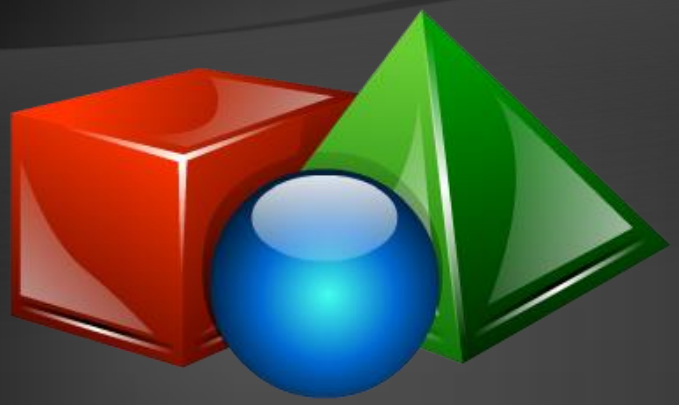
Telerik Academy



Using Objects

Objects, Properties, Primitive and Reference Types

JavaScript Fundamentals

Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

Telerik Academy

Object Types and Objects

Modeling Real-world Entities with Objects

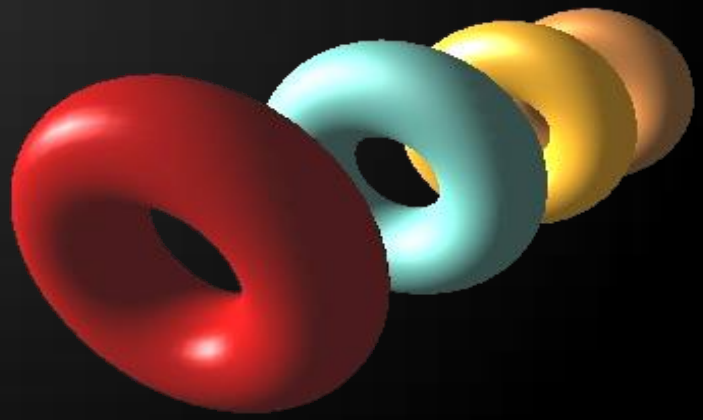
Telerik Academy

What are Objects?

- Software objects model real-world objects or abstract concepts
  - Examples:
    - bank, account, customer, dog, bicycle, queue
- Real-world objects have states and behaviors
  - Account states:
    - holder, balance, type
  - Account behaviors:
    - withdraw, deposit, suspend

What are Objects? (2)

- How do software objects implement real-world objects?
  - Use variables/data to implement states
  - Use methods/functions to implement behaviors
- An object is a software bundle of variables and related methods



Objects Represent

- checks
- people
- shopping list

Things from the real world

...

- numbers
- characters
- queues
- arrays

Things from the computer world

What is a Class?

- The formal definition of a object type:

> Object types act as templates from which an instance of an object is created at run time. Types define the properties of the object and the methods used to control the object's behavior.

