Hyperiondev

TASK

Javascript V:

Object Oriented Programming

Visit our website

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

Introduction

Welcome to The Fifth JavaScript Task!

In this Task, you will learn about the concept of object-oriented programming and how to create JavaScript objects. This is an extremely important concept! JavaScript uses objects extensively. You will see by the end of this task, how you have actually been using JavaScript objects all along.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

Copyright © 2018 Hyperion Development. All rights reserved.

WHAT IS OBJECT-ORIENTED PROGRAMMING?

Object-oriented programming (OOP) has become the dominant programming methodology for building new systems. C++, an object-oriented version of C, was the first OO language to be used widely in the programming community. This adoption inspired most contemporary programming languages like JavaScript, C#, and Objective-C to be designed as object-oriented/object-based languages. Even languages which were not primarily designed for object-oriented programming, like PHP and Python, have evolved into supporting fully fledged object-oriented programming.

So why is OOP so popular? Firstly, we as humans think in terms of objects and designers of modern technologies have used this knowledge to their advantage. For instance, we refer to the window that appears when we power up our computer as a “desktop”. We keep shortcuts and files we use frequently on this “desktop.” But we know this is no real desk! We refer to the folder that our files get transferred to when we delete them as a “recycle bin.” But is it really a bin? We use these terms in a metaphorical sense because our minds are already familiar with these objects. Software designers leverage this familiarity to avoid unnecessary complexity by playing to the cognitive biases of our minds.

You know from your experience as a programmer, and from the previous tasks in this course, that solving computation problems can be complex. It is not just the process of developing an algorithm to solve a problem that is challenging, but the problem domain is often loaded with delicate intricacies that require expert knowledge to understand. These two problems contribute to the overall complexity of designing and implementing software that solves business problems. OOP uses our mind’s natural affinity to think in terms of objects by designing a solution in terms of objects.

This is a paradigm shift away from the procedural paradigm that is described under the next subheading.

PROCEDURAL PROGRAMMING VS OBJECT-ORIENTED PROGRAMMING

Procedural code executes in a waterfall fashion. We proceed from one statement to another, in sequence. The data are separated from the code that uses them. Functions are defined as standalone modules but they have access to variables we’ve declared outside the function. These two properties: sequential execution of code and the separation of data and the code that manipulates the data are what distinguish procedural code.

In contrast, with object-oriented programming a system is designed in terms of objects which communicate with each other to accomplish a given task. Instead of

Copyright © 2018 Hyperion Development. All rights reserved.

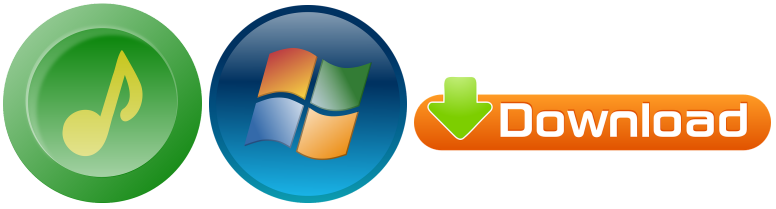
separating data and code that manipulate the data, these two are encapsulated into a single module. Data is passed from one module to the next using methods.

HOW DO WE DESIGN OO SYSTEMS?

In most OOP languages, an object is created using a **class**. A class is the blueprint from which objects are made. It consists of both data and the code that manipulates the data.

To illustrate this, let's consider an object you encounter every time you use software: a button. As shown in the images below, there are many different kinds of buttons you might interact with. All these objects have certain things in common.

1. They are described by certain **attributes**. e.g. every button has a *size*, a *background colour*, a *shape*. It could have a specific *image* on it or it could contain *text*.
2. You expect all buttons to have **methods** that do something. e.g. when you click on a button you want something to happen - you may want a file to download or an app to launch. The code that tells your computer what to do when you click on it, is written in a method. A method is just a function that is related to an object.



Although every *object* is different, you can create a single *class* that describes all objects of that kind. The class defines what **attributes** and **methods** each object created using that class contain. Each object is called an instance of a class. Each of the buttons shown in the images above is an instance of the class button.

JAVASCRIPT OBJECTS

JavaScript is an object-based programming language. If you understand objects, then you will better understand JavaScript. Almost everything in JavaScript is an object!

OOP serves to allow the components of your code to be as modular as possible. The benefits of this approach are a shorter development time and easier debugging, because you're reusing program code that has already been proven.

Copyright © 2018 Hyperion Development. All rights reserved.

Copyright © 2018 Hyperion Development. All rights reserved.

A note from our coding mentor

Ridhaa

JavaScript is not strictly an object-oriented programming language. It is an object-based scripting language. Like pure object-oriented languages, JavaScript works with objects but with JavaScript, objects are created using prototypes instead of classes. A prototype is basically a constructor function. You will learn more about this distinction later.

ES6 update: *ES6 allows us to create objects using keywords (like class, super etc) that are used in class-based languages although, under the hood, ES6 still uses functions and prototypal inheritance to implement objects. To see an example of an ES6 class, see here.*

We've covered the basic theory of what objects are and why we use them, now let's get coding and learn to create JavaScript objects.

CREATING DEFINED OBJECTS

There are more than one way of creating objects. We will consider 2 key ways in this task.

