
fact(fact)(n)

# 利用函数替换

# 等价替换

# 外提

# 规约参数

# 这就是Y-Combinator

```
const Y = le => (f => f(f))(g => le(x => g(g)(x)));
```

```
const Y = le => (f => f(f))(g => le(x => g(g)(x)));
const fact = Y(fact => n => n < 2 ? 1 : n * fact(n - 1));
fact(5);
```

```
120
```

```
=> (f => f(f))(fact)(n)
```

```
=> (f => f(f))((l => f => l(x => f(f)(x)))(P))(n);
```

```
=> (p => (f => f(f))((l => f => l(x => f(f)(x)))(p)))(P)(n);
```

```
=> (p => (f => f(f))(f => p(x => f(f)(x))))(P)(n);
```

```
fact(fact)(n)
```

**利用函数替换**

```
=> (f => f(f))(fact)(n)
```

**等价替换**

```
=> (f => f(f))((l => f => l(x => f(f)(x)))(P))(n);
```

**外提**

```
=> (p => (f => f(f))((l => f => l(x => f(f)(x)))(p)))(P)(n);
```

**规约参数**

```
=> (p => (f => f(f))(f => p(x => f(f)(x))))(P)(n);
```

---

**这就是Y-Combinator**

```
const Y = le => (f => f(f))(g => le(x => g(g)(x)));
```

```
const Y = le => (f => f(f))(g => le(x => g(g)(x)));
const fact = Y(fact => n => n < 2 ? 1 : n * fact(n - 1));
fact(5);

120
```

# 函数式编程的数学理论——λ演算