|  |  |  |
| --- | --- | --- |
| **PHP** | | |
| isset() | 判断变量是否为空 |  |
| empty() | 判断变量值是否为空 |  |
| is Array() | 判断是否为数组 |  |
| count() | 输出数组length |  |
| explode() | explode(",",string) |  |
| 数据变换 | function changeToObject(str) {  return eval('(' + str.replace(/&quot;/g, '"') + ')')  }  var slide\_datas\_format= changeToObject('{{json\_encode($slide\_datas\_format)}}') |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **laravel** | | |
| 打印输出 | **@php** echo '<pre>'; print\_r($recommends); echo '</pre>'; **@endphp** |  |
|  | **@foreach(**$res['products'] as $k=>$g**)** <div class="shop-app-product-info" id="{{$g->**product\_id**}}">  <img src="{{$g->**product\_preview**}}"  class="center-block">  <p class="shop-app-product-name">{{$g->**product\_name**}}</p>  <p class="shop-app-product-price">￥{{$g->**product\_price**}}</p> </div> **@endforeach** |  |
|  | **@for(**$i = 0; $i < count(explode(",",$product['product\_preview\_images'])); $i++**) @endfor** |  |
| $loop | **@foreach (**$users as $user**)  @if (**$loop->**first)** This is the first iteration.  **@endif   @if (**$loop->**last)** This is the last iteration.  **@endif** <p>This is user {{ $user->**id** }}</p> **@endforeach** |  |
| $loop->parent | **@foreach (**$users as $user**)  @foreach (**$user->**posts** as $post**)  @if (**$loop->**parent**->**first)** This is first iteration of the parent loop.  **@endif  @endforeach @endforeach** |  |
| $loop->index | 当前循环所迭代的索引，起始为 0。 |  |
| $loop->iteration | 当前迭代数，起始为 1。 |  |
| $loop->remaining | 循环中迭代剩余的数量。 |  |
| $loop->count | 被迭代项的总数量。 |  |
| $loop->first | 当前迭代是否是循环中的首次迭代。 |  |
| $loop->last | 当前迭代是否是循环中的最后一次迭代。 |  |
| $loop->depth | 当前循环的嵌套深度。 |  |
|  | **@php** // **@endphp** |  |
| 引入子视图 | **@include(**'shared.errors'**)** |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **补充** | | |
| 输入框只允许输入数字 | onkeyup="this.value=this.value.replace(/\D/g,'')" onafterpaste="this.value=this.value.replace(/\D/g,'')" |  |
| a标签跳转电话 | <a class="name\_tel" href="tel:18202769655"></a> |  |
| 页面回退并刷新 | **window**.**self**.**location**=**document**.**referrer**; |  |
| 单元格自适应 | white-space: nowrap; |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Vue** | | |
| 声明vue对象 | <div id="app">{{demo}}</div>  var aaa = new Vue({  **el**:'#app',  **data**:{  **demo**:'第一个demo'  } })  解绑：aaa.$destroy()  手动挂载：$mount(‘#id’) //选择器 | id声明不能放在html和body上 |
| 绑定标签属性 | <span v-bind:class='messege'></span> |  |
| 判断显示隐藏 | <span v-if = "aaa">有没有</span>  var app3 = new Vue({  **el**:'#app-3',  **data**:{  **aaa**:false//为true显示，为false不显示  } }) |  |
| for循环遍历 | <div id="app-4">  <ul>  <li v-for="value in todos">{{value}}</li>  </ul> </div>  var app4 = new Vue({  **el**:"#app-4",  **data**:{  **todos**:{**a**:123,**b**:456,**c**:789}  } }) |  |
| 事件绑定 | <div id="app-5">  <span>{{messege}}</span>  <button v-on:click='change'>改变</button> </div>  var app5 = new Vue({  **el**:'#app-5',  **data**:{  **messege**:'第一个'  },  **methods**:{  change:function(){  if(this.**messege** == '第一个'){  this.**messege** = "第二个";  }else{  this.**messege** = "第一个";  }  }  } }) |  |
| 模型 | <div id="app-6">  <p>{{messege}}</p>  <input type="text" v-model='messege'/> </div>  var app6 = new Vue({  **el**:"#app-6",  **data**:{  **messege**:'大家好'  } }) |  |
| 构建组建 | <div id="app">  <ul>  <todo-item v-for='item in dotos' v-bind:todo='item'></todo-item>  </ul> </div>  Vue.component('todo-item',{  **props**:['todo'],  **template**:'<li>{{todo.text}}</li>' })  var app = new Vue({  **el**:"#app",  **data**:{  **dotos**:[  {**text**:123},  {**text**:456},  {**text**:789}  ]  } }) | 组件声明必须在vue对象建立前面 |
