# JS

1. В чём отличие между Rest и Spread операторами?

Синтаксис для rest оператора выглядит таким же как и для spread оператора, однако он используется для деструктуризации массивов и объектов. Фактически, rest оператор противоположен spread оператору: последний раскладывает массив на элементы, тогда как первый собирает много элементов в один.

**Spread** оператор (оператор расширения) **'берет'** каждый отдельный элемент итерируемого объекта (массив) и **'распаковывает'** его в другой итерируемый объект (массив).

Что такое **итерируемые** (перебираемые) объекты? К итерируемым объектам можно отнести **все**, что можно **перебрать** с помощью цикла for..of. **Большая** часть задач, где приходится использовать оператор **spread**, касается **массивов** и **строк**.

https://stackdev.blog/blog/spread-rest-operator

2) Что такое деструктуризация и как ей пользоваться?

3) Перечислите основные методы массивов.

4) В чём отличие между методами массивов find и findIndex?

5) Как работает метод массивов map?

6) Как работает метод массивов filter?

7) Как с помощью метода sort отсортировать массив в обратном порядке?

8) Что делает метод Object.assign?

9) Чем метод Object.keys отличается от перебора свойств объекта через цикл for in?

10) Что делают методы Object.freeze и Object.seal?

11) Как мы можем считать или записать прототип конкретного объекта?

12) Может ли дочерний объект изменить свойства наследуемые у прототипа, так что это скажется на прототипе?

13) Какой контекст имеют методы, наследуемые объектом от прототипа?

14) Чем отличается прототип конструктора (Constructor.prototype) от прототипа объекта (\_\_proto\_\_)?

15) Если изменить прототип конструктора, то что будет с прототипами созданных им объектов?

16) Что такое примитивы и имеют ли они свои свойства и методы?

17) Что такое прокси? Как с помощью прокси можно реализовать обращение к отрицательным индексам массива?

18) Что такое генераторы? Как можно можно реализовать генератор, выдающий степени числа 2?

# Async

1) Сколько потоков JS обрабатывают код на странице?

2) Что такое callback функции и зачем они нужны?

3) Что такое промисы?

4) Каким образом исполнитель промиса может оповестить вызывающий код о том, что он завершил свою работу?

5) Каким образом вызывающий код может отследить выполнение промиса?

6) Каким образом реализуются цепочки промисов?

7) Что такое Thenable?

8) Что такое неявный try catch в исполнителе и обработчиках промиса?

9) В чём отличие между методами Promise.all и Promise.allSettled?

10) Что делает и зачем нужно ключевое слово async?

11) Что делает и зачем нужно ключевое слово await? Где его нельзя использовать?

12) Что такое Event Loop? Опишите схему его работы.

13) Чем отличаются микрозадачи и макрозадачи? С помощью каких методов мы можем добавить их в очередь?

14) Что такое Ajax?

15) В чём отличие между XMLHttpRequest и fetch?

16) Что такое WebSockets?

17) В чём отличие между localStorage и sessionStorage?

# Docker

1)Что такое Docker и зачем он нужен?

2) Чем docker-контейнеры отличаются от docker-образов?

3) Что такое docker volumes и зачем они нужны?

4) Что такое Dockerfile и зачем он нужен?

5) Какая команда используется для создания docker образа на основе Dockerfile?

6) Какая команда используются для запуска docker контейнера на основе образа? Как задать при этом перенаправление портов и volumes?

7) Какая команда используются для вывода всех Docker контейнеров?

8) Какая команда используются для вывода всех Docker образов?

9) Какие команды используются для остановки и удаления docker контейнеров? Какая команда используется для удаления docker образов?

10) Что такое Docker Hub и зачем он нужен?

11) Какая команда используется для загрузки образа из Docker Hub?

12) Какая команда используется для отправки своего образа в Docker Hub?

13) Что такое docker-compose и какие задачи он решает?

14) Что такое Kubernetes и какие задачи он решает?

15) Что такое CI/CD и для чего он был придуман?

# Algorithms, Patterns, SOLID

1) Что такое сложность алгоритма и как она представляется?

2) Опишите структуру “Linked list”. В каких случаях используется и какую сложность имеет на поиск, вставку и удаление элементов?

3) Опишите структуру “Stack”. В каких случаях используется и какую сложность имеет на добавление и удаление элементов?

4) Опишите структуру “Queue”. В каких случаях используется и какую сложность имеет на добавление и удаление элементов?

5) Опишите структуру “Deque”. В каких случаях используется и какую сложность имеет на добавление и удаление элементов?

6) Опишите структуру “Binary search tree”. В каких случаях используется и какую сложность имеет на поиск элементов?

7) Чем отличается Depth-First Search и Breadth-First Search?

8) Опишите структуру “Graph”. Что такое матрица смежности? Что такое списки смежности?

9) Опишите алгоритм Дейкстры. Какую задачу он решает?

10) Опишите структуры “Map” и “Set” в JavaScript. Какие задачи они решают и как с ними работать?

11) Что такое паттерны проектирования, зачем они нужны и на какие 3 группы они делятся?

12) Опишите паттерны Module и Singleton. В каких ситуациях они используются?

13) Опишите паттерны Adapter и Decorator. Чем они различаются?

14) Опишите паттерны Strategy и Command. В каких ситуациях они используются?

15) Опишите паттерны Observer и Chain of responsibility. В каких ситуациях они используются?

16) Что такое SOLID и как он расшифровывается?

17) В чем заключается принцип единственной ответственности?

18) В чем заключается принцип открытости/закрытости?

19) В чем заключается принцип подстановки Лисков?

20) В чем заключается принцип разделения интерфейса?

21) В чем заключается принцип инверсии зависимостей?