УДК 004

Метод упрощения Pixel Perfect вёрстки

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматривается метод упрощения вёрстки. Был проведён эксперимент с PSD-макетом и проверена работоспособность скрипта.
Ключевые слова: Perfect pixel, вёрстка сайтов, PSD-макет.

Pixel Perfect layout simplification method

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article discusses the method of simplifying layout. An experiment was conducted with a PSD-layout and the script was tested to work.
Keywords: Perfect pixel, site layout, PSD-layout.

Спрос на создание веб-сайтов растёт с каждым днём, как и востребованность веб-разработчиков. Веб-ресурс всегда имеет пользовательский интерфейс, именно поэтому frontend разработка является самой популярной задачей среди веб-разработчиков.

Frontend - это разработка пользовательского интерфейса и функциональности, которые работают на клиентской стороне веб-сайта или приложения.

Каждый веб-разработчик хоть раз выполнял задачу по frontend разработке. Чаще всего задача состоит в переносе пользовательского интерфейса из PSD-макета в браузер. К таким задачам идет техническое задание, в котором пишется определённый комментарий. Чаще всего они делятся на 2 типа:

1. Приближённый вариант
2. Pixel Perfect

Pixel Perfect верстка - это особая техника создания структуры html-кода, которая позволяет сверстанному html-шаблону максимально точно совпадать с оригинальным PSD-макетом пиксель в пиксель. При наложении html-шаблона на макет PSD должно произойти полное совпадение графических элементов, изображений и текста во всех его расширениях.

Естественно Perfect вёрстка более кропотлива и отнимает огромное количество времени. Существуют различные методы для упрощения работы, и мы рассмотрим один из них.

Для того, что решить проблему, нужно понимать, что отнимает больше всего времени.

Выделенные проблемы:

1. Не удобно переключаться, чтобы посмотреть размеры.
2. После каждой секции приходится проверять и исправлять.

Все это сильно мешает работе и увеличивает время выполнение в 2-3 раза.

Каждый день веб-разработчики по-своему пытаются решить проблему вёрстки по типу Pixel Perfect. В статье [1] проводится ознакомление с Pixel Perfect. На ресурсе [2] приведён свой метод для работы с Pixel Perfect. Так же на ресурсе [3] приводится свой метод реализации адаптивной Pixel Perfect вёрстки.

В данной статье рассмотрим один из методов решения всех выше поставленных проблем. В этом нам поможет JavaScript код, приведённый ниже.

```
const img = IMG ADDRESS;
const opacity_default = IMG OPACITY;
let image;
window.onload = () => {
	const sect = document.createElement('section');
	sect.id = "temp";
	sect.style.pointerEvents = 'none';
	const styles = document.createElement('style');
	styles.innerHTML = `
		#temp_img {
			width:100%;
			z-index:9999999;
			position:absolute;
			top:0;
			left:0;
		}
		.temp_nav {
			z-index:99999999;
			position:fixed;
			top:auto;
			pointer-events: visiblePainted;
			left:auto;
			right:0;
			bottom:0;
			background:rgba(0,0,0,0.5);
		}
```

```
            .temp_nav input {
                width:300px;
                height:50px;
            }
            .temp_nav button {
                width:70px;
                height:50px;
                font-size:20px;
            }
        `;
        sect.innerHTML = `
            <img src="${ img }" id="temp_img">
            <div class="temp_nav">
                <input type="range" min="0" max="1" step="0.1" value="${
opacity_default ? opacity_default : '1'}" onchange="range(this)">
                <button type="button" onclick="disp(this)">OFF</button>
            </div>
        `;
        document.head.appendChild(styles);
        document.body.appendChild(sect);

        image = document.getElementById('temp_img');
        image.style.opacity = opacity_default ? opacity_default : '1';
}
let display_is = true;
const range = (el) => {
        image.style.opacity = el.value
}
let disp_ = 'off';
const disp = (el) => {
        el.innerHTML = disp_ === 'off' ? 'on' : 'off';
        disp_ = disp_ === 'off' ? 'on' : 'off';
        image.style.display = display_is ? 'none' : 'block';
        display_is = !display_is;
}
```

Единственное, что необходимо будет менять это переменные:

1. img – путь до картинки.
2. opacity_default – прозрачность накладываемого изображения (например 0.5)

Для того, чтобы он работал необходимо подключить файл со скриптом и опередить переменные в нём.

Код делится на 2 части. Сначала берётся нужная нам картинка и растягивается равномерно по всей площади экрана, при этом она должна иметь фиксированную позицию и автоматически занимать всю свободную площадь. Тем самым при смене расширения экрана элементы буду находится

там, где нужно. Вторая часть отвечает за ползунок прозрачности. Это сделано для того, чтобы управлять слоем, который был наложен (см. рис. 1).

