Использование объектов

Что такое объекты?

- Программные объекты это модели объектов реального мира или абстрактные концепции
  - банк, банковский счет, клиент, собака, мотоцикл, очередь
- Объекты реального мира имеют **состояния** и **поведения**
  - Поля банковского счета:
    - держатель, баланс, тип
      - Методы банковского счета:
    - снять, положить, закрыть
- Каким образом программные объекты хранят параметры обьектов физического мира?
  - Используйте поля / данные / массивы для хранения состояний
  - Ипользуйте методы и функции для хранения поведений
- Объект может содержать в себе свойства и методы

Примеры объектов

- Вещи реального мира
  - людей
  - листы покупок
- Вещи компьютерного мира
  - числа
  - символы
  - очереди
  - массивы

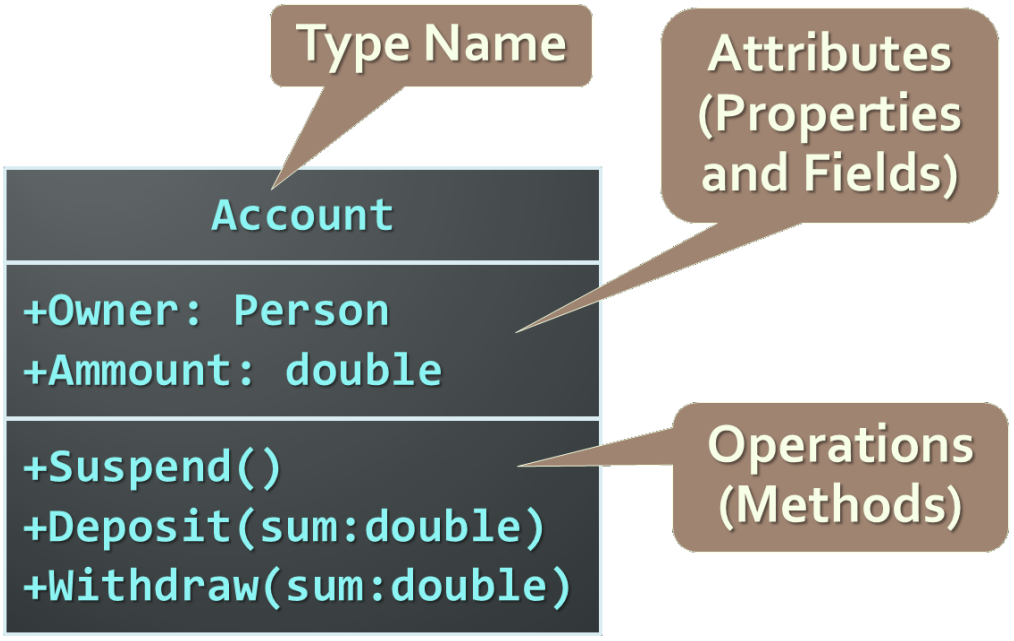
Что такое тип Object?

тип Object является шаблоном из которого пораждается объект во время выполнения программного кода.

