**Інкапсуляція у об’єктах**

Оголошені у конструкторі локальні змінні і локальні вкладені функції доступні тільки в контексті (всередині, *private*) функції-конструктора, а ті властивості і методи, які оголошені в контексті виклику (тобто із ключовим словом this) доступні на рівні об’єкта (ззовні, *public*).

**Закриті поля (ДИВ. ПРИКЛАД У КІНЦІ ДОКУМЕНТУ)**

Якщо описати формальний параметр або локальну змінну (без використання this), то значення такої змінної є недоступним ззовні, але може бути використаним методами, які є описані у конструкторі.

|  |  |
| --- | --- |
| **Загальна форма (закрите поле як локальна змінна)** | **Приклад. Опис закритого поля (доступ лише через методи)** |
| **function <конструктор> (. . .)**  **{**  **//------ Опис закритого поля як локальної змінної**  **var <локальна змінна>=<поч. значення>;**  **//----------- Метод зчитування ------**  **this. get< локальна змінна > =function()**  **{**  **. . . використання локальної змінної . . .**  **}**  **//----------- Метод запису -------------**  **this. set< локальна змінна > =function()**  **{**  **. . . використання локальної змінної . . .**  **}**  **}** | function Secret() {  //---- Закрите поле \_**secretValue** ---  var **\_secretValue** = 10;  //---- Метод зчитування ----  this.getSecretValue = function () {  return **\_secretValue**;  };  //---- Метод запису ------  this.setSecretValue = function (value) {  if (value > 0)  **\_secretValue** = value;  else  throw new Error("value is incorrect");  };  }  ============================================  var s1 = new Secret();  alert(s1.getSecretValue()); // 10  s1.\_secretValue = 90; //створення нового поля(значення закритого не змінилось)  alert(s1.getSecretValue()); //10 (значення не змінилося)  s1.setSecretValue(77); //Встановлення нового значення закритого поля |

|  |  |
| --- | --- |
| **Загальна форма (закрите поле як формальний параметр)** | **Приклад** |
| **function <конструктор> (<фор.параметр>=<поч. значення>)**  **{**  **this. <метод доступу> =function()**  **{**  **. . . використання локальної змінної (форм.парам.) . . .**  **}**  **}** | function Secret(**\_secretValue = 10**) { //**\_secretValue –закрите поле**  //---- Метод зчитування ----  this.getSecretValue = function () {  return **\_secretValue**;  };  //---- Метод запису ------  this.setSecretValue = function (value) {  if (value > 0)  **\_secretValue** = value;  else  throw new Error("value is incorrect");  };  }  var s1 = new Secret();  alert(s1.getSecretValue());  s1.\_secretValue = 90;  alert(s1.getSecretValue());  alert("--- "+s1.\_secretValue );  alert(s1.getSecretValue());  var s1 = new Secret();  alert(s1.getSecretValue()); // 10  s1.\_secretValue = 90; //створення нового поля  alert(s1.getSecretValue()); //10 (значення не змінилося)  alert("--- "+s1.\_secretValue ); |

**Використання єдиного методу як для запису так і для зчитування**

|  |  |
| --- | --- |
| **Загальна форма (закрите поле або лок.змінна, або форм.параметр)** | **Приклад** |
| **function <конструктор> (. . .)**  **{**  **var <локальна змінна>=<поч. значення>;**  **this. <метод доступу> =function(<форм.парам.>)**  **{**  //--- Як метод зчитування  **If( <якщо не передали параметрів> )**  **return <локальна змінна>;**  //--- інаше, використовується як метод доступу  **. . . встановлюємо значення локальної змінної . . .**  **}**  **}** | function Secret() {  //---- Закрите поле \_**secretValue** ---  var \_secretValue = 10;  //----- Метод зчитування і запису (одночасно)  this.SecretValue = function (value) {  //--- Як метод зчитування  if (arguments.length==0) {  return \_secretValue;  }  //--- Як метод доступу  if (value > 0)  \_secretValue = value;  else  throw new Error("value is incorrect");    };  this.setSecretValue = function (value) {  };  }  var s1 = new Secret();  alert(s1.SecretValue());  s1.SecretValue(19);  alert(s1.SecretValue());  s1.SecretValue(-19);  alert(s1.SecretValue()); |

**Методи для управління властивостями об’єкта**

Метод Object.defineProperty дозволяє оголосити властивість об’єкта та налаштувати його особливі аспекти. Це основний метод для управління властивостями у JavaScript.

