HTML5 and JavaScript Games Programming Week 3

Mario.Soflano@uws.ac.uk

Object-Oriented Programming

What is Object-Oriented programming?

https://www.youtube.com/watch?v=NUl8lcbeN2Y

Why use Object-Oriented programming?

https://www.youtube.com/watch?v=QzX7REqciPY

Object-Oriented Programming

No Class – only deals with object

JavaScript Object encapsulates the properties and methods

Inheritance and Polymorphism are normally done through the use of prototypes.

Object Creation methods:

- Object Literal
  - Singleton pattern
- Constructor Function

Object-Oriented JavaScript – Object Literal

```
var course= {
      courseSeries: “CGD”;
      courseID:”COMP0809xx”,
      courseName:”Javascript Frameworks”,
      courseDetails: {
                        duration: 12,
                        theDay: “Monday”,
                        theLectureTime: “9am-10am”,
                        theLabTime: “10am-1pm”
                  },
      courseBasicInfo: function(passedVar) {
            return this. courseID + “ ” + this. courseName +
            “ ” + passedVar;
      }
};
```

```
course.courseID;
      // COMP0809xx
course[“courseName”];
            // courseName
course.courseDetails.duration
      // 12
course.courseDetails[“theDay”]
      // Monday
course[“courseDetails”][“theLectureTime”]
      // 9am-10am
course[“courseDetails”].theLabTime
      // 10am-1pm
course. courseBasicInfo(“Sample”);
      // COMP0809xx Javascript Frameworks
Sample
```

Object-Oriented JavaScript – Constructor Function

```
function course(courseID_pass, courseName_pass, duration_pass, theDay_pass, theLectureTime_pass,
    theLabTime_pass) {
        this.courseSeries: “CGD”;
        this.courseID = courseID_pass;
        this.courseName  = courseName_pass;
        this.courseDetails.duration = duration_pass;
        this.courseDetails.theDay = theDay_pass;
        this.courseDetails.theLectureTime= theLectureTime_pass;
        this.courseDetails.theLabTime = theLabTime_pass;
        this.courseBasicInfo: function(passedVar) {
          return this. courseID + “ ” + this. courseName + “ ” + passedVar;
        }
}
```

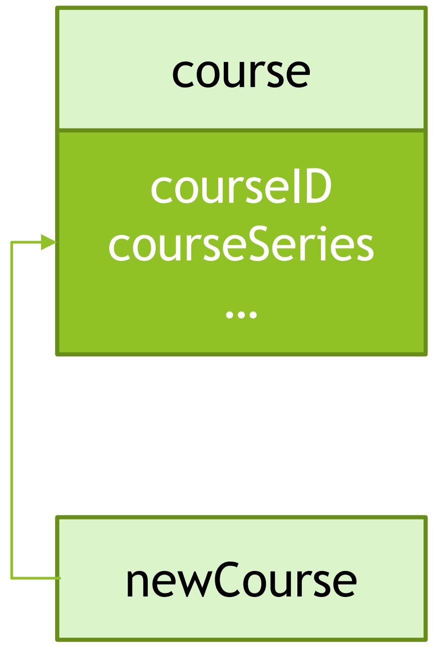
Object-Oriented JavaScript – Constructor Function

```
var course1 = new course (“COMP0809xx ", "JavaScript Frameworks", 12, "Monday", " 9am-10am ", "10am-1pm");
var course2 = new course (“COMP0810xx ", “HTML5", 6, “Tuesday", “10am-11am", "1pm-3pm");


course1.courseName;                          // JavaScript Frameworks
course2.courseName;                          // HTML5
course1.courseDetails.theLectureTime;        // 9am-10am
course2["courseDetails"]["theLectureTime"];  // 10am-11am
course1.courseBasicInfo(“Sample”);           // COMP0809xx JavaScript Frameworks Sample
course2.courseBasicInfo(“Sample”);           // COMP0810xx HTML5 Sample
```

Object-Oriented JavaScript – Object Literal vs Constructor Function

Object Literal – One change will affects all

```
var newCourse = course;
course.courseSeries;          // CGD
newCourse.courseSeries;       // CGD

newCourse.courseSeries = "Not CGD";
course.courseSeries;          // Not CGD
newCourse.courseSeries;       // Not CGD
```