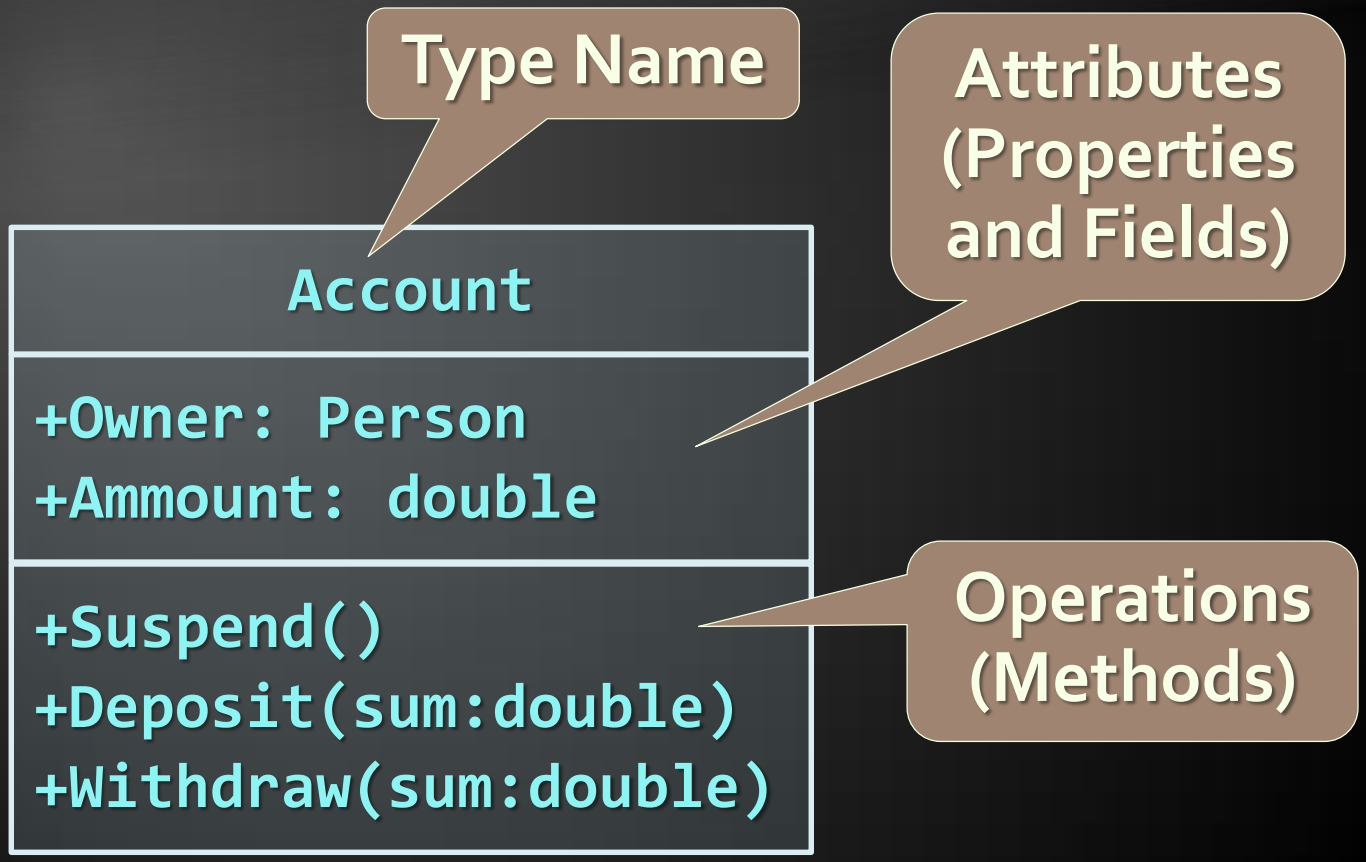
Definition by Google

Object Types

- Object Types provide the structure for objects
  - Define their prototype, act as template
- Object Types define:
  - Set of attributes
    - Represented by variables and properties
    - Hold their state
  - Set of actions (behavior)
    - Represented by methods
- A type defines the methods and types of data associated with an object

