

《Web前端开发》理论考试

高级试卷分析

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

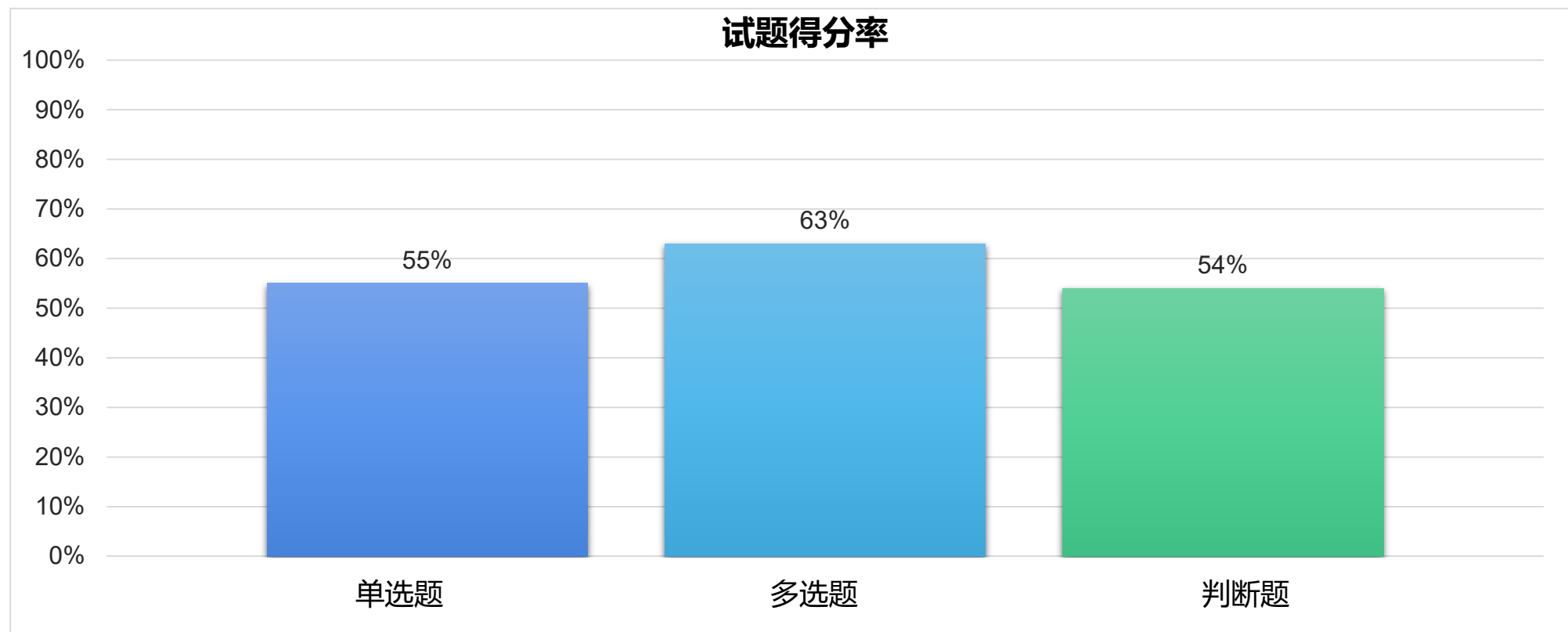


01

总体介绍

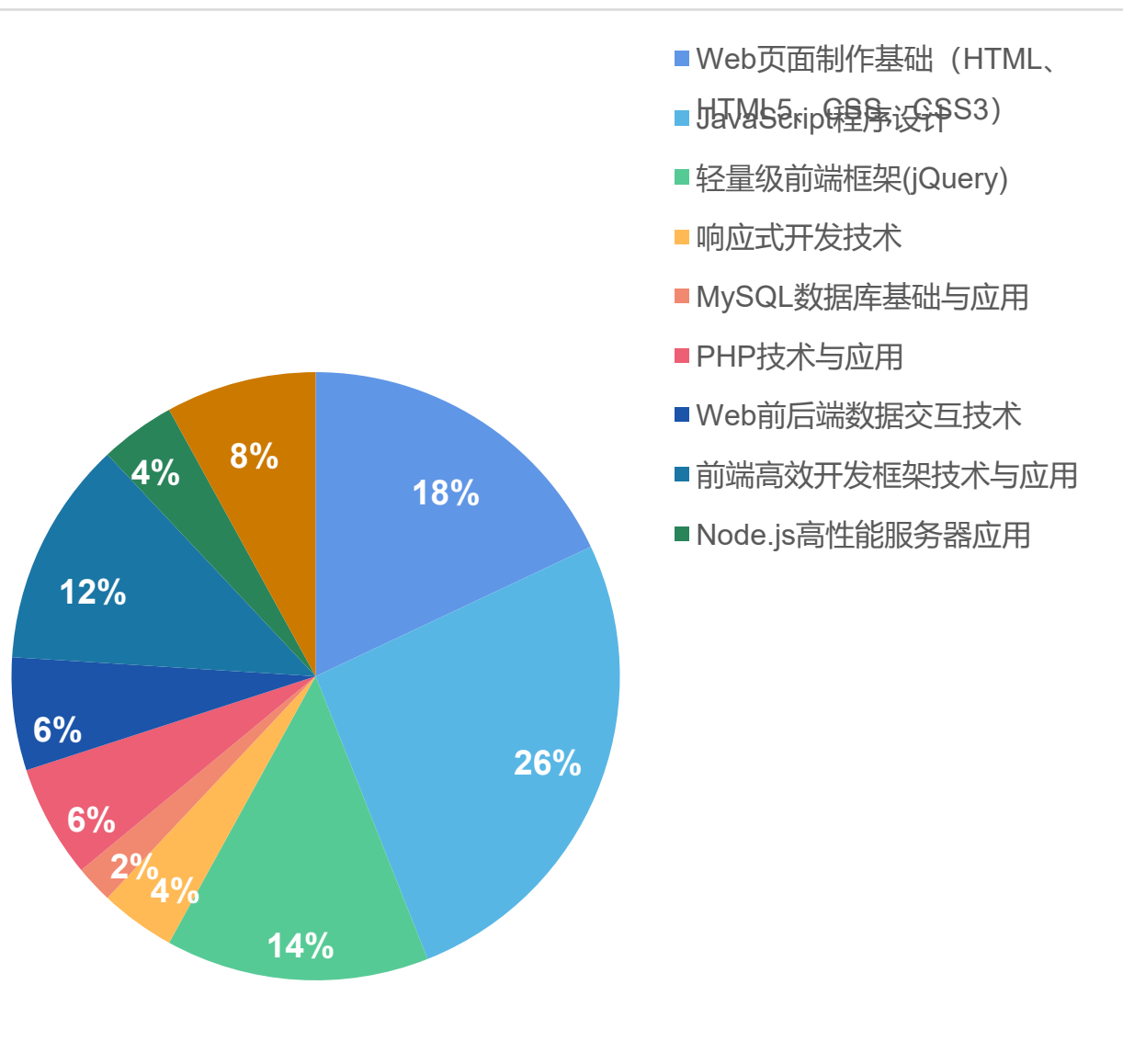
>>> 试卷结构

题型	试题数	得分率
单选题	30	55%
多选题	15	63%
判断题	5	54%



>>> 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)	9	18%
JavaScript程序设计(ES6)	13	26%
轻量级前端框架(jQuery)	7	14%
响应式开发技术	2	4%
MySQL数据库基础与应用	1	2%
PHP技术与应用	3	6%
Web前后端数据交互技术	3	6%
前端高效开发框架技术与应用	6	12%
Node.js高性能服务器应用	2	4%
网站性能优化	4	8%



