ВОПРОСЫ ПО JAVASCRIPT

1 – Типы данных в JS

1.1 Назовите типы данных?

В JS выделяют 8 типов данных

*Первая подгруппа "примитивные":

- Number
- String
- Boolean - true/false
- Null
- Undefined
- Symbol - уникальный идентификатор, чаще его заменяет id или index
- Bigint - предоставляет возможность работать с целыми числами произвольной длины

*Вторая подгруппа - object

1.1 Чем отличается null и undefined?

undefined мы можем увидеть когда:
- пытаемся получить доступ к несуществующему свойству объекта,
- в функции, которая ничего не возвращает,
- в переменной, которой не было присвоено значения
null — означает отсутствия значения и его можно со временем переопределить

2. Операторы в JS

2.1 Назовите отличие = == ===
= присвоение
примеры:

```
let name = 'Anna';
const someFunction = () => {
  console.log(name);
};

someFunction()
```

== нестрогое равенство, проверяет на равенство
=== строгое равенство, проверяет на идентичность
примеры:
1 === '1' false
1 == '1' true

Во избежание ошибок, связанных с некорректным типом данных, рекомендуется использовать строгое неравенство ===

2.2 что такое унарный +?

Знак + используется в двух формах: бинарной и унарной.
бинарный - складывает значения, конкатенация в случае строк

```
let a = 2;
let b = 3;
console.log(a + b); // 5
```

```
let a = "2";
let b = "3":
console.log(a + b); // 23
```

унарный используется для преобразования строки в число

```
let a = "2";
let b = "3":
console.log(+a + +b); // 5
```

3. В чем разница между var, let и const?

var - такие переменные, являются глобальными, они могут доступны из любого места в коде

Данный способ объявления переменный устарел, и использовать его необходимо аккуратно или отказаться совсем,
возможны ошибки в коде из-за конфликтов переменных.

let - переменная, значение, которой может меняться
const - не изменяется, можно менять значение свойства объекта, объявленного с помощью const, но не само свойство!

4. Что такое this?

this - используется для получения доступа к информации внутри объекта

пример:

```
const user = {
   firstname: "Brad",
   lastname: "Pitt"
}


function Hello () {
   alert("Welcome" + user.firstname)
}
```

но мы можем использовать функцию и как метод

```
function Hello() {
  alert("Welcome" + this.firstname)
}
```

Чаще рекомендуется исключить this из кода, чтобы избежать все возможные проблемы, с потерей доступа к информации
- чем сложнее наш код, тем больше в этом вероятность.

5. Что такое event.target?

event.target — это элемент, в котором произошло событие
из него мы можем получить определенные данные
например, чтобы получение значение введеное пользователем в form
event.target.value
определить кликнутый элемент,
например, чтобы закрыть модальное окно по клику на overlay
event.target.classList.contains('overlay')

6. Функции

6.1 Что такое Callback Function?

Callback Function или функция обратного вызова - это функция, вызов которой отложен на будущее, при наступление какого-то события,
или при выполние кода другой функцией

Пример 1 - вызов функции по клику на кнопку:

```
const btn = document.querySelector('button');

btn.addEventListener('click', function()){
   console.log('Clicked button');
}
```

пример 2 - показать данные, полученные из API

```
async function getIp() {
   const res = await fetch('link API');
   const result = await res.json();
   getInfo(result.city); //вызыв функции getInfo
}

 getIp();
```

6.2 Что такое Arrow Functions?

Arrow Functions или стрелочные функции - способ создания функций в JS.

