愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第一章 课程简介与环境安装

第1集 课程简介及学后水平分析

简介：课程简介及学后水平分析

- 针对人群
  - 小白，但是前端基础想得到拓展，有其他编程语言的基础的
- 难易程度
  - 看参照物， HTML和CSS毫不夸张的来说是最简单
  - JavaScript是入门简单，但是学好不简单
- 如何学习
  - 练习的时间 > 听课的时间
  - 编程大部分实操课
- 讲课风格
  - 注重基础
  - 由浅入深
  - 配合练习
  - 口音问题
- 学习顺序
  - 按照课程顺序来学习

    - 如果你跳了，你可能会遇到一些问题，可以在群里提问
- 学后水平
    - 掌握原生js进行开发
    - 了解面试题考察知识点
    - 会用js操作html

第2集 前端编程第一步—开发环境准备

简介：准备开发所需要的环境

- 显示扩展名(window的需要设置)
- 安装浏览器(地址变动的话百度下)
    - 谷歌 [https://www.google.cn/chrome/]
- 安装编辑器
    - vscode [https://code.visualstudio.com/]
    - 保存的快捷键是 `Ctrl` + `S`
    - 下载了中文的插件

- 
EXTENSIONS
在应用商店中搜索扩展
已启用
27
第二步
在这搜索Chinese
Auto Close Tag 0.5.6
Jun Han
Beautify css/s... 2.3.3
Beautify css, sass an...
michelemelluso
Chinese (Sim... 1.44.0
中文(简体)
Microsoft
Community M... 1.4.1
Dart
Dart language suppor...
需要重新加载
sy Sass 0.0.6
Built-in, easy to use S...
Integrates ESLint Jav...
Dirk Baeumer
Flutter 3.9.1
Flutter support and d...
需要重新加载
推荐
2
Docker 1.0.0
Makes it easy to crea...
Microsoft
Install
Cloud Code 1.2.1
Makes working with K...
Google Cloud
Install
最后
install这个插件，
之后重启vscode
第一步
点这个图标
  - 老师开了自动保存

- 建议记笔记的工具
  - typora [https://www.typora.io/]

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第二章 入门第一步之初识JavaScript

第1集 JavaScript是一门怎样的编程语言

简介：介绍JavaScript是一门怎样的编程语言

- 脚本语言
  - 必须要有个解释器
  - 脚本引擎
  - 不止JavaScript
    - JScript 微软支持IE
    - VBScript 微软
    - PHP
- 占前端的比重 (Html + Css + JavaScript)
  - 80% 占前端代码
  - JavaScript编程语言， Html和Css是标记语言
- 特点
  - 入门简单，深入理解很难
  - 应用上
  - 理解上
- 组成
  - ECMAScript
    - ECMA欧洲计算机制造联合会
    - 中立

        - 指定脚本语言的规范ECMAScript
        - 语法、变量、关键词、对象。。。
    - DOM
        - Document Object Model
        - 操作文档 (w3c规范)
    - BOM
        - Browser Object Model
        - 操作浏览器
            - 滚动条
            - 事件

第2集 JavaScript你在哪里

简介：详解为什么JavaScript要写在这个位置

- 内部引用
  - `<script>`标签内
- 外部引用
  - 通过`script`标签的`src`属性引入js文件
- 注意
  - 不要同时使用内部引用和外部引用
  - `script`标签上可以不写type，但是写了一定不要写错
    - 故意写错，存放html模板
- 推荐使用外部
  - 样式，结构和逻辑分离

第3集 JavaScript中的基本语法及术语

简介：详解JavaScript中的基本语法

```
var a, b, c, d;
3.14
a = 'nick';
a = 888;
b = 2;
c = 3;
d = b + c;

var if
```

- 语句
  - 值、运算符、表达式、关键词和注释
  - 语句会按照它们被编写的顺序逐一执行
  - 以分号结束语句不是必需的，是代码风格问题
- 字面量
  - 一般固定的值称为字面量
  - 比如：`3.14` 、`nick`
  - `{}`对象字面量
  - `[]`数组字字面量
  - `function test() {}`函数字面量
- 变量
  - 使用`var`定义变量
  - 变量声明是弱类型的，可以随时更改
  - 变量可以通过变量名访问，字面量是一个恒定的值
- 运算符
  - 赋值，算术和位运算符
  - 条件，比较及逻辑运算符
  - 在运算符旁边（= + - * /）添加空格是个好习惯：
- 关键字

    - 用于标识要执行的操作
    - 比如`var`是告诉浏览器创建一个变量
- 保留字
    - 以后可能会实现的关键字
    - 可以不记，编辑器会变色关键字

第4集 JavaScript中面向对象的概念

简介：介绍面向对象的概念

- 什么是面向对象

> 维基百科：
> 面向对象程序设计（英语：Object-oriented programming，缩写：OOP）是种具有对象概念
> 的程序编程典范，
> 同时也是一种程序开发的抽象方针。它可能包含数据、属性、代码与方法。对象则指的是类的实
> 例!!!它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性
> 和扩展性，对象里的程序可以访问及经常修改对象相关连的数据。在面向对象程序编程里，计算机
> 程序会被设计成彼此相关的对象

- 什么是面向过程
    - 我今天想吃茄子肉末
    - 买茄子，买肉， 买葱。。。
    - 洗菜，切片，剁肉
    - 炒菜
    - 吃上了
- 什么是面向对象
    - 去饭店，找个厨师
    - 点个茄子肉末
    - 完事！！！
- 在js里，万物皆对象！！！！
- JavaScript中怎么创建对象

```
var Person = new Object()
对象里面有属性，有方法
```

