# 前端

## 一、单选题

1-5：BADCB 6-10：DCACB 11-15：ACCBD

16-20：DADCC 21-25：DBBDC 26-30：AABBA

31-35：ACBDB 36-40：DBCDB 41-45：DCDAA

## 二、简答题

46、

添加： someNode.appendChild(newNode);

移动： someNode.insertBefore(newNode,null);

移除： someNode.removeChild(someNode.firstChild);

someNode.remove()(删除自身)

复制： someNode.cloneNode(true深复制/ false浅复制)

创建： createDocumentFragment() //创建一个DOM片段

createElement() //创建一个具体的元素

createTextNode() //创建一个文本节点

替换： replaceChild()

查找：

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值

getElementById() //通过元素Id，唯一性

47、



48、

总结

Vue 优点： \* 平缓的学习曲线。 \* 干净的代码。 \* 轻量级的框架。

React 优点： \* 灵活。 \* 拥有大型的技术生态系统。 \* 良好的组件化设计。

Angular 优点： \* 基于 TypeScript。 \* 面向对象编程。

建议：

1. 小型、业务简单的项目建议使用 Vue 框架。
2. 大型、业务复杂的项目建议使用 React 框架。

49、

function sum() {

var nResult = 0;

for (var i = 0, l = arguments.length; i &lt; l; i++) {

nResult += (!isNaN(arguments[i]) &amp;&amp; window.parseFloat(arguments[i]))||0;

}

return nResult.toFixed(3) \* 1000 / 1000;

}

50、

function traverse(oNode) {

var aResult = [];

oNode = oNode || document.body;

if (oNode.style) {

var nWidth = window.parseInt(oNode.style.width, 10) || 0;

var nHeight = window.parseInt(oNode.style.height, 10) || 0;

if (nWidth &gt; 50 &amp;&amp; nHeight &gt; 50) {

aResult.push(oNode);

}

}

var aChildNodes = oNode.childNodes;

if (aChildNodes.length &gt; 0) {

for (var i = 0, l = aChildNodes.length; i &lt; l; i++) {

var oTmp = aChildNodes[i];

aResult = aResult.concat(traverse(oTmp));

}

}

return aResult;

}

# 后端

## 一、单选题

1-5：BCADA 6-10：ACBCC 11-15：DBDCB

16-20：CBBAB 21-25：CBBCA 26-30：BDCDA

31-35：CDCAB 36-40：BC[ABD]BA 41-45：CCCAC

46-50：DCCBA 51-55：BCDDA 56-60：ACBDA

61-65：BBDAB

## 二、简答题

**66、答：**

a.String是只读字符串，也就意味着String引用的字符串内容是不能被改变的。

b.StringBuffer/StringBuilder类表示的字符串对象可以直接进行修改。

c.StringBuilder和StringBuffer的方法完全相同，区别在于它是在单线程环境下使用的，因为它的所有方法都没有被synchronized修饰，因此效率也比StringBuffer要高。

d.StringBuffer由于所有方法都被synchronized修饰，所以它是线程安全的。

**67、答：**

Java中的方法重载发生在同一个类里面两个或者是多个方法的方法名相同但是参数不同的情况。与此相对，方法覆盖是说子类重新定义了父类的方法。方法覆盖必须有相同的方法名，参数列表和返回类型。覆盖者可能不会限制它所覆盖的方法的访问。

**68、答：（可能还有其他的，没有列举完）**

@SpringBootApplication:

包含@Configuration、@EnableAutoConfiguration、@ComponentScan通常用在主类上。

@Repository:

用于标注数据访问组件，即DAO组件。

@Service:用于标注业务层组件。

@RestController:  
用于标注控制层组件(如struts中的action)，包含@Controller和@ResponseBody。  
  
@ResponseBody  
表示该方法的返回结果直接写入HTTP response body中  
一般在异步获取数据时使用，在使用@RequestMapping后，返回值通常解析为跳转路径，加上@responsebody后返回结果不会被解析为跳转路径，而是直接写入HTTP response body中。比如异步获取json数据，加上@responsebody后，会直接返回json数据。

