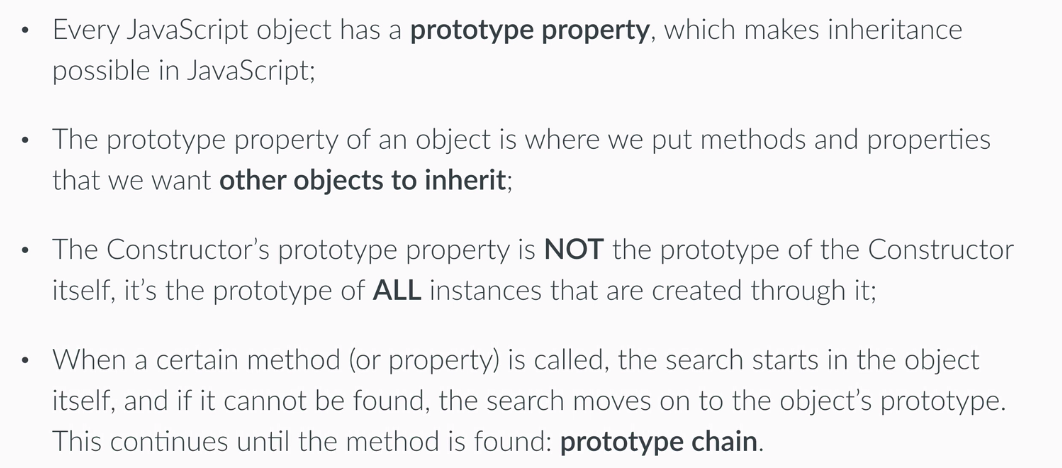
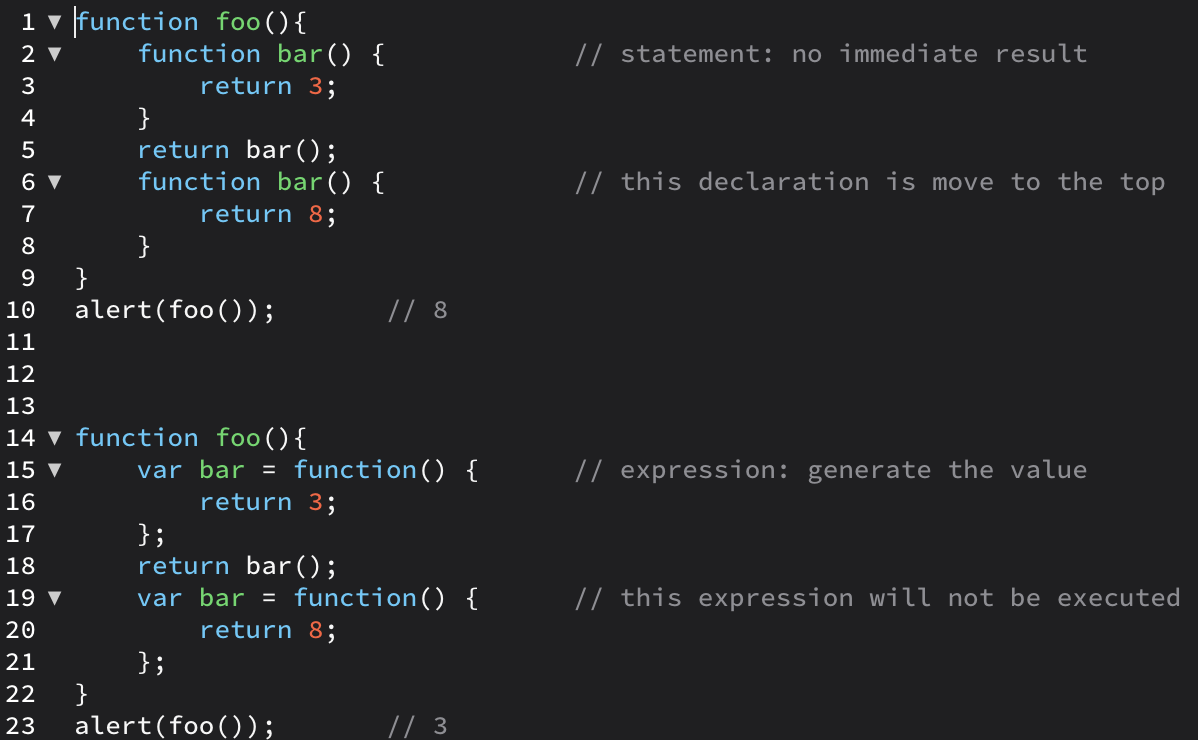
JavaScript

* almost everything in it is an Obj
* OO programming: use a lot of Objs, properties, methods
* constructor/prototype in JS is like Class in other programming language
* Prototype-based language: inheritance works by using prototype

