Homework Assignment 8

jan.schulz@devugees.org

1. Call and Apply

- Each function in JavaScript is an object, too.
- Each function has methods, like other objects, as well.
- As we learned before: **call()** and **apply()** are methods of functions
  - **call(object, argument1, argument2, ...)** calls a function and injects another **this**-variable as **object.**
  - **apply(object, [argument1, argument2, ...])** does the same as **call()** except it accepts an array of arguments instead of an argument list.

1. Call and Apply

1. Compile this code and analyze the object **gonzo** in the console. What did **call()** do?
2. Create a function in the object alfred setLastName(lastname) that attaches an attribute **lastname** to alfred and sets it to the parameter lastname.
3. Use **call()** again to borrow setLastName() on **gonzo** with the parameter „Gonzales“**.**

```
var alfred = {
	name: 'Alfred',
	count: 0,
	sayYourName: function() {
		if(this.count === undefined)
			this.count = 0;

		console.log( 'My name is ' + this.myName );
		this.count++;
	}
}

var gonzo = {
	myName: 'Gonzo'
}

alfred.sayYourName.call(gonzo);
```

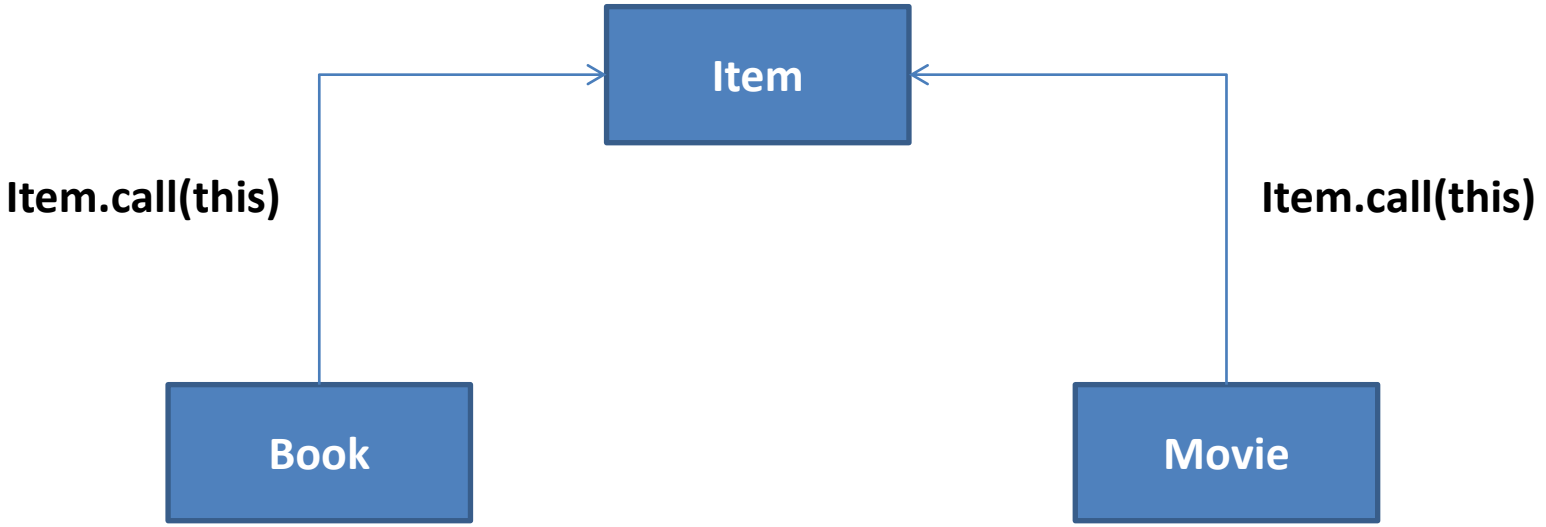
1. Call and Apply

```
var john = {
	name: 'john',
	age: 26,
	job: 'teacher',
	presentation: function(style, timeOfDay) {
		if(style === 'formal') {
			console.log('Good ' + timeOfDay
				+ ' Ladies and Gentlemen I am '
				+ this.name + ', I am a '
				+ this.job + ' and I am '
				+ this.age + ' years old.');
		}
		else if(style === 'friendly') {
			console.log('Hey whatsup.'
				+ 'I am '
				+ this.name + ', I am a '
				+ this.job + ' and I am '
				+ this.age + ' years old.'
				+ 'Have a nice ' + timeOfDay);
		}
	}
};

john.presentation('formal', 'morning');

var emily = {
	name: 'Emily',
	age: 35,
	job: 'designer'
};
```

1. Analyze this code and describe briefly what it does.
2. Use **call()** to use the function **presentation()** from the john-object on the emily-object with the parameters “friendly“ and “evening“.
3. Do (2) again with **apply()**.

2. Inheritance

call() and **apply()** can be used to borrow constructor functions from other objects, which can be used as parent-objects. We will use **call()** only.



