This дает доступ к объекту из метода. Значением this является объект перед точкой, в контексте которого вызван метод.

Если this внутри нового объекта, но не внутри функции, контекст не меняется!!!

Что выведет alert в этом коде? Почему?

var name = "";

var user = {

name: "Василий",

export: function() {

return {

value: this

};

}

};

alert( user.export().value.name );

решение

**Ответ: Василий.**

Во время выполнения user.export() значение this = user.

При создании объекта { value: this }, в свойство value копируется ссылка на текущий контекст, то есть на user.

Получается что user.export().value == user.

Так как внутри объекта нет функции, то контекст не меняется !!!!!!

var user = {

name: "Василий",

export: function() {

return {

value: function() {

return this;}

};

}

};

alert( user.export().value().name ); // вернет unidentified

var name = "";

var user = {

name: "Василий",

export: function() {

return {

value: function() {

return this;}

};

}

};

alert( user.export().value() ); //вернет сам объект value

Есть объект «лестница» ladder:

var ladder = {

step: 0,

up: function() { // вверх по лестнице

this.step++;

},

down: function() { // вниз по лестнице

this.step--;

},

showStep: function() { // вывести текущую ступеньку

alert( this.step );

}

};

Сейчас, если нужно последовательно вызвать несколько методов объекта, это можно сделать так:

ladder.up();

ladder.up();

ladder.down();

ladder.showStep(); // 1

Модифицируйте код методов объекта, чтобы вызовы можно было делать цепочкой, вот так:

ladder.up().up().down().up().down().showStep(); // 1

Как видно, такая запись содержит «меньше букв» и может быть более наглядной.

Такой подход называется «чейнинг» (chaining) и используется, например, во фреймворке jQuery.

решение

Решение состоит в том, чтобы каждый раз возвращать текущий объект. Это делается добавлением return this в конце каждого метода:

var ladder = {

step: 0,

up: function() {

this.step++;

return this;

},

down: function() {

this.step--;

return this;

},

showStep: function() {

alert( this.step );

return this;

}

}

ladder.up().up().down().up().down().showStep(); // 1