**Вопрос 16: Объект текстовое поле**

**Теория:**  
В JavaScript текстовое поле обычно связано с HTML-элементом <input type="text"> или <textarea>. Объект текстового поля в DOM предоставляет свойства и методы для работы с ним, такие как value (для получения или установки значения), placeholder, disabled, focus(), blur() и другие. Это позволяет динамически управлять содержимым поля ввода.

**Пример кода:**

<input type="text" id="myTextField" placeholder="Введите текст">

<button onclick="getText()">Получить текст</button>

<button onclick="setText()">Установить текст</button>

<script>

    // Получение объекта текстового поля

    const textField = document.getElementById("myTextField");

    // Функция для получения текста

    function getText() {

        alert("Введённый текст: " + textField.value);

    }

    // Функция для установки текста

    function setText() {

        textField.value = "Новый текст";

    }

</script>

В этом примере мы можем получить или изменить текст в поле ввода с помощью свойства value.

### Вопрос 17: Объект флажок

**Теория:**  
Объект флажок (checkbox) в JavaScript связан с элементом <input type="checkbox">. Основное свойство — checked, которое возвращает true, если флажок отмечен, и false, если нет. Также можно использовать value для получения значения флажка и методы вроде addEventListener для отслеживания изменений состояния.

**Пример кода:**

<input type="checkbox" id="myCheckbox"> Согласен с условиями

<button onclick="checkStatus()">Проверить статус</button>

<script>

    const checkbox = document.getElementById("myCheckbox");

    // Проверка состояния флажка

    function checkStatus() {

        if (checkbox.checked) {

            alert("Флажок отмечен!");

        } else {

            alert("Флажок не отмечен.");

        }

    }

    // Пример обработки события изменения

    checkbox.addEventListener("change", function() {

        console.log("Состояние флажка изменилось: " + checkbox.checked);

    });

</script>

Здесь мы проверяем, отмечен ли флажок, и добавляем обработчик события change.

### Вопрос 18: Объект переключатель

**Теория:**  
Объект переключатель (radio button) связан с элементом <input type="radio">. Переключатели обычно используются в группах, где можно выбрать только один вариант. Все переключатели в группе имеют одинаковое значение атрибута name. Свойство checked используется для определения, выбран ли конкретный переключатель.

**Пример кода:**

<input type="radio" name="color" id="red" value="red"> Красный

<input type="radio" name="color" id="blue" value="blue"> Синий

<button onclick="getSelectedColor()">Получить выбор</button>

<script>

    function getSelectedColor() {

        const radios = document.getElementsByName("color");

        for (let radio of radios) {

            if (radio.checked) {

                alert("Выбранный цвет: " + radio.value);

                break;

            }

        }

    }

</script>

В примере мы перебираем группу переключателей и определяем, какой из них выбран, используя свойство checked.

### Вопрос 19: Функции: способы создания и использования

**Теория:**  
В JavaScript функции можно создавать несколькими способами:

1. **Объявление функции** (function declaration): function имя() {}.
2. **Функциональное выражение** (function expression): const имя = function() {}.
3. **Стрелочные функции** (ES6): const имя = () => {}.  
   Функции могут принимать параметры, возвращать значения с помощью return и вызываться в любом месте кода после их определения (для объявленных функций — даже до).

**Пример кода:**

// 1. Объявление функции

function sayHello() {

    return "Привет!";

}

console.log(sayHello());

// 2. Функциональное выражение

const add = function(a, b) {

    return a + b;

};

console.log(add(3, 5)); // 8

// 3. Стрелочная функция

const multiply = (x, y) => x \* y;

console.log(multiply(4, 2)); // 8

// Пример с параметрами и вызовом

function greet(name) {

    return `Привет, ${name}!`;

}

console.log(greet("Алекс")); // Привет, Алекс!

Этот код демонстрирует разные способы создания функций и их использование.

**Вопрос 20: Передача функций дальше форм и элементов**

**Теория:**  
Передача функций "дальше" форм и элементов обычно означает использование функций в качестве обработчиков событий (например, onclick, onsubmit) или передачу их в качестве аргументов (callback-функции). Например, можно передать функцию для обработки данных формы или событий элемента.

**Пример кода:**

<form id="myForm">

    <input type="text" id="userInput" placeholder="Введите имя">

    <button type="submit">Отправить</button>

</form>

<script>

    // Функция-обработчик

    function handleSubmit(event) {

        event.preventDefault(); // Предотвращаем отправку формы

        const input = document.getElementById("userInput");

        alert("Вы ввели: " + input.value);

    }

    // Привязка функции к событию формы

    const form = document.getElementById("myForm");

    form.addEventListener("submit", handleSubmit);

    // Передача функции как callback

    function processData(data, callback) {

        const result = `Обработанные данные: ${data}`;

        callback(result);

    }

    processData("Тест", function(result) {

        console.log(result); // Обработанные данные: Тест

    });

</script>

В примере функция handleSubmit передаётся как обработчик события формы, а processData демонстрирует передачу функции как callback.