| v-once | 只允许渲染一下，数据改变，视图不变 |  |
| v-html |  |  |
| 筛选 | <div id="app2">  <div>{{messege|fil1|fil2}}</div> </div>  var app2 = new Vue({  **el**:'#app2',  **data**:{  **messege**:'ccc'  },  **filters**:{  fil1:function(value){  if(value=="aaa"){  return "123"  }else{  return "456"  }  },  fil2:function(value){  if(value=='ccc'){  return '789'  }else{  return '159'  }  }  } }) | 如果没有return默认返回空“” |
| .的形式表示事件得调用 | <form v-on:submit.prevent="onSubmit"></form> |  |
| 缩写 | <!-- 完整语法 --> <!-- 缩写 -->  <a v-bind:href="url"></a> <a :href="url"></a>  <!-- 完整语法 --> <!-- 缩写 -->  <a v-on:click="doSomething"></a> <a @click="doSomething"></a> |  |
| 计算属性 | <div id="app">  <span>{{messges}}</span>  <span>{{aaa}}</span> </div>  var app = new Vue({  **el**:"#app",  **data**:{  **messges**:'hello'  },  **computed**:{  aaa:function(){  return this.**messges**.split('').reverse('').join('');  }  } }) |  |
| jspang（Vue） | | |
| directive  自定义指令 | <p v-cheng='color'>{{num}}</p> <button @click='add'>加一</button>  Vue.directive('cheng',function(el,binding){  **console**.log(el);  **console**.log(binding);  el.**style**.**color**=binding.**value**; })  Vue.directive('cheng',{bind,inserted,update,componentUpdated}) | jspang  后面接一个对象，分别是五个周期函数 |
| extend扩展实力构造器 | var authorExtend=Vue.extend({  **template**:'<p><a :href="authorURL">{{authorName}}</a></p>',  data:function(){  return {  **authorName**:'Lion',  **authorURL**:'http://baidu.com'  }  } }) new authorExtend().$mount('aaa') new authorExtend().$mount('#aaa') |  |
| Vue.set全局操作 | function change(){  Vue.**set**(app,'count',6) // 改编数据的对象（app/mydata） 属性 改变的值 } |  |
| 生命周期（钩子函数） | beforeCreate:function(){},初始化之后 created:function(){},创建完成 beforeMount:function(){},挂载之前 mounted:function(){},被创建 beforeUpdate:function(){},数据更新前 updated:function(){},被更新后 activated:function(){},activated deactivated:function(){},deactivated beforeDestroy:function(){},销毁之前 destroyed:function(){}销毁之后 |  |
| Template  三种方式使用template | <div id="app">  <div id="tem\_1">  <p>11</p>  </div> </div> <div id="app2">  </div> <div id="app3">  <script type="x-template" id="demo3">  <div>  <p>33</p>  <p>33</p>  </div>  </script> </div>  var app=new Vue({  **el**:'#app',  **data**:{  **message**:'hello Vue!'  },  **template**:'#tem\_1' }); var app2 = new Vue({  **el**:'#app2',  template:`<div><p>222</p><p>222</p><p>222</p></div>` }) var app3=new Vue({  **el**:'#app3',  **template**:'#demo3' }) | template模板标签里面只能放一个标签，所以一般都是放一个大盒子 |
| 组件component | //全局组件 Vue.component('Lion',{  **template**:'<div style="color: red"><div>全剧组件1</div><div>全剧组件1</div></div>' })  //局部组建 var app2 = new Vue({  **el**:'#app2',  **components**:{  **'liona'**:{  **template**:'<div style="color: green;">局部组建</div>'  },  **'ynn2'**:{  **template**:'<div style="color: green;">局部组建2</div>'  }  } }) | 局部：components  全局：  component |
| component的props属性，绑定 | <lion v-bind:mess="mess"></lion>  var app = new Vue({  **el**:'#app',  **data**:{  **mess**:'传值'  },  **components**:{  **'lion'**:{  **template**:'<div>我是12345，你是 {{ mess }} </div>',  **props**:['mess']  }  } }) |  |
| 子父组件 | var ynn={  **template**:'<div style="color: green">我是子组件</div>' } var app = new Vue({  **el**:'#app',  **components**:{  **'lion'**:{  **template**:'<div style="color: red">我是父组件<ynn></ynn></div> ',  **components**:{  **'ynn'**:ynn  }  }  } }) |  |
