INTELLECTHUB

Курс по подготовке к техническому собеседованию

Вопрос:

Какие типы данных присутствуют в JavaScript?

INTELLECTHUB

Ответы на: Какие типы данных присутствуют в JavaScript?

Плохой ответ: Существуют string, number, boolean

Хороший ответ: Существует 7 типов данных: number, string, boolean, symbol, null, undefined, object

Отличный ответ: Существует 8 типов данных. 7 примитивных и 1 сложный тип данных. Примитивные типы: number, string, boolean, symbol, bigInt, null, undefined. Сложный тип - object. Он используется для коллекций данных и для объявления более сложных сущностей. Также он передается по ссылке, а простые типы данных по значению

Вопрос:

Что такое NaN?

Ответы на: Что такое NaN?

Плохой ответ: NaN - не число.

Отличный ответ: NaN (not a number) - это значение получаемое в результате выполнения числовой операции над не числовым значением. Его тип данных number. NaN не равен самому себе. Проверку на NaN можно сделать с помощью Number.isNaN

Вопрос:

В чем разница между null и undefined?

Ответы на: В чем разница между null и undefined?

Плохой ответ: undefined - неопределенное значение, если мы объявили переменную но не задали ей значение. null - это пустое значение.

Отличный ответ: undefined (неопределенный) представляет собой значение по умолчанию: 1 - переменной, которая объявлена без инициализации; 2 - функции, которая ничего не возвращает явно; 3 - несуществующего свойства объекта. null - это значение «отсутствия значения».
Присваивается переменной явно.

Вопрос:

Чем отличается строгое и не строгое равенство (=== и ==)?

Ответы на: Чем отличается строгое и не строгое равенство (=== и ==)?

Плохой ответ: Строгое сравнение более точное.

Отличный ответ: Строгое сравнивает значения по типу данных и значению. Не строгое приводит значения к одному типу а потом эти значения сравнивает.

Вопрос:

Почему результатом сравнения двух похожих объектов является false?

Ответы на: Почему результатом сравнения двух похожих объектов является false?

Плохой ответ: Потому что объекты - это сложные типы данных.

Отличный ответ: Потому что объекты - это ссылочные типы данных. 2 одинаковы с виду объекта имеют разные ссылки, и по этому операция сравнения будет возвращать false.

Вопрос:

Как проверить 2 объекта на идентичность?

Ответы на: Как проверить 2 объекта на идентичность?

Плохой ответ: Использовать JSON.stringify для приведения объектов в строку и сравнить их уже как 2 строки.

Отличный ответ: 1 - использовать JSON.stringify для приведения объектов в строку и сравнить их уже как 2 строки. Такой подход имеет ограничения. Если в объекте будут методы или symbol - JSON.stringify не сможет конвертировать их в строку. 2 - написать или использовать из библиотеки функцию deepEqual которая будет проходить циклом по ключам двух объектов и будет проверять значение этих ключей. Функция должна работать рекурсивно.

Вопрос:

Как сделать копию объекта?

Ответы на: Как сделать копию объекта?

Плохой ответ: Использовать spread оператор.

Хороший ответ: Существует понятие глубокого и поверхностного копирования. Для поверхностного копирования можно использовать spread оператор (...). Для глубокого - подход JSON.stringify - JSON.parse для конвертирования объекта в строку и потом обратно.

Отличный ответ: Существует понятие глубокого и поверхностного копирования. Глубокое - это копирование объекта и все его уровней вложенности. Поверхностное - это копирование только первого уровня вложенности, а для остального копируется ссылка. Способы поверхностного копирования: 1 - использовать spread оператор (...) 2 - использовать Object.assign() Способы глубокого копирования: 1 - использовать JSON.stringify - JSON.parse для конвертирования объекта в строку и потом обратно. Имеет ограничения по копированию методов и symbols 2 - написать или использовать из библиотеки функцию deepClone которая будет рекурсивно проходить по ключам объекта и копировать их в новый объект

Вопрос:

Чем отличаются переменные var, let и const?