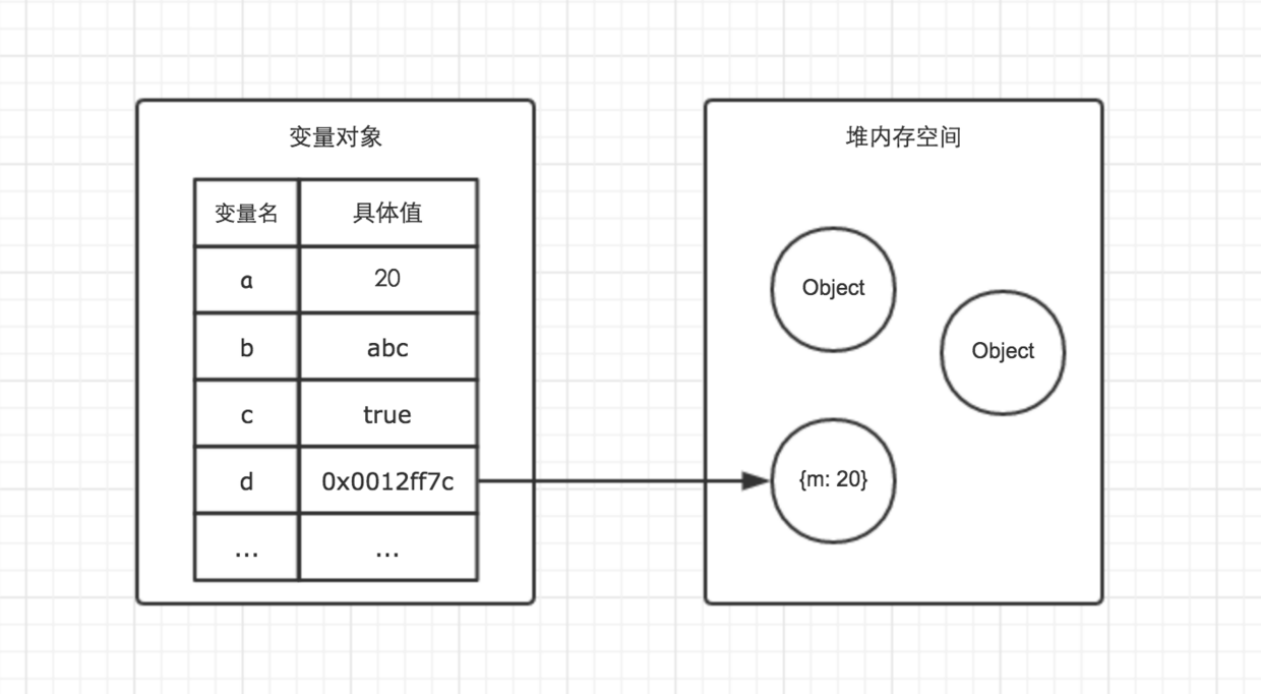
愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第三章 关于JavaScript中变量的那些事

第1集 js中的变量的类型

简介：介绍js变量的基本概念

- 栈内存
  - 原始类型保存在栈内存里
- 堆内存



- 7种原始类型(基本类型、简单类型)
  - Boolean(布尔值)
    - 计算机 非真即假
  - Null
    - 指的是你声明了一个对象未设置值
    - 可以理解尚未创建的对象
  - Undefined
    - 声明了变量，但是没有赋值

```
var a = 2;
var a
```

- Number
  - 超过一定的范围，会丢失精度

```
0.1 + 0.1 === 0.2
```

- String
  - 在js里 字符串使用单引号或者双引号围住的
- Symbol
  - es6提出的概念
- BigInt

- 引用类型(复杂类型)
  - Object
    - 万物皆对象
- 原始类型和引用类型
  - 原始类型 值是保存在栈内存，按值保存
  - 引用类型 栈内存里保存的指针， 堆内存里保存的是值

第2集 变量的声明规范

简介：详解js当中的变量声明规范

- 变量声明和变量赋值
    - 单一声明
    - 多个声明
- 声明规范
    - 虽然每个公司都有自己的规范
    - 必须是以字母、下划线 (_) 或者美元符号 ($) 开头
    - 变量是对大小写敏感的
    - 关键字和保留字不能用
- 建议
    - 定一个语义化变量名
    - 小驼峰命名变量 myHeader
    - 构造函数是用每个单词都是大写的 Person
    - 建议不要用拼音

第3集 js基础知识之变量运算

简介：讲解js当中的变量是怎么运算的

- 运算优先级高于赋值
- 同类型的运算
    - 直接值进行运算
- 不同类型的运算
    - 类型转换，值进行运算

第4集 js当中常见的运算符

简介：介绍什么是操作符

- 运算符
  - 是一类数学符号，可以根据两个值(或变量)产生结果
- 算数运算符
  - +
  - -
  - *
  - /
  - %
  - 可以和等号组合(复合赋值运算符)
    - a += 2 => a = a + 2
    - a -= 2 ... ...
- 关系运算符
  - 大小比较
  - 等值比较
- 自增自减
  - `++`
  - `--`

第5集 js当中的真真假假

简介：介绍js中什么是真什么是假

- 真
  - 满足条件就是真
  - 变量可转为布尔值true
- 假
  - 不满足条件就是假
  - 变量可转为布尔值false

- `true`和`false`就是js告诉用户的真真假假

- 真与假的运算
  - 与 `&&`
  - 或 `||`
  - 非 `!`

```
// 6个假变量
1. false (布尔型)
2. null (用于定义空的或者不存在的引用)
3. undefined (未定义值)
4. 0 (数字类型)
5. "" '' (空字符串) (字符型)
6. NaN(not a Number)
```

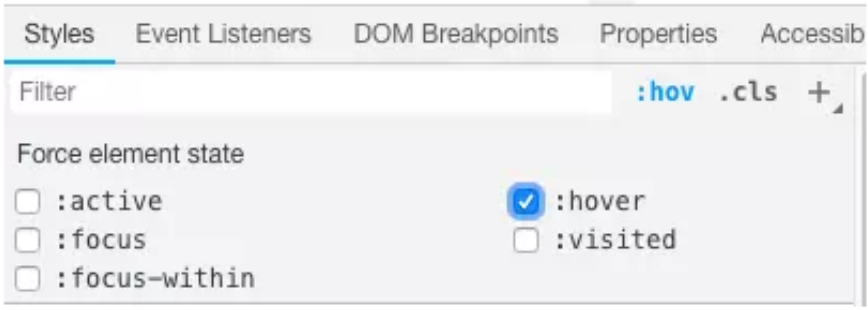
愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

