Массивы

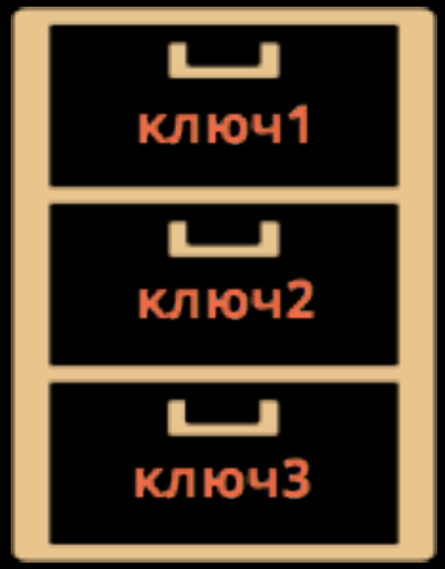
JavaScript

Объекты как ассоциативные массивы

- Объекты в JavaScript сочетают в себе два важных функционала
  - Ассоциативный массив: структура, пригодная для хранения любых данных
  - Языковые возможности для объектно-ориентированного программирования

Ассоциативные массивы

- Ассоциативный массив — структура данных, в которой можно хранить любые данные в формате ключ-значение



Основные операции над объектом

- Создание
  - var person = new Object();
  - var person = {};
- Присвоение.свойств
  - person.name = 'Вася';
  - person.age = 25;
- Удаление свойств
  - delete person.age;

Проверка существования свойства

- if ("name" in person) {
  - alert("Свойство name существует!");
- }
  if (person.name === undefined) {
  - alert("Свойство lalala несуществует!");
- }

Доступ через []

- Существует альтернативный синтаксис работы со свойствами, использующий квадратные скобки объект['свойство']
  - var person = {};
  - person['name'] = 'Вася';
  - person['любимый стиль музыки'] = 'Джаз';
  - alert(person['любимый стиль музыки']);
  - alert(person.любимый стиль музыки;

Доступ к свойству через переменную

- var person = {};
- person.age = 25;
- var key = 'age';
- alert(person[key]);

Объявление со свойствами

- var menuSetup = {
  - width: 300,
  - height: 200,
  - title: "Menu“
- };
- var menuSetup = {};
- menuSetup.width = 300;
- menuSetup.height = 200;
- menuSetup.title = 'Menu';

Вложенные объекты

- var user = {
  - name: "Таня",
  - age: 25,
  - size: {
    - top: 90,
    - middle: 60,
    - bottom: 90
  - }
- };

Перебор свойств

- Для перебора всех свойств из объекта используется цикл по свойствам for..in
- for (key in obj) {
  - ... делать что-то с obj[key] ...
- }

Пример

- var menu = {
  - width: 300,
  - height: 200
  - title: "Menu"
- };

- for (var key in menu) {
  - alert("Ключ: " + key + " значение: " + menu[key]);
- }

Порядок свойств

- Соглашение говорит, что если имя свойства — нечисловая строка, то такие ключи всегда перебираются в том же порядке, в каком присваивались
- С другой стороны, если имя свойства — число или числовая строка, то все современные браузеры сортируют такие свойства в целях внутренней оптимизации

