

Lecture 4

Object Oriented Programming

- What are the 3 pillars?



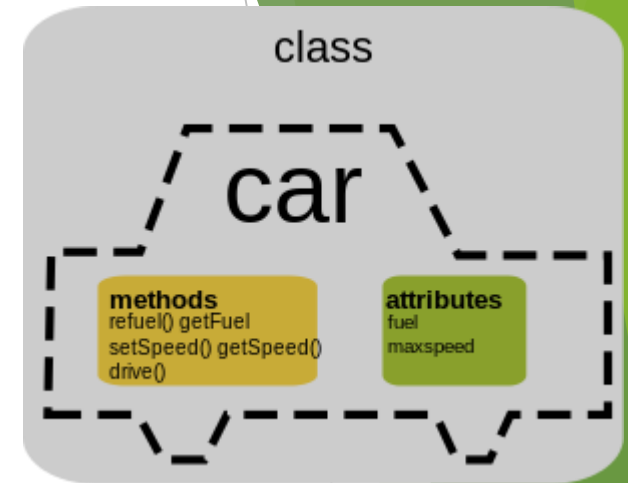
Object Oriented Programming

- What are the 3 pillars?
 - Encapsulation
 - Inheritance
 - Polymorphism



Object Oriented Programming

- ▶ What are the 3 pillars?
 - ▶ Encapsulation
 - ▶ Incorporation of data & operations into one package (a class)
 - ▶ Data can only be accessed through that package
 - ▶ “Information Hiding”
 - ▶ Inheritance
 - ▶ Polymorphism



Object Oriented Programming

- ▶ What are the 3 pillars?

- ▶ Encapsulation

- ▶ Inheritance

- ▶ Allows programmers create new classes based on an existing class
 - ▶ Methods and attributes from the parent class are inherited by the newly-created class
 - ▶ New methods and attributes can be created in the new class, but don't affect the parent class's definition

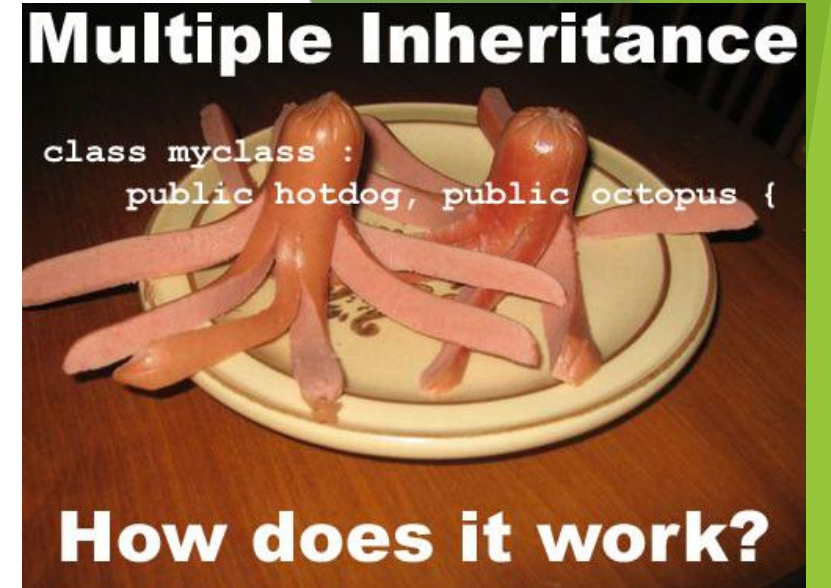
- ▶ Polymorphism

Q: What's the object-oriented way to become wealthy?

A: Inheritance

Object Oriented Programming

- ▶ What are the 3 pillars?
 - ▶ Encapsulation
 - ▶ Inheritance
 - ▶ Polymorphism
 - ▶ Allows creation of methods which describe the way to do some general function (Example: The “drive” method in the automobile class)
 - ▶ Polymorphic methods can adapt to specific types of objects.
 - ▶ E.g. Abstract methods in Java



Is JavaScript an OOP language?

