

《Web前端开发》理论考试

中级试卷分析

马庆槐 2021年3月



www.miiteec.org.cn

目录

1. 总体介绍

2. 典型试题分析



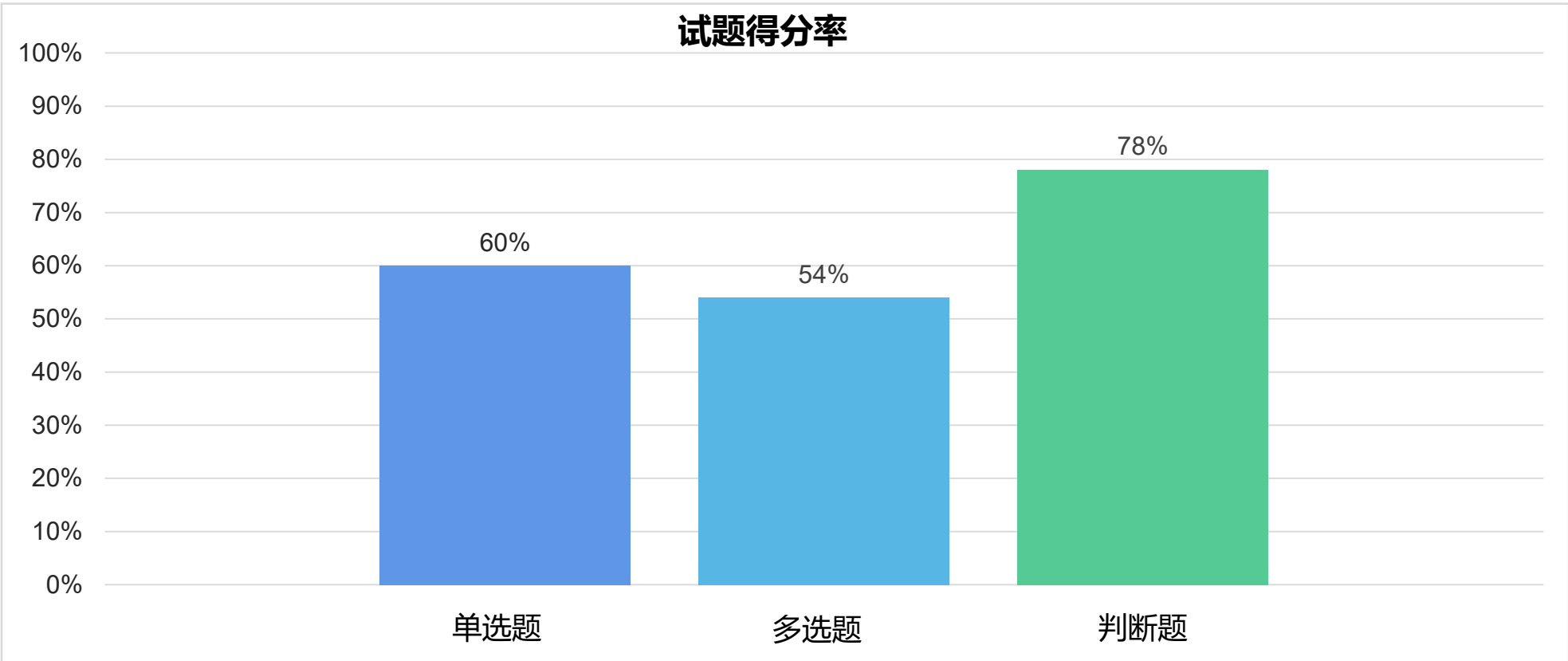
01

总体介绍

>>> 试卷结构

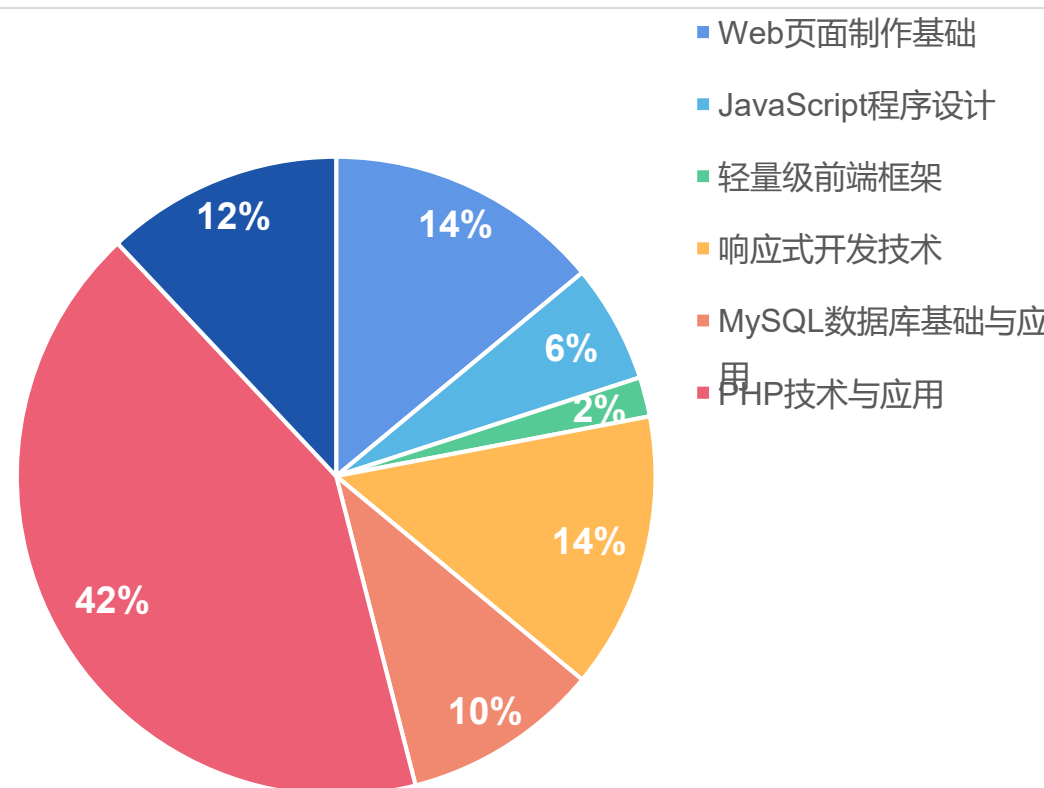


| 题型 | 试题数 | 得分率 |
|-----|-----|-----|
| 单选题 | 30 | 60% |
| 多选题 | 15 | 54% |
| 判断题 | 5 | 78% |



