Access modifier static, getter/setter

Module: Advanced Programming with JavaScript

Mục tiêu

- Phân biệt được biến kiểu dữ liệu nguyên thủy và biến tham chiếu
- Phân biệt được biến của lớp và biến của đối tượng
- Phân biệt được phương thức của lớp và phương thức của đối tượng
- Khai báo và sử dụng được các biến static
- Khai báo và sử dụng được các phương thức static
- Trình bày được các access modifier
- Triển khai được getter/setter

Biến tham chiếu và biến tham trị

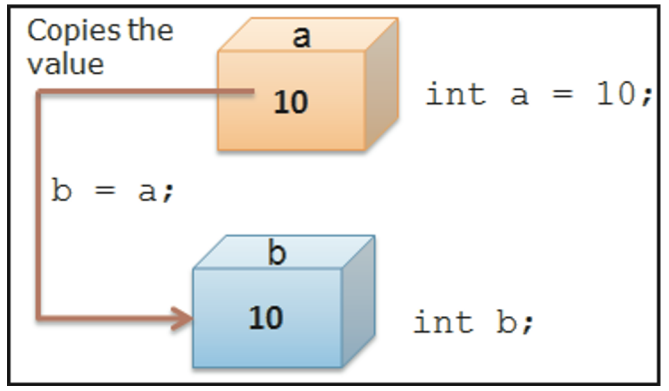
Biến kiểu dữ liệu nguyên thuỷ

Biến tham chiếu

Primitive data types

- Các biến thuộc kiểu dữ liệu nguyên thuỷ (như number, boolean) lưu trữ **giá trị** của chúng trong vùng nhớ được cấp
- Giá trị của một biến có thể được gán cho một biến khác
- Ví dụ:

```
let a = 10;
let b = a;
```



  - Thao tác này sao chép giá trị của biến a (được lưu trong vùng nhớ được cấp cho a) cho biến b (lưu vào vùng nhớ được cấp cho b)

