

Object Oriented Programming for AP CSP

by Jiyeon Kim

General Overview

This unit is designed to be inserted into the end of an AP CSP curriculum. It will provide an overview and an introduction to object oriented programming. Students will learn how to make objects, give them attributes, and create functions for those objects. The summative assessment for the unit will be a project where students display trading cards.

Motivation for Unit

There were 2 motives for creating this unit. First, I found my students were struggling using functions that involved dot notation. Second, many of my students take AP CSA after my course. I think this unit will be a good connection to that unit to provide that background.

Standards Referenced

(select one of the standards sets reviewed in class (CSTA, NY, MA, RI), include a link and a brief explanation as to why you selected that set)

Tools Used

HTML, CSS, Javascript. Bootstrap to supplement CSS styling.

Atom and browser for observing and troubleshooting websites. Students can use any documentation available for HTML, CSS, JS, and Bootstrap.

Resources

- html elements: https://www.w3schools.com/html/html_elements.asp
 - bootstrap: <https://getbootstrap.com/docs/5.1/getting-started/introduction/>
 - JS: <https://www.w3schools.com/js/default.asp>
 - personal slide show adapted from Code.org
-

Lessons

Total length: 2 Weeks

- Lesson 0: Introduction to objects, spotting objects in real life, working with objects in JS. Examples will delve into functions as well. HW: come up with an object with attributes from your life. List of options for students to choose from
- Lesson 1: Writing your example from HW in JS code, coming up with a toString() function, add object function, getAttribute() functions, setAttribute() function. HW: finish what was started in class

- Lesson 2: Quiz, Introduce project, have students brain storm what they'd like to work on. Fill out a worksheet dealing with what kind of attributes they want they would like to include
 - Lesson 3: Introduce BootStrap tools Work on project, share out progress with partners, troubleshoot.
 - Lesson 4: Introduce bootstrap part 2 Continue to work on project. Present on what they have thus far. Turn in by next class
-

Assesments

- Formative assessments:
 - Homework assignments to create a new type of object. Could be related to cars, movies, book characters, etc
 - Quiz on vocabulary relating to object oriented programming and predicting code results
 - Summative Assessment: Project of creating a deck of trading cards with beautiful website to display these cards.
-

00_IntroductionToObjects

Main objective:

- Introduce what objects are
- Gain solid foundation of what creating an object is capable of doing

Standards (NYSED)

- 9-12.CT.4: Implement a program using a combination of student-defined and third-party functions to organize the computation.
- 9-12.CT.7: Design or remix a program that utilizes a data structure to maintain changes to related pieces of data

In class exercises

- Task 0: Introduce the idea of an object
 - Class definition of an object in real life (colloquial definition)
 - Show students example of a coded object using the examples from previous slides
- Task 1: Code bird objects
- Task 2: Create functions for bird objects
- Task 3: notes on functions statements vs function expressions
- Task 4: create more functions with function expressions
- Task 5: if time, start on homework


Notes/Handouts

00 Introduction to Objects


What is an object?

Come up with a class definition for 'object'

Examples 1:



Examples 2:




Come up with something you can classify as an object

- Your object must have many types, e.g. we have a car object but there are many variations of this car object. Come up with a minimum 3 variations, each with 3 characteristics
- Hint: Humans make classifications of objects all the time.
 - Science/animals are a great example of categorizing things
 - Do you ever look up 'stats' for a specific team?


Object name	Property 1	Property 2	Property 3

What is an object in computer science? Example 1



```
//example 1
var car1 = {
  type: "sedan",
  numWheels: 4,
  color: "brown",
  seats: 5
}
var car2 = {
  type: "convertible",
  numWheels: 4,
  color: "red",
  seats: 5
}
```

What is an object in computer science? Example 2



```
var bird1 = {
  name: "robin",
  color: ["brown", "orange"],
  flight: true,
  wingspan: 74 //base 10 grams
}
var bird2 = {
  name: "cardinal",
  color: "brown",
  flight: true,
  wingspan: 45 //base 10 grams
}
var bird3 = {
  name: "big bird",
  color: ["yellow"],
  flight: false,
  wingspan: 127000,
}
```

Code along with me!
We'll be referring to this bird example often.

