Oregon State Parks API

Getting Set Up:

No API Key is needed to use the Oregon State Parks API, the only requirements that anything used is credited to and presented as coming from OPRD.

“Artwork, photos, images and text (“data”) stored on **oregonstateparks.org** may be subject to copyright. All use of data from this website must be credited “courtesy Oregon Parks and Recreation Department,” and must be presented as originating from OPRD.”

http://oregonstateparks.org/index.cfm?do=v.page&id=52

Make sure you don’t use the OPRD shield and wordmark without visiting the Data Terms of Use page and getting separate permission.

In order to work with the API, you will want to get this set up on a server to avoid CORS errors. All of the calls to the API will be GET requests to the following URL and are returned as a string representation of a JSON object.

http://oregonstateparks.org/data/index.cfm/

There are five ways to get data from the API:

Parks: Returns a list of all Oregon State parks and their latitude, longitude, narrative, name and id.

Park Photos: Returns all photos for a specified park in both thumbnail and full size

Park Features: Returns a list of parks and their features that can be searched based on their id or name or the features class or title. When looking at OregonStateParks.org these features show up as icons on the park’s page.

Park Events: Returns a list of events at parks that can be searched based on park, event category, event id, description, or date specifiers.

Features: Returns a list of all possible features and their classes.

Parks

To get a list of all parks a simple GET request can be sent to:

http://oregonstateparks.org/data/index.cfm/parks

If you are using node.js, you can set this up with a basic http request. The data will be returned in chunks, so make sure to concatenate those together before attempting to parse to JSON. Once you have the data stored as a JSON object, you can pull out the data you are interested in using.

var options = {

host: 'oregonstateparks.org',

path: '/data/index.cfm/parks'

};

callback = function(response) {

var string = '';

response.on('data', function (chunk) {

string += chunk;

});

response.on('end', function() {

var data = JSON.parse(string);

// Your code here

});

}

http.request(options,callback).end();

Response:

[ { "park\_latitude":44.659645,

"park\_narrative":"<p>\n\tDiggers, this park&#39;s for you! Razor clamming is a favorite activity as well as surfing. If you plan to visit prime Newport attractions like the Oregon Coast Aquarium and Hatfield Marine Science Center, you must stop in for a refreshing picnic at Agate Beach.</p>\n<p>\n\tAgate Beach Wayside is situated between the new and old highway 101. There is a large parking lot with room for trailer parking. The beach access trail consists of a tunnel that goes underneath the old highway 101 to a large sandy beach area.</p>\n",

"park\_name":"Agate Beach State Recreation Site",

"park\_id":152,

"park\_longitude":-124.056381

},

…

]

This is a good request to use if you want a list of park names and ids, since most of the other requests that can be used in the API can use this information to return more specific information.

Park Photos

In order to use this request, you will need to know the id of the park that you are interested in. I would recommend first setting up the previous request and using the list of park names and ids to be able to search for specific parks. You **must** include the park id in this request or you will not get a response.

This request needs to be sent in the following format, where ## is replaced with the id of the park you are interested in.

http://oregonstateparks.org/data/index.cfm/parkPhotos?parkId=##

I used the same format for the http request, with a slight modification to the options:

var options = {

host: 'oregonstateparks.org',

path: '/data/index.cfm/ parkPhotos?parkId=’ + parkId

};

Once parsed, the returned JSON object consists of pairs of photos in thumbnail and full size.

[  
   {  
      "thumbFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\thumbs\\Hat Rock State Park\_thumb.jpg",  
      "imageFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\Hat Rock State Park.jpg"  
   },  
   {  
      "thumbFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\thumbs\\Hat Rock\_DSC\_1186-east6\_thumb.jpg",  
      "imageFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\Hat Rock\_DSC\_1186-east6.jpg"  
   },  
   {  
      "thumbFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\thumbs\\hrpond\_thumb.jpg",  
      "imageFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\hrpond.jpg"  
   },  
   {  
      "thumbFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\thumbs\\hrupperpicnic1\_thumb.jpg",  
      "imageFile": "http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\hrupperpicnic1.jpg"  
   }  
]

In order to be able to use these links, some modification is required. As you can see, the image names have spaces in the URI. There are functions in JavaScript such as encodeURI() that can encode a URI for you, however you cannot use this function on these strings.