- ▶ Well, not really ...
- ▶ JavaScript is known as an "object-inspired" language
- ▶ It uses objects by way of supporting inheritance and encapsulation, but it doesn't provide support for polymorphism.



“Object-Oriented” JavaScript

- ▶ More like “Object-Inspired” JavaScript
- ▶ We can create new, custom, re-usable objects in JavaScript that include their own methods, properties and events.
- ▶ Consider the following problem: “I want to record the colour, brand, horsepower and price of several cars.”



Solution:

- ▶ Without OOP Design:
 - ▶ Uses parallel arrays
 - ▶ Can be confusing
 - ▶ Difficult to keep track of which car has which colour, brand, etc.
- ▶ With OOP Design:
 - ▶ Calls to an API that contains the custom car object
 - ▶ Much cleaner code
 - ▶ Re-usable object

Javascript Object

- ▶ An *object* is a unique entity in a script and/or web page.
- ▶ An *object* has:
 - ▶ *Object properties*
 - ▶ *Object methods*
 - ▶ Can react to *events* in that object's environment
- ▶ A property is a characteristic that describes an object.
 - ▶ Similar to an adjective in grammar.
 - ▶ Properties are given values (hair colour is a property; i.e. brown or red)
- ▶ A *method* is something an object can do.
 - ▶ Similar to a verb in grammar.

Javascript Comments

- ▶ In lab submissions, make sure to include comments
 - ▶ Marks will be deducted for inadequately commented code

Javascript Object and Constructors

- ▶ Have seen some already
- ▶ JavaScript has a library of "blueprints"/constructor methods available to it that define the nature of some objects.

Creating a variable

1

2

3

```
var userName = new String("Dilbert");
```

What's going on?

1. The reserved word "**var**" tells JavaScript to allocate some space in memory for a variable.
2. We give a label to that variable, giving it the name **userName**.
3. The "**=**" means "*gets the value off*" in programming. It does NOT mean "equals."

Creating a Variable

4

5

```
var userName = new String("Dilbert");
```

4. The reserved word "**new**" is called an *instance operator* and it tells JavaScript to expect that we'll be creating a new *instance* of object from an existing "blueprint", in this case, from the String blueprint.
5. We use the constructor method **String()** to copy the string blueprint to our **userName** variable.

Creating a Variable

6

```
var userName = new String("Dilbert");
```

6. The **String()** method allows us to supply an optional *initial value* for our new string. If we choose to, we need to supply a value that is of string-type. This is called *variable initialization*. Notice the word **"Dilbert"** is in quotations? The quotations are a way of identifying the word as a string. When we give a variable its first value, we also *type* the variable. JavaScript is an *implicitly typed* language.

Creating Variables for Primitives

- ▶ For any of JavaScript's *primitive data types* (strings, integers, floats, Booleans), you can create a variable without calling on the constructor:
`var userName = "Dilbert";`
- ▶ Although this is syntactically correct, try using constructor methods. Why? Since other, more formal, languages (like Java) demand constructors.
- ▶ Good to get used to the habit of using them in JavaScript.
- ▶ Practice makes perfect!

Variable Assignment: Input

```
userName = window.prompt("Your name?", "");
```

Variable Name

Method:
Creates a box
to ask a
question

Method Arguments:
Question to be asked &
default value

Assignment Operator:
"Gets the value of"

Dot Operator

- ▶ Did you notice in the previous example that the window object connected to its prompt method using a period? The period is called a *dot operator*.
- ▶ A dot operator is a way of showing how things are connected to objects. In the previous example, the dot operator connected the window object to its prompt method.