@Component

泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。

@ComponentScan  
组件扫描。个人理解相当于<context:component-scan>，如果扫描到有@Component @Controller @Service等这些注解的类，则把这些类注册为bean。  
  
@Configuration  
指出该类是 Bean 配置的信息源，相当于XML中的<beans></beans>，一般加在主类上。

@Bean:  
相当于XML中的<bean></bean>,放在方法的上面，而不是类，意思是产生一个bean,并交给spring管理。

@EnableAutoConfiguration  
让Spring Boot根据应用所声明的依赖来对Spring框架进行自动配置，一般加在主类上。  
  
@AutoWired:  
byType方式。把配置好的Bean拿来用，完成属性、方法的组装，它可以对类成员变量、方法及构造函数进行标注，完成自动装配的工作。  
当加上（required=false）时，就算找不到bean也不报错。

@Qualifier：  
当有多个同一类型的Bean时，可以用@Qualifier("name")来指定。与@Autowired配合使用  
  
@Resource(name="name",type="type")  
没有括号内内容的话，默认byName。与@Autowired干类似的事。

@RequestMapping：

RequestMapping是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

该注解有六个属性：

params:指定request中必须包含某些参数值是，才让该方法处理。  
headers:指定request中必须包含某些指定的header值，才能让该方法处理请求。  
value:指定请求的实际地址，指定的地址可以是URI Template 模式  
method:指定请求的method类型， GET、POST、PUT、DELETE等  
consumes:指定处理请求的提交内容类型（Content-Type），如application/json,text/html;  
produces:指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回

@RequestParam：用在方法的参数前面。

@PathVariable:路径变量。参数与大括号里的名字一样要相同。

@Profiles  
Spring Profiles提供了一种隔离应用程序配置的方式，并让这些配置只能在特定的环境下生效。

任何@Component或@Configuration都能被@Profile标记，从而限制加载它的时机。

@ConfigurationProperties  
Spring Boot将尝试校验外部的配置，默认使用JSR-303（如果在classpath路径中）。  
你可以轻松的为你的@ConfigurationProperties类添加JSR-303javax.validation约束注

1. **答：**

a.减少开发和测试时间。

b.使用JavaConfig有助于避免使用XML。

c.避免大量的Maven导入和各种版本冲突。

d.提供意见发展方法。

e.通过提供默认值快速开始开发。

f.没有单独的Web服务器需要。这意味着你不再需要启动Tomcat，Glassfish或其他任何东西。

g.需要更少的配置 因为没有web.xml文件。只需添加用@ Configuration注释的类，然后添加用@Bean注释的方法，Spring将自动加载对象并像以前一样对其进行管理。您甚至可以将@Autowired添加到bean方法中，以使Spring自动装入需要的依赖关系中。

h．基于环境的配置 使用这些属性，您可以将您正在使用的环境传递到应用程序：-Dspring.profiles.active = {enviornment}。在加载主应用程序属性文件后，Spring将在（application{environment} .properties）中加载后续的应用程序属性文件。

**70、答：**

微服务架构是一种架构模式或者说是一种架构风格，它提倡将单一应用程序划分一组小的服务，每个服务运行在其独立的自己的进程中，服务之间相互协调、互相配合，为用户提供最总价值。