>>> 考试范围：分值分布

| 知识模块 | 试题数 | 占比 |
|------------------------------------|-----|-----|
| Web页面制作基础 (HTML+HTML5+CSS+CSS3) | 7 | 14% |
| JavaScript程序设计 | 3 | 6% |
| 轻量级前端框架 (jQuery) | 1 | 2% |
| 响应式开发技术 | 7 | 14% |
| MySQL数据库基础与应用 | 5 | 10% |
| PHP技术与应用 | 21 | 42% |
| Web前后端数据交互技术 | 6 | 12% |



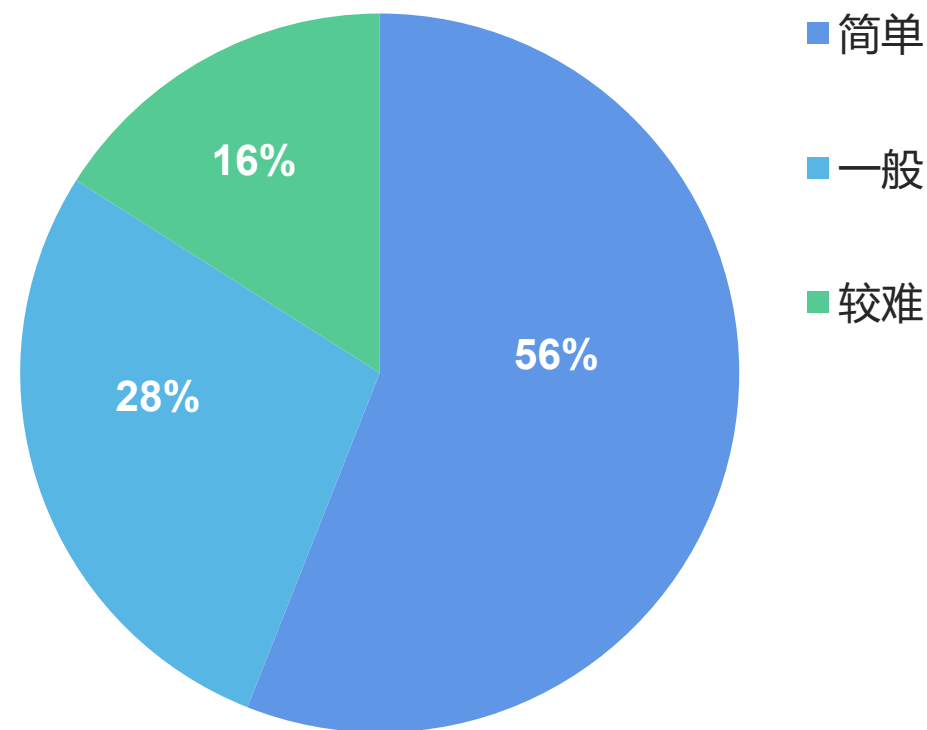
>>> 考试范围：考核点分布

| 知识模块 | 考核点(职业技能) |
|------------------------------------|---|
| Web页面制作基础 (HTML、HTML5、CSS、CSS3) | HTML文档声明、HTML元素、HTML5语义化元素、新增表单控件、HTML5废弃元素、浏览器兼容性； CSS引入、基本语法、选择器、背景 |
| JavaScript程序设计 | JavaScript数组、DOM操作、面向对象、继承 |
| 轻量级前端框架(jQuery) | jQuery选择器 |
| 响应式开发技术 | Bootstrap栅格系统、图片样式、表格样式、组件、弹性布局、媒体查询 |
| MySQL数据库基础与应用 | MySQL数据类型、SQL查询、limit、视图、存储过程、触发器、数据库备份与恢复 |
| PHP技术与应用 | 开发环境、PHP基本语法、注释、输出、运算符、函数、数组、面向对象、图像GD库、文件\$_FILES、超全局变量、Session、Cookie、mysqli操作数据库 |
| | Laravel框架：路由、模板、CSRF |
| Web前后端数据交互技术 | Ajax、XMLHttpRequest 对象、JSON数据格式 |

>>> 难易度分析

| 难易度 | 类型 | 题目数 | 占比 | 合计 |
|-----|-----|-----|-----|-----|
| 简单 | 单选题 | 17 | 34% | 56% |
| | 多选题 | 6 | 12% | |
| | 判断题 | 5 | 10% | |
| 一般 | 单选题 | 9 | 18% | 28% |
| | 多选题 | 5 | 10% | |
| 较难 | 单选题 | 4 | 8% | 16% |
| | 多选题 | 4 | 8% | |

