Object Composition

Closures, Revealing Module Pattern, Object Inheritance, Prototypes

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Content

Have a Question?

SoftUni Foundation

sli.do

#JSCORE

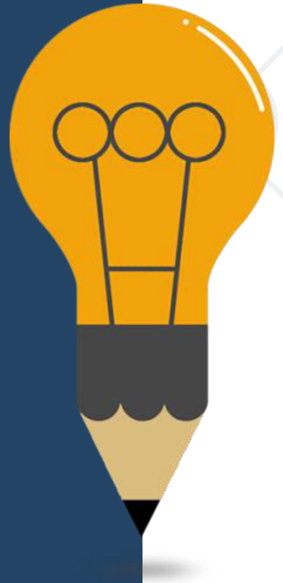
Object Composition

Objects Holding Other Objects

What is Object Composition?

- **Object composition** == combining simple objects or data types into more complex ones

```
let student = {
  firstName: 'Maria',
  lastName: 'Green',
  age: 22,
  location: { lat: 42.698, lng: 23.322 }
}
console.log(student);
console.log(student.location.lat);
```



Composing Objects

SoftUni Foundation

```
let name = "Sofia";
let population = 1325744;
let country = "Bulgaria";
let town = { name, population, country };
console.log(town); // Object {name: "Sofia", population: 1325744, country: "Bulgaria"}
```

Combine variables into object

```
town.location = { lat: 42.698, lng: 23.322 };
console.log(town); // Object {…, location: Object}
```

Combining Data with Functions

```
let rect = {
  width: 10,
  height: 4,
  grow: function(w, h) {
    this.width += w; this.height += h;
  },
  print: function() {
    console.log(`[${this.width} x ${this.height}]`);
  }
};
rect.grow(2, 3);
rect.print(); // [12 x 7]
```

Printing Objects: toString() Function

```
let rect = {
  width: 10,
  height: 4,
  toString: function() {
    return `rect[${this.width} x ${this.height}]`;
  }
};
console.log(rect); // Object {width: 10, height: 4}
// This will invoke toString() to convert the object to String
console.log('' + rect); // rect[12 x 7]
```

Problem: Order Rectangles by Size

- You are given a set of rectangles (**width** x **height**) as nested arrays
- **Order** them by their **area**, then by **width** (descending)

```
[3, 4], [5, 3], [3, 4], [3, 5], [12, 1]
```

```
[5, 3], [3, 5], [12, 1], [3, 4], [3, 4]
```

```
[2, 2.5], [2.5, 2]
```

```
[2.5, 2], [2, 2.5]
```

Solution: Order Rectangles by Size

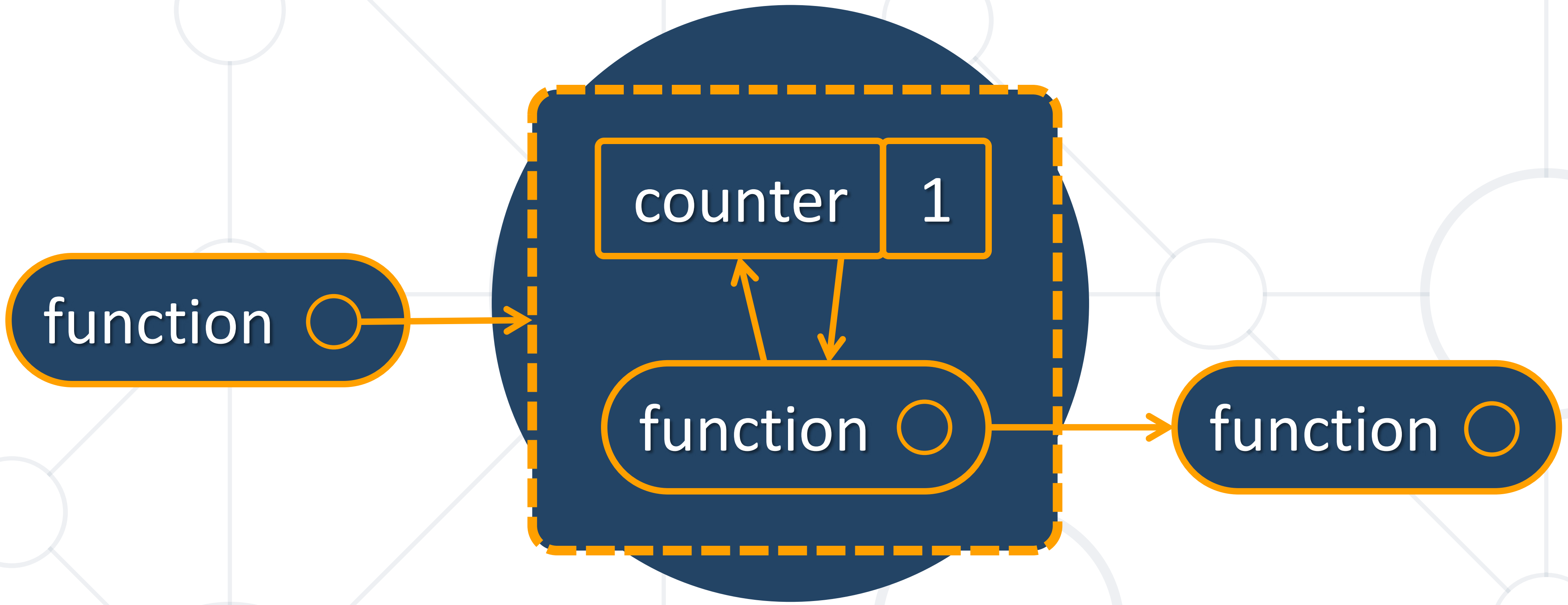
```
function createRect(width, height) {
  let rect = {
    width: width,
    height: height,
    area: () => rect.width * rect.height,
    compareTo: function(other) {
      let result = other.area() - rect.area();
      return result || (other.width - rect.width);
    }
  };
  return rect;
}
```

Solution: Order Rectangles by Size (2)

```
function orderRects(rectsData) {
  let rects = [];
  for (let [width, height] of rectsData) {
    let rect = createRect(width, height);
    rects.push(rect);
  }
  rects.sort((a,b) => a.compareTo(b));
  return rects;
}
```

```
orderRects([[3, 4], [5, 3], [3, 4], [3, 5], [12, 1]])
```

Check your solution here: https://judge.softuni.bg/Contests/334



Closures

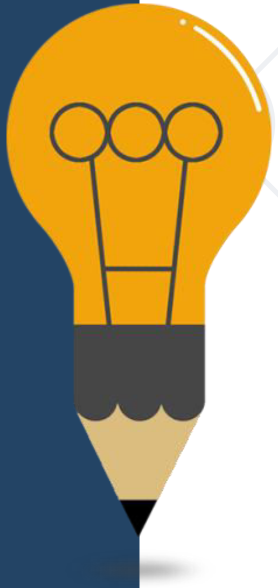
Enclosing Object State in a Function

What is Closure?

