hillel IT school

Lesson 8

План заняття

- Основи роботи з функціями;
- Варіанти створення функцій;
- Параметри функцій, значення за промовчанням;
- Передача параметрів функції;
- Область видимості функції;
- Псевдо-масив arguments;
- Повернення значення з функції;
- Функція, що самовизивається (IIFE);

Function

Найчастіше нам треба повторювати одну й ту саму дію в багатьох частинах програми. Щоб не повторювати той самий код у багатьох місцях, придумані функції. Функції є основними "будівельними блоками" програми.

Функції – це "підпрограма" (або іменуємий блок коду), яку можна викликати із зовнішнього (або внутрішнього, у разі рекурсії) стосовно функції коду. Як і програма, функція складається з послідовності інструкцій, званої тілом функції. Значення можуть бути передані у функцію, а функція поверне значення.

У JavaScript функції є об'єктами першого класу, тобто: вони є об'єктами і з ними можна взаємодіяти і передавати їх так само, як і будь-який інший об'єкт. Якщо бути точним, функції це об'єкти Function.

Щоб повернути значення, відмінне від значення за замовчуванням, функції має бути інструкція **return**, яка вказує, що саме потрібно повернути. Функція без нього поверне значення за промовчанням. У випадку конструктора, викликаного ключовим словом new, значення за замовчуванням — це значення його параметра this. Для інших функцій значення за замовчуванням буде undefined.

Параметри виклику функції називають аргументами функції. Аргументи передаються у функцію за значенням. Якщо функція змінює значення аргументу, ця зміна не відображається на глобальному стані або функції, що викликає. Однак посилання на об'єкти - це теж значення, і вони відрізняються тим, що якщо функція змінює властивості об'єкта за посиланням, ця зміна видно зовні функції.

Змінні, оголошені всередині функції, видно лише всередині цієї функції.

Function

Пам’ятайте:

- Змінні, оголошені всередині функції, видно лише всередині цієї функції;
- Функція має доступ до зовнішніх змінних;
- Функция обладает полным доступом к внешним переменным и может изменять их значение;
- Зовнішня змінна використовується тільки якщо всередині функції немає такої локальної;

Синтаксис

```
function [name]([param]  [, param] [..., param]) {
  statements
}
```

Варіанти створення функцій

1. Функції виду "function declaration statement"

```
function foo(number) {
  return number * number;
}
```

2. Функції виду "function definition expression"

```
const bar = function (number) {
  return number * number;
};
```

function declaration statement

Оголошення функції (function definition або function declaration, або function statement) складається з ключового слова function і наступних частин:

1. Ім'я функції.
2. Список параметрів (приймаються функцією) укладених у круглі дужки () та розділених комами.
3. Інструкції, які будуть виконані після виклику функції, укладають фігурні дужки { }.

```
function foo(obj) {
    obj.name = 'Sergey'
}
```

Такий варіант об’яви буде підійматись, тобто працює hosting;

function definition expression

За синтаксисом є інструкцією (statement), ще функція може бути виду "function definition expression". Така функція може бути анонімною (вона не має імені).

Ім'я може бути і присвоєно для виклику самої себе всередині самої функції та для відладчика (debugger) для ідентифікованих функцій у стек-треках (stack traces; "trace" - "слід" / "відбиток").

Функції "function definition expression" зручні, коли функція передається аргументом іншої функції.

```
const foo = function () {
    console.log('Hello')
}
const bar = function inner() {
    console.log('Hello')
    console.log(inner)
}
```

Аргументи функції

У JavaScript параметри функції, яким при її викликі не передаються значення, за замовчуванням приймають значення undefined. Однак у деяких випадках може бути корисно встановити інше значення за замовчуванням. Саме для таких випадків призначено параметри за замовчуванням.

```
function showMessage(from,  text = "dummy text") {
  alert( from + ": " + text );
}
```

У JavaScript параметри за промовчанням обчислюються щоразу, коли функція викликається без відповідного параметра.

return

Директива return може знаходитись у будь-якому місці тіла функції. Як тільки виконання доходить до цього місця, функція зупиняється, і значення повертається в код, що її викликав.

Можна використовувати return і без значення. Це призведе до негайного виходу із функції.

Порожній return аналогічний return undefined

Ніколи не додавайте переклад рядка між return та його значенням

1. return
2. (some + long + expression + or + whatever * f(a) + f(b))

Назви функцій

Функція – це. Тому ім'я функції зазвичай є дієсловом. Воно має бути коротким, точним і описувати дію функції, щоб програміст, який читатиме код, отримав правильне уявлення у тому, що виконує функція.

Як правило, використовуються дієслівні префікси, що позначають загальний характер дії, після яких слідує уточнення. Зазвичай у командах розробників діють угоди щодо значень цих префіксів.

Наприклад, функції, що починаються з "show", зазвичай щось показують.

Функції, що починаються з...

- "get..." – повертають значення,
- "calc..." – щось обчислюють,
- "create..." – щось створюють,
- "check..." – щось перевіряють та повертають логічне значення, тощо.

Важливо!

Одна функція – одна дія

Function scope

Змінні оголошені у середині функції не можуть бути доступними де-небудь поза цією функцією, тому змінні (які потрібні саме для функції) оголошують тільки в scope функції. При цьому функція має доступ до всіх змінних та функцій, оголошених усередині її scope. Тобто функція оголошена у глобальному scope має доступ до всіх змінних у глобальному scope. Функція оголошена всередині іншої функції ще має доступ і до всіх змінних батьківської функції та інших змінних, до яких ця батьківська функція має доступ.

arguments

Усередині функції отримати доступ до її аргументів можна через об'єкт arguments.

- arguments: Об'єкт, схожий на масив, що містить усі аргументи, передані в поточну функцію.
- arguments.callee (Застаріло): Функція, що виконується в даний момент.
- arguments.caller: Функція, що викликала поточну функцію.
- arguments.length: Число аргументів, переданих у функцію.

```
function foo() {
   console.log(arguments)
}
```

IIFE

Immediately Invoked Function Expression – це JavaScript функція, яка виконується відразу після того, як вона була визначена.

Це тип виразів, також відомий як Self-Executing Anonymous Function, який складається із двох основних частин.

1. Перша - це анонімна функція з лексичною областю видимості, укладеним всередині Оператора угруповання (). Завдяки цьому змінні IIFE замикаються в межах, і глобальна область видимості ними не засмічується.
2. Друга частина створює функціональний вираз (), що миттєво виконується, завдяки якому JavaScript-движок виконує функцію безпосередньо.

```
(function () {
    console.log('Hello');
})();
```

Дякую за увагу