难易度分析





02

典型试题分析



多选题

01. JavaScript继承



单选题

02. Bootstrap栅格系统



单选题

03. MySQL存储过程



单选题

04. PHP运算符



多选题

05. Laravel框架



多选题

以下JavaScript实现继承的方式，正确的是（ ）。

- A、原型链继承
- B、构造函数继承
- C、组合继承
- D、关联继承

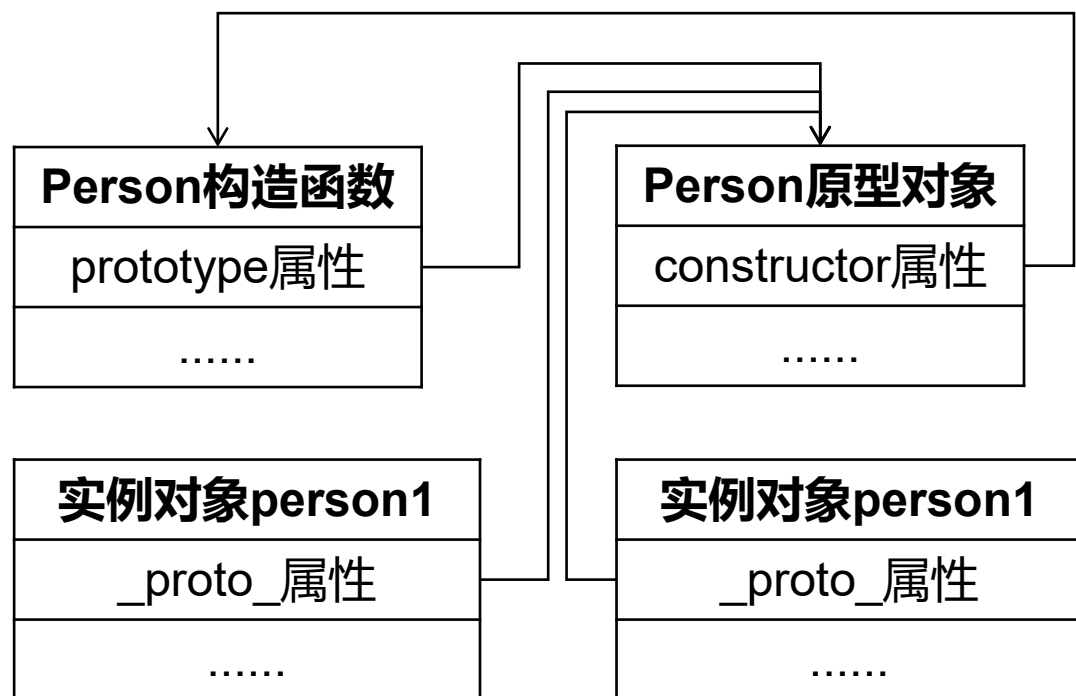
考核知识和技能：

- ✓ JavaScript对象、构造函数
- ✓ JavaScript原型链继承
- ✓ JavaScript构造函数继承
- ✓ JavaScript组合继承

多选题：JavaScript对象

JavaScript对象：构造函数、原型、实例的关系：

- (1) 每个构造函数都有一个原型对象，构造函数都包含一个指向原型对象的指针prototype。
- (2) 原型对象都包含一个指向构造函数的指针constructor。
- (3) 每一个实例都包含一个指向原型对象的内部指针__proto__。

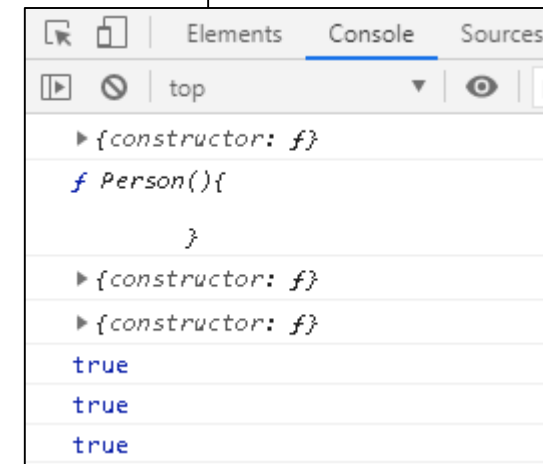


```
//Person构造函数
function Person(){

}

//Person构造函数的原型对象
console.log(Person.prototype);
console.log(Person.prototype.constructor);

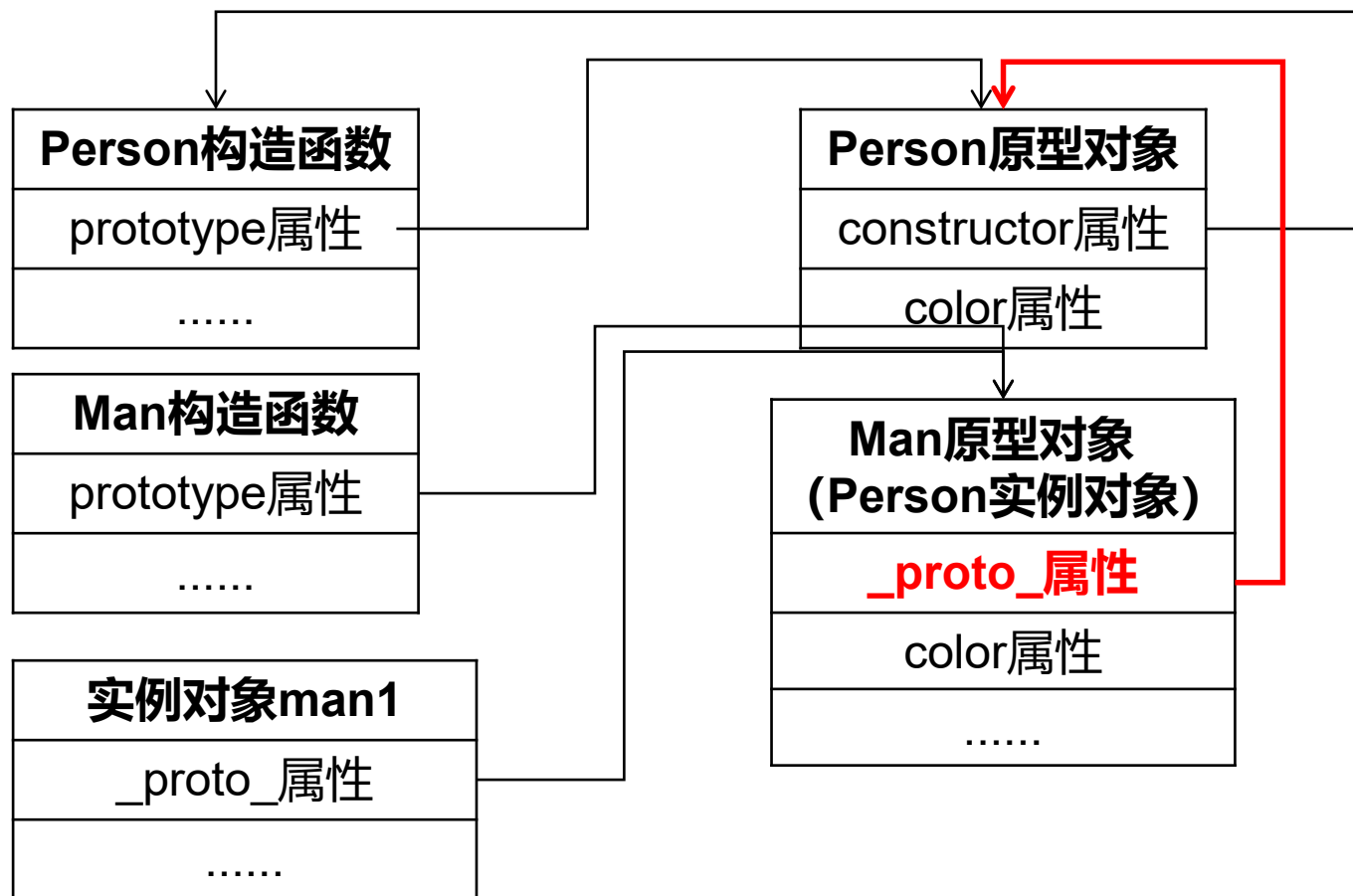
//Person1和Person2实例对象
var person1 = new Person();
var person2 = new Person();
console.log(person1.__proto__);
console.log(person2.__proto__);
//构造函数和每个实例对象指向同一个原型对象
console.log(person1.__proto__ === person2.__proto__);
console.log(person1.__proto__ === Person.prototype);
console.log(person2.__proto__ === Person.prototype);
```



