Informationsvisualisierung & Visual Analytics

Tutorial

Michael Krone

Today’s outline

- New web server: Flask

- Starting a new visualization application (with Flask)

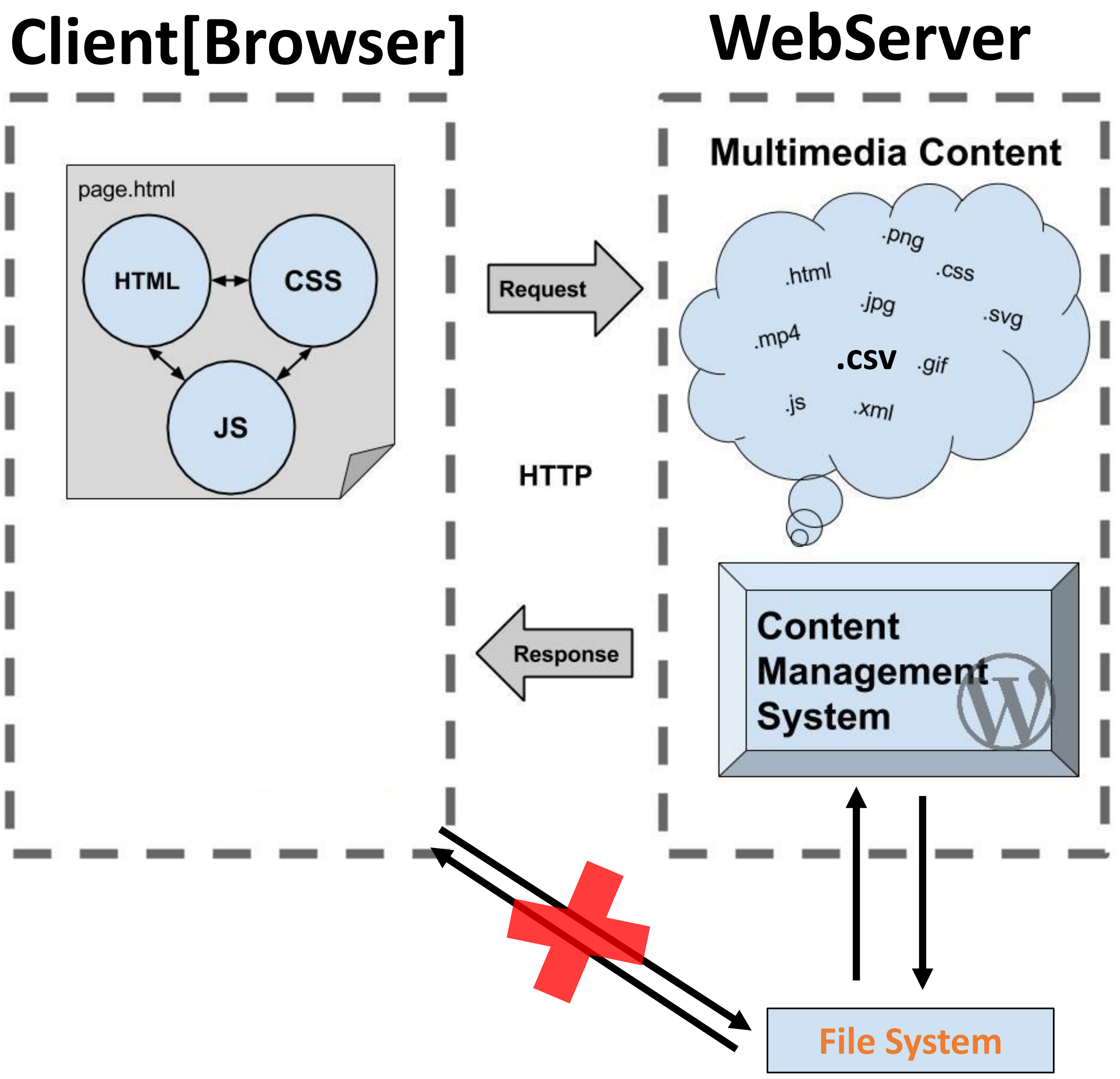
Web-Server: Flask

Flask

web development,
one drop at a time

Basic principle of a WebServer

Client <-> Server: Architecture



http://media2mult.uni-osnabrueck.de/pmwiki/fields/wp15/m2m.d/JavaSpring.Webserver/media/image/012_Client_Server_Modell.jpg (modified)

Server application frameworks

node js
LoopBack.io
Node.js API Framework
express
ASP.net
flask
RAILS
Spring
django
restify

https://optimalbi.com/wp-content/uploads/2016/07/APIBlogBanner-v2-768x480.png

Why Flask?

- Allows to access the filesystem (load data files)
- Is a microframework (written in python)
  - Modular → add extensions (upload handling, …)
- Helps to build a modular, structured web application
- HTML templates
  - Create templates (base.html)
  - Extend templates (extend base.html with extend.html)
  - Easy creation of multiple HTML pages (@app.route('/newpage'))
- You will use Flask in assignment 5 :)

Install Flask

Flask
web development,
one drop at a time

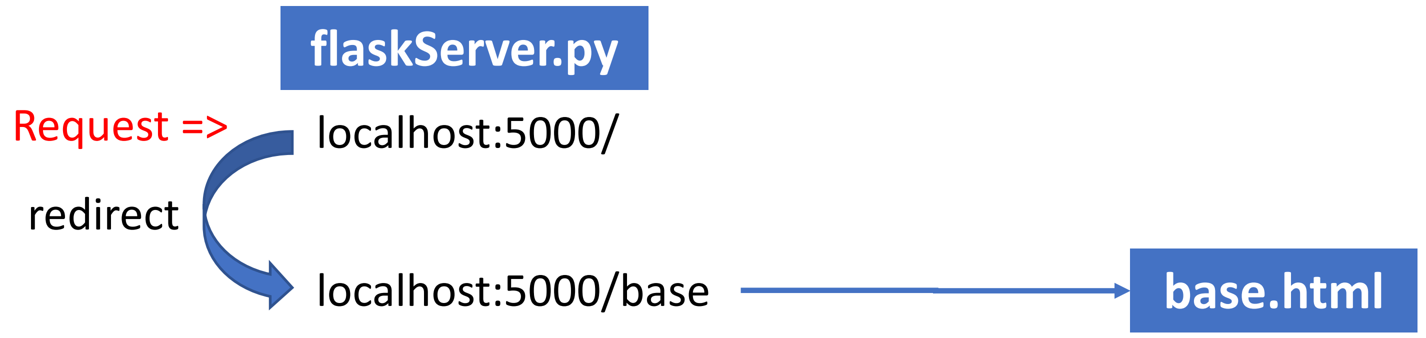
- command line: pip3 install flask / or pip install flask on windows
- more installation infos:
  - https://flask.palletsprojects.com/en/1.1.x/installation/
- Python should be already installed
- Maybe need to install pip3 first:

