**CSS**

**css\_4. Что такое padding?**

Свойство padding — это внутренний отступ — позволяет оттолкнуть контент от границ родительского элемента.

В качестве значения можно указать одно, два, три или четыре числа. Браузер будет читать это значение по-разному в зависимости от количества указанных значений.

* padding: 10px 15px 20px 30px — читается по часовой стрелке начиная с верхней границы. Сверху—> справа—> снизу—> слева.
* padding: 15px — для всех четырёх сторон 15 px.
* padding: 15px 20px — Первое значение для верхнего и нижнего отступов. Второе значение для боковых отступов.
* padding: 10px 5px 25px — первое значение для верхнего отступа, второе значение для боковых, последнее для нижнего отступа.

Вертикальные отступы не схлопываются так как margin.

**css\_14. Что делает border и какие значения может принимать?**

Свойство border отвечает за отрисовку рамки - видимой границы блока. Само свойство border является шорткатом (короткой записью) и объединяет в себе значение для нескольких полных свойств: border-width, border-style, border-color. Значения можно писать в любом порядке, интерпретатор браузера разберётся. Главное, разделять их пробелами. Но общепринятым является порядок: ширина → стиль → цвет рамки. Можно рисовать фигуры рамками.

**css\_14.1 border-style**

Задает стиль рамки элемента. Свойство border-style является шорткатом для свойств: border-bottom-style, border-left-style, border-right-style, border-top-style. Может принимать значения:

* none — отключает рамку (значение по умолчанию).
* solid — сплошная рамка. Этот стиль используется чаще всего.
* dotted — рамка состоит из точек с пробелами между ними.
* dashed — рамка состоит из коротких чёрточек с пробелами между ними.
* double — рамка состоит из двух сплошных линий с небольшим нерегулируемым отступом между ними.

Есть еще и другие варианты рамок объемные с тенями.

**css\_14.2 border-color**

Управляет цветом рамки. Можно писать одно, два, три или четыре значения, разделяя их пробелами. В зависимости от этого получится разный результат:

Одно значение - цвет рамки со всех сторон одинаковый.

Два значения - первое устанавливает цвет рамки сверху и снизу, второе — слева и справа.

Три значения - первое значение устанавливает цвет рамки сверху, второе — слева и справа, а третье — снизу.

Четыре значения — четыре значения, устанавливает для каждой стороны свой цвет, поочерёдно: для верхней, правой, нижней и левой рамки.

Значение по умолчанию соответствует значению color у элемента.

**css\_14.3 border-width**

Управляет толщиной рамки элемента. Значением может быть число с любыми единицами измерения, доступными в вебе, кроме процентов. Разрешается указывать одно, два, три или четыре значения. Для задания толщины можно использовать переменные - thin, medium или thick, значение переменных определяется браузерами.

**css\_14.4 border-radius**

Задаёт радиус закругления каждого из углов элемента. Значение в процентах будет высчитываться от размеров элемента: горизонтальные размеры будут высчитываться от ширины элемента, а вертикальные — соответственно, от высоты. Это свойство является шорткатом для четырёх свойств border-top-left-radius, border-top-right-radius, border-bottom-right-radius, border-bottom-left-radius.

Свойство border-radius может быть задано как:

одно, два, три или четыре значения - для задания обычного радиуса углов.

одна, две, три или четыре пары значений разделённые "/". Используется для задания эллиптического скругления.

**css\_14.5 Можно ли задать стили для каждой рамки отдельно (например, отдельно для левой и для правой рамок)?**

Можно! При помощи следующих свойств: border-top, border-right, border-bottom, border-left. Все четыре рамки могут быть разными по толщине, цвету и стилю.

**React**

**rct\_1. Что такое React?**

Это JavaScript-библиотека для создания пользовательских интерфейсов. React использует компонентную архитектуру, декларативный стиль программирования и быстрые, удобные для разработчика абстракции DOM, iOS, Android и др. React обеспечивает высокую производительность благодаря виртуальной модели DOM. Плюс огромное сообщество, которое предоставляет множество различных готовых компонентов.

**rct\_1.1. Что такое ReactDOM?**

Библиотека, которая содержит методы для рендеринга DOM, а также рендеринга на стороне сервера.

**rct\_2. Что такое декларативное UI программирование? Чем оно отличается от императивного подхода?** 

Императивное программирование — оно же «приказывающее» или «повелительное» — это парадигма (стиль написания исходного кода компьютерной программы), в которой описываются последовательные команды, которые должен совершить процессор, чтобы получить необходимый нам результат. Т.е. это такой стиль программирования, при котором описывается, как добиться желаемого результата.

Языки: C, C++;

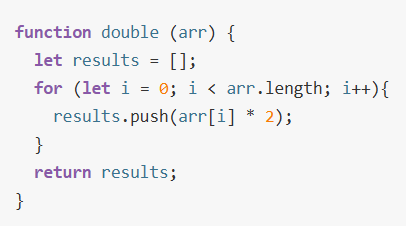
Декларативное программирование — это парадигма (стиль написания исходного кода компьютерной программы), программирования, в которой описывается ожидаемый результат, а не способ его получения.