Пример

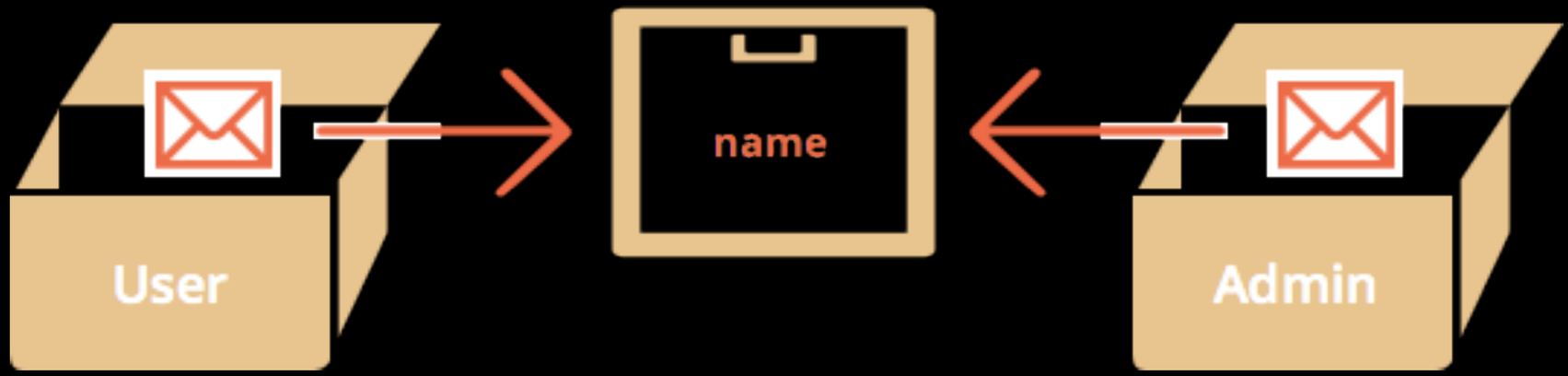
- var codes = {
  - "7": "Россия",
  - "38": "Украина",
  - "1": "США"
- };
- Console.log(codes);
- Object {1: "США", 7: "Россия", 38: "Украина"}

Пример 2

- var codes = {
  - "+7": "РФ",
  - "+38": "Украина",
  - "+1": "США"
- };
- Console.log(codes);
- Object {+7: "РФ", +38: "Украина", +1: "США"}

Копирование по ссылке

- В переменной, которой присвоен объект, хранится не сам объект, а «ссылка» на него
- При копировании переменной с объектом — копируется эта ссылка, а объект остается в единственном экземпляре



Клонирование объектов

- var user = {
  - name: "Вася",
  - age: 30
- };
- var clone = {};
- for (var key in user) {
  - clone[key] = user[key];
- }
- clone.name = "Петя";
- clone alert(user.name); // "Вася"

Массивы с числовым индексом

- *Массив* — разновидность объекта, которая предназначена для хранения пронумерованных значений и предлагает дополнительные методы для удобного манипулирования такой коллекцией
- Они обычно используются для хранения упорядоченных коллекций данных, например — списка товаров на странице, студентов в группе и т.п

Создание / заполнение / доступ

- Создание
  - var fruits = [];
- Заполнение
  - fruits = ["Яблоко", "Апельсин", "Слива"];
- Доступ
  - alert(fruits[0]);
  - alert(fruits[1]);
  - alert(fruits[2]);

Применение массивов

- Очередь (коллекция элементов в которой элементы добавляются в конец, а обрабатываются с начала)
- Стек (коллекция элементов, в которой элементы добавляются в конец и берутся с конца)

POP

- Удаляет последний элемент из массива и возвращает его

- var fruits = ["Яблоко", "Апельсин", "Груша"];
- alert(fruits.pop()); // "Груша"
- alert(fruits); // "Яблоко, Апельсин"

Push

- Добавляет элемент в конец массива

- var fruits = ["Яблоко", "Апельсин"];
- fruits.push("Груша");
- alert(fruits); // Яблоко, Апельсин, Груша