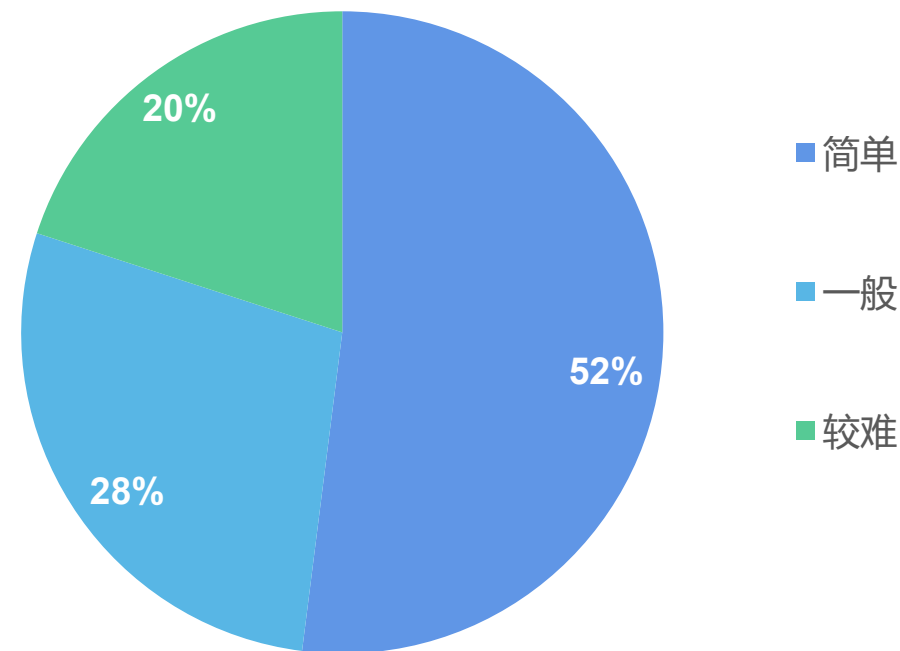
>>> 考试范围：考核点分布

知识模块	考核点(职业技能)
Web页面制作基础 (HTML、HTML5、CSS、CSS3)	HTML标签、CSS引入、选择器、圆角边框、2D转换、Canvas、SVG等
JavaScript程序设计	(1) JavaScript 数据类型、变量、数组、函数、定时器、自定义事件、DOM操作等 (2) ES6 常量、Symbol、箭头函数、解构赋值、模块、数组的扩展、函数的扩展、对象的扩展、babel等
轻量级前端框架(jQuery)	jQuery选择器、DOM操作等
响应式开发技术	Bootstrap栅格系统
MySQL数据库基础与应用	MySQL数据库备份与恢复
PHP技术与应用	PHP数据类型、变量、运算符、数组、数学函数、数组函数等
Web前后端数据交互技术	Ajax、XMLHttpRequest
前端高效开发框架技术与应用	Vue实例对象、指令、渲染、监听、路由以及传参、Vuex等
Node.js高性能服务器应用	Node.js特点、事件模块
网站性能优化	Webpack默认配置文件、插件、打包、前端性能优化方式等

>>> 难易度分析

难易度	题型	题目数	占比	合计
简单	单选题	16	32%	52%
	多选题	8	16%	
	判断题	2	4%	
一般	单选题	9	18%	28%
	多选题	4	8%	
	判断题	1	2%	
较难	单选题	5	10%	20%
	多选题	3	6%	
	判断题	2	4%	

难易度占比





02

典型试题分析



单选题

01. Less



多选题

02. ES6



单选题

03. PHP



判断题

04. Vue.js



多选题

05. Vue.js 路由



单选题

关于LESS的语法，下列说法错误的是？（ ）

- A、可以传递参数的class，就像函数一样
- B、class中不能嵌套class
- C、可以编辑颜色
- D、在CSS中可以使用表达式赋值

考核知识和技能：

- ✓ CSS预处理语言
- ✓ Less变量
- ✓ Less赋值
- ✓ Less混合
- ✓ Less嵌套

1、CSS预处理语言

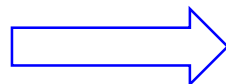
CSS只是一门描述性语言，不能定义变量，也不能嵌套，不利于维护和复用。

CSS预处理语言（Less、Sass），扩充了CSS语言，增加了如变量、混合、嵌套、函数等功能。

Less代码可以转换为相同的CSS代码，如下所示：

```
@width: 10px;  
@height: @width + 10px;  
  
#header {  
  width: @width;  
  height: @height;  
}
```