Ответы на: Чем отличаются переменные var, let и const?

Плохой ответ: var - старый способ создания переменных. let и const новый способ создания переменных

Отличный ответ: var - переменную можно инициализировать после объявления, можно изменять, имеет функциональную область видимости, имеет hoisting. Сейчас почти не используют при написании нового кода. let - переменную можно инициализировать после объявления, можно изменять, имеет блочную область видимости, не имеет hoisting. const - переменную обязательно нужно инициализировать во-время объявления, нельзя изменять, имеет блочную область видимости, не имеет hoisting

Вопрос:

Как узнать является ли объект массивом?

Ответ на: Как узнать является ли объект массивом?

Для этого можно использовать метод Array.isArray.

Вопрос:

Какие перебирающие методы массивов вы знаете?

Ответы на: Какие перебирающие методы массивов вы знаете?

Плохой ответ: Назвать не все методы. К примеру только forEach и map.

Отличный ответ: Есть следующие методы:
- forEach – для перебора массива.
- filter – для фильтрации массива.
- every/some – для проверки массива.
- map – для трансформации массива в массив.
- reduce/reduceRight – для прохода по массиву с вычислением значения.
- sort - сортирует массив

Вопрос:

Как объединить массивы?

Ответы на: Как объединить массивы?

Хороший ответ: Можно использоваться spread оператор и распылить эти массив в один общий массив.

Отличный ответ: 1 - Можно использовать метод concat(). 2 - Можно использовать spread оператор и распылить эти массив в один общий массив

Вопрос:

Как узнать находится ли элемент в массиве?

Ответы на: Как узнать находится ли элемент в массиве?

Хороший ответ: Использовать метод includes, который возвращает true если елемент находиться в массиве.

Отличный ответ: 1 - Использовать метод includes, который возвращает true если елемент находиться в массиве. 2 - Использовать метод indexOf который возвращает индекс найденного элемента в массиве или -1 если элемента в массиве нет. 3 - также можно использовать метод find, который возвращает найденный элемент, или возвращает undefined.

Вопрос:

Что такое поднятие (hoisting) ?

Ответы на: Что такое поднятие (hoisting) ?

Плохой ответ: Это поднятие переменных var и function declaration вверх.

Отличный ответ: Это механизм в JavaScript, в котором переменные типу var и объявления функций, передвигаются вверх своей области видимости перед тем, как код будет выполнен.

INTELLECTHUB

Вопрос:

Каким будет значение переменной var при обращении к ней до ее объявления?

Ответы на: Каким будет значение переменной var при обращении к ней до ее объявления?

Значением переменной будет undefined.

Вопрос:

Что будет если обратится к переменной let/const до её объявления?

Ответы на: Что будет если обратится к переменной let/const до её объявления?

Будет ошибка типа ReferenceError. Переменной еще не существует.

Вопрос:

Что такое область видимости (Scope)?

INTELLECTHUB

Ответы на: Что такое область видимости (Scope)?

Плохой ответ: Это область доступности переменной. Есть глобальная и локальная область видимости.

Хороший ответ: Это область доступности переменной или функции. Есть глобальная, функциональная и блочная область видимости.

Отличный ответ: Это место, где (или откуда) мы имеем доступ к переменным или функциям. JS имеем 4 типа областей видимости: глобальная, функциональная, блочная (ES6) и область видимости eval.

Вопрос:

Чем Function Declaration отличается от Function Expression?

Ответы на: Чем Function Declaration отличается от Function Expression?

Плохой ответ: Function Expression это присвоение функции в переменную. Function Declaration это просто создание функции.

Отличный ответ: Function Expression создаётся, когда выполнение доходит до него, и затем уже может использоваться. Function Declaration можно использовать во всем скрипте (или блоке кода, если функция объявлена в блоке).

