Дәріс-10. Функиялар

Жоспар:

1 Функцияларды шақыру
2 Стектер
3 Кезектер
4 Бинарлы бұтақтар

Функциялары сондай-ақ JavaScript тілінің түйінді элементтері болып табылады. Негізгі функциялары өте қарапайым:

function add(x, y) { var total = x + y; return total; }

Бұл мысалда функциялары туралы іс жүзінде білу керектердің барлығы көрсетілгендей. JavaScript функцияларда нөл немесе одан да көп параметрлерді қабылдай алады. Функциялар кез келгенді білдіруі және өз айнымалыларын анықтауды қамтуы мүмкін, олар осы функциялар үшін жергілікті. return нұсқауы маңызы бар мәнді қайтару үшін пайдаланылады және тоқтату функциясын орындайды. Егер return нұсқаулық функцияда жоқ (немесе бар, бірақ көрсетілген қайтарылатын мәні) болса, онда JavaScript undefined қайтарады. Функцияны оған параметрлерін мүлдем бермей шақыртуға болады. Мұндай жағдайда, олардың мәндері undefined тең:

add(); // NaN // Нельзя проводить сложение undefined и undefined

Көбірек аргументтер беруге болады функцияны күтуге қарағанда:

add(2, 3, 4); // 5 // используются только первые два аргумента, "4" игнорируется

Бұл қажетсіз болып көрінуі мүмкін, бірақ шын мәнінде функциялары "артық" аргументпен arguments псевдомассивті көмегімен қол жеткізе алады, онда маңызы бар функциялардың барлық аргументтері берілген. Шексіз аргументтер қабылдайтын функциялар жазайық:

function add() { var sum = 0; for (var i = 0, j = arguments.length; i < j; i++) { sum += arguments[i]; } return sum; } add(2, 3, 4, 5); // 14 Немесе орташа мәндерді есептеу үшін функция құрамыз: function avg() { var sum = 0; for (var i = 0, j = arguments.length; i < j; i++) { sum += arguments[i]; } return sum / arguments.length; } avg(2, 3, 4, 5); // 3.5 28 Өте ыңғайлы, бірақ шағын проблема бар. Бұл функция үтірмен бөлінген цифрлардың тізімін ғана түсінеді. Ал егер массивке сандар беру керек болса? Функция құрайық: function avgArray(arr) { var sum = 0; for (var i = 0, j = arr.length; i < j; i++) { sum += arr[i]; } return sum / arr.length; } avgArray([2, 3, 4, 5]); //

3.5 Бұл жағдайда, егер сіз функциялардың бірінші нұсқасын пайдаланғыңыз келсе, оны қайтадан жазбай-ақ, онда JavaScript–та аргументтер массивімен функцияны шақыру мүмкіндігі бар. Бұл үшін apply() әдісі пайдаланылады: avg.apply(null, [2, 3, 4, 5]); // 3.5 Екінші аргумент apply() әдісімен массив беріледі, функциялар аргументтер ретінде берілетін. Бірінші аргумент туралы кейінірек айтамыз. Функциялардың әдістерінің болуы, сондай-ақ, бұл-шын мәнінде, олар объектілері болып табылады. JavaScript –та анонимді функциялар жасауға болады:

var avg = function() { var sum = 0; for (var i = 0, j = arguments.length; i < j; i++) { sum += arguments[i]; } return sum / arguments.length;

} Бұл жазба function avg() семантикалық тең жазу. Бұл әр түрлі қызықты фокустарды пайдалануға мүмкіндік береді. Міне, қараңызшы, жергілікті айнымалыларды "жасыру" функциясы қалай болады:

var a = 1; var b = 2; (function() { var b = 3; a += b; })(); a; // 4 b; // 2 JavaScript - та рекурсивті функцияларды шақыру мүмкіндігі бар. Бұл иерархиалық құрылымдар деректерімен (мысалы, DOM жұмыс жасау кезінде кездесетін) жұмыс істеу кезінде пайдалы болуы мүмкін. function countChars(elm) { if (elm.nodeType == 3) { // TEXT_NODE return elm.nodeValue.length; } var count = 0; for (var i = 0, child; child = elm.childNodes[i]; i++) { count += countChars(child); } return count; } Мұнда мәселелермен бетпе-бет келеміз: функцияны рекурсивті қалай шақыруға болады, егер онда атау жоқ болса? Бұл үшін JavaScript–та аталған функционалдық өрнектеу бар.

