**Оглавление**

[**JAVASCRIPT** 10](#_Toc76483865)

[1. Какие типы данных существуют в javascript? 10](#_Toc76483866)

[2. В чем разница между операторами "==" и "==="? 10](#_Toc76483867)

[3. Какими способами можно объявить переменную? 10](#_Toc76483868)

[4. В чем разница между null и undefined? 11](#_Toc76483869)

[6. Стрелочные функции и их отличие от функций, объявленных через function 12](#_Toc76483870)

[7. Что такое замыкание? 12](#_Toc76483871)

[8. Что такое шаблонные литералы и для чего они нужны? 12](#_Toc76483872)

[9. Что такое set и map? 13](#_Toc76483873)

[10. Как определить наличие свойства в объекте? 13](#_Toc76483874)

[11. Какие способы создания объекта вы знаете? 14](#_Toc76483875)

[12. Какие значения являются falsy(ложными) значениями? 14](#_Toc76483876)

[13. Что такое Promise? 15](#_Toc76483877)

[14. Как использовать async/await для асинхронных запросов? 15](#_Toc76483878)

[15. Для чего нужен оператор spread? 16](#_Toc76483879)

[16. Как избежать ссылочной зависимости при копировании объекта? Глубокое копирование объекта 16](#_Toc76483880)

[18. Как поменять контекст функции? 17](#_Toc76483881)

[19- Что такое тернарный оператор? 18](#_Toc76483882)

[20. Что такое деструктуризация 18](#_Toc76483883)

[21. Какие способы работы с асинхронным кодом вы знаете? 19](#_Toc76483884)

[22. e.preventDefault() и e.stopPropagination() для чего нужны? 20](#_Toc76483885)

[23. Как отслеживать и обрабатывать ошибки в javascript? 20](#_Toc76483886)

[24. Что такое DOM дерево? 21](#_Toc76483887)

[25. Что такое event loop? 21](#_Toc76483888)

[26. Что такое прототипное наследование? 22](#_Toc76483889)

[27. Как получить свойство объекта? 22](#_Toc76483890)

[28. Всплытие и погружение событий 22](#_Toc76483891)

[29. Optional chaining оператор. Для чего нужен? 23](#_Toc76483892)

[30. Shadow DOM. 24](#_Toc76483893)

[31. function expression и function declaration. Что это такое и в чем разница? 24](#_Toc76483894)

[32. Для чего нужны полифилы? 24](#_Toc76483895)

[33. Расскажите о функциях - конструкторах. 24](#_Toc76483896)

[34. Как можно получить список полей и список значений объекта? 25](#_Toc76483897)

[35. Как сделать наследование класса в ES6? 25](#_Toc76483898)

[36. Микро и макро таски в JavaScript. 26](#_Toc76483899)

[37. Что такое генераторы в javascript и как они работают? 26](#_Toc76483900)

[38. Какие способы хранения данных в браузере вы знаете? 27](#_Toc76483901)

[39. В чем отличие sessionStorage от localStorage? 28](#_Toc76483902)

[41. Как сервер может запретить чтение cookie из браузера? 28](#_Toc76483903)

[42. Что представляют из себя регулярные выражения и для чего они нужны? 28](#_Toc76483904)

[43. WeakSet и WeakMap отличие от Map и Set. 28](#_Toc76483905)

[44. Почему два объекта с одинаковыми полями при сравнении дают false? 29](#_Toc76483906)

[45. Почему у примитивов (number,string …) мы можем вызывать методы? Например ‘some string’.toLowerCase()? 29](#_Toc76483907)

[46. Как проверить из какого класса был создан объект? 30](#_Toc76483908)

[47. Напишите код, который будет выводить в логи каждые 5 секунд время пребывания на сайте в секундах. 30](#_Toc76483909)

[48. Как остановить интервал? 30](#_Toc76483910)

[49. Что представляет из себя чистая функция? 31](#_Toc76483911)

[50. Что представляет из себя функция высшего порядка? 31](#_Toc76483912)

[51. Зачем нужны промисы, если можно работать с асинхронным кодом с помощью функций обратного вызова (callbacks)? 31](#_Toc76483913)

[52. Что будет, если добавить скрипт перед тегом body? 32](#_Toc76483914)

[53. Как проверить, является ли объект массивом? 33](#_Toc76483915)

[54. Что такое рекурсия? Как предотвратить бесконечную рекурсию 33](#_Toc76483916)

[**CSS** 34](#_Toc76483917)

[1. Свойство display. Какие значения принимает и как они работают? 34](#_Toc76483918)

[2. Keyframes для чего нужно и как использовать? 34](#_Toc76483919)

[3. Какие бывают значения у свойства position? Расскажите, как ведёт себя каждое свойство. 35](#_Toc76483920)

[4. Как сделать кастомный чекбокс? 36](#_Toc76483921)

[5. Как отцентровать блок по горизонтали и вертикали? 36](#_Toc76483922)

[6. Расскажите о свойстве transition. 36](#_Toc76483923)

[7. Что делает box-sizing:border-box? 36](#_Toc76483924)

[8. Как обрабатывает WEB страницу браузер? 36](#_Toc76483925)

[9. Что такое inline стили? 37](#_Toc76483926)

[10. Чем border отличается от outline? 37](#_Toc76483927)

[11. Что вы знаете о резиновой верстке? 37](#_Toc76483928)

[12 - Что такое БЭМ? 37](#_Toc76483929)

[13. Что такое теги и атрибуты? 38](#_Toc76483930)

[14 - Как при нажатии на ссылку открывать страницу в новом окне? 38](#_Toc76483931)

[15. В чем различие между строчными и блочными элементами? 38](#_Toc76483932)

[16. Разница между margin и padding 39](#_Toc76483933)

[17. Что такое семантические элементы и для чего они нужны? 39](#_Toc76483934)

[18. Свойство overflow, что делает и когда использовать? 39](#_Toc76483935)

[19. vh vw при указании размеров 39](#_Toc76483936)

[20. Как изменить внешний вид курсора? 40](#_Toc76483937)

[21. Как изменить направление оси Flexbox контейнера? 40](#_Toc76483938)

[22. Из чего строится размер элемента? 40](#_Toc76483939)

[23. Как спозиционировать один элемент относительно другого? 40](#_Toc76483940)

[24. За что отвечает z-index 41](#_Toc76483941)

[25. Какие виды инпутов бывают? 41](#_Toc76483942)

[26. За что отвечает justify-content и align-items у flexbox контейнера 41](#_Toc76483943)

[27. Как убрать маркер у списка? 41](#_Toc76483944)

[28. Что вы знаете о приоритете селекторов? Специфичность. 41](#_Toc76483945)

[29. Какие css свойства для браузера самые тяжелые? 42](#_Toc76483946)

[30. Что такое псевдоклассы и какие вы используете чаще всего? 42](#_Toc76483947)

[31. Какие бывают значения у background-size? Кратко о каждом 42](#_Toc76483948)

[32. Как повернуть блок на 45 градусов?? 42](#_Toc76483949)

[33. Какие псевдоэлементы вы знаете и используете? 42](#_Toc76483950)

[34. Flexbox, какое свойство отвечает за перенос дочерних элементов на новую строку при переполнении 42](#_Toc76483951)

[35. !important для чего используется? 43](#_Toc76483952)

[36. Разница между <script> <script async> <scrypt defer> 43](#_Toc76483953)

[37. Как увеличить в размере при наведении элемент, не сдвигая соседние? 43](#_Toc76483954)

[38. Кратко о медиа запросах в CSS. 43](#_Toc76483955)

[39. Какое свойство используется для перевода текста в заглавные или строчные буквы? 44](#_Toc76483956)

[40. Для чего рекомендуется использовать атрибуты data? 44](#_Toc76483957)

[41. Как сделать анимацию бесконечно повторяющейся?? 44](#_Toc76483958)

[42. Что такое селектор атрибутов? 44](#_Toc76483959)

[43. Как изменить стили для кнопок с атрибутом disabled? 45](#_Toc76483960)

[44. Как изменить стили для элемента span, который следует прямо за элементом input? 45](#_Toc76483961)

[45. С помощью какого селектора можно добавить стиль на каждый элемент на странице? 45](#_Toc76483962)

[46. Какое свойство позволяет вам спрятать элемент, но сохранить занимаемое им пространство на странице? 46](#_Toc76483963)

[47. Как сделать тень, падающую от блока? 46](#_Toc76483964)

[48. Как сделать тень, падающую от текста? 46](#_Toc76483965)

[49. Для чего используются css препроцессоры? 46](#_Toc76483966)

[50. Что такое миксины в препроцессорах? 46](#_Toc76483967)

[51. Что такое bootstrap? 47](#_Toc76483968)

[**React** 48](#_Toc76483969)

[1. Что делает setState() 48](#_Toc76483970)

[2. Что такое VirtualDom 48](#_Toc76483971)

[3. Как отрисовать массив элементов 48](#_Toc76483972)

[4. Разница между контролируемыми и неконтролируемыми компонентами 48](#_Toc76483973)

[5. Методы жизненного цикла компонента 49](#_Toc76483974)

[6. Какие React хуки вы знаете и используете 50](#_Toc76483975)

[7. useState особенности использования 50](#_Toc76483976)

[8. useEffect особенности использования 50](#_Toc76483977)

[9. Как отследить демонтирование функционального компонента? 51](#_Toc76483978)

[10. Что такое State менеджер и какой вы используете? 51](#_Toc76483979)

[11. В каких случаях можно использовать локальное состояние, а в каких лучше использовать глобальный State? 52](#_Toc76483980)

[12. Redux. Что такое редьюсер и какие параметры он принимает? 52](#_Toc76483981)

[13. Redux. Что такое экшн и как изменить состояние? 52](#_Toc76483982)

[14. Что такое JSX? 52](#_Toc76483983)

[15. Что такое PROPS? 53](#_Toc76483984)

[16. Отличие в записях 53](#_Toc76483985)

[17. useMemo для чего нужен и когда использовать? 53](#_Toc76483986)

[18. useCallback для чего нужен и когда использовать? 54](#_Toc76483987)

[19. useContext для чего нужен и когда использовать? 54](#_Toc76483988)

[20. useRef для чего нужен и когда использовать? 54](#_Toc76483989)

[21. React.memo для чего нужен и когда использовать? 55](#_Toc76483990)

[22. Расскажите о React fiber? 55](#_Toc76483991)

[23. Что такое React fragment? 56](#_Toc76483992)

[24. Расскажите о React Reconciliation 56](#_Toc76483993)

[25. Для чего нужны ключи key в списках? 57](#_Toc76483994)

[26. Асинхронные actions в redux с помощью thunk. 57](#_Toc76483995)

[27. Как отрисовать блок по условию? 57](#_Toc76483996)

[28. Как отследить изменение поля объекта в функциональном компоненте? 58](#_Toc76483997)

[29. Как получить доступ к дом элементу в React. 58](#_Toc76483998)

[**Vue** 59](#_Toc76483999)

[1. Что такое двустороннее связывание? 59](#_Toc76484000)

[2. Какими способами можно реализовать двустороннее связывание для input? 59](#_Toc76484001)

[3. Что такое props? 59](#_Toc76484002)

[4. Как работает реактивность в Vue? 60](#_Toc76484003)

[5. Что такое Composition API в Vue 3? 61](#_Toc76484004)

[6. Особенности использования v-model в Vue 2 и Vue 3. 61](#_Toc76484005)

[7. Для чего нужен $emit? 61](#_Toc76484006)

[8. Постраничная навигация в Vue. 62](#_Toc76484007)

[9. Что такое computed свойства и как они работают? 62](#_Toc76484008)

[10. Как отследить изменение модели? 63](#_Toc76484009)

[11. Особенности слежения за «глубокими» объектами. 63](#_Toc76484010)

[12. Что такое интерполяция? 63](#_Toc76484011)

[13. Как отрисовать компонент по условию? 64](#_Toc76484012)

[14. Разница между v-if и v-show. 64](#_Toc76484013)

[15. Что такое миксины и как их использовать? 65](#_Toc76484014)

[16. Что такое директивы? 65](#_Toc76484015)

[17. Как создать пользовательскую директиву? Особенности использования. 66](#_Toc76484016)

[18. Расскажите о жизненном цикле компонента Vue. 66](#_Toc76484017)

[19. В каком методе жизненного цикла необходимо делать первичную загрузку данных с сервера? 66](#_Toc76484018)

[20. В каком методе жизненного цикла необходимо делать очистку (удалять слушатели, очищать хранилище и т.д.)? 67](#_Toc76484019)

[21. Как сделать стили локальными для компонента? 67](#_Toc76484020)

[22. Как отрисовать несколько компонентов на основе массива? 67](#_Toc76484021)

[23. Зачем указывать key при использовании директивы v-for? 67](#_Toc76484022)

[24. Как отследить изменение поля объекта? 68](#_Toc76484023)

[25. Как добавить слушатель события на элемент? 68](#_Toc76484024)

[26. Что такое модификаторы? 68](#_Toc76484025)

[27. Какие модификаторы есть у событий? 69](#_Toc76484026)

[28. Какие модификаторы есть у v-model? 69](#_Toc76484027)

[29. Как добавить анимацию на удаление\добавление элемента в список? 69](#_Toc76484028)

[30. Как зарегистрировать компонент глобально в Vue 3? 70](#_Toc76484029)

[31. Как передать данные из родительского компонента в дочерний, не используя props и store. 70](#_Toc76484030)

[32. Как добавить класс на элемент по условию? 70](#_Toc76484031)

[33. Как динамически изменять стили у элемента? 71](#_Toc76484032)

[34. Расскажите о слотах в Vue. 71](#_Toc76484033)

[35. Каким способом можно получить DOM элемент во Vue? 72](#_Toc76484034)

[36. Предназначение Keep-alive в Vue 72](#_Toc76484035)

[37. Телепорты в Vue. Зачем нужны? 73](#_Toc76484036)

[38. Как подключить внешний плагин в Vue 3? 74](#_Toc76484037)

[39. Как создать собственный плагин в Vue 3? 74](#_Toc76484038)

[40. При использовании хуков жизненного цикла в миксине и при подключении этого миксина в компонент, в какой последовательности будут вызываться хуки? 74](#_Toc76484039)

[41. Почему не стоит использовать в качестве ключей (key) индексы элемента массива? 75](#_Toc76484040)

[42. Почему этот код не работает? array.filter(elem => elem % 2 !== 0) 75](#_Toc76484041)

[43. Можно ли использовать v-if и v-for на одном элементе? Объясните свой ответ. 75](#_Toc76484042)

[44. Можно ли изменять computed свойства? 75](#_Toc76484043)

[45. Для чего нужен vuex и какие проблемы он решает? 76](#_Toc76484044)

[46. Расскажите о state и getters в vuex. 76](#_Toc76484045)

[47. Расскажите о мутациях и действиях в vuex. В чем отличие? 76](#_Toc76484046)

[48. Как использовать store внутри компонента? 77](#_Toc76484047)

[49. Как принудительно обновить компонент в Vue? 77](#_Toc76484048)

[50. Для чего нужны асинхронные компоненты в Vue? 77](#_Toc76484049)

[**Общие вопросы** 79](#_Toc76484050)

[1.Что такое HTTP 79](#_Toc76484051)

[2. Из чего состоит HTTP запрос 79](#_Toc76484052)

[3. Какие методы http запросов вы знаете 79](#_Toc76484053)

[4. В чем семантическое отличие PUT и PATCH 79](#_Toc76484054)

[5. Что такое websockets? 80](#_Toc76484055)

[6. Что такое REST API 80](#_Toc76484056)

[7. Что такое WebRTC? 80](#_Toc76484057)

[8. Что такое Git? 80](#_Toc76484058)

[9. Как сделать коммит в Git? 80](#_Toc76484059)

[10. Как создать новую ветку и перейти на нее в Git? 80](#_Toc76484060)

[12. Merge и rebase отличия. 80](#_Toc76484061)

[13. На каких 3 принципах базируется ООП? Расскажите кратко о каждом. 80](#_Toc76484062)

[14. Какие паттерны проектирования вы чаще всего используете? 80](#_Toc76484063)

[15. Для чего нужен package.json? 80](#_Toc76484064)

# **JAVASCRIPT**

## 1. Какие типы данных существуют в javascript?

**Подробнее -** [**Типы данных**](https://learn.javascript.ru/types)

* Числа – Number.
* Строки – String.
* Логический тип – Boolean.
* Object – обычный JavaScript объект.
* null - это просто специальное значение, которое представляет собой «ничего», «пусто» или «значение неизвестно».
* undefined - означает, что «значение не было присвоено». Если переменная объявлена, но ей не присвоено никакого значения, то её значением будет undefined.
* Symbol - это уникальный и неизменяемый тип данных, который может быть использован как идентификатор для свойств объектов.
* BigInt - используется для создания больших чисел.

const bigInt = 1234567890123456789012345678901234567890n;

## 2. В чем разница между операторами "==" и "==="?

**Подробнее -** [**Операторы сравнения**](https://learn.javascript.ru/comparison)

Оператор '==' проверяет на абстрактное равенство, т.е. он **выполняет** необходимые преобразования типов перед выполнением сравнения на равенство.

Оператор '===' проверяет на строгое равенство, т.е. он **не** будет выполнять преобразование типа, поэтому, если два значения не одного типа, при сравнении он вернет false.

## 3. Какими способами можно объявить переменную?

**Подробнее -** [**Переменные**](https://learn.javascript.ru/variables)

Объявить переменную можно 4 способами

1. a = 5;
2. var a = 5;
3. let a = 5;
4. const a = 5;

2ой способ с помощью ключевого слова var и он аналогичен первому. Переменные обьявленные таким способом имеют глобальную или функциональную область видимости и не имеют блочную, что является большим минусом. Такой способ объявления является устаревшим.

Let и const являются предпочтительным способом обьявить переменную, имеют блочную область видимости, т.е объявленная внутри, например, блока if переменная не будет видна снаружи, const является неизменяемым, но если это объект, то можно менять поля, если это массив то можно менять\добавлять элементы в массив.

## 4. В чем разница между null и undefined?

**Подробнее -** [**Null и Undefined**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/null)

Оба варианта означают пустое значение. Если мы инициализируем переменную, но не присваиваем ей значение, туда помещается специальный «маркер», который отображается при выводе на экран как undefined. Null присваиваем самостоятельно.

Null - Это просто специальное значение, которое представляет собой «ничего», «пусто» или «значение неизвестно». Если необходимо очистить значение переменной, мы делаем q = null.

undefined означает, что «значение не было присвоено».