JS Variables and User-Defined Functions

- ▶ For some functionality, you cannot achieve by only using the built-in functions.
- ▶ You can define a function as follows:
 - ▶ Functions are declared using the function reserved word
 - ▶ The return value is not declared, nor are the types of the arguments

```
Var myVar = value;  
function <function_name> (parameters)  
{  
    // code segments;  
}
```

JS Objects

```
Var myVar = value;  
function <function_name> (parameters)  
{  
    // code segments;  
}
```

```
Var myObject = {  
    myVar : value,  
    <function_name> : function(parameters)  
    {  
        // code segments;  
    }  
}
```

Lab

- ▶ Ball and pizza object and animation
- ▶ Encapsulate ball/pizza into object with resize, move and draw methods

HTML5 Canvas Animation

- ▶ Until now, the biggest limitation is, that once a shape gets drawn, it stays that way.
- ▶ If we need to move it we have to redraw it and everything that was drawn before it.
- ▶ It takes a lot of time to redraw complex frames and the performance depends highly on the speed of the computer

HTML5 Canvas Basic Animation Steps

- ▶ Clear the canvas
 - ▶ Unless the shapes you'll be drawing fill the complete canvas (for instance a backdrop image), you need to clear any shapes that have been drawn previously. The easiest way to do this is using the `clearRect()` method.
- ▶ Draw animated shapes
 - ▶ The step where you do the actual frame rendering.
- ▶ Controlling an animation:
 - ▶ `requestAnimationFrame(callback)`
 - ▶ Tells the browser that you wish to perform an animation and requests that the browser call a specified function to update an animation before the next repaint.

requestAnimationFrame(callback) - ball.html

```
<script type="text/javascript">
  // Gets a handle to the element with id canvasOne.
  var canvas = document.getElementById("canvas-for-ball");
  // Get a 2D context for the canvas.
  var ctx = canvas.getContext("2d");

  // The vertical location of the ball.
  var y = 10;
  // A function to repeat every time the animation loops.
  function repeatme() {
    // Draw the ball (stroked, not filled).
    ctx.beginPath();
    ctx.arc(50, y, 3, 0, 2 * Math.PI);
    ctx.stroke();

    // Update the y location.
    y += 1;
    window.requestAnimationFrame(repeatme);
  }
  // Get the animation going.
  repeatme();
</script>
```


Lab Instructions (Week 4)

- ▶ Put a border around the canvas so that we can see its edges.
- ▶ Given ball.html, Stop the ball moving when it hits the bottom of the canvas.
 - ▶ You might use an if statement for this purpose.
- ▶ Clear the canvas at each step of the animation, so that only one copy of the ball is visible at a time. You can use the clearRect method for this.
 - ▶ `ctx.clearRect(left, top, width, height);`
- ▶ Represent the ball as an object, rather than by a global y value and hard-coded x and r values.
- ▶ Change the code so that the ball starts moving up the way once it reaches the bottom of the screen.
- ▶ Change the code so that the ball bounces from the bottom to the top of the screen, and back again, repeatedly.

Lab Instructions (Week 4)

- ▶ Give the ball a horizontal velocity, as well as a vertical one, and have it bounce off the left and right sides of the canvas also.
- ▶ Change the code so that the pizza from lab 2 rather than the ball bounces
- ▶ Advanced:
 - ▶ Give the ball/pizza a downwards acceleration, so that it gets faster as it falls, and slower as it ascends.
- ▶ Material:
 - ▶ Mozilla's docs on `Window.requestAnimationFrame()` - <https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame>

Lab Submission

- ▶ Submit 3 separate HTML files and 1 Word file containing 3 screenshots
 - ▶ Lab 1 - demonstrating drawing rectangles, circles, transparency, CSS border, Canvas resizing, Pacman
 - ▶ Lab 2 - demonstrating a ball object with resize, move and draw functions. Also applying same functions to drawing a pizza with 7 slices
 - ▶ Lab 3 - demonstrating ball and pizza animation (as described above)
 - ▶ Screenshots of each of the 3 labs
- ▶ Submission Deadline: October 10th @15:00
- ▶ Marks will be awarded for well-commented, well-indented code
- ▶ Code must run direct from HTML file - no console input should be necessary
- ▶ Marks will be deducted for lateness, not following above instructions.