EXAMPLE 46: CREATE A SHAPE OBJECT AS `prototype` FOR CIRCLE AND RECTANGLE OBJECTS

EXAMPLE 46: CREATE A SHAPE OBJECT AS `prototype` FOR CIRCLE AND RECTANGLE OBJECTS

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS WITH ALL PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE `prototype` OF THE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

EXAMPLE 46:

prototype

1. CREATE A SHAPE OBJECT CONSTRUCTOR
THAT WILL BE AN ABSTRACTION OF THE ARTIFACTS

LET'S TAKE THIS STEP BY STEP :-)

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

EXAMPLE 46:

prototype

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

```
function Shape(shapeName) {
  console.log("Inside the shape object constructor");
  this.shapeName = shapeName;
  this.draw = function() {
    console.log("I am a " + this.shapeName + " and I am
drawing myself");
  }
}
```

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

```
function Shape(shapeName) {
  console.log("Inside the Shape object constructor");
  this.shapeName = shapeName;
  this.draw = function() {
    console.log("I am a " + this.shapeName + " and I am
drawing myself");
  }
}
```

SHAPE OBJECT CONSTRUCTOR

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

```
function Shape(shapeName) {
  console.log("Inside the shape object constructor");
    this.shapeName = shapeName;
    this.draw = function() {
      console.log("I am a " + this.shapeName + " and I am
drawing myself");
    }
}
```

PROPERTIES SHARED BY BOTH CIRCLE AND RECTANGLE

EXAMPLE 46:

prototype

1. CREATE A SHAPE OBJECT CONSTRUCTOR WITH ALL PROPERTIES THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

EXAMPLE 46:

`prototype`

1. CREATE A SHAPE OBJECT CONSTRUCTOR
   THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
   PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
   RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

```
function Circle(r) {
  console.log("Inside the circle object constructor");
  this.radius = r;
}
```

```
function Rectangle(l,b){
  console.log("Inside the rectangle object constructor"
  this.length = l;
  this.breadth = b;
}
```

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

```
function Circle(r) {
    console.log("Inside the circle object constructor");
    this.radius = r;
}
```

```
function Rectangle(l,b){
    console.log("Inside the rectangle object constructor"
    this.length = l;
    this.breadth = b;
}
```

OBJECT CONSTRUCTORS FOR BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

```
function Circle(r) {
  console.log("Inside the circle object constructor");
  this.radius = r;
}
```

```
function Rectangle(l,b){
  console.log("Inside the rectangle object constructor");
  this.length = l;
  this.breadth = b;
}
```

PROPERTIES SPECIFIC TO EITHER CIRCLE OR RECTANGLE

EXAMPLE 46:

`prototype`

1. CREATE A SHAPE OBJECT CONSTRUCTOR
   THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
   PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
   RECTANGLE OBJECT CONSTRUCTORS IS A SHAPE OBJECT

EXAMPLE 46:

`prototype`

1. CREATE A SHAPE OBJECT CONSTRUCTOR
THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
RECTANGLE OBJECT CONSTRUCTORS

3. SPECIFY THAT THE RECTANGLE OBJECT CONSTRUCTORS

`prototype` IS A JAVASCRIPT KEYWORD, AND IS A PROPERTY OF EVERY OBJECT CONSTRUCTOR

```
Circle.prototype = new Shape("Circle");
Rectangle.prototype = new Shape("Rectangle");
```

3. SPECIFY THAT THE RECTANGLE OBJECT CONSTRUCTORS

`prototype` IS A JAVASCRIPT KEYWORD, AND IS A PROPERTY OF EVERY OBJECT CONSTRUCTOR

NOW - EVERY OBJECT OF THE CIRCLE CLASS WILL FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT A

```
Circle.prototype = new Shape("Circle");
Rectangle.prototype = new Shape("Rectangle");
```

3. SPECIFY THAT THE RECTANGLE OBJECT CONSTRUCTORS

`prototype` IS A JAVASCRIPT KEYWORD, AND IS A PROPERTY OF EVERY OBJECT CONSTRUCTOR

NOW - EVERY OBJECT OF THE CIRCLE CLASS WILL INHERIT FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT A

```
Circle.prototype = new Shape("Circle");
Rectangle.prototype = new Shape("Rectangle");
```

NOW - EVERY OBJECT OF THE RECTANGLE CLASS WILL INHERIT FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT B

NOW - EVERY OBJECT OF THE CIRCLE CLASS WILL
FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT A

NOW - EVERY OBJECT OF THE RECTANGLE CLASS WILL
FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT B

THIS IS VERY VERY DIFFERENT FROM THE WAY JAVA OR C++ DO

THIS IS VERY VERY DIFFERENT FROM THE WAY JAVA OR C++ DO INHERITANCE - THERE EACH OBJECT OF CIRCLE OR RECTANGLE HAS ITS OWN COPY OF THE SHAPE OBJECT!

