1. Что такое JavaScript и чем он отличается от других языков программирования?

Ответ:

JavaScript — это интерпретируемый язык программирования высокого уровня, используемый для создания динамического контента в веб-приложениях. Его основное отличие от других языков — тесная интеграция с браузерами и HTML/CSS, а также возможность работать как в клиентской, так и серверной среде (например, с помощью Node.js). Основные черты:

динамическая типизация (тип переменной определяется в процессе выполнения программы);

событийно-ориентированная модель;

мощная работа с DOM (Document Object Model).

2. Какие типы данных существуют в JavaScript?

Ответ:

Примитивные типы данных:

Number (числа, например: 42, -3.14),

String (строки, например: 'Hello'),

Boolean (логические значения: true или false),

Null (представляет намеренное отсутствие какого-либо значения),

Undefined (неопределенное значение, переменная объявлена, но не инициализирована),

Symbol (уникальный и неизменяемый идентификатор).

Объектные типы данных:

Object (например: объекты, массивы, функции).

3. Что такое Hoisting?

Ответ:

Hoisting — это механизм JavaScript, при котором объявления переменных и функций "поднимаются" вверх области видимости во время выполнения программы. Пример:

console.log(a); // undefined

var a = 5;

hoistedFunction();

function hoistedFunction() {

console.log("Это работает!");

}

4. Какая разница между var, let и const?

Ответ:

var\*\*: переменная имеет функциональную область видимости и поддерживает поднятие (hoisting). Возможно переобъявление внутри одного блока.

let\*\*: имеет блочную область видимости, не поднимается. Переменную можно переназначить, но нельзя переобъявить в одном блоке.

const\*\*: имеет блочную область видимости, значение константы нельзя переназначить или переобъявить.

5. Что такое объект и как его создать?

Ответ:

Объект — это структура данных, которая содержит пары "ключ-значение". Методы создания объектов:

Использование литерала:

const obj = { key: 'value', number: 42 };

С помощью конструктора:

const obj = new Object();

obj.key = 'value';

С помощью функции:

function CreateUser(name, age) {

this.name = name;

this.age = age;

}

const user = new CreateUser('John', 30);

6. Объясните разницу между == и ===.

Ответ:

== сравнивает значения, игнорируя типы данных. Например: '5' == 5 // true.

=== сравнивает и значения, и типы данных. Например: '5' === 5 // false.

7. Что такое замыкание (closure) в JavaScript?

Ответ:

Замыкание — это функция, которая имеет доступ к своей внешней области видимости, даже после завершения выполнения этой внешней функции. Это позволяет "запоминать" локальные переменные. Пример:

function outerFunction(outerVariable) {

return function innerFunction(innerVariable) {

console.log(Outer: ${outerVariable}, Inner:${innerVariable});

}

}

const newFunction = outerFunction('outside');

newFunction('inside'); // Outer: outside, Inner: inside

8. Какую роль выполняет слово this в JavaScript?

Ответ:

this — это ключевое слово, которое определяет контекст исполнения функции. Оно изменяется в зависимости от способа вызова функции:

В глобальной области: указывает на объект window (в браузере).

В методах объекта: указывает на сам объект.

В стрелочных функциях: наследуется из родительского контекста.

9. Что такое Promise и зачем он нужен?

Ответ:

Promise — это объект для работы с асинхронными операциями. Он представляет результат операции, которая еще не завершилась. У него три состояния:

pending (ожидание),

fulfilled (успешно выполнено),

rejected (отклонено).

Пример:

const promise = new Promise((resolve, reject) => {

if (true) resolve('Success');

else reject('Error');

});

promise

.then((res) => console.log(res))

.catch((err) => console.log(err));

10. Что такое async/await?

Ответ:

async/await — это высокая абстракция над промисами, которая делает код асинхронных операций похожим на синхронный:

async function fetchData() {

const response = await fetch('https://api.example.com');

const data = await response.json();

return data;

}

11. Что такое Event Loop?