Стрелочные функции создаются быстрее и имеют более читаемый синтаксис, чем функциональные выражения.
В стрелочных функциях опускается слово function

Отличия:
1) нет необходимости прописывать return для возврата значения
примеры

```
let sum = (a, b) => a + b;

let sum = function(a, b) {
```

```
  return a + b;
};
```

2) Если мы передаем один параметр, его можно не оборачивать в круглые скобки
пример 1:

```
btn.addEventListener('click', e => {
   e.preventDefault();
}

btn.addEventListener('click', function(e)){
   e.preventDefault();
}
```

пример 2

```
array.forEach(element => {
   console.log(element);
}); // опускаем скобки в (element) =>, так как парамет всего 1

array.forEach((element, index) => {
   console.log(element);
   console.log(index);
}); // необходимы скобки, так как параметра уже 2 - element и index
```

3) стрелочная функция неименная, поэтому ее необходимо присвоить к переменной, такой способ объявления функции называется Function Expression

6.3 Function Expression и Function Declaration - в чем отличия?

примеры:

```
function sum(a, b) {
  return a + b;
} // Function Declaration


const sum = function(a, b) {
```

```
  return a + b;
```
} // Function Expression - объявление функции в контексте какого-либо выражения, например присваивания

Популярность Function Expression тесно связан с тем, что на смену приходят стрелочные функции из-за удобства, но которые не могут быть именные.

Главное отличие, в том, что в Function Expression функция должна быть описана, до ее вызова, в Function Declaration функция может быть вызыван и до и после вызова

6.4 Что такое область видимости функции?

Переменные, функции и параметры, объявленные внутри функции, доступны только внутри этой функции.
Данные можно передавать с помощью аргументов

6.5 Что такое аргумерт и параметр функции?

Аргумент – это значение, которое передаётся функции при её вызове.
Параметр – это переменная, указанная в круглых скобках в объявлении функции.

пример:

```
async function getIp() {
   const res = await fetch('link API');
   const result = await res.json();
   getInfo(result.city);
}

 getIp();

//result.city - аргумент функции

async function getInfo(data) {
   const res = await fetch(`${api.endpoint}weather?q=${data}&units=metric&lang=ru&appID=${api.key}`);
   const result = await res.json();
```

```
    displayResult(result);
}
```

data - парамет функции
result - аргумент функции

6.6 Что такое замыкание?

Closures или замыкание - способность функции во время создания запоминать ссылки на переменные и параметры

пример 1:
В данном примере, глобальная область видимости является частью замыкания.

```
let username = "Anna"

function hello() {
    console.log(`Hello ${username}`);
}

hello();
```

пример 2:
displayHello является частью замыкания

```
function user() {
    let username = 'Anna';
    return function displayHello() {
        console.log(`Hello ${username}`);
    };
}

let hello = user();
hello();
```

6.7 Что такое рекурсия?
метод, который предполагает создание функции, которая вызывает саму себя до тех пор пока программа не достигнет необходимого результата.

пример

```
let i = 0;
function counter() {
   days++;
   console.log(days);
   counter();
}
   counter();
```

7. Что изменилось в ES6?

Из основного:
1) Стрелочные функции (Arrow Functions) -
2) Классы (Classes) - больше можно узнать здесь https://learn.javascript.ru/class?ysclid=ldph8xwtej912459943
(из-за сложности не нашло популярности, примеры вы сможете увидеть в React классовых компонентах)
3) Шаблонные строки (Template Strings) - `Hello, ${username}!`
4) Деструктуризация (Object Destructuring).

```
let [firstName, lastName] = ["Jone", "Smit"];
alert(firstName);
alert(lastName);
```

5) Операторы rest и spread.
spread передаем данные массива на другие данные,
rest — получаем все параметры функции и помещаем их в массив
6) Блочная область видимости (ключевые слова «let» и «const»).

8. Что такое use strict?

Строгий режим по написанию кода.

Строгий режим делает следующее:

- Выбрасывает ошибки, когда в коде используются некоторые небезопасные конструкции.
- Выключает функции языка, которые запутывают код и потому не должны использоваться.
- Предотвращает использование слов, которые могут быть использованы в качестве ключевых в будущем.

9. Что такое промисы?

