8.3 Classes (ES6)

Classes in EcmaScript 6

EcmaScript 6 (ES6) simplifies class declarations, introducing syntax that looks more familiar to a Java or C# programmer. Although the syntax is different, the underlying prototype model is not changed.

A class is declared by using the ***class keyword*** followed by a class name. The class is implemented by declaring methods within braces. Each method declaration is similar to a function declaration, but without the `function` keyword. The method name ***constructor()*** is reserved for the class constructor.

PARTICIPATION ACTIVITY 8.3.1: CityState ES6 class.

1 2 3 4 2x speed

```
class CityState {
   constructor(city, state) {
      this.city = city;
      this.state = state;
   }

   toHTML() {
      return this.city + ", " + this.state;
   }
}

let Austin = new CityState("Austin", "Texas");
let Madison = new CityState("Madison", "Wisconsin");
console.log(Austin.toHTML());
```

Austin:
city: "Austin"
state: "Texas"

Madison:
city: "Madison"
state: "Wisconsin"

The Austin object's toHTML() is called to return an HTML string containing the city and state name.

Captions

1. The CityState class is declared with a constructor method and a toHTML() method.
2. new CityState is used to construct Austin, an instance of CityState. The method named "constructor" is called automatically.
3. The Madison object is constructed similarly.
4. The Austin object's toHTML() is called to return an HTML string containing the city and state name.

Feedback?

PARTICIPATION ACTIVITY 8.3.2: Classes.

Refer to the `CityState` class in the animation above.

1) Which line properly constructs a `CityState` instance, referenced by a variable named `Boise`, for Boise, Idaho?

- `CityState Boise = new CityState("Boise", "Idaho");`
- `let Boise = CityState("Boise", "Idaho");`
- `let Boise = new CityState("Boise", "Idaho");`

Correct

`let Boise` declares the `Boise` variable, and `new CityState("Boise", "Idaho");` constructs the `CityState` instance.

2) If `Boise` is an instance of `CityState`, the statement `console.log(Boise.state);` _____.

- logs `Boise`'s state, Idaho, to the console
- logs nothing, since `state` is a private property
- throws an exception. The proper statement is `console.log(Boise.getState());`

Correct

The `Boise` object's `city` and `state` properties are accessed as `Boise.city` and `Boise.state`, respectively.

3) What is wrong with the following attempt to declare a `toString()` method for `CityState`?

```
class CityState {

constructor(city, state) {
      this.city = city;
      this.state = state;
   }

   toHTML() {
      return this.city + ", " + this.state;
   }

   function toString() {
      return this.city + ", " + this.state;
   }
}
```

- `toString()` requires at least 1 parameter.
- The "function" keyword must be removed. ✓
- The name "toString" is reserved and must be changed.

Correct

Methods in classes do not begin with the function keyword. Removing "function" makes the method declaration valid.

Feedback?

Inheritance

The ***extends keyword*** allows one class to inherit from another. In the inheriting class' constructor, calling the ***super()*** function calls the parent class' constructor. `super()` must be called before using the `this` keyword in the inheriting class' constructor.

Figure 8.3.1: Person class and inheriting Student class.

```
class Person {
   constructor(name, age) {
      this.name = name;
      this.age = age;
   }

   sayHello() {
      console.log("Hello.  My name is " + this.name + ".");
   }
}

// Student inherits from Person
class Student extends Person {
   constructor(name, age, gpa) {
      super(name, age);     // Call parent constructor
      this.gpa = gpa;
   }

   // Override parent's sayHello method
   sayHello() {
      console.log("Hi, I'm " + this.name + " with a " + this.gpa + "
GPA!");
   }
}
```

Feedback?

PARTICIPATION ACTIVITY 8.3.3: Inheritance.

Suppose a `GraduateStudent` class, that inherits from `Student` is being defined. The class adds a `status` member, which is set to either "Masters" or "PhD". Match the blank lines to the appropriate code pieces.

```
class GraduateStudent __(1)__ {
   constructor(name, age, gpa, mastersOrPhd) {
      __(2)__(name, age, gpa);
      __(3)__ = mastersOrPhd;
   }

   // Override parent's sayHello method
   sayHello() {
      __(4)__("Hi, I'm " + this.name + ", a " + this.status +
         " student with a " + this.gpa + " GPA!");
   }
}
```

If unable to drag and drop, refresh the page.

3	`this.status` The parent class constructor assigns to inherited members: `name`, `age`, and `gpa`. The `GraduateStudent` constructor method must assign to the new `status` member.	Correct
4	`console.log` Like the parent class' `sayHello()` method, `console.log()` is used to show a message about the graduate student.	Correct
1	`extends Student` `extends Student` is added after the `GraduateStudent` class name to inherit from the `Student` class.	Correct
2	`super` `super` is used to call the parent class constructor, which assigns members: `name`, `age`, and `gpa`.	Correct

Reset

Feedback?