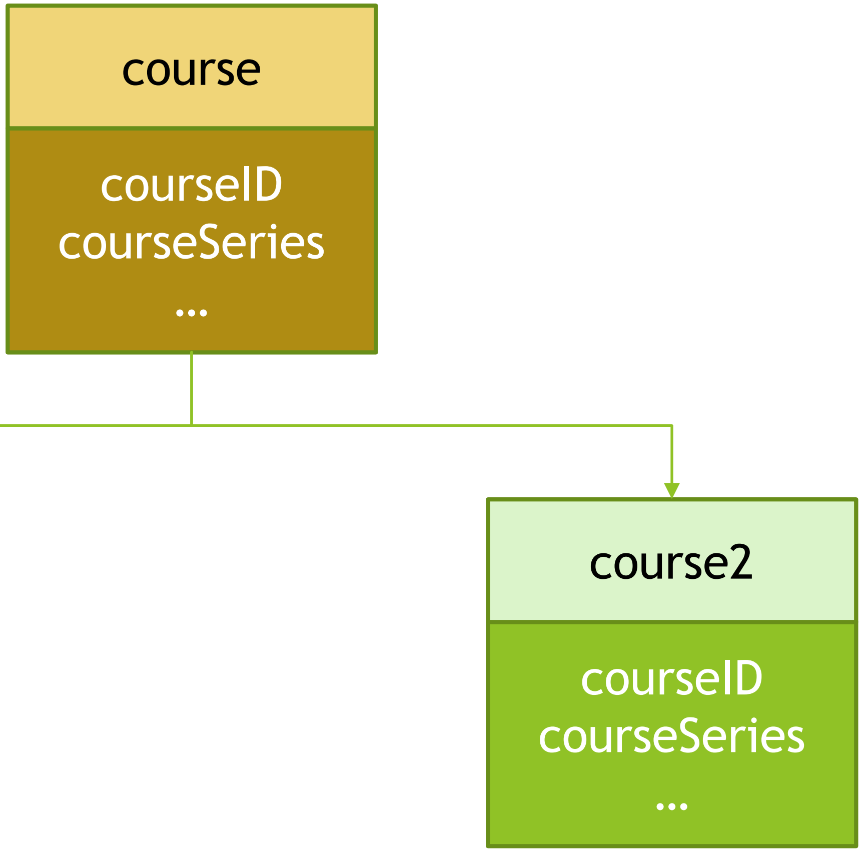


Object-Oriented JavaScript – Object Literal vs Constructor Function

Constructor Function – allows multiple instances

```
var course1 = new course (“COMP0809xx ", "JavaScript Frameworks", 12, "Monday", " 9am-
                        10am ", "10am-1pm");
var course2 = new course (“COMP0810xx ", “HTML5", 6, “Tuesday", “10am-11am", "1pm-
                        3pm");

course1.courseSeries;    // CGD

course2.courseSeries;    // CGD


course2.courseSeries = “Not CGD”;

course1.courseSeries;    // CGD

course2.courseSeries;    // Not CGD
```

Object-Oriented JavaScript – Object Literal vs Constructor Function



Object-Oriented JavaScript - Object

Adding a new property to an object:

```
course.degree = “Computer Games Development”;
```

Adding new method to an object or overriding existing method:

```
course.totalHours = function() {
    return this.courseDetails.duration * 4;
}
```

Delete a variable from an object;

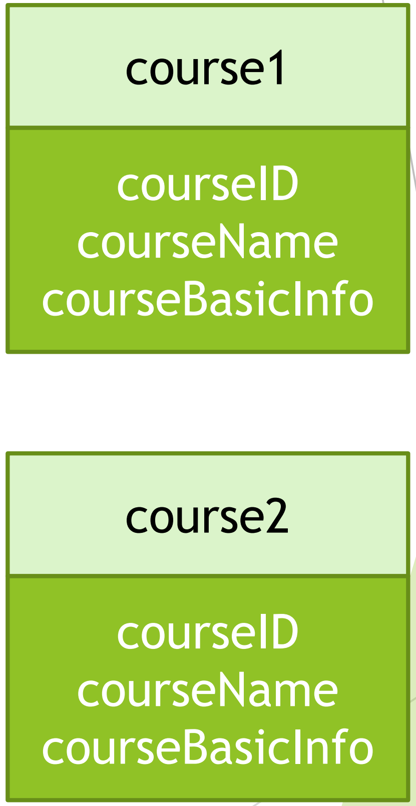
```
course.tempVar = ‘this is a temporary variable’;
delete course.tempVar ;
```

