JavaScript基础（一）

1. 关于类型

- 9种类型
  - 值类型（基本类型）
    - Undefined
    - Null
    - Boolean
    - String
    - Number
    - Symbol — ES6 新加入
  - 引用类型（对象类型）
    - Object
    - Function
    - Array
- 3种扩展类型
  - 正则：RegExp
  - 日期：Date
  - 错误：Error

2. 关于对象（Object）

- 实现
  - JS是面向对象的语言
  - JS与其他面向对象的语言有差异
    1. 允许动态为对象添加、删除属性
    2. 除了基本类型，函数也称为对象的一个属性
  - 对象在不同语言中的实现不同
    - CPP/Java/Python中叫做类：Class（热门，好理解）
    - JS中叫做原型：Prototype（冷门，不易理解）
- 本质
  - 对象具有唯一性 — 没有完全相同的两片叶子（内存地址）
  - 对象具有状态 — 叶子有翠绿和枯黄的不同状态（属性）
  - 对象具有行为 — 叶子可以长在枝干上，也可能会飘落（方法）
  - JS中统称为属性
- JS的特色（不同于其他面向对象语言） — 对象非常灵活，可以在运行时添加、删除属性
- 对象的两类属性
  - **#理解：** 对JavaScript 来说，属性并非只是简单的名称和值，JavaScript 用一组特征（attribute）来描述属性（property）
  - 数据类属性
    - 四个特征
      - value：原值
      - writable：可否被修改
      - enumerable：决定 for in 能否枚举该属性
      - configurable：决定该属性能否被删除或者改变特征值
    - **#说明：** 通常用于定义属性的代码会产生数据属性，其中的 writable、enumerable、configurable 都默认为 true；可使用内置函数 Object.getOwnPropertyDescripter 来查看
  - 访问类属性
    - 四个特征
      - getter：函数或 undefined，在取属性值时被调用
      - setter：函数或 undefined，在设置属性值时被调用
      - enumerable
      - configurable
    - **#理解：** 可以视为一种函数的语法糖
- ES6起，JS提供了class关键字，可以更符合“习惯”进行面向对象编程

3. 面向原型的OOP编程

- **#定义：** 原型（prototype）是对象的一个内部属性，它是一个指针，指向另一个对象。这个指针允许一个对象访问另一个对象的属性和方法，从而实现继承。原型系统多与高动态性语言配合
- **#原理&特征**
  1. 所有对象都有一个私有字段 [[prototype]]，即是对象（Object）的原型
  2. 读一个属性，如果对象本身没有，则会继续访问对象的原型，直到原型为空或者找到为止
  3. 是实现抽象、继承和复用（面向对象）的核心机制
- 与面向class编程的差异
  - **面向类：** 先有类，再从类去实例化一个对象。类与类之间又可能会形成继承、组合等关系 ——ES6 开始，JS有class关键字
  - **面向原型：** 提倡开发者去关注一系列对象实例的行为，然后再关心如何将对象归类到具体的原型
- **#Tips：** JavaScript 并非第一个使用原型的语言，在它之前，self、kevo 等语言已经开始使用原型来描述对象了
- **#意义：** 是 JavaScript 实现面向对象的一种方式，也是JavaScript的核心工作机制
- **#发展：** 从 ES6 以来，JavaScript 提供了一系列内置函数，以便更为直接地访问操纵原型
  - Object.create 根据指定的原型创建新对象，原型可以是 null
  - Object.getPrototypeOf 获得一个对象的原型
  - Object.setPrototypeOf 设置一个对象的原型

4. 面向class的OOP编程

- **#定义：** 自ES6开始，Nodejs V8.5.开始，JS引入了class关键字，从此开始了类似CPP/JAVA那样的更容易理解的OOP编程
- **#特征**
  - JS的class支持类（静态）属性/方法、实例属性/方法，且互相隔离
    - 类属性可以在定义类时用static关键字定义
    - 类属性/方法还可以在类定义后添加；classX.attr=...
  - 支持构造器函数
  - 支持getter、setter方法
  - 支持私有属性：#field，不可在类定义体外访问
  - 支持继承、组合：Extends、super
- **#原理：** 仍然是基于原型链的运行时模型