Getters and setters

A class method declaration preceded by the ***get keyword*** defines a getter method for a property. A class method declaration preceded by the ***set keyword*** defines a setter method for a property. Defining either a getter or setter method named X, adds a property named X to each class instance. Ex: A `Square` class may have a getter method declared as:

```
get area() {
   return this.width * this.width;
}
```

After creating a `Square` instance named `s1`, the expression `s1.area`, without parentheses, gets the square's area.

A get method must not have parameters. A set method must have 1 parameter. A property can be defined via a getter only, a setter only, or both a getter and setter.

PARTICIPATION ACTIVITY 8.3.4: Getters and setters can be used to convert an angle between degrees and radians.

1 2 3 4 2x speed

```
class Angle {
  constructor(angleRadians) {
     this.radians = angleRadians;
  }

  get degrees() {
     return this.radians * 180.0 / Math.PI;
  }

  set degrees(angleDegrees) {
     this.radians = angleDegrees * Math.PI / 180.0;
  }
}

let angle1 = new Angle(Math.PI);
console.log(angle1.degrees);
angle1.degrees = 270.0;
console.log(angle1.radians);
```

angle1:

radians: 4.71238898038469

console:

```
180
4.71238898038469
```

"radians" is the angle1 object's only numerical property. The get and set allow "degrees" to be accessed like a property, even though the object never internally stores the angle in degrees.

Captions

1. angle1 is constructed as an instance of the Angle class, storing the angle in radians.
2. Accessing angle1's "degrees" property calls the corresponding getter. The angle, converted from radians to degrees, is returned.
3. Assigning to the "degrees" property calls the setter to convert the angle to radians.
4. "radians" is the angle1 object's only numerical property. The get and set allow "degrees" to be accessed like a property, even though the object never internally stores the angle in degrees.

Feedback?

PARTICIPATION ACTIVITY 8.3.5: Getter and setter methods.

1) If a getter method for property X is added to a class, a setter method for X _____.

- must also be added
- can be added, but is not required
- cannot be added

Correct

A getter can be declared with or without an accompanying setter.

2) A setter method _____.

- requires 0 parameters
- requires 1 parameter
- can have any number of parameters

Correct

A setter method takes exactly 1 parameter, which is the value to set.

3) Suppose `obj` is an instance of a class with a property named `value`, which has both a getter and setter. The statement below calls the _____.

```
obj.value += 10;
```

- getter only
- setter only
- getter first, then the setter
- setter first, then the getter

Correct

The statement is equivalent to

```
obj.value = obj.value + 10;
```

The getter is called first to retrieve the value, 10 is added, and the setter is called to assign the result.

Feedback?

Static methods

A ***static method*** is a method that can be called without creating an instance of the class. A static method is declared with the ***static*** keyword preceding the method name. The method is called with the syntax: `ClassName.methodName(arguments)`.

PARTICIPATION ACTIVITY 8.3.6: StringOps class with static methods.

1 2 2x speed

```
class StringOps {
   static isLowerCase(str) {
      return str.toLowerCase() === str;
   }

   static countChar(str, char) {
      let count = 0;
      for (let i = 0; i < str.length; i++) {
         if (str.charAt(i) === char) {
            count++;
         }
      }
      return count;
   }
}

console.log(StringOps.isLowerCase("abc"));
console.log(StringOps.isLowerCase("zyBooks"));
console.log(StringOps.countChar("zyBooks", "o"));
```

console:

```
true
false
2
```

StringOps.countChar("zyBooks", "o") is called to count the number of 'o' characters in the string "zyBooks".

Captions

1. The StringOps class has a static isLowerCase() method, which is called with the syntax: StringOps.isLowerCase(...).
2. StringOps.countChar("zyBooks", "o") is called to count the number of 'o' characters in the string "zyBooks".

Feedback?

PARTICIPATION ACTIVITY 8.3.7: StringOps class and static method concepts.

1) The following is a valid static method declaration that could be inserted into the `StringOps` class:

```
static
getClassName() {
   return
"StringOps";
}
```

- (●) True
- ( ) False

Correct

The method name is preceded by `static`, and all other aspects of the declaration are correct. A static method can have 0 parameters, if desired.

2) A class can have both static and non-static methods.

- (•) True
- ( ) False

Correct

Both static and non-static methods can be declared in a single class. Non-static methods can access instance data. Utility functions that don't require instance data are commonly added as static methods.

3) Non-static methods can also be called with the `ClassName.methodName(arguments)` syntax.

- ( ) True
- (•) False

Correct

Non-static members are added to the class constructor function's prototype, not the constructor function itself. So non-static methods cannot be called with the same syntax as static methods.

Feedback?

Exploring further:

- Classes (MDN)

CHALLENGE ACTIVITY 8.3.1: Classes ES6.

530096.4000608.qx3zqy7

Start

Create a Weasel class with a constructor that takes two arguments and initializes properties weight and gender in that order.

```
1
2 /* Your solution goes here */
3
```

1 2 3 4

Check Next

Feedback?

How was this section?

Provide section feedback