Chapter 3 && Chapter 4: Objects and Functions

本笔记主要来自于《Javascript Good Parts》第三章和第四章

1. Objects

    1. 在Javascript中，基本类型包含number，string，boolean，null和undefined。这五种类型都是immutable，其余的数据类型都是objetcs，这包含了数组array，函数function和正则表达式regular expression等等。
    2. 如何定义一个object：
    在Javascript中，定义一个object不需要像其他语言那样专门定义一个类，可以直接用一对大括号`{}`来表示一个object，这个object可以是一个空object，里面也可以进行一些数据的初始化；看下面例子：

    ```
    var empty_object = {};
    var storage = {
        "first_name":"Yi",
        "Last_name":"Qiu"
    };
    ```

    当然除了上面的例子，一个object里面还可以存放其他的object，因此链表和树的表达方式就比较简单了。
    3. object中数据的获取：
    要获取object中的数据，有两种方式，以上面的`storage`为例：
        - `storage["first_name"];`
        - `storage.first_name;`

    4.
    ```
    var middle = storage.middle || "none";
    var status = storage.status || "unknown";
    ```

    如果想对object中没有的数据进行获取以及初始化，可以使用逻辑符号`||`：

    ```
    storage.person && storage.person.name
    ```

    如果object中没有该数据，那么强行获取该数据的结果就是`undefined`,对`undefined`进行操作会跑出TypeError异常，为了避免这种情况，可以进行预先检查，像下面这样：

    5. object中数据的更新：
    对object中的元素进行更新很简单，直接进行重新赋值就好了；如果object中原本没有改元素，那么更新将会往object中添加改元素；
    6. object reference：
    Javascript中的object reference和其他语言并没有什么太大区别，可以按照其他语言那样进行理解；
    7. object prototype：
    Javascript中的原型(prototype)机制是很难理解的一部分。在之后的笔记中会有专门来介绍Javascript原型机制的，在这里只需知道Javascript中所有的object都是来自于一个最基本的Object.prototype。Javascript中的原型机制类似于其他语言中的继承，upper prototype property对下级object可见，而且原型机制是动态的，一旦某个upper prototype更新（包括添加和更新数据），其所有下级object立马可见；
    8. object reflection：
    这里主要介绍两个操作方法：
        - `typeof`:这个操作符可以判定object中的数据类型；
        - `hasOwnProperty`：这个方法可以判定object是否含有改元素；

    9. object enumeration：
    object中property的遍历：

        - `for in`操作，这种操作类似于java语言中的遍历map，遍历变量是key。值得注意的是，这种遍历方法有两个特性：
            - 这种遍历方法会遍历整个prototype chain，也就是说位于prototype chain上所有的property都会遍历到；
            - 这种遍历方法和object存放元素的顺序不存在必然关系，因此输出的顺序不一定是你想要的顺序；

        - `for`遍历：这种遍历方法我们可以自己定义一个array，这个array里面含有所有的我们需要的元素的key，这种方法我们可以将遍历的范围限定在我们需要的元素，同时也保证了输出元素的顺序
        以下是两种遍历方式的代码：

        ```
        // for in
        var name;
        for (name in storage) {
            document.writeln(name + ": " + storage[name]);
        }
        // for
        var i;
        var properties = [
            "first_name",
            "last_name",
            "middle_name"
        ];
        for (i = 0; i < properties.length; i++) {
            document.writeln(properties[i] + ": " + storage
        [properties[i]]);
        }
        ```

    10. object delete：
    `delete`关键字可以用于删除object中的特定元素，**但是delete不会删除整个原型链中的数据，只会删除当前object中的元素**；

    ```
    delete storage.first_name;
    ```

2. Functions