**МЕТОДИ СТВОРЕННЯ ОБ’ЄКТІВ**

1. З використанням літералів
2. З використанням класів-конструкторів
3. З використанням класів тільки в ES-6

Створення об’єктів-літералів

|  |  |
| --- | --- |
|  |  |
| **{**    **властивість1: значення1,**  **властивість2: значення2,**  **...**    **функція-метод 1: function (форм.парам.)**  **{**  **. . .**  **}**  **функція-метод 2: function (форм.парам.)**  **{**  **. . .**  **}**  **}** | var user = {  //------ Властивості ------  name: "Іван",  surname: "Петров",  age: 25,  //------- Методи ------  sayHi: function () {  alert('Привіт!');  }  sayBye: function () {  alert('До побачення!');  }  }; |

**Опис функцій конструкторів**

|  |  |
| --- | --- |
| Варіант 1. (поля у функції-конструкторі, методи у прототипі) |  |
| **function ім’я функції-конструктора {**  **//-------- Опис полів -----**  **this.властивість1 = значення1;**  **this.властивість2 = значення2;**    **. . .**  **//--------- Опис методів -----**  **this.функція-метод 1 = function (форм.парам.)**  **{**  **. . .**  **}**  **this.функція-метод 2 = function (форм.парам.)**  **{**  **. . .**  **}**  **. . .**  **}** | //ВАРІАНТ №1 (І ПОЛЯ І МЕТОДИ У ФУНКЦІЇ ОПИСУЄМО)  function Range(defaultMin, defaultMax) {  //--------- Опис полів  this.min = defaultMin;  this.max = defaultMax;  //--------- Методів  this.isInRange = function (value) {  //if (value >= this.min && value <= this.max) {  // return true;  //}  //else {  // return false;  //}  return (value >= this.min && value <= this.max);  }  this.getRandomValueFromRange = function () {  return this.min + Math.floor(Math.random() \* (this.max –  this.min + 1));  }  this.toString = function () {  return "[" + this.min + " ; " + this.max + "]";  }  } |

|  |  |
| --- | --- |
| Варіант 2. (поля у функції-конструкторі, методи у прототипі) |  |
| **function ім’я функції-конструктора {**  **//-------- Опис полів -----**  **this.властивість1 = значення1;**  **this.властивість2 = значення2;**  **. . .**  **}**  **//--------- Опис методів -----**  **ім’я функції.prototype. метод 1 = function (форм.парам.)**  **{**  **. . .**  **}**  **ім’я функції.prototype. метод 2 = function (форм.парам.)**  **{**  **. . .**  **}**  **. . .** | ВАРІАНТ №2 (ПОЛЯ У ФУНКЦІЇ, МЕТОДИ У ПРООТИПІ)  function Range(defaultMin, defaultMax) {  //--------- Опис полів  this.min = defaultMin;  this.max = defaultMax;  }  //--------- Методів  Range.prototype.isInRange = function (value) {  //if (value >= this.min && value <= this.max) {  // return true;  //}  //else {  // return false;  //}  return (value >= this.min && value <= this.max);  }  Range.prototype.getRandomValueFromRange = function () {  return this.min + Math.floor(Math.random() \* (this.max - this.min  + 1));  }  Range.prototype.toString = function () {  return "[" + this.min + " ; " + this.max + "]";  } |

**Опис класів у** [**ES-2015**](https://learn.javascript.ru/es-modern)

|  |  |
| --- | --- |
|  |  |
| **class** Назва класу {    **constructor**(**форм.параметри**){  **//--- опис полів об’єкта ---**  this.ім’я поля1=поч.значення;  this.ім’я поля2=поч.значення ;  . . .  }  ім’я методу1 ( форм.параметри )  {  тіло методу  }  ім’я методу2 ( форм.параметри )  {  тіло методу  }  . . .  }  Опис класу.  1) Складовими опису є тільки функції  2) властивості описуємо у конструкторі | 'use strict';  class User {  constructor(name, age) {  this.name = name;  this.age=age;  }  sayHi() {  alert(this.name);  }  toString() {  return “User :”+this.name;  }  } |

Статичні поля і методи

## [Статические свойства](https://learn.javascript.ru/es-class" \l "%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B5-%D1%81%D0%B2%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0)

Класс, как и функция, является объектом. Статические свойства класса User – это свойства непосредственно User, то есть доступные из него «через точку».

Для их объявления используется ключевое слово static.

Например:

'use strict';

class User {

constructor(firstName, lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

static createGuest() {

return new User("Гость", "Сайта");

}

};

let user = User.createGuest();

alert( user.firstName ); // Гость

alert( User.createGuest ); // createGuest ... (функция)

Как правило, они используются для операций, не требующих наличия объекта, например – для фабричных, как в примере выше, то есть как альтернативные варианты конструктора. Или же, можно добавить метод User.compare, который будет сравнивать двух пользователей для целей сортировки.

Также статическими удобно делать константы:

'use strict';

class Menu {

static get elemClass() {

return "menu"

}

}

alert( Menu.elemClass ); // menu