



Scandcenter			
668	1240	572	
Cocupled	Danacity	Free	

Bruuns			
159 Occupied	953 Beneaty	794	

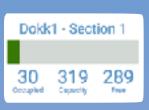








Busgadehuset			
66 000,045	105 Capacity	39	





Dashboard for parking garages in Aarhus

In this assignment you will populate a dashboard with live data from the Open Data Aarhus initiative. You will be working with Php to access the data, parsing the data and preparing it for output.

What you will be working with:

- Using a php class to load an external data set
- Php dates
- Php objects
- Php arrays
- Control structures
- JSON

Step 1 - Get the data

Get the data from Open Data Aarhus with Php.

1) Look at the markup

Open "index.php". We will be working on this file.

Discuss the markup for each card. How is the graph built? How is it set up in the CSS? What happens when you change the width of the .indicator element? What does the following classes on .indicator do:

- .low
- .med
- .high

2) Use the class "Request" to load data

Open "/class_request/example.php".

Discuss what is going on here.

In your index file, use this class to load the following JSON file:

```
http://www.odaa.dk/api/action/datastore_search?resource_id=2a82a145-0195-4081-a13c-b0e587e9b89c
```

This is the data stream from Open Data Aarhus that is updated every 5 minutes with information about how many cars have entered/exited from parking garages in Aarhus.

3) Decode the JSON

Notice go the data received is returned as a text string. Discuss why that is. To decode the JSON from this string, we can use the built-in php method json decode (). This will return an object from the string.

Discuss what happened with the string.

Step 2 - Parse the data

Once the data has been received, it has to be parsed, meaning that we need to prepare the data for how we want to use it.

1) Get the garage name from array

Look at the data we receive. Use var_dump() or similar to output it.
The data we receive contains a set of records with the registered data. Notice that this does not contain the name of the garage, only a garageCode.

Load in the array in **garageNames.php** in your index file.

With a foreach loop, echo out the garage names,

```
foreach ($garageNames as $garage) {
     $garageName = $garage["garageName"];
     $garageCode = $garage["garageCode"];
     echo "The code ".$garageCode." goes with the name ".$garageName;
}
```

2) Loop through the result data

The result data is an object and we can use the following code to access it.

```
// loop through all of the result records from the live data
foreach ($data->result->records as $record) {
         $recordCode = $record->garageCode;
         echo "This record has the garageCode: ".$recordCode;
}
```

3) Combine the two in a new array

We want to combine them so that each record is name.

```
// create new array to store the combined data
$dataList = array();

The array has to be structured so each record has the following:
$dataList[] = array(
        "name"=>$garageName,
        "data"=>$record,
);
```

You should get something similar to this var_dump(): array(2) { ["name"]=> string(10) "Nørreport" ["data"]=> object(stdClass)#9 (5) { ["date"]=> string(19) "2017/03/03 07:45:06" ["vehicleCount"]=> int(28) ["_id"]=> int(1)["totalSpaces"]=> int(65) ["garageCode"]=> string(9) "NORREPORT"

4) Reformat the timestamp

}

The format of the timestamp is not the standard format for how databases handles timestamps, so we need to reformat the timestamp.

Using the php data object we can create a new instance and format it the way we want it.

Inside your records loop, add the following to reformat the timestamp:

```
$date = date_create($record->date);
$timestamp = date_format($date,"Y-m-d H:i:s");
$record->date = $timestamp;
```

Look up and discuss what the "Y-m-d H:i:s" part mean (http://php.net/manual/en/function.date.php).

The result should be that the time stamp is changed from this format:

```
"2017/03/03 07:45:06" to this:
"2017-03-03 07:45:06"
```

Hint: If you get warnings regarding the timezone, set it explicitly by using: date_default_timezone_set("Europe/Copenhagen");

Step 3 - Update the graph

Now that we have the data in the \$dataList array we can use it to populate our HTML.

1) Loop through \$dataList and populate the <article> structure for each card.

Some code snippets to help you:

```
foreach ($dataList as $record) {
}
<?php echo $record["name"];?>
<?php echo $record["data"]->vehicleCount;?>
```

Hint: To get the information for the "Free" stat, calculate it with the info you already have.

2) Adjust size of graph

Firstly, calculate an \$occupancy rate that is a percentage of free space.

Use the round () method to round of the number to an integer. (http://php.net/manual/en/function.round.php)

Adjust the width of the graph by creating an inline style with the \$occupancy number as a percentage width.

```
style="width:<?php echo $occupancy;?>%"
```

Discuss how this changes the graph

3) Add color levels

Based upon the \$occupancy value, determine which level each graph belongs to:

- Above 75% it should be "high"
- Above 50% it should be "med"
- Flse it should be "low"

Hint: Use an if/elseif/else statement.

Assign the above levels as classes to the div.indicator.