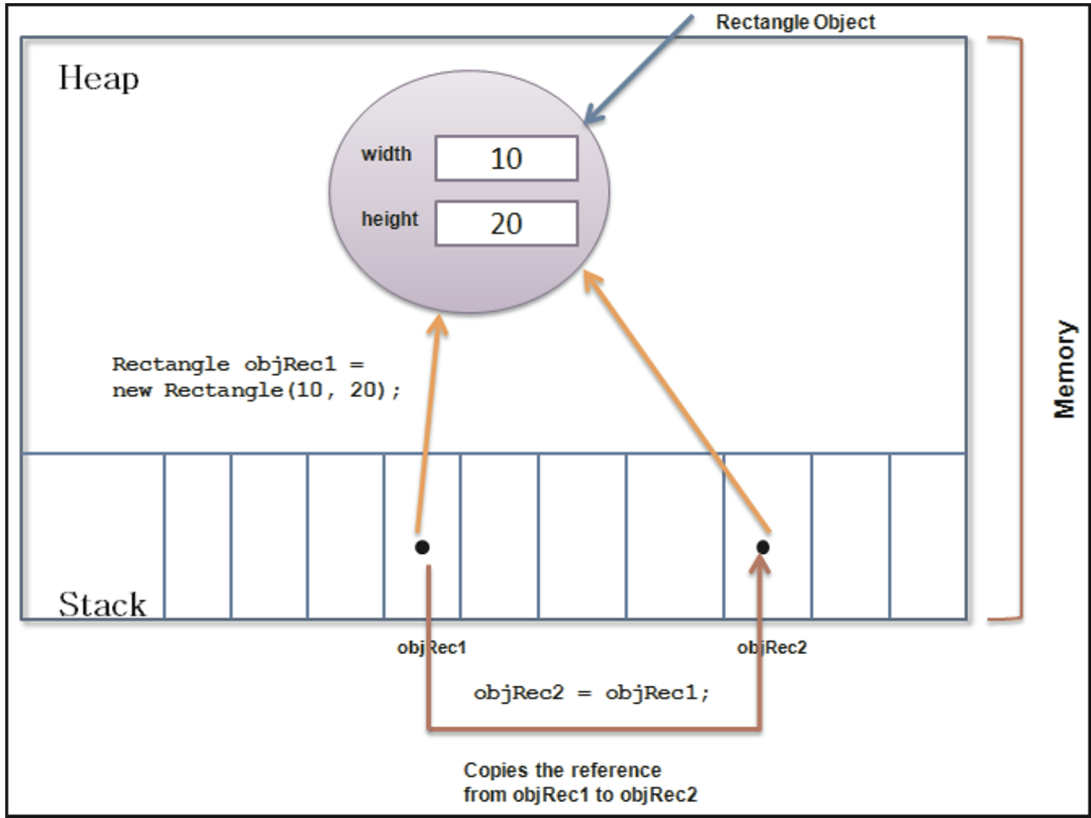
Reference data types

- Các biến thuộc kiểu dữ liệu tham chiếu (chẳng hạn như Scanner, Person, Customer…) lưu trữ **tham chiếu** của đối tượng ở trong vùng nhớ được cấp
- Có thể gán giá trị tham chiếu của một biến cho một biến khác
- Ví dụ:

```
let rectangleObj1: Rectangle = new Rectangle(10, 20);

let rectangleObj2: Rectangle = rectangleObj1;
```

  - Thao tác này sao chép **địa chỉ** được lưu trong biến rectangleObj1 sang biến rectangleObj2
  - Không có ảnh hưởng nào xảy ra đối với đối tượng thực tế trong bộ nhớ



Primitive data type: Ví dụ

```
function swap(int first, int second) {
    let temp: number = first;
    let first: number  = second;
    let second: number  = temp;
}

function main() {
    let a: number = 5;
    let b: number = 10;

    swap(a, b);

    console.log("a = " + a);
    console.log("b = " + b);
}
```

Kết quả:

```
a = 5
b = 10
```

Reference data type: Ví dụ

```
class Person{
    public name: string;

    constructor(name: string){
        this.name = name;
    }
}
```

```
function swap(first: Person, second: Person){
    let temp: string = first.name;
    first.name = second.name;
    second.name = temp;
}

function main() {
    let a: Person = new Person("John");
    let b: Person = new Person("Bill");

    swap(a, b);

    console.log("a.name = " + a.name);
    console.log("b.name = " + b.name);
}
```

Kết quả:

```
a.name = Bill
b.name = John
```

Từ khoá static

Static property Static method

Từ khoá static

- Từ khoá static được sử dụng để khai báo các thuộc tính và phương thức của lớp (khác với thuộc tính và phương thức của đối tượng)
- Các thành phần static trực thuộc lớp, thay vì trực thuộc đối tượng
- Biến static còn được gọi là biến của lớp (class variable)
- Phương thức static còn được gọi là phương thức của lớp (class method)
- Có thể truy xuất các thành phần static bằng cách sử dụng lớp hoặc đối tượng
- Không cần khởi tạo đối tượng vẫn có thể sử dụng các thành phần static

Static property

- Cú pháp khai báo *static property*:

```
modifier static variable_name: data_type;
```

- Ví dụ:

Khai báo biến static:

```
class Application{
   public static language: string = "english";
}
```

Truy xuất biến static:

```
console.log("Current language: " + Application.language);
```

Static method

- Cú pháp khai báo static method:

```
modifier static method_name(): data_type {
   //body
}
```

- Ví dụ:
  - Khai báo phương thức static

```
class Application{
   public static getVersion(): string{
      return "1.0";
   }
}
```

  - Gọi phương thức static

```
console.log("Current version: " + Application.getVersion());
```