Вопрос:

Чем стрелочная функция отличается от обычной?

Ответы на: Чем стрелочная функция отличается от обычной?

Плохой ответ: Отличается синтаксисом

Хороший ответ: Стрелочная функция не имеет собственного this. Она берет его по месту создания. В обычной функции this устанавливается во время вызова. Стрелочная функция имеет короткий синтаксис не явного возврата значения.

Отличный ответ: Стрелочная функция не имеет собственного this. Она берет его по месту создания. В обычной функции this устанавливается во время вызова. Стрелочная функция не имеет arguments. Стрелочная функция не имеет prototype. Стрелочная функция имеет короткий синтаксис не явного возврата значения.

Вопрос:

Есть ли аналог arguments для стрелочной функции?

Ответы на: Есть ли аналог arguments для стрелочной функции?

Аналога нет. Но можно использовать rest оператор (...) для того, чтобы собрать все параметры с которыми вызвана функция в массив.

Вопрос:

Что такое лексическое окружение (Lexical Environment)?

Ответы на: Что такое лексическое окружение (Lexical Environment)?

Плохой ответ: Это объект в котором хранятся переменные созданные внутри функции.

Отличный ответ: Это свойства внутреннего объекта функции который создается во время ее вызова. Туда записываются аргументы, функции и переменные. Также там находится ссылка на внешнее лексическое окружение.

Вопрос:

Что является глобальным лексическим окружение?

INTELLECTHUB

Ответы на: Что является глобальным лексическим окружение?

Плохой ответ: window

Отличный ответ: В не строгом режиме это window. В строгом - undefined.

Вопрос:

Что такое замыкание (Closures)?

Ответы на: Что такое замыкание (Closures)?

Плохой ответ: Это функция вместе со всеми внешними переменными, которые ей доступны.

Отличный ответ: Это способность функции во время создания запоминать ссылки на переменные, функции и параметры, находящиеся в текущем лексическом окружении, а также в лексическом окружении родительской функции и так до глобального лексического окружения.
Замыкание подразумевает именно внешние переменные, а не саму функцию.

Вопрос:

Для чего используют замыкания?

Ответы на: Для чего используют замыкания?

1 - Часто для создания приватных переменных и функций (инкапсуляция).
2 - Для сохранения промежуточных параметров вызова функции (каррирование).

Вопрос:

Что такое IIFE?

Ответы на: Что такое IIFE?

Immediately Invoked Function Expression - это функция, которая вызывается или выполняется сразу же после создания или объявления.

Вопрос:

Что такое this?

Ответы на: Что такое this?

Плохой ответ: Это контекст вызова функции.

Отличный ответ: Это ссылка на контекст вызова функции. Контекстом является объект который в данный момент выполняет или вызывает функцию. Для стрелочной функции - это объект в котором она создана, а в обычной функции - которым она вызвана.
Контекстом может быть:
1 - this в объекте - указывает на сам объект
2 - this в классе - указывает на экземпляр класса
3 - глобальным контекстом является window (или undefined в режиме use strict)

Вопрос:

Как можно подменить контекст вызова функции?

INTELLECTHUB

Ответы на: Как можно подменить контекст вызова функции?

Плохой ответ: Можно использовать метод bind.

Отличный ответ: Есть 3 метода: call, apply, bind. call и apply вызывают функцию из заданным контекстом. bind возвращает новую функцию с уже навсегда привязанным контекстом.

Вопрос:

Можно ли изменить контекст функции которую возвратил метод bind?

Ответы на: Можно ли изменить контекст функции которую возвратил метод bind?

Нет, bind привязывает контекст навсегда.

Вопрос:

Можно ли подменить контекст вызова стрелочной функции?

Ответы на: Можно ли подменить контекст вызова стрелочной функции?

Плохой ответ: Нет, нельзя.

