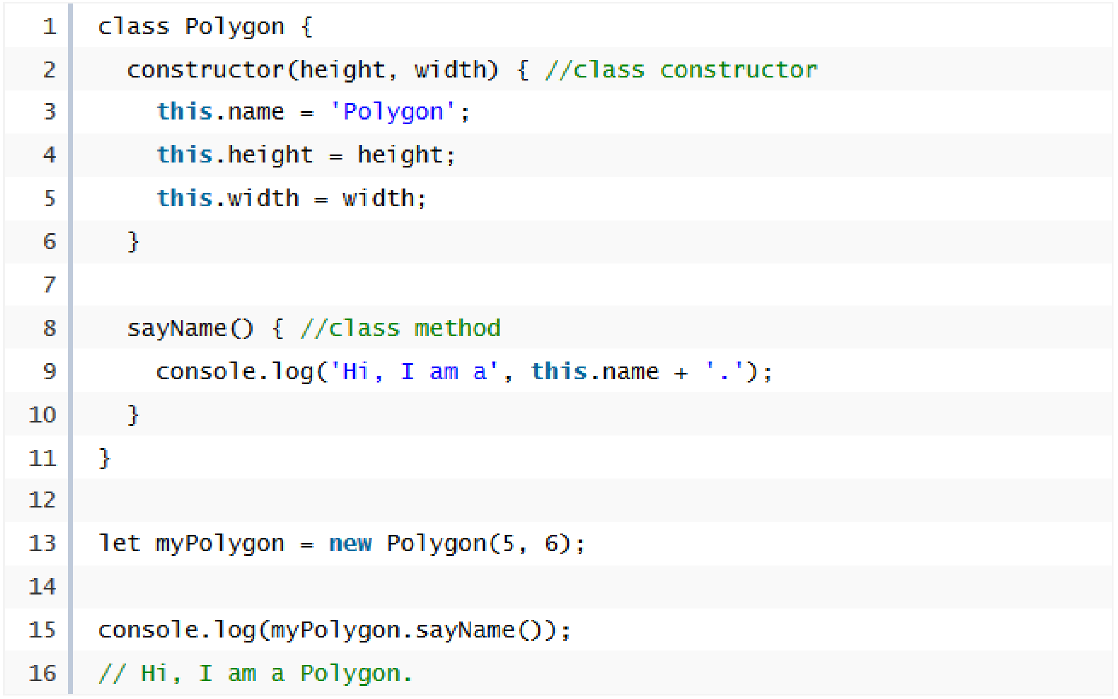
Advanced JavaScript

Object Oriented JavaScript

ITI – Assiut Branch
Eng. Hany Saad

Classes in ES6

- ES6 introduces JavaScript classes that are built upon the existing prototype-based inheritance.
- The new syntax makes it more straightforward to create objects, take leverage of inheritance, and reuse code.
- Classes are in fact "special functions", and just as you can define function expressions and function declarations, the class syntax has two components: class expressions and class declarations.



```
class Polygon {
  constructor(height, width) { //class constructor
    this.name = 'Polygon';
    this.height = height;
    this.width = width;
  }

  sayName() { //class method
    console.log('Hi, I am a', this.name + '.');
  }
}

let myPolygon = new Polygon(5, 6);

console.log(myPolygon.sayName());
// Hi, I am a Polygon.
```

Demo!

Closures

Closures

- Closure is one of the most powerful features of JavaScript.
- A closure is an expression (typically a function) that can have free variables together with an environment that binds those variables (that "closes" the expression).
- It is created when the inner function is somehow made available to any scope outside the outer function.
- If the inner function manages to survive beyond the life of the outer function; the variables and functions defined in the outer function will live longer than the outer function itself, since the inner function has access to the scope of the outer function.
- In short words:
    - a closure is the local variables for a function — kept alive after the function has returned

Closures (cont.)

- **Example:**

```
function sayHello2(name) {
    var text = 'Hello ' + name; // Local variable
    var sayAlert = function() { alert(text); }
    return sayAlert;    //returning reverence to the inner func.
}
var say2 = sayHello2('Bob');
//say2 holds a reference to the inner func. That access the outer func variables.
say2(); // alerts "Hello Bob"
```

- The above code has a closure because the anonymous function function() { alert(text); } is declared inside another function, sayHello2() in this example. In JavaScript, if you use the function keyword inside another function, you are creating a closure.

- In JavaScript, if you declare a function within another function, then the local variables can remain accessible after returning from the function you called. This is demonstrated above, because we call the function say2() after we have returned from sayHello2(). Notice that the code that we call access the variable text, which was a local variable of the function sayHello2().

- The anonymous function can reference text which holds the value 'Hello Bob' because the local variables of sayHello2() are kept in a closure.

- The magic is that in JavaScript a function reference also has a secret reference to the closure it was created in.

Closures (cont.)

- **Another Example (Problem):**

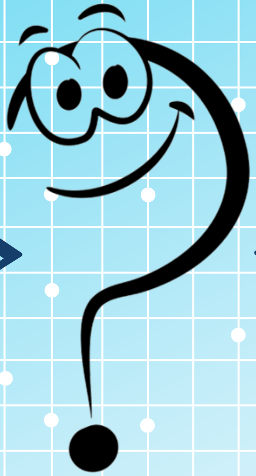
```
function closureTest(){
    var arr = [];
    for(var i = 0; i < 3; i ++) {
        arr.push(function(){
                alert(i);
            });
    }
    return arr;
}
var cFn = closureTest();
cFn[0](); //3
cFn[1](); //3
cFn[2](); //3
```

- Note that when you run the example, “3" is alerted three times! This is because there is only one closure for the local variables for closureTest.
- When the anonymous functions are called on the line cFn[0](); they all use the same single closure, and they use the current value for i and item within that one closure (where i has a value of 3 because the loop had completed, and item has a value of ‘3').

Closures (cont.)

- **Another Example (Solution):**

```
function closureTest(){
                var arr=[]
                var i;
                for(var i = 0; i < 3; i ++){
                arr.push((function(j){ return function(){
                                                                  alert(j);
                                                  }
                                  })(i)
                );
                }
 return arr;
}
var cFn = closureTest();
cFn[0](); //0
cFn[1](); //1
cFn[2](); //2
```

<SCRIPT >  </SCRIPT>

```
<script>document.writeln(“Thank You!”)</script>
```