# 前端试题

## 判断题（每题1分，共10分）

1. CSS相对定位是脱离文档流，相对于自身的定位方式（×）
2. 相对长度单位em是相对于html根元素的font-size进行计算的（×）
3. 父元素设置了line-height: 200%，行高为固定值，其子元素字号大小不影响行高。（√）
4. 所有对象都有原型。（×）
5. JavaScript 全局执行上下文为你做了两件事：全局对象和 this 关键字。（√）
6. typeof方法可以精确的判断变量类型。（×）
7. 箭头函数体内的this对象，就是定义该函数时所在的作用域指向的对象，而不是使用时所在的作用域指向的对象。（√）
8. Vue的v-on指令可以绑定多个方法。（√）
9. Vuex中的mutation和action都可以直接变更状态。（×）
10. Vue子组件的style标签上添加scoped属性，则父组件无法修改子组件的样式。（×）

## 单选题（每题3分，共30分）

#### 正确的CSS选择器优先级关系是哪个？（B）

A: ID选择器 >内联样式 > 类/属性选择器 > 标签/伪元素/伪类选择器

B: 内联样式 > ID选择器 > 类/属性/伪类选择器 > 标签/伪元素选择器

C: ID选择器 >内联样式 > 类/属性/伪类选择器 > 标签/伪元素选择器

D: 内联样式 > ID选择器 > 类/属性选择器 > 标签/伪元素/伪类选择器

#### JavaScript 中的一切都是？（D）

A: 数字与对象

B: 函数与对象

C: 只有对象

D: 基本类型与对象

#### 事件传播的三个阶段是什么？（C）

A: 目标 > 捕获 > 冒泡

B: 冒泡 > 目标 > 捕获

C: 捕获 > 目标 > 冒泡

D: 目标 > 冒泡 > 捕获

#### sessionStorage可访问多长时间？（A）

A: 当用户关掉标签页时。

B: 当用户关掉整个浏览器，而不只是关掉标签页。

C: 永远，数据不会丢失。

D: 当用户关闭电脑时。

#### 未使用严格模式的情况下，定义函数 function getStr() { str = "some text" }并执行，关于变量str说法正确的是？（B）

A: 变量str作为函数的局部变量被创建

B: 变量str作为全局变量被创建

C: 变量str在函数执行后可被回收

D: 报变量str未定义的错误

#### 下面代码返回的是什么？（B）

const firstPromise = new Promise((res, rej) => {

setTimeout(res, 500, "one");

});

const secondPromise = new Promise((res, rej) => {

setTimeout(res, 100, "two");

});

Promise.race([firstPromise, secondPromise]).then(res => console.log(res));

A: "one"

B: "two"

C: "one" "two"

D: "two" "one"

#### ((a,b)=>a+b)(1,'2')的执行结果是什么？（B）

A: 3

B: “12”

C: NaN

D: TypeError

#### Vue中Dom渲染在哪个生命周期内完成？（D）

A: beforeCreate

B: beforeMount

C: created

D: mounted

#### Vuex中哪个属性能执行异步操作？（C）

A: getters

B: mutations

C: actions

D: modules

#### 以下哪个是Vue Router的路由独享守卫？（C）

A: beforeEach

B: beforeResolve

C: beforeEnter

D: afterEach

## 多选题（每题5分，共20分）

#### 怪异盒模型下计算元素总宽度包括以下哪些？（ABC）

A: width

B: padding

C: border

D: margin

#### JS中常见的内存泄漏包括以下哪些？（BCD）

A: 定义的全局变量

B: 闭包

C: 代码中的console.log

D: 被遗忘的计时器和回调函数

#### 下面哪些为假值？（ACE）

A: 0

B: new Number(0)

C: undefined

D: new Boolean(false)

E: Symbol('foo')===Symbol('foo')

#### 以下哪些是宏任务？（ABDE）

A: setTimeout

B: setInterval

C: Promise.then

D: setImmediate

E: requestAnimationFrame

F: MutationObserver

#### Vue 不能检测数组变化的操作有哪些？（AC）

A: vm.items[indexOfItem] = newValue

B: vm.items.splice(indexOfItem, 1, newValue)

C: vm.items.length = newLength

D: Vue.set(vm.items, indexOfItem, newValue)

## 简答题（每题4分，共20分）

#### CSS中transition和animation的区别是什么？

* transition 是过渡，是样式值的变化的过程；animation 是动画也叫关键帧，通过和 keyframe 结合可以设置中间帧的状态；
* transition 需要通过用户或者js来配合触发，animation 配合 @keyframe 可以直接播放，不需用户触发；
* animation 可以设置更多的属性，比如循环次数，动画结束的状态等等；
* animation 可以结合 keyframe 设置每一帧的状态，但是 transition 只能设置开始和结束的状态；

