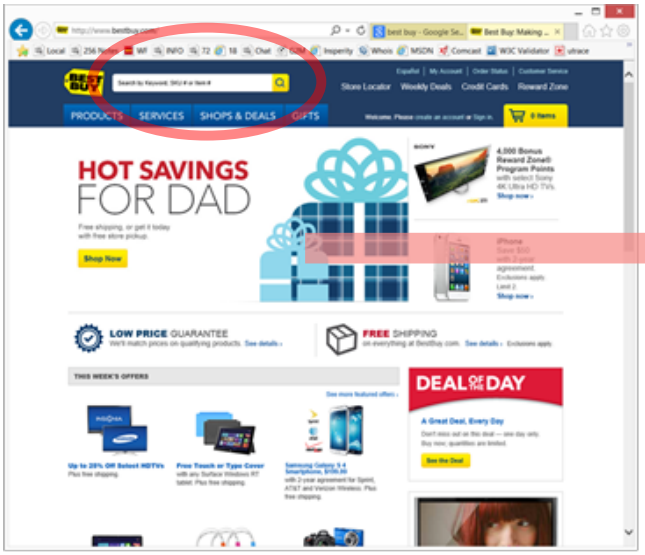
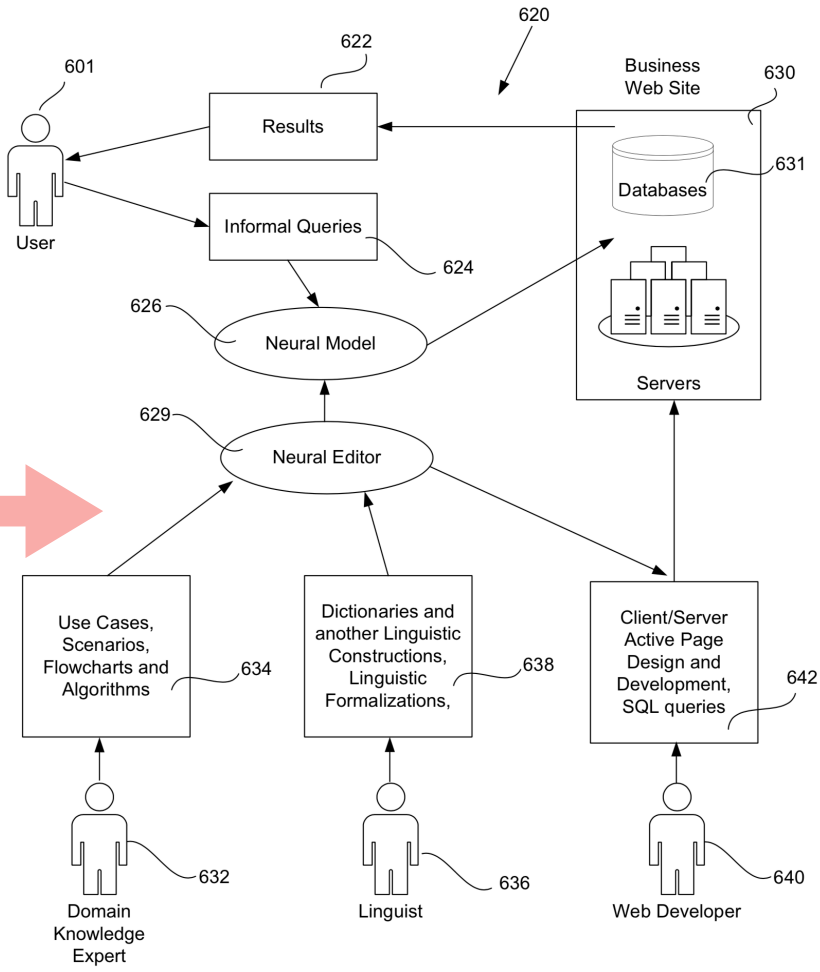
ChatBot in 7 Days