优点：

1、每个服务足够内聚，足够小，代码容易理解。

2、开发简单，开发效率提高，一个服务可能就是专一的只干一件事。

3、微服务能够被小团队开发，这个团队可以是2到5个开发人员组成。

4、微服务是松耦合的，是有功能意义的服务，无论是在开发阶段或部署阶段都是独立的。

5、微服务能使用不同的语言开发。

6、易于第三方集成，微服务允许容易且灵活的方式集成自动部署，通过持续集成集成工具，如Jenkins、Hudson等。

7、微服务易于被一个开发人员理解，修改和维护，这样小团队能够更关注自己的工作成果。无需通过合作体现价值。

8、微服务允许你融合最新技术。

9、微服务知识业务逻辑代码，不会和HTML和CSS其他界面组件混合。

10、每个微服务都有自己的存储能力，可以有自己的数据库，也可以由统一的数据库。

缺点：

1、开发人员要处理分布式系统的复杂性。

2、多服务运维难度，随着服务的增加，运维的压力也在增加。

3、系统部署依赖。

4、服务间通讯成本。

5、数据一致性。

6、系统集成测试。

7、性能监控。

**71、答：**

sleep()方法是属于Thread类中的，需捕获异常；而wait()方法则属于Object类中的，不需捕获异常。

sleep()方法导致了程序暂停执行指定的时间，让出cpu给其他线程，但是它的监控状态依然保持着，当指定的时间到了又会自动恢复运行状态。在调用sleep()方法的过程中，线程不会释放对象锁。

当调用wait()方法的时候，线程会放弃对象锁，进入等待此对象的等待锁定池，只有针对此对象调用notify()方法后，对应线程才进入对象锁定池准备。

**72、答：**

[1、TCP面向连接（如打电话要先拨号建立连接）;UDP是无连接的，即发送数据之前不需要建立连接](https://m.nowcoder.com/answer/1994242?tagId=604&pos=1&type=0&onlyWrong=false&source=home)

[2、TCP提供可靠的服务。也就是说，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达;UDP尽最大努力交付，即不保证可靠交付。TCP通过校验和，重传控制，序号标识，](https://m.nowcoder.com/answer/1994242?tagId=604&pos=1&type=0&onlyWrong=false&source=home)[滑动窗口](https://blog.csdn.net/qq_35428201/article/details/81501418" \t "https://m.nowcoder.com/_blank)、确认应答实现可靠传输。如丢包时的[重发控制](https://blog.csdn.net/qq_35428201/article/details/81501418" \t "https://m.nowcoder.com/_blank)，还可以对次序乱掉的分包进行顺序控制。

3、UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信或广播通信。

4.每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信

5、TCP对系统资源要求较多，UDP对系统资源要求较少。

6、 TCP传输单位称为TCP报文段，UDP传输单位称为用户数据报。

7、TCP注重数据安全性；UDP数据传输快，但是其安全性却一般。

**73、答：**

import java.util.concurrent.CountDownLatch;

public class SleepSort {

public static void sleepSortAndPrint(int[] nums) {

final [CountDownLatch](http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/CountDownLatch.html) doneSignal =

new [CountDownLatch](http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/CountDownLatch.html)(nums.length);

for (final int num : nums) {

new [Thread](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html)(new [Runnable](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Runnable.html)() {

public void run() {

doneSignal.countDown();  
 try {

doneSignal.await();

//using straight milliseconds produces unpredictable

//results with small numbers

//using 1000 here gives a nifty demonstration  
 [Thread](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Thread.html).sleep(num \* 1000);

[System](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/System.html).out.println(num);  
 } catch ([InterruptedException](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/InterruptedException.html) e) {

e.printStackTrace();  
 }  
 }  
 }).start();  
 }  
 }

public static void main([String](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/String.html)[] args) {

int[] nums = new int[args.length];

for (int i = 0; i < args.length; i++)

nums[i] = [Integer](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Integer.html).parseInt(args[i]);

sleepSortAndPrint(nums);

}  
}

# 发布运维

## 一、选择题

1-5：BABAA

6：A

7：B

8：ABC

9：C

10：BC

11：ABCD

## 二、填空题

1：docker rmi -f $(docker ps -aq)

2：docker build -t 镜像名字:tag dockerfile文件所在目录

3： docker tag imageName:tag 镜像名字:tag

4：-p 80001-8010:90001-9010

5： 3 、 --from-literal --from-file= configmap的yaml

## 三、简答题

1. 答案暂缺
2. 答案暂缺
3. 答案暂缺
4. 答案暂缺
5. 答：构建->自测试->转测试->CMO归档->版本发布