>>> 多选题：JavaScript继承

1、原型链继承

ECMAScript中描述了原型链的概念，并将原型链作为实现继承的主要方法。其基本思想是利用原型让一个引用类型继承另一个引用类型的属性和方法。



```
//Person构造函数
function Person(){
    this.color = ['red','yellow'];
}
```

```
//Man构造函数
function Man(){
}
```

//原型链继承

Man.prototype = new Person();

//实例对象man1和man2

```
var man1 = new Man();
```

```
var man2 = new Man();
```

//访问原型链上属性

```
console.log(man1.color);
```

```
console.log(man2.color);
```

//修改原型链上属性

```
man1.color.push('blue');
```

```
console.log(man1.color);
```

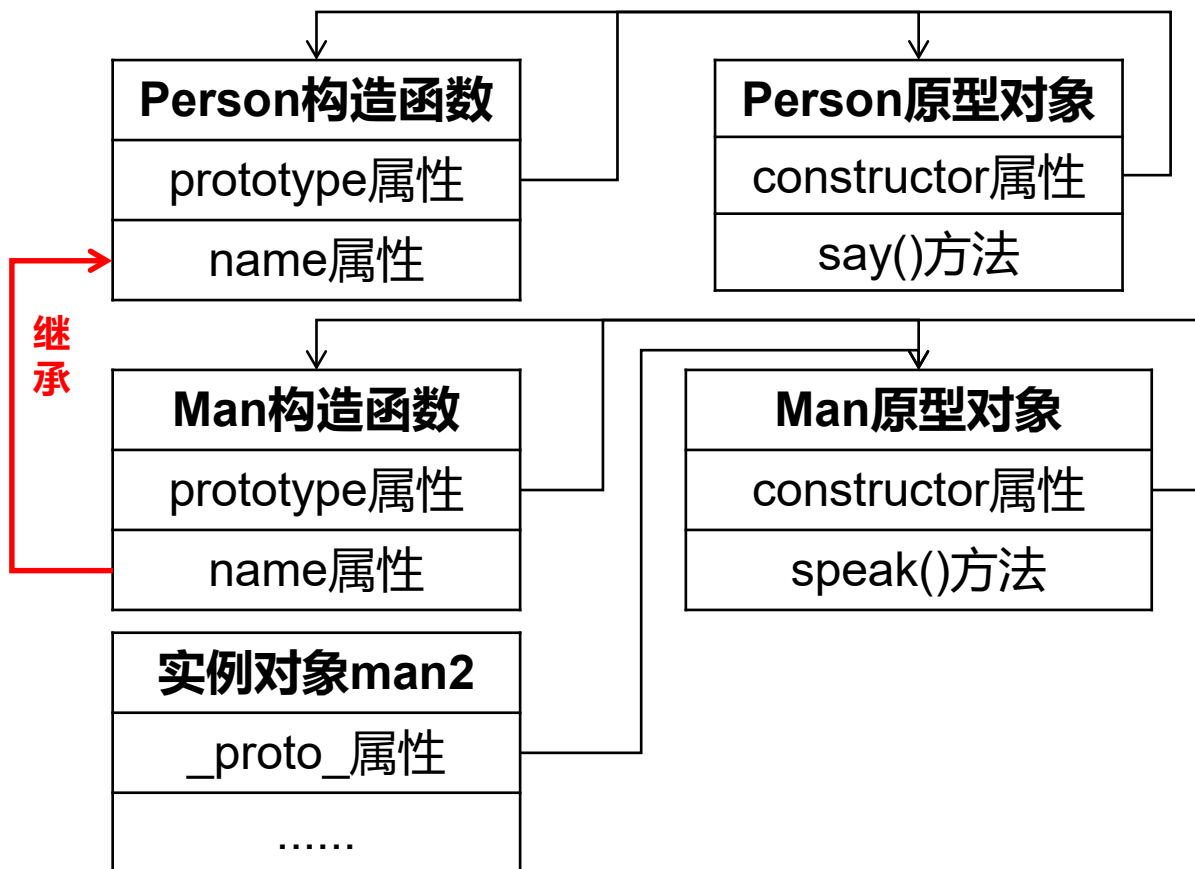
```
console.log(man2.color);
```

```
▶ (2) ["red", "yellow"]
▶ (2) ["red", "yellow"]
▶ (3) ["red", "yellow", "blue"]
▶ (3) ["red", "yellow", "blue"]
```