Less



```
#header {  
  width: 10px;  
  height: 20px;  
}
```

CSS

2、Less

Less 是一门 CSS 预处理语言，它扩充了 CSS 语言，增加了诸如变量、混合（mixin）、函数等功能，让 CSS 更易维护、扩充和复用。

Less编写和运行：

(1) 第一种：

直接在html页面引用.less文件，然后借助less.js去编译less文件动态生成css样式，这种方式适用于开发模式。

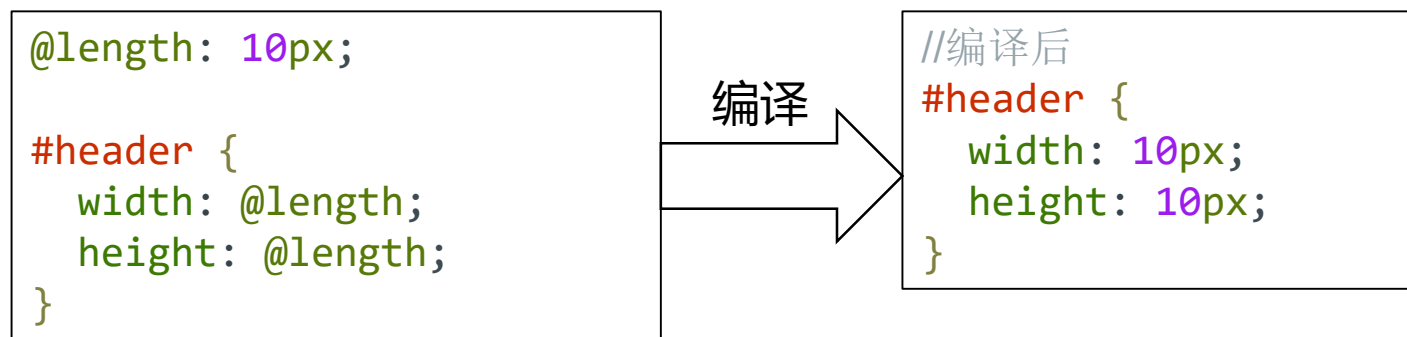
less.js：下载less.js文件，然后引入该文件；也可以直接使用CDN的方式引用less.js

(2) 第二种：

首先写好.less文件的语法，然后借助工具生成对应的.css文件，然后直接引用.css文件即可，这种方式适用于运行环境。
用node.js编译less文件。

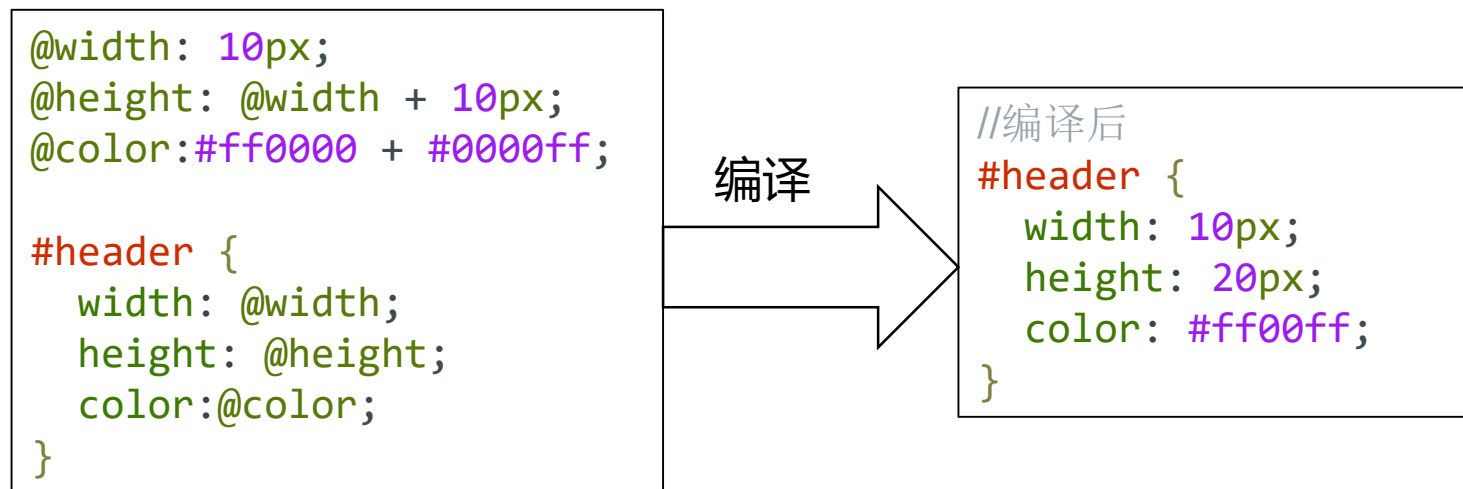
3、Less变量

Less中使用@符号定义变量。



4、Less赋值

Less中对于变量的赋值除了使用固定的数值之外，还可以使用表达式进行赋值。



5、Less混合

混合是一种将一组属性从一个规则集包含（或混入）到另一个规则集的方法，类似于函数。

混合参数：

`.public(@w:默认值,@h:默认值...){}`

```
//@w 表示宽度, @h 表示高度
.public(@w,@h:20px) {
    width:@w;
    height: @h;
}
#header {
    .public(10px);    // 调用
}
```