#### 防抖和节流有什么区别？如何实现？

* 防抖是频繁触发的情况下，只有足够的空闲时间才执行代码一次。
* 节流是指一定时间内js方法只执行一次。
* 防抖需要一个延时器来辅助实现，延迟执行需要执行的代码，如果方法多次触发，把上次记录的延迟执行代码用clearTimeout清除，重新开始计时，如果计时完毕则执行代码。
* 节流是声明一个变量当标志位，记录当前代码是否执行，如果正在执行，取消这次方法执行直接返回，如果空闲则正常触发方法执行。

#### 如何实现深度优先遍历和广度优先遍历？

* 深度优先遍历是从某个顶点出发，首先访问这个顶点，然后找出刚访问这个结点的第一个未被访问的邻结点，然后再以此邻结点为顶点，继续找它的下一个顶点进行访问。重复此步骤，直至所有结点都被访问完为止。
* 广度优先遍历是从某个顶点出发，首先访问这个顶点，然后找出刚访问这个结点所有未被访问的邻结点，访问完后再访问这些结点中第一个邻结点的所有结点，重复此方法，直到所有结点都被访问完为止。

#### 简述Vue父子组件加载渲染和子组件更新两种生命周期的执行顺序。

* 父子组件加载渲染过程：

父组件beforeCreate -> 父组件created -> 父组件beforeMount -> 子组件beforeCreate -> 子组件created -> 子组件beforeMount -> 子组件mounted -> 父组件mounted

* 子组件更新过程

父组件beforeUpdate -> 子组件beforeUpdate -> 子组件updated -> 父组件updated

#### 请写出以下代码的执行结果

async function async1() {

console.log('async1 start');

await async2();

console.log('async1 end');

}

async function async2() {

console.log('async2');

}

console.log('script start');

setTimeout(function(){

console.log('setTimeout');

}, 0)

async1();

new Promise(function(resolve) {

console.log('promise1');

resolve();

}).then(function() {

console.log('promise2');

})

console.log('script end');

以下是执行结果：

script start

async1 start

async2

promise1

script end

async1 end

promise2

setTimeout

## 编程题（每题10分，共20分）

#### 将dom节点转成json数据

页面上存在的dom节点，系统会随机在节点中生成文档片段，请按照如下需求实现 dom2json 函数

1、dom节点作为参数传入函数

2、dom2json要分析整个节点的dom结构，并将其结构转换为对应的json对象

3、需要获取dom结构的标签名称(tag)，所有属性(attributes)，子节点(children)

4、文档片段中的属性形式均为 name="value"，解析之后的格式为{name: value}, 属性值为String类型，不需要做解析

5、随机生成的文档片段中，只包含 nodeType 为1(element)和3(text)的节点，不需要考虑其他节点类型

6、纯文本也视为一个节点, json格式为 {tag: 'text', content: '文本内容'}，content为文本内容执行trim后的结果，如果该结果为空，则忽略当前节点

7、返回结果中的标签名称不区分大小写

8、如果节点不包含属性值或者子节点，其对应的结果中需要保留attributes以及children字段，例如 {tag: 'div', attributes: {}, children: []}

function dom2json(selectors) {

const jsContainer = document.querySelector(selectors)

function domJson(dom) {

var obj = {

tag: getTagName(dom)

}

if (dom.nodeType == 1) {

var attrs = getTagAttrs(dom)

if (attrs) obj.attributes = attrs;

obj.children = Array.from(dom.childNodes).filter(child => {

return !(child.nodeType == 3 && !child.textContent.trim())

}).map(child => domJson(child))

return obj

}

if (dom.nodeType == 3) {

obj.content = texthandle(dom.textContent)

return obj

}

}

function texthandle(str) {

return str.replace(/\s/g, '')

}

function getTagName(dom) {

return dom.nodeName.toLocaleLowerCase().replace('#', '')

}

function getTagAttrs(dom) {

var attr = Array.from(dom.attributes)

var obj = {}

attr.forEach(atr => obj[atr.name] = atr.value)

return attr.length ? obj : null;

}

return domJson(jsContainer)

}

#### 如何实现一个零延迟的定时器？

(function () {

var timeouts = [];

var messageName = 'zero-timeout-message';

function setZeroTimeout(fn) {

timeouts.push(fn);

window.postMessage(messageName, '\*');

}

function handleMessage(event) {

if (event.source == window && event.data == messageName) {

event.stopPropagation();

if (timeouts.length > 0) {

var fn = timeouts.shift();

fn();

}

}

}

window.addEventListener('message', handleMessage, true);

window.setZeroTimeout = setZeroTimeout;

})();