shift

- Удаляет из массива первый элемент и возвращает его

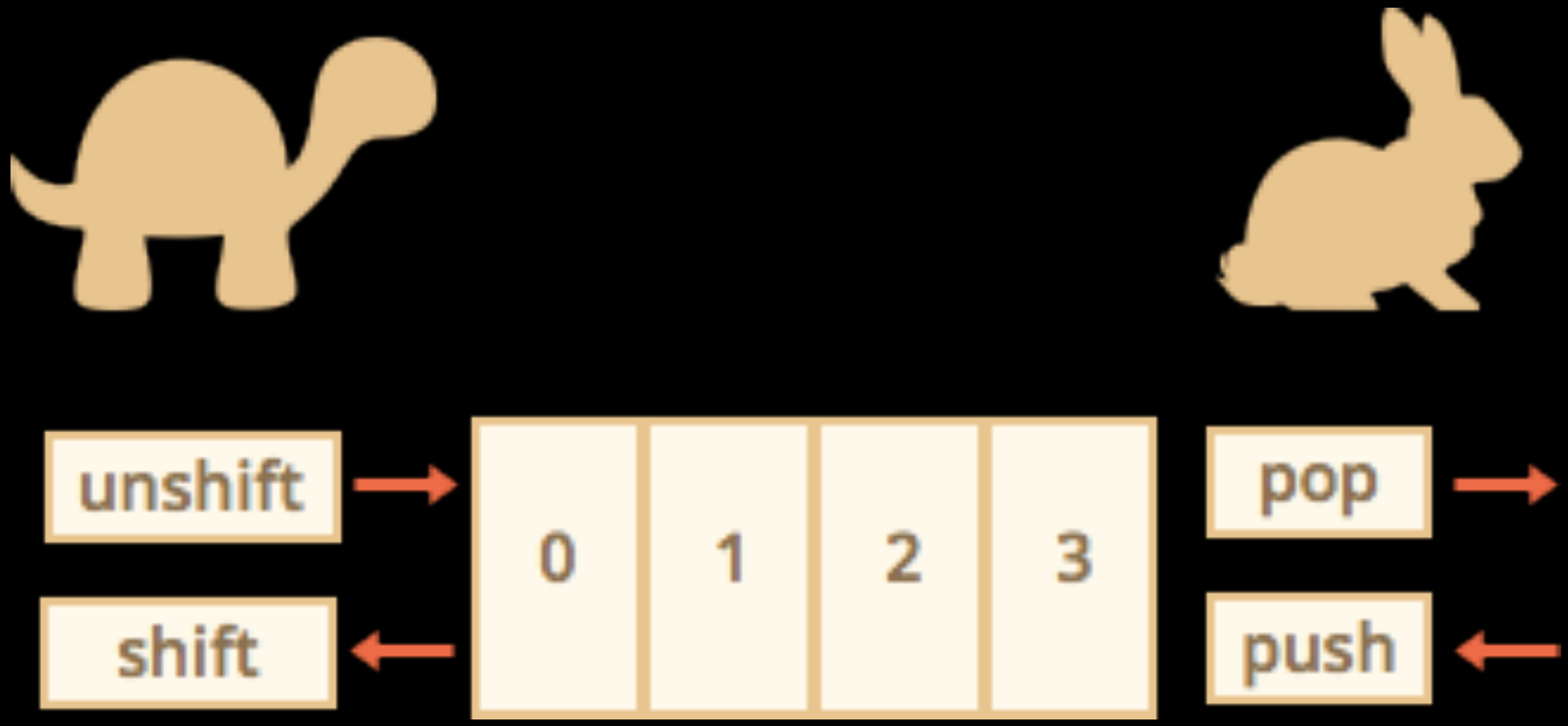
- var fruits = ["Яблоко", "Апельсин", "Груша"];
- alert(fruits.shift()); // "Яблоко“
- alert(fruits); // "Апельсин, Груша"

unshift

- Добавляет элемент в начало массива

- var fruits = [" Апельсин ", " Груша "];
- fruits.unshift(" Яблоко ");
- alert(fruits); // Яблоко, Апельсин, Груша

Влияние на быстродействие



Перебор элементов и length

- var arr = ["Яблоко", "Апельсин", "Груша"];
- for (var i = 0; i < arr.length; i++) {
  - alert(arr[i]);
- }

- length = последний индекс + 1
- При уменьшении length массив обрезается
- arr.length = 2;

Создание через new

- Обычно new Array(элементы, ...) создаёт массив из данных элементов, но если у него один аргумент-число new Array(число), то он создает массив *без элементов, но с заданной длиной* (массив, у которого все элементы undefined)