How to use Neural Networks to create Chatbot in a week

Development Diary: February 22, 2018 - February 28, 2018

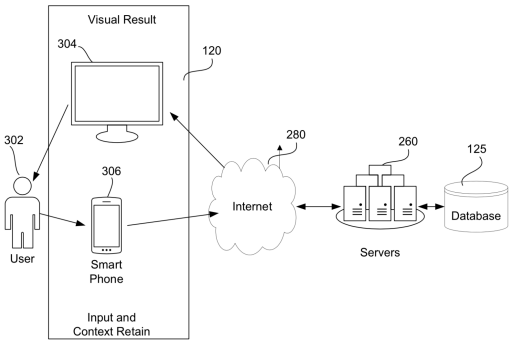
Best Buy

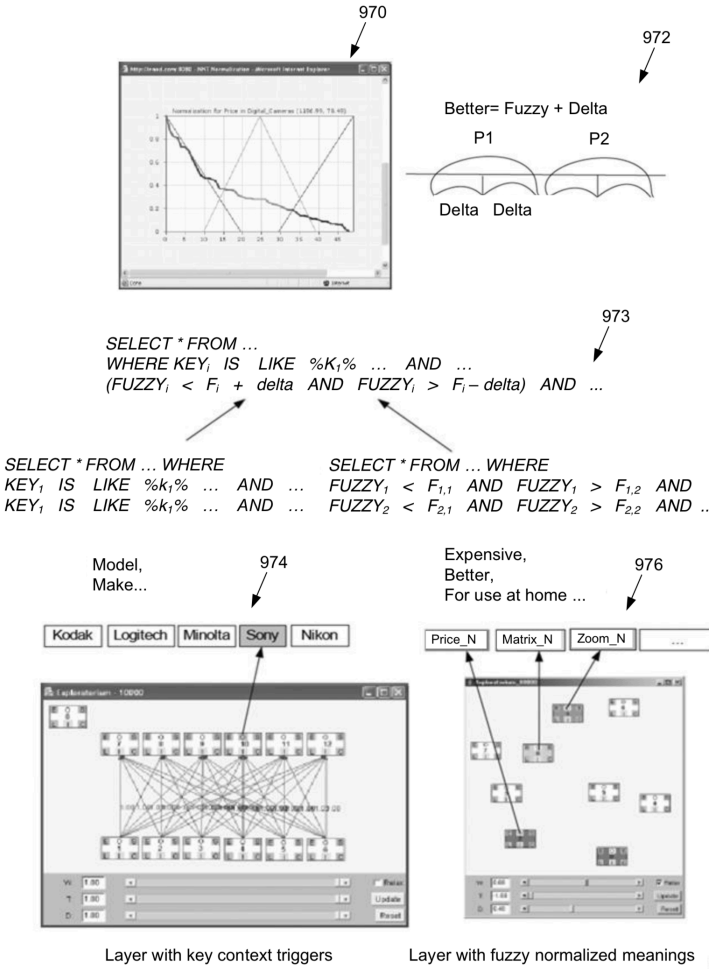




Introduction

In this project we will create a ChatBot or Virtual Assistant, capable to run in multi channel environment (chat, voice, PC, smartphone, TV) by using neural networks NNOD SDK and very basic development resources. The goal is to show how easy is the process of building interactive contextual search Assistant when using Neural Networks based Expert System. As an example we will use BestBuy website and its Development API. The process will takes one man-week, and I will keep the diary with daily report.





Day 1

Project preparation. Agile vs Waterfall. Data Source.

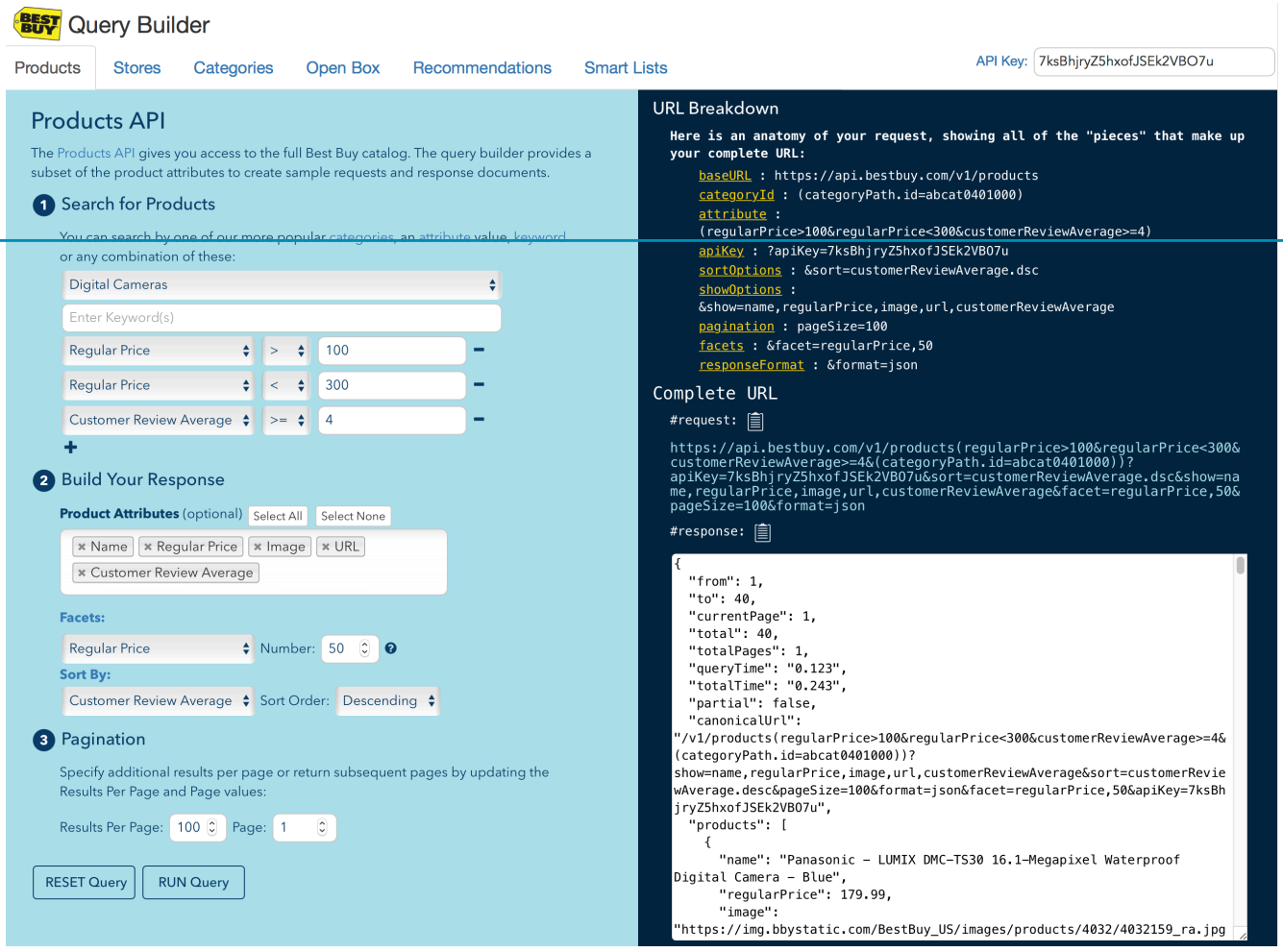
Before to start any projects, somehow, the decision about planning and final goal must be made. But I will be the only one working on it during the next 7 days, so all traditional function: management, architectural design, UI, front and back programming, QA and few others, will be implemented inside the one man's mind. Personally, I prefer the Waterfall model, but considering that I will have minimum possible resources allocated to this project, I've decided to use Agile as my personal method to communicate. Which means that "virtual manager", "virtual architect", etc. will "virtually" interact in my mind representing the model of real team cooperation.

Learning Best Buy API.

The very first step - registration on BestBuy Development site and obtaining API key. To do this we registered on https://developer.bestbuy.com and feel application form to obtain API Key. This Key required to send requests to BestBuys database.

The next step is to lear and experiment with BestBuy API (BB API). It takes some time to understand the structure, component, formats of API. The most important for us is Query Builder - http://bestbuyapis.github.io/bby-query-builder/#/productSearch Most of the time we spent to understand the structure of JSDON queries and responds.

For example, informal request - “Show me cameras around $200 with the best reviews” can be reformulated in formal request:



At the same time, on the background, we start architectural design and preparation for presentation of responses in more dynamic visual form.