If you use encodeURI(), it will encode both the spaces and the forward slashes in the string:

Before

http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\hrupperpicnic1.jpg

After

http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat%20Rock%20State%20Park%5Chrupperpicnic1.jpg

However, in order to get the picture, you need to leave the forward slashes and only encode the spaces. You can do this, by finding the spaces in the string and replacing them with ‘%20’.

var string = http://oregonstateparks.org/index.cfm?do=main.loadImage&Image=Hat Rock State Park\\hrupperpicnic1.jpg

string.replace(/ /g, ’%20’);

Once that has been done, you can use those strings in your html to display the photos.

Park Features

Park features can be searched by park id, park name, icon title or icon class. Icon title and class refer to the features and the general categories. For example, **Bike Path** is the icon title and **bike** is the icon class. You can get a list of all of the possible titles and classes with the general request to features that will be discussed later. The park features request is sent to:

http://oregonstateparks.org/data/index.cfm/parkFeatures?

With the following possible parameters:

parkId=

parkName=

iconTitles=

iconClasses=

If no parameters are specified or parkId=0 is used, a list of all features of all parks is returned.

If you want to request multiple parks by name, icons by title or by class, you can do so by separating parameters with commas, but make sure there is no space between the term and the comma:

http://oregonstateparks.org/data/index.cfm/parkFeatures?parkName=Agate%20Beach%20State%20Recreation%20Site,Ainsworth%20State%20Park

http://oregonstateparks.org/data/index.cfm/parkFeatures?iconTitles=Beach%20Access,Fishing

http://oregonstateparks.org/data/index.cfm/parkFeatures?iconClasses=beach,bike

The information is returned as a collection of objects with a new object for each feature. This means that if a park has multiple features, the same data will be repeated multiple times. Make sure to take this into account when you use it:

[

{

"park\_latitude":44.659645,

"featureClass":"beach",

"park\_name":"Agate Beach State Recreation Site",

"park\_id":152,

"park\_longitude":-124.056381,

"ada":0,

"featureTitle":"Beach Access"

},

{

"park\_latitude":44.659645,

"featureClass":"fishing",

"park\_name":"Agate Beach State Recreation Site",

"park\_id":152,

"park\_longitude":-124.056381,

"ada":0,

"featureTitle":"Fishing"

},

…

]

Again, the http request can be set up in the same way as previous calls, with a modification to the options

var options = {

host: 'oregonstateparks.org',

path: '/data/index.cfm/parkFeatures?' + string

};

A nested loop in the callback can handle parsing out the repeated data and turn the results into a list of park objects that contains its features

var parks = [];

for (var i = 0; i < data.length; i++) {

var park = {};

var features = [];

park.latitude = data[i].park\_latitude;

park.longitude = data[i].park\_longitude;

park.name = data[i].park\_name;

park.id = data[i].park\_id;

park.ada = data[i].ada;

features.push({'class': data[i].featureClass, 'title': data[i].featureTitle});

while (i < data.length - 1 && data[i+1].park\_id == data[i].park\_id) {

features.push({'class': data[i+1].featureClass, 'title': data[i+1].featureTitle});

i++;

}

park.features = features;

parks.push({'park': park});

}

Events

Requests for event data are sent to:

http://oregonstateparks.org/data/index.cfm/parkEvents?

With the following possible parameters:

parkId=

categoryId=

descr=

dateFrom=

dateTo=

eventId=

Park id and event id are fairly straight forward. Below are a list of the current category ids that are used:

Category Id

0 = all

1 = General

2 = For Kids

3 = Nature

4 = State Parks Day

Multiple words can be used to search the event description and a like search is used. Each keyword must be sent as a new descr=keyword pair:

http://oregonstateparks.org/data/index.cfm/parkEvents?descr=food&descr=fun

Dates must be sent in the format MM/DD/YYYY and dateFrom returns all dates including and after the date provided, while dateTo returns all dates before and not including the date provided.

http://oregonstateparks.org/data/index.cfm/parkEvents?dateTo=12/15/2015

Multiple parameters can be sent as well:

http://oregonstateparks.org/data/index.cfm/parkEvents?parkId=103&dateTo=12/15/2015

The default search will return all events beginning the day the request is made.

[  
   {  
      "event\_description": "Come join in for an early morning run on Thanksgiving day!",  
      "park\_event\_id": 44958,  
      "park\_name": "Historic Columbia River Highway State Trail",  
      "event\_end": "November, 26 2015 12:00:00",  
      "park\_id": 113,  
      "event\_start": "November, 26 2015 08:00:00",  
      "event\_location": "West HCRH trailhead",  
      "allDay": 0,  
      "park\_event\_category\_descr": "General",  
      "title": "Turkey Trot"  
   },

…

]