Ответ:

Event Loop — это механизм обработки событий в JavaScript, позволяющий выполнять асинхронные задачи. Он управляет стеком вызовов и очередью задач.

12. Как работает метод .map()?

Ответ:

Метод .map() создает новый массив, вызывая функцию для каждого элемента текущего массива:

const numbers = [1, 2, 3];

const squared = numbers.map(num => num \* num);

console.log(squared); // [1, 4, 9]

13. Что такое прототипное наследование?

Ответ:

Прототипное наследование позволяет объектам наследовать свойства и методы от других объектов через связанный прототип ([[Prototype]]).

14. Какая разница между функциями и стрелочными функциями?

Ответ:

Стрелочные функции:

Не имеют своего значения this (зависит от контекста вызова).

Невозможно использовать как конструкторы (нет prototype).

15. Как завершить выполнение цикла досрочно?

Ответ:

С помощью break.

Пример:

for (let i = 0; i acc + num, 0);

console.log(sum); // 6

19. Что такое document.querySelector() и как он работает?

Ответ:

Метод document.querySelector() возвращает первый элемент, соответствующий селектору CSS:

const element = document.querySelector('.className');

20. Какая разница между null и undefined?

Ответ:

null — это намеренное отсутствие значения.

undefined означает, что значение переменной еще не присвоено.

1. Что такое строгий режим ("use strict") в JavaScript?

Ответ:

Строгий режим — это способ включить более строгую интерпретацию JavaScript. Например:

Запрещает использование необъявленных переменных.

Предотвращает дублирование параметров функций.

Включается добавлением строки "use strict"; в начало скрипта или функции.

2. Как в JavaScript объявляются функции?

Ответ:

Функции можно объявить тремя способами:

Function Declaration (объявление функции):

function myFunc() {

console.log("Hello!");

}

Function Expression (функциональное выражение):

const myFunc = function() {

console.log("Hello!");

};

Стрелочная функция:

const myFunc = () => console.log("Hello!");

3. Какая разница между ссылочными и примитивными типами данных?

Ответ:

Примитивные типы копируются по значению, тогда как массивы и объекты копируются по ссылке:

let a = 10;

let b = a;

b = 20; // a = 10

const obj1 = { key: "value" };

const obj2 = obj1;

obj2.key = "newValue"; // obj1.key изменился на "newValue"

4. Как создать и вызвать самовызывающуюся функцию (IIFE)?

Ответ:

Самовызывающаяся функция (IIFE — Immediately Invoked Function Expression) вызывается сразу после ее создания:

(function() {

console.log("I am an IIFE!");

})();

5. Как работает метод .call() и .apply()?

Ответ:

Оба метода позволяют выполнять функцию в контексте другого объекта:

function greet(greeting) {

console.log(${greeting},${this.name});

}

const person = { name: "John" };

greet.call(person, "Hello"); // Hello, John

greet.apply(person, ["Hi"]); // Hi, John

6. Для чего нужен метод .bind()?

Ответ:

Метод .bind() возвращает новую функцию с привязанным контекстом:

const person = { name: "John" };

function greet() {

console.log(this.name);

}

const boundGreet = greet.bind(person);

boundGreet(); // John

7. Что такое таймеры в JavaScript и как они работают?

Ответ:

С помощью setTimeout (задержка) и setInterval (циклический вызов) JavaScript позволяет выполнять код через заданное время:

setTimeout(() => console.log("Через 1 секунду"), 1000);

const interval = setInterval(() => console.log("Повторяем каждые 2 секунды"), 2000);

clearInterval(interval); // чтобы остановить интервал.

8. Что такое Декструктуризация (Destructuring) в JavaScript?

Ответ:

Деструктуризация позволяет извлекать значения из массивов и объектов:

const arr = [1, 2, 3];

const [first, second] = arr; // first = 1, second = 2

const obj = { a: 1, b: 2 };

const { a, b } = obj; // a = 1, b = 2

9. Что такое стрелочные функции и какова их особенность?