| component标签 | <component v-bind:is='who'>wos</component>  <button @click='change'>切换</button>  var linoA={  **template**:'<div>我是lionA</div> ' }; var linoB={  **template**:'<div>我是lionB</div> ' } var app= new Vue({  **el**:'#app',  **data**:{  **who**:'linoA'  },  **components**:{  **'lionA'**:linoA,  **'lionB'**:linoB  },  **methods**:{  change:function(){  if(this.**who**=='lionA'){  this.**who**='lionB'  }else if(this.**who**=='lionB'){  this.**who**='lionA'  }  }  } }) | 绑定的属性必须为is  v-bind:is=“” |
| 选项  propsData | var order = Vue.extend({  **template**:'<div><p>{{mess}}</p><p>{{aaa}}</p></div>',  data:function(){  return {  **mess**:'我是一个扩展'  }  },  **props**:['aaa'] }) new order({**propsData**:{**aaa**:123456}}).$mount('#app') |  |
| 选项  computed | <p>{{newmess}}</p> <p>{{**mess**}}</p>  var app=new Vue({  **el**:'#app',  **data**:{  **mess**:100  },  **computed**:{  newmess:function(){  return '￥'+this.**mess**+'元'  }  } }) |  |
| 选项  methods | var order= {  **template**:'<button>组件加2</button>' } var app=new Vue({  **el**:'#app',  **data**:{  **mess**:1  },   **methods**:{  add:function(num,a){  **console**.log(a);  if(!num==''){  this.**mess**+=num  }else{  this.**mess**++  }  }  },  **components**:{  **order**:order  }, }) |  |
| 选项  watch监听 | var arr=['第一种','第二种','第三种']; var app=new Vue({  **el**:"#app",  **data**:{  **wendu**:10,  **zuo**:arr[1]  },  **methods**:{  add:function(){  this.**wendu**+=5  },  reduce:function(){  this.**wendu**-=5  }  },  **watch**:{  wendu:function(newV,oldV){  if(newV>=40){  this.**zuo** =arr[2]  }else if(newV>=20&&newV<40){  this.**zuo** =arr[1]  }else{  this.**zuo** =arr[0]  }  }  } })  eg:app.$**watch**('xxx',**function**(){}) | 局部：watch  全局：$watch |
| 选项:  混入mixins | var addlog={  **data**:{  **aaa**:20  },  **methods**:{  addadd:function(){  this.aaa++  }  } } var app=new Vue({  **el**:'#app',  **data**:{  **num**:1,  },  **methods**:{  add:function(){  this.**num**++  }  },  **mixins**:[addlog] })  全局混入：Vue.mixin（） | 局部：mixins  全局：mixin |
| 引用JQ | <button @click='change()'>change</button> <button @click='changeagaine()'>changeagaine</button> <button onclick='app.onadd()'>onadd</button>  var app=new Vue({  **el**:'#app',  **data**:{  **mess**:123  },  **methods**:{  change:function(){  $('#app p').html('改变了')  },  changeagaine:function(){  $('#app p').html('又改变了')  },  onadd:function(){  $('#app p').html('外部改变')  }  } }) |  |
|  |  |  |
|  |  |  |
|  |  |  |
| 实例方法 | 1. $mount()挂在 2. $destory()卸载 3. $forceUpdate()更新 4. $nextTick()数据修改 |  |
| 实例事件 | <button onclick='reducebox()'>reduce</button>  app.$on('reduce',function(){  this.**num**-- }) function reducebox(){  app.$emit('reduce') } | $on |
|  | <button onclick="oncebox()">once</button>  app.$once('**once**',function(){  this.**num**=this.**num**\*10 }) function oncebox(){  app.$emit('**once**') } | $once |
|  | <button onclick="offbox()">off</button>  function offbox(){  app.$off('reduce') } | $off |
| 内置组件 | <div id="app">  <lion>  <span slot='name'>{{**dialogdata**.**name**}}</span>  <span slot='url'>{{**dialogdata**.**url**}}</span>  <span slot='skill'>{{**dialogdata**.**skill**}}</span>  </lion> </div> <template id='demo'>  <div>  <p>名称：<slot name='name'></slot></p>  <p>地址：<slot name='url'></slot></p>  <p>标签：<slot name='skill'></slot></p>  </div> </template>  var lion={  **template**:'#demo' } var app=new Vue({  **el**:'#app',  **data**:{  **dialogdata**:{  **name**:'百度',  **url**:'www.baidu.com',  **skill**:'web前端'  }  },  **components**:{  **lion**:lion  } }) |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Sea.JS | | |
| 配置参数 | seajs.config |  |
| 定义模块 | define(function(require,exports,module){  var module1 = require('./module-1.js');//引入其他模块  module1.**num**() }) |  |