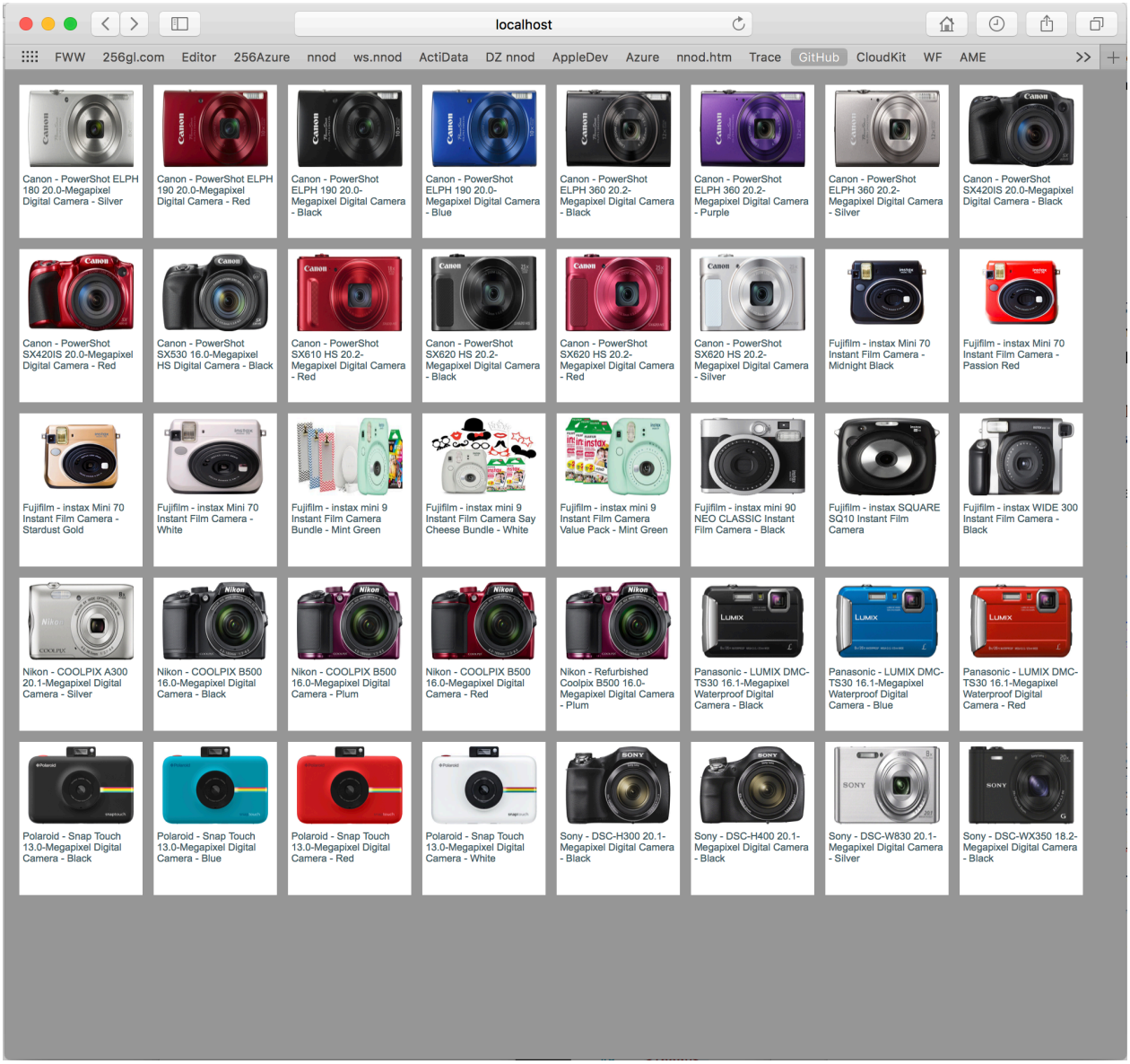
To make this solution generic as possible, we will use basic Web Server and will build interactive chat page using only standard JavaScript.

Our Neural Network API published as Open Source under GNU license in GitHub, and we will add this solution into the same GitHub depository.

Local Development Environment.

The whole process will be made in Mac OS but there are no any differences in the implementation for Windows. We will use Apache Web Server on Mac and IIS Web Server on Windows.

To make the very first call to BB API we can use PHP:



```
<body style="background-color:#999999;">
<?php
     $url = "https://api.bestbuy.com/v1/
     products(customerReviewAverage%3E=4&regularPrice%3E150&regularPrice%3C250&(ca
     tegoryPath.id=abcat0401000))?
     apiKey=xxxxxx&sort=name.asc&show=manufacturer,name,image,url,regularPrice&fac
     et=regularPrice,50&pageSize=100&format=json";
     $data = file_get_contents($url);
     $json = json_decode($data, true);
     $array = $json["products"];
     $db_counter = 1;
     foreach ($array as $obj) {
         $db_counter = $db_counter + 1;
         echo "<div class=\"thumbnail_wrapper\"><table cellspacing=\"0\"
     cellpadding=\"0\"><tr>\n" .
         "<td id=\"td_thumbnail_" . $db_counter . "\"
     class=\"td_thumbnail_image\">\n" .
         "<a href=\"#\"><img src=\"" . $obj['image']  . "\" height='70' width='95'
     " .
         "\" onclick=\"javascript:openThumbnailLink('" . $obj['url'] . "'); return
     false;\"></a></td>\n" .
         "</tr><tr><td id=\"td_thumbtitle_" . db_counter . "\"
     class=\"td_thumbnail_title\">\n" . " " . $obj['name'] . "\n" .
         "</td></tr></table></div> \n\n";
     }
     ?>
</body>
```

The result:

Day 1 conclusion.