>>> 多选题：JavaScript继承

2、构造函数继承

构造函数继承的思路相当简单，在构造函数内部调用需要继承的构造函数。通常使用构造函数的`apply()`或者`call()`方法来实现。



```
//Person构造函数
function Person(name){
    this.name = name;
}
Person.prototype.say = function(){
    console.log(this.name);
}
//Man构造函数
function Man(name){
    //构造函数继承
    Person.call(this,name);
}
Man.prototype.speak = function(){
    console.log('speak');
}
//实例对象man1和man2
var man1 = new Man('Mike');
var man2 = new Man('Jake');
console.log(man1.name);
console.log(man2.name);
//可以调用自身原型上的方法
man1.speak();
man2.speak();
//无法调用继承的构造函数原型上定义的方法
man1.say();
man2.say();
```

| |
|---|
| Mike |
| Jake |
| speak |
| speak |
| Uncaught TypeError: man1.say is not a function at <anonymous>:25:6 |

>>> 多选题：JavaScript继承

3、组合继承

集合继承是将原型链继承和构造函数继承模式组合到一起的模式，其思路是使用原型链实现对原型属性和方法的继承，再通过构造函数继承来实现对实例属性的继承。

```
//Person构造函数
function Person(name){
    this.name = name;
    this.color = ['red','yellow'];
}

Person.prototype.say = function(){
    console.log(this.name);
}

//Man构造函数
function Man(name){
    //构造函数继承方式
    Person.call(this,name);
}
//原型链继承方式
Man.prototype = new Person();
```

```
//实例对象man1和man2
var man1 = new Man('Mike');
var man2 = new Man('Jake');
console.log(man1.name);
console.log(man2.name);
man1.say();
man2.say();

man1.color.push('blue');
console.log(man1.color);
console.log(man2.color);
```

| |
|---------------------------------|
| Mike |
| Jake |
| Mike |
| Jake |
| ▶ (3) ["red", "yellow", "blue"] |
| ▶ (2) ["red", "yellow"] |

单选题

Bootstrap中，下面可以实现列偏移的类是？（ ）

- A、.col-md-push-*
- B、.col-md-pull-*
- C、.col-md-move-*
- D、.col-md-offset-*

考核知识和技能：

- ✓ Bootstrap 栅格系统
- ✓ 列偏移
- ✓ 列排序

>>> 单选题：Bootstrap栅格系统

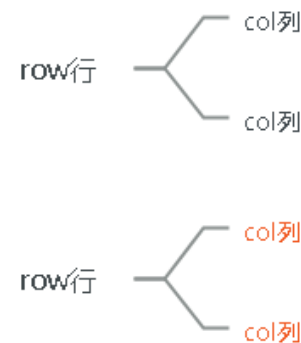
1、Bootstrap栅格系统

Bootstrap栅格系统通过创建一系列容器（container），在容器里建立行（row），行中可以建立列（col），以此来布局和对齐内容。

```
<div class="container">
  <div class="row">
    <div class="col-md-4">列1</div>
    <div class="col-md-8">列2</div>
  </div>
  <div class="row"></div>
</div>
```

| | |
|----|----|
| 列1 | 列2 |
| | |

Container容器

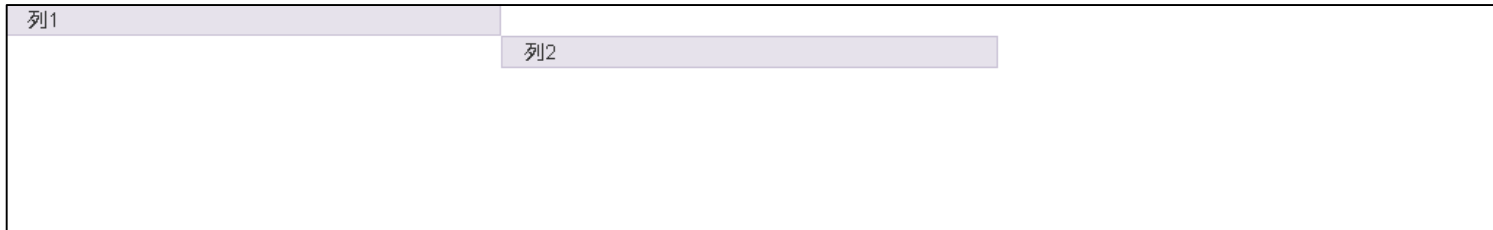


| | 超小屏幕 (新增规格)<576px | 小屏幕 次小屏≥576px | 中等屏幕 窄屏≥768px | 大屏幕 桌面显示器≥992px | 超大屏幕 大桌面显示器≥1200px |
|------------------------|----------------------|------------------|------------------|--------------------|-----------------------|
| .container 最大宽度 | None (auto) | 540px | 720px | 960px | 1140px |
| 类前缀 | .col- | .col-sm- | .col-md- | .col-lg- | .col-xl- |
| 列 (column) 数 | 12 | | | | |