| 公开模块方法 | define(function(require,exports,module){  function num(){  **console**.log(111);  }  exports.**num** = num }) | module.exports=  exports  也可以使用return |
| 公开模块对象 | define(function(require,exports,module){  var obj ={};  obj.num=function(a,b){  return (+a)+(+b)  };  obj.reduce=function(a,b){  return (+a)-b  }  module.exports=obj }) |  |
| 使用main文件中的方法 | define(function(require,exports,module){  function mainShow(){  **console**.log('mian');  }  exports.**mainShow** = mainShow; })  seajs.use('./main.js',function(main){  main.**mainShow**() }) |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **nodeJS常用包** | | |
| formidable | https://www.npmjs.com/package/formidable | 表单上传 |
| silly-datetime | https://www.npmjs.com/package/silly-datetime | 时间 |
| ejs | https://www.npmjs.com/package/ejs | 模本引擎 |
| jade | https://www.npmjs.com/package/jade |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **nodeJS** | | |
| 引用模块 | // 引用所需功能 var http = require('http'); |  |
| 创建服务器 | var server = http.createServer(function (req,res) {  //res：request res：respones}) |  |
| 返回头 | **res**.writeHead(200,{**"Content-type"**:"text/html;charset=UTF-8"}); |  |
| 返回数据 | res.end("哈哈哈，这是第一个demo"); |  |
| 监听 | //运行服务器，监听3000端口（端口号可以任改） server.listen(3000,"127.0.0.1"); |  |
| 书写文本 | res.write("<h1>我是一级标题</h1>") | 写在end后面 |
| req.url | /iinde/fdkk/de?wa=11&qq=22 |  |
| url.parse(req.url) | {  protocol: null,  slashes: null,  auth: null,  host: null,  port: null,  hostname: null,  hash: null,  search: '?wa=11&qq=22',  query: 'wa=11&qq=22',0  pathname: '/iinde/fdkk/de',  path: '/iinde/fdkk/de?wa=11&qq=22',  href: '/iinde/fdkk/de?wa=11&qq=22' } |  |
| url.parse(req.url，true) | query会以对象输出 ：query: { wa: '11', qq: '22' }, |  |
| 创建文件夹 | fs.mkdir("./album/aaa"); |  |
| stat检测状态 | fs.stat("./album/bbb",function(err,data){  console.log(data.isDirectory());  });  data.isDirectory()//检测是否是文件夹  data.isFile()//检测是否是文件 | data是fs.Stats对象 |
| path模块  获取扩展名 | path.extname(pathname); |  |
| npm命令 | npm init 配置package文件  npm install 安装一依赖 |  |
| 绝对路径 | \_\_dirname+'/text.txt' |  |
| 转换数据 | querystring.**parse** |  |
| 获取post请求函数（写死） | req.addListener('data',function(chunk){  alldate+=chunk })  req.addListener('end',function(){  console.log(alldate.toString());  var obj = querystring.**parse**(alldate.toString())  console.log(obj); }) | 会将数据传输分成每一小段，在end函数中输出最后的数据 |
| 请求方式 | req.**method** |  |
| 字符串小写 | toLowerCase() |  |
| formidable插件 | http.createServer(function(req,res){  if (req.**url** == '/dopost' && req.**method**.toLowerCase() == 'post') {  // 创建一个对象  var form = new formidable.**IncomingForm**();  //图片上传位置  form.**uploadDir** = "./uploads";  //执行里面的回调函数的时候，表单已经全部接收完毕了。  form.parse(req, function(err, fields, files) {  console.log(fields);//纯文件以外其他信息  console.log(files);//上传文件信息  //console.log(util.inspect({fields: fields, files: files}));  res.end("end")  });  return;  } }).listen(3000,'192.168.1.49') |  |
|  | util.**inspect**({**fields**: fields, **files**: files}) |  |
| 修改文件名 | fs.rename(oldpath,newpath,function(err){}) |  |
|  |  |  |
| **express框架** | | |
| 基本引用express框架 | var express = require('express'); var app = express();  app.get('/',function(req,res){  res.send("hello world") }) | get请求网址 |
| 获取正则匹配 | req.**params**[0] |  |
| router简单实用 | app.get('/aaa/:add',function(req,res){  var routerName = req.**params**["add"];  if(/[\d]{6}/.test(routerName)){  res.send(routerName)  }else{  res.send('错误')  } }) | 无视大小写 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |