类型，值和变量

- 总论
  - 数据类型：能够表示并操作的值得类型叫做数据类型
    - 原始类型：数字，字符串，布尔值，null，undefined
    - 对象类型：对象（Object）。对象是属性（property）的集合，每个属性都是由“key/value”构成。
      - 普通的对象是“命名值”的无序集合
      - 特殊的对象
        - 数组（Array）：带编号的值得有序集合
        - 函数（Function）：具有和它相关联的可执行代码的对象
          - 函数用来初始化一个新建的对象（new function()），则为构造函数
          - 每个构造函数定义了一类对象，因此类可以看做为对象类型的子类型
          - 因此数组（Array）是一种类，函数（Function）也是一种类
        - 日期（Date）：日期的对象
        - 正则（RegExp）：正则表达式的对象
        - 错误（Error）：JS程序运行时错误和语法错误的对象
        - 可以定义构造函数来定义所需要的类
    - 可变类型：值可修改
      - 数组和字符串，JS程序可以更改对象的属性值和数组元素的值
    - 不可变类型：值不可修改
      - 数组，布尔值，null，undefined
      - 字符串随刻看做是由字符组成的数组，但其是不可变的。JS未提供修改已知字符串的文本内容的方法
  - 内存管理机制：可以自动对内存进行垃圾回收，不用担心对象的销毁和内存回收。当没有引用指向一个对象时，该对象就会被自动销毁，内存资源自动被回收
  - JS中，只有null和undefined是无法拥有方法的，其他的对象，包括数字，布尔值，字符串都可以拥有方法。
  - JS可以自由地进行数据类型的转换
  - JS的变量是无类型的，其可以被赋予任何类型的值。
  - JS采用词法作用域，不在函数内声明的变量称为全局变量。在函数内声明的变量具有函数作用域，只在函数内可见
- 数字
  - 特点
    - JS不区分整数值和浮点数值，所有数值均用浮点数值表示
    - JS用64位浮点格式表示数字
    - 根据其数字格式，整数范围为-2^53~2^53，包含边界值。实际操作则是基于32位整数

        - 一个数字直接出现在JS程序中，称之为数字直接量。在数字直接量前添加（-），可以得到相应的负值，但（-）并不是数字直接量语法的组成部分
    - 整型直接量
        - JS支持十进制，十六进制的整型直接量
        - ECMAScript不支持八进制直接量，也许有些JS的实现支持八进制，但最好不要用八进制。ES6的严格模式也是禁止八进制的
    - 浮点型直接量
        - 传统写法：由整数部分，小数点，小数部分组成。例如：1.1
        - 指数计数法：在实数后跟e或E，后面再跟正负号，其后在加一个整型的指数。例如: 1.1E-2 ， 3.2e6
    - 算术运算
        - 使用运算符进行算术运算，例如： + - * / % 等
        - 通过作为Math对象的属性定义的函数和常量来实现，例如：Math.random()，Math.floot(3.2)
        - JS在溢出，下溢，被零整除不会报错。
            - 溢出以Infinity表示，负值则是 -Infinity
            - 下溢返回0，若为负值则返回“负零”。其几乎和正常的零完全一样，一般很少用到
            - 被零整除返回Infinity或 -Infinity，但零除零则返回NaN
            - 无穷大除无穷大，开方负数等操作均返回NaN
            - NaN和任何值都不相等，包括其本身。NaN == NaN // => false
            - 用isNaN()和isFinite()来判断是否为NaN
              isNaN()：参数若为NaN或一个非数字值（如字符串和对象）则返回true
              isFInite()；参数若不是NaN、Infinity或 -Infinity时返回true
            - 负零值和正零值是相等的（用严格测试“===”也如此），只在他们作为除数之外不相等
    - 二进制浮点数和四舍五入错误
        - JS采用IEEE-754浮点数表示法，是一种二进制表示法。可以精确表示1/2、1/4这些，但不能精确表示1/10、1/100这些。
        - 0.3 - 0.2 == 0.1 // => false ，实际上0.3 - 0.2 = 0.099 999 999 999 999 98
    - 日期和时间
        - Date()构造函数，用来创建日期和时间的对象。它不像数字那样是基本的数据类型
- 文本
    - 特点
        - 由16位值组成的不可变的有序序列
        - 常见的Unicode字符是通过16位内码表示，代表单个字符。有些不能表示为16位的Unicode字符用两个16位值组成的序列表示。所以长度为2的字符创可能指标是一个Unicode字符
        - JS不会对字符串做标准化的加工，不能保证字符串是合法的UTF-16格式
    - 字符串直接量
        - 单引号可以包含双引号，双引号可以包含单引号
        - 可以拆分为数行，每行以（\）结束，若希望在字符串知己诶昂中另起一行，可以使用转义字符\n