Access modifier

Access modifier

- Access modifier là các từ khoá được sử dụng để quy định mức độ truy cập đến lớp và các thành phần của lớp
- Các mức truy cập:
  - *public*: có thể truy cập từ bất cứ đâu
  - *private*: các phương thức và thuộc tính chỉ được phép truy xuất trong cùng một lớp
  - *protected*: các phương thức và thuộc tính được phép truy xuất trong cùng một lớp và ở các lớp con (kế thừa)

Getter và Setter

Truy cập trực tiếp vào các trường

- Sử dụng từ khoá public khi khai báo thuộc tính sẽ cho phép truy cập trực tiếp vào các thuộc tính đó
- Ví dụ:

  Khai báo lớp Person sau cho phép truy cập trực tiếp vào trường name

```
class Person{
   public name: string;
}
```

```
let person: Person = new Person();
person.name = "John";
```

- Nhược điểm:
  - Không kiểm soát được truy cập vào thuộc tính
  - Gây khó khăn cho việc duy trì, dễ phát sinh bug

Khai báo getter/setter

- Cú pháp khai báo getter:

  get propertyName(): returnType

  - *Đối với các thuộc tính kiểu boolean* thì tên getter bắt *đầu bằng chữ is*:

    get isPropertyName(): **boolean**

- Cú pháp khai báo setter:

  set propertyName(propertyValue: dataType): **void**

Nested class và anonymous class

Nested class

Anonymous class

Nested class

- Nested class (lớp lồng nhau) là một lớp được khai báo bên trong lớp khác
- Cú pháp:

```
class OuterClass {
    ...
    static class StaticNestedClass {
        ...
    }
    class InnerClass {
        ...
    }
}
```

Khi nào sử dụng nested class?

- Để khai báo các lớp mà chỉ được sử dụng ở một nơi duy nhất
  - Ẩn một lớp ở bên trong lớp khác sẽ giúp cho các package gọn gàng hơn
- Để tăng tính bao gói (encapsulation)
- Giúp mã nguồn dễ đọc hơn
  - Các nested class nhỏ được đặt cạnh nơi sử dụng chúng giúp cho việc quản lý dễ dàng hơn

Static nested class

- Static nested class trực thuộc lớp ở bên ngoài (thay vì trực thuộc đối tượng của lớp bên ngoài)
- Static nested class không thể truy xuất đến các thành phần của lớp bên ngoài
- Sử dụng tên của lớp bên ngoài để truy cập đến lớp bên trong
- Ví dụ:

```
let nestedObject: OuterClass.StaticNestedClass
                                = new OuterClass.StaticNestedClass();
```

Inner class (non-static class)

- Inner class trực thuộc một đối tượng của lớp bên ngoài
- Inner class có thể truy xuất đến các thành phần của lớp bên ngoài
- Sử dụng tham chiếu của một đối tượng của lớp bên ngoài để truy xuất đến Inner class
- Ví dụ:

```
let outerObject: OuterClass = new OuterClass();
let innerObject: OuterClass.InnerClass = outerObject.new InnerClass();
```

Local class

- Local class (lớp địa phương) là lớp được khai báo và sử dụng bên trong một khối lệnh
- Không thể sử dụng local class bên ngoài khối lệnh được khai báo
- Ví dụ:

```
class Customer{
  public boolean validateAddress(customerAddress: string){
    class Address{
      address: string;
      constructor(address: string){
        this.address = address;
      }

      public validate(): boolean {
        //body
      }
    }

    let address: Address = new Address(customerAddress);
    return address.validate();
  }
}
```

Demo

Tổng kết

- Biến tham trị chứa giá trị của nó trong vùng nhớ được cấp
- Biến tham chiếu chứa tham chiếu đến đối tượng trong vùng nhớ được cấp
- Từ khoá static được sử dụng để khai báo các thành phần thuộc lớp
- Getter/setter là cơ chế để kiểm soát truy cập đến các trường dữ liệu của đối tượng
- Nested class là lớp được khai báo bên trong lớp khác
- Local class là lớp được khai báo bên trong một khối lệnh