Table of Contents

jinshw.github.io

第一章 移动端适配

第一节 **HTML5** 移动端适配

- rem 适配:

  基准 `html{font-size: 10px;}`

- HTML5 机型适配

机型	宽度(可视区域)	屏幕比例	**Html font-site**	元素宽度**(px)**	元素宽度**(rem)**
iphone6 plus(基准)	414px()	1	10px	200px	10rem
iphone6	375px	375/414=0.9057	(375/414)*10=9.058 px		
iphone5	320px	320/414=0.7729	(320/414)*10=7.729 px		
iphone4	320px	320/414=0.7729	(320/414)*10=7.729 px		
ipad&Mini	768px	768/414=1.8551	(768/414)*10=18.551 px		...
荣耀6 plus、红米2、小米2	360	360/414=0.8695652173913043			
魅族4 pro	384	384/414=0.927536231884058			

1. 浏览器字体最小可以设置12px，在小于12px不起作用。
2. 计算设备大小：$(window).width()

移动端浏览器匹配机型

方案：使用js动态计算出设备的宽度，进而计算出html中的font-size大小,其他标签使用rem单位。

1. iPhone6s、iPhone6 :测试完成 (OK)
2. 视频格式mp4
3. 双栏布局：float:left;
   - 使用box-sizing:border-box; 使border包括在盒子中。
4. `<img>` 标签是否可以隐藏一部分: 使用背景图片裁切

```
<section  b-id='box14344445966'
style='width:1.55rem; height:0.45rem;top:0.6rem;left:2.25rem;
position: relative;
background-position: -2.99rem -3.96rem;
background-image: url(/images/test01.jpg);
background-repeat: no-repeat;
background-size:5rem 5rem;'  >
</section>
```

5. 适配方案：使用js动态计算出设备的宽度，进而计算出html中的font-size大小（依照iPhone6 plus为基准，414px）

```
var _cssText = document.querySelector("html").style.cssText;
document.querySelector("html").style.cssText = _cssText +"font-size:"+100*($(window).width())/414+"px !important";
```

6. 注意：测试中出现荣耀**6plus** 使用UC浏览器预览，js不能修改**html**标签的font-size 大小， 解决：设置UC浏览器字体大小为默认
7.

微信端适配

方案：使用 `@media` 适配移动端，例如下面适配iPhone6 plus，其他使用rem单位

```
@media screen and (min-width: 414px) and (max-width: 767px) {
    html{
        font-size: 100px;
    }
}
```

1. 微信中左右留白15px
2.

第二节 微信端标签替换

- 微信端固有的代码

  img {

  ```
      height: auto!important;
  ```

  }

  .rich_media_content p {

  ```
      clear: both;
      min-height: 1em;
      white-space: pre-wrap;
  ```

  }

  .rich_media_content * {

  ```
      max-width: 100%!important;
      box-sizing: border-box!important;
      -webkit-box-sizing: border-box!important;
      word-wrap: break-word!important;
  ```

  }

  body {

  ```
      -webkit-touch-callout: none;
      font-family: "Helvetica Neue",Helvetica,"Hiragino Sans GB","Microsoft   YaHei",Arial,sans-serif;
      background-color: #f3f3f3;
      line-height: inherit;
  ```

  }

  html {

  ```
      -ms-text-size-adjust: 100%;
      -webkit-text-size-adjust: 100%;
      line-height: 1.6;
  ```

  }

需要转化标签方案

1. 在内联CSS中添加 `!important`
2. 替换 `.rich_media_content p` `img` `body` `html` 样式
3. 在JS中用正则表达式替换字符串
4. `.rich_media_content p` css替换：

```
.rich_media_content p {
    clear: both;
    min-height: 1em;
    white-space: normal;// 浏览器空白忽略(待定)
}
```

5. 测试问题：

- `<video>` 视频标签在微信中，Android手机刚进入时是视频位置显示是灰色，点击一下才可以播放。iPhone6 不显示视频。
- `<iframe>` 视频标签微信中显示：Android手机可以显示播放，iPhone6P 也可以显示播放，限制: `data-src` 和 `src` 属性必须是可访问视频页面 。
- iphone5、iPhone6 优酷 `iframe` 视频不显示
- 腾讯视频获取外链

  方案一：*flash*外链 http://v.qq.com/cover/0/0n27nmk72me0ymg.html?vid=n0176p10sc0 （腾讯视频网站真正视频地址）

  http://cache.tv.qq.com/qqplayerout.swf?vid=n0176p10sc0 （？前内容 加上 vid后的内容） 移动端不支持（浏览器和微信都不支持）