编译

```
//编译后
#header {
    width: 10px;
    height: 20px;
}
```

6、Less嵌套

Less提供嵌套，可以将选择器进行嵌套使用，编译后会变成后代选择器。

```
.div {  
  width: 120px;  
  height: 120px;  
  .a {  
    width: 100%;  
    height: 100%;  
    span {  
      font-size: 18px;  
    }  
  }  
}
```

编译

```
//编译后  
.div {  
  width: 120px;  
  height: 120px;  
}  
.div .a {  
  width: 100%;  
  height: 100%;  
}  
.div .a span {  
  font-size: 18px;  
}
```

多选题

关于JavaScript箭头函数的描述，正确的是？（ ）

- A、使用箭头符号=>定义
- B、参数超过1个的话，需要用小括号()括起来。
- C、函数体语句超过1条的时候，需要用大括号{ }括起来，用return语句返回。
- D、函数体内的this对象，绑定使用时所在的对象。

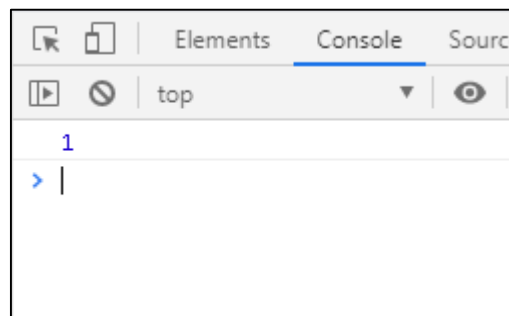
考核知识和技能：

- ✓ JavaScript函数
- ✓ 箭头函数
- ✓ this关键字

1、JavaScript函数

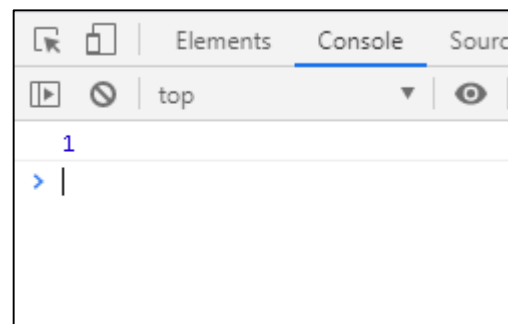
(1) 在ES5以及之前的版本中可以使用function关键字定义函数。

```
function output(p) {  
    return v;  
};  
  
console.log(output(1));
```



(2) ES6允许使用“箭头”(=>)定义函数，具有简洁表达特点。

```
var output = p=>v;  
  
console.log(output(1));  
  
// 等同于  
function output(p) {  
    return v;  
};
```



2、ES6箭头函数

(1) 如果箭头函数不需要参数或需要多个参数，需要使用小括号()括起来。

```
var f = () => 5;  
// 等同于  
var f = function () { return 5 };  
  
var sum = (num1, num2) => num1 + num2;  
// 等同于  
var sum = function(num1, num2) {  
    return num1 + num2;  
};
```

(2) 如果箭头函数的代码块部分多于一条语句，就要使用大括号将它们括起来，并且使用return语句返回。

```
var sum = (num1, num2) => {  
    let all = num1 + num2;  
    return all;  
}
```

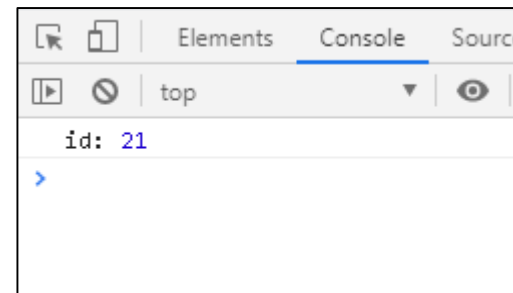
```
// 等同于  
var sum = function(num1, num2) {  
    let all = num1 + num2;  
    return all;  
};
```