Code your object example

Add another bird object to our code from the previous slide.

Great. I made an object. Now what.

```
function fly(obj){
  if (obj.flight == true){
    console.log("flap flap");
  }else{
    console.log("ground ground");
  }
}
fly(bird2);
fly(bird3);
```

You can also add functions to your objects. It's what makes them so useful! What do you think this will show in the console?

There are 2 ways to define functions in JS

```
var bird3 = {
  name: "big bird",
  color: ["yellow"],
  flight: false,
  wingspan: 127000,
  fly: function(){
    if (this.flight == true){
      console.log("flap flap");
    }else{
      console.log("ground ground");
    }
  }
}
```

```
function fly(obj){
  if (obj.flight == true){
    console.log("flap flap");
  }else{
    console.log("ground ground");
  }
}
```

What's different about these two ways?

The way you use them is different too!

```
var bird3 = {
  name: "big bird",
  color: ["yellow"],
  flight: false,
  wingspan: 127000,
  fly: function(){
    if (this.flight == true){
      console.log("flap flap");
    }else{
      console.log("ground ground");
    }
  }
}
bird3.fly();
```

```
function fly(obj){
  if (obj.flight == true){
    console.log("flap flap");
  }else{
    console.log("ground ground");
  }
}
fly(bird3);
```

This is Dot Notation. We're going to stick to this version. It's more common in other object oriented languages. (Java)

Dot Notation

```
var bird3 = {
  name: "big bird",
  color: "yellow",
  flight: false,
  avweight: 127886,
  fly: function() {
    if (this.flight == true) {
      console.log("flap flap");
      alert(
        console.log("ground ground"));
    }
  }
}
```

We use dot notation all the time because all of JS is built on objects. Where do you see dot notation here? Add circles.

10

Add another function to birds together

```
var bird3 = {
  name: "big bird",
  color: "yellow",
  flight: false,
  avweight: 127886,
  fly: function() {
    if (this.flight == true) {
      console.log("flap flap");
      alert(
        console.log("ground ground"));
    }
  }
}
```

What else do birds do? Let's add another function.

11

Code your object function

Add a function that goes with your object. Try to make sure it makes "sense".
E.g. My bird objects had a function called "fly".
Copy and paste your code here.

12

HW

- Using your examples from class (slide 3), code your own objects.
- You must also have 2 functions to go with your objects.

13

Resources:

- Object Introduction: https://www.w3schools.com/js/js_objects.asp
- Ways to declare functions: <https://www.telerik.com/blogs/four-ways-to-create-a-function-in-javascript>

Assignments: Practice with objects

- Students will think of a "set" of entities on a Ex: Superheroes, colleges, people, plants
- They will then come up with a bunch of properties superheroes have Ex: powers, origin, company, enemy, weakness
- Students will then create a set of objects (at least 5) using their example
- the students must be able to print their objects and their properties to their console using a loop.

01_GettingComfortableWithObjects

Main objective:

- Have students continue to practice using objects and thinking about their utility

Standards (NYSED)

- 9-12.CT.4: Implement a program using a combination of student-defined and third-party functions to organize the computation.
- 9-12.CT.7: Design or remix a program that utilizes a data structure to maintain changes to related pieces of data

In class exercises

- Students will get a re-cap of last class and what objects are
- Students will get an introduction of dot notation
- Students will have a code-along segment of teacher directed learning where students create a variety of objects and create functions for them.
- Students will problem solve through an array of object based problems requiring students to create objects and functions that go with objects. These will be student directed but teacher help available
- Students will then incorporate that problem solving logic with HTML DOM elements. This assignment is designed to prep them for their project with super heroes.