- Closure == state maintained (closed) inside a function
  - Hidden from the outside world
- Example: counter with closures

```
function counterClosure() {
  let counter = 0;
  function getNextCount() {
    console.log(++counter);
  };
  return getNextCount;
}
```

```
let count =
  counterClosure();
count(); // 1
count(); // 2
count(); // 3
count(); // 4
count(); // 5
```



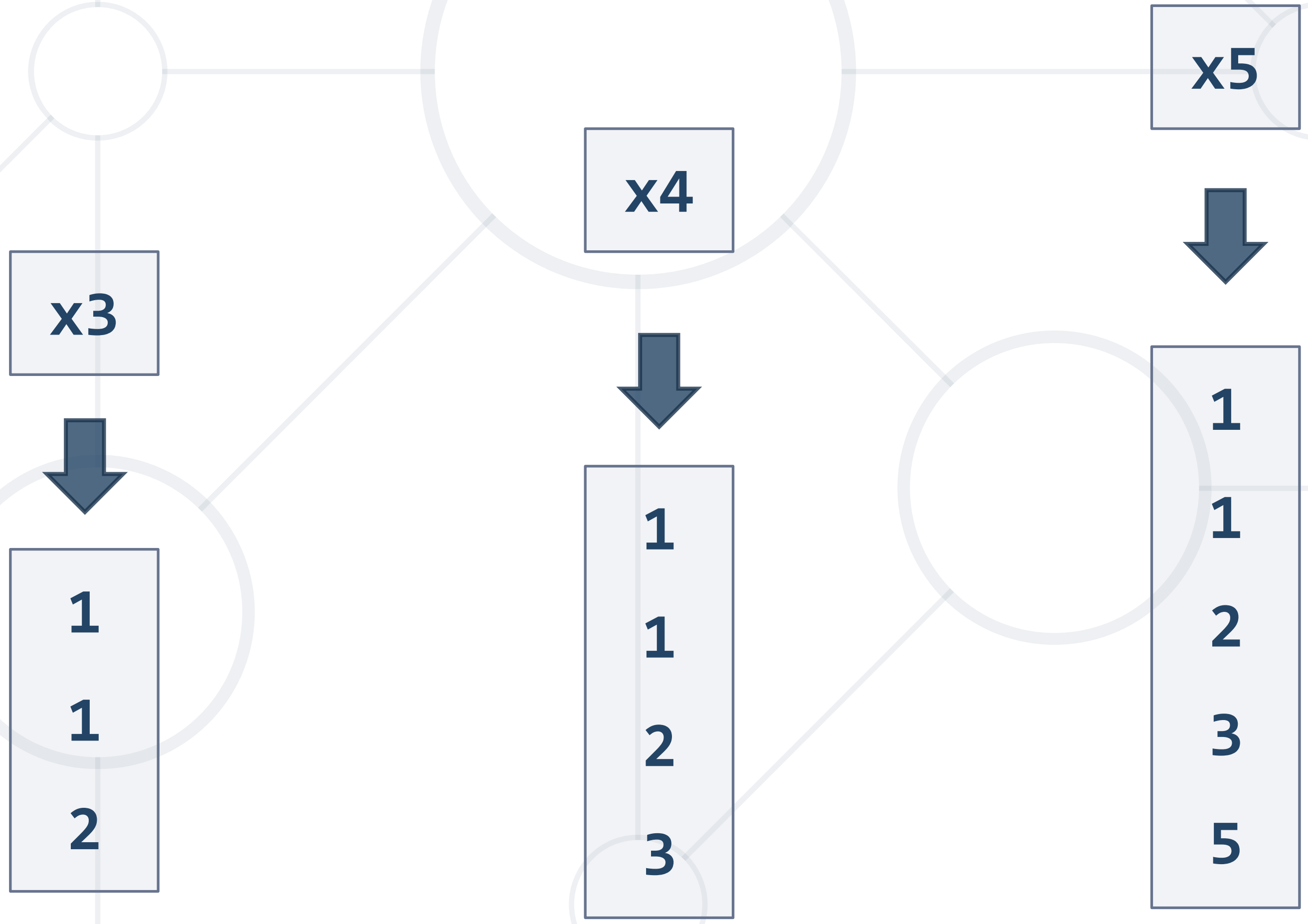
Closures – Shorter Syntax with IIFE

```
let counter = (function() {
  let num = 0;
  return function() {
    console.log(++num);
  };
})();
counter(); // 1
counter(); // 2
counter(); // 3
```

```
let counter = (() => {
  let num = 0;
  return () =>
    console.log(++num);
})();
counter(); // 1
counter(); // 2
counter(); // 3
counter(); // 4
```