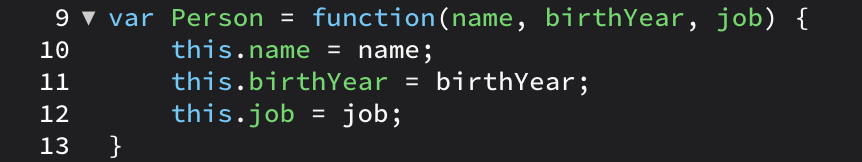


Statement VS expression



JS code fragments:

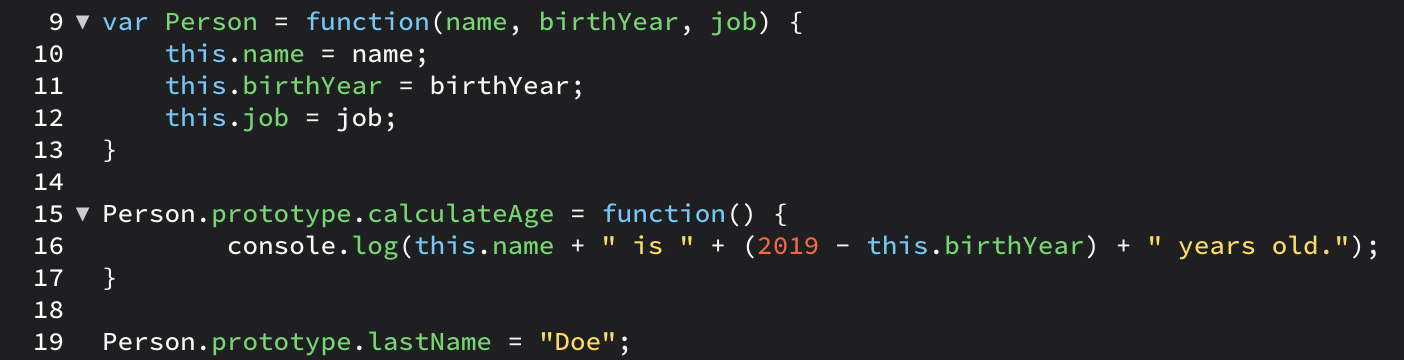
* constructor function (**Class**):
  + name it with **CAPITAL** letter
  + Person eg:



* instance of constructor (**Object**):
  + Julian eg: // instance of Person **object/constructor/class**

../../Screen%20Shot%202019-08-19%20at%203.05.43%20PM.png

* process:
  + **new:** create empty *Person* object
  + constructor function(*Person*) is called
  + **this:** points to the empty *Person* object // this: defaultly point to global object
  + assign those **values** to *julian* variable
* inheritance
  + take calculateAge function out from Person constructor, and add it to prototype
  + same for the lastName property



* + julian.hasOwnProperty(name) // true
  + julian.hasOwnProperty(lastName) // false 🡪 this is inheritant property
  + julian instanceOf Person // true

**Closure**:

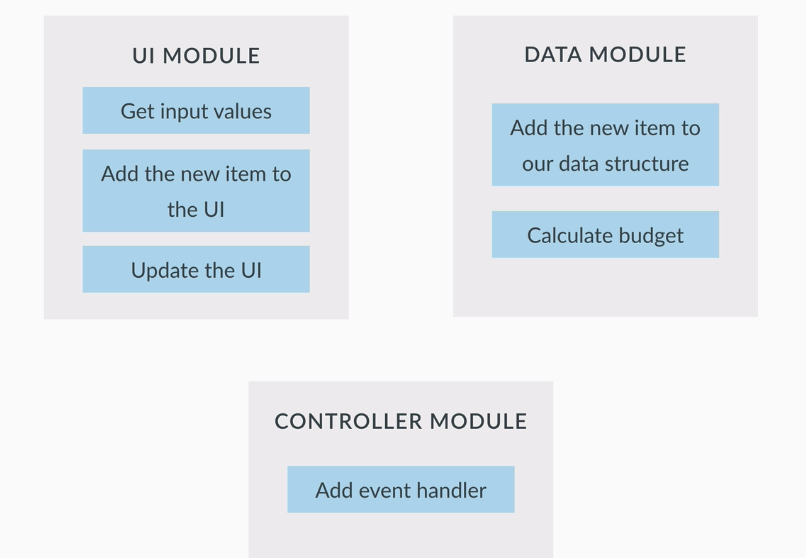


* **Chain Scope**（链式作用域）：f2 被包涵在f1内部；f1内局部变量对于f2可见，反之不ok。子对象会向上一层一层寻找父对象变量
* **Closure**（闭包）：
  + 如果在一个内部函数里，对在外部作用域（但不是在全局作用域）的变量进行引用，那么内部函数就被认为是闭包(closure)。
  + 能够读取其他函数里局部变量的函数
  + 定义在函数内部的函数；用来链接函数内外
* Closure注意：
  + 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题，在IE中可能导致内存泄露。解决方法是，在退出函数之前，将不使用的局部变量全部删除。
  + 闭包会在父函数外部，改变父函数内部变量的值。所以，如果你把父函数当作对象（object）使用，把闭包当作它的公用方法（Public Method），把内部变量当作它的私有属性（private value），这时一定要小心，不要随便改变父函数内部变量的值。

