# Web前端

## HTML

### form表单

1.其实，要真正意义上禁用所有的表单元素很简单，嵌套在<fieldset>元素中，然后设置<fieldset>元素disabled就可以了，代码示意如下：

<form>

<fieldset disabled>

<legend>表单标题</legend>

<...>

</fieldset>

</form>

## Css

### :focus-visible伪类聚焦轮廓

应用的场景是：元素聚焦，同时聚焦轮廓浏览器认为应该显示

### Scroll Snap是前端必备技能

ul {

width: 375px; height: 667px;

scroll-snap-type: x mandatory;

-webkit-overflow-scrolling: touch;

white-space: nowrap;

overflow-y: hidden;

}

li {

display: inline-block;

width: 100%; height: 100%;

scroll-snap-align: center;

}

滚动父容器元素设置scroll-snap-type:x mandatory，水平滚动，强制定位，子列表元素设置scroll-snap-align:center让列表在滚动容器的中间显示，于是效果达成。

滚动终止检测可以无需判断前后滚动距离是否相等，因为无论是惯性还是Snap定位scroll事件也是持续触发的。因此，可以直接这样：

// 定时器，用来检测水平滚动是否结束

var timer = null;

// 滚动事件开始

container.addEventListener('scroll', function () {

clearTimeout(timer);

// 重新新的定时器

timer = setTimeout(function () {

// 无滚动事件触发，认为停止滚动了

// ... 做你想做的事情，如回调处理

}, 100);

});

原理如下，遍历所有的列表元素，检测列表元素的左边缘相对于滚动容器左边缘（如果是左对齐-scroll-snap-align:left）或中心（居中对齐）或右边缘（右对齐）的位置。当然，如果列表元素尺寸和滚动容器尺寸一致，则左中右边缘检测都可以。

JS示意：

[].slice.call(container.children).forEach(function (ele, index) {

if (Math.abs(ele.getBoundingClientRect().left - container.getBoundingClientRect().left) < 10) {

// 此刻的ele元素就是当前定位的元素

// ... 你可以对ele做你想做的事情

} else {

// 此刻的ele元素不是当前定位的元素

}

});

### CSS shapes

.shape {

float: left;

width: 200px; height: 300px;

/\* 文字环绕这个鹦鹉 \*/

shape-outside: url(./birds.png);

/\* 鹦鹉赋色并显示 \*/

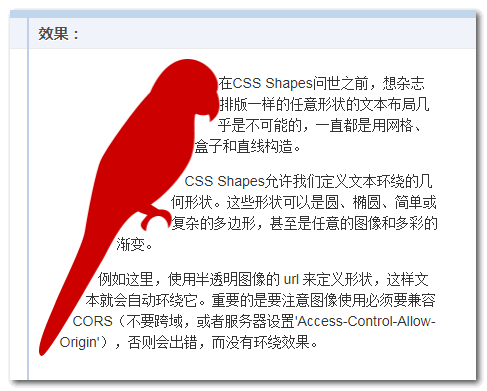
background-color: #cd0000;

-webkit-mask: url(./birds.png) no-repeat;

mask: url(./birds.png) no-repeat;

}

实现的效果如下截图所示：



### CSS滤镜和混合模式处理的图片如何上传下载？

SVG foreignObject元素与视觉存储

##### CSS代码：

.clarendon-filter {

filter: contrast(1.2) saturate(1.35);

display: inline-block;

position: relative;

}

.clarendon-filter::before {

content: '';

display: block;

height: 100%;

width: 100%;

top: 0; left: 0;

position: absolute;

background: rgba(127,187,227,.2);

mix-blend-mode: overlay;

pointer-events: none;

}

##### HTML代码：

<h4>原始效果</h4>

<p><img src="./example.jpg" width="256" height="192"></p>

<h4>CSS图像处理效果（右键另存是原图）</h4>

<div id="input" class="clarendon-filter">

<img src="./example.jpg" width="256" height="192">

</div>

<h4>后台存储的处理后的图像（右键另存是合成图）</h4>

<p><img id="output" width="256" height="192"></p>

##### JS代码：

// 输入

var eleInput = document.getElementById('input');

// 输出

var eleOutput = document.getElementById('output');

/\*\*

\* CSS渲染的图像效果转换成纯图像

\* @param {[type]} dom [description]

\* @return {[type]} 返回的是图像的base64信息

\*/

var cssRenderImage2PureImage = function (dom, callback) {

if (!dom) {

return this;

}

// 尺寸

var width = dom.offsetWidth;

var height = dom.offsetHeight;

// 复制DOM节点

var cloneDom = dom.cloneNode(true);

cloneDom.setAttribute('xmlns', 'http://www.w3.org/1999/xhtml');

// 图像元素处理

var eleImg = cloneDom.querySelector('img');

// 图像变成base64

var tempImg = new Image();

tempImg.onload = function () {

var canvas = document.createElement('canvas');

canvas.width = width;

canvas.height = height;

// 上下文

var context = canvas.getContext('2d');

// 绘制在画布上

context.drawImage(tempImg, 0, 0, width, height);

// 图像转换

eleImg.src = canvas.toDataURL();

// 进一步转换

var svgImg = new Image();

svgImg.onload = function () {

// canvas绘制合成图

context.clearRect(0, 0, width, height);

context.drawImage(svgImg, 0, 0, width, height);

// 使用jpeg格式

callback && callback(canvas.toDataURL('image/jpeg'));