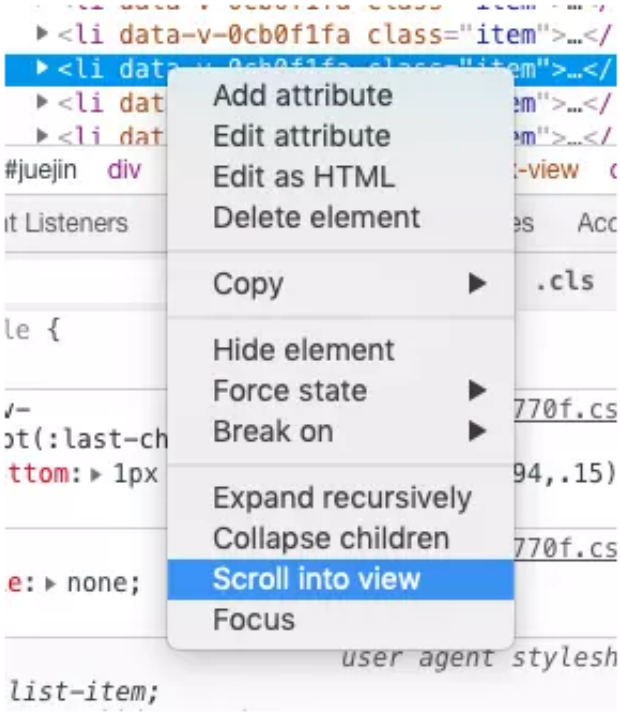
第四章 JavaScript中的流程控制语句

第1集 学js之前必须掌握的调试技巧

简介：学习js中必须掌握的调试技巧

- html 和 css 调试技巧
  - 给a元素设置了不同状态下的样式



- 快速定位元素
  - 

- 谷歌调试技巧
  - `console.log`
  - `debugger`

第2集 js基础语法之如何进行判断

简介：讲解js如何进行判断

- 语法

```
if （条件）{
   条件可以是表达式或者变量
   // 语句1
} else {
   // 语句2
}
```

- == 和 ===的区别
  - `==`会先判断数据类型一不一致，不一致的话就会转换数据类型
  - ===直接进行值判断

- 注意
  - 判断条件是否互斥
    - else if 有个好处，是满足了条件后，直接停止了
    - 保证有个else，避免出现异常

第3集 使用三元运算改造判断

简介：讲解js如何进行判断

```
0
NaN(Not a Number)
undefined
null
'', ""
false
```

- 语法

  ```
  条件 ? 表达式1 : 表达式2
  // 条件为truthy时，执行表达式1，否则执行表达式2
  ```

- 使用场景
  - 非真即假的场景
- 注意
  - 自带return

    ```
    a > 0 ? str = 'ss' : str = 'sss'
    ```

- 条件链

  ```
  str = a > 0 ? (a > 3 ? '大于3' : '小于等于3') : '小于等于0'
  ```

第4集 详解js中的switch语句

简介：介绍switch语句的使用

- 语法

```
switch(表达式) {
      case n:
          代码块
          break;
      case n:
          代码块
          break;
      default:
          默认代码块
}
```

- 和`if`的选择
  - 有3，4种以上条件的时候，就可以考虑使用`switch`
- 注意
  - 内部严格按照`===`的规则，一定要值和类型相等才行
  - 使用break语句打断程序

第5集 js基础语法之如何写一个for循环

简介：讲解如何写一个for循环

- 计算数字1 加到数字 10的和
- 语法

```
for (语句 1; 语句 2; 语句 3) {
      要重复的代码块
}
// 语句1 声明一个变量
// 语句2 指定循环跳出条件
// 语句3 控制变量的变化
```

- 流程
  - 语句 1 在循环（代码块）开始之前执行。
  - 语句 2 定义结束循环（代码块）的一个条件。
  - 语句 3 会在循环（代码块）运行完成后执行。

- 注意
  - 语句1，2，3都不是必需的
  - 语句2不写的话，必须在代码块里使用break中断循环

第6集 js基础语法之while循环

简介：讲解如何写while循环

- 语法

```
while (条件) {
    代码块
}


do {
    代码块
}

while (条件);
```

- 注意
  - 别忘了改变条件中的变量，不然就写了个死循环
  - `do while`始终比`while`多一次循环！！！

第7集 流程控制课堂作业

简介：布置课后练习

- 1. 在浏览器的控制台输出 `Welcome to 小滴课堂, 今天星期几`
  - 提示Date对象中的getDay方法
- 2. 打印1-100之间7的倍数的个数及总和

- 3. 最难一题(输出用 `document.write` 方法， 两个循环(循环套循环))

```
// 使用for循环输出一下图形 换行可以使用
document.write('<br/>')
*
**
***
****
*****
```

第8集 流程控制课堂作业讲解(上)

简介：讲解流程控制课堂作业

- 题目1： 在浏览器的控制台输出 `Welcome to 小滴课堂, 今天星期几`
  - 知识点：
    - getDay方法
    - switch或者if判断
    - 字符串拼接(可能)

- 题目2： 打印1-100之间7的倍数的个数及总和
  - 知识点：
    - 循环

第9集 流程控制课堂作业答案(下)

简介：讲解流程控制课堂作业答案

```
// 使用for循环输出一下图形 换行可以使用
document.write('<br/>')

*
**
***
****
*****
```

- 知识点
  - 嵌套循环
- 注意
  - 判断跳出循环的条件，使用`break`跳出循环

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第五章 JavaScript重点知识之函数

第1集 什么是函数及函数的组成

简介：详解函数的概念及什么是函数

- 语法

```
function name(参数 1, 参数 2, 参数 3) {
    要执行的代码
}
```

- 示例

```
function sum(a, b) {
  // return undefined
 return a * 10 + b
}
```

- 注意
  - `return`只能在函数里使用

第2集 函数中的重要概念之形参和实参

简介：详解函数中的重要概念之形参和实参

- 形参
  - 声明函数的时候定义的一个参数
- 实参
  - 调用函数的时候传的一个参数
- 形参和实参是一一对应的
  - 数量可以不对应
  - 函数可以设置默认参数
  - 定义形参的时候需要语义化
  - 实参可以是字面量也可以是变量

- 生活中的例子
  - 电饭煲
    - 煮饭
      - 加米，加水
    - 香菇焖鸭
      - 加水， 加香菇， 加肉。。。

第3集 js函数基本知识点之声明的方法

简介：介绍函数声明的方法

- 声明方式
  - 函数声明

```
function name(参数) {
  // 执行语句
}
// Date
// Array
```

  - (匿名)函数表达式
  - new Function()
- 注意
  - 声明函数过程中，函数里的语句是不会执行
  - 只有当调用函数的时候，函数里的语句才会执行

第4集 js函数基本知识点之返回值

简介：介绍函数声明及返回值的概念

- 函数的作用

    - 在恰当的时机里， 执行一段代码
    - 将值处理后，返回回来
- `return` 的作用
    - 返回值
    - 中断函数
    - 只能写在函数体里面

第5集 js函数基本知识点之变量作用域

简介：讲解变量的作用域及作用域链

- 变量
  - 全局变量
    - 挂载到`window`对象上的
  - 局部变量
    - 函数体内部声明的变量
- 作用域链
  - 内层函数是可以访问外层函数声明的一个变量

第6集 js中函数隐藏的参数

简介：详解js中函数隐藏的参数*arguments*

- *arguments*
  - 是用来取参的
  - 传入的实参都能在函数体里通过arguments类数组取到
  - 具有数组的一些特点
    - 通过索引取参
    - 有长度

第7集 通用编程思想之递归

简介：详解递归这种编程思想的应用

- 什么是递归
  - 函数自己调用自己
- 使用递归的时候
  - 要有终止递归的条件,不然就变成死循环了

第8集 js函数基本知识点之立即执行函数

简介：详解什么是立即执行函数

- 括号的作用
  - 帮助我们调用函数
- 什么是立即执行函数
  - IIFE：immediately-invoked function expression
- 特点
  - 自动执行
  - 执行完成以后销毁
- 写法

```
(function() {

})();

// w3c建议
(function() {

}());
```

- 如何证明销毁了
- 可以传参数
- 注意
  - 以`()` `[]`开头的语句，前面的语句必须加分号，否则会解析错误

```
(function test1() {
  ...
})()

var test2 = function() {
  ...
}()

function test3() {
  ...
}()
```

