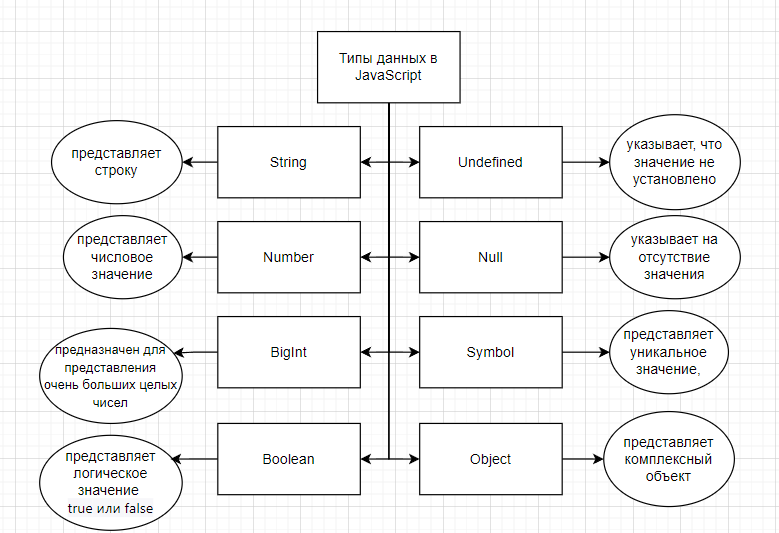
**Основы JavaScript:**

1. Каковы типы данных есть в JavaScript? Составить диаграмму.
2. В чем разница между let, const и var при объявлении переменных?

var - это старое ключевое слово JavaScript для объявления переменных. Оно имеет функциональную область видимости и подвергается всплытию (hoisting).

let и const - это новые ключевые слова, добавленные в стандарт ECMAScript 2015 (ES6), для объявления переменных. Они отличаются от var в следующем:

* Переменные, объявленные с помощью let, имеют блочную область видимости, тогда как переменные, объявленные с помощью var, имеют функциональную область видимости.
* let позволяет объявлять переменные, которые могут быть переназначены, тогда как const создает переменные с неизменными значениями.
* При объявлении переменной с помощью let или const, переменная не будет подвергаться всплытию (hoisting), в отличие от переменных, объявленных с помощью var.

1. Что такое hoisting?

Hoisting (всплытие) представляет процесс доступа к переменным до их определения. Возможно, данная концепция выглядит немного странно, но она связана с работой компилятора JavaScript. Компиляция кода происходит в два прохода. При первом проходе компилятор получает все объявления переменных, все идентификаторы. При этом никакой код не выполняется, методы не вызываются. При втором проходе собственно происходит выполнение. И даже если переменная определена после непосредственного использования, ошибки не возникнет, так как при первом проходе компилятору уже известны все переменные.

**Функции:**

1. Как объявить функцию в JavaScript?

Функции в JavaScript объявляются с помощью ключевого слова "function", после которого следует имя функции. Наименование функции подчиняется тем же правилам, что и наименование переменной: оно может содержать только цифры, буквы, символы подчеркивания и доллара ($) и должно начинаться с буквы, символа подчеркивания или доллара.

После имени функции в скобках идет перечисление параметров. Даже если параметров у функции нет, то просто идут пустые скобки. Затем в фигурных скобках идет тело функции, содержащее набор инструкций.

function имя\_функции(параметры){

// Инструкции

}

1. В чем разница между function declaration и function expression?

Function declaration объявляется до выполнения кода и доступна для использования в любом месте скрипта.

Function expression присваивает функцию переменной и является частью исполняемого кода. Function expression необходимо объявить до использования.

1. Что такое стрелочные функции?

Стрелочные функции (Arrow functions) — это сокращенный синтаксис для объявления функций в JavaScript. Они позволяют написать функцию более кратко и являются анонимными. Стрелочные функции имеют несколько отличий от обычных функций — они не имеют собственного контекста исполнения (this), не могут быть использованы в качестве конструкторов объектов и не имеют ключевого слова "arguments". Кроме того, если стрелочная функция содержит только одно выражение, то можно опустить фигурные скобки и оператор return.

Стрелочные функции определяются с помощью оператора =>, перед которым в скобках идут параметры функции, а после - собственно тело функции.

**Объекты и массивы:**

1. Как создать объект в JavaScript?

Первый способ заключается в использовании конструктора Object:

const user = new Object();

Второй способ создания объекта представляет использование фигурных скобок (литеральная нотация):

const user = {};

1. Как добавить новый элемент в конец массива?

Чтобы добавить новый элемент в конец массива JavaScript, вы можете использовать метод push().

1. Как перебрать все элементы массива?

Чтобы перебрать все элементы массива JavaScript, вы можете использовать цикл for или метод forEach(), правильное использование цикла for...in, цикл for...of (неявное использование итератора) или явное использование итератора.

**Обработка событий:**

1. Как назначить обработчик события элементу?

Назначение обработчика событий элементу можно сделать с помощью метода addEventListener() - принимает два параметра: название события без префикса on и функцию обработчика этого события, атрибута onclick, который определяет обработчик нажатия элемента.

1. Что такое event delegation?

Event delegation (делегирование событий) - это подход, при котором обработчик события назначается не непосредственно элементу, а его родителю. Таким образом, события, происходящие с потомками элемента, будут обрабатываться общим обработчиком. Это позволяет уменьшить количество обработчиков событий в коде и упростить его поддержку. Для этого используется метод closest(), который позволяет получить ближайший родительский элемент, соответствующий заданному селектору.

1. Как отменить стандартное поведение события?

Для отмены стандартного поведения события можно использовать методы preventDefault() и stopPropagation().

Метод preventDefault() позволяет отменить действие, которое обычно выполняется по умолчанию при возникновении события. Например, при нажатии на ссылку предотвращает переход по этой ссылке.

Метод stopPropagation() позволяет остановить распространение события по цепочке родительских элементов. Например, при клике на дочерний элемент останавливает выполнение обработчиков событий на родительских элементах.

**Асинхронность:**

1. Как работает синхронный и асинхронный код в JavaScript?

В JavaScript синхронный код выполняется последовательно от начала до конца, поэтому каждая операция блокирует выполнение следующей операции до тех пор, пока текущая операция не завершится. Это может приводить к задержкам и блокировкам в работе программы.

Асинхронный код в JavaScript позволяет выполнять операции параллельно, без блокировки других операций. Вместо ожидания завершения текущей операции, асинхронный код продолжает выполняться, а результат операции становится доступным позднее. Это позволяет создавать более отзывчивые и эффективные программы.

1. Что такое Promise?

Promise - это объект в JavaScript, представляющий результат выполнения асинхронной операции. При создании Promise передается функция, которая содержит асинхронный код. Promise может находиться в одном из трех состояний: ожидания (pending), выполнено успешно (fulfilled) или выполнено с ошибкой (rejected). При завершении операции Promise переходит в одно из выполненных состояний и возвращает результат операции или ошибку.

1. Как использовать async/await?

Оператор async определяет асинхронную функцию, в которой, как предполагается, будет выполняться одна или несколько асинхронных задач. Функция, объявленная с ключевым словом "async", всегда возвращает Promise.

Внутри асинхронной функции мы можем применить оператор await. Он ставится перед вызовом асинхронной функции или операции, которая представляет объект Promise, чтобы остановить выполнение функции, пока асинхронная операция не завершится и не вернет результат.