  方案二：*iframe*（使用方案二） 事例：把vid=后内容换成实际视频地址内容。

```
  <iframe  " src="http://v.qq.com/iframe/player.html?vid=n0176p10sc0&amp;width=300&amp;height=200&amp;auto=0" allowfullscreen="" frameborder="0" style="width:100%;height:3rem"></iframe>
```

第三节 JS正则表达式

- 学习地址：https://msdn.microsoft.com/zh-cn/library/101eysae(v=vs.80).aspx
-

第二章 **NodeJs**学习笔记

第一节 **Express**笔记

1. Express调试debug启动： node-dev --debug ./bin/www
2. Express 启动命令：npm start
3.

第二节 **NodeJs** 笔记

第三章

第一节 **gitbook**学习笔记

- gitbook启动命令：gitbook serve -p 4000
-

第二节

面试过程中总结的知识点和题目

第一节 斐波那契数列

- 题目描述

一只青蛙一次可以跳上1级台阶，也可以跳上2级。求该青蛙跳上一个n级的台阶总共有多少种跳法

第一种方法

```
function  cal(n){
    if(n<=0){
        return -1;
    }
    if(n==1||n==2){
        return n;
    }else{
        return cal(n-1)+cal(n-2);
    }
}
```

第二种方法

```
function JumpFloor(n){
if(n<0){
    return 0
}
var fibArry = [0,1,2]
if(n<3){
  return fibArry[n]
}
var nReturn;
var fibFirst=1,fibTow=2;
for (var i = 3; i <= n; i++)
  {
      nReturn =fibFirst + fibTow;
      fibFirst=fibTow ;
      fibTow = nReturn;
  }
  return nReturn;
}
```

- 题目描述

  一只青蛙一次可以跳上1级台阶，也可以跳上2级……它也可以跳上n级。求该青蛙跳上一个n级的台阶总共有多少种跳法。

假设f(n)是n个台阶跳的次数。

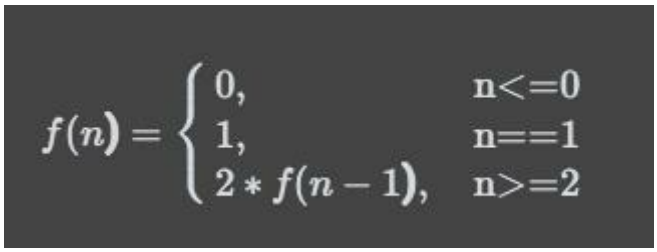
解析：

1. f(1) = 1
2. f(2) 会有两个跳得方式，一次1阶或者2阶，这回归到了问题f(1),f(2) = f(2-1) + f(2-2)
3. f(3) 会有三种跳得方式，1阶、2阶、3阶，那么就是第一次跳出1阶后面剩下：f(3-1);第一次跳出2阶，剩下f(3-2)；第一次3阶，那么剩下f(3-3).因此结论是
   f(3) = f(3-1)+f(3-2)+f(3-3)

4. f(n)时，会有n中跳的方式，1阶、2阶...n阶，得出结论：

f(n) = f(n-1)+f(n-2)+...+f(n-(n-1)) + f(n-n) => f(0) + f(1) + f(2) + f(3) + ... + f(n-1) == f(n) = 2*f(n-1)

所以，可以得出结论



java实现

```
public class Solution {
    public int JumpFloorII(int target) {
        if(target==0){
            return 0;

        }
        if(target==1){
            return 1;
        }
        return 2*JumpFloorII(target-1);
    }
}
```

javascript实现

```
function JumpFloorII(n){
    if(n==0)
        return 0;
    if(n==1)
        return 1;
    return 2*JumpFloorII(n-1);
}
```

- 题目描述

我们可以用2*1的小矩形横着或者竖着去覆盖更大的矩形。请问用n个2*1的小矩形无重叠地覆盖一个2*n的大矩形，总共有多少种方法？

解析：斐波那契数列

1. f（1）=1
2. f（2）=2
3. f(n)=f(n-1)+f(n-2)

java实现

```
public class Solution {
    public int RectCover(int target) {
        if(target==0){
            return 0;
        }
        if (target == 1) {
            return 1;
```

```
        }
        if (target == 2) {
            return 2;
        }
        return RectCover(target - 1) + RectCover(target - 2);

    }
}
```

javascript实现

第一种方式：递归

```
function RectCover(n){
    if(n==0){
        return 0;
    }
    if(n==1){
        return 1;
    }
    if(n==2){
        return 2;
    }

    return RectCover(n-1)+RectCover(n-2);

}
```

第二种方式：循环

```
function rectCover(n)
{
    if(n==0||n==1||n==2){
        return n;
    }

    var nReturn;
    var one = 1,two = 2;
    for(var i=3;i<=n;i++){
        nReturn = one + two;
        one = two;
        two = nReturn;
    }
    return nReturn;
}
```