括号执行的是表达式

- 函数声明变成表达式的方法

```
!function() {}
```

第9集 js重点知识之函数作业

简介：课堂布置函数作业

- 题目一：编写一个函数，计算三个数字的大小，按从小到大顺序输出
  - 固定3个变量
- 题目二：输入某个数字，计算数字的阶乘
  - 通过window.prompt
  - 递归函数实现这道题
  - `6! = 1 * 2 * 3 * 4 * 5 * 6`
  - `n! = 1 * 2 * 3 * 4 * … … * n`

第10集 js复习巩固之讲解函数作业

简介：讲解作业

- 编写一个函数，计算三个数字的大小，按从大到小顺序输出
- 输入某个数字，计算数字的阶乘
    - n! = 1 * 2 * 3 * 4 * ... ... * n

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第六章 重难点知识之JavaScript中的闭包

第1集 一个计数器引发的思考

简介：讲解闭包的一个引子

```
// 设计一个函数用来计数，并且用于计数的变量不能被外部修改，只能通过调用函数修改

// 初始化计数器
var count = 0;
```

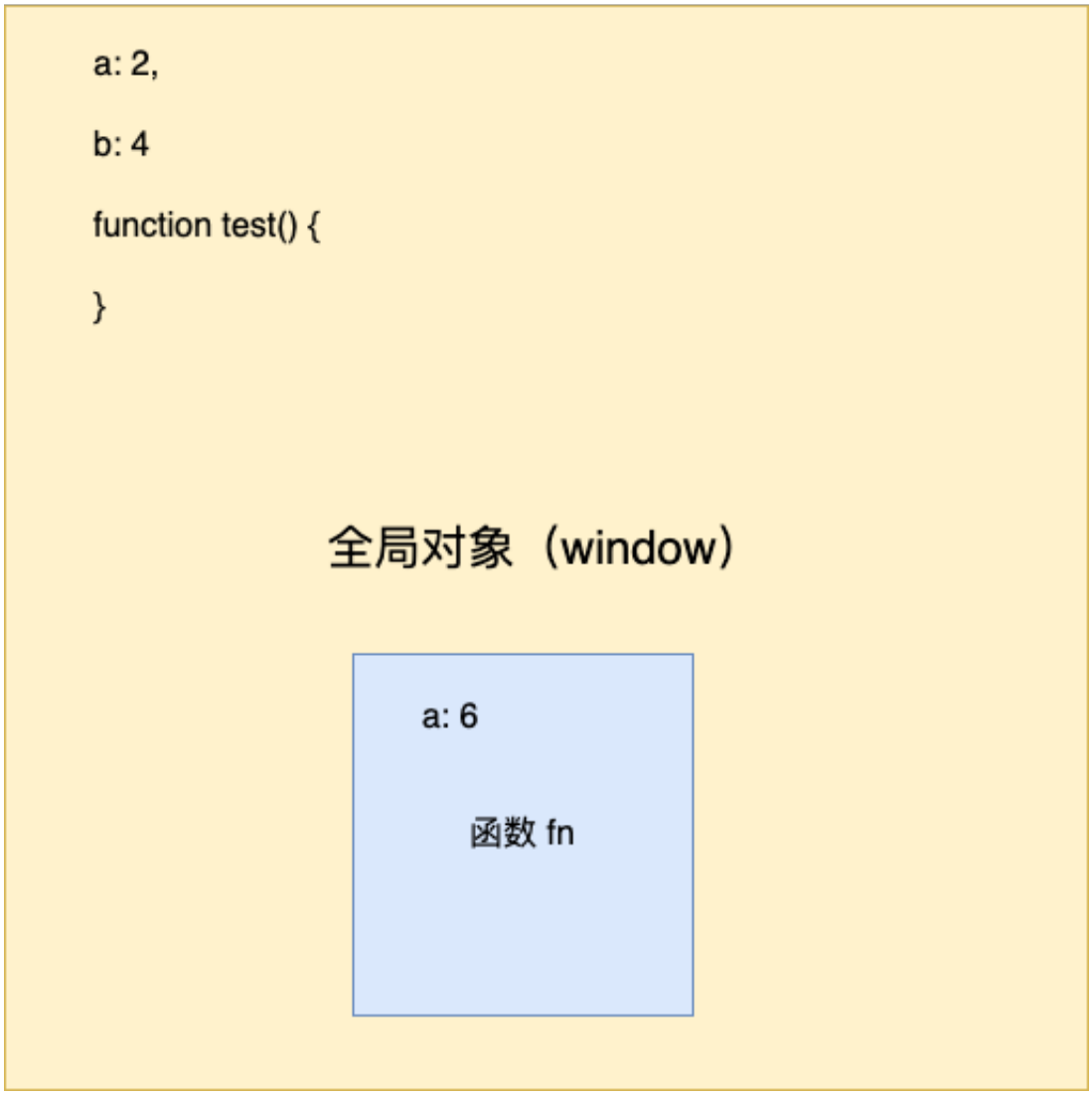
```
// 递增计数器的函数
function add() {
  count += 1;
}

// 调用三次 add()
add();
add();
add();
```

第2集 什么是函数上下文

简介：讲解函数上下文

- 上下文
  - 全局上下文
  - 函数上下文
- 函数上下文在什么时候产生
  - 函数执行的时候会形成自己的上下文(环境，对象)
- 函数上下文在什么时候销毁
  - 函数执行结束的时候就结束

图解函数上下文



第3集 什么是闭包

简介：讲解什么是闭包

- 为什么需要是闭包
  - **JavaScript 变量属于本地或全局作用域**
  - **全局变量能够通过闭包实现局部（私有）**
    - 只有调用函数才能改变变量
- 一句话总结闭包
  - **闭包是指有权访问另一个函数作用域中的变量的函数**
- 产生闭包
  - 是通过调用函数时返回其内部的函数
- 闭包的副作用
  - 产生内存泄漏
    - 比如说我本来要销毁函数的上下文，被强行保存下来了，保存在内存当中
- 通过闭包能实现什么
  - 实现外界访问函数体内部的变量