тип Object

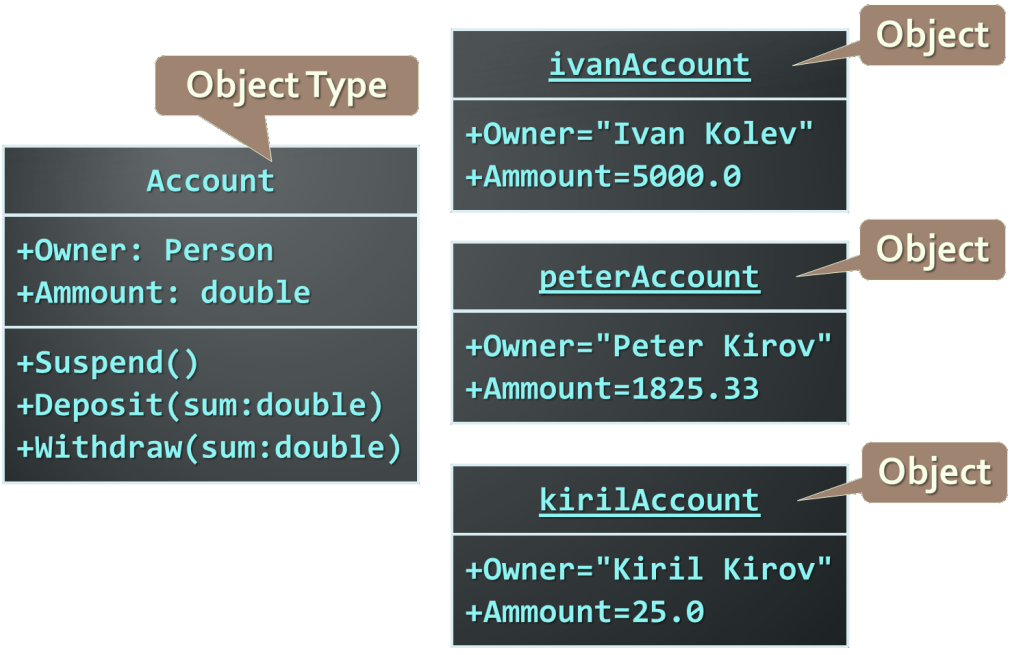
- тип Object определяет структуру для объекта
- Определяет ее прототип в качестве шаблона
- Тип Object определяет:
  - Установленные **атрибуты**

        - Представленные переменными и свойствами
        - Хранит их **состояние**
    - Определяет их действия называемые **поведением**
        - Представленные методами
- Определяет методы и типы данных ассоциированные с объектом



Объекты

- **object** это конкретный **экземпляр** относящийся к типу object
- object создается типом object путем порождения **экземпляра**
- Обьект имеет состояние
    - установленные значения для его атрибутов
    - *Type:* Счет
    - *Objects:* Счет Ивана, Счет Васи



Обзор объектов

- JavaScript спроектирован на простой обьектно-ориентированной парадигме
  - Обьект является совокупностью **свойств**
- Каждое свойство объекта имеет имя и содержит значение
  - значением свойства обьекта может быть **метод** (функция) или **поле** (переменная)
- В JS есть много предопределенных обьектов
  - `Math`, `document`, `window`, и др.
- Объект может быть создан разработчиком

Свойства объекта

- Каждое из **свойств** объекта
  - Свойства это значения ассоциированные с объектом
  - К свойствам обьекта можно получить доступ с помощью (`.` оператора) или `[]` - индекса:

```
let arrStr = arr.join(', '); // свойство массива join
let length = arr.length;  // свойство массива length
let words = text.split(' ');
let words = text['split'](' ');
```