**5. map, filter, reduce, foreach для чего нужны, особенности использования**

**Подробнее -** [**Методы массивов**](https://learn.javascript.ru/array-methods)

Каждый из этих методов так или иначе итерируется по массиву, но каждый имеет свое предназначение:

* Метод forEach ничего не возвращает, его используют только для перебора, как более «элегантный» вариант, чем обычный цикл for.
* Метод «arr.filter(callback[, thisArg])» используется для фильтрации массива через функцию. Он создаёт новый массив, в который войдут только те элементы arr, для которых вызов callback(item, i, arr) возвратит true.
* Метод «arr.map(callback[, thisArg])» используется для трансформации массива. Он создаёт новый массив, который будет состоять из результатов вызова callback(item, i, arr)для каждого элемента arr.
* Метод «arr.reduce(callback[, initialValue])» используется для последовательной обработки каждого элемента массива с сохранением промежуточного результата.

## 6. Стрелочные функции и их отличие от функций, объявленных через function

**Подробнее -** [**Стрелочные функции**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Functions/Arrow_functions)

* Стрелочные функции не имеют argumemts.
* Синтаксис
* У стрелочных функций нет своего this. Если идет обращение к this, то он берется снаружи.
* Не могут быть функциями – конструкторами. Т.е не могут вызываться с помощью new.

## 7. Что такое замыкание?

**Подробнее -** [**Замыкания**](https://learn.javascript.ru/closure)

Замыкание – это функция вместе со всеми внешними переменными, которые ей доступны. Грубо говоря, на примере, есть функция, которая имеет вложенную функцию, и вложенная функция будет замыкать и сохранять в себе переменные из родительской.

|  |
| --- |
| function parent() {  const a = 5;  return function child() {  console.log(5); // child замыкает в себе переменную a;  }  } |

## 8. Что такое шаблонные литералы и для чего они нужны?

**Подробнее -** [**Шаблонные литералы**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Template_literals)

Шаблонные литералы - косые кавычки, в которых возможен перенос строки и в них также можно встраивать выражения.

let b = 5;

const a = `Сумма равна ${b} `

## 9. Что такое set и map?

**Подробнее -** [**Set и Map**](https://learn.javascript.ru/map-set)

Map – это коллекция, структура данных, работающая по принципу ключ/значение, как и Object. Но основное отличие от объекта в том, что Map позволяет использовать ключи любого типа.

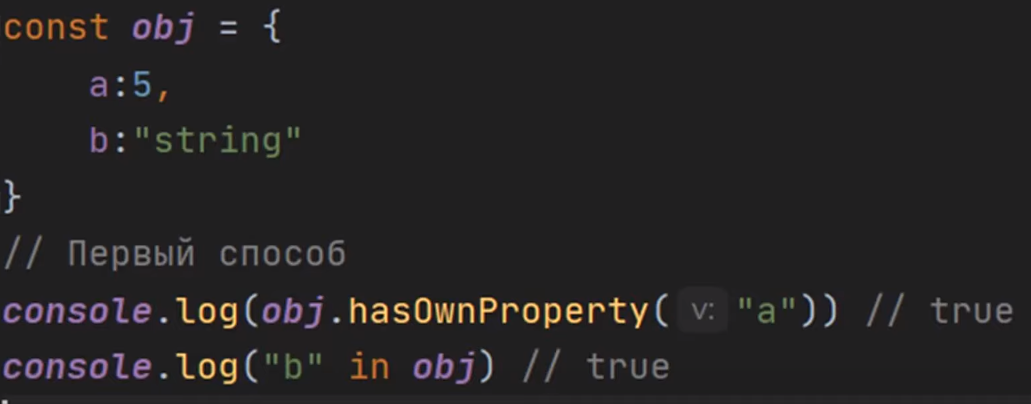
Объект Set – это особый вид коллекции: «множество» значений (без ключей), своего рода массив, где каждое значение может появляться только один раз.

## 10. Как определить наличие свойства в объекте?

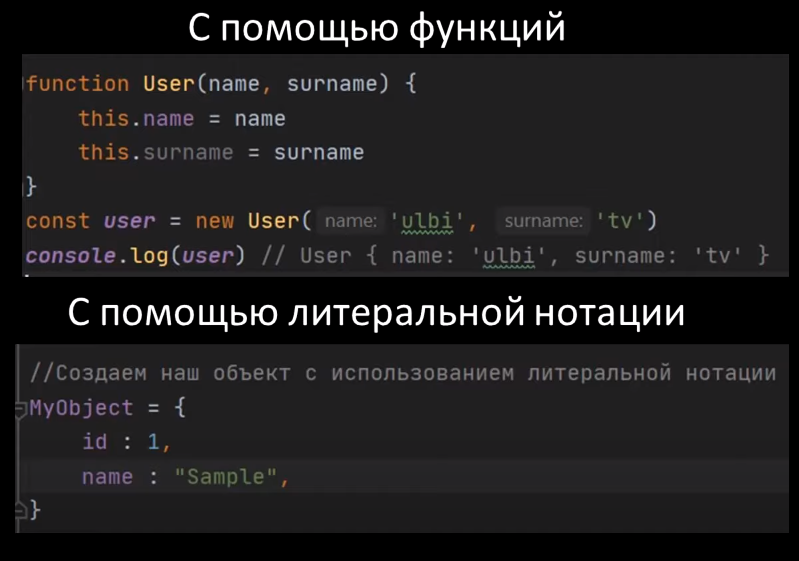
**Подробнее -** [**hasOwnProperty**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Object/hasOwnProperty)

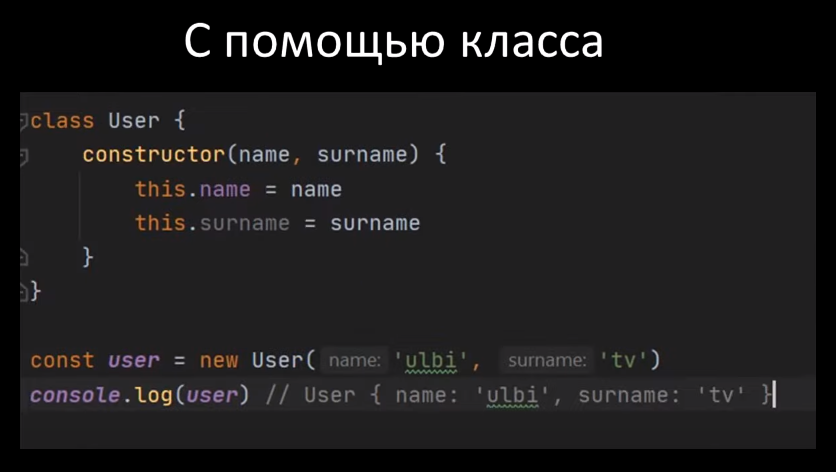
Первый способ - с помощью функции hasOwnProperty, который есть у каждого обьекта.

Второй способ с помощью оператора in. С использованием этого оператора надо быть осторожным, так проверяются по цепочке все прототипы.

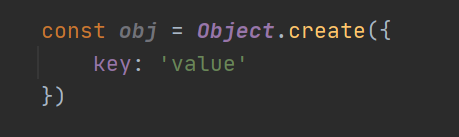


## 11. Какие способы создания объекта вы знаете?





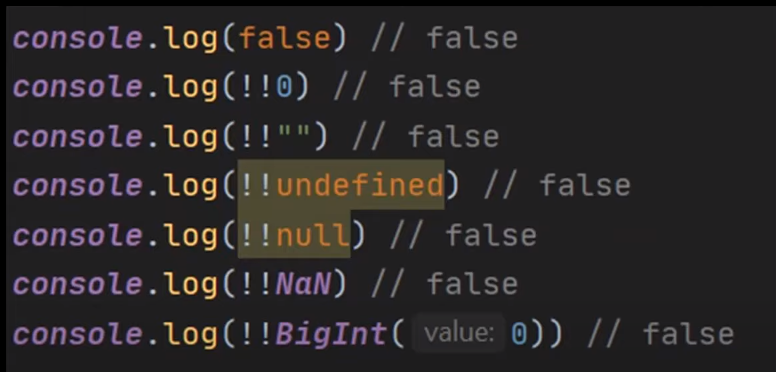
C помощью функции create.



## 12. Какие значения являются falsy(ложными) значениями?

**Подробнее -** [**falsy значения**](https://webformyself.com/kak-na-samom-dele-v-javascript-rabotayut-operatory-i/)

Falsy значение – значение, которое при приведении к логическому типу возвращает false.

****

## 13. Что такое Promise?

**Подробнее -** [**Promise**](https://learn.javascript.ru/promise-basics)

Promise – это специальный объект, предназначенный для работы с ассинхронным кодом и который содержит своё состояние.

Вначале pending («ожидание»), затем – одно из: fulfilled («выполнено успешно») или rejected («выполнено с ошибкой»).

На promise можно навешивать колбэки двух типов:

* onFulfilled – срабатывают, когда promise в состоянии «выполнен успешно».
* onRejected – срабатывают, когда promise в состоянии «выполнен с ошибкой».

Способ использования, в общих чертах, такой:

1. Код, которому надо сделать что-то асинхронно, создаёт объект promise и возвращает его.
2. Внешний код, получив promise, навешивает на него обработчики.
3. По завершении процесса асинхронный код переводит promise в состояние fulfilled (с результатом) или rejected (с ошибкой). При этом автоматически вызываются соответствующие обработчики во внешнем коде.

## 14. Как использовать async/await для асинхронных запросов?

**Подробнее -** [**async/await**](https://learn.javascript.ru/async-await)

Существует специальный синтаксис для работы с промисами «async/await».

Функция, обозначенная как async всегда вернет Promise.

Ключевое слово await заставит интерпретатор JavaScript ждать до тех пор, пока промис справа от await не выполнится. После чего оно вернёт его результат, и выполнение кода продолжится. await нельзя использовать в обычных функциях.

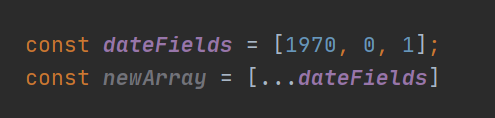
## 15. Для чего нужен оператор spread?

**Подробнее -** [**Spread**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Spread_syntax)

Оператор spread, он же три точки – предназначен для разворачивания массивов или объктов.

Spread syntax позволяет расширить доступные для итерации элементы (например, массивы или строки) в местах

* для функций: где ожидаемое количество аргументов для вызовов функций равно нулю или больше нуля
* для элементов (литералов массива)
* для выражений объектов: в местах, где количество пар "ключ-значение" должно быть равно нулю или больше (для объектных литералов)



## 16. Как избежать ссылочной зависимости при копировании объекта? Глубокое копирование объекта

**Подробнее -** [**Глубокое копирование**](https://www.youtube.com/watch?v=iDTZKdfJEcw&ab_channel=UlbiTV)

Если объект не содержит внутренних объектов, например:

|  |
| --- |
| const obj = {  key: ‘value’,  field: 1  }  В таком случае можно воспользоваться стандартными приемами:  const copy = {…obj}  или  const copy = Object.assign({}, obj} |

Если объект СОДЕРЖИТ внутренние объекты:

|  |
| --- |
| const obj = {  key: {  field: 1  }  } |

В таком случае необходимо делать глубокое копирование:

1. Воспользоваться костыльным, медленным способом:

const copy = JSON.parse(JSON.stringify(obj))

Такой способ подойдет для объекта без прототипа и без функций.

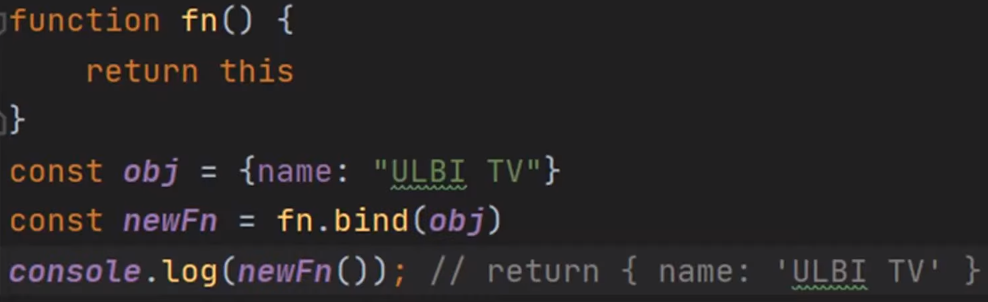
2. реализовать рекурсивную функцию копирования полей.

3. Воспользоваться библиотекой lodash, функцией deep clone

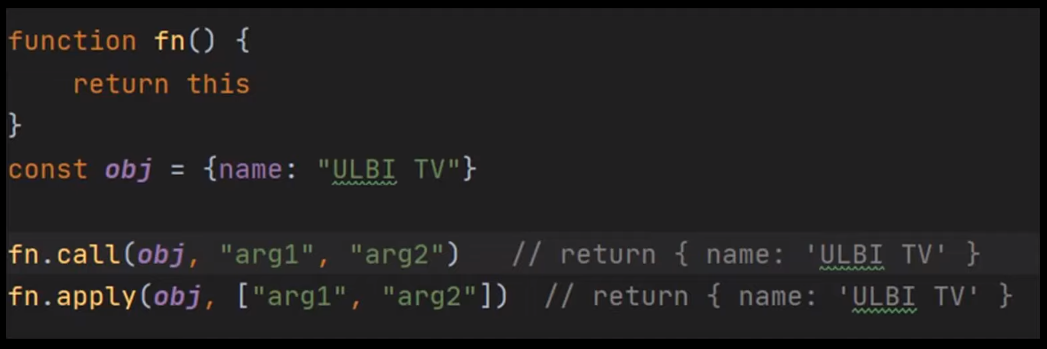
## 18. Как поменять контекст функции?

**Подробнее -** [**call, bind, apply**](https://learn.javascript.ru/call-apply-decorators)

1. C помощь функции bind, которая возвращает новую функцию с привязанным контекстом.



2. с помощью функций call() и apply(). Их основное отличие в том, что call принимает последовательность аргументов, а apply вторым параметром массив аргументов.



## 19- Что такое тернарный оператор?

**Подробнее -** [**тернарный оператор**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Conditional_Operator)

Это аналогичная запись if else. Оператор представлен знаком вопроса ?. Его также называют «тернарный», так как этот оператор, единственный в своём роде, имеет три аргумента.

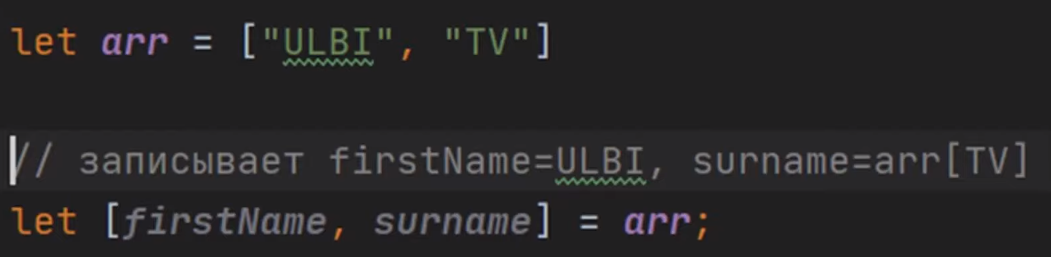
условие ? выражение1 : выражение2

## 20. Что такое деструктуризация

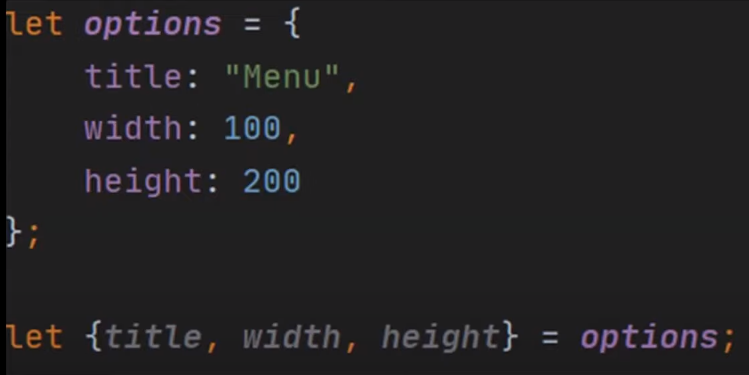
**Подробнее -** [**деструктуризация**](https://learn.javascript.ru/destructuring-assignment)

В JavaScript есть две чаще всего используемые структуры данных – это Object и Array. Деструктурирующее присваивание – это специальный синтаксис, который позволяет нам «распаковать» массивы или объекты в кучу переменных, так как иногда они более удобны.

Пример деструктуризации массива



Пример деструктуризации обьекта

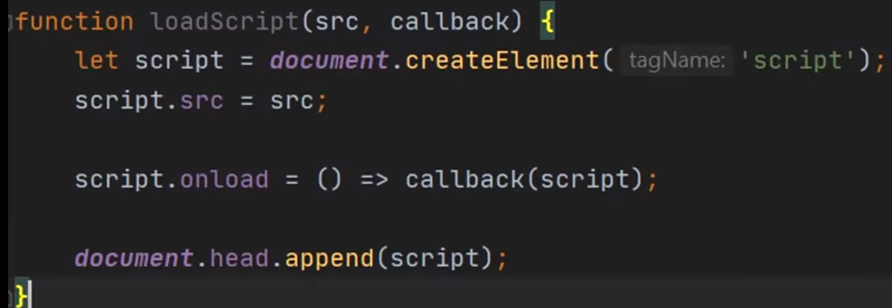


## 21. Какие способы работы с асинхронным кодом вы знаете?

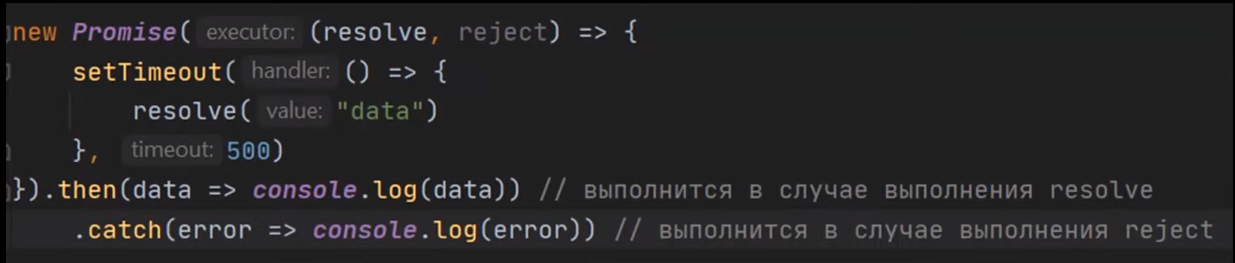
**Подробнее -** [**Асинхронный JavaScript**](https://developer.mozilla.org/ru/docs/Learn/JavaScript/Asynchronous/Introducing)

Существуют 3 способа обработки асинхронных событий:

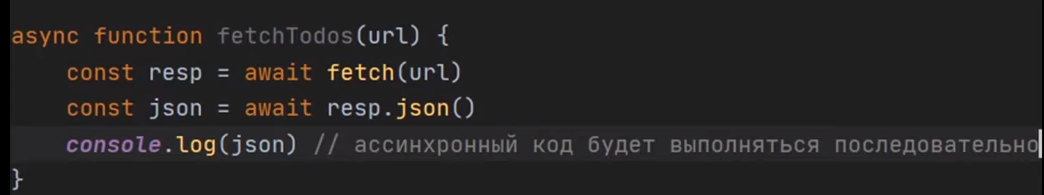
Функции обратного вызова (callback)



Промисы



async/await



## 22. e.preventDefault() и e.stopPropagination() для чего нужны?

**Подробнее -** [**действия браузера по умолчанию**](https://learn.javascript.ru/default-browser-action)

**Подробнее -** [**Всплытие и погружение**](https://learn.javascript.ru/bubbling-and-capturing%20)

Многие события автоматически влекут за собой действие браузера. Например:

* Клик по ссылке инициирует переход на новый URL.
* Нажатие на кнопку «отправить» в форме – отсылку её на сервер.