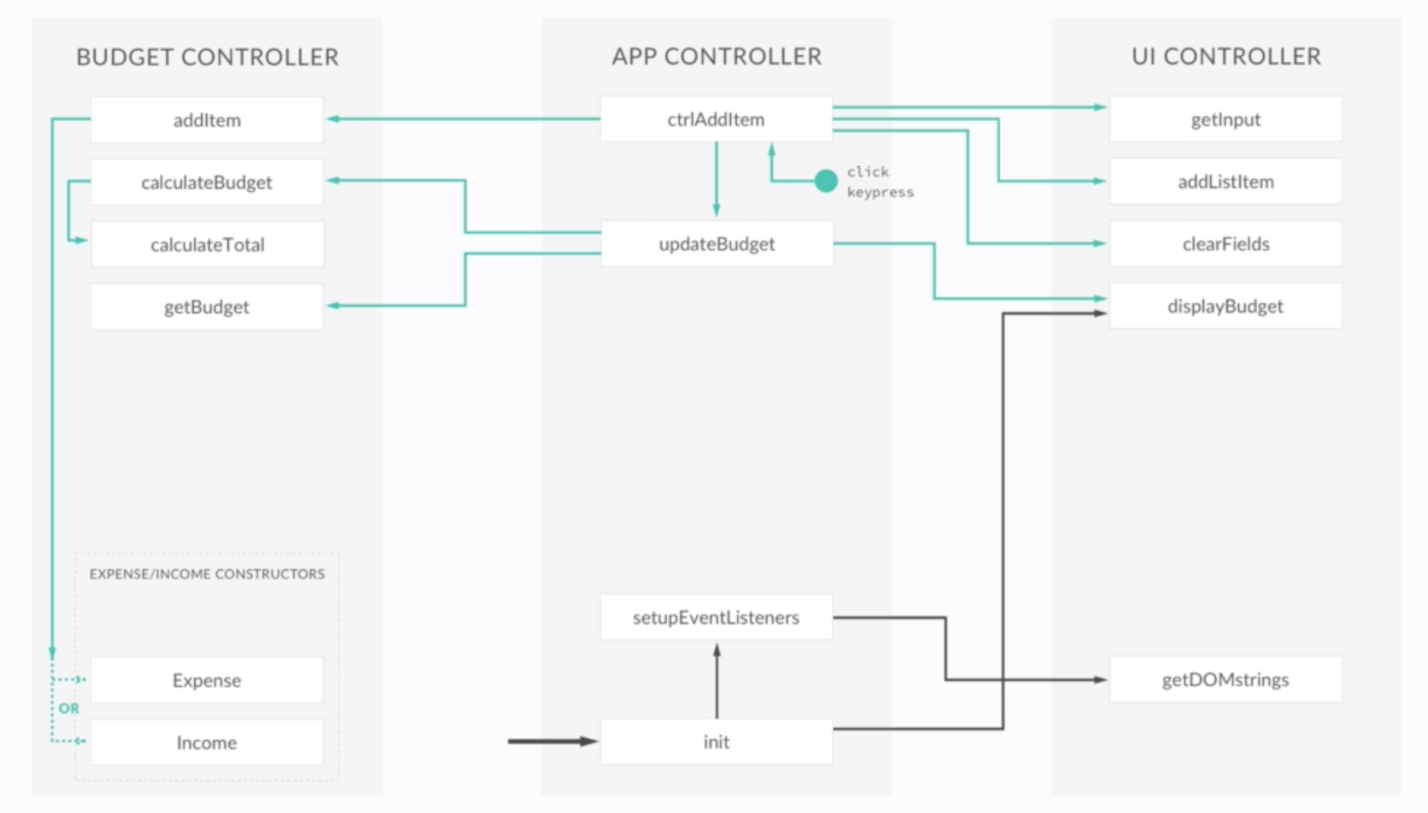
**this: is depending on which object call the function!!!!!!!!!**

**Building a project:**

1. to-do list / user stories
2. structure code:
   1. **modules: UI module + Data module + Controller module**

****

1. module pattern



1. event bubbling; target element; event delegation