Telerik Academy

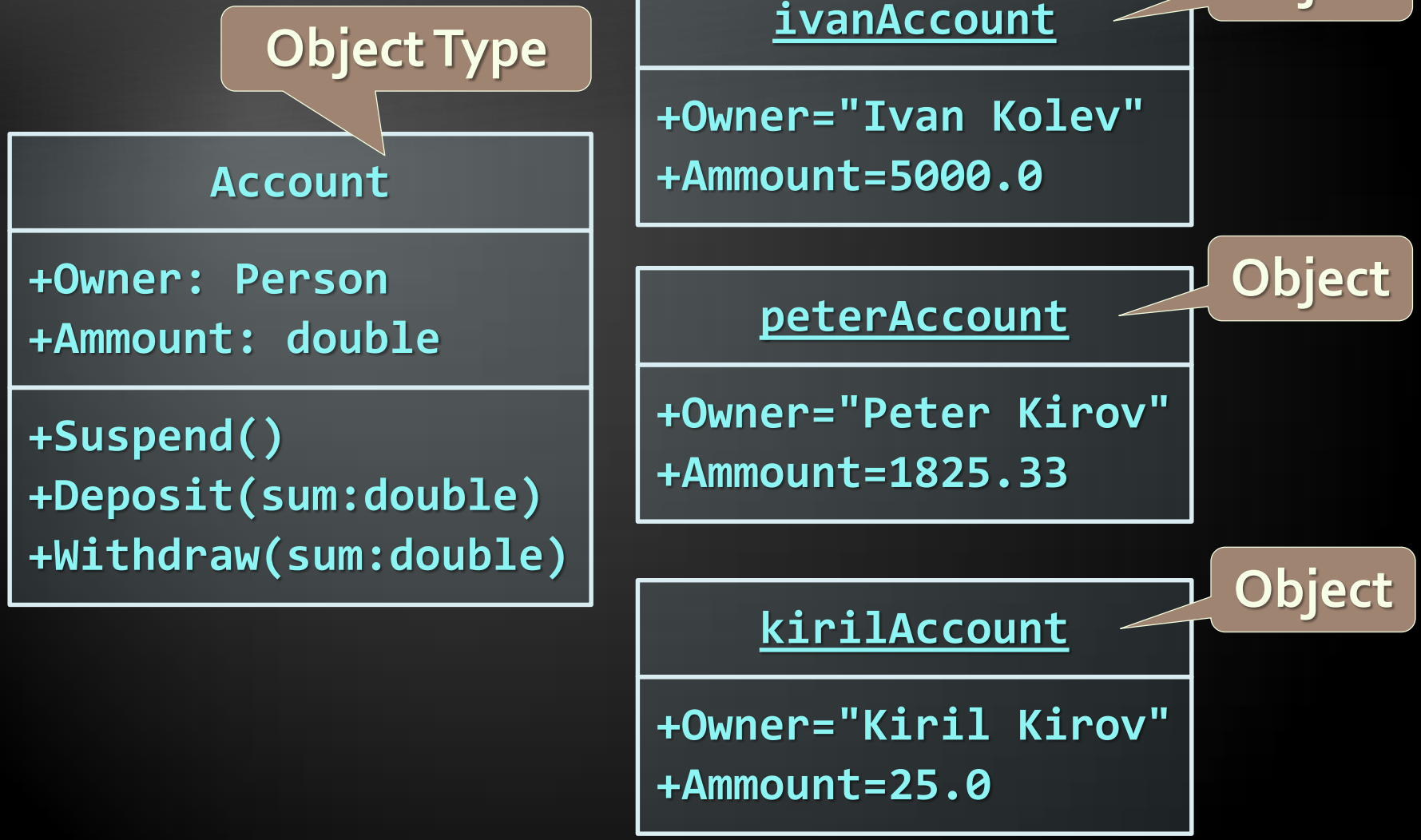
Object Types – Example



Objects

- An object is a concrete instance of a particular object type
- Creating an object from an object type is called instantiation
- Objects have state
  - Set of values associated to their attributes
- Example:
  - Type: Account
  - Objects: Ivan's account, Peter's account

Objects – Example



Telerik Academy

JavaScript Objects Overview

What are Objects?

Telerik Academy

Objects Overview

- JavaScript is designed on a simple object-based paradigm
  - An object is a collection of properties
- An object property is association between a name and a value
  - A value of property can be either a method (function) or a field (variable)
- Lots of predefined objects available in JS
  - Math, document, window, etc…
- Objects can be created by the developer

Object Properties

- Each object has properties
  - Properties are variables attached to the object
  - Properties of an object can be accessed with a dot-notation:

```
var arrStr = arr.join(', '); // property join of Array
var length = arr.length;  // property length of Array
var words = text.split(' ');
```

Telerik Academy

Objects and Properties

Live Demo

Telerik Academy

Object and Primitive Types

The Types in JavaScript

Telerik Academy

Reference and Primitive Types

- JavaScript is a typeless language
  - Variables don't have type, but their values do
- JavaScript has six different types:
  - `Number`, `String`, `Boolean`, `Null`, `Undefined` and `Object`
- `Object` is the only object type
  - It is copied by reference
- `Number`, `String`, `Boolean`, `Null`, `Undefined` are primitive types
  - Copied by value

Telerik Academy

Reference and Primitive Types (2)

- The primitive types are `Boolean`, `Number`, `String`, `Undefined` and `Null`
  - All the other types are actually of type object
    - Including arrays, dates, custom types, etc…

```
console.log(typeof new Object() === typeof new Array()); // true
console.log(typeof new Object() === typeof new Date()); // true
console.log(typeof new Array() === typeof new Date()); // true
```