Ответ:

Стрелочные функции — это лаконичный способ записи функций. Они не имеют своего контекста this, а наследуют его из внешнего контекста:

const person = {

name: "John",

greet: function() {

setTimeout(() => {

console.log(this.name); // John

}, 1000);

}

};

10. Какие циклы существуют в JavaScript?

Ответ:

for: обычный цикл.

for...of: перебор элементов итерируемого объекта.

for...in: перебор ключей объекта.

while: выполняется, пока условие истинно.

do...while: сначала выполняет тело цикла, затем проверяет условие.

11. Что такое "цепочка методов" в JavaScript?

Ответ:

Цепочка методов — это возможность вызывать несколько методов подряд:

const result = [1, 2, 3]

.filter(num => num > 1)

.map(num => num \* 2)

.reduce((acc, num) => acc + num, 0);

console.log(result); // 8

12. Что такое оператор typeof и какие типы он возвращает?

Ответ:

typeof проверяет тип данных:

console.log(typeof 42); // number

console.log(typeof "hello"); // string

console.log(typeof null); // object (особенность языка)

13. Как проверить, является ли объект массивом?

Ответ:

С помощью метода Array.isArray(arr):

const arr = [1, 2, 3];

console.log(Array.isArray(arr)); // true

14. Как преобразовать строку в число (и наоборот)?

Ответ:

Из строки в число: Number(), parseInt(), parseFloat(), или с помощью унарного плюса (+).

Из числа в строку: .toString() или String().

15. Объясните работу оператора ?? (Nullish Coalescing Operator).

Ответ:

?? возвращает второй операнд, если первый равен null или undefined:

const result = null ?? "default";

console.log(result); // "default"

16. Как работает оператор ??=?

Ответ:

Этот оператор присваивает значение переменной только в случае, если её текущее значение null или undefined:

let value;

value ??= 10;

console.log(value); // 10

17. Какие есть способы клонирования объекта в JavaScript?

Ответ:

Поверхностное копирование:

const copy = { ...original };

Глубокое копирование:

const deepCopy = JSON.parse(JSON.stringify(original));

18. Как работает метод .toFixed() числа?

Ответ:

Возвращает строку с округленным числом до заданного количества знаков после запятой:

const num = 3.14159;

console.log(num.toFixed(2)); // "3.14"

19. Что такое строгая и слабая типизация?

Ответ:

JavaScript — язык со слабой типизацией, что означает автоматическое приведение типов. Например:

console.log('5' - 2); // 3

console.log('5' + 2); // "52"

20. Объясните, что такое объект window.

Ответ:

window — это глобальный объект в браузере, содержащий все глобальные функции, переменные, свойства и методы браузера.

21. Что такое объект navigator?

Ответ:

navigator предоставляет информацию о среде браузера — тип, язык, онлайн-статус, агент пользователя и т. д.

22. Как обработать ошибку JavaScript с помощью try..catch?

Ответ:

Обработчик try..catch ловит и обрабатывает ошибки выполнения:

try {

// Код с возможной ошибкой

throw new Error("Ошибка!");

} catch (err) {

console.log(err.message);

}

23. Какая разница между .innerHTML и .textContent?

Ответ:

innerHTML изменяет или читает разметку (включая HTML).

textContent работает только с текстом (без HTML).

24. Как работает метод .slice() для массивов?

Ответ:

Метод .slice() возвращает новый массив, начиная с указанного индекса:

const arr = [1, 2, 3, 4];

console.log(arr.slice(1, 3)); // [2, 3]

25. Что такое Mutable и Immutable?

Ответ:

Mutable:\*\* данные могут изменяться (например, массивы, объекты).

Immutable:\*\* данные неизменяемы (например, строки).

26. Как проверить, является ли объект пустым?