Чтобы предотвратить дефолтное поведение браузера используется функция preventDefault.

Когда на элементе происходит событие, обработчики сначала срабатывают на нём, потом на его родителе, затем выше и так далее, вверх по цепочке предков.

Всплытие идёт с «целевого» элемента прямо наверх. По умолчанию событие будет всплывать до элемента <html>, а затем до объекта document, а иногда даже до window, вызывая все обработчики на своём пути.

Но любой промежуточный обработчик может решить, что событие полностью обработано, и остановить всплытие.

Для этого нужно вызвать метод event.stopPropagation()

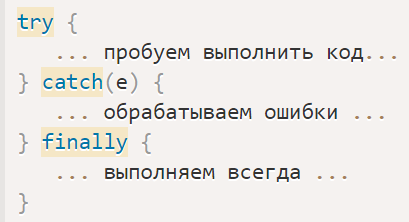
## 23. Как отслеживать и обрабатывать ошибки в javascript?

**Подробнее -** [**Обработка ошибок**](https://learn.javascript.ru/try-catch)

В js есть синтаксическая конструкция try..catch, которая позволяет «ловить» ошибки и вместо падения делать что-то более осмысленное. Сначала выполняется код внутри блока try {...}. Если в нём нет ошибок, то блок catch(err) игнорируется: выполнение доходит до конца try и потом далее, полностью пропуская catch.

Если же в нём возникает ошибка, то выполнение try прерывается, и поток управления переходит в начало catch(err). Переменная err (можно использовать любое имя) содержит объект ошибки с подробной информацией о произошедшем.

Блок finaly вызовется независимо от того произошла ошибка или нет.



## 24. Что такое DOM дерево?

**Подробнее -** [**DOM**](https://learn.javascript.ru/dom-nodes)

Основой HTML-документа являются теги.

В соответствии с объектной моделью документа («Document Object Model», коротко DOM), Каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом. Все эти объекты доступны при помощи JavaScript, мы можем использовать их для изменения страницы.

DOM – это представление HTML-документа в виде дерева тегов. Вот как оно выглядит:



Каждый узел этого дерева – это объект.

## 25. Что такое event loop?

**Подробнее** - [**Наглядный пример**](https://www.youtube.com/watch?v=377qAu37OTE&ab_channel=UlbiTV)

**Подробнее -** [**Event loop**](https://learn.javascript.ru/event-loop)

## 26. Что такое прототипное наследование?

**Подробнее -** [**Прототипное наследование**](https://learn.javascript.ru/prototype-inheritance)

Изначально каждый объект в javascript обладает свойством — прототипом. Вы можете добавлять в него методы и свойства. На основе прототипа можно создавать другие объекты. Создаваемый объект автоматически унаследует свойства и своего прототипа. Если свойство в новом объекте отсутствует, то будет произведен его поиск в прототипе.

## 27. Как получить свойство объекта?

**Подробнее -** [**Объекты**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Working_with_Objects)

Первый способ через точку obj.a – статичный.

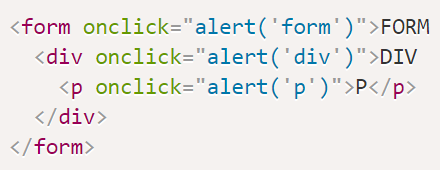
Второй способ через квадратные скобки obj["a"] – динамический.

## 28. Всплытие и погружение событий

**Подробнее -** [**Всплытие и погружение**](https://learn.javascript.ru/bubbling-and-capturing)

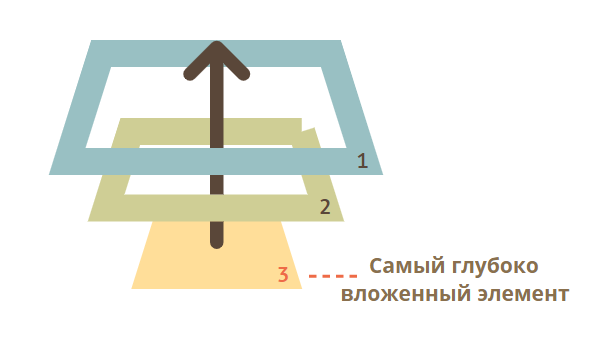
Всплытие.

Принцип всплытия очень простой. Когда на элементе происходит событие, обработчики сначала срабатывают на нём, потом на его родителе, затем выше и так далее, вверх по цепочке предков.



В данном примере алерты появятся в следующей последовательности

p -> div -> form



Именно по такому принципу работает всплытие.

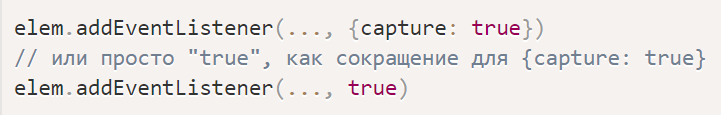
Погружение

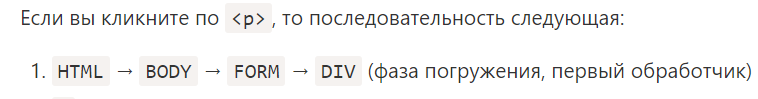
Существует ещё одна фаза из жизненного цикла события – «погружение» (иногда её называют «перехват»). Она очень редко используется в реальном коде, однако тоже может быть полезной.

Стандарт DOM Events описывает 3 фазы прохода события:

1. Фаза погружения (capturing phase) – событие сначала идёт сверху вниз.
2. Фаза цели (target phase) – событие достигло целевого(исходного) элемента.
3. Фаза всплытия (bubbling stage) – событие начинает всплывать.

Чтобы поймать событие на стадии погружения, нужно использовать третий аргумент capture вот так:





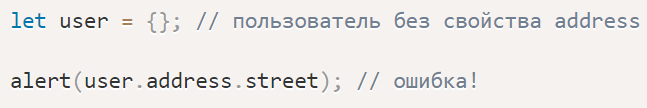
## 29. Optional chaining оператор. Для чего нужен?

**Подробнее -** [**Опциональная цепочка**](https://learn.javascript.ru/optional-chaining)

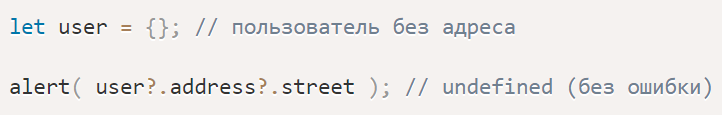
**Видео пример -** [**видео**](https://youtu.be/fcS4azdSR_0)

Например, рассмотрим объекты для пользователей user. У большинства пользователей есть адрес user.address с улицей user.address.street, но некоторые адрес не указали.

В этом случае при попытке получить свойство user.address.street будет ошибка:



Опциональная цепочка ?. останавливает вычисление и возвращает undefined, если часть перед ?. имеет значение undefined или null.



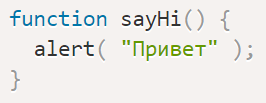
## 30. Shadow DOM.

**Подробнее -** [**Shadow DOM**](https://learn.javascript.ru/shadow-dom)

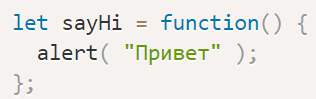
Здесь лучше почитать статью, она небольшая.

## 31. function expression и function declaration. Что это такое и в чем разница?

Function declaration – привычный способ объявить функцию. Например:



Существует ещё один синтаксис создания функций, который называется Function Expression (Функциональное Выражение).



В коде выше функция создаётся и явно присваивается переменной, как любое другое значение. По сути без разницы, как мы определили функцию, это просто значение, хранимое в переменной sayHi.

* Function Declaration обрабатываются перед выполнением блока кода. Они видны во всём блоке.
* Функции, объявленные при помощи Function Expression, создаются, только когда поток выполнения достигает их.

## 32. Для чего нужны полифилы?

**Подробнее –** [**Полифилы**](https://learn.javascript.ru/polyfills)

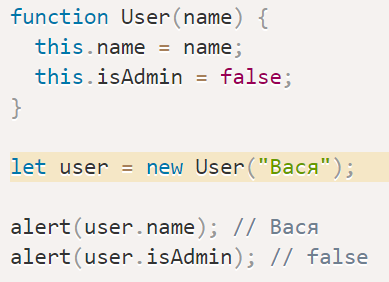
«Полифил» (англ. polyfill) – это библиотека, которая добавляет в старые браузеры поддержку возможностей, которые в современных браузерах являются встроенными.

## 33. Расскажите о функциях - конструкторах.

**Подробнее -** [**Функция конструктор**](https://learn.javascript.ru/constructor-new)

Функции-конструкторы являются обычными функциями. Но есть два соглашения:

1. Имя функции-конструктора должно начинаться с большой буквы.
2. Функция-конструктор должна вызываться при помощи оператора "new".



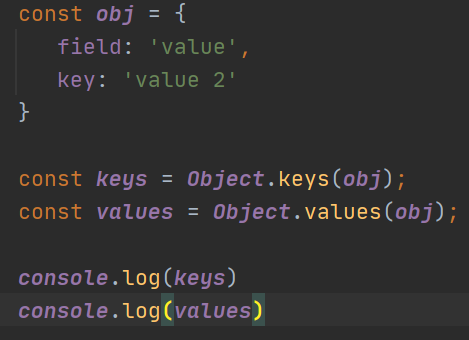
Когда функция вызывается как new User(...), происходит следующее:

1. Создаётся новый пустой объект, и он присваивается this.
2. Выполняется код функции. Обычно он модифицирует this, добавляет туда новые свойства.
3. Возвращается значение this.

## 34. Как можно получить список полей и список значений объекта?

Получить поля/ключи можно с помощью Object.keys.

Получить значения можно с помощью Object.values

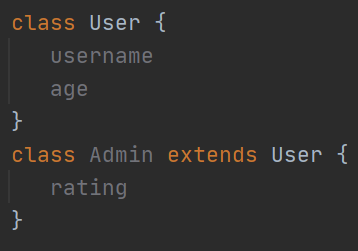


## 35. Как сделать наследование класса в ES6?

**Подробнее -** [**Классы**](https://learn.javascript.ru/class)

**Подробнее -** [**Наследование**](https://learn.javascript.ru/class-inheritance)

Расширение\наследование класса происходит с помощью ключевого слова extends, после которого идет название родительского класса. Например:



## 36. Микро и макро таски в JavaScript.

**Подробнее -** [**Микро и Макро задачи**](https://learn.javascript.ru/event-loop)

Важно понимать, что такое event loop см. 25 вопрос.

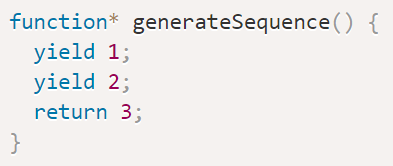
## 37. Что такое генераторы в javascript и как они работают?

**Подробнее –** [**Генераторы**](https://learn.javascript.ru/generators)

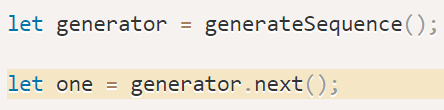
Обычные функции возвращают только одно-единственное значение (или ничего).

Генераторы могут порождать (yield) множество значений одно за другим, по мере необходимости. Генераторы отлично работают с перебираемыми объектами и позволяют легко создавать потоки данных.

Для объявления генератора используется специальная синтаксическая конструкция: function\*, которая называется «функция-генератор».



Основным методом генератора является next(). При вызове он запускает выполнение кода до ближайшей инструкции yield <значение> (значение может отсутствовать, в этом случае оно предполагается равным undefined). По достижении yield выполнение функции приостанавливается, а соответствующее значение – возвращается во внешний код



## 38. Какие способы хранения данных в браузере вы знаете?

**Подробнее –** [**LocalStorage, sessionStorage**](https://learn.javascript.ru/localstorage)

**Подробнее -** [**куки**](https://learn.javascript.ru/cookie)

**Подробнее -** [**IndexedDB**](https://learn.javascript.ru/indexeddb)

Существует несколько подходов к хранению данных в браузере:

1. LocalStorage, SessionStorage - позволяют хранить пары ключ/значение в браузере. Что в них важно – данные, которые в них записаны, сохраняются после обновления страницы. Объекты хранилища localStorage и sessionStorage предоставляют одинаковые методы и свойства:

* setItem(key, value) – сохранить пару ключ/значение.
* getItem(key) – получить данные по ключу key.
* removeItem(key) – удалить данные с ключом key.
* clear() – удалить всё.
* key(index) – получить ключ на заданной позиции.
* length – количество элементов в хранилище.

Оба хранилища в качестве ключей и значений могут использовать только строки, поэтому объекты не забываем преобразовывать с помощью JSON.stringify.

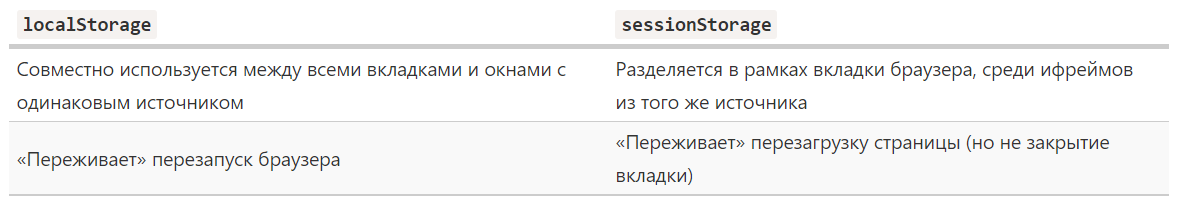
2. Куки – это небольшие строки данных, которые хранятся непосредственно в браузере. Куки обычно устанавливаются веб-сервером при помощи заголовка Set-Cookie. Затем браузер будет автоматически добавлять их в (почти) каждый запрос на тот же домен при помощи заголовка Cookie. одно куки вмещает до 4kb данных, разрешается более 20 куки на сайт (зависит от браузера).

3. IndexedDB – это встроенная база данных, более мощная, чем localStorage.

* Хранилище ключей/значений: доступны несколько типов ключей, а значения могут быть (почти) любыми.
* Поддерживает транзакции для надёжности.
* Поддерживает запросы в диапазоне ключей и индексы.
* Позволяет хранить больше данных, чем localStorage.

Для традиционных клиент-серверных приложений эта мощность обычно чрезмерна. IndexedDB предназначена для оффлайн приложений, можно совмещать с ServiceWorkers и другими технологиями.

## 39. В чем отличие sessionStorage от localStorage?



## 41. Как сервер может запретить чтение cookie из браузера?

**Подробнее –** [**HttpOnly**](https://learn.javascript.ru/cookie%23httponly)

С помощью флага HttpOnly сервер может запретить любой доступ к куки из JavaScript. Мы не можем видеть такое куки или манипулировать им с помощью document.cookie.

Эта настройка используется в качестве меры предосторожности от определённых атак, когда хакер внедряет свой собственный JavaScript-код в страницу и ждёт, когда пользователь посетит её. Это вообще не должно быть возможным, хакер не должен быть в состоянии внедрить свой код на ваш сайт, но могут быть ошибки, которые позволят хакеру сделать это.

## 42. Что представляют из себя регулярные выражения и для чего они нужны?

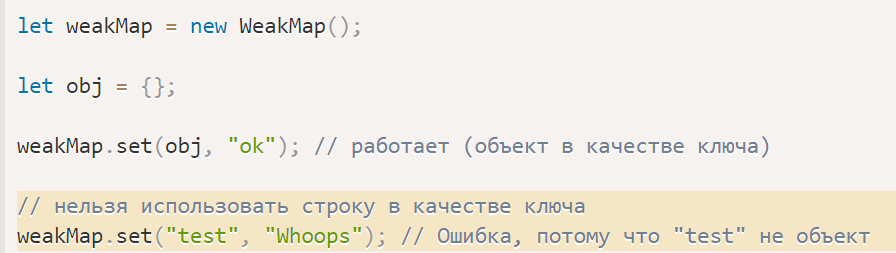
**Подробнее –** [**Регулярные выражения**](https://learn.javascript.ru/regular-expressions)

Регулярные выражения – строки, заданные по особым правилам, шаблонам. Это мощный инструмент, который позволяет находить\заменять сложные конструкции в строках.

## 43. WeakSet и WeakMap отличие от Map и Set.

**Подробнее –** [**WeakMap и WeakSet**](https://learn.javascript.ru/weakmap-weakset)

Первое его отличие от Map в том, что ключи в WeakMap должны быть объектами, а не примитивными значениями:



Движок JavaScript хранит значения в памяти до тех пор, пока они достижимы (то есть, эти значения могут быть использованы). Обычно свойства объекта, элементы массива или другой структуры данных считаются достижимыми и сохраняются в памяти до тех пор, пока эта структура данных содержится в памяти.

Например, если мы поместим объект в массив, то до тех пор, пока массив существует, объект также будет существовать в памяти, несмотря на то, что других ссылок на него нет.

В случае с WeakMap и WeakSet это работает иначе. Как только объект становится недостижим, он также удаляется из структуры данных.

## 44. Почему два объекта с одинаковыми полями при сравнении дают false?

**Подробнее –** [**Сравнение по ссылке**](https://learn.javascript.ru/object-copy%23sravnenie-po-ssylke)

const a = {key: 5}

const b = {key: 5}

a == b // true или false?

Если вы ответили true, то вы ошиблись. Объекты сравниваются по ссылкам на область в памяти. С точки зрения JavaScript объекты a и b разные, хотя и имеют одинаковые поля. Объекты равны только в том случае, если это один и тот же объект.