Отличный ответ: Нет. Стрелочная функция не имеет методов call, apply, bind а так же своих аналогов. Так же она использует контекст в котором создана а не контекст в котором вызвана.

Вопрос:

Что такое прототип объекта?

Ответы на: Что такое прототип объекта?

Плохой ответ: Это механизм, через который работает наследование в JavaScript.

Отличный ответ: Это шаблон объекта. Он используется как запасной вариант для свойств и методов, существующих в данном объекте. Это также один из способов обмена свойствами и функциональностью между объектами. Это основная концепция прототипного наследования в JS.

Вопрос:

Как работает прототипное наследование в JavaScript?

Ответы на: Как работает прототипное наследование в JavaScript?

Хороший ответ: Когда мы хотим прочитать свойство из объекта, а оно отсутствует - JavaScript попытается его прочитать из прототипа объекта.

Отличный ответ: Когда мы хотим прочитать свойство из объекта, а оно отсутствует - JavaScript попытается его прочитать из прототипа объекта. Если свойства нет в прототипе, JavaScript попытается его прочиться из прототипа прототипа, и т.д. до тех пор, пока свойство не будет найдено или цепочка прототипов не закончится. В таком случае JavaScript возвратит undefined.

Вопрос:

Как создать объект в котором не будет прототипа?

Ответы на: Как создать объект в котором не будет прототипа?

Плохой ответ: Создать обьект а потом в прототип записать значение null.

Отличный ответ: Использовать Object.create(). Данный метод принимает первым аргументом объект, который будет являться прототипом объекта, который он возвратит. Если мы вызовем Object.create() из аргументом null, будет создан объект без прототипа.

Вопрос:

Как проверить является ли свойство объекта личным свойством или это свойство прототипа?

Ответы на: Как проверить является ли свойство объекта личным свойством или это свойство прототипа?

Можно использовать метод hasOwnProperty который возвращает true или false в зависимости от того содержит ли объект указанное свойство в качестве собственного свойства или нет.

Вопрос:

Как запретить изменять объект?

Ответы на: Как запретить изменять объект?

Хороший ответ: Есть метод Object.freeze(), который "замораживает" объект от изменений.

Отличный ответ: Есть метод Object.freeze(), который "замораживает" объект от изменений. Данный метод работает только в одну сторону. Отменить действие данного метода уже невозможно. Есть метод Object.seal(), который запрещает добавлять новые свойства, но уже существующие свойства изменять можно. Так же есть метод Object.preventExtensions(), который запрещает добавлять новые свойства в объект.

Вопрос:

Что такое дескрипторы свойств объекта?

Ответы на: Что такое дескрипторы свойств объекта?

Плохой ответ: Дескриптор - это объект конфигурации свойства в объекте.

Отличный ответ: Дескриптор - это объект конфигурации свойства в объекте. Он имеет 4 свойства: value - значение свойства объекта. writable - указывает можно ли изменять значение данного свойства enumerable - указывает будет ли видно свойство во время перебора свойств объекта configurable - указывает можно ли добавлять или удалять свойства объекта, а так же можно ли изменять дескрипторы его свойств.

Вопрос:

Чем отличается функция конструктор и класс?

Ответы на: Чем отличается функция конструктор и класс?

Класс - это синтаксический сахар над функцией конструктором. При создании экземпляра класса, методы описанные в нем попадают в прототип, а методы описанные внутри функции конструктора попадут в сам экземпляр. Для того, чтобы методы описанные внутри функции конструктора попали в прототип их нужно отдельно туда добавить.

Вопрос:

Что нужно сделать, чтобы метод класса попал в его экземпляр?

Ответы на: Что нужно сделать, чтобы метод класса попал в его экземпляр?

Такой метод нужно описать внутри конструктора

Вопрос:

Присутствует ли в JavaScript множественное наследование?

Ответы на: Присутствует ли в JavaScript множественное наследование?

Плохой ответ: Нет.