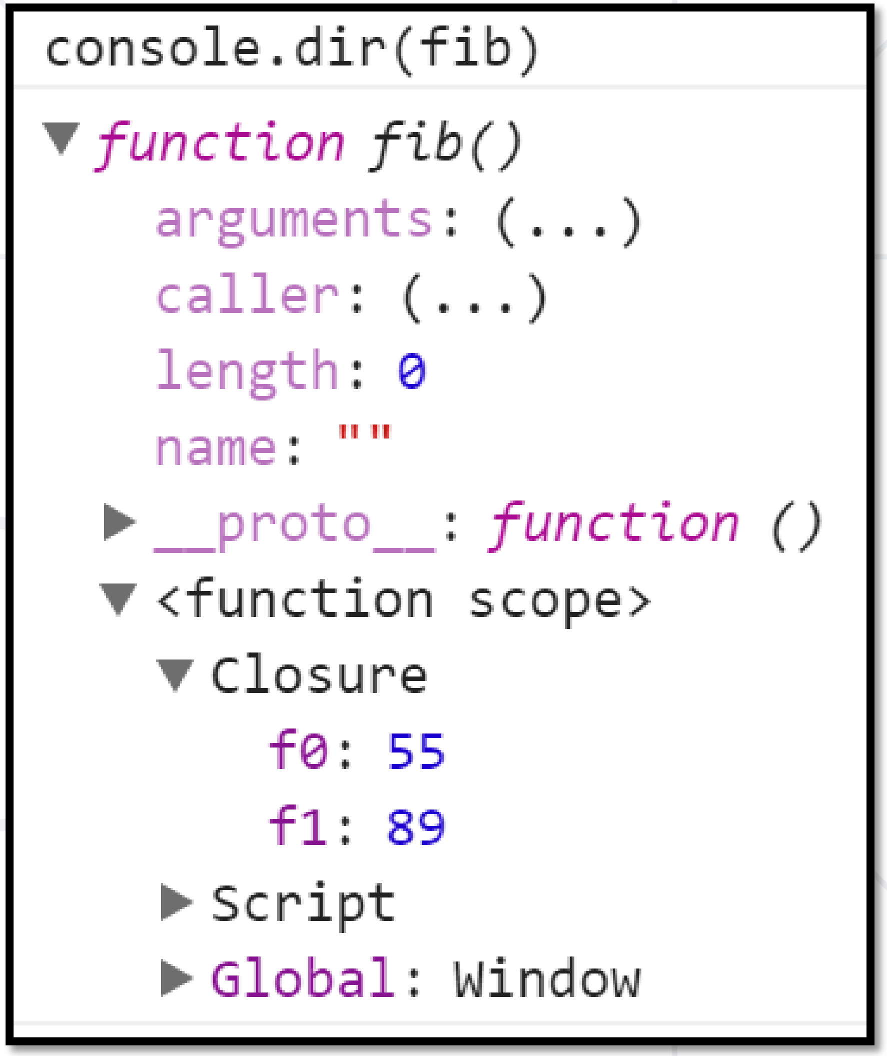
Problem: Fibonacci with Closures

- Using closures write a JS function that returns the next **Fibonacci** number, each time it's called



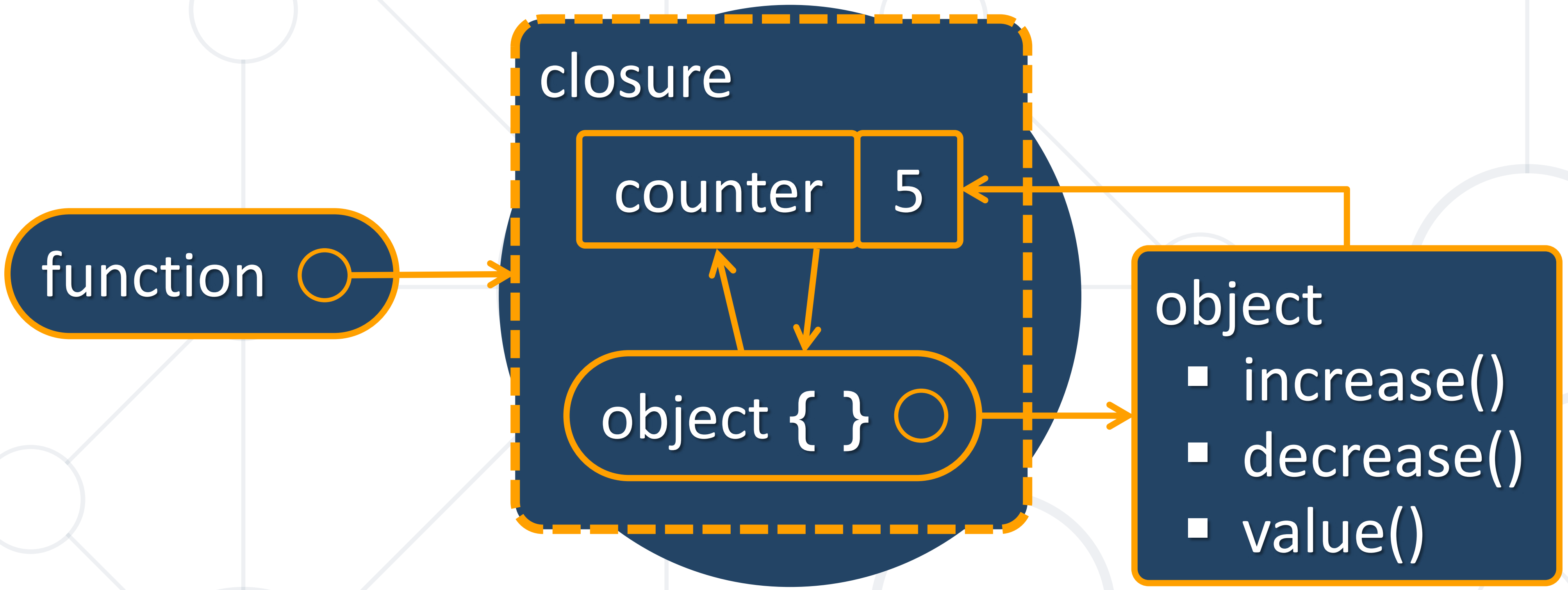
Solution: Fibonacci with Closures

```
function getFibonator() {
  let f0 = 0, f1 = 1;
  return function() {
    let f2 = f0 + f1;
    f0 = f1;
    f1 = f2;
    return f1;
  };
}
let fib = getFibonator();
fib(); // 1
fib(); // 1
fib(); // 2
```



```
console.dir(fib)
▼ function fib()
    arguments: (...)
    caller: (...)
    length: 0
    name: ""
  ▶ __proto__: function ()
  ▼ <function scope>
    ▼ Closure
        f0: 55
        f1: 89
    ▶ Script
    ▶ Global: Window
```

Check your solution here: https://judge.softuni.bg/Contests/334



Module and Revealing Module Patterns

"Module" Pattern (with Object Literal)

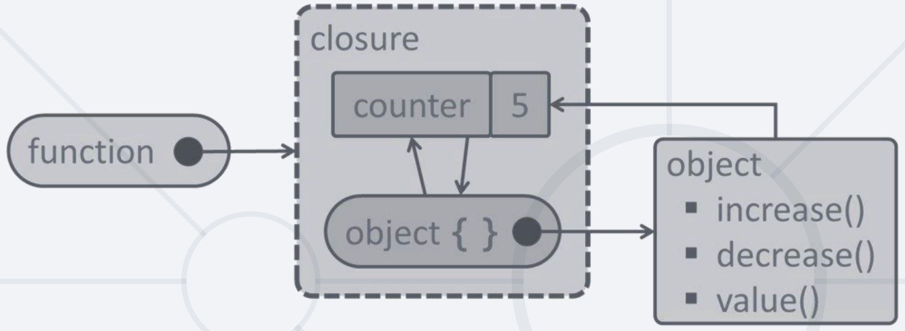
```
let moduleObj = {
  count: 0, // public
  increase: function(num) { return this.count += num },
  decrease: function(num) { return this.count -= num },
  value: function() { return this.count }
};

moduleObj.count = 2; // the counter is accessible
console.log(moduleObj.value()); // 2
console.log(moduleObj.increase(5)); // 7
console.log(moduleObj.decrease(1)); // 6
```

object
- count
- increase()
- decrease()
- value()