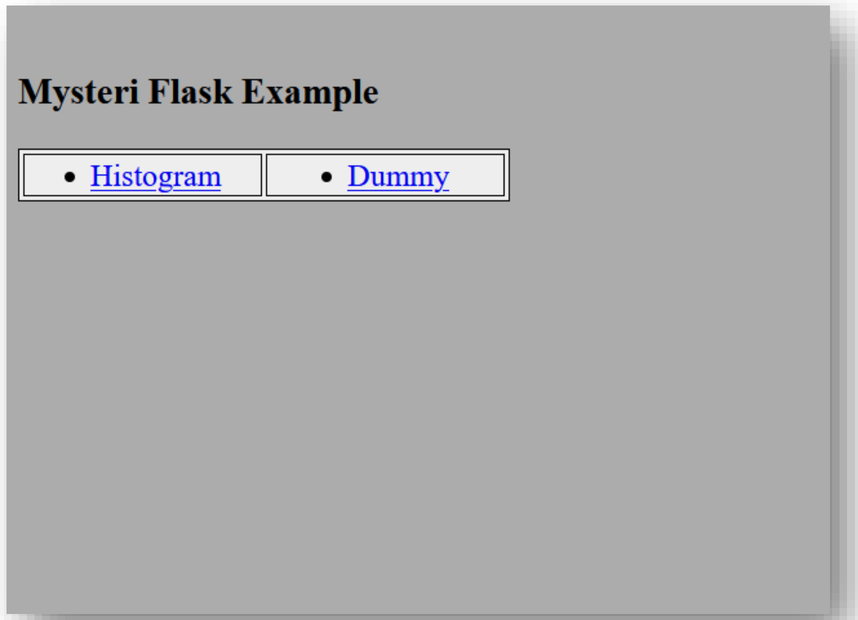
  sudo apt-get -y install python3-pip

Test your installation:

- Open terminal/command line / console
- Write “python3” press enter / or on windows “python” / “py”
- Write “import flask” press enter
- => there should be no error message

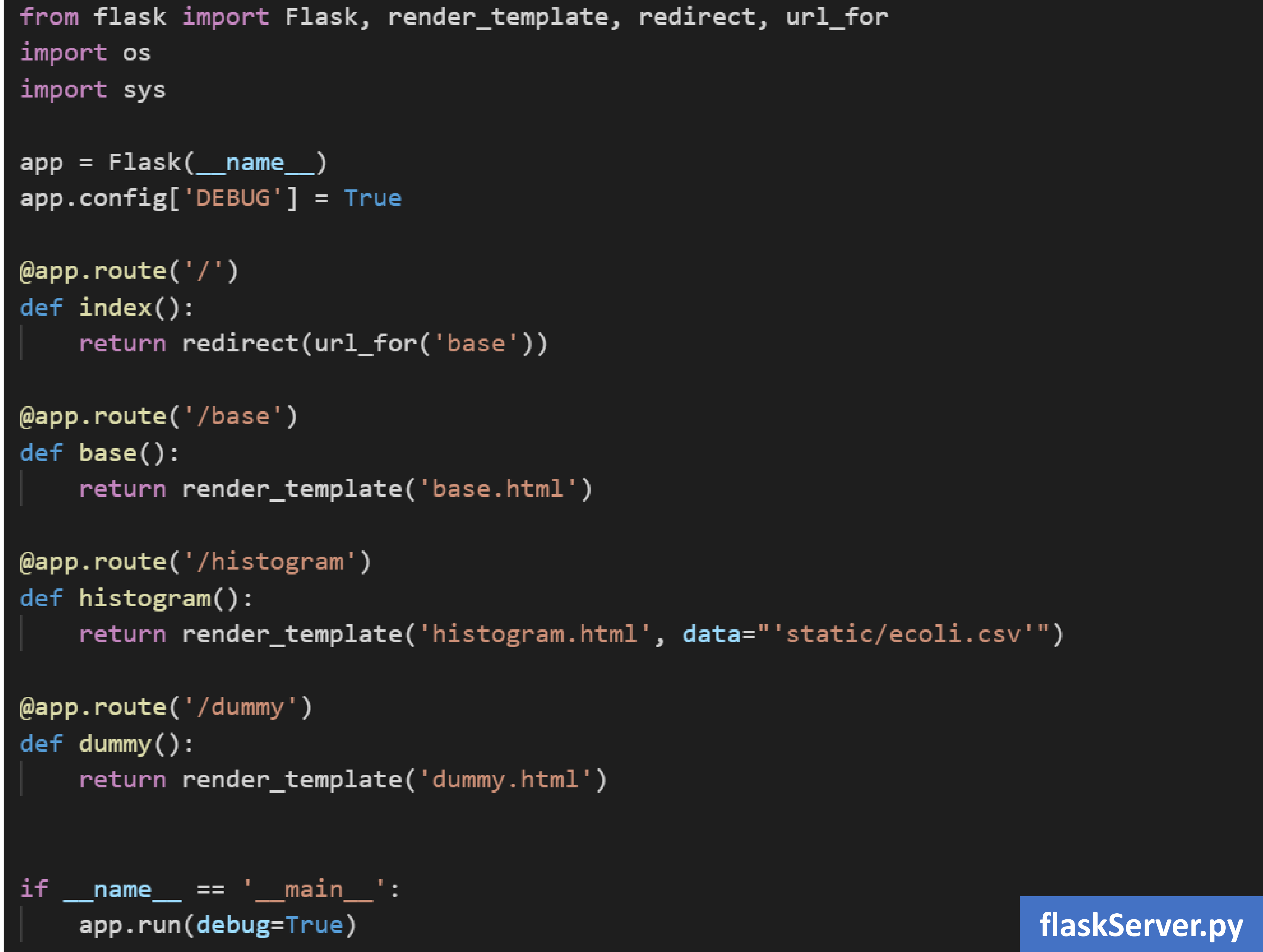
Functional Page Overview




Mysteri Flask Example

- Histogram
- Dummy

A Basic flask server setup



```python
from flask import Flask, render_template, redirect, url_for
import os
import sys

app = Flask(__name__)
app.config['DEBUG'] = True

@app.route('/')
def index():
    return redirect(url_for('base'))

@app.route('/base')
def base():
    return render_template('base.html')

@app.route('/histogram')
def histogram():
    return render_template('histogram.html', data="'static/ecoli.csv'")

@app.route('/dummy')
def dummy():
    return render_template('dummy.html')


if __name__ == '__main__':
    app.run(debug=True)
```