3、this关键字

箭头函数体内的this对象，就是定义时所在的对象，而不是使用时所在的对象。

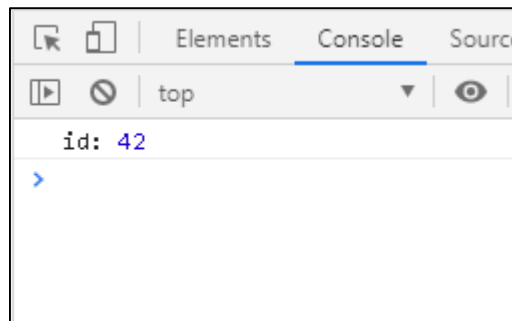
(1) function定义函数

```
function foo() {  
    setTimeout(function(){  
        console.log('id:', this.id);  
    }, 100);  
}  
var id = 21;  
  
foo.call({ id: 42 });
```



(2) 箭头函数

```
function foo() {  
    setTimeout(() => {  
        console.log('id:', this.id);  
    }, 100);  
}  
var id = 21;  
  
foo.call({ id: 42 });
```



单选题

PHP中以下哪一项不能把字符串\$s1和\$s2组成一个字符串？（ ）

- A、 \$s1 + \$s2
- B、 "{\$s1}{\$s2}"
- C、 \$s1.\$s2
- D、 implode("", array(\$s1,\$s2))

考核知识和技能：

- ✓ 变量
- ✓ 运算符
- ✓ 数据类型转换
- ✓ 字符串
- ✓ 数组

1、PHP变量定义

PHP 中的变量用一个美元符号后面跟变量名来表示。变量名只能由数字、字母、下划线组成且只能由字母和下划线开头。变量名区分大小写。

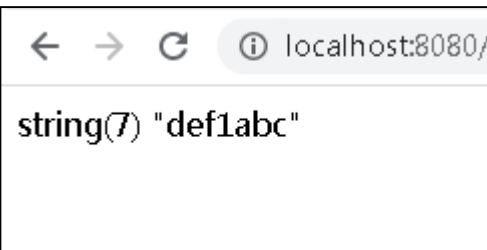
```
<?php  
$s1 = 123;  
$s2 = '1abc';
```

2、PHP运算符

(1) 算术运算符（如+、-、*、/等）

(2) 字符串运算符（.）

```
<?php  
$s1 = 'def';  
$s2 = '1abc';  
  
var_dump($s1.$s2);
```



← → ↻ ⓘ localhost:8080/
string(7) "def1abc"

注：vardump()函数可以打印变量或表达式的所有信息包括值、类型等。

2、PHP类型转换

PHP是一门弱类型语言，在变量定义中不需要明确的类型定义，变量类型是根据使用该变量的值所决定的。

(1) **类型强制转换**：在要转换的变量之前加上用括号括起来的目标类型即可进行强制转换。

语法：(目标类型)变量名

(2) **隐式类型转换**：指PHP自动转换变量的类型，通常发生于使用运算符时多个变量类型不一致的情况下。

①算术运算符会将两边的变量转换为数字类型进行运算。

```
<?php
$s1 = 123;
$s2 = '1abc';
var_dump($s1+$s2);
```

②字符串连接符会将两边的变量转换为字符串类型进行运算。

```
<?php
$s1 = 123;
$s2 = '1abc';
var_dump($s1.$s2);
```

```
← → ↻ ⓘ localhost:8080/
string(7) "1231abc"
```

```
← → ↻ ⓘ localhost:8080/
Notice: A non well formed numeric value encountered in
int(124)
```

3、PHP数组定义

(1) array()语法

PHP可以用 array() 语言结构来新建一个数组。它接受任意数量用逗号分隔的键（key） => 值（value）对。其中键的值可以省略，此时会使用从0开始依次递增的数字赋予键的值。

```
<?php
$s1 = 123;
$s2 = '1abc';
var_dump(array($s1,$s2));
var_dump(array('s1'=>$s1,'s2'=>$s2));
```

```
array(2) { [0]=> int(123) [1]=> string(4) "1abc" } array(2) { ["s1"]=> int(123) ["s2"]=> string(4) "1abc" }
```