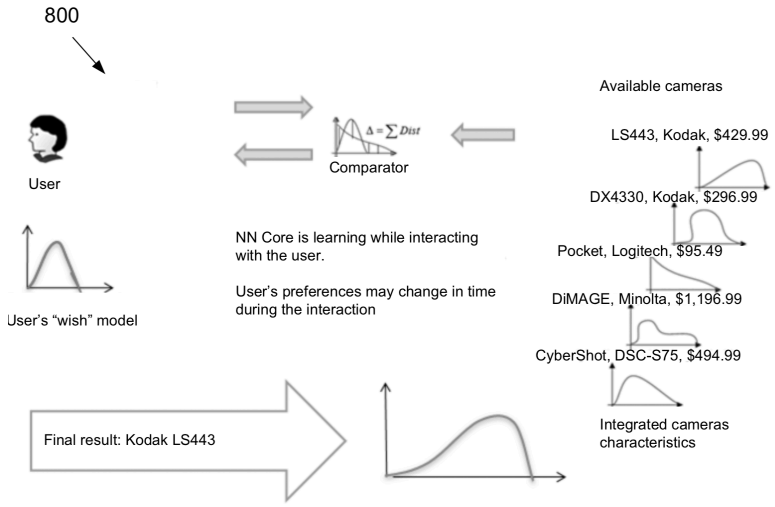
BB API Key obtained and we have full access to BB DataBase. The very first Web Page with implementation of requests to BB API built. Clear understanding of BB API now allow to start building Linguistic model for interactive search.

Day 2

Architectural/Marketing.

Value - one of the most critical questions needed to be answered before and during any development. What is the criteria we should put to measure the project advantages from similar? In our case we will use Cost of service and User's Information Demand satisfaction.

When User get an idea about buying something, he/she need to gather some information about the product. And before the very last moment when sale completed, the information for decision making will be gathered from many sources. Web, personal notes, social networks and finally expert are the network of participants in interactive process of finding the best possible item from many available in variety of sources.



Basically the process includes collecting data from different sources, compare result and repeat the process until the threshold level for the decision reached.

The cost of this process has two components: User's cost, which includes the time and potential difference if the item is wrong, or the same item can be purchased cheaper. And sale's coat, which includes the cost of consulting, and potential cost of loosing the client if he/she decided to switch to different store because of whatever reason.

We will use Best Buy Web site as one of the sources for measuring success. Criteria #1 will be based on the time required for the user to find enough information about the product and make decision to move forward and visit Best Buy or to switch to another retailer.

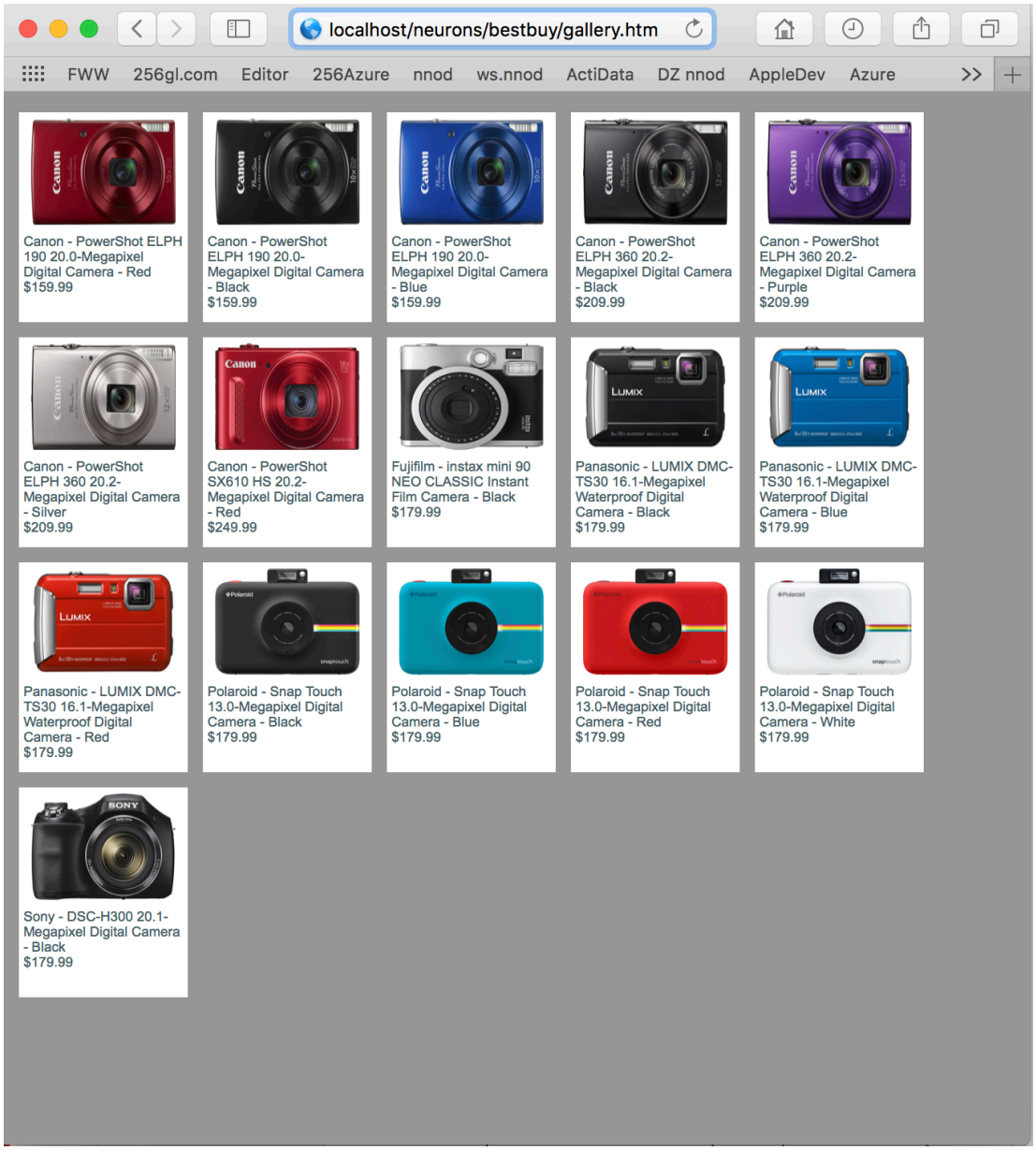
PHP -> JavaScript.

Even PHP is very popular and widely supported language, wherever it is possible we will try to use JavaScript for the reason of minimization of support and scalability. If our solution will be fully JavaScript based and will not require any external server support, but only BB API, that's mean we automatically solve the problem of scalability.