## 45. Почему у примитивов (number,string …) мы можем вызывать методы? Например ‘some string’.toLowerCase()?

**Подробнее –** [**методы примитивов**](https://learn.javascript.ru/primitives-methods)

JavaScript позволяет нам работать с примитивными типами данных – строками, числами и т.д., как будто они являются объектами. У них есть и методы.

Для того, чтобы нам был доступен такой функционал каждый примитив имеет свой собственный «объект-обёртку», который называется: String, Number, Boolean и Symbol. Таким образом, они имеют разный набор методов.

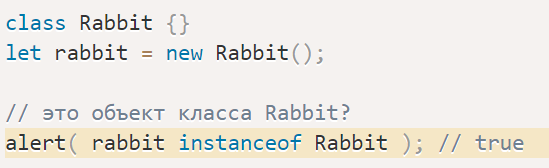
К примеру, существует метод str.toUpperCase(), который возвращает строку в верхнем регистре.

## 46. Как проверить из какого класса был создан объект?

**Подробнее –** [**Оператор instanceof**](https://learn.javascript.ru/instanceof)

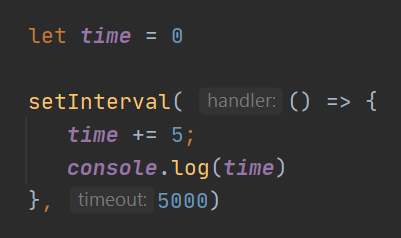
Оператор instanceof позволяет проверить, к какому классу принадлежит объект, с учётом наследования.

Такая проверка может потребоваться во многих случаях.



## 47. Напишите код, который будет выводить в логи каждые 5 секунд время пребывания на сайте в секундах.

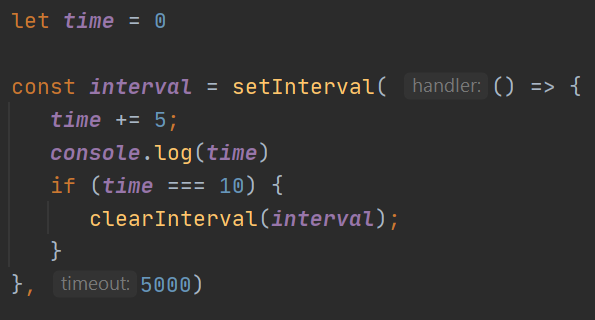
**Подробнее –** [**Интервалы**](https://learn.javascript.ru/settimeout-setinterval%23setinterval)



## 48. Как остановить интервал?

**Подробнее –** [**остановить интервал**](https://puzzleweb.ru/javascript/window_clearinterval.php)

Для остановки интервала предназначена функция clearInterval. Пример кода представлен ниже.



## 49. Что представляет из себя чистая функция?

**Подробнее –** [**Чистая функция**](https://habr.com/ru/post/437512/)

Функция должна удовлетворять двум условиям, чтобы считаться «чистой»:

— Каждый раз функция возвращает одинаковый результат, когда она вызывается с тем же набором аргументов

— Нет побочных эффектов (например, не изменяет внешние переменные)

## 50. Что представляет из себя функция высшего порядка?

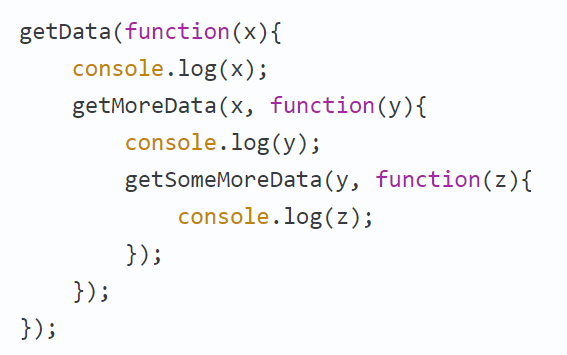
**Подробнее –** [**Функции высшего порядка**](https://habr.com/ru/post/261723/)

Функция высшего порядка — это функция, которая может принимать другую функцию в качестве аргумента или возвращать другую функцию в качестве результата.

## 51. Зачем нужны промисы, если можно работать с асинхронным кодом с помощью функций обратного вызова (callbacks)?

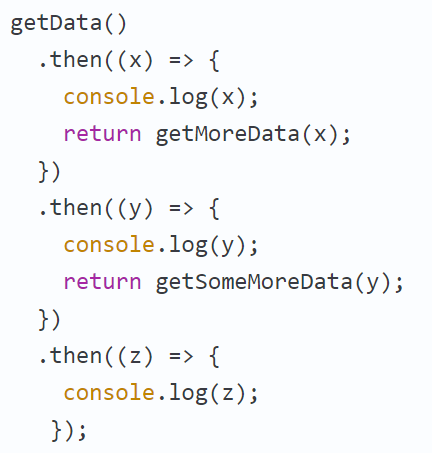
**Подробнее –**

Предположим, что мы хотим асинхронно получить некоторые данные с сервера, используя обратные вызовы мы сделали бы что-то вроде этого



Такой подход называется callback hell (адом обратных вызовов), поскольку каждый обратный вызов вложен внутрь другого, и каждый внутренний обратный вызов зависит от его родителя.

Мы можем переписать приведенный выше фрагмент используя промисы:



Используя промисы, мы четко видим последовательность выполнения, такой код становится намного читабельнее.

## 52. Что будет, если добавить скрипт перед тегом body?

**Подробнее –** [**Подключение скриптов**](https://learn.javascript.ru/script-async-defer)

В современных сайтах скрипты обычно «тяжелее», чем HTML: они весят больше, дольше обрабатываются.

Когда браузер загружает HTML и доходит до тега <script>...</script>, он не может продолжать строить DOM. Он должен сначала выполнить скрипт. То же самое происходит и с внешними скриптами <script src="..."></script>: браузер должен подождать, пока загрузится скрипт, выполнить его, и только затем обработать остальную страницу.

Это ведёт к двум важным проблемам:

* Скрипты не видят DOM-элементы ниже себя, поэтому к ним нельзя добавить обработчики и т.д.
* Если вверху страницы объёмный скрипт, он «блокирует» страницу. Пользователи не видят содержимое страницы, пока он не загрузится и не запустится:

## 53. Как проверить, является ли объект массивом?

Для этого предназначен специальный метод, возвращающий true или false Array.isArray().

## 54. Что такое рекурсия? Как предотвратить бесконечную рекурсию

**Подробнее –** [**Рекурсия**](https://learn.javascript.ru/recursion)

Рекурсия — это такой способ организации обработки данных, при котором программа вызывает сама себя непосредственно, либо с помощью других программ.

Рекурсивная функция состоит из:

* Условие остановки или же Базовый случай
* Условие продолжения или Шаг рекурсии — способ сведения задачи к более простым.

Базовый случай является обязательным условием, иначе произойдет переполнения стека вызовов из за бесконечного вызова функции.

# **CSS**

Видео формат - [ссылка](https://www.youtube.com/watch?v=4YRG6cMAASI)

## 1. Свойство display. Какие значения принимает и как они работают?

**Подробнее -** [**display css**](https://developer.mozilla.org/ru/docs/Web/CSS/display)

CSS свойство display определяет, как должен отображаться определенный элемент HTML.

1. display: none - Самое простое значение. Элемент не показывается, вообще.

2. display: block - Блочные элементы располагаются один над другим, вертикально. Блок стремится расшириться на всю доступную ширину.

3. display: inline - Элементы располагаются на той же строке, последовательно. Ширина и высота элемента определяются по содержимому Менять их вручную нельзя.

4. display: inline-block - элемент является строчным при этом можно менять ширину высоту.

5. display: flex – <https://youtu.be/eVZEwEQg4pg>

6. display: grid – <https://youtu.be/MEOR2b69Pl4>

Есть и другие, но они используются крайне редко.

## 2. Keyframes для чего нужно и как использовать?

**Подробнее –** [**анимации**](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Animations/Using_CSS_animations)

Keyframes позволяет описать ключевые кадры анимации. CSS анимации позволяют анимировать переходы от одной конфигурации CSS стилей к другой. CSS-анимации состоят из двух компонентов: стилевое описание анимации и набор ключевых кадров, определяющих начальное, конечное и, возможно, промежуточное состояние анимируемых стилей.



## 3. Какие бывают значения у свойства position? Расскажите, как ведёт себя каждое свойство.

**Подробнее -** [**position css**](https://developer.mozilla.org/ru/docs/Web/CSS/position)

Свойство position позволяет сдвигать элемент со своего обычного места

static - Статическое позиционирование производится по умолчанию. Элемент считается не позиционированным.

Relative - относительное позиционирование сдвигает элемент относительно его обычного положения. Для того, чтобы применить относительное позиционирование, необходимо указать элементу CSS-свойство position: relative и координаты left/right/top/bottom.

Absolute – при абсолютном позиционировании элемент исчезает с того места, где он должен быть и позиционируется заново. Остальные элементы, располагаются так, как будто этого элемента никогда не было. Координаты top/bottom/left/right для нового местоположения отсчитываются от ближайшего позиционированного родителя, т.е. родителя с позиционированием, отличным от static. Если такого родителя нет – то относительно документа.

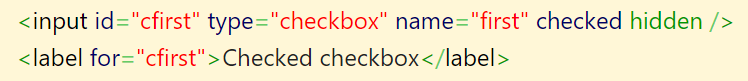
Fixed – фиксированное позиционирование замораживает блок на месте и когда страницу прокручивают, фиксированный элемент остаётся на своём месте и не прокручивается вместе со страницей.

sticky - похож на fixed, но крепится в рамках какого-то блока, а не всего документа.

## 4. Как сделать кастомный чекбокс?

**Подробнее -** [**кастомный checkbox**](https://habr.com/ru/post/144104/)

Перед чекбоксом создается label и привязывается к инпуту. После чего инпут скрывается, а label стилизуется так, как необходимо



## 5. Как отцентровать блок по горизонтали и вертикали?

**Подробнее** - [блок по центру](https://learn.javascript.ru/css-center)

Существует несколько способов, но самый простой из них - сделать родительский элемент display: flex; и задать два свойства:

align-items: center;

justify-content: center;

## 6. Расскажите о свойстве transition.

**Подробнее -** [**transition**](https://developer.mozilla.org/ru/docs/Web/CSS/transition)

Transition позволяет определять переходное состояние между двумя состояниями элемента. Различные состояния могут быть определены с помощью псевдоклассов, таких как :hover или :active или установлены динамически с помощью JavaScript.

## 7. Что делает box-sizing:border-box?

**Подробнее -** [**box-sizing**](https://developer.mozilla.org/ru/docs/Web/CSS/box-sizing)

CSS свойство box-sizing определяет как вычисляется общая ширина и высота элемента. При значении border-box cвойства width и height включают контент, внутренний отступ и границы, но не включают внешний отступ.

## 8. Как обрабатывает WEB страницу браузер?

**Подробнее –** [**обработка страницы браузером**](https://habr.com/ru/post/224187/)

1. Из полученного от сервера HTML-документа формируется DOM (Document Object Model).
2. Загружаются и распознаются стили, формируется CSSOM (CSS Object Model).
3. На основе DOM и CSSOM формируется дерево рендеринга, или render tree — набор объектов рендеринга (Webkit использует термин «renderer», или «render object», а Gecko — «frame»). Render tree дублирует структуру DOM, но сюда не попадают невидимые элементы (например — <head>, или элементы со стилем display:none;). Также, каждая строка текста представлена в дереве рендеринга как отдельный renderer. Каждый объект рендеринга содержит соответствующий ему объект DOM (или блок текста), и рассчитанный для этого объекта стиль. Проще говоря, render tree описывает визуальное представление DOM.
4. Для каждого элемента render tree рассчитывается положение на странице — происходит layout. Браузеры используют поточный метод (flow), при котором в большинстве случаев достаточно одного прохода для размещения всех элементов (для таблиц проходов требуется больше).
5. Наконец, происходит отрисовка всего этого добра в браузере — painting.

## 9. Что такое inline стили?

Это стили, которые пишутся прям в html и они имеют самый высокий приоритет (не считая !important)

## 10. Чем border отличается от outline?

**Подробнее -** [**outline**](https://learn.javascript.ru/outline)

1. Outline не влияет на положение элемента и его размеры.
2. Outline не позволяет задать рамку с определённой стороны элемента (только сразу со всех).
3. На outline рамку не действует скругление углов, устанавливаемое с помощью свойства border-radius.

## 11. Что вы знаете о резиновой верстке?

Резиновая вёрстка — это когда вы задаёте всему макету и отдельным его частям не фиксированную ширину, а эластичную — в процентах. За исключением минимальной и максимальной ширины.

## 12 - Что такое БЭМ?

**Подробнее -** [**БЭМ**](https://ru.bem.info/methodology/)

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste».

## 13. Что такое теги и атрибуты?

**Подробнее –** [**теги**](https://html5book.ru/html-tags/)

**Подробнее -** [**атрибуты**](https://html5book.ru/html-attributes/)

HTML-теги — основа языка HTML. Теги используются для разграничения начала и конца элементов в разметке.

Каждый HTML-документ состоит из дерева HTML-элементов и текста. Каждый HTML-элемент обозначается начальным (открывающим) и конечным (закрывающим) тегом. Открывающий и закрывающий теги содержат имя тега.

HTML-атрибуты сообщают браузеру, каким образом должен отображаться тот или иной элемент страницы. Атрибуты позволяют сделать более разнообразными внешний вид информации, добавляемой с помощью одинаковых тегов.

Значение атрибута заключается в кавычки "". Названия и значения атрибутов не чувствительны к регистру, но, тем не менее, рекомендуется набирать их в нижнем регистре.

## 14 - Как при нажатии на ссылку открывать страницу в новом окне?

**Подробнее** - [target](https://www.internet-technologies.ru/articles/target-blank-i-drugie-znacheniya-atributa-target.html)

атрибутом target="\_blank"

## 15. В чем различие между строчными и блочными элементами?

**Подробнее** – [блочные и строчные](https://ru.w3docs.com/uchebnik-html/html-elementy.html)

Элементы в HTML также делятся на блочные и строчные. **Блочными** называют элементы, которые являются строительными блоками веб-страниц. Их используют для разделения содержимого веб-страницы на логические блоки, такие как меню, шапка сайта, блок с контентом, “подвал” сайта и т.д. Блочные элементы пишутся с новой строки; до и после этих элементов в браузере автоматически добавляется разрыв строки.

Блочными являются элементы <address>, <article>, <aside>, <blockquote>, <canvas>, <dd>, <div>, <dl>, <dt>, <fieldset>, <figcaption>, <figure>, <footer>, <form>, <h1>-<h6>, <header>, <hr>, <li>, <main>, <nav>, <noscript>, <ol>, <output>, <p>, <pre>, <section>, <table>, <tfoot>, <ul> и <video>.

У всех блочных элементов есть открывающие и закрывающие теги.

**Строчными** элементами размечают части содержимого элементов. Они занимают только ограниченное тегами пространство, браузер не добавляет автоматически разрыв строки.

К строчным относятся элементы <a>, <abbr>, <acronym>, <b>, <bdo>, <big>, <br>, <buttom>, <cite>, <code>, <dfn>, <em>, <i>, <img>, <input>, <kbd>, <label>, <map>, <object>, <q>, <samp>, <script>, <select>, <small>, <span>, <strong>, <sub>, <sup>, <textarea>, <time>, <tt> и <var>.

## 16. Разница между margin и padding

**Подробнее** - [margin](http://htmlbook.ru/css/margin)

**Подробнее -** [**padding**](http://htmlbook.ru/css/padding)

Margin – внешний отступы.

Padding – внутренние отступы.

## 17. Что такое семантические элементы и для чего они нужны?

**Подробнее** - [**Семантические теги**](https://medium.com/@stasonmars/%D1%81%D0%B5%D0%BA%D1%80%D0%B5%D1%82%D1%8B-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D1%81%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B8%CC%86-%D0%B2%D0%B5%D1%80%D1%81%D1%82%D0%BA%D0%B8-%D0%B2-html5-c7cd5e6f1ebb)

С html5 были введены так называемые семантические теги, которые определяют конкретные части страницы. Правильное использование их улучшает СЕО оптимизацию, что позволяет поисковым роботам лучше индексировать страницу. Добавляя семантические HTML теги на ваши страницы, вы даете дополнительную информацию, которая помогает поисковикам понимать роли и относительную важность разных частей ваших страниц.

## 18. Свойство overflow, что делает и когда использовать?

**Подробнее** – [overflow](https://developer.mozilla.org/ru/docs/Web/CSS/overflow)

Свойство CSS overflow определяет, необходимо ли для переполненного блочного элемента содержимое обрезать, предоставить полосы прокрутки или просто отобразить.

## 19. vh vw при указании размеров

**Подробнее** -

Если мы указывам размеры в %, то это процент от размеров родителя.

vh vw являются относительными единицами и составляют:

vh - 1% от высоты окна браузера

vw - 1% от ширины окна браузера

## 20. Как изменить внешний вид курсора?

**Подробнее** – [cursor](http://htmlbook.ru/css/cursor)

Для изменения внешнего вида курсора предназначено css свойство cursor.

## 21. Как изменить направление оси Flexbox контейнера?

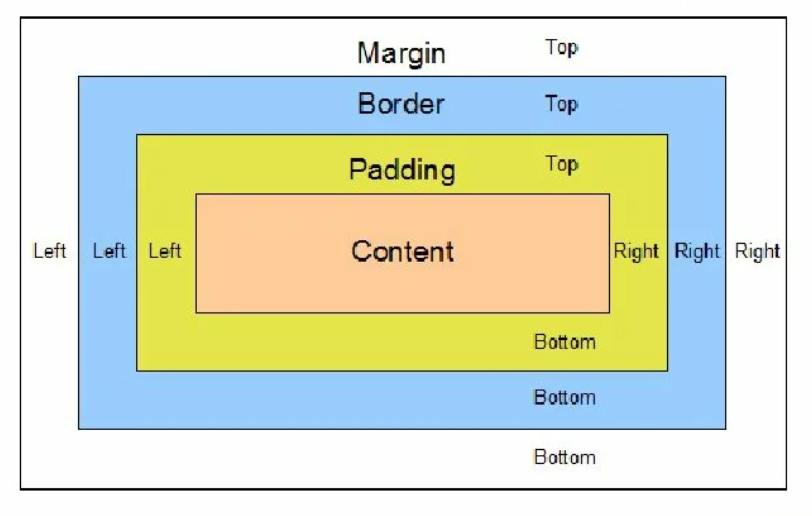
**Подробнее** – [flex-direction](https://developer.mozilla.org/ru/docs/Web/CSS/flex-direction)

Для изменения направления оси flexbox контейнера предназначено свойство flex-direction.