Notes

```
var student = {  
  lastname: "Kim",  
  firstname: "Jiyeon",  
  grade: 21,  
  schedule: ["AP CSP", "AP Chem", "Core Chem"],  
  getFullName: function(){  
    return this.firstname + " " + this.lastname;  
  }  
}  
  
var phone = {  
  os: "android",  
  model: "samsung s20",  
  camera: "12mp",  
  battery: "4000mAh",  
  getModelName: function(){  
    return this.model;  
  }  
}
```

Handouts

<h3>01_GettingComfortableWith Objects</h3>	<p>Arrays are objects</p> <pre>var sesameSt = ["big bird", "elmo", "cookie monster", "oscar"];</pre> <p>This array has a few different attributes. Can you name them?</p>	<p>Arrays also have functions that can be used with arrays</p> <p>They use <u>dot notation</u>.</p> <pre>var sesameSt = ["big bird", "elmo", "cookie monster", "oscar"]; sesameSt.push("Ernie"); sesameSt.splice(1,1);</pre> <p>Predict the output of the lines above.</p>
1	2	3
<p>Code-alongs</p> <p>Let's create a few types of objects so we can get comfortable. For each, we'll come up with attributes and functions that go with each object type</p> <ol style="list-style-type: none">1. Student2. Smartphones3. Musical Artist4. Athlete	<p>Object Problem Set</p> <p>This problem set is designed to get you to practice more!</p>	<p>Object Problem Set 2</p> <p>This problem set is designed to get you to practice more but this time work in HTML elements as well.</p>
4	5	6

Resources

- Edabit Objects Challenges: <https://edabit.com/challenges>

Assignments:

- Study for quiz next class
- Finish whatever problem set wasn't finished in class

02_BrainstormingProject

Main objective

- Assess students' basic knowledge in constructing an object
- Introduce bootstrap as a designing library

Standards

- 9-12.CT.4: Implement a program using a combination of student-defined and third-party functions to organize the computation.
- 9-12.CT.7: Design or remix a program that utilizes a data structure to maintain changes to related pieces of data

In class exercises:

- Give students a quiz to assess beginning object knowledge
- Give students the project document to work on and brainstorm
- Students must have completed the project planning document and have coded 1 of their 10 objects before they leave class.

Notes/Handouts

Objects Quiz

Name: _____

1. Define the following:

a. Object: _____

b. Property: _____

c. Method: _____

Use the space below to write your code.

2. Create an object `rectangularPrism` with 3 attributes of width, height, and depth. You can assign whatever values you want to this object
3. Create a "getter" method for the width.
4. Create a "setter" method for the height.
5. Create a `getVolume` method and return the object's volume. You can use the object as a parameter.
Volume = width x height x depth
6. Create a `getSurfaceArea` method and return the object's surface area. You must be able to use dot notation and not use the object as a parameter.
Surface area = $2(\text{width} \times \text{height}) + 2(\text{height} \times \text{depth}) + 2(\text{width} \times \text{depth})$

02_BrainstormingProject

Goal: Create an interactive trading card webpage where the user can select for a “card” using an attribute. Must have 2 tags you can sort by.

Requirements:

- Must fill out this project planning document and hand in with project
- Must implement object oriented programming for each trading card object
- Must have at least 10 cards for the user to view
- Each trading card object must have at least 3 attributes
- User must be able to see each card that they pick
- Users must be able to see all relevant cards if they sort by a certain attribute.
- Each of your objects must be an element in your array.
- Create a function that allows users to search for a specific attribute.

Project Planning Document

1. What type of trading card do you think you will implement?
2. What are the attributes?
3. What will be your main reference if you get stuck on how to code a function?
4. Jot down some questions you have here:
5. Information Gathering: Use the table below to gather information for your trading card implementation.

