**Subclasses with ES6**

Now that we've looked at creating classes in JavaScript. Let's use the

new super and extends keywords to extend a class.

**class** **Tree** {

constructor(size = '10', leaves = {spring: 'green', summer: 'green', fall: 'orange', winter: null}) {

**this**.size = size;

**this**.leaves = leaves;

**this**.leafColor = null;

}

changeSeason(season) {

**this**.leafColor = **this**.leaves[season];

**if** (season === 'spring') {

**this**.size += 1;

}

}

}

**class** **Maple** **extends** **Tree** {

constructor(syrupQty = 15, size, leaves) {

**super**(size, leaves);

**this**.syrupQty = syrupQty;

}

changeSeason(season) {

**super**.changeSeason(season);

**if** (season === 'spring') {

**this**.syrupQty += 1;

}

}

gatherSyrup() {

**this**.syrupQty -= 3;

}

}

**const** myMaple = **new** Maple(15, 5);

myMaple.changeSeason('fall');

myMaple.gatherSyrup();

myMaple.changeSeason('spring');

Both Tree and Maple are JavaScript classes. The Maple class is a "subclass" of Tree and uses the extends keyword to set itself as a "subclass". To get from the "subclass" to the parent class, the super keyword is used. Did you notice that super was used in two different ways? In Maple's constructor method, super is used as a function. In Maple's changeSeason() method, super is used as an object!

**Compared to ES5 subclasses**

Let's see this same functionality, but written in ES5 code:

**function** **Tree**(size, leaves) {

**this**.size = (**typeof** size === "undefined")? 10 : size;

**const** defaultLeaves = {spring: 'green', summer: 'green', fall: 'orange', winter: null};

**this**.leaves = (**typeof** leaves === "undefined")? defaultLeaves : leaves;

**this**.leafColor;

}

Tree.prototype.changeSeason = **function**(season) {

**this**.leafColor = **this**.leaves[season];

**if** (season === 'spring') {

**this**.size += 1;

}

}

**function** **Maple** (syrupQty, size, leaves) {

Tree.call(**this**, size, leaves);

**this**.syrupQty = (**typeof** syrupQty === "undefined")? 15 : syrupQty;

}

Maple.prototype = Object.create(Tree.prototype);

Maple.prototype.constructor = Maple;

Maple.prototype.changeSeason = **function**(season) {

Tree.prototype.changeSeason.call(**this**, season);

**if** (season === 'spring') {

**this**.syrupQty += 1;

}

}

Maple.prototype.gatherSyrup = **function**() {

**this**.syrupQty -= 3;

}

**const** myMaple = **new** Maple(15, 5);

myMaple.changeSeason('fall');

myMaple.gatherSyrup();

myMaple.changeSeason('spring');

Both this code and the class-style code above achieve the same functionality.