Getters and Setters in Object Literals

const person =

{

    \_firstName: 'John',

    \_lastName: 'Doe',

}

Object.defineProperties(person,

{

    firstName:

    {

        get()

        {

            return this.\_firstName;

        },

        set(value)

        {

            if(typeof value === 'string')

            {

                this.\_firstName = value;

                return;

            }

            console.log(`Invalid first name : ${value}`);

        },

        configurable: true,

        enumerable: false

    },

    lastName:

    {

        get()

        {

            return this.\_lastName;

        },

        set(value)

        {

            if(typeof value === 'string')

            {

                this.\_lastName = value;

                return;

            }

            console.log(`Invalid last name : ${value}`);

        },

        configurable: true,

        enumerable: false

    },

    fullName:

    {

        get()

        {

            return `${this.\_firstName} ${this.\_lastName}`;

        },

        configurable: true,

        enumerable: false

    }

});

console.log(person.fullName); // John Doe

person.firstName = 'Jane';

person.lastName = 'Smith';

console.log(person.fullName); // Jane Smith

person.firstName = 123; // Invalid first name : 123

person.lastName = 456; // Invalid last name : 456

console.log(person.fullName); // Jane Smith

for(let key in person)

{

    console.log(key);

}

//\_firstName

//\_lastName

Property descriptors cannot have both getters/setters and the writable attribute.

const person =

{

    \_firstName: 'John',

    \_lastName: 'Doe',

    get firstName()

    {

        return this.\_firstName;

    },

    get lastName()

    {

        return this.\_lastName;

    },

    get fullName()

    {

        return `${this.\_firstName} ${this.\_lastName}`;

    },

    set firstName(value)

    {

        if(typeof value === 'string')

        {

            this.\_firstName = value;

            return;

        }

        console.log(`Invalid first name : ${value}`);

    },

    set lastName(value)

    {

        if(typeof value === 'string')

        {

            this.\_lastName = value;

            return;

        }

        console.log(`Invalid first name : ${value}`);

    }

}

Object.defineProperties(person,

{

    firstName:

    {

        configurable: true,

        enumerable: false

    },

    lastName:

    {

        configurable: true,

        enumerable: false,

    },

    fullName:

    {

        get()

        {

            return `${this.firstName} ${this.lastName}`

        },

        configurable: true,

        enumerable: false

    }

});

console.log(person.fullName); // John Doe

person.firstName = 'Jane';

person.lastName = 'Smith';

console.log(person.fullName); // Jane Smith

person.firstName = 123; // Invalid first name : 123

person.lastName = 456; // Invalid last name : 456

console.log(person.fullName); // Jane Smith

for(let key in person)

{

    console.log(key);

}

//\_firstName

//\_lastName

Getters and Setters in Constructor Functions

function Person(\_firstName, \_lastName)

{

    let firstName = \_firstName;

    let lastName = \_lastName;

    Object.defineProperties(this,

    {

        firstName:

        {

            get()

            {

                return firstName;

            },

            set(value)

            {

                if(typeof value === 'string')

                {

                    firstName = value;

                    return;

                }

                console.log(`Invalid first name : ${value}`);

            },

            configurable: true,

            enumerable: false

        },

        lastName:

        {

            get()

            {

                return lastName;

            },

            set(value)

            {

                if(typeof value === 'string')

                {

                    lastName = value;

                    return;

                }

                console.log(`Invalid last name : ${value}`);

            },

            configurable: true,

            enumerable: false

        },

        fullName:

        {

            get()

            {

                return `${firstName} ${lastName}`;

            },

            configurable: true,

            enumerable: false

        }

    });

}

const person = new Person("John", "Doe");

console.log(person.fullName); // John Doe

person.firstName = 'Jane';

person.lastName = 'Smith';

console.log(person.fullName); // Jane Smith

person.firstName = 123; // Invalid first name : 123

person.lastName = 456; // Invalid last name : 456

console.log(person.fullName); // Jane Smith

for(let key in person)

{

    console.log(key);

}

//\_firstName

//\_lastName

Getters and Setters in Classes

const \_firstName = new WeakMap();

const \_lastName = new WeakMap();

class Person

{

    constructor(firstName, lastName )

    {

        \_firstName.set(this, firstName);

        \_lastName.set(this, lastName);

    }

    get firstName()

    {

        return \_firstName.get(this);

    }

    get lastName()

    {

        return \_lastName.get(this);

    }

    get fullName()

    {

        return `${\_firstName.get(this)} ${\_lastName.get(this)}`;

    }

    set firstName(value)

    {

        if(typeof value === 'string')

        {

            \_firstName.set(this, value);

            return;

        }

        console.log(`Invalid first name : ${value}`);

    }

    set lastName(value)

    {

        if(typeof value === 'string')

        {

            \_lastName.set(this, value);

            return;

        }

        console.log(`Invalid last name : ${value}`);

    }

}

Object.defineProperties(Person.prototype,

{

    firstName:

    {

        configurable: true,

        enumerable: false

    },

    lastName:

    {

        configurable: true,

        enumerable: false,

    },

    fullName:

    {

        configurable: true,

        enumerable: false

    }

});

const person = new Person("John", "Doe");

console.log(person.fullName); // John Doe

person.firstName = 'Jane';

person.lastName = 'Smith';

console.log(person.fullName); // Jane Smith

person.firstName = 123; // Invalid first name : 123

person.lastName = 456; // Invalid last name : 456

console.log(person.fullName); // Jane Smith

for(let key in person)

{

    console.log(key);

}

//\_firstName

//\_lastName