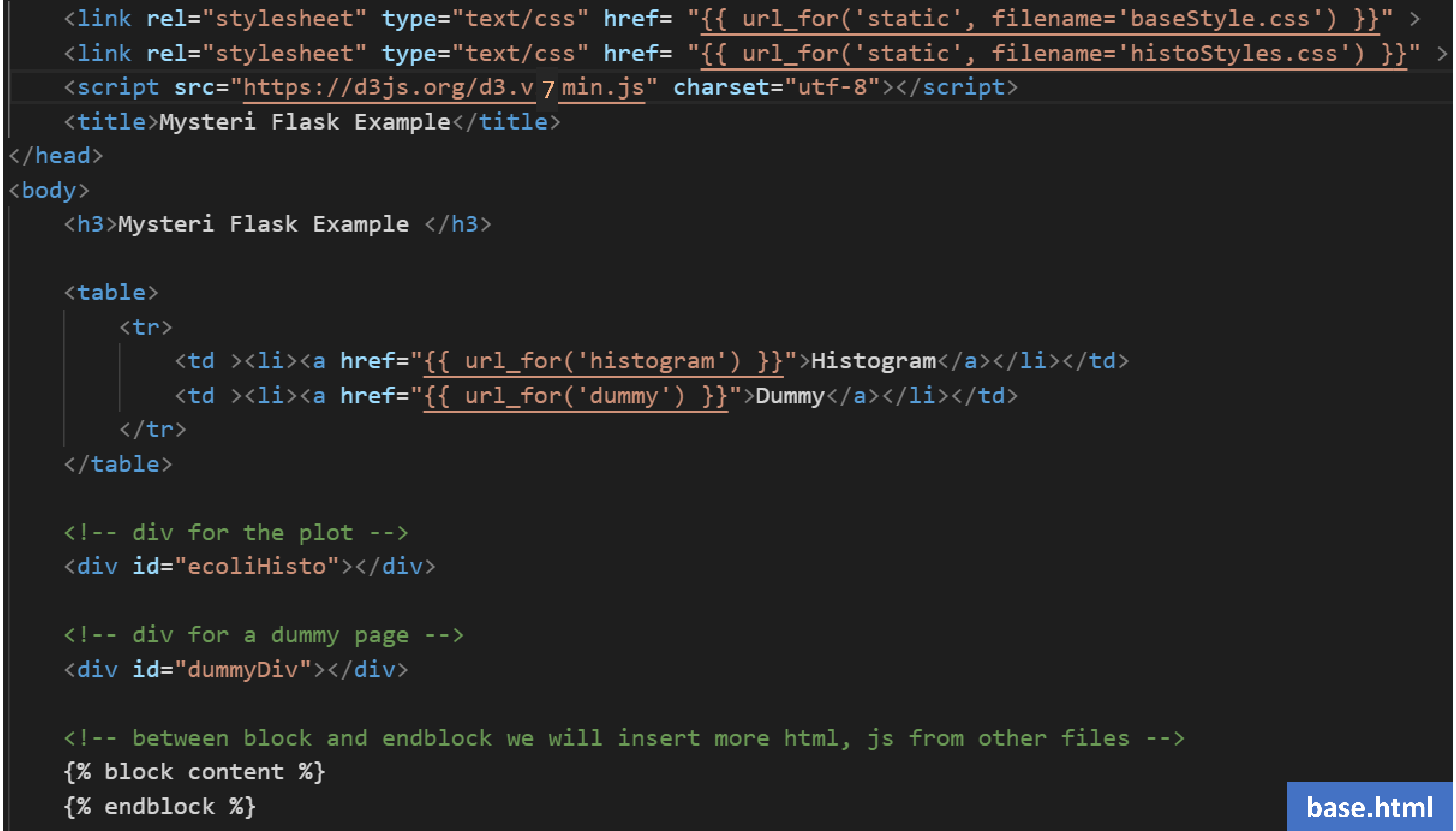
flaskServer.py

Special things of flaskServer.py

- app = Flask(__name__) → instantiated Flask application in “app”

- @app.route(“/example”)
  - makes backend stuff for us in background
  - to have at the end the page address “/example” (used in browser)

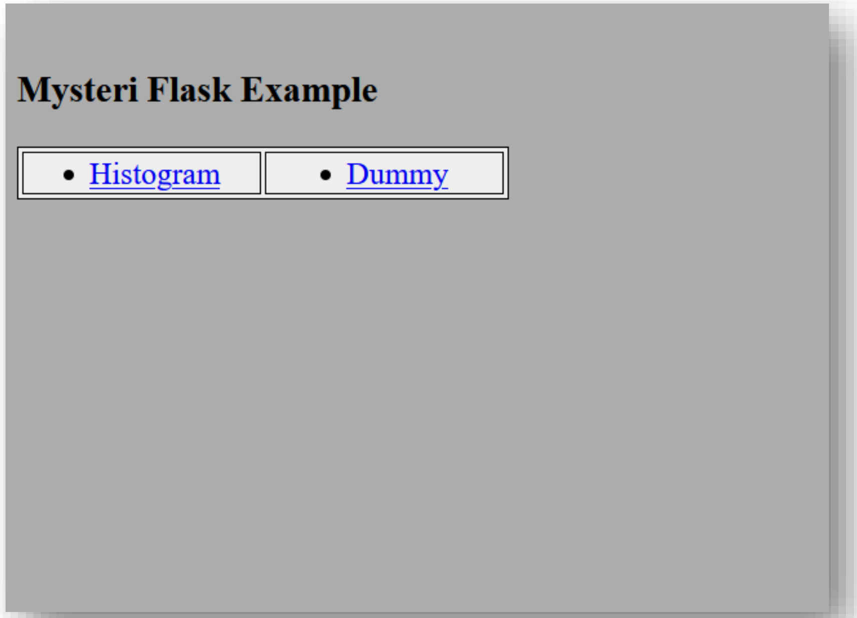
- __name__ = name of module
  - when running python script directly it will be “main”
  - need for development:
    - if __name__ == ‘__main__’:
      app.run(debug=True)



```html
    <link rel="stylesheet" type="text/css" href= "{{ url_for('static', filename='baseStyle.css') }}" >
    <link rel="stylesheet" type="text/css" href= "{{ url_for('static', filename='histoStyles.css') }}" >
    <script src="https://d3js.org/d3.v7.min.js" charset="utf-8"></script>
    <title>Mysteri Flask Example</title>
</head>
<body>
    <h3>Mysteri Flask Example </h3>

    <table>
        <tr>
            <td ><li><a href="{{ url_for('histogram') }}">Histogram</a></li></td>
            <td ><li><a href="{{ url_for('dummy') }}">Dummy</a></li></td>
        </tr>
    </table>

    <!-- div for the plot -->
    <div id="ecoliHisto"></div>

    <!-- div for a dummy page -->
    <div id="dummyDiv"></div>

    <!-- between block and endblock we will insert more html, js from other files -->
    {% block content %}
    {% endblock %}
```

