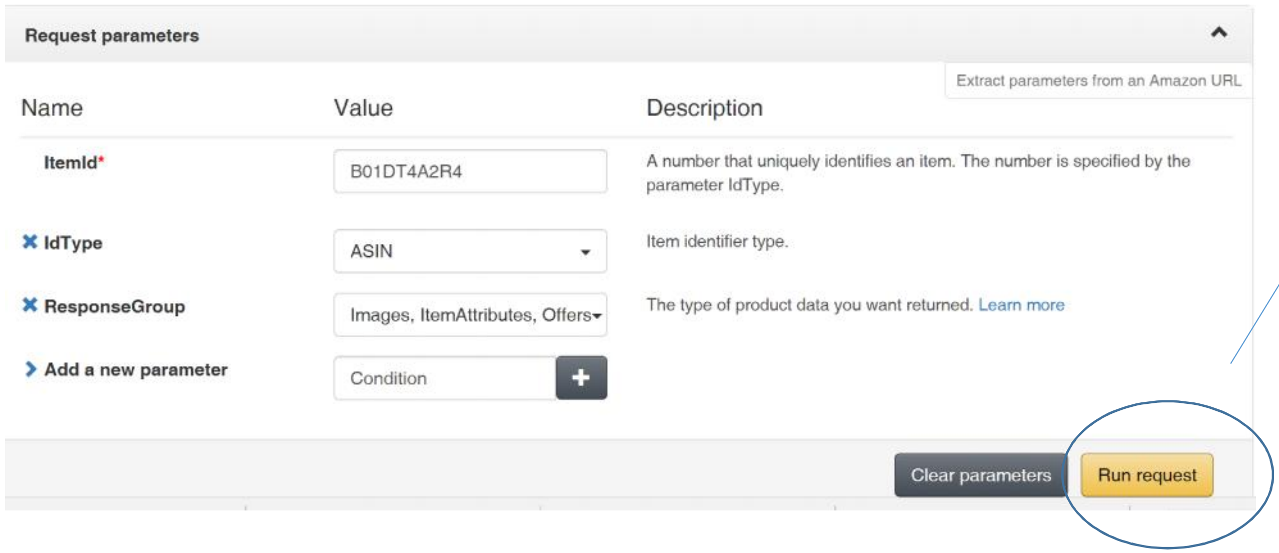
Technology Demonstration

Amazon's Product Advertising API

Associate ID: saleha-20
AWSAccessKeyId=AKIAJWKWBFZEI7MBOP3A
AWSSecretKey=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

Did test calls on Amazon's scratchpad at:
http://webservices.amazon.com/scratchpad/index.html

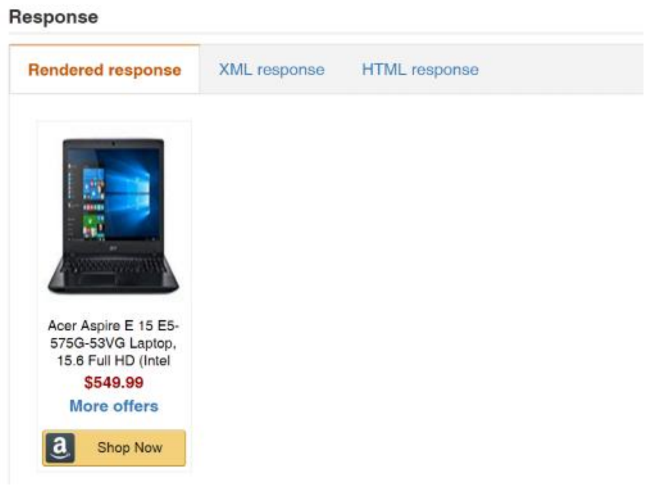


Request parameters

Extract parameters from an Amazon URL

Name	Value	Description
ItemId*	B01DT4A2R4	A number that uniquely identifies an item. The number is specified by the parameter IdType.
✖ IdType	ASIN	Item identifier type.
✖ ResponseGroup	Images, ItemAttributes, Offers	The type of product data you want returned. Learn more
› Add a new parameter	Condition	

Using API to get product details



Response

Acer Aspire E 15 E5-575G-53VG Laptop, 15.6 Full HD (Intel
$549.99
More offers

API returns reviews as iframe URL, so cannot use API to get reviews. Will need to use curl

- Have used curl previously for course catalog assignment

```
curl http://www.catalog.gatech.edu/courses-grad/ae/index.html \
http://www.catalog.gatech.edu/courses-grad/apph/index.html \
http://www.catalog.gatech.edu/courses-grad/ase/index.html >>
raw.html
```

Using minify

- // CLI command to put all HTML content in one line
- // uses "html_minifier", NPM package

```
html-minifier raw.html --collapse-whitespace --minify-js --minify-css -o
no_whitespace.html
```