> 关联知识：源码、补码、反码
>
> - 计算机中的符号数有三种表示方法，即原码、反码和补码。三种表示方法均有符号位和数值位两部分，符号位都是用0表示“正”，用1表示“负”，而数值位，三种表示方法各不相同
> - 特征：
>
> ```
> 1、一个负整数（或原码）与其补数（或补码）相加，和为模。
> 2、对一个整数的补码再求补码，等于该整数自身。
> 3、补码的正零与负零表示方法相同。
> ```
>
> - 求补码：**(a)** 正数的补码与源码相同 .（**b**） 负整数的补码，将其对应正数二进制表示所有位取反（包括符号位，0变1，1变0）后加1

- 题目描述

输入一个整数，输出该数二进制表示中1的个数。其中负数用补码表示。

解析：利用位运算来实现 思路： 将n与n-1做与运算，会把最右边的1去掉。 比如： 1100 & 1011 = 1000 ， 即 12 & 11 = 8

再利用计算器count++计算出有多少个 1 即可

```
function NumberOf1(n)
{
     var count = 0;
     while (n != 0) {
          count++;
          n = n & (n - 1);
     }
     return count;
}
```

> n&(n-1)妙用
>
> - 求某一个数的二进制表示中**1**的个数
>
> ```
>  while (n >0 ) {
>
>      count ++;
>
>      n &= (n-1);
>  }
> ```
>
> - 判断一个数是否是**2**的方幂：n > 0 && ((n & (n - 1)) == 0)
> - 计算**N!**的质因数**2**的个数：
>   容易得出N!质因数2的个数 = [N / 2] + [N / 4] + [N / 8] + ...
>
>   下面通过一个简单的例子来推导一下过程：N = 10101(二进制表示）
>
>   现在我们跟踪最高位的1，不考虑其他位假定为0，
>
>   则在
>
>   [N / 2] 01000
>
>   [N / 4] 00100
>
>   [N / 8] 00010
>
>   [N / 8] 00001
>
>   则所有相加等于01111 = 10000 - 1
>
>   由此推及其他位可得：(10101)!的质因数2的个数为10000 - 1 + 00100 - 1 + 00001 - 1 = 10101 - 3(二进制表示中1的个数)
>
>   推及一般N!的质因数2的个数为N -（N二进制表示中1的个数）

- 题目描述

输入一个整数数组，实现一个函数来调整该数组中数字的顺序，使得所有的奇数位于数组的前半部分，所有的偶数位于位于数组的后半部分，并保证奇数和奇数，偶数和偶数之间的相对位置不变。

解析：

> 输入：
>
> 每个输入文件包含一组测试案例。
> 对于每个测试案例，第一行输入一个n，代表该数组中数字的个数。
> 接下来的一行输入n个整数。代表数组中的n个数。
>
> 输出：
>
> 对应每个测试案例，
> 输入一行n个数字，代表调整后的数组。注意，数字和数字之间用一个空格隔开，最后一个数字后面没有空格。
>
> 样例输入：
>
> ```
> 5
> 1 2 3 4 5
> ```
>
> 样例输出：
>
> ```
> 1 3 5 2 4
> ```
>
> 对于普通的交换顺序，只要设两个指针分别从头跟尾扫描，当分别遇到不符合条件的位置时停止，然后交换位置，这样只要时间O(n)，空间O(1)，但是这样做会改变原来的顺序，要保持原来的位置，可以借用一个队列，先将一类数放到正确的位置上，再将队列里的数放到剩下的位置。时间仍为O(n)，但是要用额外空间，大小取决于输入。最坏情况下空间为O(n)。

第一种方法（复杂）

```
function reOrderArray(array)
{

    if(array == null){
        return []
    }
    var len = array.length;
    var temp=0;
    var _index = 0;
    for(var i=0;i<len;i++){
        _index = i;
        if(array[i]%2>0){
            _index = i;
            continue;
        }else{//偶数

            for(var j=i+1;j<len;j++){
                if(array[j]%2>0){
                    temp = array[_index]
                    array[_index] = array[j]
                    array[j] = temp

                }else{
                    _index = j;
                    continue;
                }
```

```
            }

        }
    }
}
```

第二种方法：

```
function reOrderArray(a){
    var arryA = []
    var arryB = []

    for(var i=0;i<a.length;i++){
        if(a[i]%2>0){
            arryA.push(a[i])
        }else{
            arryB.push(a[i])
        }
    }
    a.splice(0,a.length)
    for(var k=0;k<arryA.length;k++){
        a.push(arryA[k])
    }
    for(var k=0;k<arryB.length;k++){
        a.push(arryB[k])
    }

return a

}
```

结束