Object-Oriented JavaScript - Prototype

- A prototype is an object and every function created automatically gets a prototype property that points to a new blank object
- The constructor function is the prototype of the objects, (e.g. The *"course"* constructor function is the prototype of *course1* object and *course2* object)
- Can copy and reuse prototypes
- Advantage: saving on memory, reusability
- It is considered good practice to name constructor function with an upper-case first letter.
- The advantage of attaching functions to the prototype is that only a single copy of the function is created. If you make a change to the prototype of an object, then all the objects which share that prototype are updated with the new function.
- Prototypes can be combined to form more complex objects. Think of prototyping mentally as "attaching" a method to an object after it's been defined, in which all object instances then instantly share.

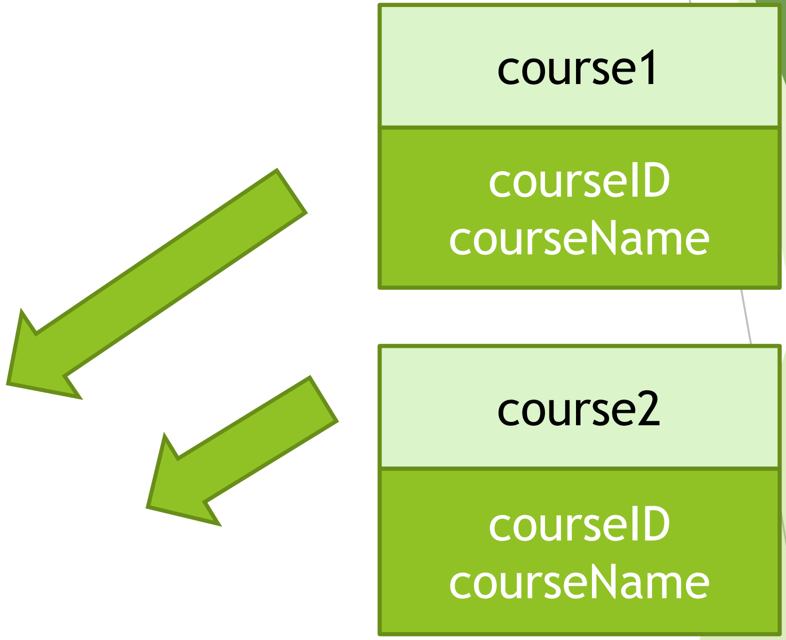
Object-Oriented JavaScript - Prototype Function

```
function course(courseID_pass, courseName_pass) {
        this.courseID = courseID_pass;
        this.courseName  = courseName_pass;
        this.courseBasicInfo: function(passedVar) {
          return this. courseID + “ ” + this. courseName + “ ” + passedVar;
        }
}

var course1 = new course (“COMP0809xx ", "JavaScript Frameworks")
var course2 = new course (“COMP0810xx ", “HTML5");
```



Object-Oriented JavaScript - Prototype Function

```
function course(courseID_pass, courseName_pass) {
            this.courseID = courseID_pass;
            this.courseName  = courseName_pass;
}

course.prototype.courseBasicInfo = function(passedVar) {
       return this. courseID + “ ” + this. courseName + “ ” + passedVar;
}
var course1 = new course (“COMP0809xx ", "JavaScript Frameworks")
var course2 = new course (“COMP0810xx ", “HTML5");
```



While using prototype, the objects do not need to carry the prototype function but instead the objects can use the function directly. Compared to the object method (method which declared inside the function), the prototype function will only have 1 copy in the memory. The prototype function also allows all objects which shared the same prototype to be manipulated simultaneously

Resources

- Stefanov, S., Sharma, K. C. (2013). Object-Oriented JavaScript. 2nd Ed. Packt Publishing
- http://javascriptissexy.com/oop-in-javascript-what-you-need-to-know/
- Object Literal vs Function Constructor: https://www.youtube.com/watch?v=O3JSPhwKowA