## 22. Из чего строится размер элемента?

**Подробнее** - [размеры блоков](https://brainoteka.com/blogs/css-spravochnik/zadanie-razmerov-blokov-css)

Размер элемента строится из ширины и высоты содержимого, внутренних отступов, рамки, и внешних отступов



## 23. Как спозиционировать один элемент относительно другого?

**Подробнее** - [position-relative](https://learn.javascript.ru/position%23position-relative)

Для дочернего элемента задать свойство position: relative и с помощью top/right/bottom/left управлять расположением.

## 24. За что отвечает z-index

**Подробнее** - [**z-index**](http://htmlbook.ru/css/z-index)

Свойство z-index в CSS управляет вертикальным порядком расположения элементов, которые перекрываются. z-index влияет только на элементы, которые имеют значение позиции, которое не является статическим.

Без какого-либо значения z-index элементы складываются в том порядке, в котором они появляются в DOM (самый нижний элемент на том же уровне иерархии отображается сверху). Элементы с нестатическим позиционированием (и их дочерние элементы) всегда будут отображаться поверх элементов со статическим позиционированием по умолчанию, независимо от иерархии HTML

## 25. Какие виды инпутов бывают?

**Таблица с видами инпутов здесь -** [**виды инпутов**](https://meliorem.ru/frontend/html/spisok-tipov-polej-input/)

## 26. За что отвечает justify-content и align-items у flexbox контейнера

**Подробнее** – [justify and align](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Flexible_Box_Layout/Aligning_Items_in_a_Flex_Container)

Оба свойства отвечают за центрирование элементов во flex контейнере. Для центрирования элемента по перекрёстной оси используется свойство align-items. Для центрирования элемента по главной, используется свойство justify-content.

## 27. Как убрать маркер у списка?

**Подробнее** – [маркеры у списка](https://webref.ru/recipe/2287)

Чтобы скрыть отображение маркеров в списке применяется стилевое свойство list-style-type со значением none. Его следует добавить к селектору ul или li.

## 28. Что вы знаете о приоритете селекторов? Специфичность.

**Подробнее** - [Специфичность](https://developer.mozilla.org/ru/docs/Web/CSS/Specificity)

Специфичность селекторов (selector's specificity) определяет их приоритет в таблице стилей. Чем специфичнее селектор, тем выше его приоритет.

Каждый селектор имеет вес.

Элемент – 1

Класс – 10

id – 100

inline style – 1000

!important – имеет наибольший вес.

## 29. Какие css свойства для браузера самые тяжелые?

Большое количество подключенных шрифтов, тени, анимации, прозрачность.

## 30. Что такое псевдоклассы и какие вы используете чаще всего?

**Подробнее** - [Псевдоклассы](https://developer.mozilla.org/ru/docs/Web/CSS/Pseudo-classes)

Псевдоклассы описывают характеристики элементов, такие как динамическое состояние — нажатая ссылка, язык кодировки — абзац на французском языке и т.д. Они не отображаются в исходном документе и не принадлежат дереву документа DOM. Самые часто используемые hover, focus, checked, disabled и другие

## 31. Какие бывают значения у background-size? Кратко о каждом

**Подробнее** - [background-size](https://developer.mozilla.org/ru/docs/Web/CSS/background-size)

**cover**

Масштабирует изображение с сохранением пропорций так, чтобы его ширина или высота равнялась ширине или высоте блока.

**contain**

Масштабирует изображение с сохранением пропорций таким образом, чтобы картинка целиком поместилась внутрь блока.

## 32. Как повернуть блок на 45 градусов??

transform: rotateX(45deg);

## 33. Какие псевдоэлементы вы знаете и используете?

**Подробнее** - [Псевдоэлементы](https://developer.mozilla.org/ru/docs/Web/CSS/Pseudo-elements)

Псевдоэлемент в CSS — это ключевое слово, добавляемое к селектору, которое позволяет стилизовать определённую часть выбранного элемента. Например, псевдоэлемент ::first-line может быть использован для изменения шрифта первой строки абзаца.

Самые часто используемые: after, before, placeholder,first-letter

## 34. Flexbox, какое свойство отвечает за перенос дочерних элементов на новую строку при переполнении

**Подробнее** - [flex wrap](https://developer.mozilla.org/ru/docs/Web/CSS/flex-wrap)

Свойство flex-wrap: wrap; позволяет переносить flex элементы на новую строку при переполнении.

## 35. !important для чего используется?

**Подробнее** - [important](http://htmlbook.ru/css/%21important)

Позволяет повысить приоритет селектора и переопределить какие-то стили.

## 36. Разница между <script> <script async> <scrypt defer>

**Подробнее** - [script-async-defer](https://learn.javascript.ru/script-async-defer)

Браузер загружает и отображает HTML постепенно. Особенно это заметно при медленном интернет-соединении: браузер не ждёт, пока страница загрузится целиком, а показывает ту часть, которую успел загрузить.

Если браузер видит тег <script>, то он по стандарту обязан сначала выполнить его, а потом показать оставшуюся часть страницы. Т.е. загрузка получается синхронной сверху вниз.

Если скрипт – внешний, то пока браузер не выполнит его, он не покажет часть страницы под ним.

При указании атрибута async и defer браузер начинает подгружать скрипты в асинхронном режиме и не тормозит отображение html.

Но defer используют в том случае, когда важен порядок загрузки скриптов, скрипты будут загружаться синхроно. но последовательно. И второе отличие, defer сработает только тогда, когда весь html будет обработан браузером, тоесть async начинает подгружать сразу, а defer после загрузки html.

## 37. Как увеличить в размере при наведении элемент, не сдвигая соседние?

**Подробнее** - [transform](https://developer.mozilla.org/ru/docs/Web/CSS/transform)

для этого необходимо воспользоваться свойством transform.

CSS3-трансформации позволяют сдвигать, поворачивать и масштабировать элементы. Трансформации преобразовывают элемент, не затрагивая остальные элементы веб-страницы, т.е. другие элементы не сдвигаются относительно него.

## 38. Кратко о медиа запросах в CSS.

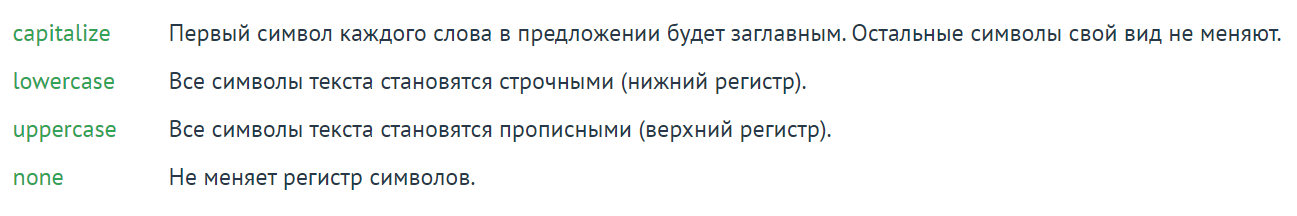
**Подробнее** – [media\_queries](https://developer.mozilla.org/ru/docs/Web/CSS/Media_Queries/Using_media_queries)

Медиавыражения используются в тех случаях , когда нужно применить разные CSS-стили, для разных устройств по типу отображения (например: для принтера, монитора или смартфона), а также конкретных характеристик устройства (например: ширины окна просмотра браузера), или внешней среды (например: внешнее освещение). Учитывая огромное количество подключаемых к интернету устройств, медиавыражения являются очень важным инструментом при создании веб-сайтов и приложений, которые будут правильно работать на всех доступных устройствах, которые есть у ваших пользователей.

## 39. Какое свойство используется для перевода текста в заглавные или строчные буквы?

**Подробнее** - [text-transform](https://webref.ru/css/text-transform)

text-transform



## 40. Для чего рекомендуется использовать атрибуты data?

**Подробнее** - [data\_attributes](https://developer.mozilla.org/ru/docs/Learn/HTML/Howto/Use_data_attributes)

Они предназначены для:

* Хранения значений.
* Создания всплывающих подсказок без применения скриптов.
* Определения стиля элемента на основе значения атрибута.
* Получения и изменения значений через скрипты.

## 41. Как сделать анимацию бесконечно повторяющейся??

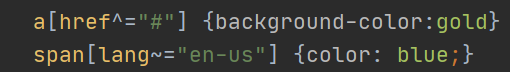
**Подробнее** - [animation-iteration-count](https://webref.ru/css/animation-iteration-count)

animation-iteration-count: infinite

## 42. Что такое селектор атрибутов?

**Подробнее** – [Attribute\_selectors](https://developer.mozilla.org/ru/docs/Web/CSS/Attribute_selectors)

Селекторы атрибутов отбирают элементы по наличию атрибута или его значению.



## 43. Как изменить стили для кнопок с атрибутом disabled?

**Подробнее** - [Attribute\_selectors](https://developer.mozilla.org/ru/docs/Web/CSS/Attribute_selectors)

Чтобы отобрать элемент, у которого установлен атрибут disabled, нужно использовать селекторы атрибутов.

button[disabled] {

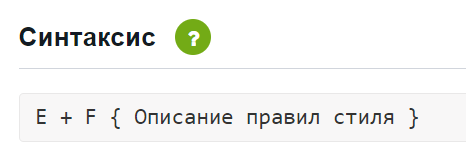
opacity: 0.24;

}

## 44. Как изменить стили для элемента span, который следует прямо за элементом input?

**Подробнее** – [соседний селектор](https://webref.ru/css/selector/adjacent)

Для этого предназначен соседний селектор. Соседними называются элементы веб-страницы, когда они следуют непосредственно друг за другом в коде документа.



## 45. С помощью какого селектора можно добавить стиль на каждый элемент на странице?

**Подробнее** – [Универсальный селектор](http://htmlbook.ru/samcss/universalnyy-selektor)

Иногда требуется установить одновременно один стиль для всех элементов веб-страницы, например, задать шрифт или начертание текста. В этом случае поможет универсальный селектор, который соответствует любому элементу веб-страницы.

Для обозначения универсального селектора применяется символ звёздочки (\*) и в общем случае синтаксис будет следующий.



## 46. Какое свойство позволяет вам спрятать элемент, но сохранить занимаемое им пространство на странице?

Скрыть элемент, но сохранить занимаемое им пространство можно двумя способами:

1. visibility: hidden;

2. opacity: 0;

## 47. Как сделать тень, падающую от блока?

**Подробнее** – [box-shadow](https://developer.mozilla.org/ru/docs/Web/CSS/box-shadow)

**Потренироваться можно** [**здесь**](https://css3gen.com/box-shadow/)

Для создания тени, падающей от блока, используется свойство box-shadow.

## 48. Как сделать тень, падающую от текста?

**Подробнее** – [текст и тень](http://htmlbook.ru/blog/tekst-i-ten)

Для создания тени, падающей от текста, используется свойство text-shadow.

## 49. Для чего используются css препроцессоры?

**Подробнее** – [препроцессоры](https://habr.com/ru/post/434098/)

Препроцессор — это программа, которой на вход дается код написанный на языке препроцессора, а на выходе мы получаем CSS, который мы можем дать на вход нашему браузеру. Препроцессоры упрощают процесс верстки, если уметь их использовать.

Существует несколько представителей, например: Sass(.sass, .scss), Less(.less) и Stylys(.stylus). Чаще всего препроцессоры добавляют новые возможности в css, а именно:

* Вложенность.
* Миксины.
* Дополнительные функции.
* Модульность.
* Переменные и тд.

## 50. Что такое миксины в препроцессорах?

**Подробнее** – [миксины](https://sass-scss.ru/guide/)

Миксин — это функция, которая принимает аргументы и применяет правила, зависящие от этих аргументов, к данному селектору. Миксины позволяют создавать группы деклараций CSS, которые вам придется использовать по нескольку раз на сайте. Вы даже можете передавать переменные в миксины, чтобы сделать их более гибкими.

## 51. Что такое bootstrap?

**Подробнее** – [bootstrap](https://getbootstrap.com/)

Bootstrap – библиотека, представляющая большое количество заготовленных стилей, компонентов, по типу кнопок, модальных окон, инпутов и тд.

# **React**

## 1. Что делает setState()

**Подробнее -** [**setState**](https://ru.reactjs.org/docs/faq-state.html)

Состояние хранит в себе какие-то данные, и чтобы изменить эти данные необходимо вызвать функцию setState().

Метод setState() следит за изменением состояния (state) компонента. state — это объект. Когда состояние меняется, компонент рендерится повторно и мы видим в браузере компонент с обновленными данными.

## 2. Что такое VirtualDom

**Подробнее -** [**Virtual DOM**](https://ru.reactjs.org/docs/faq-internals.html)

VirtualDOM это копия DOM дерева и вместо того, чтобы взаимодействовать с DOM напрямую, мы работаем с его легковесной копией. Мы можем вносить изменения в копию, исходя из наших потребностей, а после этого React применяет изменения к реальному DOM.

При этом происходит сравнение DOM-дерева с его виртуальной копией, определяется разница и запускается перерисовка того, что было изменено.

Такой подход работает быстрее, потому как не включает в себя все тяжеловесные части реального DOM.

## 3. Как отрисовать массив элементов

**Подробнее -** [**Списки и ключи**](https://ru.reactjs.org/docs/lists-and-keys.html)

Для того чтобы отрисовать список элементов используется функция массивов map, внутри которой мы пишем jsx разметку, при этом React требует указывать ключ key для каждого элемента итерации. Ключи помогают React идентифицировать, какие элементы были изменены, добавлены или удалены.

Ключи должны быть заданы элементам внутри массива, чтобы предоставить элементам постоянный идентификатор

## 4. Разница между контролируемыми и неконтролируемыми компонентами

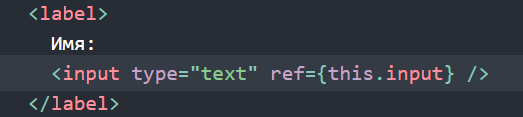
**Подробнее -** [**Управляемые компоненты**](https://ru.react.js.org/docs/forms.html)

**Подробнее -** [**Неуправляемые компоненты**](https://ru.reactjs.org/docs/uncontrolled-components.html)

В HTML элементы формы, такие как <input>, <textarea> и <select>, обычно поддерживают собственное состояние и обновляют его в соответствии с пользовательскими входными данными. В React изменяемое состояние обычно хранится в свойстве state компонентов и обновляется только с помощью setState().

Мы можем объединить всё это вместе, сделав состояние React «единственным источником данных (истины)». Затем компонент React, который отрисовывает форму, также контролирует, что происходит в этой форме при последующем вводе данных пользователем. Элемент поля ввода формы, значение которого контролируется React подобным образом, называется «**контролируемым компонентом**».

Вместо того, чтобы писать обработчик события для каждого обновления состояния, вы можете использовать **неуправляемый компонент** и читать значения из DOM через ref



## 5. Методы жизненного цикла компонента

**Подробнее -** [**Жизненный цикл**](https://ru.react.js.org/docs/react-component.html)

Существует четыре различных этапа жизненного цикла компонента React:

* Инициализация: На этом этапе компонент React готовит установку начального состояния и параметров по умолчанию.
* Монтирование: Компонент React готов для монтирования в DOM браузера. Этот этап охватывает методы жизненного цикла componentWillMount и componentDidMount.
* Обновление: На этом этапе компонент обновляется двумя способами, отправляя новые свойства и обновляя состояние. Этот этап охватывает методы жизненного цикла shouldComponentUpdate, componentWillUpdate и componentDidUpdate.
* Размонтирование: На этом последнем этапе компонент не нужен и отключается из DOM браузера. Этот этап включает метод жизненного цикла componentWillUnmount.

## 6. Какие React хуки вы знаете и используете

**Подробнее -** [**Краткий обзор хуков**](https://ru.reactjs.org/docs/hooks-overview.html)

Здесь лучше рассказать то, какие хуки вы используете на своей практике. Хук — это специальная функция, которая позволяет «подцепиться» к возможностям React. Например, хук useState предоставляет функциональным компонентам доступ к состоянию React. Основные React хуки:

useState;

useEffect;

useContext;

useRef;

useMemo;

useCallback;

## 7. useState особенности использования

**Подробнее -** [**useState**](https://ru.reactjs.org/docs/hooks-state.html)

UseState - Возвращает значение с состоянием и функцию для его обновления.

Во время первоначального рендеринга возвращаемое состояние (state) совпадает со значением, переданным в качестве первого аргумента (initialState).

Функция setState используется для обновления состояния. Она принимает новое значение состояния и ставит в очередь повторный рендер компонента. Функция setState может принимать параметром, как и новое значение, так и функцию callback, которая параметром принимает предыдущее значение.

## 8. useEffect особенности использования

**Подробнее -** [**useEffect**](https://ru.reactjs.org/docs/hooks-effect.html)

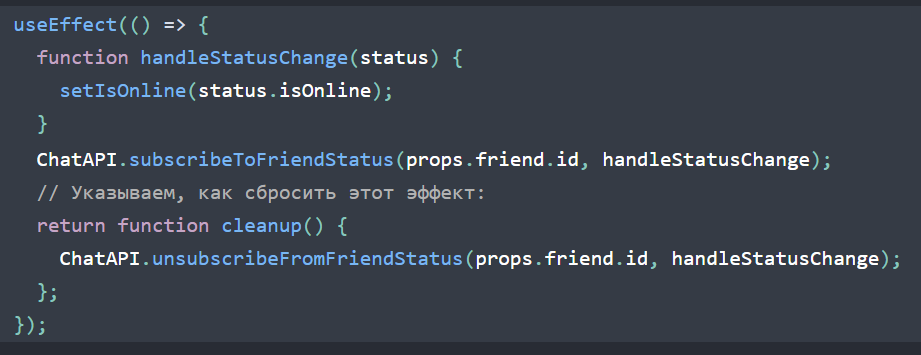
Хук эффекта даёт вам возможность выполнять побочные эффекты в функциональном компоненте. useEffect - Мутации, подписки, таймеры, логирование и другие побочные эффекты не допускаются внутри основного тела функционального компонента (называемого этапом рендеринга React). Это приведёт к запутанным ошибкам и несоответствиям в пользовательском интерфейсе.

Вместо этого рекомендуется использовать useEffect. Функция, переданная в useEffect, будет запущена после того, как рендер будет зафиксирован на экране или же если передать вторым параметром массив зависимостей, то функция будет вызываться каждый раз после изменения одной из зависимостей.

## 9. Как отследить демонтирование функционального компонента?

**Подробнее -** [**useEffect со сбросом**](https://ru.reactjs.org/docs/hooks-effect.html%23effects-with-cleanup)

Часто эффекты создают ресурсы, которые необходимо очистить (или сбросить) перед тем, как компонент покидает экран, например, подписку или идентификатор таймера. Чтобы сделать это, функция, переданная в useEffect, может вернуть функцию очистки. Функция очистки запускается до удаления компонента из пользовательского интерфейса, чтобы предотвратить утечки памяти. Кроме того, если компонент рендерится несколько раз (как обычно происходит), предыдущий эффект очищается перед выполнением следующего эффекта.