	Attribute 1:	Attribute 2:	Attribute 3:
Var name 1: <input type="text"/>			
Var name 2: <input type="text"/>			
Var name 3: <input type="text"/>			
Var name 4: <input type="text"/>			
Var name 5: <input type="text"/>			
Var name 6: <input type="text"/>			
Var name 7: <input type="text"/>			
Var name 8: <input type="text"/>			
Var name 9: <input type="text"/>			
Var name 10: <input type="text"/>			

Resources

Assignments

- Finish project planning document for HW
- Code 1 out of 10 objects

03_BootstrapIntroWorkTime

Main objective:

- Provide time for students to work on projects
- Get feedback from peers
- Troubleshoot work

Standards

- 9-12.CT.10: Collaboratively design and develop a program or computational artifact for a specific audience and create documentation outlining implementation features to inform collaborators and users.
- 9-12.DL.2: Communicate and work collaboratively with others using digital tools to support individual learning and contribute to the learning of others.

In class exercises:

- Teacher guided lecture: Teacher introduces bootstrap library for CSS Styling.
<https://getbootstrap.com/docs/5.1/getting-started/introduction/#css>
 - Teacher will make note of components which will allow students to display their “hero card” as a rotating deck
 - <https://getbootstrap.com/docs/5.1/components/carousel/>
- Students are given 30 minute work time
- Partner critique: Divides into two, 1 partner stays at the computer and presents. Other partner participates in a gallery walk. After 1 round, partner positions switch. Goal is to have this part done in 20 minutes.
- Students get a final 20 minute work time for today

Notes n/a

Handouts n/a

Resources

- <https://getbootstrap.com/docs/5.1/getting-started/introduction/#css>
- <https://getbootstrap.com/docs/5.1/components/carousel/>

Assignments

Finish project

04_Feedback

Main objective

- Allow students time to work on their own projects

Standards

- 9-12.CT.4: Implement a program using a combination of student-defined and third-party functions to organize the computation.
- 9-12.CT.7: Design or remix a program that utilizes a data structure to maintain changes to related pieces of data
- 9-12.CT.9: Systematically test and refine programs using a range of test cases, based on anticipating common errors and user behavior

In class exercises

- Have students work on their projects for the first half of class. ~25 minutes
- Class discussion of constructive feedback. ~5 minutes
- Have students get ready for feedback. They will fill out “glows and grows” chart for each student they interact with. The goal is to get each student through 3 rounds of feedback. ~15 minutes
- Have students implement their feedback and work on finishing their projects. ~30 minutes
- ~5 minutes left for transition and wrap up time.

Resources:

- Continued use of any online resource for JS/HTML/CSS
 - We predominantly rely on W3 Schools, Mozilla, and Bootstrap

Assignments: Project due by next class

Notes/Handouts: (next page)

04_Feedback

The most important part of any engineering is getting feedback.

(Def) Constructive Feedback: _____

Round 1:

Glows ✨	Grows 🌱

Round 2:

Glows ✨	Grows 🌱

Round 3:

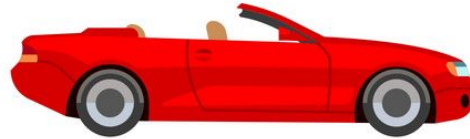
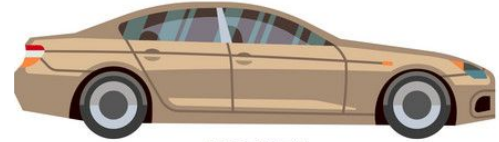
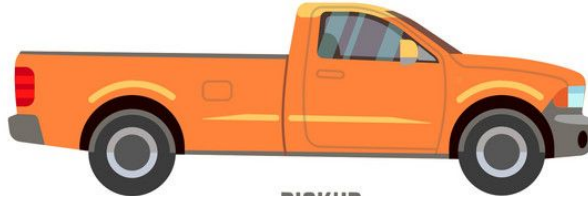
Glows ✨	Grows 🌱

00 Introduction to Objects

What is an object?