Ответ:

С помощью Object.keys():

console.log(Object.keys({}).length === 0); // true

27. Что такое event.preventDefault()?

Ответ:

Метод предотвращает поведение действия по умолчанию (например, отправку формы):

form.addEventListener('submit', e => e.preventDefault());

28. Как работают модули ES6?

Ответ:

Модули позволяют экспортировать и импортировать функционал между файлами:

export const x = 42;

import { x } from './module.js';

29. Как работает debugger в JavaScript?

Ответ:

debugger приостанавливает выполнение скрипта для отладки в консоли разработчика.

30. Как узнать размер массива?

Ответ:

Свойство .length возвращает длину массива:

const arr = [1, 2, 3];

console.log(arr.length); // 3

1. Что такое шаблонные строки (template strings) в JavaScript?

Ответ:

Шаблонные строки используют обратные кавычки (` `) и позволяют вставлять переменные и выражения с помощью $

{}`:

const name = "John";

console.log(`Привет,

${name}!`); // Привет, John!

2. Что такое короткие замыкания (short-circuiting) с использованием || и &&?

Ответ:

|| возвращает первый истинный операнд (или последний при отсутствии истинных операндов).

&& возвращает первый ложный операнд (или последний, если все истинны).

Пример:

const result1 = 0 || "fallback"; // "fallback"

const result2 = "value" && "next"; // "next"

3. Как работает цикл for...in?

Ответ:

for...in используется для перебора ключей объекта или индексов массива:

const obj = { a: 1, b: 2 };

for (let key in obj) {

console.log(key, obj[key]);

}

4. Как работает цикл for...of?

Ответ:

for...of используется для перебора значений итерируемых объектов (например, массивов, строк):

const arr = [1, 2, 3];

for (let value of arr) {

console.log(value);

}

5. Что такое объект arguments и где он используется?

Ответ:

arguments — это массивоподобный объект, содержащий все аргументы, переданные функции.

Примечание: в стрелочных функциях arguments недоступен.

function sum() {

console.log(arguments);

}

sum(1, 2, 3); // { 0: 1, 1: 2, 2: 3 }

6. Какие события происходят в процессе загрузки страницы?

Ответ:

Основные этапы:

DOMContentLoaded: DOM готов, но ресурсы (картинки) еще могут загружаться.

load: страница и все ресурсы полностью загружены.

beforeunload/unload: срабатывают при закрытии окна.

7. Как работает .hasOwnProperty()?

Ответ:

Метод проверяет, принадлежит ли свойство непосредственно объекту (а не унаследовано из прототипа):

const obj = { key: "value" };

console.log(obj.hasOwnProperty("key")); // true

console.log(obj.hasOwnProperty("toString")); // false

8. Что такое глубокое и неглубокое копирование объекта?

Ответ:

Неглубокое копирование (shallow copy)\*\*: создается новая оболочка, но вложенные объекты сохраняют ссылки на оригинал. Пример: { ...obj }.

Глубокое копирование (deep copy)\*\*: создается полная независимая копия объекта, включая вложенные объекты. Пример:

const deepCopy = JSON.parse(JSON.stringify(obj));

9. Объясните работу функции .split() и .join() для строк.

Ответ:

.split(delimiter) разбивает строку на массив с использованием разделителя:

const str = "1,2,3";

console.log(str.split(",")); // ["1", "2", "3"]

.join(delimiter) соединяет элементы массива в строку с использованием разделителя:

const arr = ["a", "b", "c"];

console.log(arr.join("-")); // "a-b-c"

10. Что такое event.target и event.currentTarget?

Ответ:

event.target\*\*: элемент, на котором произошло событие.

event.currentTarget\*\*: элемент, на котором срабатывает обработчик события.

11. Как преобразовать NodeList в массив?

Ответ:

C помощью оператора распыления ... или метода Array.from():

const nodeList = document.querySelectorAll("div");

const arr = [...nodeList];

12. Как узнать, поддерживает ли браузер определенную функцию?