Method 1: Using Object Literal (easiest)

Let's consider the following object declaration:

```
var car = {
      brand:"Porsche",
      model:"GT3",
      year:2004,
      colour:"White",
      howOld:function(){getFullYear() - this.year;}
};
```

Here an object is declared with four different properties, which is then stored in an object variable called car. This object also contains a method called "howOld".

Copyright © 2018 Hyperion Development. All rights reserved.

- *Object Properties/Attributes:* An object comprises a collection of named values, called properties. To demonstrate this concept, we'll consider the car object:

Property	Value
Brand	Porsche
Model	GT3
Year	2004
Colour	White

- *Object Methods:* Methods (functions), are a series of actions which can be carried out on an object. An object can have a primitive value, a function, or even other objects as it's properties. Therefore an object method is simply a property which has a function as it's value. Notice that the car object above contains a method called "howOld". This object method will calculate how old the car is by checking the current year and subtracting the production year of the car. You'll notice "this.year" was used instead of the "car.year". "***this***" is a keyword, that you can use within a method to refer to the *current object*. "this" is used because it's a property within the same object, thus it simply looks for a "year" property within the object and uses that value.

This first method for creating objects that we have considered is used to create a single object at a time. What if you want to create a number of objects that all have the same properties and methods but different values? The second method that you will consider next provides an effective way of creating many objects using a single object constructor.

Method 2: Using Object Constructor

The second way of creating an object is using an object constructor. A *constructor* is a special type of function that is used to make or construct a number of different objects. A constructor function in JavaScript is basically used to implement a class. Remember that a class is the blueprint from which objects are made. It consists of both data (properties) and the code that manipulates the data (methods).

Consider the example below. The function, 'carDescription' is the constructor. It is used to create 3 different car objects: car, car2 and car3.

Copyright © 2018 Hyperion Development. All rights reserved.

```
function carDescription(brand, model, year, colour) {
  this.brand = brand;
  this.model = model;
  this.year = year;
  this.colour = colour;
};

let car = new carDescription("Porsche", "GT3", 2012, "White");
let car2 = new carDescription("Ford", "Fiesta", 2015, "Red");
let car3 = new carDescription("Opel", "Corsa", 2014, "White");
```

The dot or bracket notation can be used to modify any value in an object. For example, if you had your Porsche spray painted black, you could change the colour property of your car as shown below:

```
car.colour = "Black";
```

To delete properties of an object, you use the delete keyword:

```
delete car.colour;
```

SOME OBJECTS THAT YOU HAVE ALREADY ENCOUNTERED

As noted previously, JavaScript is an object-based language. Most of the built-in code that you have worked with so far has, therefore, used objects. For example, whenever a web page is loaded an object called document is created. This object contains methods and properties that describe and can be used to manipulate the webpage's structure and content.

When you use code such as
`let htmlSelect = document.getElementById('personList');`
you have been using the document object. Similarly, every time you create an array, you are actually creating an array object that is defined by an array class. The whole JavaScript programming languages is built using objects.

FUNCTION MEETS OBJECT

Now that you have a good understanding of functions and objects, let's consider the declaration of an object with implementation through a function (after all, you do want the object to serve a purpose):

```
var loaded = {};
```


Copyright © 2018 Hyperion Development. All rights reserved.

```
loaded.testing = function(signal) {
        alert("Hello World! " + signal);
        loaded.signal = signal;
}
loaded.testing("This page has loaded!");
```

Here a blank object is created. An object method is created with the intention of displaying a message to the user when the page has loaded. The function is called with a particular value which is then set as the value of the loaded.signal property.

Copyright © 2018 Hyperion Development. All rights reserved.

Instructions

Before you get started please be sure to examine the example files for this task.

Compulsory Task

Follow these steps:

- Create a file called 'cars.html'. You are going to create a webpage for a car salesman. The page will display details about cars that are for sale. Create whatever HTML you deem appropriate for this task and style the page as you like. However, be sure that the information about the cars for sale are displayed on the webpage programmatically (see the next point) - not hard coded with HTML.

- Create a file 'cars.js'. Within cars.js:
    - Create 5 car objects using a constructor function.
    - Each car object should have the following properties: make, model, colour, image, registration number, price.
    - Each car object should also include a showMore() method. This method should display a dialogue that displays all the details about the specific car object. Hint: See more about <dialog> here.
    - Create a function that will be used to display the make, model and image of each car object in 'cars.html' when 'cars.html' is loaded.
- Whenever a user clicks on an image of a car, the showMore() method should be called and all the information about the car, including the registration number, price etc should be displayed.

Once you have completed the task in line with the instructions above, click the button below to request your mentor to review your work and provide feedback. If you have any questions while attempting the task, leave your mentor a note on the comments.txt file in your Student Dropbox folder.

Completed the task(s)?

Ask your mentor review your work!

Copyright © 2018 Hyperion Development. All rights reserved.

Review work

Things to look out for:

1. Make sure that you have installed and setup all programs correctly. Follow the instructions in the FAQs document that accompanies this task to configure Sublime Text for debugging and executing JavaScript.

2. If you are not using Windows, please ask your mentor for alternative instructions if needed.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.

Hyperiondev

Copyright © 2018 Hyperion Development. All rights reserved.