2. Inheritance

PARENT

```
function Item( name, price ) {
    this.name = name;
    this.price = price;
    this.sold = false;
}

Item.prototype.sell = function() {
    this.sold = true;
}
```

CHILDREN

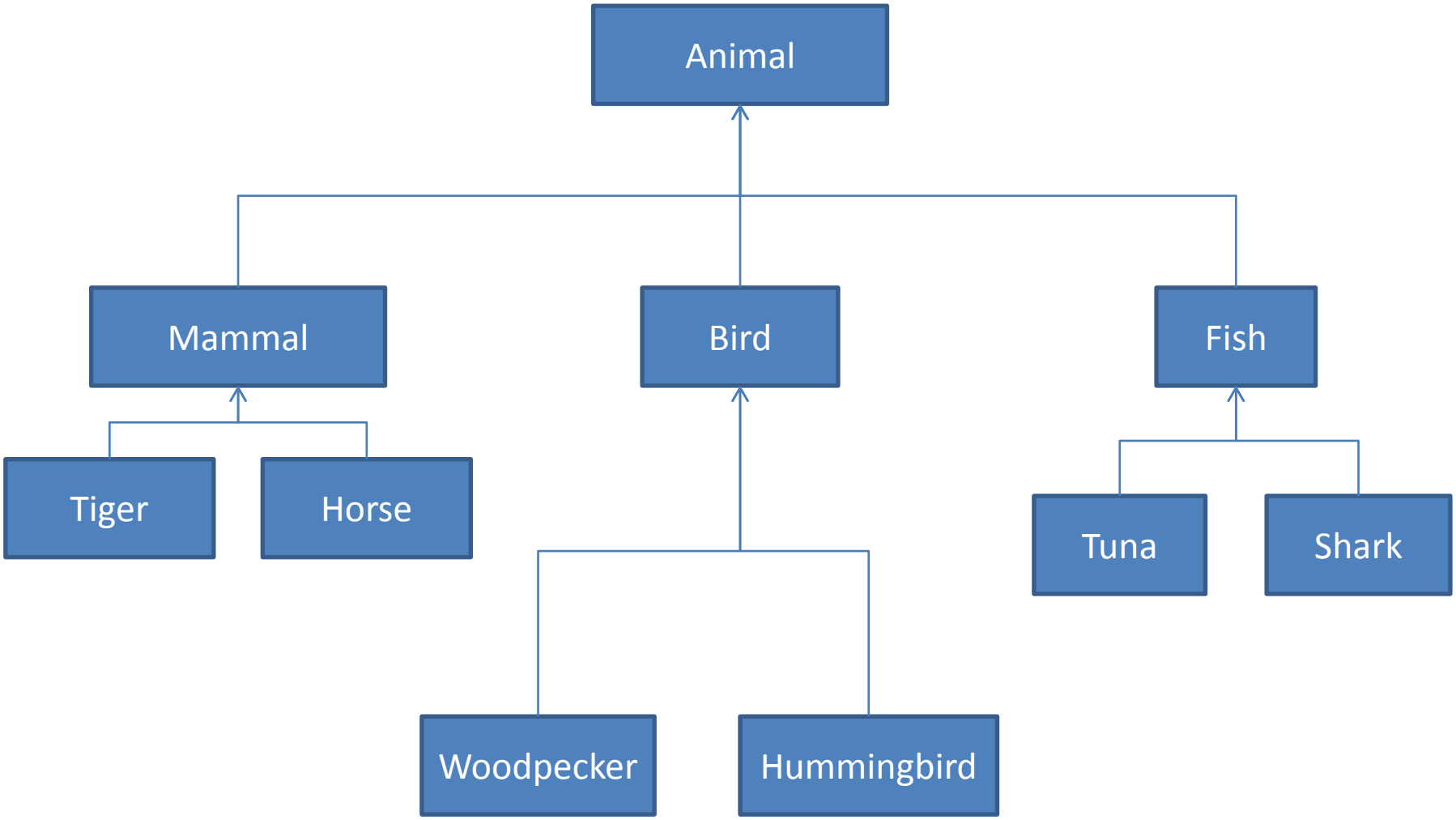
```
function Book( name, price, author ) {
        Item.call(this, name, price);
        this.author = author;
        this.category = 'book';
}

Book.prototype =
Object.create(Item.prototype);

function Movie( name, price, director ) {
        Item.call(this, name, price);
        this.director = director;
        this.category = 'movie';
}

Movie.prototype =
Object.create(Item.prototype);
```

2. Inheritance

1. Compile the code and analyze it. Note that **call()** is used to call the constructor of another object and note that Object.create() is used to create the object‘s prototype based of another object‘s prototype.
2. Create one movie “Casino“ from

   “Martin Scorsese“ and one book “IT“ from “Stephen King“.
2. Sell them both.
3. Create a new function constructo**r ComicBook** that inherits from **Book** and introduces a new attribute minAge which will set to 6 if it is undefined or less than 6.
4. Create the comic book “Jessica Jones“ from “Marvel“ with minAge 12.
5. Sell it.

3. Multi-Level Inheritance



3. Multi-Level Inheritance

1. With your knowledge from (2) Inheritance, please create the Function constructors according to the Animal diagram and consider the following rules:
    1. Each animal has a **name** that is set when it is constructed.
    2. All animals can **sleep**, **eat** and **die** (use functions for this, e.g. **sleep()**)
    3. Mammals and birds can **breathe**.
    4. Fishes can **swim**.
    5. Birds can **fly**.
    6. Tigers and Sharks can kill, whereas **kill()** expects one parameter **otherAnimal.** Kill() calls the die() function o**f otherAnimal**.
2. Create one tiger with name „Vitaly“, one Shark with name „Nemo“, one horse with name „Fury“.
3. Nemo is hungry and kills Fury and Vitaly. Then Nemo eats.
4. Nemo dies.