Ответ:

Используйте оператор in или проверьте, существует ли объект/метод:

if ('geolocation' in navigator) {

console.log("Geolocation поддерживается!");

}

13. Какие методы используются для работы с локальным хранилищем (localStorage)?

Ответ:

localStorage.setItem("key", "value"); // Сохранить значение

localStorage.getItem("key"); // Получить значение

localStorage.removeItem("key"); // Удалить значение

localStorage.clear(); // Очистить хранилище

14. Что такое debounce и для чего он нужен?

Ответ:

Debounce откладывает выполнение функции до тех пор, пока события не перестанут происходить. Полезно для оптимизации.

function debounce(fn, delay) {

let timeout;

return function(...args) {

clearTimeout(timeout);

timeout = setTimeout(() => fn.apply(this, args), delay);

};

}

15. Что такое throttling?

Ответ:

Throttling ограничивает частоту выполнения функции в определенный интервал времени:

function throttle(fn, limit) {

let lastTime = 0;

return function(...args) {

const now = Date.now();

if (now - lastTime >= limit) {

lastTime = now;

fn.apply(this, args);

}

};

}

16. Какая разница между undefined и not defined?

Ответ:

undefined: переменная объявлена, но значение не присвоено.

not defined: переменная вообще не объявлена.

17. Как объявить массив с фиксированной длиной в JavaScript?

Ответ:

Используйте Array():

const arr = new Array(5); // массив длиной 5, заполнен undefined

18. Как отменить событие по умолчанию?

Ответ:

Используйте event.preventDefault():

link.addEventListener('click', (e) => e.preventDefault());

19. Что такое eval() и почему его использование не рекомендуется?

Ответ:

eval() выполняет строку как JavaScript-код. Это опасно, так как снижает производительность и может вызвать уязвимости безопасности.

20. Как работает оператор delete для объектов?

Ответ:

Оператор delete удаляет свойство объекта:

const obj = { key: "value" };

delete obj.key;

console.log(obj); // {}

21. Какие состояния Promise существуют?

Ответ:

pending — ожидание.

fulfilled — завершено успешно.

rejected — завершено с ошибкой.

22. В чем разница между синхронным и асинхронным кодом?

Ответ:

Синхронный код выполняется последовательно. Асинхронный код позволяет запускать операции без блокировки выполнения других операций, используя коллбеки, Promise, async/await.

23. Как работает метод .catch() у Promise?

Ответ:

.catch() обрабатывает исключения и ошибки, возникающие в цепочке Promise:

fetch("https://api.example.com")

.then(response => response.json())

.catch(error => console.error(error));

24. Что такое fetch()?

Ответ:

Метод для выполнения HTTP-запросов. Пример:

fetch('https://jsonplaceholder.typicode.com/posts')

.then(response => response.json())

.then(data => console.log(data));

25. Что такое DOM?

Ответ:

DOM (Document Object Model) — это интерфейс для работы с HTML-документом как с деревом объектов.

26. Как добавить элемент в DOM?

Ответ:

Используйте методы appendChild, append или innerHTML:

const div = document.createElement('div');

div.textContent = "Привет!";

document.body.appendChild(div);

27. Как удалить элемент из DOM?

Ответ:

Используйте метод remove() или removeChild():

const element = document.querySelector('div');

element.remove();

28. Как получить общее количество элементов на странице?

Ответ:

Используйте document.querySelectorAll('\*').length.

29. Что такое bubbling и capturing в событиях?

Ответ:

Bubbling (всплытие):\*\* событие проходит от дочернего элемента к родителю.

Capturing (погружение):\*\* событие идет от родителя к дочернему элементу.

30. Какую роль играет атрибут async в ``?

Ответ:

Скрипт загружается асинхронно, не блокируя загрузку HTML-документа, и выполняется по мере готовности.