## 10. Что такое State менеджер и какой вы используете?

State менеджер решает несколько проблем, во первых это хорошая практика, отделять данные и логику по работе с ними от компонентов, во вторых, если использовать локальное состояние и передавать его из компоненты в компонент код становится запутанным, поскольку вложенность компонентов может быть большой. Имея глобальное хранилище, мы можем обращаться к нему с любого компонента и получать\изменять данные. Вместе с React используют чаще всего либо Redux, либо MobX.

## 11. В каких случаях можно использовать локальное состояние, а в каких лучше использовать глобальный State?

Локальное состояние рекомендуется использовать в тех случаях, когда оно используется только в рамках 1го компонента и не планируется передавать его в других компоненты. Также локальное состояние используется в компоненте какого-то отдельного элемента списка. Если же декомпозиция на компоненты предполагает вложенность с передачей данных по иерархии, то лучше использовать global state.

## 12. Redux. Что такое редьюсер и какие параметры он принимает?

**Подробнее -** [**reducers**](https://redux.js.org/tutorials/fundamentals/part-3-state-actions-reducers%23writing-reducers)

Reducer это чистая функция, которая принимает параметрами state и action. Внутри редюсера мы отслеживаем тип полученного actions и в зависимости от него мы изменяем состояние и возвращаем новый объект состояния.

## 13. Redux. Что такое экшн и как изменить состояние?

**Подробнее -** [**actions**](https://redux.js.org/faq/actions%23actions)

Action - это простой js объект, у которого обязательно должно быть поле с типом.

{

type: “SET\_PAGE”

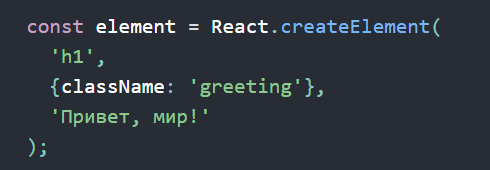
}

Также опционально можно добавить какие-то данные. Для того что бы изменить состояние необходимо вызвать функцию dispatch, в которую передаем action.

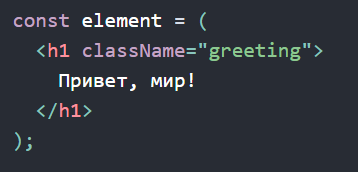
## 14. Что такое JSX?

**Подробнее -** [**jsx**](https://ru.react.js.org/docs/introducing-jsx.html)

По умолчанию чтобы создавать элементы в реакт используется такой синтаксис



Но мы привыкли видеть его вот таким



именно такая разметка и называется jsx. Это некое расширение языка упрощающее восприятие кода и разработку

## 15. Что такое PROPS?

**Подробнее -**

Props – данные, которые передаются в компонент из родительского. Props доступны только для чтения и не могут быть изменены.

## 16. Отличие в записях

**Подробнее -** [**props**](https://reactjs.org/docs/components-and-props.html)

На верхнем скриншоте мы передаем в функцию, изменяющее состояние значения, а на нижнем колбэк, отличие в том, что

Если мы вызовем сэт стэйт с верхнего скриншота несколько раз подрят, состояние изменится лишь единожды, тк

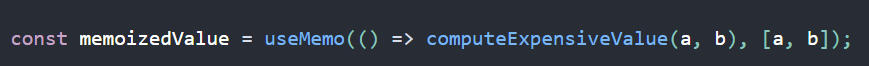
во все вызовы попадет текущее состояние, в случае колбэка, мы работаем с предыдущим состоянием, и в случае нескольких

вызовов состояние изменится ровно столько раз, сколько было вызовов

## 17. useMemo для чего нужен и когда использовать?

**Подробнее -** [**useMemo**](https://ru.reactjs.org/docs/hooks-reference.html%23usememo)

useMemo используется для того, чтобы закешировать\замемоизировать результат вычислений.



Передайте «создающую» функцию и массив зависимостей. useMemo будет повторно вычислять мемоизированное значение только тогда, когда значение какой-либо из зависимостей изменилось. Эта оптимизация помогает избежать дорогостоящих вычислений при каждом рендере.  
Первым параметром функция принимает callback, в котором проходят вычисления, а вторым массив зависимостей, функция будет повторно проводить вычисления только при изменении хотя бы одной из зависимостей.

## 18. useCallback для чего нужен и когда использовать?

**Подробнее -** [**useCallback**](https://ru.reactjs.org/docs/hooks-reference.html%23usecallback)

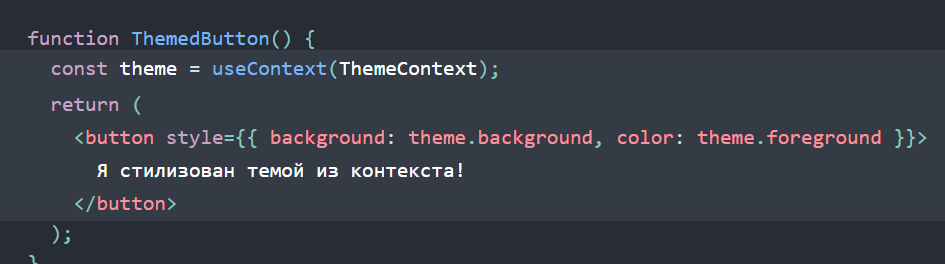
Хук useCallback вернёт мемоизированную версию колбэка, который изменяется только, если изменяются значения одной из зависимостей. Это полезно при передаче колбэков оптимизированным дочерним компонентам, которые полагаются на равенство ссылок для предотвращения ненужных рендеров

## 19. useContext для чего нужен и когда использовать?

**Подробнее -** [**useContext**](https://ru.reactjs.org/docs/hooks-reference.html%23usecontext)

В типичном React-приложении данные передаются сверху вниз (от родителя к дочернему компоненту) с помощью пропсов. Однако, подобный способ использования может быть чересчур громоздким для некоторых типов пропсов (например, выбранный язык, UI-тема), которые необходимо передавать во многие компоненты в приложении. Контекст предоставляет способ делиться такими данными между компонентами без необходимости явно передавать пропсы через каждый уровень дерева.

Компонент, вызывающий useContext, всегда будет перерендериваться при изменении значения контекста. Если повторный рендер компонента затратен, вы можете оптимизировать его с помощью мемоизации.

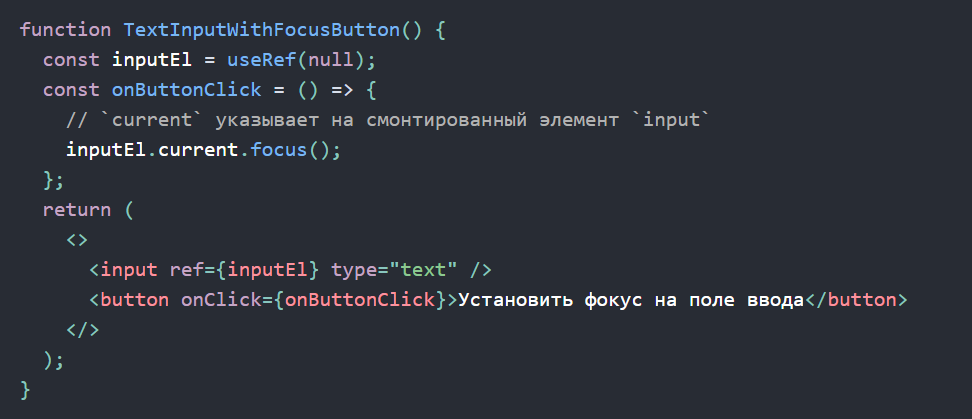


## 20. useRef для чего нужен и когда использовать?

**Подробнее -** [**useRef**](https://ru.reactjs.org/docs/hooks-reference.html%23useref)

useRef возвращает изменяемый ref-объект, свойство .current которого инициализируется переданным аргументом (initialValue). Возвращённый объект будет сохраняться в течение всего времени жизни компонента и не будет изменяться от рендера к рендеру.

Обычный случай использования — это доступ к потомку в императивном стиле. Т.е. используя ref, мы можем явно обратиться к DOM элементу.



## 21. React.memo для чего нужен и когда использовать?

**Подробнее -** [**memo**](https://ru.reactjs.org/docs/react-api.html%23reactmemo)

React.memo — это компонент высшего порядка.

Если ваш компонент всегда рендерит одно и то же при неменяющихся пропсах, вы можете обернуть его в вызов React.memo для повышения производительности в некоторых случаях, мемоизируя тем самым результат. Это значит, что React будет использовать результат последнего рендера, избегая повторного рендеринга.

React.memo затрагивает только изменения пропсов. Если функциональный компонент обёрнут в React.memo и использует useState, useReducer или useContext, он будет повторно рендериться при изменении состояния или контекста.

## 22. Расскажите о React fiber?

**Подробнее -** [**react fiber**](https://ru.reactjs.org/docs/faq-internals.html)

Fiber — новый механизм согласования в React 16, основная цель которого сделать рендеринг виртуального DOM инкрементным.

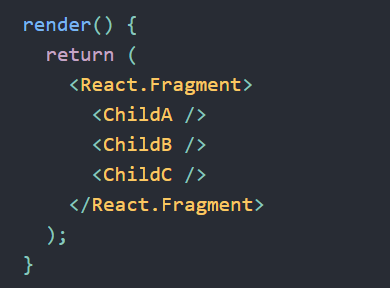
React Fiber — прогрессивная реализация ключевого алгоритма React. Это кульминационное достижение двухгодичных исследований команды разработчиков React.

Цель Fiber в увеличении производительности при разработке таких задач как анимация, организация элементов на странице и движение элементов. Ее главная особенность — это инкрементный рендеринг: способность разделять работу рендера на единицы и распределять их между множественными фреймами.

## 23. Что такое React fragment?

**Подробнее -** [**fragment**](https://ru.reactjs.org/docs/fragments.html)

Возврат нескольких элементов из компонента является распространённой практикой в React. Фрагменты позволяют формировать список дочерних элементов, не создавая лишних узлов в DOM.



## 24. Расскажите о React Reconciliation

**Подробнее -** [**Согласование**](https://ru.reactjs.org/docs/reconciliation.html%23gatsby-focus-wrapper)

Reconciliation - это алгоритм React, используемый для того, чтобы отличить одно дерево элементов от другого для определения частей, которые нужно будет заменить. Сверка — это алгоритм, за которым стоит то, что мы привыкли называть «Virtual DOM». Определение звучит как-то так: когда вы рендерите React приложение, дерево элементов, которое описывает приложение генерируется в зарезервированной памяти. Это дерево потом включается в рендеринг окружение — на примере браузерного приложения, оно переводится в набор DOM операций. Когда состояние приложения обновляется (обычно вызовом setState), новое дерево генерируется. Новое дерево сравнивается с предыдущим, чтоб просчитать и включить именно те операции, которые нужны для перерисовки обновленного приложения.

## 25. Для чего нужны ключи key в списках?

**Подробнее -** [**Ключи**](https://ru.reactjs.org/docs/lists-and-keys.html%23keys)

Ключи помогают React определять, какие элементы были изменены, добавлены или удалены. Их необходимо указывать, чтобы React мог сопоставлять элементы массива с течением времени.

Лучший способ выбрать ключ — это использовать строку, которая будет явно отличать элемент списка от его соседей. Чаще всего вы будете использовать ID из ваших данных как ключи.

## 26. Асинхронные actions в redux с помощью thunk.

**Подробнее -** [**redux thunk**](https://habr.com/ru/post/483314/)

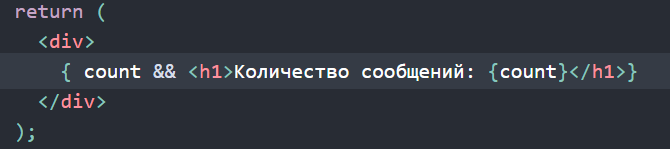
Для того, чтобы использовать redux thunk необходимо подключить его как middleware. Action creators должны возвращать не просто объект, а функцию, которая параметром принимает dispatch. Cм. скриншот.

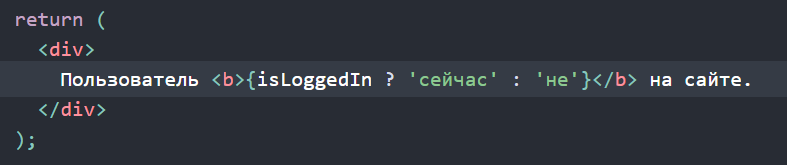


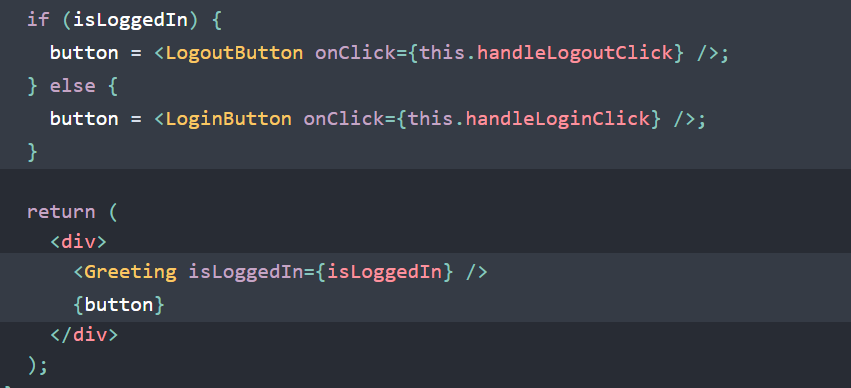
## 27. Как отрисовать блок по условию?

**Подробнее -** [**Условный рендеринг**](https://ru.reactjs.org/docs/conditional-rendering.html)

Можно использовать любые условные операторы, в том числе тернарный. Примеры на скришотах.







## 28. Как отследить изменение поля объекта в функциональном компоненте?

**Подробнее -** [**UseEffect**](https://ru.reactjs.org/docs/hooks-reference.html%23useeffect)

Для этого необходимо воспользоваться хуком useEffect и в массив зависимостей передать поле объекта.

## 29. Как получить доступ к дом элементу в React.

**Подробнее –** [**refs**](https://ru.reactjs.org/docs/refs-and-the-dom.html)

Рефы создаются с помощью React.createRef() или с помощью хука useRef() и прикрепляются к React-элементам через ref атрибут. Обращаясь к созданной ссылке мы можем получить доступ к дом элементу с помощью ref.current.

# **Vue**

## 1. Что такое двустороннее связывание?

**Подробнее -** [**v-model**](https://v3.ru.vuejs.org/ru/guide/forms.html%23%D0%BE%D0%B1%D1%8B%D1%87%D0%BD%D0%BE%D0%B5-%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)

Двусторонняя привязка позволяет динамически менять значения на одном конце привязки при изменениях на другом конце. Как правило, двусторонняя привязка применяется при работе с элементами ввода, например, элементами типа input.

Во Vue для двустороннего связывания предназначена директива v-model. По умолчанию v-model работает по схеме, представленной в таблице.



## 2. Какими способами можно реализовать двустороннее связывание для input?

1. Как атрибут :value связать с одной из моделей, которую возвращает функция data(), и прослушивать событие oninput, доставать оттуда event.target.value и помещать результат в модель.

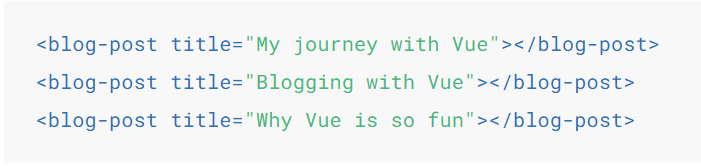
2. Использовать директиву v-model.

## 3. Что такое props?

**Подробнее -** [props](https://ru.vuejs.org/v2/guide/components-props.html)

Входные параметры — это пользовательские атрибуты, которые вы можете установить на компоненте. Когда значение передаётся в атрибут входного параметра, оно становится свойством экземпляра компонента. Чтобы передать заголовок в компонент нашей записи блога, мы можем включить его в список входных параметров, которые принимает компонент, с помощью опции props.

Компонент может принимать столько входных параметров, сколько захотите, и по умолчанию любое значение может быть передано в любой входной параметр.

## 4. Как работает реактивность в Vue?

**Подробнее –** [реактивность](https://v3.ru.vuejs.org/ru/guide/reactivity.html)

**\* Реактивность в Vue 2 и Vue 3 работает с отличиями.**

Реактивность — концепция, которая позволяет приспосабливаться к изменениям декларативным способом. Это означает, что при любом изменении данных происходит автоматическое изменение выводимого значения в шаблоне.

Фреймворк Vue.js сконфигурирован таким образом, чтобы автоматически добавлять возможность реактивности для любого из создаваемых в модели данных свойств: data, computed property, props и т. п.

Такая автоматизация предоставляет ряд несомненных преимуществ при создании приложений на Vue.js:

* сокращает время на разработку приложения
* делает исходный код приложения более лаконичным
* помогает минимизировать когнитивную нагрузку разработчика

Когда возвращается простой объект JavaScript из функции data компонента, Vue обернёт его в Proxy (opens new window)с обработчиками для get и set. Прокси были представлены в ES6 и позволяют Vue 3 избавиться от ограничений системы реактивности, которые существовали в предыдущих версиях Vue. Proxy — объект, который содержит в себе другой объект или функцию и позволяет «перехватывать» их.

Проксируемый объект невидим для пользователя, но под капотом он позволяет Vue отслеживать зависимости и уведомлять о считывании свойств или их изменениях.

## 5. Что такое Composition API в Vue 3?

**Подробнее -** [Composition API](https://v3.ru.vuejs.org/ru/guide/composition-api-introduction.html%23%D0%BE%D0%BF%D1%86%D0%B8%D1%8F-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0-setup)

## 6. Особенности использования v-model в Vue 2 и Vue 3.

**Vue 2**

Используется лишь одна директива v-model и по умолчанию она работает с атрибутом value и событием input.