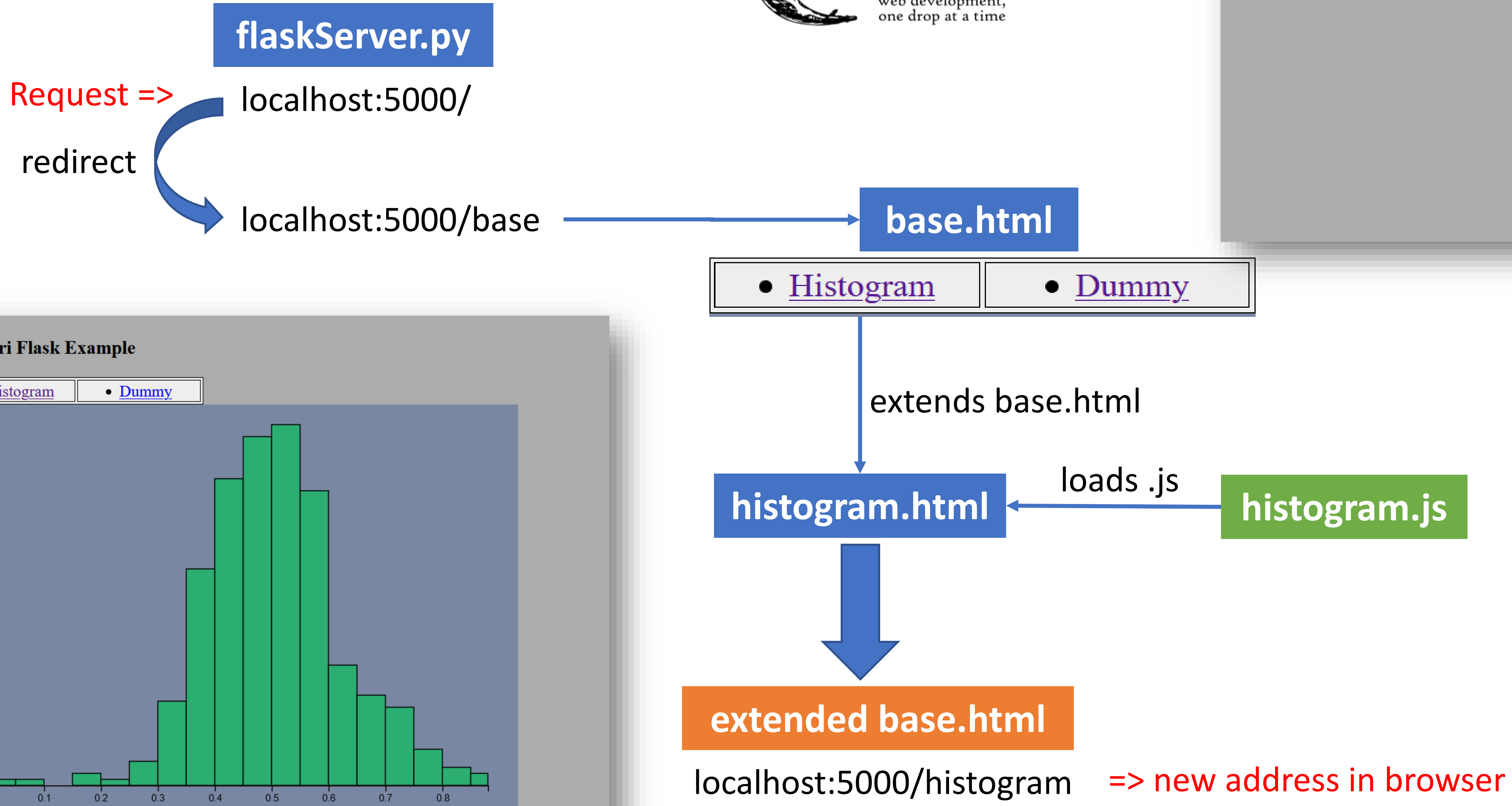
base.html

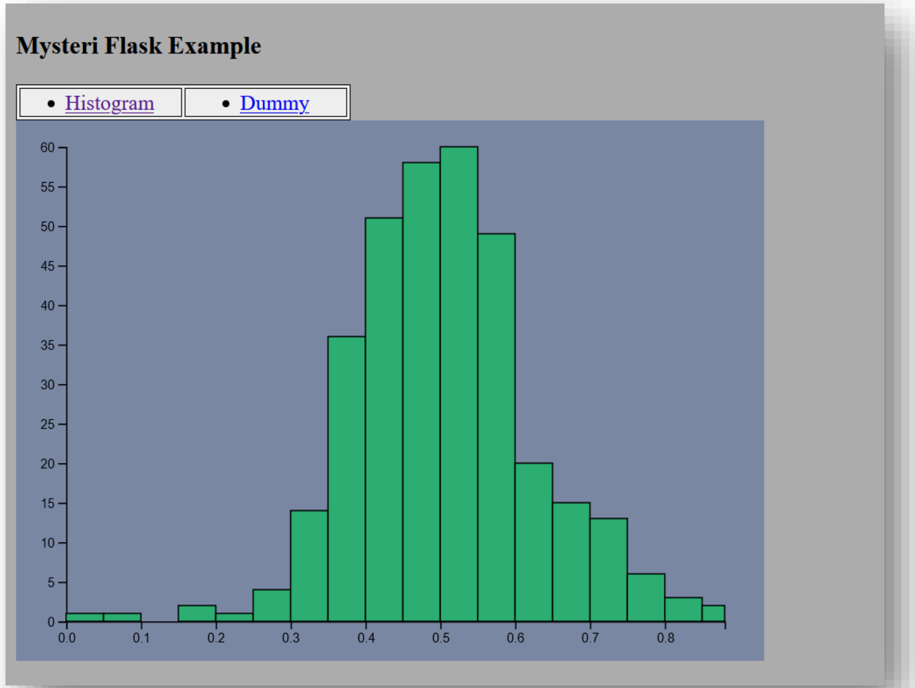
Functional Page Overview



Mysteri Flask Example

- Histogram
- Dummy

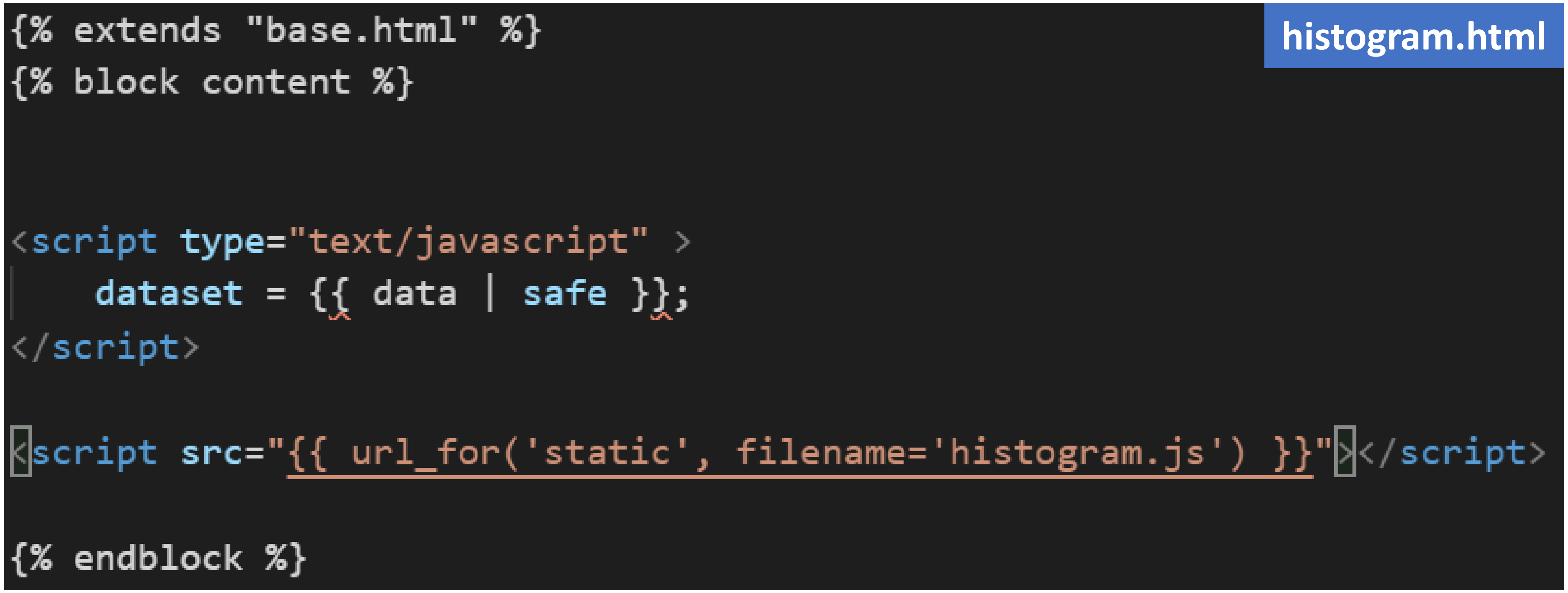




Mysteri Flask Example

- Histogram
- Dummy

HTML file of the histogram page