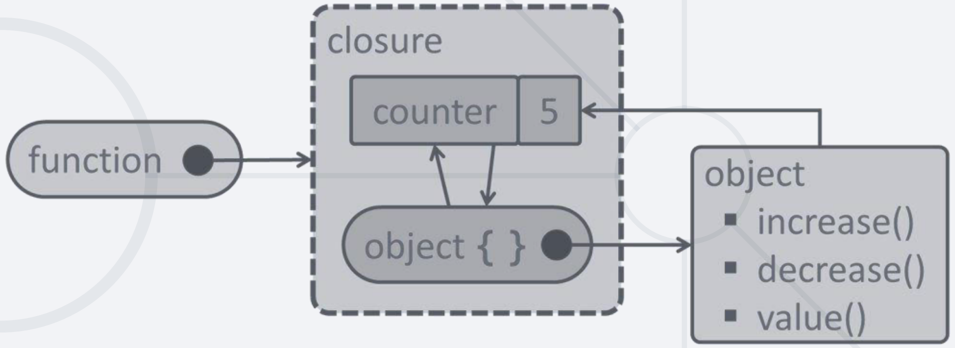
"Module" Pattern (with Closure)

```
let module = (function() {
  let count = 0; // private
  return {
    increase: (num) => count += num,
    decrease: (num) => count -= num,
    value: () => count,
  };
})();
console.log(module.value()); // 0
console.log(module.increase(5)); // 5
console.log(module.decrease(2)); // 3
console.log(module.count); // undefined (not accessible)
```



"Revealing Module" Pattern (with Closure)

```javascript
let revModule = (function() {
  let count = 0; // private
  function change(amount) { return count += amount; }
  function increase(num) { return change(num); }
  function decrease(num) { return change(-num); }
  function value() { return count; }
  return { increase, decrease, value };
})();
console.log(revModule.value()); // 0
console.log(revModule.increase(5)); // 5
console.log(revModule.decrease(2)); // 3
console.log(module.count); // undefined (not accessible)
```



Problem: List Processor

- Using a **closure** (IIFE holding a state inside it) implement a command execution engine to **process list commands** like shown below

```
add hello
add again
remove hello
add again
print
```

```
again, again
```

```
hello
```

```
hello, again
```

```
again
```

```
again, again
```

```
again, again
```

Solution: List Processor

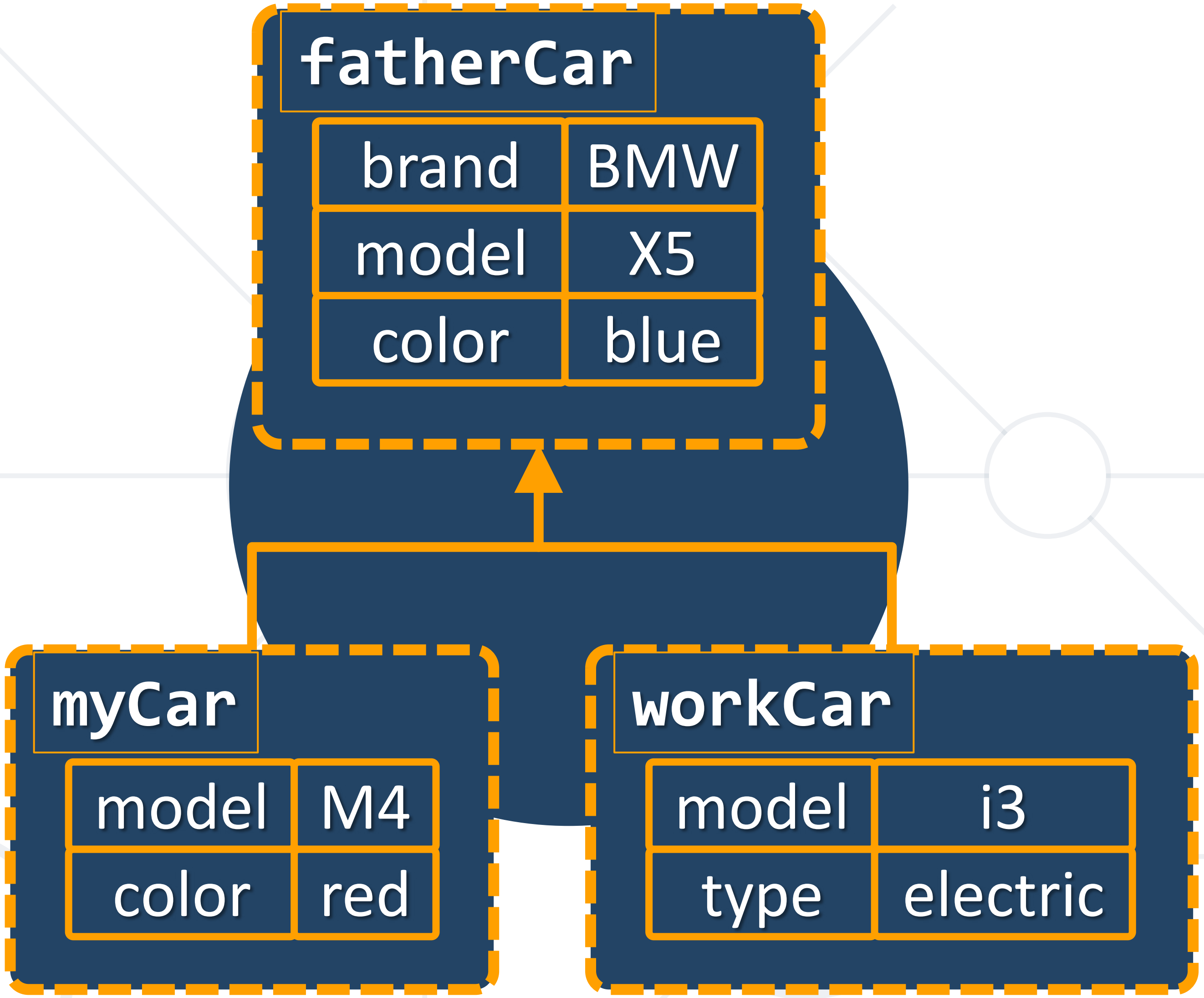
```
let commandProcessor = (function() {
  let list = [];
  return {
    add: (newItem) => list.push(newItem),
    remove: (item) => list = list.filter(x => x != item),
    print: () => console.log(list)
  }
})();
```

Solution: List Processor (2)

```
function processCommands(commands) {
  let commandProcessor = (function(){ … })();
  for (let cmd of commands) {
    let [cmdName, arg] = cmd.split(' ');
    commandProcessor[cmdName](arg);
  }
}
```

```
processCommands(['add hello', 'add again', 'remove hello',
'add again', 'print']);
```