- 转义字符
  - （\）后加一个字符，就不再表示他们的字面含义了。
  - \o------->NULz字符（\u0000）
  - \b------->退格符（\u0008）
  - \t------->水平制表符（\u0009）
  - \n------->换行符（\u000A）
  - \v------->垂直制表符（\u000B）
  - \f------->换页符（\u000C）
  - \r------->回车符（\u000D）
  - \"------->双引号（\u0022）
  - \'------->撇号或单引号（\u0027）
  - \\------->反斜线（\u005C）
  - \xXX------->由两位十六进制数XX指定的Lation-1字符
  - \uXXXX------->由4位十六进制数XXXX指定的Unicode字符
- 字符串的使用
  - （+）运用于数字，表示两数相加。作用于字符串，表示字符串链接
  - length属性可得到字符串的长度，即包含的16位值得个数
  - 字符串是不变的，例如replace()和toUpperCase的方法都返回新字符串，而不是改变字符串本身
  - 字符串可以作为只读数组
- 模式匹配
  - RegExp()构造函数，创建“正则表达式”对象
  - RegExp不是JS的基本类型
  - RegExp虽不是基本数据类型，但有直接量写法/HTML/或/HTML/i
  - 字符串有可以接受RegExp参数的方法

- 布尔值
  - 这个类型只有两个值，保留字true和false
  - 任意JS的值都能转换为布尔值。undefined，null，0，-0，NaN，空字符串都会被转换成false
  - 所有其他的值，包括所有对象（数组）都会转换成true
  - 布尔值有toString()方法
  - “&&”逻辑与，当且仅当两个操作数都是真值时才返回true，否则返回false
  - “||”逻辑或，如果两个操作数其中之一为真就返回true，两个都为假则返回false
  - “!”一元操作符执行布尔非，若操作数为真，则返回false，若为假，则返回true
- null和undefined
  - null描述“空值”，null为一个特殊的对象值，typeof null // => "object"。其含义为“非对象”
  - undefined表示“未定义”，表示一种更深层次的“空值”。typeof undefined // => "undefined"
  - 在ES3中undefined可读可写，在ES5中只可读
  - null和undefined都是假值，都没有任何属性和方法
  - undefined表示系统级的，出乎意料的或类似错误的值得空缺，null表示程序级的，正常的或意料之中的值的空缺

    - 若想将他们赋值给变量或属性，或作为参数传入函数，最好使用null
- 全局对象
    - 当JS解释器启动时，或加载新页面时，将创建一个新的全局对象，并给一组定义的初始属性
    - 全局属性，比如undefined、Infinity和NaN
    - 全局函数，比如isNaN()、parseInt()、eval()
    - 构造函数，比如Date()、RegExp()、String()、Object()和Array()
    - 全局对象，比如Math和JSON
    - 在客户端JS中，Window对象充当了全局对象。
- 包装对象
    - 再次强调，对象是属性的集合，当属性为一个函数时，称其为方法
    - 对于字符串，数字，布尔值，调用其属性可以在表面上看做是：创建一个临时对象，调用其属性，再销毁该对象。注意，从实现上并非如此，只可看做是如此。
    - 该临时对象称为包装对象，包装对象被看做一种实现细节，不用特别关注。
    - 原始值和对象是不同的
    - 包装对象和原始值的关系和区别：
        - 可以通过String()，Number()，Boolean()显示的创建包装对象
        - JS在必要时会将包装对象转为
        - 原始值不能设置新属性，也不能改变原属性的值，只能读取
        - 包装对象可以设置新属性，不能改变原属性的值
        - 原始值和包装对象“==”视为相等，“===”视为不等
        - 原始值和包装对象的typeof不同，一个为“string”，“number”，“boolean”包装对象为“object”
- 不可变的原始值和可变的对象引用
    -