histogram.html

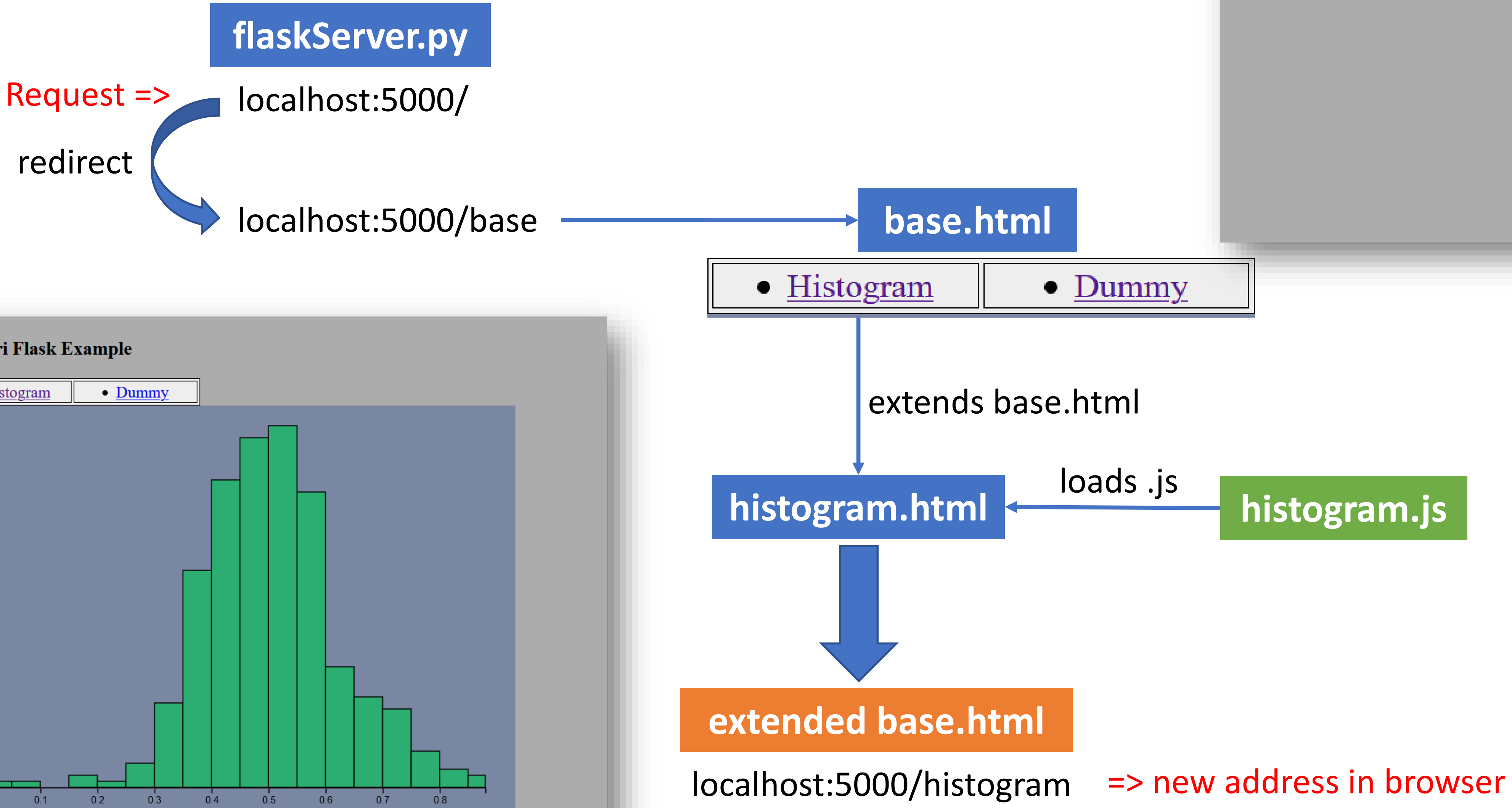
```
{% extends "base.html" %}
{% block content %}


<script type="text/javascript" >
    dataset = {{ data | safe }};
</script>

<script src="{{ url_for('static', filename='histogram.js') }}"></script>

{% endblock %}
```