Changing PHP to JavaScript is relatively easy:

```
<!DOCTYPE HTML>
<html>
<head>
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
    <meta name="apple-mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style" content="black" />
    <meta name="viewport" content="minimum-scale=1.0, maximum-scale=5.0, user-
scalable=yes, initial-scale=1.0" />
    <meta name="viewport" content="user-scalable=yes">

    <title>256gl NNOD</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
    <script type="text/javascript">
        function openThumbnailLink(url) {
            window.open(url, "_blank", "toolbar=no, scrollbars=yes, resizable=yes,
top=10, left=10, width=800, height=800");
            return true;
        };
    </script>
</head>
<body style="background-color:#999999;">
    <div id="demo"></div>
    <script>
        var xmlhttp = new XMLHttpRequest();
        xmlhttp.onreadystatechange = function() {
            if (this.readyState == 4 && this.status == 200) {
                var myObj = JSON.parse(this.responseText);
                var result = myObj.products;
                var x = "";
                for (i in result) {
                    x = x + "<div class=\"thumbnail_wrapper\"><table
cellspacing=\"0\" cellpadding=\"0\"><tr>\n" +
                        "<td id=\"td_thumbnail_" + i + "\"
class=\"td_thumbnail_image\">\n" +
                        "<a href=\"#\"><img src=\"" + result[i].image + "\"
height='70' width='95' " +
                        "\" onclick=\"javascript:openThumbnailLink('" +
result[i].url + "'); return false;\"></a></td>\n" +
                        "</tr><tr><td id=\"td_thumbtitle_" + i + "\"
class=\"td_thumbnail_title\">\n" + " " + result[i].name + "<br>$" +
result[i].regularPrice + "\n" +
```

```
                    "</td></tr></table></div> \n\n";
                }
                document.getElementById("demo").innerHTML = x;
            }
        };
        url = "https://api.bestbuy.com/v1/
products(customerReviewAverage%3E=4&regularPrice%3E150&regularPrice%3C250&(categor
yPath.id=abcat0401000))?
apiKey=xxxxxx&sort=name.asc&show=manufacturer,name,image,url,regularPrice&facet=re
gularPrice,50&pageSize=100&format=json";
        xmlhttp.open("GET", url, true);
        xmlhttp.send();
    </script>
</body>
</html>
```



Parameters in HTTP Request

BB API allows to send request only with certain relatively restricted set of parameters. Currently we can only request products defined by: Category, CustomerReview, Manufacturer, Color and price. The formal request to BB Database then can be build only with variety of these parameters.

We will use basic HTTP Get method:

http://localhost/neurons/bestbuy/gallery.htm?color=silver&manufacturer=apple&customerReviewAverage=3&price=1500

Our request page (gallery.htm) have basic arguments parser and few logical adjustments.

It is almost impossible in informal request define price as an exact number, so we will convert the price into the range and if no exact criteria provided, the price for request will be calculated as:

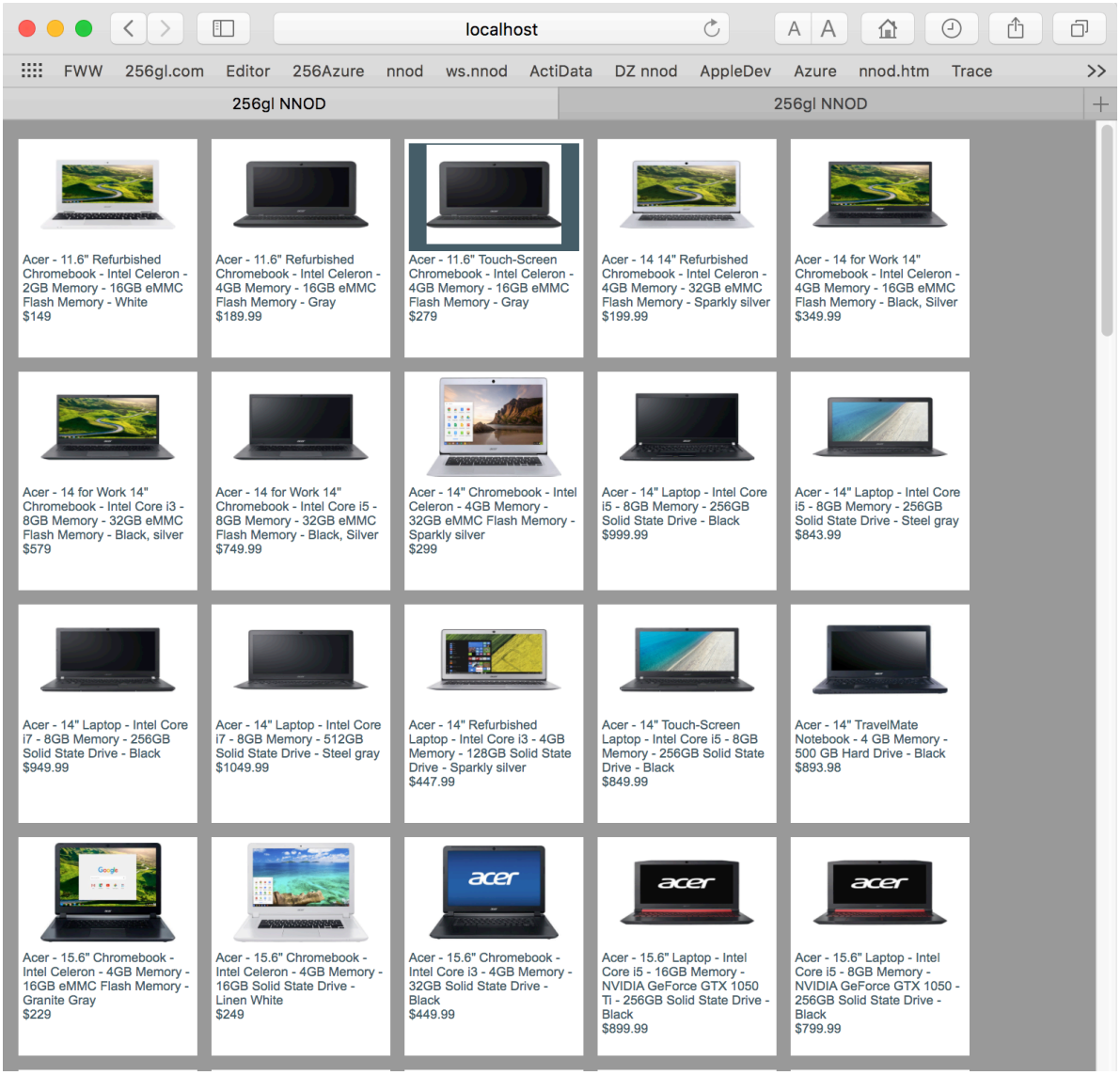
```
regularPrice_gt = price - Math.round(price / 4)
regularPrice_lt = price + Math.round(price / 4)

price >= regularPrice_gt && price <= regularPrice_lt
```