Синтаксис:

**Object.defineProperty(obj, prop, descriptor)**

Аргументи:

**obj** – об’єкт, для якого оголошується властивість

**prop** – Ім'я властивості, яке потрібно оголосити або модифікувати

**descriptor** – об’єкт, який описує поведінку властивості та який може містити наступні поля:

1. **value** - значення властивості (за замовчуванням undefined). Наприклад:

var user = {};

// 1. простое присваивание

user.name = "Вася";

// 2. указание значения через дескриптор

Object.defineProperty(user, "name", { value: "Вася" });

1. **writable** - значення властивості можна змінювати, якщо true (за замовчуванням false). Наприклад:

var user = {};

Object.defineProperty(user, "name", {

value: "Вася",

writable: false, // запретить присвоение "user.name="

});

1. **configurable** - якщо true, то властивість можна видаляти, а також змінювати його в подальшому за допомогою нових викликів **defineProperty** (за замовчуванням false). Наприклад:

var user = {};

Object.defineProperty(user, "name", {

value: "Вася",

configurable: false // заборонити видалення "delete user.name"

});

1. **enumerable** - якщо true, то властивість братиме участь в переборі for..in. За замовчуванням true. Наприклад:

var user = {

name: "Вася",

toString: function() { return this.name; }

};

// відмічаємо toString як такий, що не підлягає перебору в for..in

Object.defineProperty(user, "toString", {enumerable: false});

// Приклад використання

for(var key in user) alert(key); // name

1. **get** - функція, яка повертає значення властивості. За замовчуванням undefined. Наприклад:

var user = {

firstName: "Вася",

surname: "Петров"

};

Object.defineProperty(user, "fullName", {

get: function() {

return this.firstName + ' ' + this.surname;

}

});

// Приклад використання

alert(user.fullName); // Вася Петров

1. **set** - функція, яка записує значення властивості. За замовчуванням undefined. Наприклад:

var user = {

firstName: "Вася",

surname: "Петров"

};

Object.defineProperty(user, "fullName", {

set: function(value) {

let split = value.split(' ');

this.firstName = split[0];

this.surname = split[1];

}

});

// Приклад використання

user.fullName = "Петя Іванов";

alert( user.firstName ); // Петя

alert( user.surname ); // Іванов

***Зауваження***. JavaScript дозволяє також визначати геттери та сеттери безпосередньо у літералах. Наприклад:

var user = {

firstName: "Вася",

surname: "Петров",

get fullName() {

return this.firstName + ' ' + this.surname;

},

set fullName(value) {

var split = value.split(' ');

this.firstName = split[0];

this.surname = split[1];

}

};

// Приклад використання

alert(user.fullName); // Вася Петров (із геттера)

user.fullName = "Петя Іванов";

alert(user.firstName); // Петя (задав сеттер)

alert(user.surname); // Іванов (задав сеттер)

Створення закритих полів з використанням Object.defineProperty

|  |  |
| --- | --- |
| Використання локальних змінних в якості закритих полів(можна використовувати як локальні змінні, так і формальні параметри) |  |
| function MyClass() {  //Опис закритого поля (локальної змінної)  var \_**властивість**;  //Опис властивості з методами доступу до закритого поля  Object.defineProperty(this, " властивість ", {  get: function () {  return \_**властивість**;  },  set: function (value) {  if (value>=0) {  this.\_**властивість** = value;  }  else {  throw new Error("errrrr");  }  }  })  } | function MyClass() {  // Опис закритого поля (локальної змінної) \_myProp  var \_myProp=25;  // Опис властивості myProp з методами доступу  Object.defineProperty(this, "myProp", {  get: function () {  return \_myProp;  },  set: function (value) {  if (value>=0) {  \_myProp = value;  }  else {  throw new Error("errrrr");  }  }  })  }  var r = new MyClass();  alert(r.myProp);  r.myProp = 23;  alert(r.myProp);  r.myProp = -23; |

|  |  |
| --- | --- |
| Створюється додаткове поле |  |
| function MyClass() {  Object.defineProperty(this, " властивість ", {  get: function () {  return this.\_**властивість**;  },  set: function (value) {  if (value>=0) {  this.\_**властивість** = value;  }  else {  throw new Error("errrrr");  }  }  })  } | function MyClass() {  Object.defineProperty(this, "myProp", {  get: function () {  return this.\_myProp;  },  set: function (value) {  if (value>=0) {  this.\_myProp = value;  }  else {  throw new Error("errrrr");  }  }  })  }  var r = new MyClass();  alert(r.myProp);  r.myProp = 23;  alert(r.myProp);  r.myProp = -23; |