Check your solution here: https://judge.softuni.bg/Contests/334



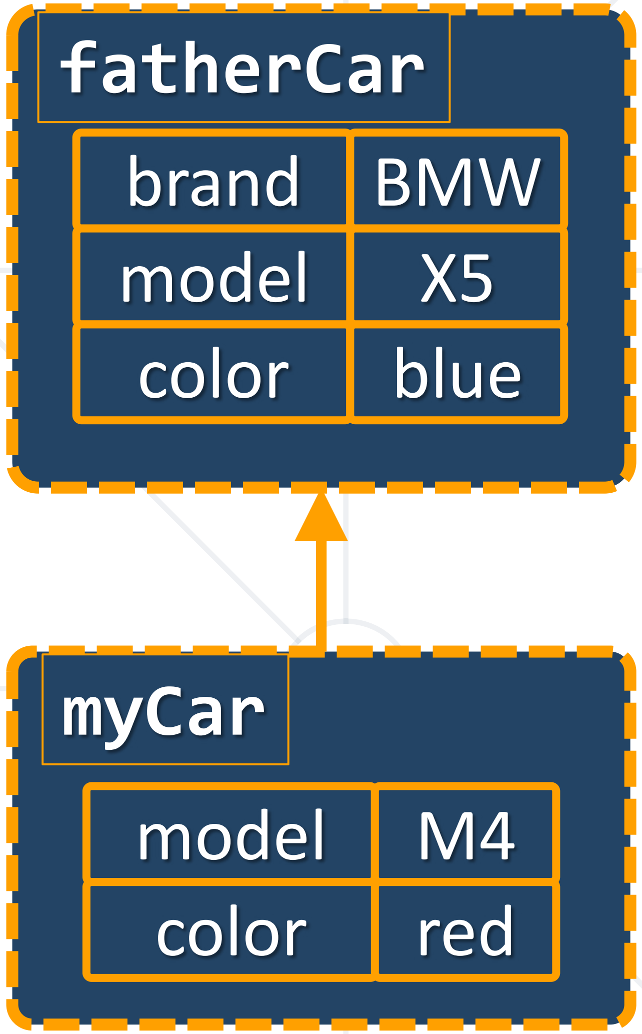
Object Inheritance

Object Inheritance

```
let fatherCar = { brand: 'BMW',
  model: 'X5', color: 'blue',
  toString: function() { return `[brand:
    ${this.brand}, model: ${this.model},
    color: ${this.color}]`; }
};
console.log('' + fatherCar);
```

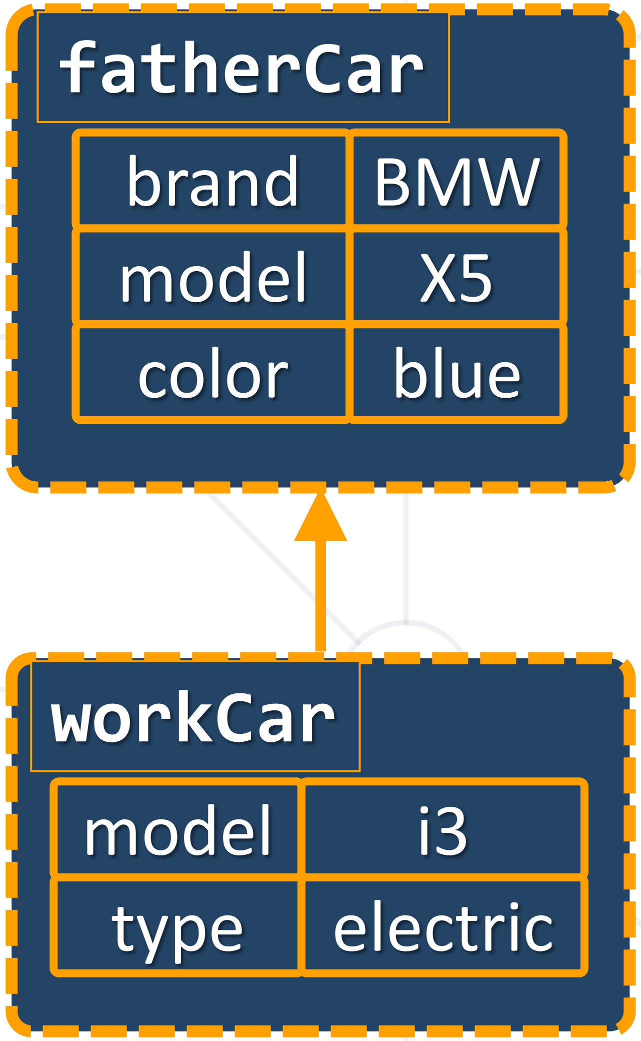
```
let myCar = Object.create(fatherCar);
myCar.model = 'M4';
myCar.color = 'red';
console.log('' + myCar);
```