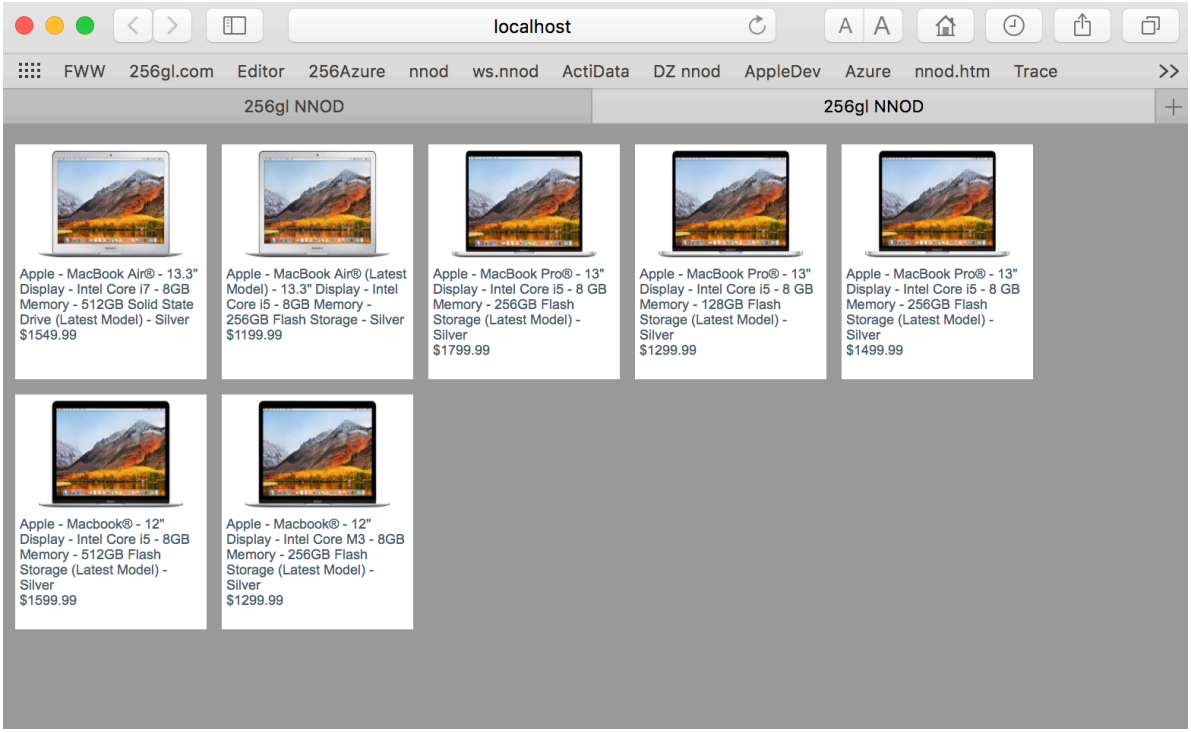
The whole arguments parsing for the page is following:

```
var api_key = "7ksBhjryZ5hxofJSEk2VBO7u";
var baseURL = "https://api.bestbuy.com/v1/products";
var categoryId = get_arguments["categoryId"];
if (categoryId === undefined)  categoryId = "abcat0502000";

//        abcat0502000 - Laptops
//        abcat0501000 - Computers
//        abcat0401000 - Cameras
//        pcmcat209400050001 - phones
//        abcat0204000 - Hesad phones
//        pcmcat241600050001 - Home Audio
//        pcmcat254000050002 - Home Automation
//        pcmcat209000050006 - iPads
//        abcat0904000 - Ovens
//        abcat0901000 - Refrigirators
//        abcat0101000 - TVs
//        abcat0910000 - Washers
//        pcmcat310200050004 -Speakers

var customerReviewAverage = get_arguments["customerReviewAverage"];
```

```
if (customerReviewAverage === undefined)
{
      customerReviewAverage = "";
}else
{
      customerReviewAverage = customerReviewAverage - 0.1
      customerReviewAverage = "&customerReviewAverage>=" + customerReviewAverage;
}
var manufacturer = get_arguments["manufacturer"];
if (manufacturer === undefined)
{
      manufacturer = "";
}else
{
      manufacturer = "&manufacturer=" + manufacturer;
}
var color = get_arguments["color"];
if (color === undefined)
{
      color = "";
}else
{
      color = "&color=" + color;
}
var regularPrice_gt = 0;
var regularPrice_lt = 100000;
var price = get_arguments["price"];
if (price === undefined || price === "")
{
      price = "regularPrice>=" + regularPrice_gt +
            "&regularPrice<=" + regularPrice_lt;
}else
{
      regularPrice_gt = Math.round(price) - Math.round(price / 4);
      regularPrice_lt = Math.round(price) + Math.round(price / 4);
      price = "regularPrice>=" + regularPrice_gt +
            "&regularPrice<=" + regularPrice_lt;

}
var url = baseURL + "(" + price + customerReviewAverage + manufacturer + color +
"&(categoryPath.id=" + categoryId + "))" +
"?apiKey=" + api_key +
"&sort=name.asc&show=manufacturer,name,image,url,regularPrice&facet=regularPrice,
50&pageSize=100&format=json";
```

With this update we can now request different category of products and update search criteria.