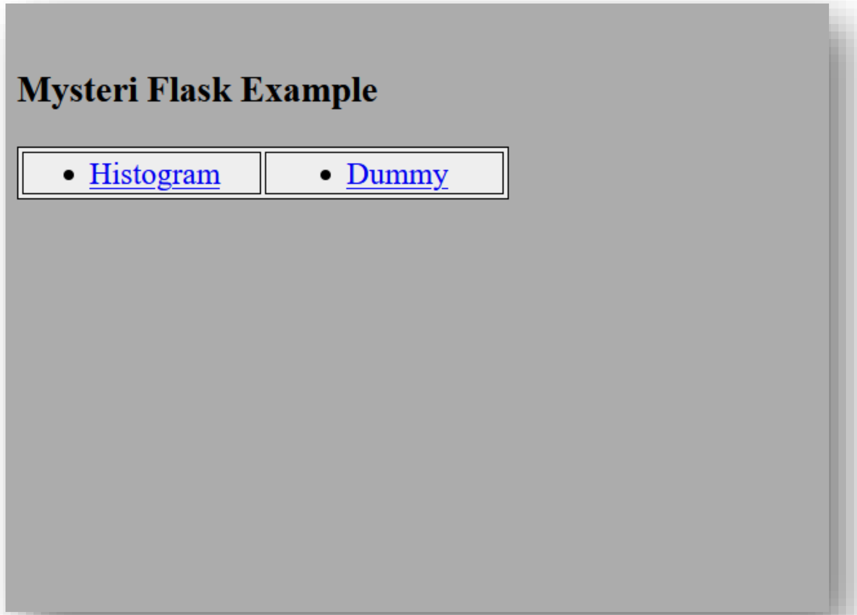
JavaScript / D3 file of histogram page

histogram.js

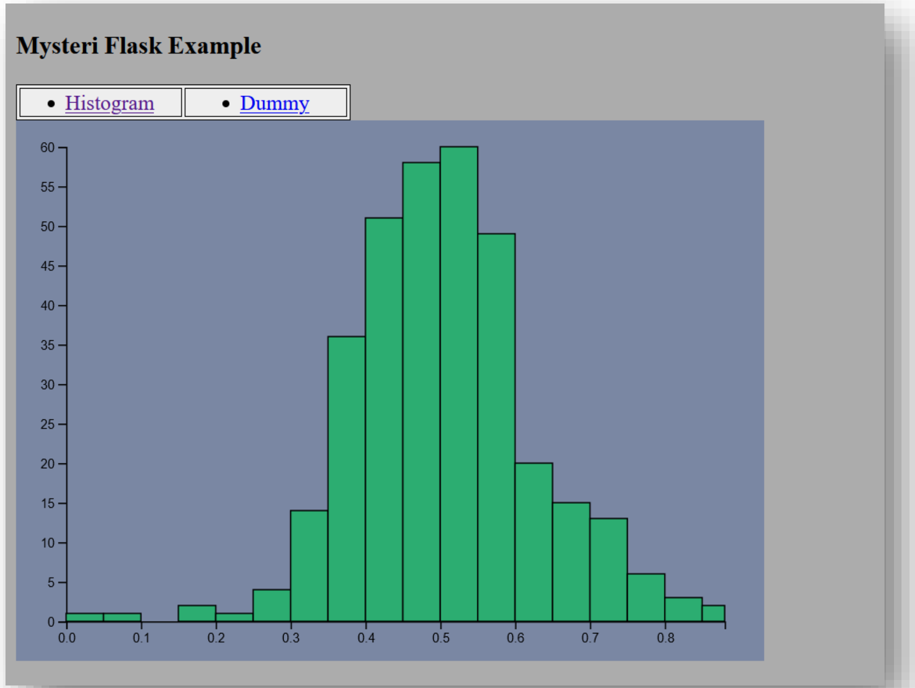
```
// set svg dimensions and margins
const svgWidth = 500 ;
const svgHeight = 360;
const margin = {top: 20, right: 30, bottom: 30, left: 38};

// append the svg object to the body of the page
let svg = d3.select("#ecoliHisto")
  .append("svg")
  .attr("width", svgWidth + margin.left + margin.right)
  .attr("height", svgHeight + margin.top + margin.bottom)
  .attr("class", "chart")
  .append("g")
  .attr("transform","translate(" + margin.left + "," + margin.top + ")");
```