Мысал, функцияларды пайдалану: var charsInBody = (function counter(elm) { if (elm.nodeType == 3) { // TEXT_NODE return elm.nodeValue.length; } var count = 0; for (var i = 0, child; child = elm.childNodes[i]; i++) { count += counter(child); } return count; })(document.body); Мысалда функциялардың аты ғана функциялардың ішіндегі ең қолжетімділік. Бұл кодты оңтайландырады және оқылуды жақсартады. Меншікті объектілер Классикалық объектно-ориентирленген бағдарламалау (ООБ) объектілері — бұл деректер мен әдістердің коллекциясы, осы деректермен жүзеге асатын. JavaScript-бұл тіл, типтік негізделген, және класс сияқты ұғымдар C немесе Java тілдерінде анықтауы жоқ. Класс орнына JavaScript-та функциялар пайдаланады. Объектінің жеке деректерін қамтитын өрістің аты және тегі. Аттарды белгілеудің екі түрі бар:

"Аты Тегі" немесе "Тегі, Аты". Объектілер мен функциялардың көмегімен мыналарды жасауға болады:

function makePerson(first, last) { return { first: first, last: last } } function personFullName(person) { return person.first + ' ' + person.last; } 30 function

personFullNameReversed(person) { return person.last + ', ' + person.first } s = makePerson("Simon", "Willison");

personFullName(s); // Simon Willison personFullNameReversed(s);

// Willison, Simon Жұмыс істейді, бірақ бұл код жарамсыз. Мұндай тәсілмен жаһандық объектпен ондаған функциялар болады.

Бұларды объектіге функцияны тіркеп түзетуге болады. Бұл барлық функциялары олар объектілер:

function makePerson(first, last) { return { first: first, last: last, fullName: function() { return this.first + ' ' + this.last; }, fullNameReversed: function() { return this.last + ', ' + this.first; } } }

s = makePerson("Simon", "Willison") s.fullName(); // Simon Willison s.fullNameReversed();

// Willison, Simon Ал мыналар жаңалар: 'this' кілт сөзі. Егер 'this' функциялар ішінде пайдаланылса, ол ағымдағы объектте сілтеме жасайды. Кілт сөздер мәні функцияларды шақыру әдістеріне байланысты болады. Егер функцияны шақыру, объектіге үндеу жолдау, нүкте арқылы немесе тік төртбұрышты жақшалар болса, онда 'this' осы объектіге тең болады. Өзге жағдайда 'this' жаһандық объектіге сілтеме болады. Бұл жиі қателіктерге әкеледі. Мысалы: s = makePerson("Simon", "Willison") var fullName = s.fullName; fullName(); // undefined undefined fullName() шақыру кезінде, 'this' жаһандық объектіге сілтеме алады. Ал жаһандық объектіде айнымалылар first және last анықталған жоқ, онда екі undefined иеленеміз. 'this' кілт сөздер ерекшелігін пайдаланып makePerson функция кодын жақсартуға болады: function Person(first, last) { this.first = first; this.last = last; this.fullName = function() { return this.first + ' ' + this.last; 31 }; this.fullNameReversed = function() { return this.last + ', ' + this.first; }; } var s = new Person("Simon", "Willison"); Мысалда қолданған жаңа кілт сөз: 'new'. Ол 'this' тығыз байланысты. Бұл кілт сөз, жаңа бос объект жасайды, одан кейін аталған функцияны шақырады, athis бұл жаңа объектіге сілтеме алады. Функцияларды шақыруға арналған 'new' конструкторлар деп аталады. Келісім бар, оған сәйкес барлық функциялары-конструкторлар бас әріптен басталып жазылады. Әр қашан конструктор көмегімен жаңа обьект құрылады, қайтадан құрамыз және екі жаңа функцияларды жасаймыз.

Бақылау сұрақтар:

1. Java тілінде неше түрлі нұсқауыштар қолданылады?
2. Java тілінде динамикалық жадымен жұмыс істеудің неше тәсілі бар?
3. Тізіммен қандай операцияларды орындауға болады?