localhost
FWW 256gl.com Editor 256Azure nnod ws.nnod ActiData DZ nnod AppleDev Azure nnod.htm Trace
256gl NNOD
256gl NNOD
Acer - 11.6" Refurbished Chromebook - Intel Celeron - 2GB Memory - 16GB eMMC Flash Memory - White $149
Acer - 11.6" Refurbished Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Gray $189.99
Acer - 11.6" Touch-Screen Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Gray $279
Acer - 14 14" Refurbished Chromebook - Intel Celeron - 4GB Memory - 32GB eMMC Flash Memory - Sparkly silver $199.99
Acer - 14 for Work 14" Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Black, Silver $349.99
Acer - 14 for Work 14" Chromebook - Intel Core i3 - 8GB Memory - 32GB eMMC Flash Memory - Black, silver $579
Acer - 14 for Work 14" Chromebook - Intel Core i5 - 8GB Memory - 32GB eMMC Flash Memory - Black, Silver $749.99
Acer - 14" Chromebook - Intel Celeron - 4GB Memory - 32GB eMMC Flash Memory - Sparkly silver $299
Acer - 14" Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Black $999.99
Acer - 14" Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Steel gray $843.99
Acer - 14" Laptop - Intel Core i7 - 8GB Memory - 256GB Solid State Drive - Black $949.99
Acer - 14" Laptop - Intel Core i7 - 8GB Memory - 512GB Solid State Drive - Steel gray $1049.99
Acer - 14" Refurbished Laptop - Intel Core i3 - 4GB Memory - 128GB Solid State Drive - Sparkly silver $447.99
Acer - 14" Touch-Screen Laptop - Intel Core i5 - 8GB Memory - 256GB Solid State Drive - Black $849.99
Acer - 14" TravelMate Notebook - 4 GB Memory - 500 GB Hard Drive - Black $893.98
Acer - 15.6" Chromebook - Intel Celeron - 4GB Memory - 16GB eMMC Flash Memory - Granite Gray $229
Acer - 15.6" Chromebook - Intel Celeron - 4GB Memory - 16GB Solid State Drive - Linen White $249
Acer - 15.6" Chromebook - Intel Core i3 - 4GB Memory - 32GB Solid State Drive - Black $449.99
Acer - 15.6" Laptop - Intel Core i5 - 16GB Memory - NVIDIA GeForce GTX 1050 Ti - 256GB Solid State Drive - Black $899.99
Acer - 15.6" Laptop - Intel Core i5 - 8GB Memory - NVIDIA GeForce GTX 1050 - 256GB Solid State Drive - Black $799.99