Functional Page Overview



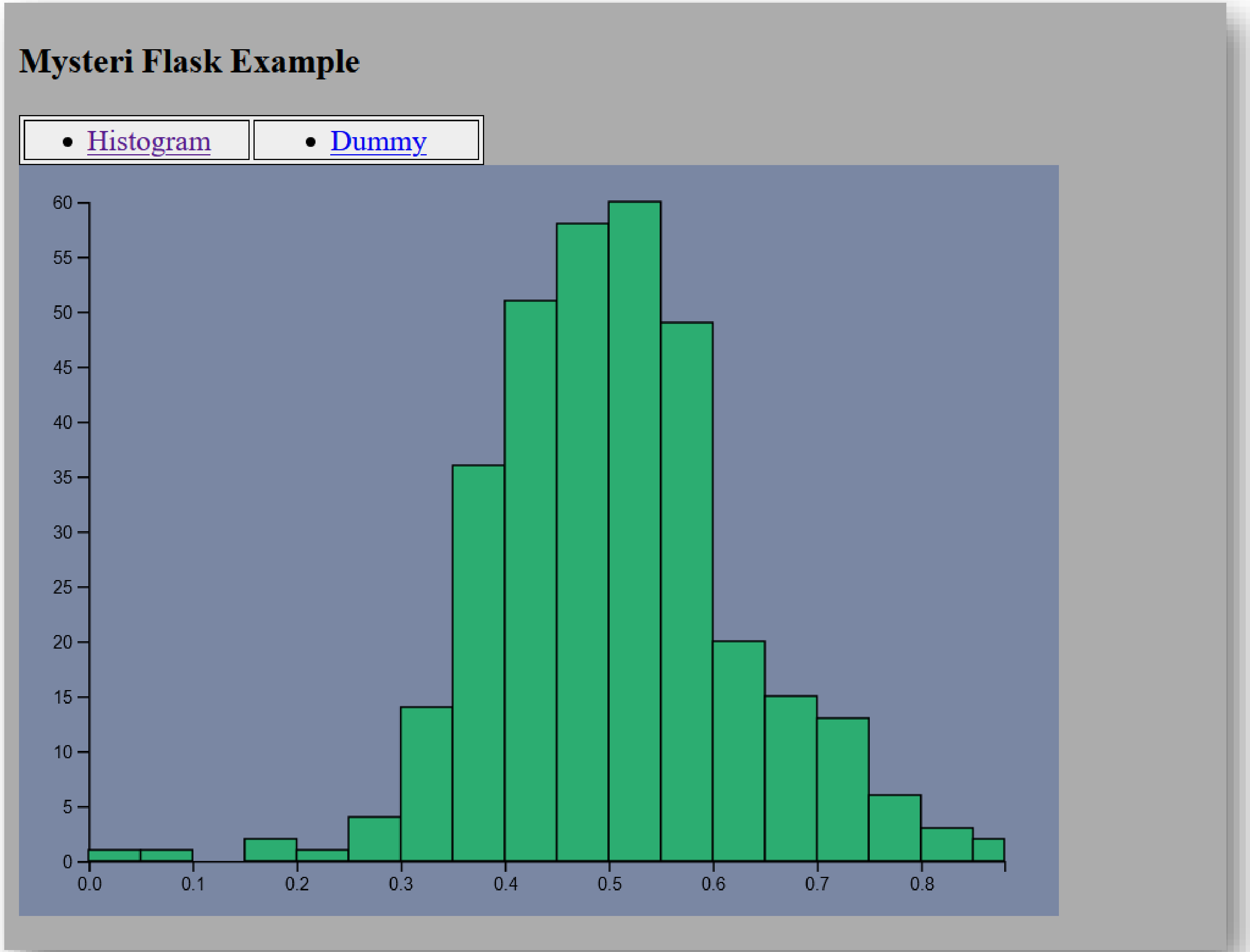

Mysteri Flask Example

- Histogram
- Dummy


Mysteri Flask Example

- Histogram
- Dummy

Testing Flask


Mysteri Flask Example

- Histogram
- Dummy

- Start the flask server from your source directory and load the page

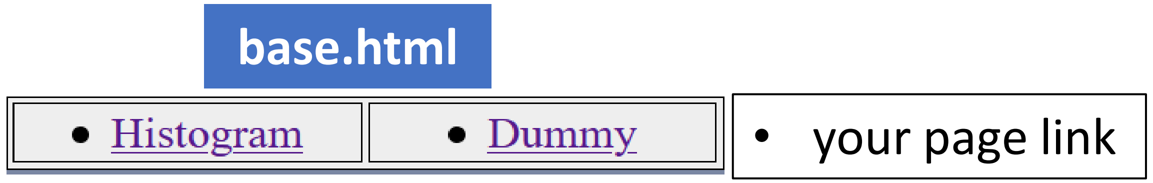
- python ./flaskServer.py

- http://localhost:5000/
- http://localhost:5000/histogram

- You should see the left picture as result

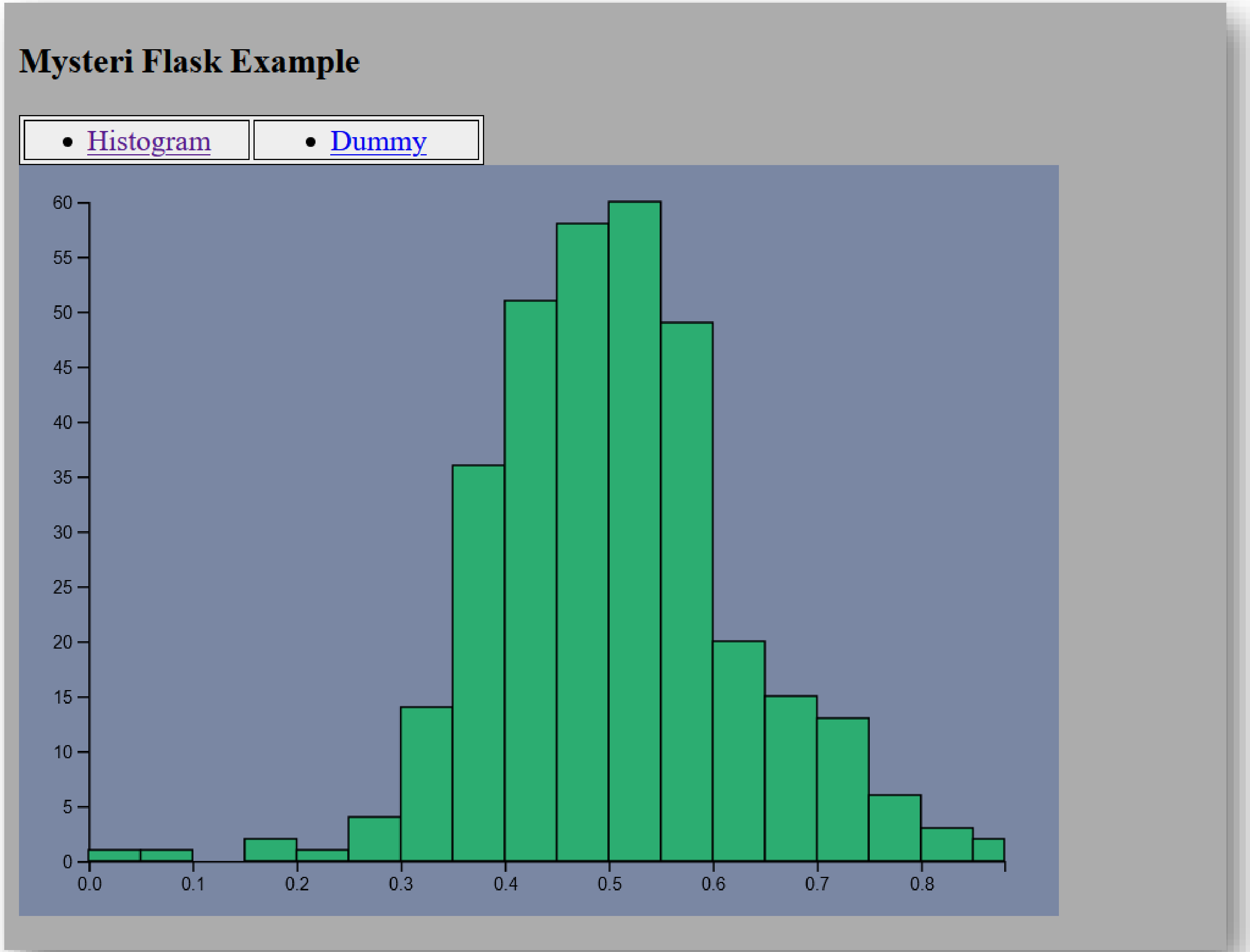
Your task now - Testing flask

- Add a new page with some basic content
- Make the page also available via a new link besides the two other ones



Continuing a new visualization

The Histogram

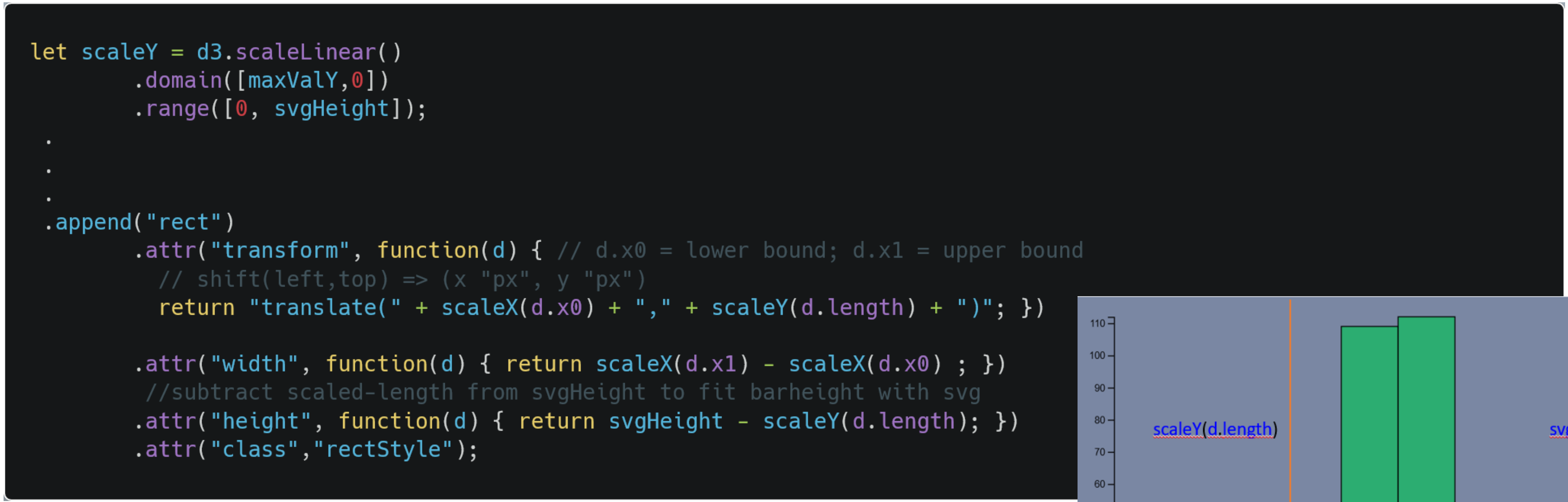


Mysteri Flask Example

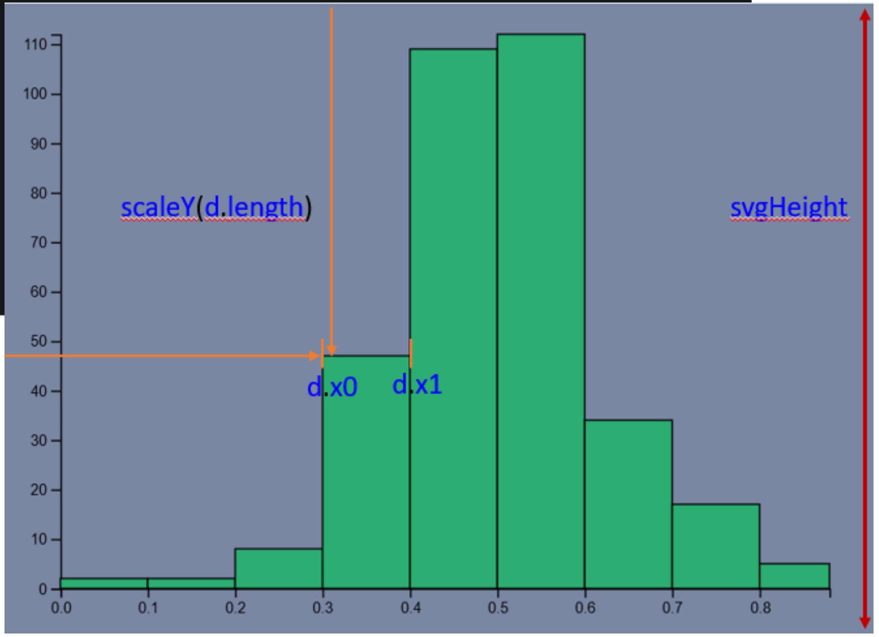
- Histogram
- Dummy

- Start the flask server from your source directory and load the page
- Python ./flaskServer.py
- http://localhost:5000/
- http://localhost:5000/histogram
- You should see the left picture as result

Histogram – Apply Scales to Bar-Width and Bar-Height



```
let scaleY = d3.scaleLinear()
        .domain([maxValY,0])
        .range([0, svgHeight]);
 .
 .
 .
 .append("rect")
        .attr("transform", function(d) { // d.x0 = lower bound; d.x1 = upper bound
          // shift(left,top) => (x "px", y "px")
          return "translate(" + scaleX(d.x0) + "," + scaleY(d.length) + ")"; })

        .attr("width", function(d) { return scaleX(d.x1) - scaleX(d.x0) ; })
         //subtract scaled-length from svgHeight to fit barheight with svg
        .attr("height", function(d) { return svgHeight - scaleY(d.length); })
        .attr("class","rectStyle");
```



Histogram - Your task now

- try to change the number of bins

- create an input field on your website for user input (number of bins)
    - Add an “update” button to apply the number of bins

Histogram - Bins

```
.thresholds(scaleX.ticks(15));
```