JS第一周第一天

`javascript`

1.页面的渲染方式

```
1.html parser  将html解析成html tree
2.css parser 将style sheets 解析成 style rules
3.将html tree 和 style rules 合并成attachment
4.layout 合并成 render tree
5.painting 计算样式信息  呈现页面
```

2.JS的引入方式

- 行内式

```
<div onclick=""></div>
```

- 内嵌式

```
<script>代码</script>
```

- 外链式

```
<script src="js文件地址">这里不写代码</script>
```

3.JS的引入位置

- 放在HTML后面
- 外链式写法
  如果想将JS文件引入写在HTML上面，为了保证DOM加载完成再去执行JS，可以在属性上加一个async `在IE浏览器中有些不支持async 使用defer属性`
- 内嵌式写法

```
window.onload=function () {
        //将代码放在这里 也可以实现静态资源都加载完成后再去执行JS代码
    }
```

4.JS的输出方式

`alert 弹窗`
1.只有一个按钮：确定
2.会阻止页面的加载（上线项目不要有alert）
3.输出的内容为字符串，如果写的不是字符串，会默认用一个方法tostring将其变成字符串
4.点击确定按钮 alert没有返回值（结果为undefined）

`prompt 弹窗`
1.有输入框
2.有两个按钮：确定和取消
3.有返回值（点击确定：结果为输入框内输入的东西 点击取消：结果为空null）

`confirm 弹窗`
1.没输入框
2.有两个按钮：确定和取消
3.有返回值（点击确定：结果为true 点击取消：结果为false）

`在控制台输出`
1.console:log ,table,dir,....
log

- 简单输出
- 可以输出多个，使用逗号隔开
- 最后输出的是一个值，如果你写的是一个表达式，则会将表达式的结果计算出来再打印
- 占位符 %s（必须放在第一个参数，提前占个位置，后面的内容依次往上补）

table

- 以表格的形式输出
- 数组

dir

- 详细输出

2.给一个标签添加文本

```
box.innerText="111"
```

这个方法只能当做纯文本，不能识别标签

```
box.innerHTML="111"
```

这个方法可以识别标签
所以以后常用的是innerHTML

5.JS的组成

宿主环境+ECMAscript

- 宿主环境：node/浏览器
- ECMAscript：核心语法 指定的语法规范（变量，语法，命名方式等）

浏览器环境下的JS三部分

- BOM：浏览器对象模型 提供了很多操作浏览器的方法和属性
- DOM：文档对象模型 提供了操作DOM元素（标签）的方法和属性
- ECMAscript：核心

6.变量和常量

- 变量：存储和代表可变的值
- 常量：不可变的值

`ES5中定义变量的关键字：var和function（函数）`
`ES6中新增：let和const（定义常量）和class（类）`

- var ：用来声明定义变量的关键字

```
var a=1;
```

`变量的命名规范`
1.可以使用数字，大小写字母，下划线，$，任意组合， 但是数字不可以作为开头。
2.严格区分大小写
3.不可以使用关键字和保留字作为变量名

4.尽量使用驼峰命名
5.变量的值可以是任意数据类型（松散的）

7.JS中的数据类型（两大类）

基本数据类型（值类型）
数字number，布尔boolean，null，undefined，字符串string，symbol（ES6中新增）

- 数字number类型 var a=1；
- 布尔boolean类型 只有俩值 true和false；
- null 空 只有null这个值
- undefined 未定义 通常声明一个变量没有赋值（var b；）结果为undefined
- 字符串string 使用单引号或双引号包起来的都是字符串

引用数据类型（地址型）
对象object：数组array，对象object，正则regexp，时间Date，Set和Map（ES6新增）…
函数function：function fn{ }

- 数组array 写在中括号里，用逗号隔开
- 对象object 用大括号包起来 {属性名：属性值}
- 正则regexp 写在//里面的一堆字符
- 时间Date 获取本地时间 标准格式的时间

8.检测数据类型

```
typeof+空格+检测的内容   检测数据类型
console.log(typeof a);
typeof的返回值为字符串
例外：console.log(typeof null)；返回值为object
```

不足：对于引用的数据类型，不能详细检测，检测的结果只有两种：”object”和”function”，除了函数为”function”，其他检测结果都为”object”。

9.数字number

0，正数负数，小数，NaN（不是一个数）
NaN：not a number，但是是number类型的

基本数据类型
isNaN：判断是不是一个数，返回值为true或false （true：不是一个数 false：是一个数）
isNaN（不是number类型的） 会默认调用Number（参数）将其变成number类型的，返回值为一个数字或者NaN

```
console.log(Number(null))--0
```

```
Number（undefined）--NaN
字符串
Number（""）--0
Number（"  "）--0 只有空格的字符串
Number（"1"）--1
Number（"1s"）--NaN
Number（"ss1"）--NaN
Number（"1 1"）--NaN
Number（"1  "）--1
Number（"  1"）--1
从第一个非空格到最后一个非空格，中间出现非数字就是NaN，否则就是找到的内容变成数字
```

引用数据类型

默认是将引用的数据类型调用tostring（ ）变成字符串之后，再变成数字

number（数组）–两种情况（数字或NaN）

number（对象）–NaN

number（正则）–NaN

number（时间）–毫秒数

parseInt（ ）转数字只保留整数，转布尔值与null和undefined时都为NaN

parseInt（字符串）

```
parseInt（""）--NaN
parseInt（"  "）--NaN
parseInt（"1px"）--1
parseInt（"1 1"）--1
parseInt（"px1"）--NaN
parseInt（"  1  "）--1
从第一个非空格字符开始查找，遇到非数字就停止查找，找到的内容变成数字，没有找到就是NaN
```

parseFloat（ ）保留小数 可以识别（.1）输出为0.1。但parseInt不识别，输出结果为NaN。