

Abstract Factory

```
function ClassFactory() {
  this.type = 'class'
  this.logType = function () {
    console.log('Type', this.type)
  }
}

function WarriorClass() {
  ClassFactory.call(this)
  this.subtype = 'warrior'
  this.logSubtype = function () {
    console.log('Subtype', this.subtype)
  }
}

function WizardClass() {
  ClassFactory.call(this)
  this.subtype = 'wizard'
  this.logSubtype = function () {
    console.log('Subtype', this.subtype)
  }
}

const $warrior = new WarriorClass()
$warrior.logType()
$warrior.logSubtype()
```

Type class
Subtype warrior

Dynamic Function From String

```
global.runFoo = function runFoo() {
  if (argument.length) {
    let args = []
    for (let i = 0; i < arguments.length; i++) {
      args.push(argument[i])
    }
    console.log('executed runFoo(' + args.join(' ') + ')')
  } else {
    console.log('executed runFoo()')
  }
}

const fn = 'runFoo'
const fnParams = [1,2,3]
global[fn]()
global[fn]
```

executed runFoo()
executed runFoo(1,2,3)

Functional Interface (Interface Segregation)

```
global.oven = function Oven() {
  this.on = false
  this.turnOn = function () {
    this.on = true
  }

  this.turnOff = function () {
    this.on = false
  }

  this.cook = function (item) {
    console.log(`${item} in the oven`)
  }
}
```

Functional Interface (Interface Segregation) (cont)

```
> }
}
// function Stove() { // ES6
global.stove = function Stove() {
  this.on = false
  this.turnOn = function () {
    this.on = true
  }
  this.turnOff = function () {
    this.on = false
  }
  this.cook = function (item) {
    console.log(Cooking ${item} in the stove)
  }
}
function ICooker(cooker) {
  // let fn = window[cooker]; // ES6
  let fn = global[cooker] // Node.js
  return new fn()
}
function Restaurant(name, cooker) {
  this.name = name
  this.cooker = new ICooker(cooker)
  this.cook = function (item) {
    this.cooker.turnOn()
    this.cooker.cook(item)
    this.cooker.turnOff()
  }
}
```

Functional Interface (Interface Segregation) (cont)

```
> }
}
const cookerTypes = {
  OVEN: 'oven',
  STOVE: 'stove',
}
const bakery = new Restaurant('Bakery', cookerTypes.OVEN)
bakery.cook('cookies')
const creperery = new Restaurant('Creperery', cookerTypes.STOVE)
creperery.cook('crepes')
```

Cooking cookies in the oven

Cooking crepes in the stove

Functional Constructor

```
function Robot(name, job) {
  this.name = name
  this.job = job
  this.introduce = function () {
    console.log(this.name)
    console.log(this.job)
  }
}
const walle = new Robot('Wall-E', 'clean')
console.log(walle instanceof Robot)
walle.introduce()
```

true

Name Wall-E

Job clean



By **graypes**

cheatography.com/graypes/

Published 24th August, 2022.

Last updated 24th August, 2022.

Page 2 of 3.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

Functional Inheritance (Liskov Substitution)

```
const Scientist = {
  init: function (name) {
    this.name = name
  },
  run_experiment: function () {
    console.log(this.name) is running e
xperiment)
  },
}

const MadScientist = {
  ...Scientist,
  run_experiment: function () {
    const sabotage = !!Math.floor (Ma -
th.random() * 2)
    if (sabotage) {
      console.log(this.name) is sab
otaging experi ment!)
    } else {
      console.log(this.name) is run
ning experiment)
    }
  },
}

const neil = Object.create(Scientist)
neil.init ('Neil')
const hubert = Object.create(MadScientist)
hubert.init ('Hubert')
neil.run_experiment()
hubert.run_experiment()
```

Neil will always return "Neil is running experiment"
 Hubert will possibly return "Hubert is running experiment" or "Hubert is sabotaging experiment!"

Open/Close Principle

```
function announce(collection) {
  console.log( collection.description)
  collection.logItems()
}

var favoriteCities = {
  items: {
    Denmark: 'Copenhagen',
    Uganda: 'Kampala',
    Uruguay: 'Montevideo',
  },
  description: 'My favorite cities around the
world:',
  logItems: function () {
    Object.keys(this.items).for -
Each(function (key) {
      console.log(this.items[key])
    }, this)
  },
}

announce(favoriteCities)
```

My favorite cities around the world:
 Copenhagen
 Kampala
 Montevideo