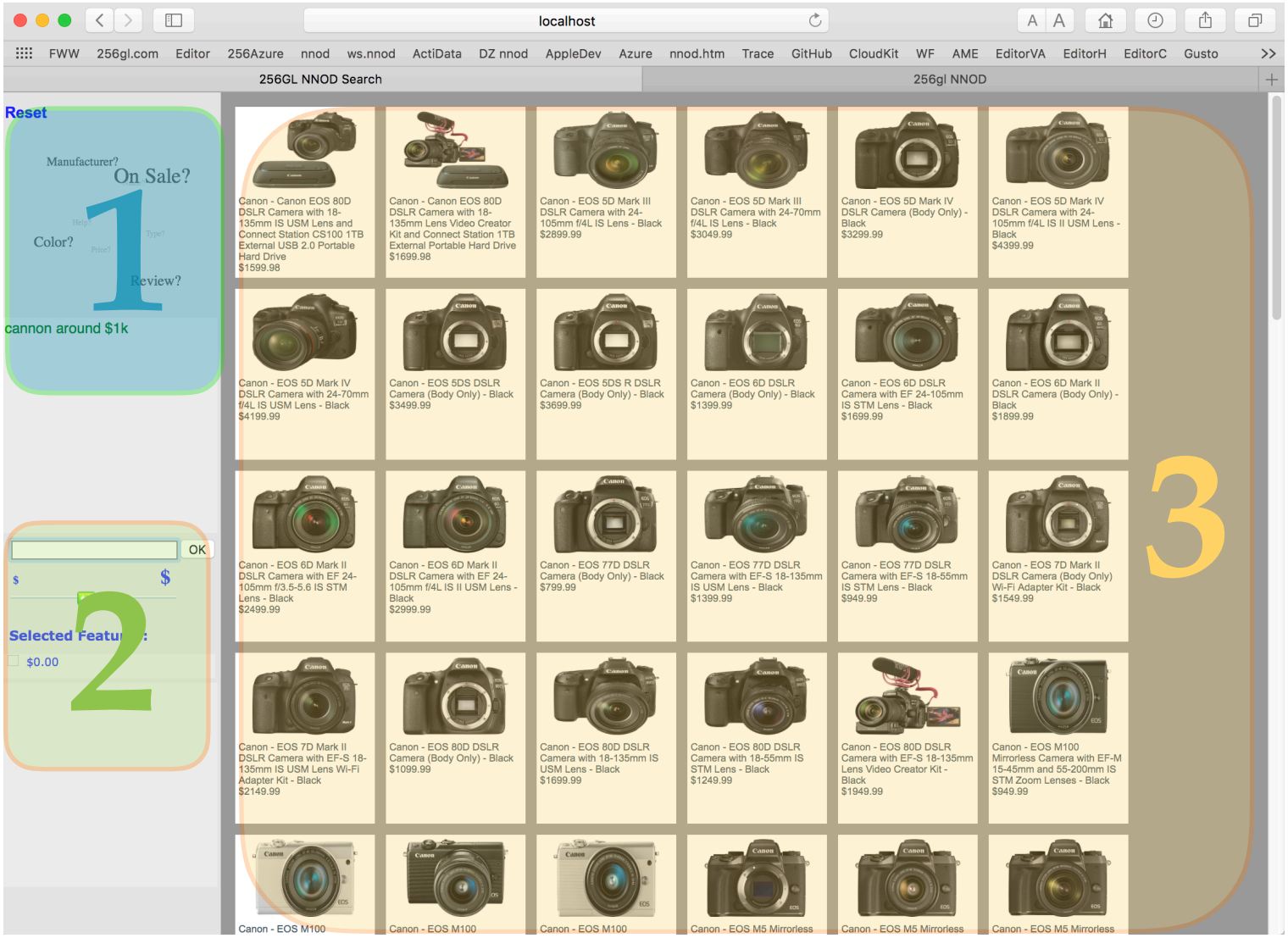

localhost
FWW 256gl.com Editor 256Azure nnod ws.nnod ActiData DZ nnod AppleDev Azure nnod.htm Trace
256gl NNOD
256gl NNOD
Apple - MacBook Air® - 13.3" Display - Intel Core i7 - 8GB Memory - 512GB Solid State Drive (Latest Model) - Silver $1549.99
Apple - MacBook Air® (Latest Model) - 13.3" Display - Intel Core i5 - 8GB Memory - 256GB Flash Storage - Silver $1199.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 256GB Flash Storage (Latest Model) - Silver $1799.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 128GB Flash Storage (Latest Model) - Silver $1299.99
Apple - MacBook Pro® - 13" Display - Intel Core i5 - 8 GB Memory - 256GB Flash Storage (Latest Model) - Silver $1499.99
Apple - Macbook® - 12" Display - Intel Core i5 - 8GB Memory - 512GB Flash Storage (Latest Model) - Silver $1599.99
Apple - Macbook® - 12" Display - Intel Core M3 - 8GB Memory - 256GB Flash Storage (Latest Model) - Silver $1299.99

Day 3

Domain of Definition, Map Informal Requests into Formal Queries

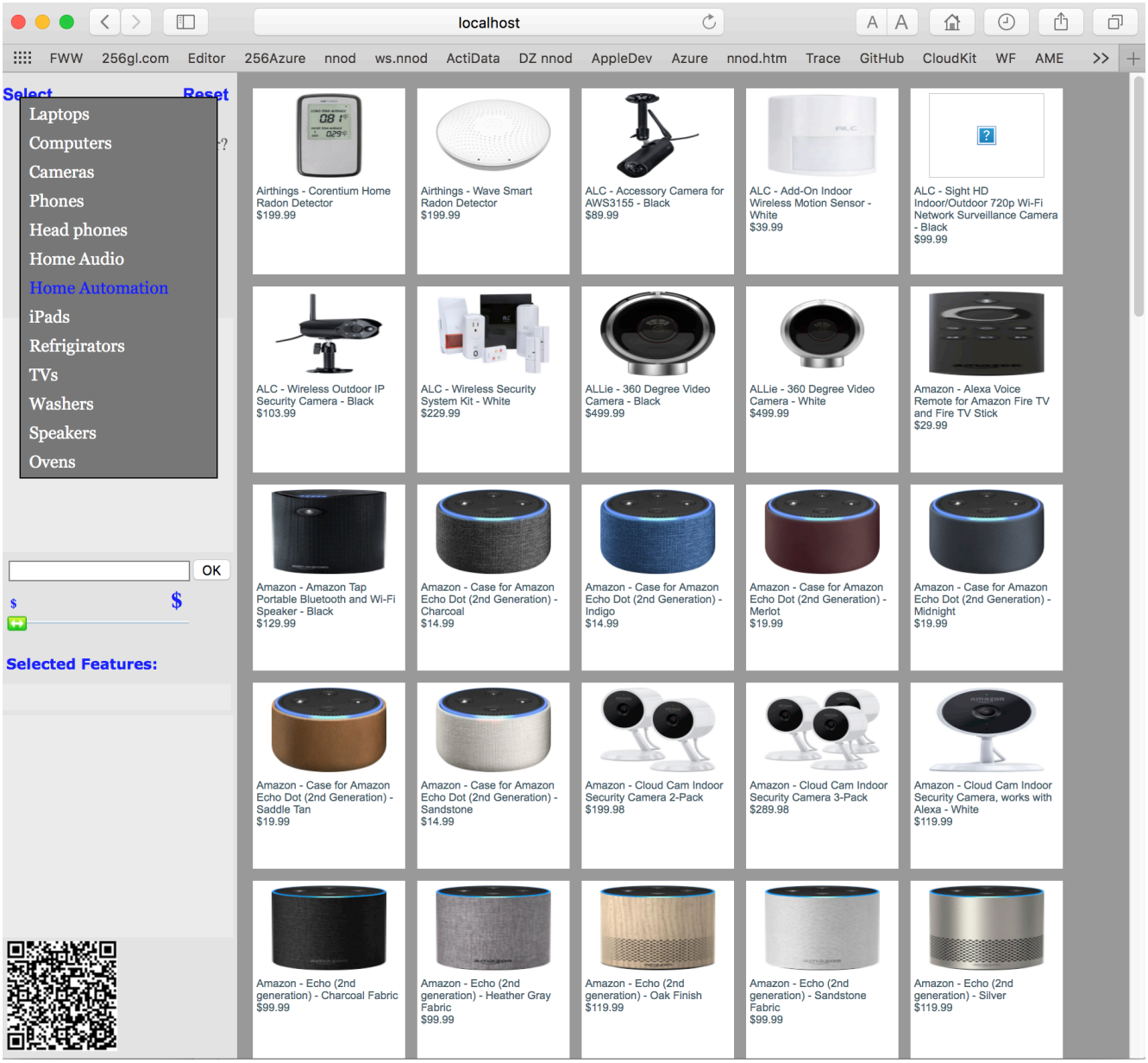
One of the most important step in ChatBot design is clear understanding of "Domain of a Function". Informal human request will be mapped into the formal query. The goal of our design is to map practically endless variety of informal request into the limited set of allowed parameters in query, which will result with limited number of items from the DataBase. In our case, Best Buy API limited by **Product, Stores, Categories** and **Recommendation** types of queries. Understanding of this limitation makes development significantly simpler and formal.

We will limit this basic ChatBot with **Product** only search but all possibilities to extend this solution with minimum modifications in the code could be dome in next version.



It is important for ChatBot not to pretend, but instead show its limit and capacity for user to avoid confusion. For this reason we combined the "content" layer together with control layer.

The new page combined three components: 1 - show available parameters for search, 2 - dialog and state of current interaction, 3 - content.



Day 4

Architecture, Integration and Multi Channel Chat

One of the goal of this project is to build not a single ChatBot, but smart communication channel, capable to interact with various data sources (in our case BB API) using different devices: PC with mouse and possible touch screen, smart phones, TV, smart home devices like Alexa. Various scenarios includes comprehensive interactive search from PC, relaxed shopping in front of TV, making shopping notes using Alexa or iPhone, and helping buyer inside the store with navigation and conversation with human expert: