Подсказка (шпаргалка) по функциям в JavaScript

Функции в JavaScript:

- Фрагмент кода, который может выполнять определенные операции и возвращать результат
- Многократно переиспользуемый фрагмент кода, то есть их используют для группировки повторяющихся задач

Объявление функции:

```
function имяФункции(параметры) {
  //Тело функции
  //Код, который выполняется при вызове функции
  //Возвращаемое значение
}
```

У функции есть:

- Имя, которое позволяет обращаться к ним из других частей кода. Имя функции должно быть уникальным в контексте, где оно используется.
- Параметры, которые передаются внутрь функции при её вызове
- Тело, код внутри фигурных скобок, который задаёт операции, которые должны быть выполнены
- Возвращаемое значение, то, что функции могут возвращать (результат работы функции). Это позволяет использовать функции для вычислений и получения данных.

Вызов функции:

Функции вызываются путём написания имени функции с круглыми скобками:

```
имяФункции(парметры); //вызов функции с параметрами
имяФункции();         //вызов функции без параметров
```

Параметры и аргументы:

- Параметры - это переменные, указанные в определении функции.
- Аргументы - это фактические значения, переданные в функцию при её вызове.

```
function функцияСПараметрами(параметр1, параметр2) {
  // использование параметров
}

функцияСПараметрами(аргумент1, аргумент2); //вызов функции с аргументами
```

Параметры по умолчанию:

```
function функцияСПараметромПоУмолчанию(параметр = значение) {
  //использование параметра
}

функцияСПараметромПоУмолчанию(); //параметр примет значение по умолчанию
функцияСПараметромПоУмолчанию(аргумент); //параметр будет равен переданному аргументу
```

Имена параметров и переменных записываются в camelCase (первое слово маленькими буквами, каждое следующее слово с большой буквы):

```
function calculateOrderTotal(productsCost, shippingCost, discount) {
  // код функции
}
```

Возвращаемое значение:

- Функция может возвращать значение с помощью оператора `return`.
- Значение, возвращаемое функцией, может быть сохранено и использовано в коде.

```
function функцияСВозвращаемымЗначением() {
  return значение;
}

let результат = функцияСВозвращаемымЗначением(); //сохранение возвращаемого значения
```

Способы создания функций:

- **Function Declaration**:

```
function sayHi() {
    console.log('Добрый день, прекрасные леди!');
}
```

- **Function Expression**:

```
const sayHi = function() {
    console.log('Добрый день, прекрасные леди!');
};
```

- **Стрелочные функции (Arrow Functions)**:

```
const sayHi = () => {
    console.log('Добрый день, прекрасные леди!');
};
```

Имя функции:

- Имя функции следует выбирать так, чтобы оно отражало действие, которое она выполняет
- Обычно используются глагольные префиксы, чтобы указать характер действия, с последующим уточнением

```
function showSomething() {
  //отобразить что-либо
}

function calculateSum() {
  //вычислить сумму
}
```

Правило поиска переменных (Scope Chain):

- Если переменная создана внутри функции, она не может быть доступна извне этой функции
- Если обратиться к переменной из функции, движок JavaScript сначала ищет её внутри функции, а затем снаружи

Область видимости переменных

- **Глобальная область видимости**

  Переменные, объявленные на верхнем уровне файла или скрипта, доступны из любой части кода:

  ```
  let globalVar = "Это глобальная переменная";

  function globalFunction() {
    console.log(globalVar); //можем обращаться к глобальной переменной из функции
  }

  console.log(globalVar); //можем использовать глобальную переменную вне функции
  globalFunction(); //вызов функции
  ```

- **Локальная область видимости**

  Переменные, объявленные внутри функции, видны только внутри этой функции:

```
let globalVar = "Это глобальная переменная";

function globalFunction() {
  console.log(globalVar); //можем обращаться к глобальной переменной из функции
}

console.log(globalVar); //можем использовать глобальную переменную вне функции
globalFunction(); //вызов функции
```

Поднятие переменных и функций (hoisting)

- Поднятие переменных - это возможность вызывать функции и обращаться к переменным до их фактического объявления

```
//мы можем вызвать функцию до её объявления
sum(2); // 4

//затем объявили функцию
function sum(num) {
  return num + 2;
}
```

Выбор элементов на странице

- `document.getElementById` - это метод получение элемента на странице по его `id`

```
<div id="myDiv">Этот элемент имеет id</div>
<script>
  //получаем элемент с id "myDiv"
  let myElement = document.getElementById('myDiv');

  //меняем текстовое содержимое элемента
  myElement.textContent = 'Новый текст';
</script>
```

- `document.querySelector` - это метод получения элемента с использованием селекторов CSS

```
<ul>
  <li>Пункт 1</li>
  <li>Пункт 2</li>
  <li>Пункт 3</li>
</ul>
<script>
  //выбираем первый элемент li на странице
  let firstLi = document.querySelector('li');

  //меняем текстовое содержимое элемента
  firstLi.textContent = 'Измененный текст';
</script>
```

Отладка JavaScript

- Логирование - использование `console.log()` для вывода информации в консоль:

```
function sum(a, b) {
  console.log('Сумма:', a + b);
  return a + b;
}

console.log(sum(2, 3));
```

- **Точки останова (breakpoints)-** это установка точек останова в среде разработки или инструментах разработчика браузера, которые приостанавливают выполнение программы:

```
function sum(a, b) {
  debugger; //точка останова
  return a + b;
}

console.log(sum(2, 3)); //выполнение программы остановится на точке остановки
```