2、列偏移

使用 `.col-md-offset-*` 类可以将列向右侧偏移。这些类实际是通过使用 `*` 选择器为当前元素增加了左侧的边距（margin）。`col-md-offset-*`，是利用 `margin-left` 实现的。`col-md-offset-*` 只能向右偏移。

```
<div class="container">
  <div class="row">
    <div class="col-md-4">列1</div>
  </div>
  <div class="row">
    <div class="col-md-4 col-md-offset-4">列2</div>
  </div>
</div>
```

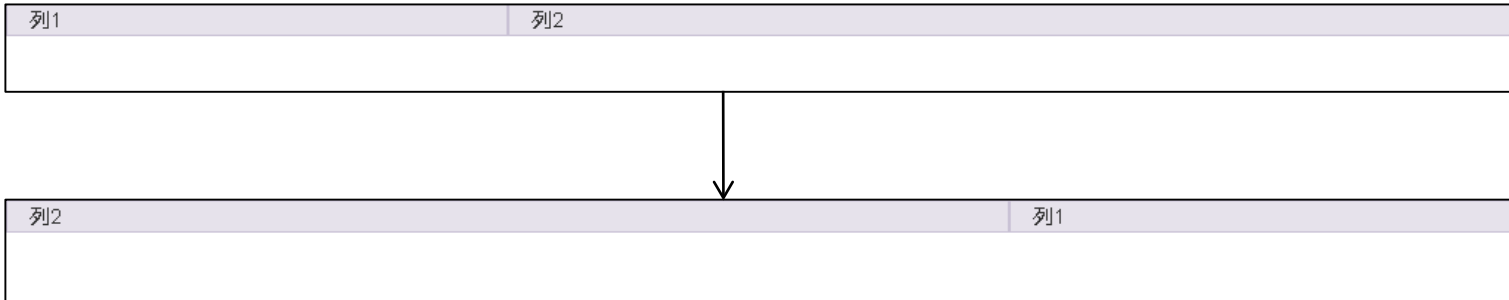


3、列排序

Bootstrap 网格系统另一个完美的特性，就是您可以很容易地以一种顺序编写列，然后以另一种顺序显示列。

通过使用 `.col-md-push-*` 和 `.col-md-pull-*` 类就可以很容易的改变列（column）的顺序。

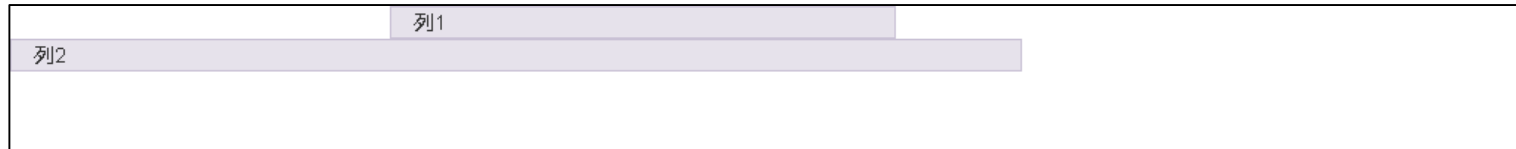
```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-push-8">列1</div>
    <div class="col-md-8 col-md-pull-4">列2</div>
  </div>
</div>
```



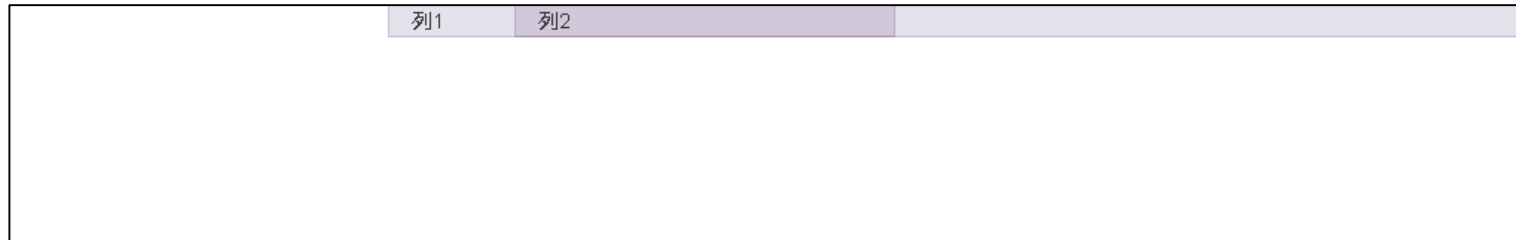
4、列偏移与列排序的区别

如果一行的偏移量+实际的宽度综合超过12，col-md-offset会换行显示，是因为margin，而push/pull只会一部分不可见，是因为相对自身定位。

```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-offset-3">列1</div>
    <div class="col-md-8">列2</div>
  </div>
</div>
```



```
<div class="container">
  <div class="row">
    <div class="col-md-4 col-md-push-3">列1</div>
    <div class="col-md-8">列2</div>
  </div>
</div>
```



从功能上来看，push和pull可以用来给元素换位置，比较灵活。

单选题

关于MySQL存储过程，说法错误的是（ ）。

- A、调用存储过程使用关键字CALL
- B、存储过程的参数在定义时，有两种参数约束，即IN、OUT
- C、创建存储过程的语法是CREATE PROCEDURE
- D、存储过程是一种在数据库中存储复杂程序，以便由外部程序调用的数据库对象