Додавання методів доступу у [**ES-2015**](https://learn.javascript.ru/es-modern)

|  |  |
| --- | --- |
| Загальна форма | Приклад |
| **class** Назва класу {    **constructor**(**форм.параметри**){  **//--- опис полів об’єкта ---**  . . .  }  //--- метод зчитування (геттер)  get ім’я властивості ( )  {  тіло методу  }  //--- метод запису (сеттер)  set ім’я властивості ( форм.параметри )  {  тіло методу  }  . . .  } | 'use strict';  class User {  constructor(firstName, lastName) {  this.firstName = firstName;  this.lastName = lastName;  }  // геттер  get fullName() {  return `${this.firstName} ${this.lastName}`;  }  // сеттер  set fullName(newValue) {  [this.firstName, this.lastName] = newValue.split(' ');  }  // вычисляемое название метода  ["test".toUpperCase()]() {  alert("PASSED!");  }  };  let user = new User("Вася", "Пупков");  alert( user.fullName ); // Вася Пупков **//Виклик get**  user.fullName = "Иван Петров"; **//Виклик set**  alert( user.fullName ); // Иван Петров **//Виклик get**  user.TEST(); // PASSED!  При чтении fullName будет вызван метод get fullName(), при присвоении – метод set fullName с новым значением. |

|  |  |
| --- | --- |
| Загальна форма(приклад з описом додаткового поля) | Приклад |
| **class** Назва класу {    **constructor**(**форм.параметри**){  **//--- опис полів об’єкта ---**  . . .  }  //--- метод зчитування (геттер)  get ім’я властивості ( )  {  тіло методу  }  //--- метод запису (сеттер)  set ім’я властивості ( форм.параметри )  {  тіло методу  }  . . .  } | 'use strict';  class Secret {  constructor() {  }  get SecretValue()  {  return this. \_**secretValue** ;  }  set SecretValue(value)  {  if (value > 0)  this.\_**secretValue** = value;  else  throw new Error("value is incorrect");  }  }  var s1 = new Secret();  alert(s1.SecretValue); //undefined  s1.SecretValue=19; // this.\_secretValue = value;  alert(s1.SecretValue); // 19  s1.SecretValue=-19; // Error |

Геттери і сеттери у літералах

|  |  |
| --- | --- |
|  |  |
| **{**    **властивість1: значення1,**  **властивість2: значення2,**  **...**    **get властивість ()**  **{**  **. . .**  **},**  **set властивість ( нове\_значення\_властивості )**  **{**  **. . .**  **}**  **}** | var user = {  firstName: "Вася",  surname: "Петров",  get fullName() {  return this.firstName + ' ' + this.surname;  },  set fullName(value) {  var split = value.split(' ');  this.firstName = split[0];  this.surname = split[1];  }  };  alert( user.fullName ); // Вася Петров (из геттера)  user.fullName = "Петя Иванов";  alert( user.firstName ); // Петя (поставил сеттер)  alert( user.surname ); // Иванов (поставил сеттер)  ====================================  var ob = {  get prop()  {  return this.\_prop;  },  set prop(value)  {  this.\_prop = value  }  }  alert(ob.prop);  ob.prop = 90;  alert(ob.prop); |

Приклад. Створити клас «Клієнт»

(**ім’я** – довільний доступ (відкрите поле),

**номер рахунку** – тільки для читання,

**кількість грошей** – контрольований доступ (і читання і запис))

<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta charset="utf-8" />

<title></title>

<script>

//name, account, money

function Client(name, account, money) {

//------------------ name (довільний доступ- відкрите поле) --------

this.Name = name;

//------------------ account (тільки для читання) ----

// Варіант 1 (з використанням закритого поля)

//var \_account = account; // опис закритого поля (один раз встановлюємо початкове значення)

//this.getAccount = function () { //опис методу зчитування (метод встановлення (set) не пишемо, тому змінити не зможемо)

// return \_account;

//}

//-------- Варіант 2 (з використанням Object.defineProperty) -----------

Object.defineProperty(this, "Account", {

value: account,

writable: false

});

//---------------- money (контрольований доступ для зчитування і запису) ------

//--- Варіант 1 (з використанням окремих функцій) (геттер і сеттер для закритого поля змінної-формального параметра money)

//this.getMoney = function () { //Метод зчитування закритого поля

// return money;

//}

//this.setMoney = function (value) { //Метод запису закритого поля

// if (value<0) {

// throw new Error("err");

// }

// money = value;

//}

//this.setMoney(money); //Контроль значення, яке передано у конструктор

//------ Варіант 2 (з використанням Object.defineProperty)------------

Object.defineProperty(this, "Money", {

get: function () {

return money; //Закрите поле money описано як формальний параметр

},

set: function (value) {

if (value >= 0) {

money = value;

}

else {

throw new Error("errrrr");

}

}

})

this.Money = money; // Контроль значення, яке передано у конструктор

this.toString = function () {

//return `${this.name} - ${this.getMoney()}`;

return `${this.Name} - ${this.Money}`;

}

}

//============================ Використання класу (приклад з аналізом на виключні ситуації) =========================

try {

var r = new Client("Ivan", "12434assd", 89);

//r.setMoney(100);

r.Money=-100;

document.write(r);

}

catch (e)

{

alert("Wrong value :"+e)

}

document.write("The end");

</script>

</head>

<body>

</body>

</html>