Функции

JavaScript

Функция

- **Функция** – фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы
- Выделяют **встроенные** и **пользовательские** функции
- Пример встроенных функций:
  - alert()
  - prompt()
  - confirm()

Объявление и вызов

- function showMessage() {
  - alert('Привет всем присутствующим!');
- }

- showMessage();
- showMessage()

Локальные переменные

- function showMessage() {
  - var message = 'Привет, я – Рома!';
  - // локальная переменная
  - alert(message);
- }

- showMessage(); // 'Привет, я – Рома!'
- alert(message); // <-- ошибка

Глобальные переменные

- var userName = 'Вася';
- function showMessage() {
  - userName = 'Рома';
  - var message = 'Привет, я ' + userName;
  - alert(message);
- }

- showMessage();
- alert(userName);

Разделение областей видимости

- var userName = 'Вася';
- function showMessage() {
  - var userName = 'Рома';
  - var message = 'Привет, я ' + userName;
  - alert(message);
- }

- showMessage();
- alert(userName);

Аргументы функции

- function showMessage(from, text) {
  - from = 'Сообщение для' + from;
  - alert(from + '\r\n' + text);
- }

- var from = "Гость";
- showMessage(from, "Зарегистрируйтесь!");
- alert(from);

Аргументы по умолчанию

- function showMessage(from, text) {
  - if (from === undefined) {
    - from = 'Админ';
  - }
  - text = text || 'текст не передан';
  - alert(from + ": " + text);
- }
- showMessage("Гость");
- showMessage();

Возврат значения

- function checkAge(age) {
  - return age > 18 ? true : false;
- }
- var age = prompt('Ваш возраст?');
- if (checkAge(age)) {
  - alert('Доступ разрешен');
- }
- else {
  - alert('В доступе отказано');
- }

Вывод кода функции

- function sayHi() {
  - alert("Привет");
- }

- alert(sayHi);

Альтернативный способ объявления функции

- «Классическое» объявление функции называется в спецификации языка «Function Declaration», а альтернативное «*Function Expression*»
  - *Function Declaration* — функция, объявленная в основном потоке кода
  - *Function Expression* — объявление функции в контексте какого-либо выражения

Function Expression

- var sayHi = function(person) {
  - alert("Привет, " + person);
- };

- sayHi('Админ');

Анонимные функции

- function ask(question, yes, no) {
  - if (confirm(question)) yes() else no();
- }
- function showOk() {
  - alert("Вы согласились.");
- }
- function showCancel() {
  - alert("Вы отменили выполнение.");
- }
- ask("Вы согласны?", showOk, showCancel);

Укороченный вариант

- function ask(question, yes, no) {
  - if (confirm(question)) yes() else no();
- }
- ask("Вы согласны?",
  function() { alert("Вы согласились"); },
  function() { alert("Вы отказались"); }
-);

new Function

- new Function(params, code):
  - params – параметры функции через запятую в виде строки
  - code – код функции в виде строки

- var sum = new Function('a, b', 'return a+b;'); var result = sum(1, 2);
- alert(result);

Рекурсия, стек

- **Рекурсия** – вызов функции самой себя
- **Стек** – абстрактный тип данных, представляющий собой список элементов, организованных по принципу *LIFO*

Рекурсия

- function pow(x, n) {
  - if (n != 1) {
    - return x * pow(x, n - 1);
  - }
  - else {
    - return x;
  - }
- }
- alert(pow(2, 3));

Контекст выполнения, стек

- **Контекст выполнения** – служебная информация, соответствующая текущему запуску функции
- Включает в себя локальные переменные функции и конкретное место в коде, на котором находится интерпретатор
- Контекст: { x: 2, n: 3, строка 1 }
- **Стек контекстов** – текущий контекст выполнения в специальной внутренней структуре данных

Таймер-планировщик

- Почти все реализации JavaScript имеют внутренний таймер-планировщик, который позволяет задавать вызов функции через заданный период времени
- Эта возможность поддерживается и в браузерах

setTimeout

- var tId = setTimeout(func, delay[, arg1,…]);
- func – функция для исполнения
- delay – задержка в милисекундах
- arg1, arg2…Аргументы, которые нужно передать функции (не поддерживаются в IE9-)

Пример

- function func(phrase, who) {
  - alert(phrase + ', ' + who);
- }
- setTimeout(func, 1000, "Привет", "Гость");

Отмена исполнения

- var timerId = setTimeout(function() {
  - alert(1)
- }, 1000);
- alert(timerId);
- clearTimeout(timerId);
- alert(timerId);

setInterval

- var tId = setInterval(func, delay[, arg1,...]);
- Смысл аргументов — тот же самый
- В отличие от setTimeout, запускает выполнение функции не один раз, а регулярно повторяет её через указанный интервал времени
- Остановить исполнение можно вызовом clearInterval(timerId)

Пример

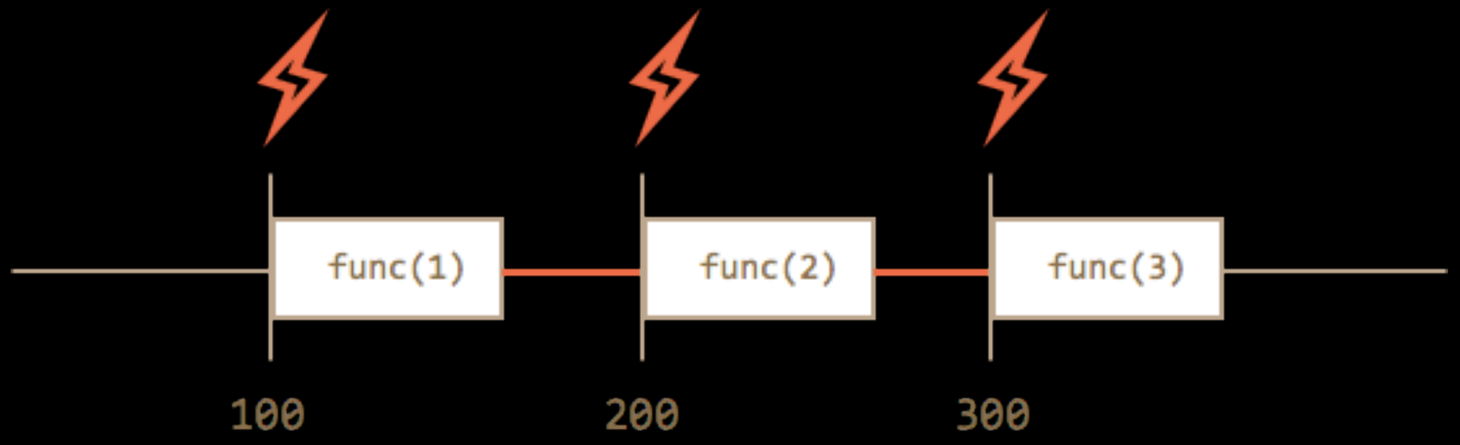
- var timerId = setInterval(function() {
  - alert("тик");
- }, 2000);
- setTimeout(function() {
  - clearInterval(timerId);
  - alert('стоп');
- }, 5000);

Рекурсивный setTimeout

- var timerId = setTimeout(function tick() {
  - alert("тик");
  - timerId = setTimeout(tick, 2000);
- }, 2000);

SetInterval

- var i = 1;
- setInterval(function() {
  - func(i);
- }, 100);



SetTimeout

- var i = 1;
- setTimeout(function run() {
  - func(i);
  - setTimeout(run, 100);
- }, 100);