Object.create() inherits an object

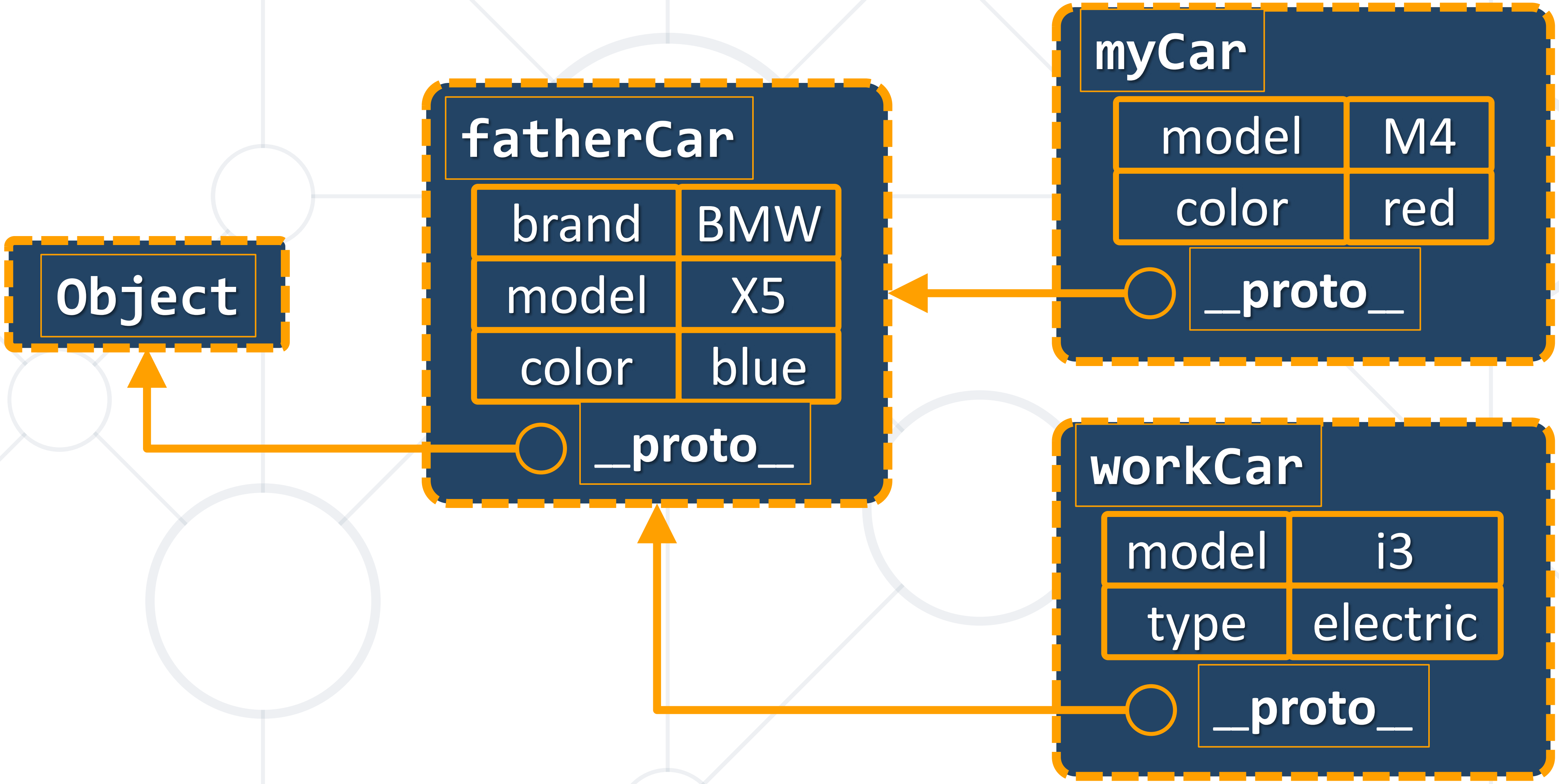


Object Inheritance (2)

```
let workCar =
  Object.create(fatherCar);
workCar.model = 'i3';
workCar.type = 'electric';
workCar.toString = function() {
  return `[brand: ${this.brand}, model:
    ${this.model}, color: ${this.color},
    type: ${this.type}]`;
}
console.log('' + workCar);
```



Prototype Chain



Prototype Chain

- Objects have **prototype** (a parent object)
  - Prototypes form a **prototype chain**

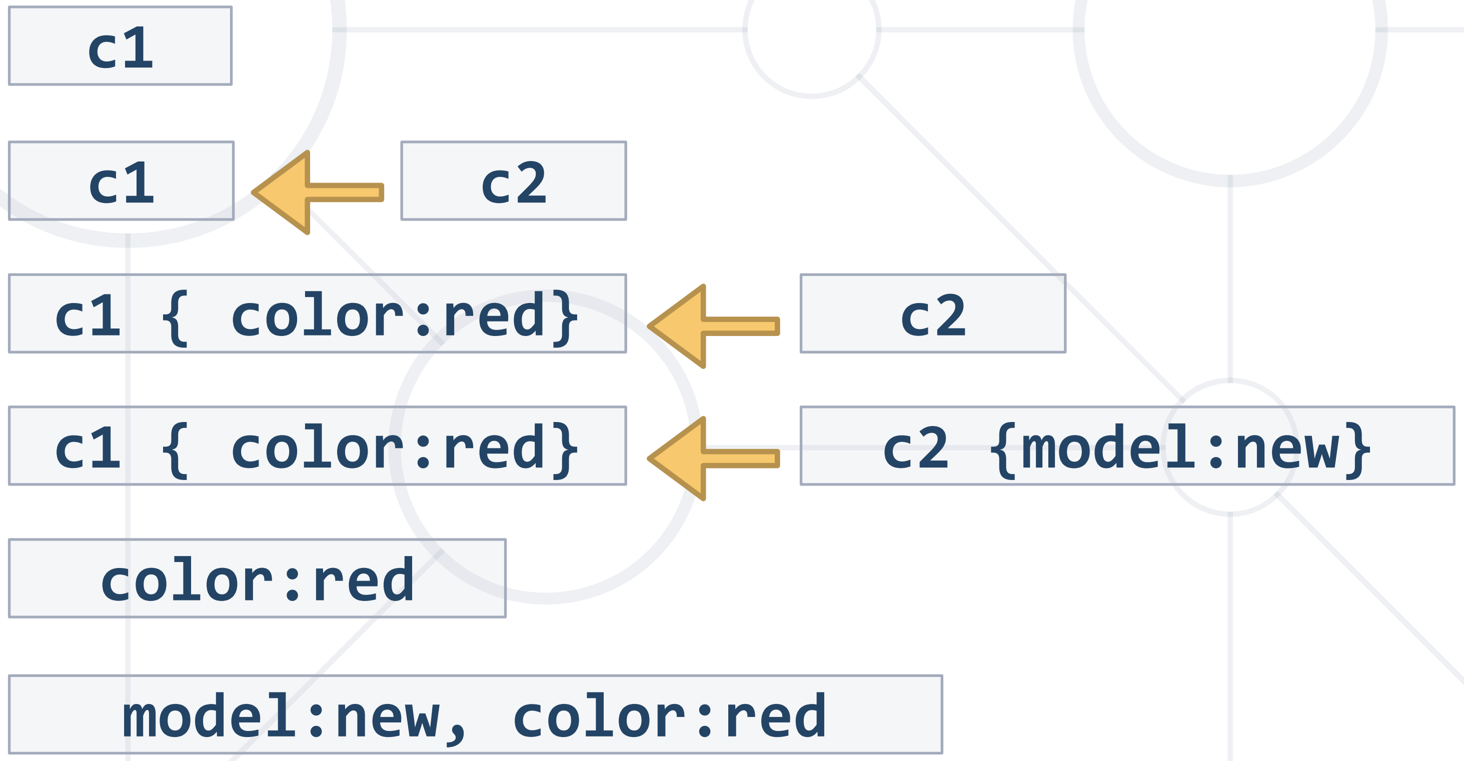
```
Object.getPrototypeOf(fatherCar);
// Object {}
Object.getPrototypeOf(myCar);
// Object {brand: "BMW", model: "X5", color: "blue"}
```

  - If a property is not found in the object itself, it is searched in the parent objects (in the prototype chain)

Problem: Object Inheritance

- Write a JS function to execute commands which create, inherit and modify objects:

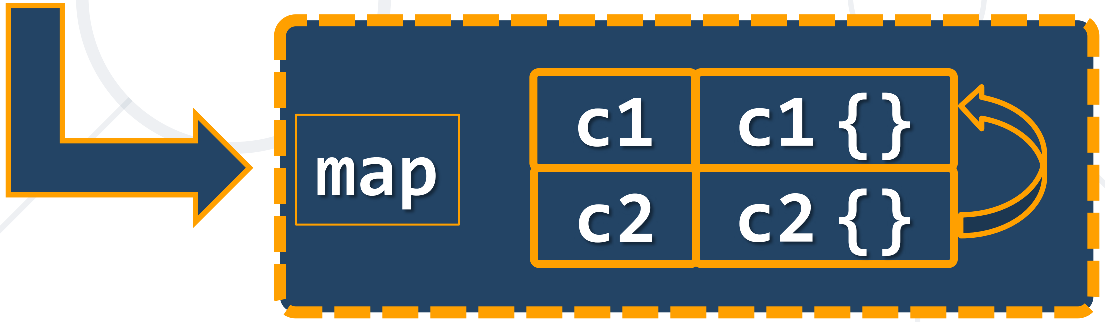
```
create c1
create c2 inherit
c1
set c1 color red
set c2 model new
print c1
print c2
```