Языки: SQL, HTML.

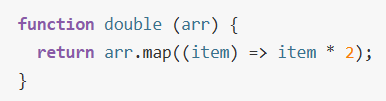
Неважно, как будут выбраны пользователи из Мексики и как браузер распарсит article. Важно, что мы получим мексиканских пользователей и новый header и paragraph на сайте, не углубляясь в инструкции.

Например: Дан массив чисел, написать функцию, которая вернет массив, где каждое число -> [1, 2, 3] -> [2, 3, 6]

Императивный стиль:



Декларативный стиль:



**rct\_3. Что такое React Element? Как его можно создать? Из чего он состоит?** 

React-элемент — это не то, что будет на экране, а описание того, что будет на экране. React-элемент описывает узел DOM в виде javascript-объекта. Объект состоит из свойств type(тип тега), props(атрибуты), children(дочерние узлы), и др.

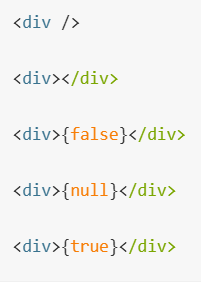
Создать можно:

1. React.createElement(type, props, children)
2. Используя синтаксис JSX.

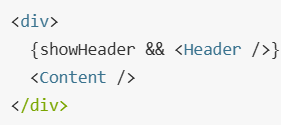
Если коротко: элементы являются экземплярами компонентов, которым могут передаваться свойства.

**rct\_3.1 Что будет, если передать булевые, null или undefined значения в children?**

Они игнорируются. Их можно использовать в качестве потомков, но данные значения не отображаются при рендеринге. Следующие JSX выражения будут отображены одинаково:



Вместе с условными выражениями это можно использовать для условного отображения элементов React:



**rct\_3.2 Что такое React Component?** 

React-компонент — это функция или класс, который принимает входные данные (опционально) и возвращает React-элемент — то есть, javascript-объект. Пользовательский интерфейс создается с помощью комонентов.

**rct\_4 Что такое JSX и зачем он нужен?**

**JSX представляет собой синтаксический сахар на базе JavaScript, позволяющий**

**записывать элементы React на JavaScript в HTML-подобной записи с <>.**

JavaScript XML (JSX) — это синтаксическое расширение JavaScript предоставляющее «синтаксический сахар» для вызовов функций и построения объектов, особенно React.createElement() (иначе говоря, он упрощает процесс написания кода, сохраняя его функциональность). JSX ускоряет создание элементов React, делая их более компактными, а также упрощает их чтение. JSX создан только для разработчиков. Сам по себе он никак не ускоряет и не улучшает веб-приложения. JSX преобразуется в тот же код, который получается и без JSX.

Основные плюсы:

- Улучшенное взаимодействие с разработчиком (DX, Developer Experience) — код проще читать благодаря XML-подобному синтаксису, который хорошо подходит для написания вложенных декларативных структур.

- Улучшенные сообщения об ошибках — React по умолчанию предполагает использование JSX и выводит полезные сообщения об ошибках. Если JSX не используется, эти сообщения будут запутывать разработчика, так как в них будет упоминаться не тот синтаксис, который он реально использует.

- Ускорение кода — при преобразовании JSX в JavaScript транспилятор одновременно оптимизирует код, благодаря чему итоговый код JavaScript исполняется быстрее, чем если бы он был введен вручную.

- Повышение эффективности работы команд — неопытным разработчикам (например, веб-дизайнерам) проще менять код, потому что JSX напоминает разметку HTML, которая им уже знакома.

- Снижение количества синтаксических ошибок — разработчики набирают на клавиатуре меньше кода, а это значит, что они сделают меньше ошибок.

Процесс преобразования JSX в JavaScript, называется транспиляция. Транспиляция, а не компиляцией, потому что код транслируется с одного исходного языка (JSX) на другой исходный язык (JavaScript). В свою очередь, полученный код JavaScript интерпретируется «реальным» компилятором, который выполняет код. Транспиляция — всего лишь преобразование синтаксиса, а не интерпретация кода.

При выполнении приложения React браузер видит только команды React.createElement, необходимые для генерирования нужной структуры. JSX существует только в редакторе. Транспилятор преобразует файлы, содержащие JSX, в «чистый» JavaScript с многочисленными вызовами React.createElement().

Процесс транспиляции заключается в следующем: исходный код переписывается, чтобы получить такие же результаты, но уже с помощью синтаксиса, понимаемого устаревшими браузерами. Любой новый синтаксис для браузера, который его не поддерживает, просто недействителен и выдает ошибку разбора.

**rct\_x Что такое stateful и stateless компоненты?**

**rct\_x.x Что делает хук useState? Как обновить состояние?**

Позволяет добавлять состояние в функциональные компоненты. React хранит это состояние между рендерами. Вызов useState возвращает массив с двумя элементами - текущее значение состояния и функцию для его обновления. Единственный аргумент useState — это начальное состояние. Чтобы обновить состояние нужно вызвать функцию обновления состояния с новым значением или коллбэком. После этого React сделает повторный рендер, в котором использует уже новое значение.