Aufgabe JavaScript

jan.schulz@devugees.org

1. Call and Apply

- Jede Funktion in JS ist auch ein Objekt!
- Also hat auch eine Funktion Methoden und Variablen
- Wir haben bereits gelernt: **call()** und **apply()** sind Methoden von Funktionen
  - **call(object, argument1, argument2, ...)** ruft eine Funktion auf und injeziert ein anderes **this**-Keyword in die Funktion
  - **apply(object, [argument1, argument2, ...])** macht das gleiche wie **call()** doch verwendet statt eine Parameterliste ein Array

1. Call and Apply

1. Führe diesen Code aus und analysiere das Objekt **gonzo** in der Konsole. Was hat **call()** gemacht?
2. Erstelle eine Funktion im Objekt alfred namens setLastName(lastname), welche ein neue Variable zu dem Objekt alfred hinzufügt namens **lastname** und es den Parameter lastname zuweist.
3. Benutze **call()** nochmal um setLastName() auf **gonzo** ausführen mit dem 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);
```

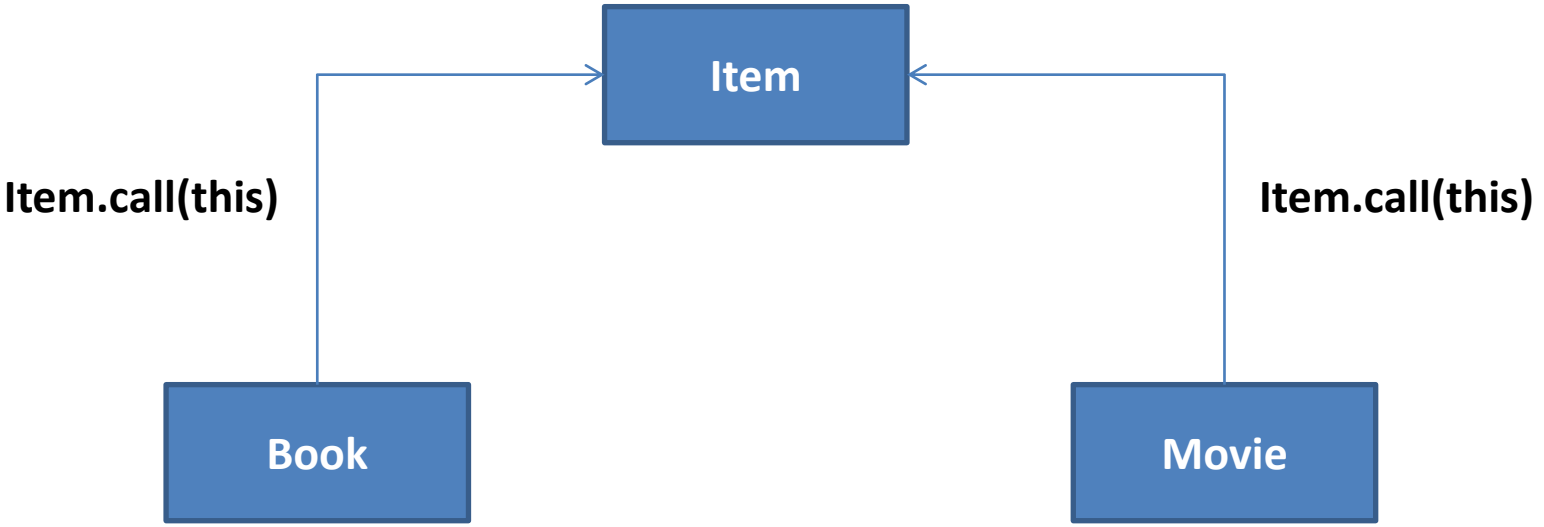
1. Call and Apply

```
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. Analysiere diesen Code kurz und beschreibe, was er tut.
2. Benutze **call()** um die Funktion **presentation()** vom john-Objekt auf dem emily-Objekt zu benutzen – mit den Parametern „friendly“ und „evening“.
3. Tue (2) nochmal mit **apply()**.

2. Vererbung

call() und **apply()** werden benutzt um Funktions-Konstruktoren von Objekten zu benutzen. Wir werden hier nur **call()** benutzen.



