JAVASCRIPT PATTERN

ECMA SCRIPT 5

THE IMMEDIATE INVOKED FUNCTION EXPRESSION

THE IMMEDIATE INVOKED FUNCTION EXPRESSION

```
(function () {
    'use strict';
    // - - - - - - - - - - -
    var onLoad = null;

    onLoad = function () {
        console.log('window loaded!');
    };

    window.addEventListener('load', onLoad)
   // - - - - - - - - - - -
}());
```

ECMA SCRIPT 5

THE MODULE BLOCK PATTERN

THE MODULE BLOCK PATTERN

```
(function () {
    'use strict';
    // - - - - - - - - - - -
    var fn = {},
        a = 42;

    fn.log = function (message) {
        console.log(message);
    };

    fn.alert = function () {};

    window.myNamespace = window.myNamespace || {};
    window.myNamespace.myModuleName = fn;
    // - - - - - - - - - - -
})();
```

INHERITANCE

```
(function () {
    // - - - - - - - - - - -
    var fn = {};

    fn.log = function (message) {
        console.log(message);
    };

    console.log('Tut was in Module 2!');

    window.app = window.app || {}; // Defaultoperator
    window.app.myModule2 = fn;

    window.app.myModule1.log('Aufruf an 1 aus 2');

    // - - - - - - - - - - -
}());
```

ECMA SCRIPT 5

SINGLETON

SINGLETON

```
var mySingleton = (function () {

  // Instance stores a reference to the Singleton
  var instance;

  function init() {

    // Singleton
    // Private methods and variables
    function privateMethod(){
        console.log( "I am private" );
    }
    var privateVariable = "Im also private";
    var privateRandomNumber = Math.random();

    return {
      // Public methods and variables
      publicMethod: function () {
        console.log( "The public can see me!" );
      },
      publicProperty: "I am also public",

      getRandomNumber: function() {
        return privateRandomNumber;
      }
    };

};
```

SINGLETON

```
return {

  // Get the Singleton instance if one exists
  // or create one if it doesn't
  getInstance: function () {

    if ( !instance ) {
      instance = init();
    }

    return instance;
  }

};

})();
```

„ https://addyosmani.com/resources/
essentialjsdesignpatterns/book/
#constructorpatternjavascript“

ECMA SCRIPT 5

CONSTRUCTOR PATTERN

OBJEKT ALS KONSTRUKTOR

```
function Person(firstName, lastName) {
  this.firstName = firstName;
  this.lastName = lastName;

  this.fullName = function() {
    return this.firstName + ' ' + this.lastName;
  };
}

// Use it like this:
var john = new Person('John', 'Doe');
john.firstName;    // "John"
john.fullName();   // "John Doe"

john.firstName = 'John';
john.fullName();   // John Doe
```

KONSTRUKTOR OHNE THIS

```
function Person(firstName, lastName) {
  var _firstName = firstName,
      _lastName = lastName;

  var my = {
    firstName: _firstName,
    lastName: _lastName
  };

  my.fullName = function() {
    return _firstName + ' ' + _lastName;
  };
```

KONSTRUKTOR OHNE THIS

```
// Getter/setters

  my.firstName = function(value) {
    if (!arguments.length) return _firstName;
    _firstName = value;

    return my;
  };

  my.lastName = function(value) {
    if (!arguments.length) return _lastName;
    _lastName = value;

    return my;
  };

  return my;
}
```

KONSTRUKTOR OHNE THIS

```
// Use it like this:

var chuck = Person('Chuck', 'Norris');

chuck.firstName('Jackie');
chuck.lastName('Chan');

chuck.fullName(); // Jackie Chan
```

„http://www.samselikoff.com/blog/some-Javascript-constructor-patterns/“

ECMA SCRIPT 5

INHERTITANCE PATTERN

PROTOTYPES IN KONSTRUKTOREN

```
var Shipment = function (state, type) {
  var
    _state = undefined,
    _type  = undefined,
  endvar;

  this.state = state || undefined;
  this.type  = type  || undefined;

  this.setState = function () {};
};
```

PROTOTYPES IN KONSTRUKTOREN

```
// Nachträglich hinzugefügte Methoden
Shipment.prototype.setState = function (value) {
 this.state = value;
};

Shipment.prototype.setType  = function (value) {
 this.type  = value;
};

Shipment.prototype.save     =
  function () {
    console.log( 'saving shipment '
                 + this.state + ', '
                 +   this.type + '.');
  };
```

```
var shipment = [];

shipment[0] = new Shipment(3, 1);
shipment[1] = new Shipment();

shipment[0].save();

shipment[1].setState(4);
shipment[1].setType(2);

shipment[1].save();
```