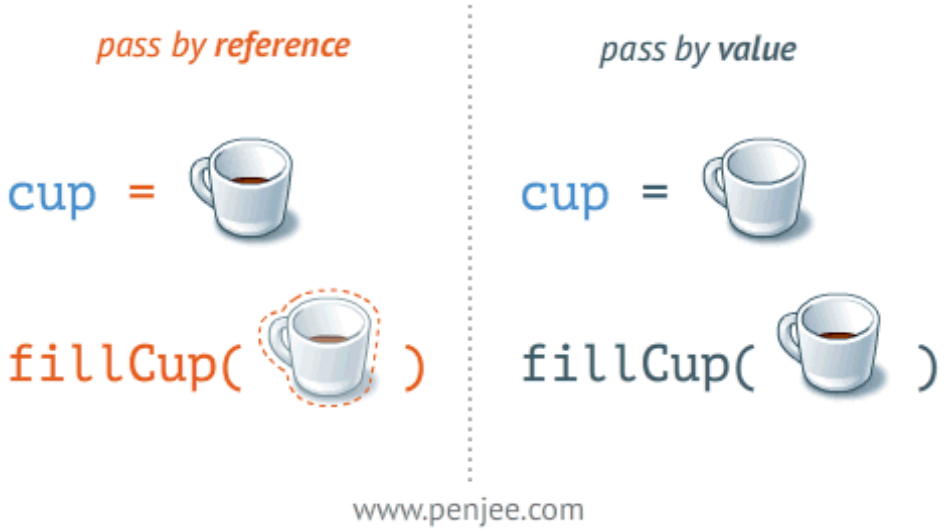
Ссылки и примитивные типы

- JavaScript **слабо типизированный** язык
  - Переменные не имеют типа, но их значения имеют
- JavaScript имеет **шесть** различных типов:
  - `Number`, `String`, `Boolean`, `Null`, `Undefined` и `Object`
- `Object` это только ссылочный тип
  - Он доступен как **ссылка** (всякий раз, когда происходит работа со значениями обьекта, это происходит через ссылки)
- `Number`, `String`, `Boolean`, `Null`, `Undefined` являются **примитивными** типами
  - Доступны как **значение** (они копируются всякий раз, когда используется их значение)
- Примитивные типы `Boolean`, `Number`, `String`, `Undefined` и `Null`
  - Все остальные типы являются обьектом `object`
    - Включая массивы, даты, другие типы..

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

- Все типы содержащие объект `Object`
  - Имеют тип `object`

Доступ по значению vs. Доступ по ссылке



Примитивные типы

Все типы данных в JavaScript, кроме объектов, являются иммутабельными (значения не могут быть модифицированы, а только перезаписаны новым полным значением). Например, в отличии от C, где строку можно по символьно корректировать, в JavaSctip строки пересоздаются только полностью. Значения таких типов называются "примитивными значениями".

- Примитивные типы доступны по **значению**
  - Когда происходит доступ к аргументу
    - Выделяется новая память
    - Значение копируется в память
    - Значение в памяти становится доступным
- Примитивные типы инициализируются с помошью литералов
- Примитивные типы имеют обьект **обертку**

```
let number = 5, // Содержит примитивное значение 5
    text = 'Hello there!', // Содержит примитивное значение
    numberObj = new Number(5); // Содержит объект со зачением 5
```

- Создать 2 строковых переменных

- Создать объект содержащий значения этих переменных
- Изменить значение переменной
- Каждый объект содержит свое значение

```
let fname = 'Вася',
    lname = 'Пупкин',
    person = { firstName: fname, lastName: lname };
lname = 'Петров';
console.log(person.lastName) // logged 'Пупкин'
```

Ссылочный тип

- `Object` является исключительно **ссылочным типом**
  - Когда осуществляется доступ к значению где-либо, оно не копируется, но передается по ссылке

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

let student = { name: 'Вася Пупкин', marks: marks };
marks[2].score = 5.50;

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

JavaScript обьект литерал

- JavaScript обьект литерал это простой способ создания объектов
  - Используются фигурные скобки:

```
let person = {
  firstName: 'Вася',
  lastName: 'Пупкин',
  toString: function () {
    return this.firstName + ' ' + this.lastName;
  }
};

// свойство обьекта могут быть использованы:
console.log(person.toString());
// выведет 'Вася Пупкин'
```

Создание объектов

- Давайте создадим 2х людей:

```
let Pupkin, Petrov;
pupkin = {
  fname: 'Вася',
  lname: 'Пупкин',
  toString: function() {
    return this.fname + ' ' + this.lname;
  }
};
petrov = {
  fname: 'Петя',
  lname: 'Петров',
  toString: function() {
    return this.fname + ' ' + this.lname;
  }
};
```

- Описывать объект приятно, но **повторять это** не очень хорошо?

Функции создания объектов

- Используйте функции для создания объектов
  - Просто передайте имя и фамилию и получите объект
    - Очень похоже на конструктор

```
let pupkin, petrov;
function makePerson(fname, lname) {
  return {
    fname: fname,
    lname: lname,
    toString: function () {
      return this.fname + ' ' + this.lname;
    }
  }
}
pupkin = makePerson('Вася', 'Пупкин');
petrov = makePerson('Петя', 'Петров');
```

- Так гораздо лучше, на правда ли?

JS свойства объекта

- JavaScript обьекты просто хранят пары ключ / значение
  - Каждое значение доступно по ключу
  - Свойства обьекта доступны по оператору точка (`obj.property`)
  - Те неменее доступ к свойствам возможен через фигурные скобки

    - Подобно массивам

```
document.write === document['write']
```

Ассоциативные массивы

- Обьекты могут применяться в качестве **ассоциативных массивов**
    - ключ (индекс) в качестве строки
    - Так же называются **словарями** или **картами**
- Ассоциативные массивы не имеют методов
    - `length` , `indexOf`

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