Promises - работа с асинхронным кодом в JS.
Они возвращают результат асинхронной функции, что позволяет вызывать функции, после того, как получент ответ от сервера - работа с API

Состояния промиса:
Ожидание pending — начальное состояние промиса. Результата еще неизвестен
Выполнено fulfilled — асинхронная операция выполнена, имеется результат. статус ответа 200
Отклонено rejected— асинхронная операция не выполнена, имеется причина. вернет статус ошибки

10. Что такое async/await?

один из способов способ написания асинхронного кода в JS
1 способ – использования then

```
function getApi(){
   return fetch('endpoint')
   .then(response => response.json())
   .then(data => {
      обработка полученных данных
   }).catch(err => {
      обработчик ошибок статус rejected
   })
}
```

2 способ – использование await

```
async function getApi(){
```

```
  try{
    const resp = await fetch('endpoint')
    const data = await res.json()
    обработка полученных данных
  } catch€{
    обработчик ошибок статус rejected
  }
}
```

11. В чем разница между методами event.preventDefault() и event.stopPropagation()?

Метод event.preventDefault() отключает поведение элемента по умолчанию
Например, перезагрузка страницы по клику на кнопку submit формы
Метод event.stopPropagation() отключает распространение события выше по DOM-дереву от ребенка к родителю

12. В чем отличие setTimeout и setInterval?

setTimeout - позволяет вызывать функцию один раз через определённый интервал времени, то есть отложить вызов функции
setInterval - позволяет вызывать функцию регулярно, повторяя вызов через определённый интервал времени.
чтобы остановить setInterval необходимо вызвать
clearInterval(timerId).
let timerId = setInterval(Countdown, 1000);
setTimeout также можно отменить clearTimeout(timerId);

13. Массивы

13.1 Как очистить массив?
let arr = [1, 2, 3];

1 способ - присвоить пустой массив
arr = [];

2 способ - обнулить длину масссива
arr.length = 0;

3 способ - метод splice()
arr.splice(0, arr.length);

4 способ - метод pop() + for()
```
for (let i = 0; i < arr.length; i++) {
  arr.pop();
}
```

13.2 Назвоите методы работы с массивами?

push() – добавляет элементы в конец
pop() – удаляет элемент из конца
shift() – удаляет элемент из начала
unshift() – добавляет элементы в начало
splice() - может удалять и добавлять элемент в массив
slice() - возвразает новый массив, копируя все числа в пределах указаных индексов
concat() - возвращает новый массив, копируя данные из других массивов

forEach() - вызывает функцию для каждого элемента массива
map() - вызывает функцию для каждого элемента массива и возвращает массив результатов выполнения этой функции.
filter() - возвращает новый массив соответсвующий условию
find() - возвращает первый элемент, отвечающий определенным условиям

includes() - возвращает true/false если/нет элемент есть в массиве
some() - возвращает true/false если элемент есть в массиве

indexOf() - возвращает номер индекса, или -1

sort() - сортировка массива
split() - разбивает строку на массив по заданному разделителю
join() - создаёт строку из элементов массива
reduce() - Он создаёт строку из элементов arr

13.3 Как проверить является ли значение массивом?

Для проверки используем isArray()
Примеры:

console.log(arr.isArray(5)) // false
console.log(arr.isArray('')) // false
console.log(arr.isArray([])) // true

14. Что такое JSON?

Формат для представления данных
JSON.stringify для преобразования объектов в JSON.
JSON.parse для преобразования JSON обратно в объект.

15. Что такое Date?

Объект Date - содержит дату и время, а также методы управления ими
его можно использовать для хранения времени создания/изменения, для измерения времени или просто для вывода текущей даты.
создать переменную let now = new Date();

Методы
getFullYear() - полный год
getYear() - устарел, не используем, чтобы отобразить не 2023, а 23, используем substring(2)
getMonth() - месяц, от 0 до 11.
getDate() - день месяца, от 1 до 31,
getHours(), getMinutes(), getSeconds(), getMilliseconds() - часы, минуты, секунды или миллисекунды.
getDay() - день недели от 0 (воскресенье) до 6 (суббота).