Come up
with a class
definition
for “object”

Examples 1:



Examples 2:

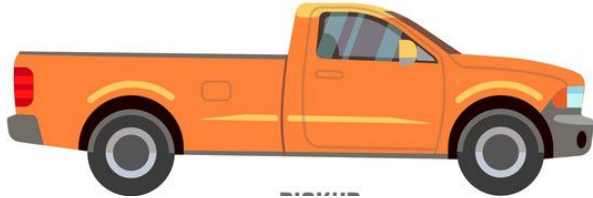
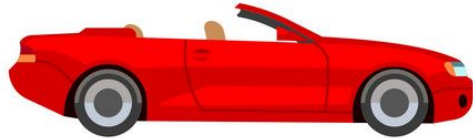


Come up with something you can classify as an object

- Your object must have many types, e.g. we have a car object but there are many variations of this car object. Come up with a minimum 3 variations, each with 3 characteristics
- Hint: Humans make classifications of objects all the time.
 - Science/animals are a great example of categorizing things
 - Do you ever look up “stats” for a specific team?

Object name	Property 1	Property 2	Property 3

What is an object in computer science? Example 1



```
//example 1
var car1 = {
  type: "sedan",
  numWheels: 4,
  color: "brown",
  seats: 5
}
var car2 = {
  type: "convertible",
  numWheels: 4,
  color: "red",
  seats: 5
}
```


What is an object in computer science? Example 2



```
//example 2
var bird1 = {
  name: "robin",
  color: ["brown","orange"],
  flight: true,
  avgWeight: 74 //mass in grams
}
var bird2 = {
  name: "cardinal",
  color: ["red"],
  flight: true,
  avgWeight: 45 //mass in grams
}
var bird3 = {
  name: "big bird",
  color: ["yellow"],
  flight: false,
  avgWeight: 127006,
}
}
```

Code along with me!
We'll be referring to this
bird example often.

Code your object example

Add another bird object to our code from the previous slide.

Great. I made an object. Now what.

```
function fly(obj){  
  if (obj.flight == true){  
    console.log("flap flap");  
  }else{  
    console.log("ground ground");  
  }  
}  
  
fly(bird2);  
fly(bird3);
```

You can also add functions to your objects. It's what makes them so useful! What do you think this will show in the console?

There are 2 ways to define functions in JS

```
var bird3 = {  
  name: "big bird",  
  color: ["yellow"],  
  flight: false,  
  avgWeight: 127006,  
  fly: function(){  
    if (this.flight == true){  
      console.log("flap flap");  
    }else{  
      console.log("ground ground");  
    }  
  }  
}
```

```
function fly(obj){  
  if (obj.flight == true){  
    console.log("flap flap");  
  }else{  
    console.log("ground ground");  
  }  
}
```

What's different about these two ways?

The way you use them is different too!

```
var bird3 = {  
  name: "big bird",  
  color: ["yellow"],  
  flight: false,  
  avgWeight: 127006,  
  fly: function(){  
    if (this.flight == true){  
      console.log("flap flap");  
    }else{  
      console.log("ground ground");  
    }  
  }  
}
```

`bird3.fly();`

```
function fly(obj){  
  if (obj.flight == true){  
    console.log("flap flap");  
  }else{  
    console.log("ground ground");  
  }  
}  
fly(bird3);
```

This is Dot Notation. We're going to stick to this version.
It's more common in other object oriented languages.
(Java)

Dot Notation

```
var bird3 = {  
  name: "big bird",  
  color: ["yellow"],  
  flight: false,  
  avgWeight: 127006,  
  fly: function(){  
    if (this.flight == true){  
      console.log("flap flap");  
    }else{  
      console.log("ground ground");  
    }  
  }  
}  
  
bird3.fly();
```

We use dot notation all the time because all of JS is built on objects. Where do you see dot notation here? Add circles.