考核知识和技能：

- ✓ 存储过程创建
- ✓ 存储过程调用
- ✓ 存储过程参数

单选题：MySQL 高级编程

1、存储过程

(1) 存储过程 (Stored Procedure) 是一种在数据库中存储复杂程序，以便外部程序调用的一种数据库对象。

(2) 存储过程格式

```
CREATE PROCEDURE <过程名>([过程参数[,...]])  
BEGIN  
    <过程体>  
END;
```

2、存储过程创建

```
mysql> delimiter $$      #将语句的结束符号从分号;临时改为两个$$ (可以是自定义)
```

```
mysql> CREATE PROCEDURE <过程名>([过程参数[,...]])  
-> BEGIN  
->     <过程体>  
-> END$$
```

Query OK, 0 rows affected (0.01 sec)

```
mysql> delimiter;      #将语句的结束符号恢复为分号
```

(1) 如果使用SQL语句包含分号字符的存储过程，会出现问题。

(2) 就需要重新定义MySQL分隔符，需要使用 **delimiter** 命令。

(3) 使用 **delimiter** 命令更改分隔符为//或\$\$。

```
MariaDB [music_web]> create procedure getAdminCount()  
-> begin  
-> select count(*) as num from admin;  
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that  
corresponds to your MariaDB server version for the right syntax to use near '' a  
t line 3  
MariaDB [music_web]>
```

未重定义分隔符报错

```
MariaDB [music_web]> delimiter $$  
MariaDB [music_web]> create procedure getAdminCount()  
-> begin  
-> select count(*) as num from admin;  
-> end  
-> $$  
Query OK, 0 rows affected (0.13 sec)
```

重定义分隔符后成功

3、调用存储过程

```
CALL <过程名>
```

4、存储过程的参数

MySQL存储过程的参数用在存储过程的定义，共有三种参数类型，

IN,OUT,INOUT,形式如：

- (1) IN 参数：表示要参数值要传入该过程
- (2) OUT 参数：表示该参数作为传出值或返回值
- (3) INOUT 参数：表示参数值需要传入并且可以被存储过程修改和返回的值

```
CREATE PROCEDURE <过程名>([IN|OUT|INOUT])  
BEGIN  
    <过程体>  
END;
```

```
MariaDB [music_web]> call getAdminCount();  
+-----+  
| num |  
+-----+  
| 2 |  
+-----+  
1 row in set (0.00 sec)  
  
Query OK, 0 rows affected (0.01 sec)
```

单选题

`$m=10;$n=++$m+10;` 此PHP语句执行后, `$m`和`$n`的值分别为 ()。

- A、 10 21
- B、 11 21
- C、 10 20
- D、 11 20

考核知识和技能:

- ✓ PHP变量
- ✓ ++
- ✓ 运算优先级

>>> 单选题：PHP 运算符优先级

1、PHP变量

PHP 中的变量用一个美元符号后面跟变量名来表示。变量名是区分大小写的。

```
<?php
$m = 10;
echo $m;//输出10
?>
```

2、递增运算符

递增/递减运算符

| 例子 | 名称 | 效果 |
|-------|----|----------------------|
| ++\$a | 前加 | \$a 的值加一，然后返回 \$a。 |
| \$a++ | 后加 | 返回 \$a，然后将 \$a 的值加一。 |
| --\$a | 前减 | \$a 的值减一，然后返回 \$a。 |
| \$a-- | 后减 | 返回 \$a，然后将 \$a 的值减一。 |

```
<?php
$m = 10;
++$m;
echo $m;//输出11
?>
```

单选题：PHP运算符优先级

3、PHP运算符优先级

- (1) 运算符优先级指定了两个表达式绑定得有多“紧密”
- (2) 如果运算符优先级相同，那运算符的结合方向决定了该如何运算。

```
<?php
$m = 10;
$n = ++$m+10; //(++$m)+10;
?>
```

按照PHP的代码规范，运算应该加上括号。列如：

```
<?php
$m = 10;
$n = (++$m) +10;
echo $m;
echo $n;
?>
```

运算优先级

| 结合方向 | 运算符 | 附加信息 |
|------|---|--|
| 不适用 | clone new | clone 和 new |
| 右 | ** | 算术运算符 |
| 不适用 | <div>++ -- ~</div> (int) (float) (string) (array) (object) (bool) @ | 类型、递增 / 递减 |
| 左 | instanceof | 类型 |
| 不适用 | ! | 逻辑运算符 |
| 左 | * / % | 算术运算符 |
| 左 | <div>+ - .</div> | 算术运算符 和 字符串运算符 |
| 左 | << >> | 位运算符 |
| 无 | < <= > >= | 比较运算符 |
| 无 | == != === !== <> <=> | 比较运算符 |
| 左 | & | 位运算符 和 引用 |
| 左 | ^ | 位运算符 |
| 左 | | 位运算符 |
| 左 | && | 逻辑运算符 |
| 左 | | 逻辑运算符 |
| 右 | ?? | null 合并运算符 |
| 左 | ? : | 三元运算符 |
| 右 | = += -= *= **= /= , = %= &= = ^= <<= >>= ??= | 赋值运算符 |
| 不适用 | yield from | yield from |
| 不适用 | yield | yield |
| 不适用 | print | print |
| 左 | and | 逻辑运算符 |
| 左 | xor | 逻辑运算符 |
| 左 | or | 逻辑运算符 |

多选题

以下Laravel路由配置代码，若相关的控制类以及方法，模板文件都存在，正确的是（ ）。

- A、Route::post("/login" , "UserController@login");
- B、Route::get("/index" , function(){ return view("index");});
- C、Route::match(["get" , "post"], "/reg" , "UserController@regist");
- D、Route::any(["get" , "post"], "/user/{id}" , function(\$id){ return "user " . \$id;});