Sample script that counts words (1/2)

```
<!DOCTYPE html>
<html>
<script src="http://d3js.org/d3.v3.min.js"></script>
<script src="d3.tip.js"></script>
<script src="data.js"></script>
<script src="new_graph.js"></script>
<link rel="stylesheet" href="styles-example.css">
<body>
<div id="target"></div>
</body>
<script>

// pass in html to add to page
// return element containing new HTML
function addHtmlToPage(htmlString){
document.getElementById('target').innerHTML = htmlString;
return document.getElementById('target');

}

// pass in html element containing data
// return nodelist of courses
function getCourseNodeList(tag){
var list = document.getElementsByClassName('courseblocktitle');
return list;
}

// pass in nodelist of courses
// return array of courses
function nodeListToArray(nodeList){

nodeList = Array.prototype.slice.call(nodeList);
return nodeList;
}
```

```
// pass in array of courses
// return course titles
function getTitles(list){
                var titles = list.map(function(node){
                return node.innerText;
                });

  return titles;
}


// pass in course titles
// return words
// filter out punctuation/numbers, make words array
function scrubTitles(titles){


                var words = titles.map(function(node){

                return node.toLowerCase().match(/([a-z]+)/g);
                });

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


   for (j=0; j<words[i].length; j++)
   {

                //filtering out common words
                words[i][j] =
words[i][j].replace(/(\bthe\b)|(\bof\b)|(\band\b)|(\bspecial\b)|(\bspe
c\b)(\bi\b)(\bii\b)+/g, "");
```

```
                //deleting empty strings
                if (words[i][j] ==="") {
                                words[i].splice(j,1);
                }

   }

   //removing the first element which contains an abbreviation for the
course code
   words[i].shift();
   //removing the last two elements which contain "credit" and
"hours"
   words[i].pop();
   words[i].pop();


  }

  return words;

}

function flattenArray(words){
                //flatten
                var wordsFlat = words.reduce(function(previous,
current){
                return previous.concat(current);
                });

                return wordsFlat;

}
```

Sample script that counts words (2/2)

```
// pass in the flat words array
// return word scores
// count the word frequency
function scores(wordsFlat){

                         //word scores
                        var scores =
wordsFlat.reduce(function(previous,current){
                        if (current in previous) {
                        previous[current] += 1;
                        } else {
                        previous[current] = 1;
                        }

                        return previous;
                        }, {});

                        return scores;

}



var titles2 =
getTitles(nodeListToArray(getCourseNodeList(addHtmlToPage(data))));
var titles3 = scrubTitles(titles2);
var wordsFlat1 = flattenArray(titles3);
var scores = scores(wordsFlat1);

graph();


</script>
</html>
```

d3 code (1/5)

```
function graph(){

  // clean up
  document.getElementById('target').innerHTML = '';

  // ------------- GRAPHING -------------

  //Improvement 1: improvement made - tooltip now shows not only word but also score of the word
  var tip = d3.tip()
    .attr('class', 'd3-tip')
    .html(function(d) { return '<span>' + d.word + ', ' + scores[d.word] + '</span>' ;})
    .offset([-12, 0]);

  var padding = 6,
      radius = d3.scale.log().range([15, 70]).domain([2, 82]),
      color = d3.scale.category10().domain([0, 15]);

  var nodes = [];
  var circle = [];
  var force;
```

d3 code (2/5)

```
var svg = d3.select("div[id=target]").append("svg")
    .attr("width", 1920)
    .attr("height", 960)
    .attr("class", "vis")
  .append("g");

svg.call(tip);

//Improvement 2: changed the graphing logic so that colors depend on word counts, not word lengths
for (var word in scores) {
  nodes.push({radius: radius(scores[word]), color: color(scores[word]), word: word, score: scores[word]});
}

force = d3.layout.force()
  .nodes(nodes)
  .size([1024, 768])
  .gravity(0.01)
  .charge(-0.01)
  .on("tick", tick)
  .start();
```

d3 code (3/5)

```
circle = svg.selectAll("circle")
  .data(nodes)
  .enter().append("circle")
  .attr("r", function(d) { return d.radius; })
  .style("fill", function(d) { return d.color; })
  .on('mouseover', tip.show)
  .on('mouseout', tip.hide)
  .call(force.drag);

function tick(e) {
  circle
      .each(cluster(10 * e.alpha * e.alpha))
      .each(collide(.5))
      .attr("cx", function(d) { return d.x; })
      .attr("cy", function(d) { return d.y; });
}
```

d3 code (4/5)

```
// Move d to be adjacent to the cluster node.
   function cluster(alpha) {
     var max = {};

     // Find the largest node for each cluster.
     nodes.forEach(function(d) {
       if (!(d.color in max) || (d.radius >
max[d.color].radius)) {
         max[d.color] = d;
       }
     });

     return function(d) {
       var node = max[d.color],
           l,
           r,
           x,
           y,
           i = -1;

       if (node == d) return;
```

```
        x = d.x - node.x;
        y = d.y - node.y;
        l = Math.sqrt(x * x + y * y);
        r = d.radius + node.radius;
        if (l != r) {
          l = (l - r) / l * alpha;
          d.x -= x *= l;
          d.y -= y *= l;
          node.x += x;
          node.y += y;
        }
      };
    }

    // Resolves collisions between d and all other circles.
    function collide(alpha) {
      var quadtree = d3.geom.quadtree(nodes);
      return function(d) {
        var r = d.radius + radius.domain()[1] + padding,
            nx1 = d.x - r,
            nx2 = d.x + r,
            ny1 = d.y - r,
```

d3 code (5/5)

```
   // Resolves collisions between d and all other circles.
   function collide(alpha) {
    var quadtree = d3.geom.quadtree(nodes);
    return function(d) {
      var r = d.radius + radius.domain()[1] + padding,
          nx1 = d.x - r,
          nx2 = d.x + r,
          ny1 = d.y - r,
          ny2 = d.y + r;
      quadtree.visit(function(quad, x1, y1, x2, y2) {
        if (quad.point && (quad.point !== d)) {
          var x = d.x - quad.point.x,
              y = d.y - quad.point.y,
              l = Math.sqrt(x * x + y * y),
              r = d.radius + quad.point.radius + (d.color !==
quad.point.color) * padding;
```

```
if (l < r) {
              l = (l - r) / l * alpha;
              d.x -= x *= l;
              d.y -= y *= l;
              quad.point.x += x;
              quad.point.y += y;
            }
          }
          return x1 > nx2
              || x2 < nx1
              || y1 > ny2
              || y2 < ny1;
        });
      };
    }

}
```