Add another function to birds together

```
var bird3 = {  
  name: "big bird",  
  color: ["yellow"],  
  flight: false,  
  avgWeight: 127006,  
  fly: function(){  
    if (this.flight == true){  
      console.log("flap flap");  
    }else{  
      console.log("ground ground");  
    }  
  }  
}  
  
bird3.fly();
```

What else do birds do? Let's add another function.

Code your object function

Add a function that goes with your object. Try to make sure it makes “sense”.

E.g. My bird objects had a function called “fly”.

Copy and paste your code here.

HW

- Using your examples from class (slide 3), code your own objects.
- You must also have 2 functions to go with your objects.

01_GettingComfortableWith Objects

How would you define an object?

Key Vocabulary:

- Object:
- Property:
- Methods:

Examples from last class?

Arrays are objects

```
var sesameSt = ["big bird", "elmo", "cookie monster", "oscar"];
```

We've used an array attribute a lot (hint: think about forloops!)

Arrays as objects

They use dot notation.

```
var sesameSt = ["big bird", "elmo", "cookie monster", "oscar"];
```

```
sesameSt.length; //gets a property of the sesameSt object
```

```
sesameSt.push("Ernie"); //applies a method of the array object
```

```
sesameSt.splice(1,1); //applies a method of the array object
```

Predict the output of the lines above.

Objects have methods

Objects often have methods/functions associated with them.

Examples:

“Getters”: a function that helps retrieve/return a property

“Setters”: a function that helps assign a new value to a property

Javascript often doesn't need these due to us being able to directly access properties. Other languages not so much. Looking at you Java. We're still going to talk about it though to help you learn other languages.

Code-alongs

Let's create a few types of objects so we can get comfortable. For each, we'll come up with attributes and functions that go with each object type

1. Student
2. Smartphones
3. Musical Artist
4. Athlete

Object Problem Set

This problem set is designed to get you to practice more!

Object Problem Set 2

This problem set is designed to get you to practice more but this time work in HTML elements as well.

Objects Quiz

Name: _____

1. Define the following:

a. Object: _____

b. Property: _____

c. Method: _____

Use the space below to write your code.

2. Create an object `rectangularPrism` with 3 attributes of width, height, and depth. You can assign whatever values you want to this object
3. Create a “getter” method for the width.
4. Create a “setter” method for the height.
5. Create a `getVolume` method and return the object’s volume. You can use the object as a parameter.
Volume = width x height x depth
6. Create a `getSurfaceArea` method and return the object’s surface area. You must be able to use dot notation and not use the object as a parameter.
Surface area = $2(\text{width} \times \text{height}) + 2(\text{height} \times \text{depth}) + 2(\text{width} \times \text{depth})$

02_BrainstormingProject

Goal: Create an interactive trading card webpage where the user can select for a “card” using an attribute. Must have 2 tags you can sort by.

Requirements:

- Must fill out this project planning document and hand in with project
- Must implement object oriented programming for each trading card object
- Must have at least 10 cards for the user to view
- Each trading card object must have at least 3 attributes
- User must be able to see each card that they pick
- Users must be able to see all relevant cards if they sort by a certain attribute.
- Each of your objects must be an element in your array.
- Create a function that allows users to search for a specific attribute.

Project Planning Document

1. What type of trading card do you think you will implement?
2. What are the attributes?
3. What will be your main reference if you get stuck on how to code a function?
4. Jot down some questions you have here:
5. Information Gathering: Use the table below to gather information for your trading card implementation.



	Attribute 1:	Attribute 2:	Attribute 3:
Var name 1:			
Var name 2:			
Var name 3:			
Var name 4:			
Var name 5:			
Var name 6:			
Var name 7:			
Var name 8:			
Var name 9:			
Var name 10:			

04_Feedback Template



The most important part of any engineering is getting feedback.

(Def) Constructive Feedback: _____


Round 1:

Glows 	Grows 

Round 2:

Glows 	Grows 

Round 3:

Glows 	Grows 