考核知识和技能：

- ✓ Laravel工程
- ✓ 配置虚拟主机
- ✓ Laravel路由、参数

>>> 多选题：Laravel工程

Laravel工程目录

| 文件或文件夹 | 说明 |
|-----------------------|--|
| /app/ | 应用的核心代码 |
| /app/Http/Controllers | 存放控制器文件 |
| /bootstrap/ | 用来存放系统启动时需要的文件 |
| /config/database.php | 应用的配置文件，基本数据库配置文件 |
| /public/ | 含有laravel框架核心的引导文件index.php，这个目录也可用来存放任何可以公开的静态资源，如css，Javascript，images等。 |
| /resources/views/ | 存放视图文件 |
| /routes/web.php | 应用程序的路由文件 |
| /.env | 环境配置文件 |
| /package.json | 该工程依赖的composer组件 |

- app
- bootstrap
- config
- database
- public
- resources
- routes
- storage
- tests
- vendor
- .editorconfig
- .env
- .env.example
- .gitattributes
- .gitignore
- .styleci.yml
- artisan
- composer.json
- composer.lock
- package.json
- phpunit.xml
- readme.md
- server.php
- webpack.mix.js

>>> 多选题：Laravel路由

Laravel路由: routes/web.php

(1) 输出字符串

```
Route::get('/', function () {  
    return 'welcome';  
});
```

← → ↻ ⓘ 127.0.0.1:9090

welcome

(2) get方法请求, 跳转视图

```
Route::get("/index", function() {  
    return view("index");  
});
```

← → ↻ ⓘ 127.0.0.1:9090/index

(3) get方法请求, 控制器处理请求

```
Route::get("路径", "控制器类@方法");
```

← → ↻ ⓘ 127.0.0.1:9090/user

index

```
Route::get("/user", "UserController@index");
```

(4) post方法请求, 控制器处理请求

```
Route::post("路径", "控制器类@方法");
```

```
Route::post("/login", "UserController@login");
```

← → ↻ ⓘ 127.0.0.1:9090/login

login

UserController控制器

```
class UserController extends Controller  
{  
    public function index() {  
        echo "index";  
    }  
    public function login() {  
        echo "login";  
    }  
}
```

多选择题：Laravel路由

(5) GET或POST请求

```
Route::match(["get", "post"], "路径", "控制器@方法");
```

```
Route::match(["get", "post"], "/reg", "UserController@regist");
```

← → ↻ ⓘ 127.0.0.1:9090/reg

regist

(6) 所有请求方式

```
Route::any("/路径", "控制器@方法");
```

```
Route::any("/user", "UserController@index");
```

← → ↻ ⓘ 127.0.0.1:9090/user

index

UserController控制器

```
class UserController extends Controller
{
    public function index() {
        echo "index";
    }

    public function login() {
        echo "login";
    }

    public function regist() {
        echo "regist";
    }
}
```

>>> 多选题：Laravel 路由

(7) 路由参数

① 单个必填参数：

```
Route::get('路径/{参数}', function (参数){ });
```

```
Route::get('/user/{id}', function ($id) {  
    return 'User ' . $id;  
});
```

← → ↻ ⓘ localhost:9090/user/1

User 1

```
Route::get('/user/{id}', "UserController@index");
```

← → ↻ ⓘ 127.0.0.1:9090/user/1

User 1

```
Route::match(["get", "post"], "/user/{id}", "UserController@index");
```

```
Route::any("/user/{id}", "UserController@index");
```

UserController控制器

```
class UserController extends Controller  
{  
    public function index($id) {  
        echo "User " . $id;  
    }  
  
    public function login($account, $password) {  
        echo "account:" . $account. ", password:" .  
$password;  
    }  
  
    public function newIndex($id = null) {  
        echo "id " . $id;  
    }  
}
```

← → ↻ ⓘ 127.0.0.1:9090/user/2

User 2

← → ↻ ⓘ 127.0.0.1:9090/user/3

User 3

>>> 多选题：Laravel 路由

②多个必填参数：

```
Route::get('路径/{参数1}/路径/{参数2}', function (参数1, 参数2) { });
```

```
Route::get('/account/{account}/password/{password}', function ($account, $password) {  
    return "account:" . $account . ", password:" . $password;  
});
```

← → ↻ ⓘ 127.0.0.1:9090/account/admin/password/admin

account:admin, password:admin

```
Route::get('/account/{account}/password/{password}', "UserController@login");
```

← → ↻ ⓘ 127.0.0.1:9090/account/admin1/password/admin1

account:admin1, password:admin1

```
Route::match(["get", "post"], "/account/{account}/password/{password}", "UserController@login");
```

← → ↻ ⓘ 127.0.0.1:9090/account/admin2/password/admin2

account:admin2, password:admin2

match方法

← → ↻ ⓘ 127.0.0.1:9090/account/admin3/password/admin3

account:admin3, password:admin3

any方法

```
Route::any("/account/{account}/password/{password}", "UserController@index");
```

>>> 多选题：Laravel 路由

③可选参数

```
Route::get('路径/{参数?}', function (参数 == null) { });
```

← → ↻ ⓘ 127.0.0.1:9090/newIndex

id

← → ↻ ⓘ 127.0.0.1:9090/newIndex/1

id 1

```
Route::get('newIndex/{id?}', "UserController@newIndex");
```

← → ↻ ⓘ 127.0.0.1:9090/newIndex

id

← → ↻ ⓘ 127.0.0.1:9090/newIndex/2

id 2

```
Route::match(["get", "post"], "newIndex/{id?}",  
"UserController@newIndex");
```

← → ↻ ⓘ 127.0.0.1:9090/newIndex

id

← → ↻ ⓘ 127.0.0.1:9090/newIndex/3

id 3

```
Route::any("newIndex/{id?}", "UserController@newIndex");
```

← → ↻ ⓘ 127.0.0.1:9090/newIndex

id

← → ↻ ⓘ 127.0.0.1:9090/newIndex/4

id 4



THANKS