(2) []语法

自PHP5.4版本起可以使用短数组定义语法，用 [] 替代 array()。其它语法格式和array()一致

```
<?php
$s1 = 123;
$s2 = '1abc';
var_dump([$s1,$s2]);
var_dump(['s1'=>$s1,'s2'=>$s2])
;
```

```
array(2) { [0]=> int(123) [1]=> string(4) "1abc" } array(2) { ["s1"]=> int(123) ["s2"]=> string(4) "1abc" }
```

4、PHP字符串变量解析

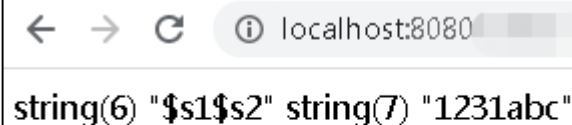
当字符串用双引号定义时，其中的变量将会被解析，分为简单和复杂两种语法规则

(1) 简单语法

当 PHP 解析器遇到一个美元符号 (\$) 时，它去组合尽量多的标识以形成一个合法的变量名。

```
<?php
$s1 = 123;
$s2 = '1abc';

var_dump('$s1$s2');//变量不能被解析
var_dump("$s1$s2");//变量可以被解析
```

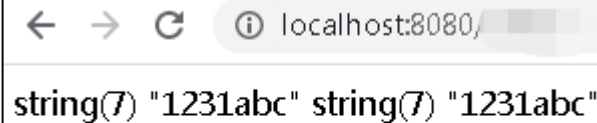


(2) 复杂语法

复杂语法不是因为其语法复杂，而是因为它可以使用复杂的表达式。使用{}花括号将复杂的表达式括起来。

```
<?php
$s1 = 123;
$s2 = '1abc';

var_dump("{ $s1 } { $s2 }");
var_dump("{ ${ 's1' } } { $s2 }");
```



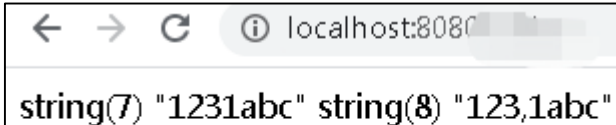
5、PHP字符串函数

(1) 一维数组转化为字符串函数 `implode (string $glue , array $pieces) : string`

`implode()`函数可以用指定的间隔符`$glue`将一维数组`$pieces`中的元素拼接成字符串。

```
<?php
$s1 = 123;
$s2 = '1abc';

var_dump(implode("",array($s1,$s2)));
var_dump(implode(", ",array($s1,$s2)));
```



(2) 字符串转化为一维数组函数 `explode () : array`

判断题

Vue中listener监听的是你定义的变量,当你定义的变量的值发生变化时, 调用对应的方法。()

考核知识和技能:

- ✓ Vue实例对象
- ✓ watch

>>> 判断题：Vue.js引入

1、Vue引入

(1) 引入 Vue.js

```
<!-- cdn引入 -->  
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

```
<!-- 本地引入 -->  
<script src="js/vue.js"></script>
```

(2) NPM 下载安装

```
# 最新稳定版  
npm install vue
```

2、Vue 实例

new vue()

el : 设置Vue实例挂载的元素

data : 保存数据

methods:定义方法

computed:计算属性

watch:侦听属性

components:components

生命周期钩子

```
<body>
  <!--Vue实例控制的元素-->
  <div id="app"></div>
  <!--导入Vue.js文件-->
  <script src="js/vue.js"></script>
  <script type="text/javascript">
    // 实例化一个Vue实例
    var vm = new Vue({// 传入一个JSON对象
      // 提供一个页面上已存在的DOM元素作为Vue实例的挂载目标
      el:"#app",
      data:{},// 保存页面中的数据
      methods: {},// 定义一些方法
      // 计算属性：定义随着所依赖数据变化的属性
      computed: {},
      // 侦听属性：观察和响应 Vue 实例上的数据变动
      watch: {},
      components:{},// 注册组件，注册局部组件
    })
  </script>
</body>
```

>>> 判断题：Vue.js watch监听

3、监听属性watch

watch选项能够监听值的变化

示例：

使用watch监听msg属性的数据变化