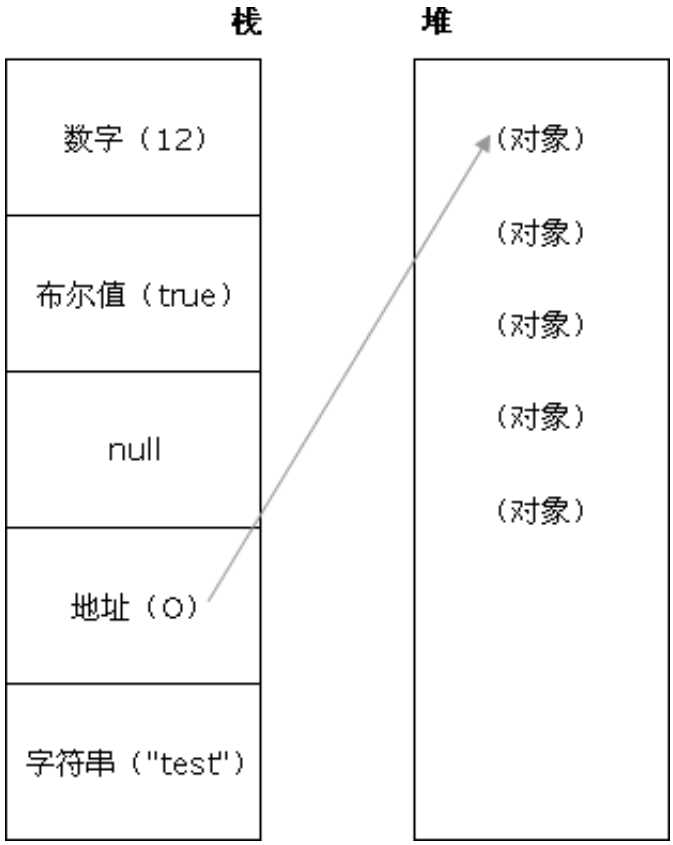
第七章 揭开JS中对象的神秘面纱

第1集 js对象的基本形式

简介：讲解对象的组成

- 除了原始值，其他都是对象
    - Undefined、Null、Boolean、Number、 Symbol、BigInt 和 String

- 可以使用`typeof`判断原始类型



- 语法

```
// 在对象里的函数叫方法(methods)


// 对象的属性名是字符串，属性的值可以是任意类型的
```

第2集 你必须知道的js对象中的常规操作

简介：讲解js对象中的一些常见操作

- 查
  - 通过`.`运算符可以访问对象属性
  - 或者`[]`操作符访问，`[]`里面可以写变量，也可以写字符串
- 增
  - 通过`.`创建属性
- 改
- 删
  - 通过`delete`运算符

```
// 键值对
// 键名就是属性名，  值就是属性值
var person = {
  name: '张三',
  age: 29,
  job: 'teacher',
  eat: function() {}
}

// console.log(person.name)

// var key = 'name'

// console.log(person['name'])

// for(var key in person) {
//   console.log(person[key])
// }

// person.sex = 1
person['sex'] = 0
person.name = 'Nick'
delete person.sex
console.log(person)

person = {}
```

第3集 对象中的方法怎么访问自己的属性

简介：讲解js对象中的方法怎么访问自己的属性

```
var obj = {
  name: 'Nick',
  sayHello: function() {
    // 方法里怎么访问自己的属性
  }
}
```

- 通过`this`我们能够访问对象自己

```
var obj2 = {
  name: 'Jack',
  sayHello: function name(params) {
    console.log('Hello, I am ' + this.name)
  }
}
```

第4集 js中创建对象的几种方式

简介：讲解js中创建对象的几种方式

- 对象字面量
  - 声明一个对象，赋值给一个变量
- 构造函数
  - Object
  - 自定义构造函数(大驼峰)
    - this指向的是对象的本身
    - 使用new 实例化一个对象，就像工厂一样

第5集 构造函数的参数应该怎么写

简介：详细介绍js当中构造函数的参数

- 固定参数

```
function Car(name, price, size) {
  this.name = name
  this.price = price
  this.size = size
}
```

  - 不知道创建对象实例的时候，传入的参数是什么，
  - 位置也要严格对应

- 不定参数

```
function Car(obj) {
  this.name = obj.name
  this.price = obj.price
  this.size = obj.size
}
```

  - 维护起来方便
  - 用户使用也方便

第6集 js底层剖析之new做了什么事

简介：讲解关键词 new做了什么事

- 没有new，直接调用构造函数
  - 构造函数内部的this指向的是window

```
function Student(obj) {
  this.name = obj.name
  this.score = obj.score
  this.grade = obj.grade
}

var stu1 = Student({
  name: 'Jack',
  score: 88,
  grade: 3
})

console.log(stu1) // 输出的是undefined，如果构造函数return this，则输出的是window对象
```

- 有new
  - 创建了一个空的对象
  - 帮助我们把对象返回回来
  - 改变了this的指向，指向了空对象

        - 小彩蛋：你可以在构造函数里输出`this`试试，然后对比有`new`没`new`

- 建议
    - 自己捋一遍思路，手动实现new关键词的作用

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第八章 JS重难点知识之原型全面剖析

第1集 原型的基本概念

简介：详解原型的基本概念知识

- `prototype`

  ```
  function Person() {}
  console.log(Person.prototype)
  ```

  是Person的一个属性，也是一个对象

- 原型的作用
  - 给构造出的对象设置公共的属性或方法
  - 建立了构造函数和实例化出来对象的联系

第2集 原型到底有什么用?

简介：讲解原型到底有什么用

- 心理准备
  - 不难，但是很杂
  - 进入笔试的时候，常考的知识点
- 原型的作用
  - 给我们构造函数实例化出来的对象设置公共的属性或者方法使用的
- 选择问题

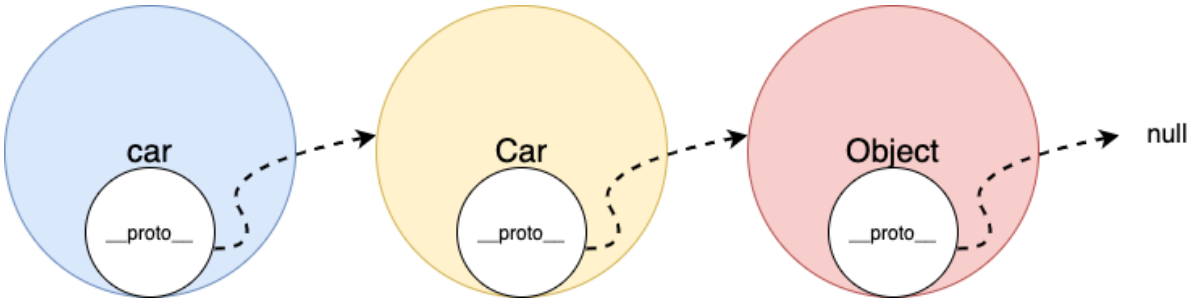
    - 方法写在原型上
    - 需要配置的属性是写在构造函数上
- 实例化对象的时候
    - 你写在构造函数里的方法和属性会重新克隆一次，会导致占用内存较高
- 实例化对象能不能对原型上的属性进行改动
    - 只有构造函数才能对原型上的属性进行改动
    - 原型说实质的话就是构造函数的属性

第3集 面试常考知识点之原型链

简介：详解js面试常考知识点之原型链

- 函数才有`prototype`属性,对象有`__proto__`属性

- 原型链是什么
  - js里万物皆对象，所以一直访问`__proto__`属性就会产生一条链条
  - 链条的尽头是null
  - 当js引擎查找对象的属性时，会先判断对象本身是否存在改属性
  - 不存在的属性就会沿着原型链网上找




```
function Car() {}

var car = new Car()
```