Многомерные массивы

- Массивы в JavaScript могут содержать в качестве элементов другие массивы
- var matrix = [
  - [1, 2, 3],
  - [4, 5, 6],
  - [7, 8, 9]
-];
- alert(matrix[1][1]);

split

- Преобразует строку в массив
- var names = 'Маша, Петя, Марина';
- var arr = names.split(', ');
- for (var i = 0; i < arr.length; i++) {
  - alert('Вам сообщение ' + arr[i]);
- }

- alert("a,b,c,d".split(',', 2)); // a,b

- alert("текст".split('')); // т,е,с,т

join

- Склеивает массив в строку
- var arr = ['Маша', 'Петя', 'Марина'];
- var str = arr.join(';');
- alert(str); // Маша;Петя;Марина

- alert(new Array(4).join("ля")); // ляляля

sort

- var arr = [2, 1, 15];
- arr.sort();
- alert(arr); // 1, 15, 2

- По умолчанию sort сортирует, преобразуя элементы к строке

Sort(fn)

- function compareNumeric(a, b) {
  - if (a > b) return 1;
  - if (a < b) return -1;
- }
- var arr = [2, 1, 15];
- arr.sort(compareNumeric);
- alert(arr); // 1, 2, 15

revers

- Метод arr.reverse() меняет порядок элементов в массиве на обратный
- var arr = [1, 2, 3];
- arr.reverse();
- alert(arr); // 3,2,1

concat

- Метод arr.concat(value1, value2, … valueN) создаёт новый массив, в который копируются элементы из arr, а также value1, value2, ... valueN

- var arr = [1, 2];
- var newArr = arr.concat(3, 4);
- alert(newArr); // 1,2,3,4

indexOf/lastIndexOf

- возвращают номер элемента searchElement в массиве arr или -1

- var arr = [1, 0, false,0];
- alert(arr.indexOf(0)); // 1
- alert(arr.indexOf(false)); // 2
- alert(arr.indexOf(‘dfdsfsdf’)); // -1

forEach

- Метод используется для перебора массива
- Для каждого элемента массива вызывается функция callback, которая принимает три параметра
  callback(item, i, arr):
  - item — очередной элемент массива
  - i — его номер
  - arr — массив, который перебирается

Пример

- var arr = ["Яблоко", "Апельсин", "Груша"];
- arr.forEach(function(item, i, arr) {
  - alert(i + ": " + item + " (массив:" + arr + ")");
- });

filter

- Метод используется для *фильтрации* массива через функцию
- Создаёт новый массив, в который войдут только те элементы arr, для которых вызов callback(item, i, arr) возвратит true

Пример

- var arr = [1, -1, 2, -2, 3];
- var positiveArr = arr.filter(function(number) {
  - return number > 0;
- });
- alert(positiveArr); // 1,2,3

map

- Метод используется для *трансформации* массива
- Создаёт новый массив, который будет состоять из результатов вызова callback(item, i, arr) для каждого элемента arr

Пример

- var names = ['HTML', 'CSS', 'JavaScript'];
- var nameLen = names.map(function(name) {
  - return name.length;
- });
- alert(nameLen); // 4,3,10

every/some

- Эти методы используется для проверки массива
  - Метод every возвращает true, если вызов callback вернёт true для *каждого* элемента arr
  - Метод some возвращает true, если вызов callback вернёт true для *какого-нибудь* элементаarr

Пример

- var arr = [1, -1, 2, -2, 3];
- function isPositive(number) {
  - return number > 0;
- }
- alert(arr.every(isPositive)); // false
- alert(arr.some(isPositive)); // true

reduce/reduceRight

- Метод используется для последовательной обработки каждого элемента массива с сохранением промежуточного результата
- Применяет функцию callback по очереди к каждому элементу массива слева направо, сохраняя при этом промежуточный результат

Аргументы функции

- Аргументы функции callback():
  - previousValue — последний результат вызова функции, он же «промежуточный результат»
  - currentItem — текущий элемент массива, элементы перебираются по очереди слева-направо
  - index — номер текущего элемента
  - arr — обрабатываемый массив

Пример

- var arr = [1, 2, 3, 4, 5]
- var res = arr.reduce(function(sum, current) {
  - return sum + current;
- }, 0);
- alert(res); // 15