};

svgImg.src = 'data:image/svg+xml;charset=utf-8,<svg xmlns="http://www.w3.org/2000/svg" width="' + width + '" height="' + height + '"><foreignObject x="0" y="0" width="100%" height="100%">'+

new XMLSerializer().serializeToString(cloneDom).replace(/#/g, '%23').replace(/\n/g, '%0A') +

document.querySelector('style').outerHTML +

'</foreignObject></svg>';

};

tempImg.src = eleImg.src;

};

// 执行

cssRenderImage2PureImage(eleInput, function (url) {

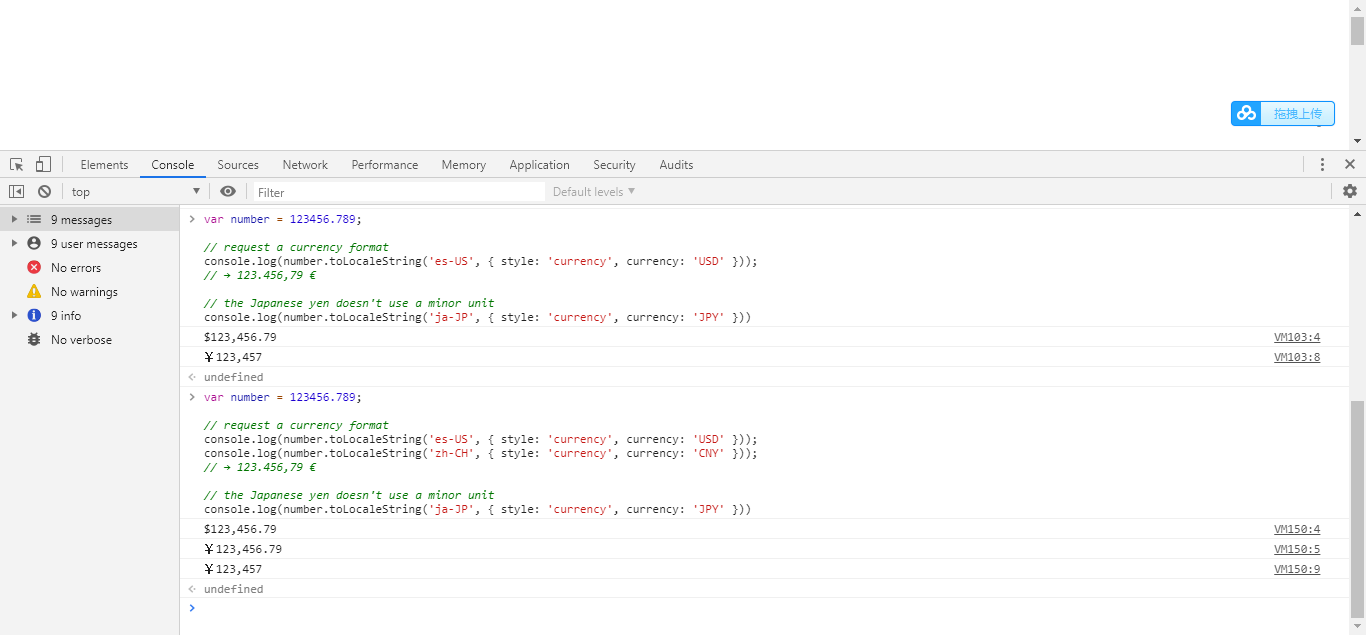
eleOutput.src = url;

});

## Js

### Number​.prototype​.toLocale​String()

可以处理金额价格相关的格式化



## React代码阅读

### 目录阅读

##### .gitignore

git忽略文件

##### .circleci

持续部署 增加开发速度

##### Fixtures

Fixtures 是测试中非常重要的一部分。他们的主要目的是建立一个固定/已知的环境状态以确保 测试可重复并且按照预期方式运行。

##### **packages**

## VUE

### Scoped CSS规范是Web组件产生不污染其他组件

使用了scoped属性之后，父组件的style样式将不会渗透到子组件中，然而**子组件的根节点元素**会**同时被设置了scoped的父css样式和设置了scoped的子css样式影响**，这么设计的**目的是父组件可以对子组件根元素进行布局**。   
　　.vue模板中的样式是根据需要按需加载，访问一个页面该组件中的样式就会追加到head标签中，如果父子组件中都对某个**子组件根节点元素**进行了控制，则父组件里的样式会被后来的覆盖。

<template>

<div id="app">

<gHeader></gHeader>

</div>

</template>

<style lang="css" scoped>

**.gHeader /deep/ .name**{ //第一种写法

color:red;

}

**.gHeader >>> .name**{ //二种写法

color:red;

}

</style>

# 后端

## Java

项目目录阅读

Gradle 发布构建工具

## MySql

varchar(20)到底可以存储多少个中文字符 10个中文字符

### Spring

# 服务器

## Ssh

### [root@106.12.26.131](mailto:root@106.12.26.131) 远程连接服务器 ssh{userName} @{Id}

## Jenkins

## 服务器常用命令

启动tomcat cd /opt/tomcat/apache-tomcat-8.5.40/apache-tomcat-8.5.40/bin

sh startup.sh

关闭tomcat cd /opt/tomcat/apache-tomcat-8.5.40/apache-tomcat-8.5.40/bin

sh shutdown.sh

发布版本sh /root/publish/jenkinsDemo.sh

jenkinsDemo网站地址 <http://106.12.26.131:8080/jenkinsDemo/index.html>

切换到jinkins工作路径 cd /root/.jenkins/workspace/jenkinsDemo