2. Vererbung

PARENT

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

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

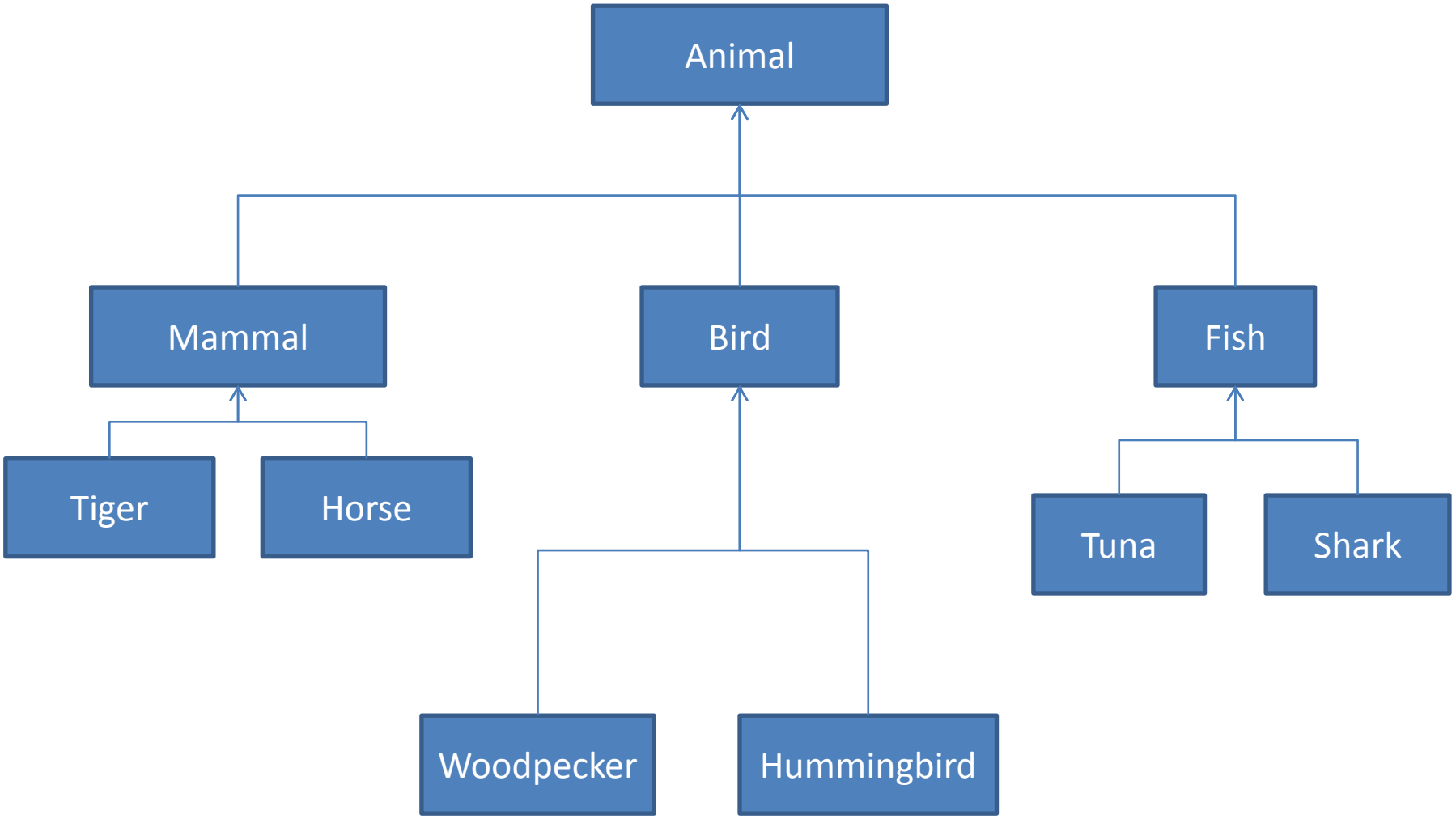
CHILDREN

```
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. Vererbung

1. Führe den Code aus und analysiere ihn. Beachte, dass **call()** benutzt wird, um den Funktions-Konstruktor eines anderen Objekts zu benutzen und dass Object.create() benutzt wird, um zwei Prototyp-Ketten miteinander zu verbinden.
2. Erstelle ein Movie-Objekt „Casino“ von „Martin Scorsese“ und ein Buch „ES“ von „Stephen King“.
2. Verkaufe sie beide.
3. Erstelle einen neuen Funktions-Konstruktor **ComicBook**, der **Book** vererbt and erstelle eine neue Variable minAge die auf 6 gesetzt wird, falls der Parameter minAge undefined oder kleiner als 6 ist.
4. Erstelle das Comic-Buch „Jessica Jones“ von „Marvel“ mit minAge gleich 12.
5. Verkaufe es.

3. Mehr-Stufige Vererbung



3. Mehr-Stufige Vererbung

1. Mit Deinem Wissen aus der Aufgabe 2: Erstelle Funktions-Konstruktoren zu dem Animal-Diagramm und beachte dabei folgende Regeln:
    1. Jedes Animal hat einen **name** wenn es erstellt wird.
    2. Jedes Animal hat die Methoden **sleep**, **eat** und **die** (benutze dafür Funktionen, z.B. **sleep()**)
    3. Mammals (Säugetiere) und birds haben die Methode **breathe**.
    4. Fishes haben die Methode **swim**.
    5. Birds haben die Methode **fly**.
    6. Tigers und Sharks haben die Methode kill, wobei **kill()** einen Parameter **otherAnimal** erwartet**.** kill() ruft die die() Funktion von **otherAnimal** auf.
2. Erstelle einen Tiger mit name „Vitaly“, einen Shark mit name „Nemo“ und ein Horse mit name „Fury“.
3. Nemo ist hungrig und tötet Fury und Vitaly. Dann isst Nemo.
4. Nemo stirbt.

4. Class Syntax

- Übersetze Deine Lösungen in die neue Ecma-Script 2016 Syntax mit class und extends!