Отличный ответ: Нет, потому что наследование в JavaScript базируется на прототипах, а у одного объекта может быть только один прототип.

Вопрос:

Что такое Promise?

Ответы на: Что такое Promise?

Плохой ответ: Это объект, который используют для ожидание выполнения асинхронной операции.

Отличный ответ: Это объект который используется для отложенных и асинхронных вычислений. Promise имеет 3 состояния:
- ожидание (pending): начальное состояние, не исполнен и не отклонен.
- исполнено (fulfilled): операция завершена успешно.
- отклонено (rejected): операция завершена с ошибкой.

Вопрос:

Для чего нужен метод Promise.all?

Ответы на: Для чего нужен метод Promise.all?

Плохой ответ: Этот метод ожидает исполнения массива промисов и возвращает результат их выполнения.

Отличный ответ: Ожидает исполнения всех промисов или отклонения любого из них. Возвращает промис, который исполниться после исполнения всех промисов. В случае, если любой из промисов будет отклонён, Promise.all будет также отклонён. У него есть аналог Promise.allSetled, который исполняется как-только все полученные промисы завершены (исполнены или отклонены), содержащий массив результатов исполнения полученных промисов.

Вопрос:

Для чего нужен метод Promise.race?

Ответы на: Для чего нужен метод Promise.race?

Плохой ответ: Ожидает исполнения первого промиса.

Отличный ответ: Ожидает исполнения или отклонения любого из полученных промисов. Возвращает промис, который будет исполнен или отклонён с результатом исполнения первого исполненного или отклонённого промиса.

Вопрос:

Для чего нужна async/await функция?

Ответы на: Для чего нужна async/await функция?

Плохой ответ: Это функция для работы с асинхронными действиями.

Отличный ответ: Ключевое слово async делает созданную функцию асинхронной. Функция async может содержать выражение await, которое приостанавливает выполнение функции async и ожидает ответа от переданного Promise, затем возобновляя выполнение функции async и возвращая полученное значение.

Вопрос:

Как обрабатывать ошибки в async/await функции?

INTELLECTHUB

Ответы на: Как обрабатывать ошибки в async/await функции?

1 - Для этого можно использовать конструкцию try...catch() 2 - вызвать метод .catch() после вызова функции

Вопрос:

Что такое event loop?

Ответы на: Что такое event loop?

Это механизм, который отвечает за выполнение кода, сбора и обработки событий и выполнения подзадач из очереди.
В концепции event loop есть несколько блоков:
- call stack - отвечает за создаёт контекст выполнения функции. Каждая вызываемая функция попадает в call stack.
- heap - это большая неструктурированная области памяти, в которой хранятся объявленные переменные, функции, и т.д.
- third party API - API, которые предоставляет окружение. К примеру, метод fecth, который предоставляется браузером.
- queue - список задач, подлежащих обработке. Каждая задача ассоциируется с некоторой функцией, которая будет вызвана, чтобы обработать эту задачу.

Ответ на: В чем разница между cookie, sessionStorage и localStorage?

localStorage:
- хранит данные бессрочно.
- очищается только с помощью JavaScript или очистки кэша браузера.
- хранит данные объёмом до 5 МБ.
- не поддерживается старыми браузерами, например, IE 7 и ниже.
- работает по правилу ограничения домена (same origin policy).

sessionStorage:
- хранит данные, пока продолжается текущая сессия вкладки.
- каждая вкладка имеет свой sessionStorage.
- хранит данные объёмом до 5 МБ.
- не поддерживается старыми браузерами, например, IE 7 и ниже.

cookie:
- хранит данные, которые передаются на сервер через заголовки.
- имеют срок хранения данных.
- объём данных от 4 Кбайт до 32 Кбайт.
- cookie могут быть защищёнными, в этом случае их содержимое нельзя получить на стороне клиента. Это важно для аутентификации при хранении пользовательских токенов.