THIS IS VERY VERY DIFFERENT FROM THE WAY JAVA OR C++ DO INHERITANCE

EACH OBJECT OF CIRCLE OR RECTANGLE HAS ITS OWN COPY OF THE SHAPE OBJECT!

THIS IS VERY VERY DIFFERENT FROM THE WAY JAVA OR C++ DO INHERITANCE - EACH OBJECT OF CIRCLE OR RECTANGLE HAS ITS OWN COPY OF THE SHAPE OBJECT!

THIS IS CALLED
PROTOTYPICAL INHERITANCE

NOW - EVERY OBJECT OF THE CIRCLE CLASS WILL
FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT A

NOW - EVERY OBJECT OF THE RECTANGLE CLASS WILL
FROM 1 SPECIFIC OBJECT OF THE SHAPE CLASS - CALL IT B

THIS IS VERY VERY DIFFERENT FROM THE WAY JAVA OR C++ DO

EXAMPLE 46:

`prototype`

1. CREATE A SHAPE OBJECT CONSTRUCTOR
THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
RECTANGLE OBJECT CONSTRUCTORS

EXAMPLE 46:

prototype

1. CREATE A SHAPE OBJECT CONSTRUCTOR
THAT ARE SHARED ACROSS BOTH CIRCLE AND RECTANGLE

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
RECTANGLE OBJECT CONSTRUCTORS

EXAMPLE 46:

prototype

1. CREATE A SHAPE OBJECT CONSTRUCTOR

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

2. CREATE CIRCLE AND RECTANGLE OBJECT CONSTRUCTORS
PROPERTIES THAT ARE SPECIFIC TO CIRCLE AND RECTANGLE OBJECTS

3. SPECIFY THAT THE
RECTANGLE OBJECT CONSTRUCTORS

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

```
var circle1 = new Circle(3);
var rectangle2 = new Rectangle(4);
console.log(circle1.radius);
console.log(rectangle2.length);
circle1.draw();
rectangle2.draw();
```

```
Inside the circle object constructor
Inside the rectangle object constructor
3
4
I am a Circle and I am drawing myself
I am a Rectangle and I am drawing myself
```

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

```
var circle1 = new Circle(3);
var rectangle2 = new Rectangle(4);
console.log(circle1.radius);
console.log(rectangle2.length);
circle1.draw();
rectangle2.draw();
```

```
Inside the circle object constructor
Inside the rectangle object constructor
3
4
I am a Circle and I am drawing myself
I am a Rectangle and I am drawing myself
```

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

```
var circle1 = new Circle(3);
var rectangle2 = new Rectangle(4);
console.log(circle1.radius);
console.log(rectangle2.length);
circle1.draw();
rectangle2.draw();
```

```
Inside the circle object constructor
Inside the rectangle object constructor
3
4
I am a Circle and I am drawing myself
I am a Rectangle and I am drawing myself
```

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

```
var circle1 = new Circle(3);
var rectangle2 = new Rectangle(4);

console.log(circle1.radius);
console.log(rectangle2.length);

circle1.draw();
rectangle2.draw();
```

```
Inside the circle object constructor
Inside the rectangle object constructor
3
4
I am a Circle and I am drawing myself
I am a Rectangle and I am drawing myself
```

NOW WE CAN GO AHEAD AND CREATE OBJECTS!

```
var circle1 = new Circle(3);
var rectangle2 = new Rectangle(4);
console.log(circle1.radius);
console.log(rectangle2.length);
circle1.draw();
rectangle2.draw();
```

```
Inside the circle object constructor
Inside the rectangle object constructor
3
4
I am a Circle and I am drawing myself
I am a Rectangle and I am drawing myself
```