第4集 插件化开发初体验—写一个插件

简介：讲解插件化开发是是什么

- 需求：写一个对两个数字进行加减乘除的计算器
- 为什么要写在函数里
  - 因为函数里声明的变量或者函数，对外界无影响
  - 为了让函数在浏览器加载的时候执行，那么还要用立即函数
- 步骤
  - 写一个立即执行函数
  - 将构造函数写在立即执行函数里
  - 将公共方法写在原型上
  - 将构造函数挂载到window上

```
;(function() {
  var Computer = function(opt) {}

  Computer.prototype = {
    plus: function(num1, num2) {
      return num1 + num2
    }
  }
  window.Computer = Computer
})()

var com = new Computer()

var sum = com.plus(21, 23)

console.log(sum)
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第九章 玩转JavaScript中重要数据结构之数组

第1集 JavaScript中数组的基本操作

简介：讲解如何声明一个数组及读写操作

- 什么是数组

```
# mdn解释
## 数组是一种类列表对象，它的原型中提供了遍历和修改元素的相关操作。JavaScript 数组的长度和元素类型都是非固定的。
```

  - 把数据一股脑的放在一起就是一个数组
  - 访问数组的数据
    - 通过索引(下标)

```
console.log(list[index])
```

- 通过`length`获取数组长度

第2集 操作数组的基本方法

简介：介绍数组的基本操作—增删改查

- 数组的基本操作—增删改查
  - 改
    - 索引
  - 查
    - 索引
  - 增
    - 索引
    - push

      push这个方法是在数组后面插入一条数据
    - unshift
      - 这个方法是在数组的第一位插入一条数据
  - 删
    - pop
      - 删除了数组当中最后一个元素
    - shift
      - 删除了数组当中第一个元素

第3集 操作数组的常用方法

简介：介绍数组的高频方法

- splice
  - 用于删除或替换元素
  - 函数有返回值，返回的是被删除的元素
  - 这个方法还会改变原来的数组

```
// 第一个参数是控制从哪里开始删除或者替换(得看第三个参数有没有值)
// 第二个参数控制删除的数量
// 第三个参数将删除了的元素替换掉，可用逗号隔开
list.splice(2, 2, 'hello','Nick')
```

    - 使用场景
      - 替换数组中的元素
      - 删除数组的一部分内容
      - 清空的数组的作用
- join
  - 将数组类型的数据转换成字符串
  - 和toString的区别 可以自定义元素之间用什么隔开

```
console.log(list.join('*'))
// 1*2*hello*Nick*true*5*6
```

- concat

第4集 面试常考知识点之复杂类型怎么判断

简介：讲解如何判断js中复杂的数据类型

- 原始数据类型

  `typeof`

```
var num = 1
var isShow = true

// typeof能直接返回原始数据类型的数据类型，以字符串小写的方式
console.log(typeof(num) === 'number')
```

- 判断引用数据类型的时候，会直接返回原型链上最后的一个对象

- 引用数据类型

  `instanceof`

  ```
  // 使用instanceof 判断引用数据类型
  // 判断是不是由这个构造函数创建出来的对象实例，返回的值是布尔类型
  var list = [1,2,3,4]
  console.log(list instanceof Array)
  ```

  - 实质
    - 会找原型链上存不存在这个构造函数，会的话就返回true

第5集 js夯实基础之数组巩固及扩展练习

简介：课堂作业—数组巩固练习

- 题目一：将字符串`'nick,jack,张三,李四'`转化成数组，并且删除jack
  - 提示：`split`方法和`indexOf`方法
- 题目二：在排好序的数组里，按照大小顺序插入数据
  - `splice`

```
// 排好序的数组
var list = [23, 32, 45, 53, 62, 68]
// 假如插入的是44  则插入到32和45之间
```

第6集 js夯实基础之讲解数组扩展练习

简介：讲解课堂作业—数组巩固练习

- 题目一：将字符串`'nick,jack,张三,李四'`转化成数组，并且删除jack
  - 提示：`split`方法和`indexOf`方法
- 题目二：在排好序的数组里，按照大小顺序插入数据
  - `splice`

```
// 排好序的数组
var list = [23, 32, 45, 53, 62, 68]
// 假如插入的是44  则插入到32和45之间
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第十章 上手企业开发必须掌握的ajax

第1集 学习ajax的前置知识—JSON

简介：讲解JSON是什么、特点及其使用场景

- JSON是什么
  - JSON(JavaScript Object Notation)是一种轻量级的数据交换格式，它基于JavaScript的一个子集，易于人的编写和阅读，也易于机器解析。 JSON采用完全独立于语言的文本格式，但是也使用了类似于C语言家族的习惯（包括C, C++, C#, Java, JavaScript, Perl, Python等）。 这些特性使JSON成为理想的数据交换语言。
  - 一言以蔽之
    - JSON是用来做数据交换的一种语言
- JSON的语法格式
  - 属性名称必须是双引号括起来的字符串
  - 最后一个属性后不能有逗号
- JSON的作用
  - 用于传输数据
- 序列化和反序列化
  - 对象序列化后可以在网络上传输，或者保存到硬盘上。

第2集 前后端交互的革命技术之ajax

简介：讲解什么是ajax及其使用场景

- 什么是ajax
  - 以前前后端是后端返回整个html
    - 每次更新一些数据，他都会整个网页刷新
  - 现在
    - ajax帮助我们向服务器发异步请求
- 什么是同步，什么是异步
  - 煮着开水，在旁边盯着，等到水开了，你才做下件事

    - 煮着开水，同时你继续做了其他事
- 原理
    - 通过`XmlHttpRequest`对象向服务器发异步请求，从服务器或的数据
    - 然后通过`js`来操作`DOM`而更新页面
    - 它是在 IE5 中首先引入的，是一种支持异步请求的技术
    - 简单的说，也就是 javascript 可以及时向服务器提出请求和处理响应，而不阻塞用户，达到无刷新的效果
- 注意
    - JavaScript是单线程的，会阻塞代码运行，所以引入`XmlHttpRequest`请求处理异步数据

第3集 手把手教你动手使用ajax

简介：详解ajax的使用

- 创建ajax对象

```
if (window.XMLHttpRequest){
  xhr = new XMLHttpRequest();
} else {// code for IE6, IE5
  xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

- 设置请求地址及方式

```
// 第一个参数是用于指定请求的方式，一般用大写
// 第2个参数代表请求的URL
// 第三个参数是表示是否异步发送请求的布尔值，如果不填写，默认为true，表示异步发送
xhr.open("get","https://api.xdclass.net/pub/api/v1/web/index_card",false)
```

- 发送请求(可选参数， null)