```
<div id="app">
  <p>{{newMsg}},{{oldMsg}}</p>
  <input type="text" v-model="msg" />
</div>
<script src="js/vue.js"></script>
<script type="text/javascript">
new Vue({
  el: '#app',
  data: {
    msg: '',
    newMsg: '',
    oldMsg: ''
  },
  watch: {
    // 监听 newMsg属性的数据变化
    msg:function(newVal, oldVal) {
      // 更新显示数据
      this.newMsg = newVal;
      this.oldMsg = oldVal
    }
  }
})
</script>
```

多选题

关于Vue路由传参数，下面说法错误的是？（ ）

- A、使用query方法传入的参数使用this.\$route.query接受
- B、使用params方式传入的参数使用this.\$route.params接受
- C、使用query方法传入的参数使用this.\$router.query接受
- D、使用params方式传入的参数使用this.\$router.params接受

考核知识和技能：

- ✓ \$router
- ✓ \$route
- ✓ 路由传参

1、路由引入

Vue.js 路由需要载入 vue-router 库

(1) 引入js文件

```
<!-- cdn引入 -->  
<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>
```

```
<!-- 本地引入 -->  
<script src="js/vue-router.js"></script>
```

(2) NPM下载安装

```
npm install vue-router
```

>>> 多选题：Vue.js 路由

2、路由定义

```
<script src="js/vue.js"></script>
<script src="js/vue-router.js"></script>
<div id="app">
  <!-- 5. 使用 <router-link>来创建导航. -->
  <!-- 通过传入 `to` 属性指定链接. -->
  <!-- <router-link> 默认会被渲染成一个 `<a>` 标签 -->
  <router-link to="/foo/123">link1</router-link>
  <router-link to="/bar">link2</router-link>
  <!--6. 使用<router-view>渲染组件 -->
  <!-- 路由匹配到的组件将渲染在这里 -->
  <router-view></router-view>
</div>
```

HTML

html

JavaScript

1 定义路由组件

2 定义路由

3 创建 router 实例

4 挂载路由对象

7 实现路由跳转

5 使用<router-link>

6 使用<router-view>
渲染组件

JavaScript

```
//1. 定义 (路由) 组件。可以从其他文件 import 进来
var Foo = { template:'<h1>Foo </h1>' };
var Bar = { template:'<h1>Bar </h1>' };
// 2. 定义路由
// 每个路由应该映射一个组件。
var routes = [
  { path: '/foo/:id', component: Foo },
  { path: '/bar', component: Bar }
]

//3. 创建 router 实例，然后传 `routes` 配置
var router = new VueRouter({
  routes// (缩写) 相当于 routes: routes
});
//4. 创建和挂载根实例。
//要通过 router 配置参数注入路由，
//从而让整个应用都有路由功能
var vm = new Vue({
  el: '#app',
  router
})
```

link1 link2

点击link1

link1 link2

Foo

3、路由实例

(1) 全局路由实例 \$router:

router是VueRouter的一个对象，通过Vue.use(VueRouter)和VueRouter构造函数得到一个router的实例对象，这个对象中是一个全局的对象，他包含了所有的路由包含了许多关键的对象和属性

在 Vue 实例，通过Vue实例自身的\$router 属性访问路由实例。

①this.\$router.push(): 实现路由跳转

②this.\$router.go(): 页面刷新、前进或后退

(2) 路由信息对象\$route:

当前路由的状态信息，包含当前 URL 的信息和 URL 匹配到的路由记录

route是一个跳转的路由对象，每一个路由都会有一个route对象，是一个局部的对象，通过Vue实例自身的\$route属性访问路由信息对象，可以获取对应的name,path,params,query等

①this.\$route.params: 动态路径参数对象

②this.\$route.query: 查询参数对象

③this.\$route.path: 当前路由的路径，如 "/login"

4、路由传参

路由传参主要有两种方式，动态路径传参params和查询参数传参query。

(1) 动态路径传参

```
//通过params属性进行动态路径传参
//生成的路由为 /foo/123
this.$router.push({ path: '/foo', params: { id: '123' } })

//通过$route的params属性获取动态路径传参
this.$route.params.id
```

(2) 查询参数传参

```
//通过query属性进行查询参数传参
//生成的路由为 /bar?id=123
this.$router.push({ path: '/bar', query: { id: '123' } })

//通过$route的query属性获取查询参数传参
this.$route.query.id
```



THANKS