INHERITANCE

```
var ShipmentRainbow =
function (state, type, typeOfDoc,reference){
  this.state      = state || undefined;
  this.type       = type  || undefined;
  this.typeOfDoc  = typeOfDoc  || 'HTML';
  this.reference  = reference  || 'Hallo';
};

ShipmentRainbow.prototype   = new Shipment();
ShipmentRainbow.constructor = ShipmentRainbow;
```

INHERITANCE

```
var shipmentRainbow = new
ShipmentRainbow(3,1,'typeOfDocText','referenceText');

shipmentRainbow.save();


console.dir(shipmentRainbow);

console.log( shipmentRainbow instanceof Shipment );
console.log( shipmentRainbow instanceof ShipmentRainbow );
```

ECMA SCRIPT 5

CHAINING PATTERN

CHAINING PATTERN

```
(function() {
  'use strict';
  // - - - - - - - - - - -

// define the class
let _fn = function() {
  this.version = '0.1';
  this.os = 'MacOS X';
  this.browser = 'Chrome';
};
```

CHAINING PATTERN

```
_fn.prototype.setVersion = function(version) {
  this.version = version;
  return this;
};

_fn.prototype.setOs = function(os) {
  this.os = os;
  return this;
};

_fn.prototype.setBrowser = function(browser) {
  this.browser = browser;
  return this;
};

_fn.prototype.save = function() {
  console.log(
    'saving ' + this.version + ', on ' +
    this.os + ' with ' + this.browser + '.'
  );
  return this;
};
```

CHAINING PATTERN

```
Let fn = new _fn();

fn.setVersion('1');
fn.setOs('MacOS XI');
fn.setBrowser('Chrome Xtra');
fn.save();

fn
  .setVersion('2')
  .setOs('MacOS XII')
  .setBrowser('Chrome Xtra Large')
  .save();
```

ECMA SCRIPT 6

KLASSEN

CLASS

```
class SimpleDate {
  constructor(year, month, day) {
    // Check that (year, month, day) is a valid date
    // ...

    this._year = year;
    this._month = month;
    this._day = day;
  }

  addDays(nDays) {
    // Increase "this" date by n days
    // ...
  }

  getDay() {
    return this._day;
  }
}
```

INSTANTIIERUNG

```
let today = new SimpleDate(2000, 2, 28);

today.addDays(1);
```

PRIVATE PROPERTIES

```
class SimpleDate {
  constructor(year, month, day) {
    let _year = year;
    let _month = month;
    let _day = day;

    // Methods defined in the constructor
    // capture variables in a closure

    this.addDays = function(nDays) {
      // Increase "this" date by n days
      // ...
    }

    this.getDay = function() {
      return _day;
    }
  }
}
```

STATIC PROPERTIES AND METHODS

```
class SimpleDate {
  static setDefaultDate(year, month, day) {
   SimpleDate._defaultDate = new SimpleDate(year, month, day);
  }

  constructor(year, month, day) {
   if (arguments.length === 0) {
      this._year = SimpleDate._defaultDate._year;
      this._month = SimpleDate._defaultDate._month;
      this._day = SimpleDate._defaultDate._day;

      return;
    }

    // Check that (year, month, day) is a valid date
    // ...

    this._year = year;
    this._month = month;
    this._day = day;
  }

  addDays(nDays) {
    // Increase "this" date by n days
    // ...
  }

  getDay() {
    return this._day;
  }
}
```

```
SimpleDate.setDefaultDate(1970, 1, 1);
let defaultDate = new SimpleDate();
```

INHERITANCE

```
class Employee {
  constructor(firstName, familyName) {
    this._firstName = firstName;
    this._familyName = familyName;
  }

  getFullName() {
    return `${this._firstName} ${this._familyName}`;
  }
}

class Manager {
  constructor(firstName, familyName) {
    this._firstName = firstName;
    this._familyName = familyName;
    this._managedEmployees = [];
  }

  getFullName() {
    return `${this._firstName} ${this._familyName}`;
  }

  addEmployee(employee) {
    this._managedEmployees.push(employee);
  }
}
```

INHERITANCE

```
class Employee {
  constructor(firstName, familyName) {
    this._firstName = firstName;
    this._familyName = familyName;
  }

  getFullName() {
    return `${this._firstName} ${this._familyName}`;
  }
}
```

INHERITANCE

```
class Manager extends Employee {
  constructor(firstName, familyName) {
    super(firstName, familyName);
    this._managedEmployees = [];
  }

  addEmployee(employee) {
    this._managedEmployees.push(employee);
  }
}
```

„ https://www.sitepoint.com/object-oriented-javascript-deep-dive-es6-classes/“