```
xhr.send()
```

- 等到浏览器返回结果接受响应

```
/*
on readystate change事件
  readyState属性：请求状态
      0       （初始化）还没有调用open()方法
      1       （载入）已调用send()方法，正在发送请求
      2       （载入完成）send()方法完成，已收到全部响应内容
      3       （解析）正在解析响应内容
      4       （完成）响应内容解析完成，可以在客户端调用了
*/

xhr.onreadystatechange=function(){
  if (xhr.readyState==4) {
```

```
    //容错处理
    if(xhr.status==200){
      alert(xhr.responseText);
    }else{
      alert('出错了，Err:'+xhr.status);
    }
  }
}
```

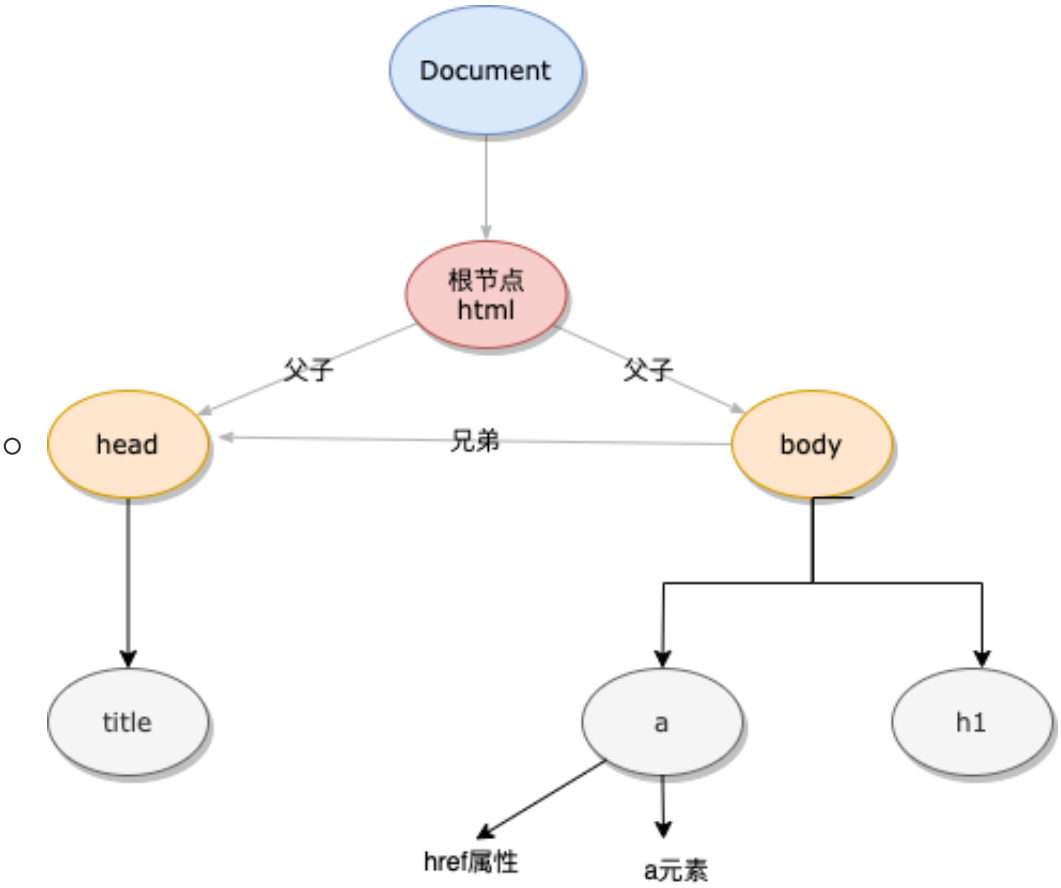
愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第十一章 JavaScript核心之DOM

第1集 DOM基础之DOM是什么

简介：详解DOM是什么

- DOM(w3c提的一个标准)
  - DOM就是文档对象模型，是一个抽象的概念
  - 定义了访问和操作HTML文档的方法
- HTML和txt文本的区别
  - HTML是有组织的结构化文件
- 什么是DOM树
  - 浏览器将结构化的文档以"树"的结构存储在浏览器内存里
  - 每个HTML元素被定义为节点
  - 这个节点有自己的属性(名称、类型、内容... ...)
  - 有自己的层级关系(parent, child, sibling)
- 图解DOM树



第2集 对DOM节点的一些常见操作

简介：详解DOM节点的一些常见操作

- 查找节点

方法	描述
document.getElementById(*id*)	通过元素 id 来查找元素
document.getElementsByTagName(*name*)	通过标签名来查找元素
document.getElementsByClassName(*name*)	通过类名来查找元素
document.querySelector(selector)	通过css选择器选择元素，无法选择伪类

- 改变元素内容

方法	描述
element.innerHTML = *new html content*	改变元素的 inner HTML
element.attribute = *new value*	改变 HTML 元素的属性值
element.setAttribute(*attribute*, *value*)	改变 HTML 元素的属性值
element.style.property = *new style*	改变 HTML 元素的样式

- 添加和删除元素

方法	描述
document.createElement(*element*)	创建 HTML 元素
document.removeChild(*element*)	删除 HTML 元素
document.appendChild(*element*)	添加 HTML 元素
document.replaceChild(*element*)	替换 HTML 元素
document.write(*text*)	可写入 HTML

第3集 DOM核心之结合事件对DOM进行操作

简介：DOM中事件的概念

- 什么是事件
  - 事件指的是在html元素上发生的事情
  - 比如图片元素被点击
  - 事件触发时，可设置触发一段js代码, 事件触发后会执行这段js代码
- 常见的HTML事件

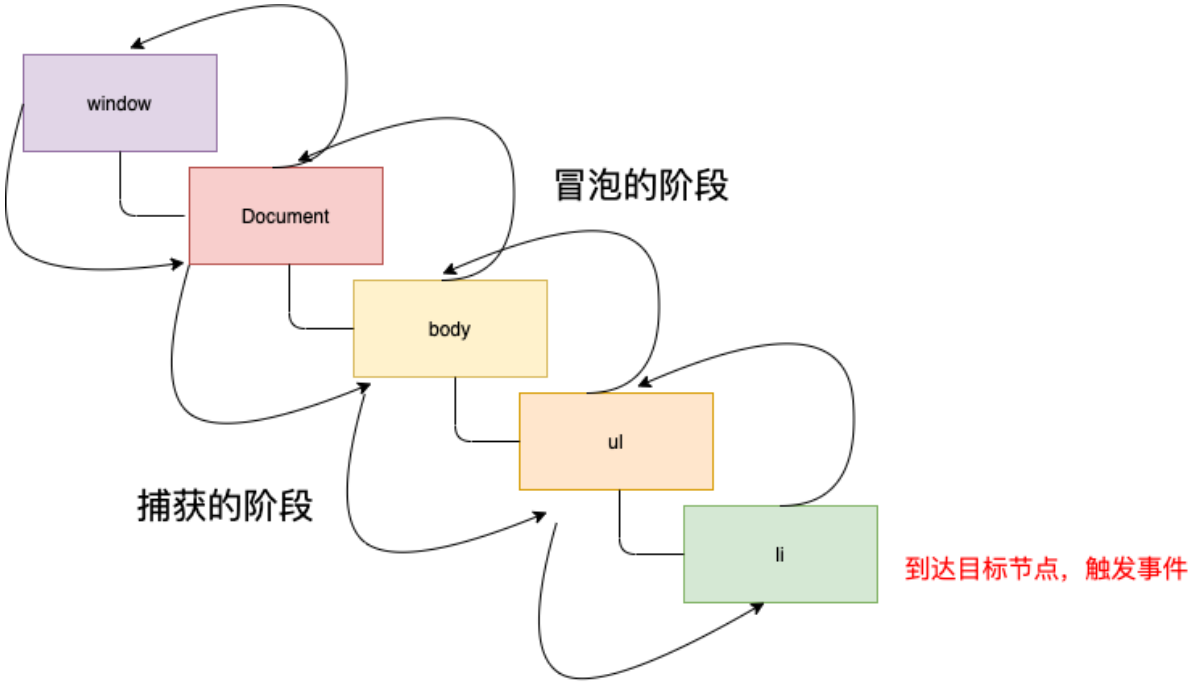
事件	描述
onchange	HTML 元素已被改变
onclick	用户点击了 HTML 元素
onmouseover	用户把鼠标移动到 HTML 元素上
onmouseout	用户把鼠标移开 HTML 元素
onkeydown	用户按下键盘按键
onload	浏览器已经完成页面加载

- 怎么对事件作出反应
  - 通过元素的事件属性
  - 启用事件监听器
    - 什么是事件监听器
      - `addEventListener`给DOM对象添加事件处理程序
      - `removeEventListener` 删除给DOM对象的事件处理程序
- onclick和addEventListener的区别
  - onclick会被覆盖
  - addEventListener可以同时注册多个，根据注册顺序，先后执行

第4集 深度剖析JS中事件的一些机制

简介：介绍JS事件中的一些机制

- 事件传播的两种机制
  - 冒泡
  - 捕获
- 图解事件捕获和事件冒泡



标准浏览器：addEventListener("click",function,"true")方法
若第三参数为true,则采用事件捕获。若为false，则采用事件冒泡。

事件捕获：先触发父元素的事件，再触发子元素的事件
事件冒泡：先触发子元素的事件，再触发父元素的事件

- 什么是事件代理
  - 思考：父级那么多子元素，怎么区分事件本应该是哪个子元素的
    - 事件代理就是利用事件冒泡，只指定一个事件处理程序，就可以管理某一类型的所有事件。

- 怎么取消冒泡或者捕获