Solution: Object Inheritance – Parser

```
function processCommands(commands) {
  let map = new Map();
  let cmdExecutor = { … };
  for (let command of commands) {
    let commandParameters = command.split(' ');
    let action = commandParameters.shift();
    cmdExecutor[action](commandParameters);
  }
}
```

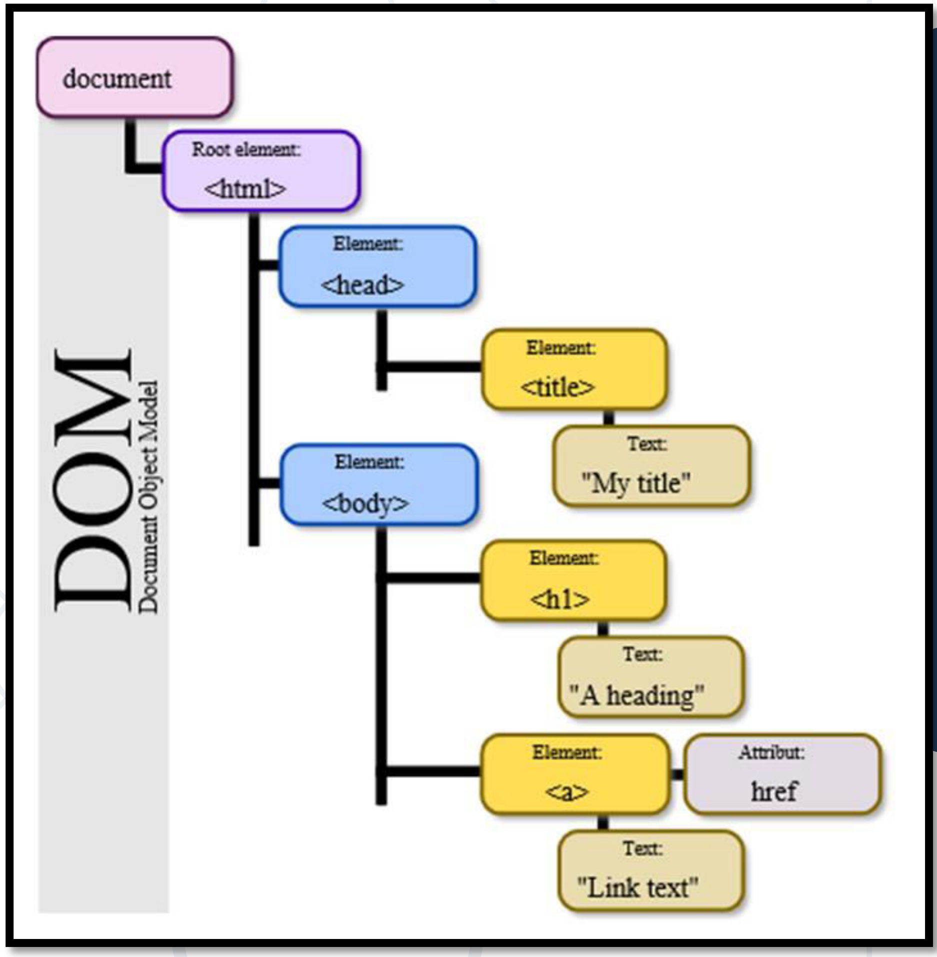
```
processCommands(['create c1','create c2 inherit c1'])
```

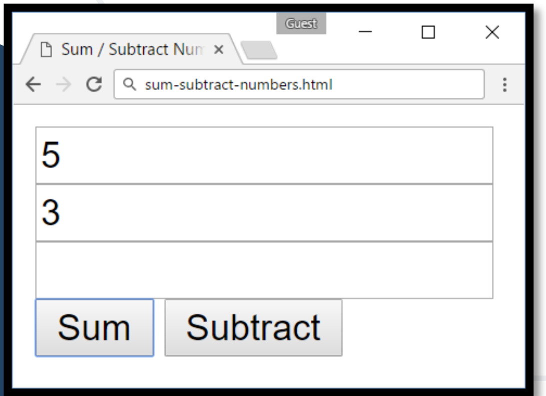


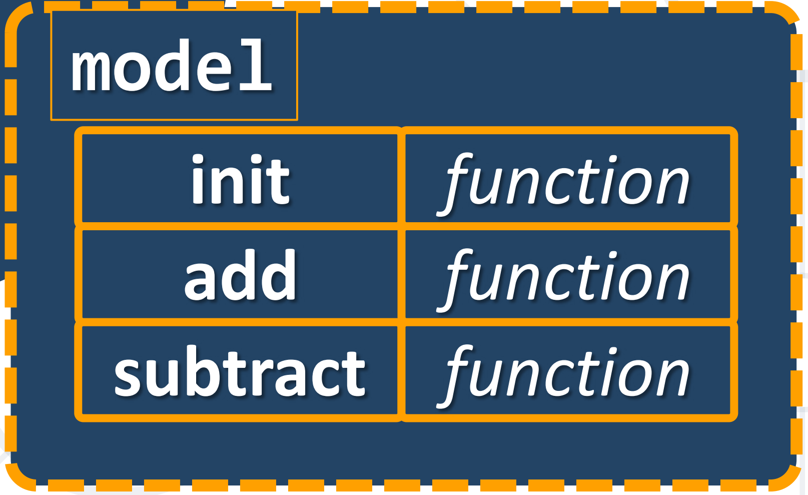
Solution: Object Inheritance – Executor

```
let cmdExecutor = {
  create: function ([objName, inherits, parent]) {
    parent = parent ? map.get(parent) : null;
    let newObj = Object.create(parent);
    map.set(objName, newObj);
    return newObj;
  },
  set: function ([objName, key, value]) {
    let obj = map.get(objName);
    obj[key] = value;
  },
  print: function ([objName]) {
    let obj = map.get(objName), objects = [];
    for (let key in obj) { objects.push(`${key}:${obj[key]}`); }
    console.log(objects.join(', '));
  }
};
```

