Javascript -> Class & extends

zz

```
class Human {
  constructor(name) {
    this.name = name;
  }

  say() {
    console.info(`I am ${this.name}`);
  }

  static cry() {
    console.info('crying ~');
  }
}

class Dancer extends Human {
  constructor(name) {
    super(name);
  }

  dance() {
    console.info(`${this.name} is dancing ~`);
  }
}

class SuperMan extends Human {
  constructor(name, level) {
    super(name);
    this.level = level;
  }

  say() {
    console.info(`I am ${this.name}, whose level is ${this.level}`);
  }

  fly() {
    console.info(`${this.name} is flying ~ so cool.`);
  }
}
```

```
const man = new Human('葫芦娃');
const dancer = new Dancer('小明');
const superman = new SuperMan('小魔仙', 10);

console.info('man ######################');
man.say();
Human.cry();
console.info('dancer ######################');
dancer.say();
dancer.dance();
Dancer.cry();
console.info('superman ######################');
superman.say();
superman.fly();
SuperMan.cry();
```

```
man ######################
I am 葫芦娃
crying ~
dancer ######################
I am 小明
小明 is dancing ~
crying ~
superman ######################
I am 小魔仙, whose level is 10
小魔仙 is flying ~ so cool.
crying ~
```

```
class Human {
  constructor(name) {
    this.name = name;
  }

  say() {
    console.info(`I am ${this.name}`);
  }

  static cry() {
    console.info('crying ~');
  }
}
```

```
'use strict';

var _createClass = function () {
  function defineProperties(target, props) {
    for (var i = 0; i < props.length; i++) {
      var descriptor = props[i];
      descriptor.enumerable = descriptor.enumerable || false;
      descriptor.configurable = true;
      if ("value" in descriptor)
        descriptor.writable = true;
      Object.defineProperty(target, descriptor.key, descriptor);
    }
  }
  return function (Constructor, protoProps, staticProps) {
    if (protoProps)
      defineProperties(Constructor.prototype, protoProps);
    if (staticProps)
      defineProperties(Constructor, staticProps);
    return Constructor;
  };
}();
var Human = function () {
  function Human(name) {
    this.name = name;
  }
  _createClass(Human, [
    {
      key: 'say',
      value: function say() {
        console.info('I am ' + this.name);
      }
    }
  ], [
    {
      key: 'cry',
      value: function cry() {
        console.info('crying ~');
      }
    }
  ]);
  return Human;
}();
```

The prototype is a property on a constructor function that sets what will become the __proto__ property on the constructed object.

__proto__, [[prototype]] and prototype

```
class Human {
  constructor(name) {
    this.name = name;
  }

  say() {
    console.info(`I am ${this.name}`);
  }

  static cry() {
    console.info('crying ~');
  }
}

class Dancer extends Human {
  constructor(name) {
    super(name);
  }

  dance() {
    console.info(`${this.name} is dancing ~`);
  }
}
```

```
var Human = function () {
  function Human(name) {
    this.name = name;
  }
  _createClass(Human, [{
    key: 'say',
    value: function say() {
      console.info('I am ' + this.name);
    }
  }], [{
    key: 'cry',
    value: function cry() {
      console.info('crying ~');
    }
  }]);
  return Human;
}();
var Dancer = function (_Human) {
  _inherits(Dancer, _Human);
  function Dancer(name) {
    return _possibleConstructorReturn(
      this,
      (Dancer.__proto__ ||
        Object.getPrototypeOf(Dancer)).call(this, name)
    );
  }
  _createClass(Dancer, [{
    key: 'dance',
    value: function dance() {
      console.info(this.name + ' is dancing ~');
    }
  }]);
  return Dancer;
}(Human);
```

```
var Human = function () {
  function Human(name) {
    this.name = name;
  }
  _createClass(Human, [{
    key: 'say',
    value: function say() {
      console.info('I am ' + this.name);
    }
  }, [{
    key: 'cry',
    value: function cry() {
      console.info('crying ~');
    }
  }]);
  return Human;
}();
var Dancer = function (_Human) {
  _inherits(Dancer, _Human);
  function Dancer(name) {
    return _possibleConstructorReturn(
      this,
      (Dancer.__proto__ ||
        Object.getPrototypeOf(Dancer)).call(this, name)
    );
  }
  _createClass(Dancer, [{
    key: 'dance',
    value: function dance() {
      console.info(this.name + ' is dancing ~');
    }
  }]);
  return Dancer;
}(Human);
```

```
Dancer.prototype.__proto__ === Human.prototype // true
```

```
function _inherits(subClass, superClass) {
  if (typeof superClass !== "function" && superClass !== null) {
    throw new TypeError("Super expression must either be null
or a function, not " + typeof superClass);
  }
  subClass.prototype = Object.create(superClass && superClass.prototype, {
    constructor: {
      value: subClass,
      enumerable: false,
      writable: true,
      configurable: true
    }
  });
  if (superClass)
    Object.setPrototypeOf
      ? Object.setPrototypeOf(subClass, superClass)
      : subClass.__proto__ = superClass;
}
```

```
Dancer.__proto__ === Human // true
```

```
function _possibleConstructorReturn(self, call) {
  return call && (typeof call === "object" || typeof call === "function")
    ? call
    : self;
}
```

```
class SuperMan extends Human {
  constructor(name, level) {
    super(name);
    this.level = level;
  }

  say() {
    console.info(`I am ${this.name},
      whose level is ${this.level}`);
  }

  fly() {
    console.info(`${this.name} is flying ~ so cool.`);
  }
}
```

```
var SuperMan = function (_Human2) {
  _inherits(SuperMan, _Human2);

  function SuperMan(name, level) {
    var _this2 = _possibleConstructorReturn(this, (SuperMan.__proto__ ||
      Object.getPrototypeOf(SuperMan)).call(this, name));
    _this2.level = level;
    return _this2;
  }

  _createClass(SuperMan, [{
    key: 'say',
    value: function say() {
      console.info('I am ' + this.name + ', whose level is ' + this.level);
    }
  }, {
    key: 'fly',
    value: function fly() {
      console.info(this.name + ' is flying ~ so cool.');
    }
  }]);

  return SuperMan;
}(Human);
```

Conclusion

```
class A {
}

class B extends A {
}

B.__proto__ === A // true

B.prototype.__proto__ === A.prototype // true

this
```

END