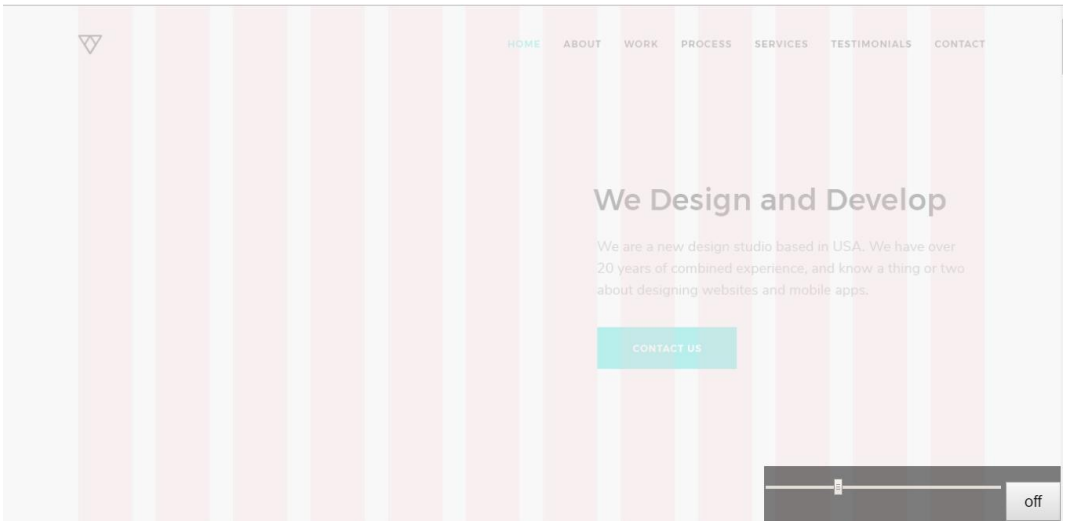

Рисунок 1. Работоспособность скрипта

На пустую страницу наложен слой нашего макета, который полностью адаптивен (то есть при изменении расширения экрана, изображение подстраивается под него). Справа в углу есть ползунок, отвечающий за прозрачность и кнопка, с помощью которой мы можем отключить скрипт. Смысл заключается в том, что мы теперь наглядно видим, как на странице должны располагаться элементы. Теперь попробуем добавить какой-нибудь элемент (см. рис. 2).

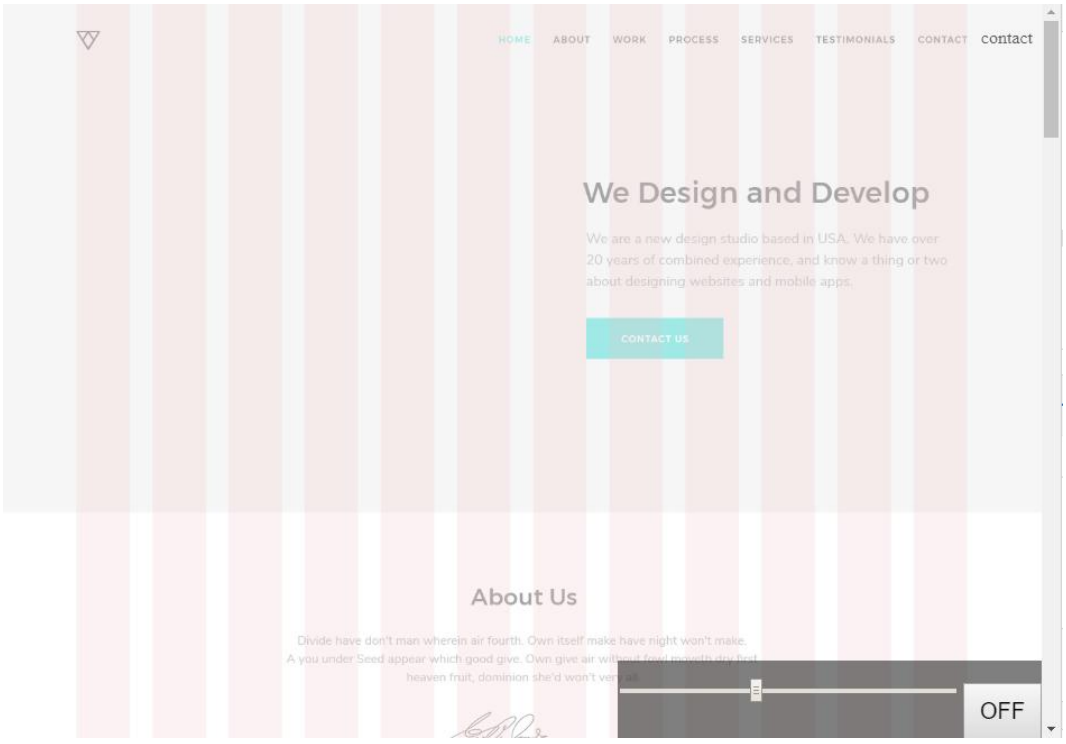

Рисунок 2. Проверка наложения элементов

Справа вверху мы видим 2 слова contact, одно сделано нами, другое находится в макете. Всё что необходимо это сделать 100% совпадение элементов и это называется Pixel Perfect вёрстка. Теперь нам не нужно по несколько раз пересматривать макет и сравнивать на глаз. Вся работа будет точно выполнена на 100% и не заставит сомневаться в размерах или месторасположении элементов.

В результате рассмотренный скрипт полностью решает проблемы данной задачи, делает разработку более удобной и простой. Сам скрипт динамичен, его можно подключать к любой странице и выбирать другие изображение.

Библиографический список

1. Знакомимся с Pixel Perfect // zencoder URL: http://zencoder.ru/web-development/pixel-perfect/ (дата обращения: 30.08.2019.)
2. Pixel perfect верстка // habr URL: https://habr.com/ru/post/195414/ (дата обращения: 30.08.2019.)
3. Адаптивный Pixel Perfect // css.yoksel URL: http://css.yoksel.ru/adaptive-pixel-perfect/ (дата обращения: 30.08.2019.)