Check your solution here: https://judge.softuni.bg/Contests/334





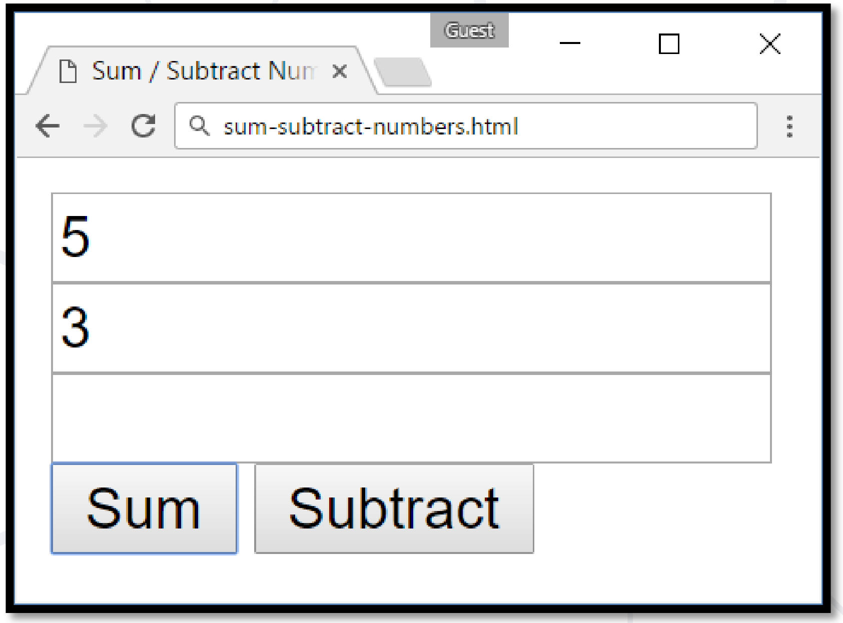


Objects Interacting with DOM

Problem: Sum / Subtract Numbers

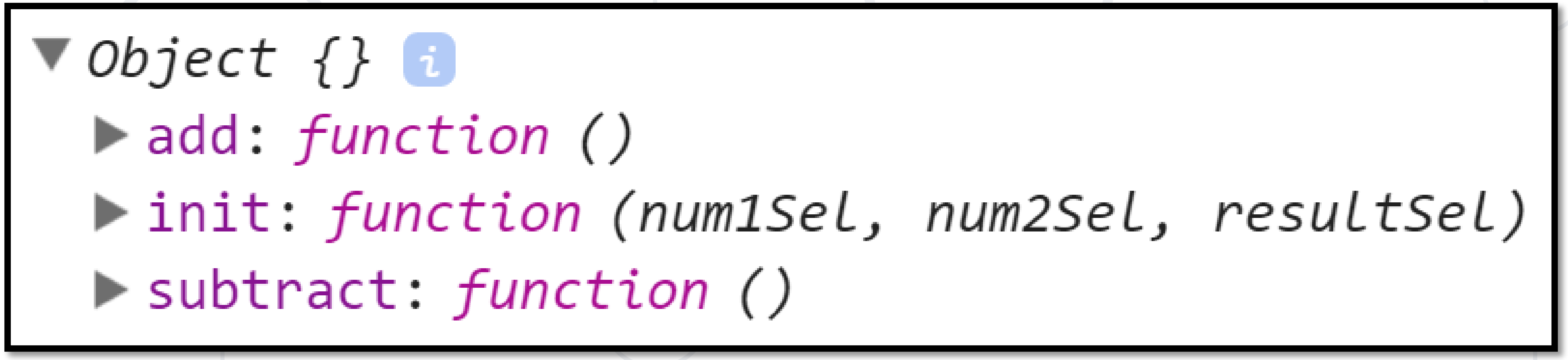
- You are given the following HTML form

```
<input type="text" id="num1" />
<input type="text" id="num2" />
<input type="text" id="result"
readonly />
<br>
<button id="sumButton">
Sum</button>
<button id="subtractButton">
Subtract</button>
```



Problem: Sum / Subtract Numbers (2)

- Write a JS function **getModel()** to return a JS object holding
  - function **init**(**num1Sel**, **num2Sel**, **resultSel**) → initializes selectors for finding the fields **num1**, **num2** and **result** in the **DOM**
  - function **add**() → calculates **result** = **num1** + **num2**
  - function **subtract**() → calculates **result** = **num1** - **num2**



```
▼ Object {}
  ▶ add: function ()
  ▶ init: function (num1Sel, num2Sel, resultSel)
  ▶ subtract: function ()
```

Problem: Sum / Subtract Numbers (3)

- This is how the **getModel()** function can be used to implement **add / subtract** operatons in the DOM tree:

```
$(function() {
  let model = getModel();
  model.init('#num1', '#num2', '#result');
  $('#sumButton').click(model.add);
  $('#subtractButton').click(model.subtract);
});
```

Solution: Sum / Subtract Numbers

```
// Solution using the "Module" pattern

function getModel() {
  let model = {
    init: function(num1Sel, num2Sel, resultSel) {
      model.num1 = $(num1Sel);
      model.num2 = $(num2Sel);
      model.result = $(resultSel);
    },

    add: () => model.action((a, b) => a + b),

    subtract: () => model.action((a, b) => a - b),
```

Solution: Sum / Subtract Numbers (2)

```
    action: function(operation) {
      let val1 = Number(model.num1.val());
      let val2 = Number(model.num2.val());
      model.result.val(operation(val1, val2));
    }
  };

  return model;
}
```

Check your solution here: https://judge.softuni.bg/Contests/334

Another Solution: Sum / Subtract Numbers

```
// Solution using the "Revealing Module" pattern

function getModel() {
  let num1, num2, result;

  function init(num1Sel, num2Sel, resultSel) {
    num1 = $(num1Sel);
    num2 = $(num2Sel);
    result = $(resultSel);
  }

  function add() { action((a, b) => a + b); }

  function subtract() { action((a, b) => a - b); }
```

Another Solution: Sum / Subtract Numbers (2)

```
  function action(operation) {
    let val1 = Number(num1.val());
    let val2 = Number(num2.val());
    result.val(operation(val1, val2));
  }

  let model = { init, add, subtract };
  return model;
}
```

Check your solution here: https://judge.softuni.bg/Contests/334

Live Exercises

Summary

- Object composition combines data and functions into JS objects

```
let r = {w:5, h:3, grow:function() { … }}
```

- The "Module" pattern hides data into a function and reveals a JS object
- The "Revealing Module" pattern hides data and functions are reveals them as JS object
- Objects can inherit parent object by **Object.create(parent)**

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech we know sports

telenor

SoftwareGroup doing it right

NETPEAK

SmartIT

Postbank Решения за твоето утре

SUPER HOSTING ®.BG

INDEAVR Serving the high achievers

INFRAGISTICS®

LIEBHERR

æternity

codexio

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

WORLD OF MYTHS

Lukanet.com

codexio

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License