- All types derive from object
  - Their type is object

Telerik Academy

Primitive and Reference Types

Live Demo

Primitive Types

- Primitive types are passed by value
  - When passed as argument
    - New memory is allocated
    - The value is copied in the new memory
    - The value in the new memory is passed
- Primitive types are initialized with type literals

```
var number = 5,
    text = 'Hello there!';
```

- Primitive types have a object type wrapper

```
var number = 5, // Holds a primitive value of 5
    numberObj = new Number(5); // Holds a object value of 5
```

Telerik Academy

Primitive Types – Example

- Assign string values to two variables
  - Create an object using their value
  - Change the value of the variables
  - Each object has its own value

```
var fname = 'Peter',
    lname = 'Johnson',
    person = { firstName: fname, lastName: lname };

lname = 'Peterson';
console.log(person.lastName) // logged 'Johnson'
```

Telerik Academy

Primitive Types

Live Demo

Reference Type

- Object is the only object type
  - When passed to a function the value is not copied, but instead a reference of it is passed

```
var marks, student;
marks = [
  { subject : 'JavaScript', score : 4.50 },
  { subject : 'OOP', score : 5.00 },
  { subject : 'HTML5', score : 6.00 },
  { subject : 'Photoshop', score : 4.00 }];

student = { name: 'Doncho Minkov', marks: marks };
marks[2].score = 5.50;

console.log(student.marks); // logs 5.50 for HTML5 score
```

Telerik Academy

Object Types

Live Demo

Telerik Academy

JavaScript Object Literal

Creating simple objects

Telerik Academy

JavaScript Object Literal

- JavaScript object literal is a simplified way to create objects
  - Using curly brackets:

```
var person = {
  firstName: 'Doncho',
  lastName: 'Minkov',
  toString: function () {
    return this.firstName + ' ' + this.lastName;
  }
}
```

  - Then the object properties can be used:

```
console.log(person.toString()); // writes 'Doncho Minkov'
```

Telerik Academy

Using JavaScript Object Literal

Live Demo

Creating Objects

- Object notations are great, but repeating code is not, right?
  - Lets make two persons:

```
var minkov, georgiev;
minkov = {fname: 'Doncho', lname: 'Minkov',
  toString: function(){ return this.fname + ' ' + this.lname;}
}
georgiev = { fname: 'Georgi', lname: 'Georgiev',
  toString: function(){ return this.fname + ' ' + this.lname;}
}
```

  - Lots of repeating code
    - Can't we use a constructor or function to create an object?

Telerik Academy

Object Building Function

- A function for building objects
  - Just pass first and last name and get a object
    - Something like a constructor

```
var minkov, georgiev;
function makePerson(fname, lname) {
  return {
    fname: fname,
    lname: lname,
    toString: function (){return this.fname + ' ' + this.lname;}
  }
}
minkov = makePerson('Doncho', 'Minkov');
georgiev = makePerson('Georgi', 'Georgiev');
```

- Much cooler, right?

Telerik Academy

Object Building Function

Live Demo

Telerik Academy

JavaScript Object Properties

Telerik Academy

JS Object Properties

- JavaScript objects are just a set of key/value pairs
  - Each value can be accessed by its key
  - Properties in objects are accessed using the dot-notation (obj.property)
  - Yet properties can be used with brackets
    - Like an array

```
document.write === document['write'] // results in true
```

Telerik Academy

JavaScript Object Properties

Live Demo

Telerik Academy

Associative Arrays

- Objects can be used as associative arrays
  - The key (index) is string instead of number
    - Also called `dictionaries` or `maps`
- Associative arrays don't have array properties
  - `length`, `indexOf`, etc…

```
function countWords(words) {
  var word, i,
      wordsCount = {};
  for (i in words) {
    word = words[i].toLowerCase();
    if (!wordsCount[word]) { wordsCount[word] = 0; }
    wordsCount[word] += 1;
  }
  return wordsCount
}
```

Telerik Academy

Associative Arrays

Live Demo

Telerik Academy

Using Objects

http://academy.telerik.com

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com
- Telerik Software Academy
  - academy.telerik.com
- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy
- Telerik Software Academy Forums
  - forums.academy.telerik.com