**Vue 3**

Можно использовать несколько v-model:

v-model=”inputValue”

v-model:visible=”bool”

v-model:contract=”contract”

Неименнованый v-model по умолчанию работает с атрибутом modelValue и событием update:modelValue

В остальных случаях работаем с одноименным атрибутом и событием update:[название атрибута]

## 7. Для чего нужен $emit?

**Подробнее -** [пользовательские события](https://ru.vuejs.org/v2/guide/components-custom-events.html)

Через props мы можем передать данные от родительского компонента в дочерний. Но что если мы хотим также передавать данные и в обратном направлении: от дочернего компонента к родителю? В этом случае необходимо определить свои события. Дочерний компонент будет генерировать событие с помощью вызова this.$emit(имя\_события), а родительский компонент будет отлавливать это событие с помощью установки атрибута v-on:название\_события и при получении события поизводить определенные действия

## 8. Постраничная навигация в Vue.

**Подробнее -** [vue router](https://router.vuejs.org/ru/)

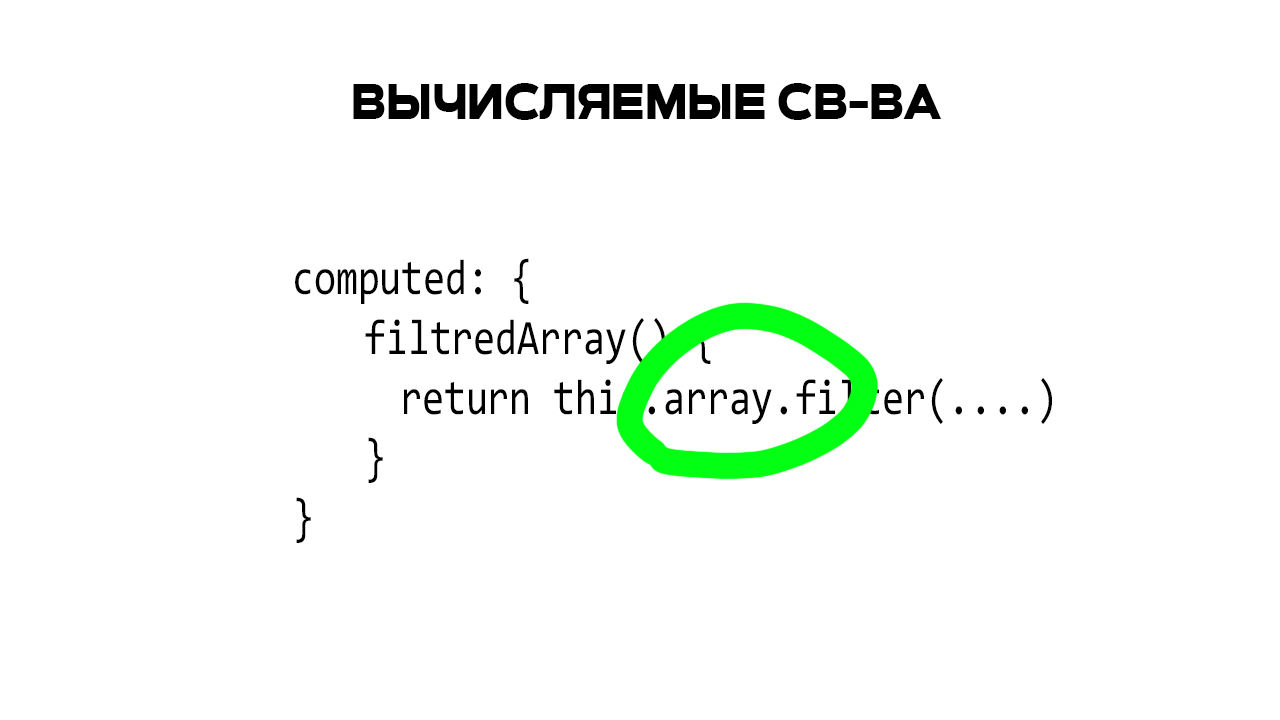
Vue Router — официальная библиотека маршрутизации для Vue.js (opens new window). Она глубоко интегрируется с Vue.js и позволяет легко создавать SPA-приложения. Включает следующие возможности:

* Вложенные маршруты/представления
* Модульная конфигурация маршрутизатора
* Доступ к параметрам маршрута, query, wildcards
* Анимация переходов представлений на основе Vue.js
* Удобный контроль навигации
* Автоматическое проставление активного CSS класса для ссылок
* Режимы работы HTML5 history или хэш, с авто-переключением в IE9
* Настраиваемое поведение прокрутки страницы

## 9. Что такое computed свойства и как они работают?

**Подробнее -** [computed](https://v3.ru.vuejs.org/ru/guide/computed.html)

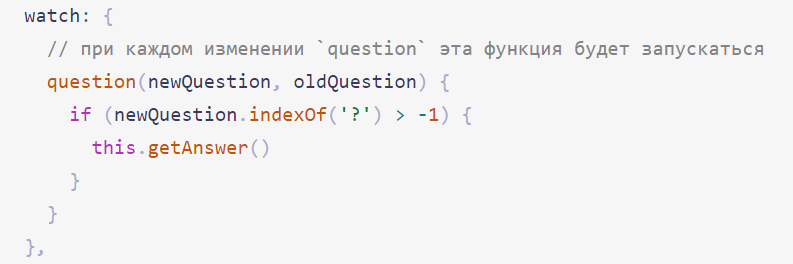
Computed св-ва – функции, которые должны возвращать результат вычислений. Вычисляемые свойства кэшируются на основе своих реактивных зависимостей. Т.е. функция будет заново производить вычисления только при изменении реактивных моделей, которые используются внутри этой функции.



## 10. Как отследить изменение модели?

**Подробнее -** [watch](https://v3.ru.vuejs.org/ru/guide/computed.html%23%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B-%D0%BD%D0%B0%D0%B1%D0%BB%D1%8E%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D0%B8)

Иногда нужно отследить факт изменений в данных. Поэтому Vue предоставляет ещё один способ реагировать на изменения данных с помощью опции watch. Это полезно, если необходимо выполнять асинхронные или затратные операции в ответ на изменение данных.

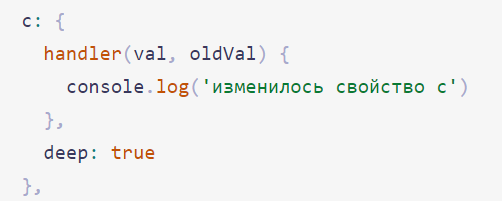


## 11. Особенности слежения за «глубокими» объектами.

**Подробнее –** [Watch API](https://v3.ru.vuejs.org/ru/api/options-data.html%23watch)

Для «глубокого» отслеживания изменений необходимо добавить deep: true.

Функция handler будет являться функцией наблюдателем.



## 12. Что такое интерполяция?

**Подробнее -** [Интерполяция](https://ru.vuejs.org/v2/guide/syntax.html)

Vue.js позволяет декларативным образом устанавливать привязку между элементами веб-страницы и данными объекта Vue. Простейшую форму привязки представляет интерполяция строк. В этом случае значение, к которому выполняется привязка, заключается в двойные фигурные скобки



## 13. Как отрисовать компонент по условию?

**Подробнее -** [Условная отрисовка](https://ru.vuejs.org/v2/guide/conditional.html)

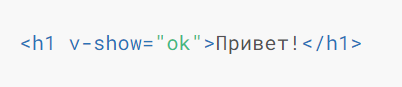
Директива v-if используется для рендеринга блока по условию. Блок будет отображаться только в том случае, если выражение директивы возвращает значение, приводимое к true.



Также можно добавить блок «иначе», используя директиву v-else



Ещё одну возможность условного отображения предоставляет директива v-show



## 14. Разница между v-if и v-show.

**Подробнее –** [v-if v-show](https://ru.vuejs.org/v2/guide/conditional.html%23v-if-%D0%B2-%D1%81%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B8-%D1%81-v-show)

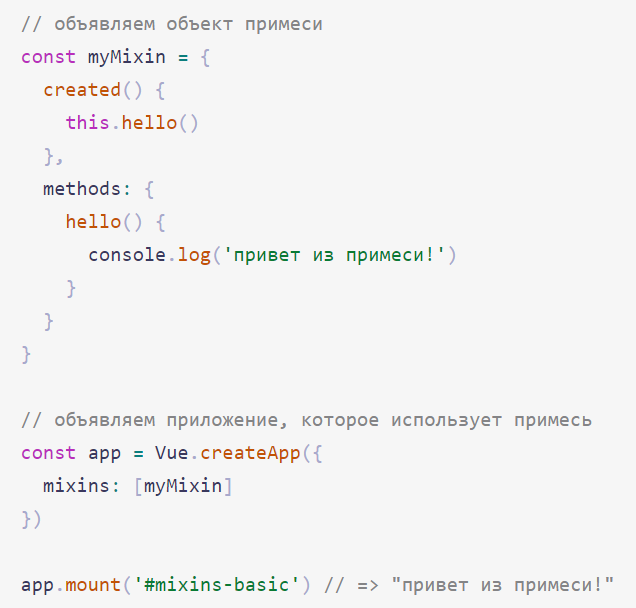
Разница в том, что элемент с v-show будет всегда оставаться в DOM, а изменяться будет лишь свойство display в его параметрах CSS.

* v-if производит «настоящую» условную отрисовку, удостоверяясь что подписчики событий и дочерние компоненты внутри блока должным образом уничтожаются и воссоздаются при изменении истинности управляющего условия.
* v-if ленивый: если условие ложно на момент первоначальной отрисовки, он не произведёт никаких действий — условный блок не будет отображён, пока условие не станет истинным.
* v-show, напротив, куда проще: элемент всегда присутствует в DOM, и только CSS-свойство переключается в зависимости от условия.
* В целом, у v-if выше затраты на переключения, а у v-show больше затрат на первичную отрисовку. Так что если вы предполагаете, что переключения будут частыми, используйте v-show, если же редкими или маловероятными — v-if.

## 15. Что такое миксины и как их использовать?

**Подробнее -** [mixins](https://v3.ru.vuejs.org/ru/guide/mixins.html%23%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D1%8B)

Примеси (mixins) — инструмент для повторного использования кода в компонентах Vue. В объекте примеси могут содержаться любые опции компонентов. При использовании примеси в компоненте все её опции будут «подмешиваться» к опциям компонента.



## 16. Что такое директивы?

**Подробнее -** [**directives**](https://v3.ru.vuejs.org/ru/api/directives.html)

Директивы Vue.js это специальные атрибуты, которые мы можем использовать внутри html шаблона компонента vue, для того чтобы тем или иным образом взаимодействовать с html тегами и не только. Все директивы обычно начинаются с буквы v, затем через дефис идет название директивы. Директивы — специальные атрибуты для добавления элементам html дополнительной функциональности.

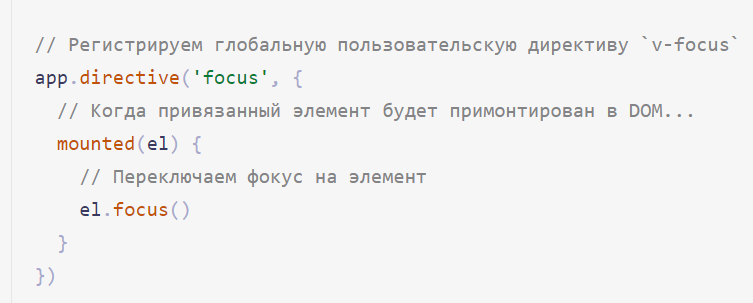
## 17. Как создать пользовательскую директиву? Особенности использования.

**Подробнее -** [**пользовательские директивы**](https://v3.ru.vuejs.org/ru/guide/custom-directive.html%23%D0%B2%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5)

Кроме использования встроенных директив (таких как v-model, v-show), Vue позволяет создавать пользовательские. При этом важно понимать, что основным механизмом для создания повторно используемого кода во Vue всё-таки являются компоненты. Но для выполнения низкоуровневых операций с DOM директивы могут быть очень полезны.

Пользовательская директива представляет из себя объект, обладающий жизненным циклом.

Данная директива будет устанавливать фокус на элемент, который использует эту директиву.



Также в директиву можно передавать динамические аргументы и некоторые данные.

## 18. Расскажите о жизненном цикле компонента Vue.

**Подробнее -** [**Жизненный цикл**](https://v3.ru.vuejs.org/ru/guide/instance.html%23%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D0%BA%D0%BB%D0%B0)

Здесь лучше посмотреть на диаграмму по ссылке, все подробно расписано.

## 19. В каком методе жизненного цикла необходимо делать первичную загрузку данных с сервера?

**Подробнее -** [**Жизненный цикл**](https://v3.ru.vuejs.org/ru/guide/instance.html%23%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D0%BA%D0%BB%D0%B0)

Получение данных с сервера рекомендуется делать в методе жизненного цикла mounted;

## 20. В каком методе жизненного цикла необходимо делать очистку (удалять слушатели, очищать хранилище и т.д.)?

**Подробнее -** [**Жизненный цикл**](https://v3.ru.vuejs.org/ru/guide/instance.html%23%D0%B4%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0-%D0%B6%D0%B8%D0%B7%D0%BD%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE-%D1%86%D0%B8%D0%BA%D0%BB%D0%B0)

Для этого предназначен метод жизненного цикла unmounted, который отрабатывает в тот момент, когда компонент демонтируется.

## 21. Как сделать стили локальными для компонента?

Для этого используется свойство scoped.



## 22. Как отрисовать несколько компонентов на основе массива?

**Подробнее -** [**v-for**](https://v3.ru.vuejs.org/ru/guide/list.html%23%D0%BE%D1%82%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%B0-%D1%87%D0%B5%D1%80%D0%B5%D0%B7-v-for)

Используйте директиву v-for для отрисовки списка элементов на основе массива данных. У директивы v-for специальный синтаксис: item in items, где items — исходный массив, а item — ссылка на итерируемый элемент массива.



## 23. Зачем указывать key при использовании директивы v-for?

**Подробнее -** [**keys**](https://v3.ru.vuejs.org/ru/api/special-attributes.html%23key)

Специальный атрибут key в первую очередь нужен в качестве подсказки для Vue и его алгоритма виртуального DOM для идентификации VNode при сравнениях обновлённого списка узлов со старым. Без ключей Vue использует алгоритм, который минимизирует перемещения элементов и по-максимуму будет стараться изменять/переиспользовать элементы одного типа. При использовании ключей элементы будут переупорядочиваться в соответствии с изменением порядка следования ключей, а элементы, чьи ключи уже отсутствуют, будут всегда удаляться/уничтожаться.

Потомки одного и того же общего родителя должны иметь уникальные ключи. Появление дубликатов ключей будет приводить к ошибкам при отрисовке.

## 24. Как отследить изменение поля объекта?

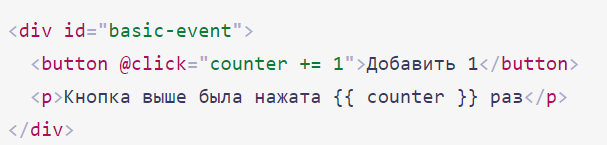
Для этого необходимо создать функцию-наблюдатель, название которой задано по шаблону представлену на скриншоте.



## 25. Как добавить слушатель события на элемент?

**Подробнее -** [**События**](https://v3.ru.vuejs.org/ru/guide/events.html%23%D0%BF%D1%80%D0%BE%D1%81%D0%BB%D1%83%D1%88%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B8)

Можно использовать директиву v-on, которую обычно сокращают до символа @, чтобы прослушивать события DOM и запускать какой-нибудь JavaScript-код по их наступлению. Используется как v-on:click="methodName" или в сокращённом виде @click="methodName".



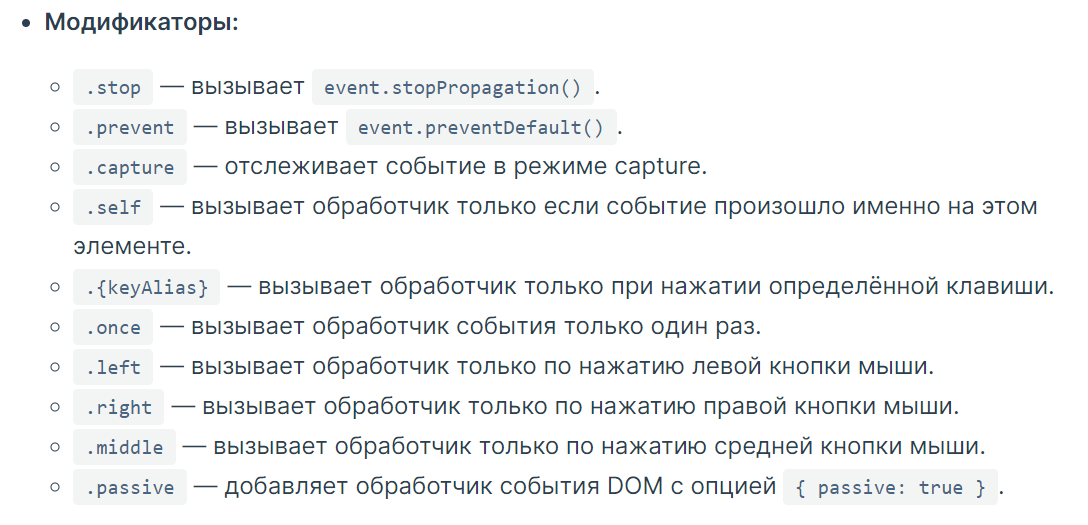
## 26. Что такое модификаторы?

**Подробнее -** [**Модификаторы событий**](https://ru.vuejs.org/v2/guide/events.html%23%D0%9C%D0%BE%D0%B4%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D1%8B-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9)

Модификаторы позволяют изменить поведение директивы. Например, для директивы v-on с помощью модификатора prevent мы можем отключить дефолтное поведение браузера.

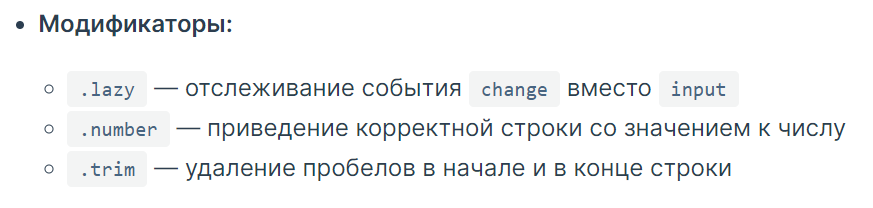
## 27. Какие модификаторы есть у событий?

**Подробнее –** [**V-ON api**](https://v3.ru.vuejs.org/ru/api/directives.html%23v-on)



## 28. Какие модификаторы есть у v-model?

**Подробнее –** [**v-model API**](https://v3.ru.vuejs.org/ru/api/directives.html%23v-model)



## 29. Как добавить анимацию на удаление\добавление элемента в список?

**Подробнее –** [**Анимации списков**](https://v3.ru.vuejs.org/ru/guide/transitions-list.html%23%D0%B0%D0%BD%D0%B8%D0%BC%D0%B0%D1%86%D0%B8%D0%B8-%D0%B4%D0%BE%D0%B1%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%B8-%D1%83%D0%B4%D0%B0%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2-%D1%81%D0%BF%D0%B8%D1%81%D0%BA%D0%B0)

Необходимо обернуть список, в котором используется директива v-if в специальный компонент <transition-group name=”название списка”>

* Каждый элемент внутри <transition-group> всегда должен быть с уникальным значением атрибута key.
* CSS-классы переходов будут применяться к внутренним элементам, а не к самой группе/контейнеру.

