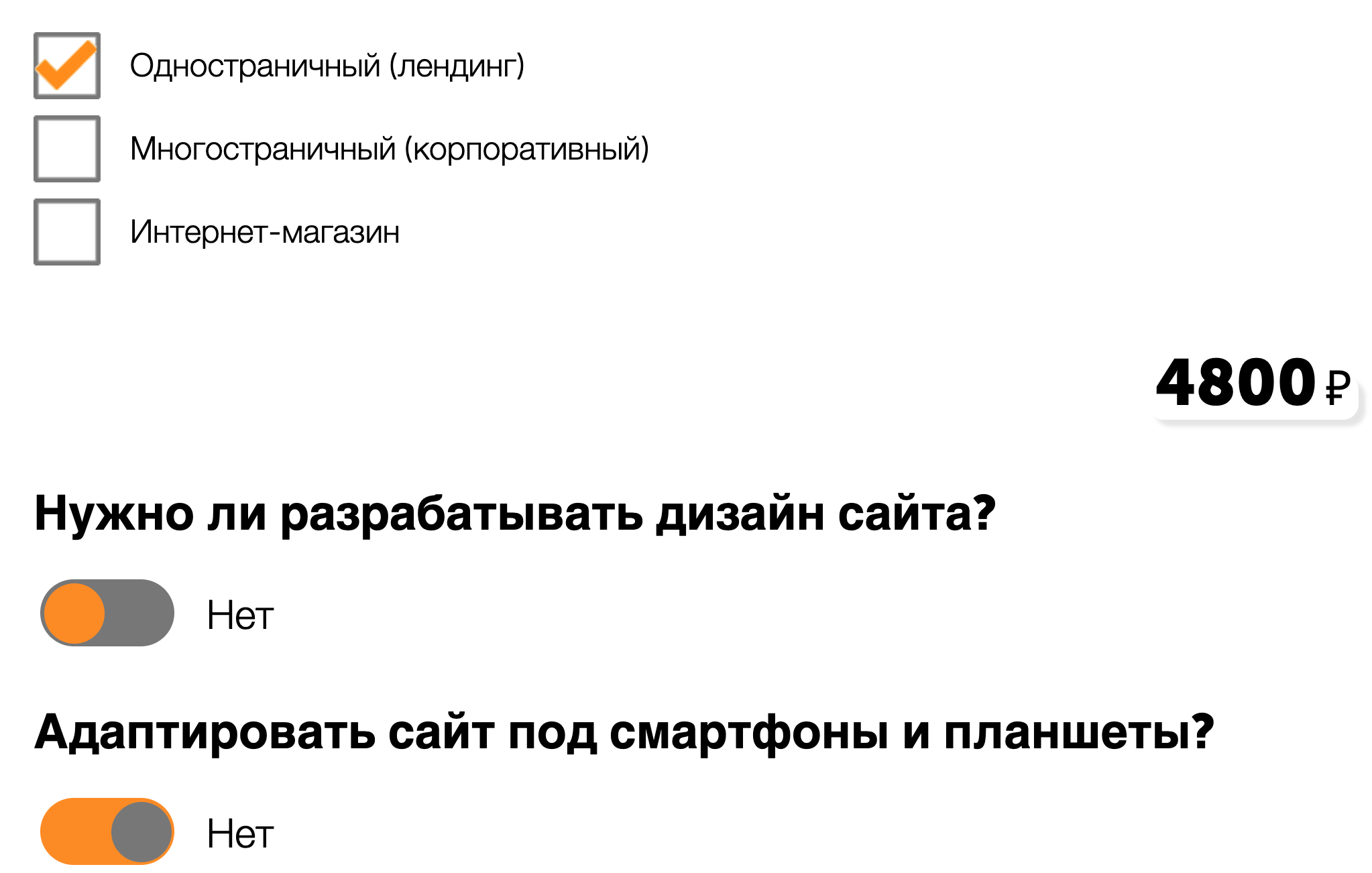
result += DATA.prise[index] \* DATA[key] / 100;

}

}

})

При проверках накидывается цена:



=========================

Но, это работает, только если начения DATA обычные number

Как быть с массивами?

Так и пишем, если

typeof(DATA[key])

не число, то если ключ у нас desktopTemplates (там проценты, а в остальных массивах просто рубли)

*else* {

*if* (key === 'desktopTemplates') {

result += DATA.price[index] \* DATA.desktopTemplates[index] / 100;

}

}

===================

Осталось получить