```
event.stopPropagation();
```

  - 拓展方法

```
// 阻止元素行为, 例如a连接跳转
event.preventDefault()
```

第5集 JavaScript中的定时器

简介：讲解JavaScript中的定时器

- 延迟执行

```
setTimeout(function, 毫秒)
```

  - 停止

```
clearTimeout(id) // 参数必须是由 setTimeout() 返回的 ID 值
```

- 定时执行

```
setInterval(function, 毫秒)
```

  - 停止

```
setInterval(id) // 参数必须是由 setInterval() 返回的 ID 值
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第十二章 JavaScript核心之Bom

第1集 BOM基础之BOM的概念及内置对象

简介：讲解什么是BOM及其常用内置对象

- 什么是BOM
  - 浏览器对象模型（Browser Object Model (BOM)）
- 内置对象
  - window
  - screen
  - location

- history

第2集 BOM基础之 JavaScript 弹出框

简介：讲解JavaScript中弹出框的类型

- 警告框

```
alert('hello')
```

- 确认框

```
var isConfirm = confirm('请确认')
console.log('下一步', isConfirm)
// 有返回值的
```

- 提示框

```
var isPrompt = prompt('请输入姓名')

console.log(isPrompt) // 是null 或者 用户输入的值
```

第3集 浏览器基本概念之Cookie

简介：剖析浏览器中Cookie的概念

- 浏览器中
  - 请求是无状态
  - 比如说你进入一个网站，根据传参，网站的后台会告诉浏览器谁谁谁登录了
- 什么是Cookie
  - Cookie 是计算机上存储浏览器的数据用的
  - 容量大小大概4KB
  - 基本语法

```
document.cookie
```

- 为什么存在Cookie
  - 浏览器关闭后，服务器会忘记用户的一切
  - 让浏览器记住用户信息
- cookie操作
  - 如何创建和读取cookie

```
// 通过Document对象
document.cookie="username=Nick; expires=Thu, 18 Dec 2043 12:00:00 GMT";


setCookie('username', 'Nick')
```

  - 删除cookie

```
// 设置过期时间
document.cookie = "username=; expires=Thu, 01 Jan
1970 00:00:00 UTC; path=/;";
```

- 设置cookie函数

```
function setCookie(cname,cvalue){
    var d = new Date();
    d.setTime(d.getTime()+
(exdays*24*60*60*1000));
    var expires = "expires="+d.toGMTString();
    document.cookie = cname+"="+cvalue+";
"+expires;
}
function getCookie(cname){
    var name = cname + "=";
    var ca = document.cookie.split(';'); // 将字符
串以; 分割数组
    for(var i=0; i<ca.length; i++) {
        var c = ca[i].trim(); // 把多余空格和回车删掉
        if (c.indexOf(name)==0) {
          return
c.substring(name.length,c.length);
        }
    }
    return "";
}
```

愿景："让编程不在难学，让技术与生活更加有趣" 更多课程请访问xdclass.net

第十三章 课程总结及职业规划

第1集 课程总结及职业规划

简介：课程总结及职业规划

- 课程总结
  - ECMAScript (js语法标准)
    - 原型
      - typeof和instanceof的区别
    - 内置对象
      - String
      - Array
      - Object
      - Date
  - DOM
    - 记忆性的知识，他需要我们在实际的开发中去记住他
  - BOM
- 遇到问题
  - 建议
    - 在学习课程的同时，遇到问题时，先尝试自己的解决
    - 博文作为我们的参考，要看的是一个标准(mdn)
    - 学习必须上手，不要止于理论
- 未来规划

- JavaScript
  - Vue(1. 三选一进行深耕)
  - React
  - Angular
    - 学习框架的同时，不要丧失了原生js的能力
    - 建议大家只认真学习一门先
    - 有时间的话，可以看看源码
      - 看他的插件是怎么封装的
      - 看他插件的方法是怎么的
  - 小程序 (2)
  - 后台 (3)
    - node - JavaScript
  - App (3)
    - UNI-App
    - React Native
  - webpack (2)
    - 项目构建和优化必备的知识