В css создаем классы, которые называем по особому шаблону. Указывается name из компонента transition group, после чего этап жизненного цикла анимации.



## 30. Как зарегистрировать компонент глобально в Vue 3?

**Подробнее –**

## 31. Как передать данные из родительского компонента в дочерний, не используя props и store.

**Подробнее –** [**Компонент**](https://v3.ru.vuejs.org/ru/api/application-api.html%23component)

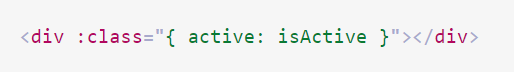
const app = createApp()

app.component(name, component)

## 32. Как добавить класс на элемент по условию?

**Подробнее –** [**Связывание CSS классов**](https://v3.ru.vuejs.org/ru/guide/class-and-style.html%23%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-css-%D0%BA%D0%BB%D0%B0%D1%81%D1%81%D0%BE%D0%B2)

Для динамической установки или удаления CSS-классов в директиву :class (сокращение для v-bind:class) можно передавать объект:



Синтаксис выше означает, что наличие класса active на элементе будет определяться истинностью (opens new window) значения свойства isActive.

Подобным образом можно управлять несколькими классами, добавляя в объект и другие поля.

## 33. Как динамически изменять стили у элемента?

**Подробнее –** [**Связывание стилей**](https://v3.ru.vuejs.org/ru/guide/class-and-style.html%23%D1%81%D0%B2%D1%8F%D0%B7%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-inline-%D1%81%D1%82%D0%B8%D0%BB%D0%B5%D0%B8)

Объектный синтаксис для :style выглядит почти как для CSS, за исключением того, что это объект JavaScript. Поэтому для указания имён свойств CSS можно использовать как camelCase, так и kebab-case (не забывайте про кавычки при использовании kebab-case)



## 34. Расскажите о слотах в Vue.

**Подробнее –** [**Слоты**](https://v3.ru.vuejs.org/ru/guide/component-slots.html%23%D1%81%D0%BE%D0%B4%D0%B5%D1%80%D0%B6%D0%B8%D0%BC%D0%BE%D0%B5-%D1%81%D0%BB%D0%BE%D1%82%D0%B0)

Слоты представляют способ создания фиксированной структуры компонента, при котором содержимое для различных частей компонента определяет родительский компонент. Во Vue.js слоты реализуются через элемент <slot>, вместо которого родительский компонент вставляет содержимое в дочерний компонент.



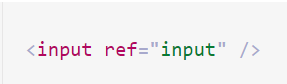
Также можно создавать именованные слоты.



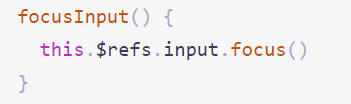
## 35. Каким способом можно получить DOM элемент во Vue?

**Подробнее –** [**Ссылки на элементы в шаблоне**](https://v3.ru.vuejs.org/ru/guide/component-template-refs.html)

Несмотря на существование входных параметров и событий, иногда может потребоваться прямой доступ к дочернему компоненту в JavaScript. Для этого можно через атрибут ref присвоить специальный ID ссылки дочернему компоненту или HTML-элементу. Например:



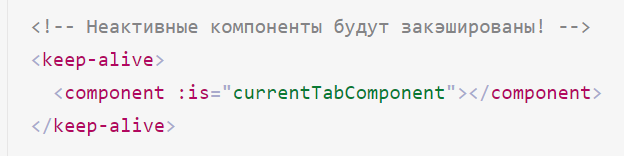
Для получения доступа к нему используется объект $refs, доступный в рамках компонента.



## 36. Предназначение Keep-alive в Vue

**Подробнее –** [**keep alive**](https://v3.ru.vuejs.org/ru/guide/component-dynamic-async.html%23%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%8B-%D1%81-keep-alive)

При переключении между компонентами может потребоваться сохранять их состояние или избежать перерисовки для производительности. Обычно пересоздание динамических компонентов полезно, но в некоторых случаях хотелось бы чтобы экземпляры компонентов вкладок кэшировались после их создания в первый раз. Для решения этой проблемы динамический компонент можно обернуть в <keep-alive>



## 37. Телепорты в Vue. Зачем нужны?

**Подробнее –** [**телепорты**](https://v3.ru.vuejs.org/ru/guide/teleport.html%23%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%81-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0%D0%BC%D0%B8)

Vue поощряет создавать пользовательские интерфейсы, инкапсулируя их и связанное с ним поведение в компоненты. Вкладывая компоненты друг в друга можно получить дерево компонентов, из которого и будет строиться пользовательский интерфейс приложения.

Но иногда случается, что часть этого шаблона логически принадлежит компоненту, хотя с технической точки зрения было бы удобнее переместить эту часть шаблона в какое-нибудь другое место в DOM или даже вне приложения Vue.

Один из частых сценариев — создание компонента, содержащего в себе полноэкранное модальное окно. В большинстве случаев, удобней когда логика модального окна внутри компонента, но позиционирование модального окна с помощью CSS становится сложной задачей, что может потребовать даже кардинальных изменений в композиции компонентов.

Компонент <teleport> позволяет перемещать элементы по ДОМ дереву.



## 38. Как подключить внешний плагин в Vue 3?

**Подробнее –** [**plugins**](https://v3.ru.vuejs.org/ru/guide/plugins.html%23%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BF%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD%D0%B0)

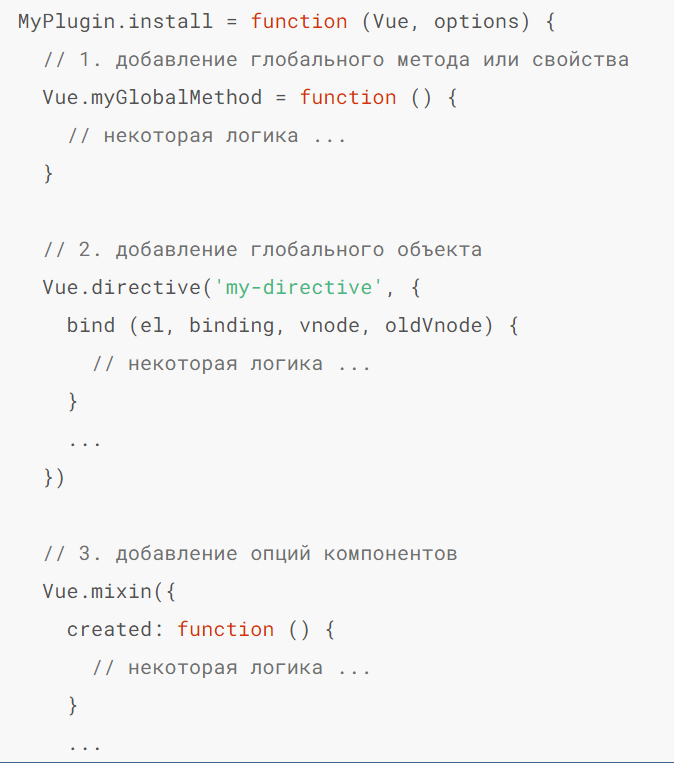
Vue.use(plugin) // v2

app.use(plugin) // v3

## 39. Как создать собственный плагин в Vue 3?

**Подробнее –** [**plugins**](https://ru.vuejs.org/v2/guide/plugins.html)

Плагин Vue.js должен содержать метод install. Этот метод будет вызываться с конструктором Vue в качестве первого аргумента, и с дополнительными опциями плагина в качестве второго (если передавались).



## 40. При использовании хуков жизненного цикла в миксине и при подключении этого миксина в компонент, в какой последовательности будут вызываться хуки?

**Подробнее –** [**Слияний опций миксин**](https://ru.vuejs.org/v2/guide/mixins.html%23%D0%A1%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D0%B5-%D0%BE%D0%BF%D1%86%D0%B8%D0%B9)

Функции хуков с одинаковыми именами объединяются в массив, чтобы все они вызывались. Хуки примеси будут вызываться **перед** собственными хуками компонента.

## 41. Почему не стоит использовать в качестве ключей (key) индексы элемента массива?

Ключ всегда должен быть связан с элементом массива и не должен изменяться. При удалении\перестановке элементов в массиве, индекс и элемент, с которым он был сопоставлен нарушается. При этом пропадает эффективность использования ключей.

## 42. Почему этот код не работает? array.filter(elem => elem % 2 !== 0)

Функция filter возвращает новый массив, необходимо присвоить результат выполнении фильтрации.

this.array = this.array.filter(…)

## 43. Можно ли использовать v-if и v-for на одном элементе? Объясните свой ответ.

**Подробнее -** [**vfor vif**](https://v3.ru.vuejs.org/ru/style-guide/%23%D1%83%D0%BD%D0%B8%D0%BA%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BB%D1%8E%D1%87%D0%B8-%D0%B4%D0%BB%D1%8F-v-for-%D0%B2%D0%B0%D0%B6%D0%BD%D0%BE)

**Никогда не используйте**v-if**на том же элементе, что и**v-for**.** **Когда Vue обрабатывает директивы, то v-if имеет более высокий приоритет чем v-for.**

Есть два распространённых случая, когда это может быть заманчиво:

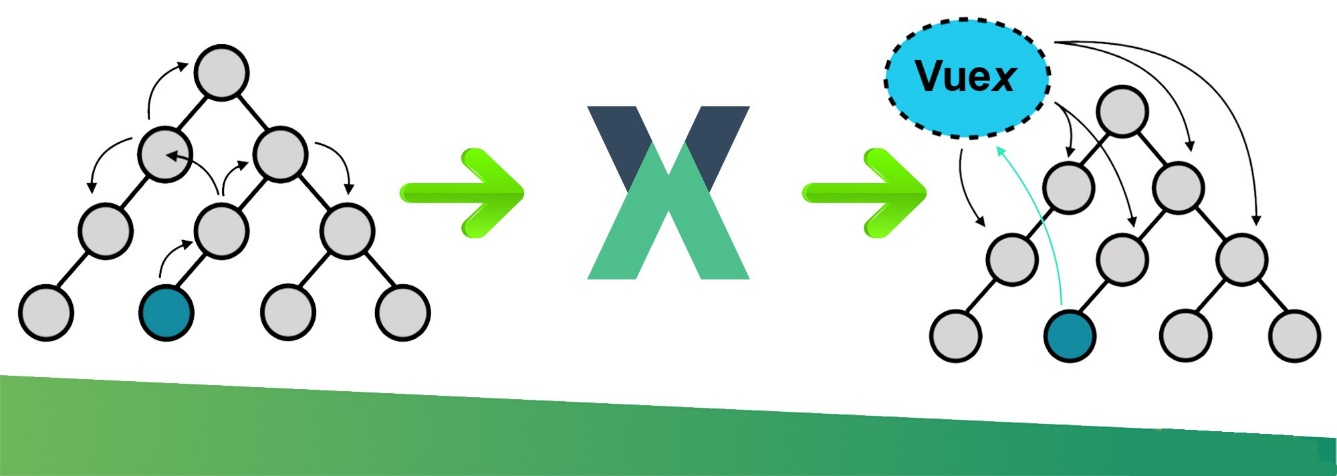
* Для фильтрации элементов в списке (например, v-for="user in users" v-if="user.isActive"). В этих случаях замените users новым вычисляемым свойством, которое возвращает отфильтрованный список (например, activeUsers).
* Чтобы избежать отображения списка, если он должен быть скрыт (например, v-for="user in users" v-if="shouldShowUsers"). В этих случаях переместите v-if выше на элемент контейнера (например, ul, ol).

## 44. Можно ли изменять computed свойства?

Изменять напрямую значение computed св-ва не рекомендуется. Необходимо изменить одну из зависимостей, которая используется внутри computed функции или же создать новую computed функцию, которая будет внутри себя работать с первой computed функцией.

## 45. Для чего нужен vuex и какие проблемы он решает?

Vuex — паттерн управления состоянием + библиотека для приложений на Vue.js. Он служит централизованным хранилищем данных для всех компонентов приложения с правилами, гарантирующими, что состояние может быть изменено только предсказуемым образом. Vuex интегрируется с официальным расширением vue-devtools (opens new window), предоставляя «из коробки» такие продвинутые возможности, как «машину времени» для отладки и экспорт/импорт слепков состояния данных.



## 46. Расскажите о state и getters в vuex.

**Подробнее -** [**state**](https://vuex.vuejs.org/ru/guide/state.html)

**Подробнее -** [**getters**](https://vuex.vuejs.org/ru/guide/getters.html)

Vuex использует единое дерево состояния — когда один объект содержит всё глобальное состояние приложения и служит «единственным источником истины». Это также означает, что в приложении будет только одно такое хранилище. Единое дерево состояния позволяет легко найти нужную его часть или делать снимки текущего состояния приложения в целях отладки. Хранилище Vuex реактивно.

В state мы храним данные, с которыми предстоит работать.

Можете считать их вычисляемыми свойствами хранилища. Как и вычисляемые свойства, результаты геттера кэшируются, на основе его зависимостей и пересчитываются только при изменении одной из зависимостей.

## 47. Расскажите о мутациях и действиях в vuex. В чем отличие?

**Подробнее -** [**mutations**](https://vuex.vuejs.org/ru/guide/mutations.html%23%D0%BC%D1%83%D1%82%D0%B0%D1%86%D0%B8%D0%B8-%D1%81-%D0%BD%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%BE%D0%B8)

**Подробнее -** [**actions**](https://vuex.vuejs.org/ru/guide/actions.html%23%D0%B4%D0%B8%D1%81%D0%BF%D0%B5%D1%82%D1%87%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%B4%D0%B5%D0%B8%D1%81%D1%82%D0%B2%D0%B8%D0%B8)

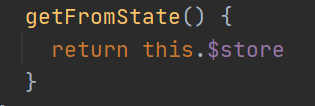
Единственным способом изменения состояния хранилища во Vuex являются мутации. Мутации во Vuex очень похожи на события: каждая мутация имеет строковый тип и функцию-обработчик. В этом обработчике и происходят, собственно, изменения состояния, переданного в функцию первым аргументом

Действия — похожи на мутации с несколькими отличиями:

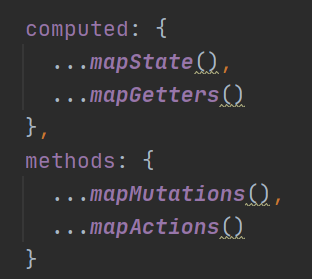
* Вместо того, чтобы напрямую менять состояние, действия инициируют мутации;
* Действия могут использоваться для асинхронных операций.

## 48. Как использовать store внутри компонента?

Получать данные из state, вызывать мутации, геттеры и actions можно с помощью объекта $store, который доступен в контексте компонента.



Также существуют специальные функции, которые позволяют получить стейт, геттеры, мутации и экшны.



## 49. Как принудительно обновить компонент в Vue?

Для принудительного ререндеринга компонента существует специальная функция $forceUpdate();

## 50. Для чего нужны асинхронные компоненты в Vue?

**Подробнее -** [**async components**](https://v3.ru.vuejs.org/ru/guide/component-dynamic-async.html%23%D0%B4%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%8B-%D1%81-keep-alive)

В больших приложениях может потребоваться разделять приложение на меньшие части и загружать компоненты с сервера только когда они необходимы. Для реализации подобного Vue предоставляет метод defineAsyncComponent



# **Общие вопросы**

## 1.Что такое HTTP

**Подробнее -** [**http**](https://habr.com/ru/post/215117/)

## 2. Из чего состоит HTTP запрос

**Подробнее -** [**Структура http**](https://ru.wikipedia.org/wiki/HTTP%23%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_HTTP-%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F)

Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:

* Стартовая строка (англ. Starting line) — определяет тип сообщения;
* Заголовки (англ. Headers) — характеризуют тело сообщения, параметры передачи и прочие сведения;
* Тело сообщения (англ. Message Body) — непосредственно данные сообщения. Обязательно должно отделяться от заголовков пустой строкой.

## 3. Какие методы http запросов вы знаете

**Подробнее -** [**Методы http**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods)

* Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.
* HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.
* POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.
* PUT заменяет все текущие представления ресурса данными запроса.
* DELETE удаляет указанный ресурс.
* CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.
* OPTIONS используется для описания параметров соединения с ресурсом.
* TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.
* PATCH используется для частичного изменения ресурса.

## 4. В чем семантическое отличие PUT и PATCH

Семантически PATCH обновляет ресурс частично, например перезаписать 1 поле в БД, а PUT перезаписывает ресурс полностью.

## 5. Что такое websockets?

**Подробнее -** [**WebSocket**](https://ru.wikipedia.org/wiki/WebSocket)

## 6. Что такое REST API

**Подробнее -** [**Rest API**](https://habr.com/ru/post/483202/)

## 7. Что такое WebRTC?

**Подробнее -** [WebRTC](https://ru.wikipedia.org/wiki/WebRTC)

## 8. Что такое Git?

**Подробнее -** [Git](https://ru.wikipedia.org/wiki/Git)

## 9. Как сделать коммит в Git?

**Подробнее -** [git-commit](https://github.com/git-guides/git-commit)

git commit –m “commit message”

## 10. Как создать новую ветку и перейти на нее в Git?

**Подробнее –** [Ветвление](https://git-scm.com/book/ru/v2/%D0%92%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-Git-%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%B8-%D1%81%D0%BB%D0%B8%D1%8F%D0%BD%D0%B8%D1%8F)

git checkout –b “branch\_name”

## 12. Merge и rebase отличия.

**Подробнее -** [merge rebase](https://medium.com/nuances-of-programming/%D0%B2%D0%BB%D0%B0%D0%B4%D0%B5%D0%B5%D1%88%D1%8C-merge-%D0%BE%D1%81%D0%B2%D0%BE%D0%B9-%D0%B8-rebase-412e36bddce2)

## 13. На каких 3 принципах базируется ООП? Расскажите кратко о каждом.

**Подробнее -** [ООП](https://habr.com/ru/company/otus/blog/525336/)

## 14. Какие паттерны проектирования вы чаще всего используете?

**Подробнее -** [Паттерны каталог](https://refactoring.guru/ru/design-patterns/catalog)

## 15. Для чего нужен package.json?

**Подробнее -** [**package json**](https://tech-wiki.online/ru/package-json.html)

**16. Расскажите о принципах SOLID.**

**Подробнее -** [**SOLID**](https://habr.com/ru/company/mailru/blog/412699/)