

AJAX and the OpenEye Tools

Andrew Dalke / Dalke Scientific
Göteborg, Sweden

```
<html>
```

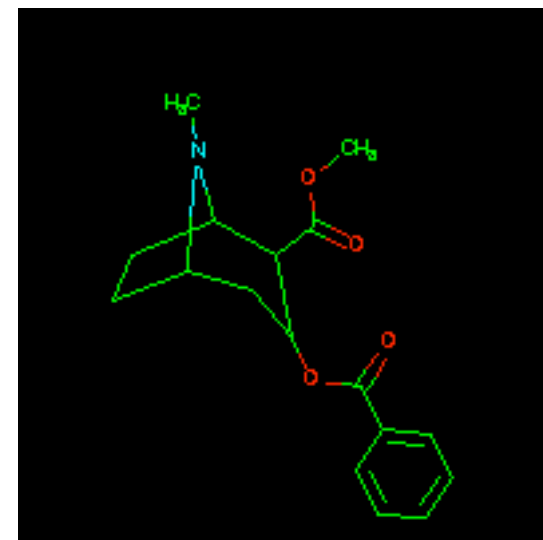
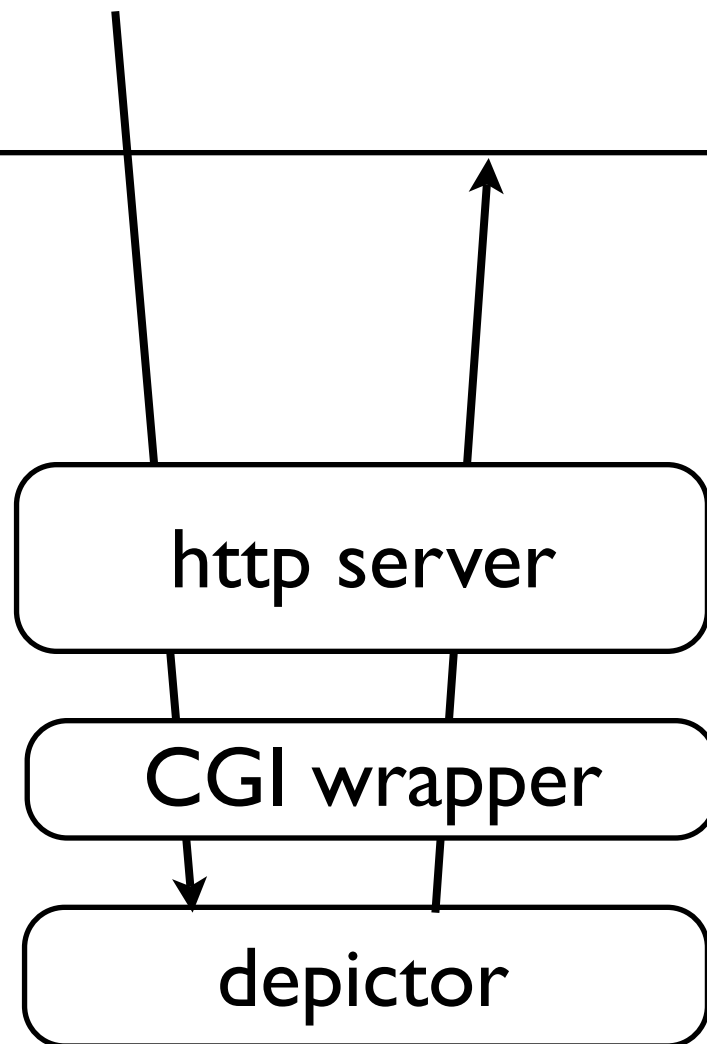
```
...
```

```

```

```
...
```

```
</html>
```

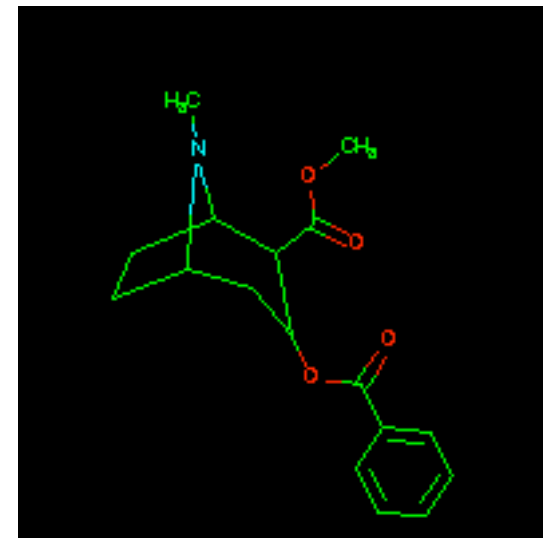
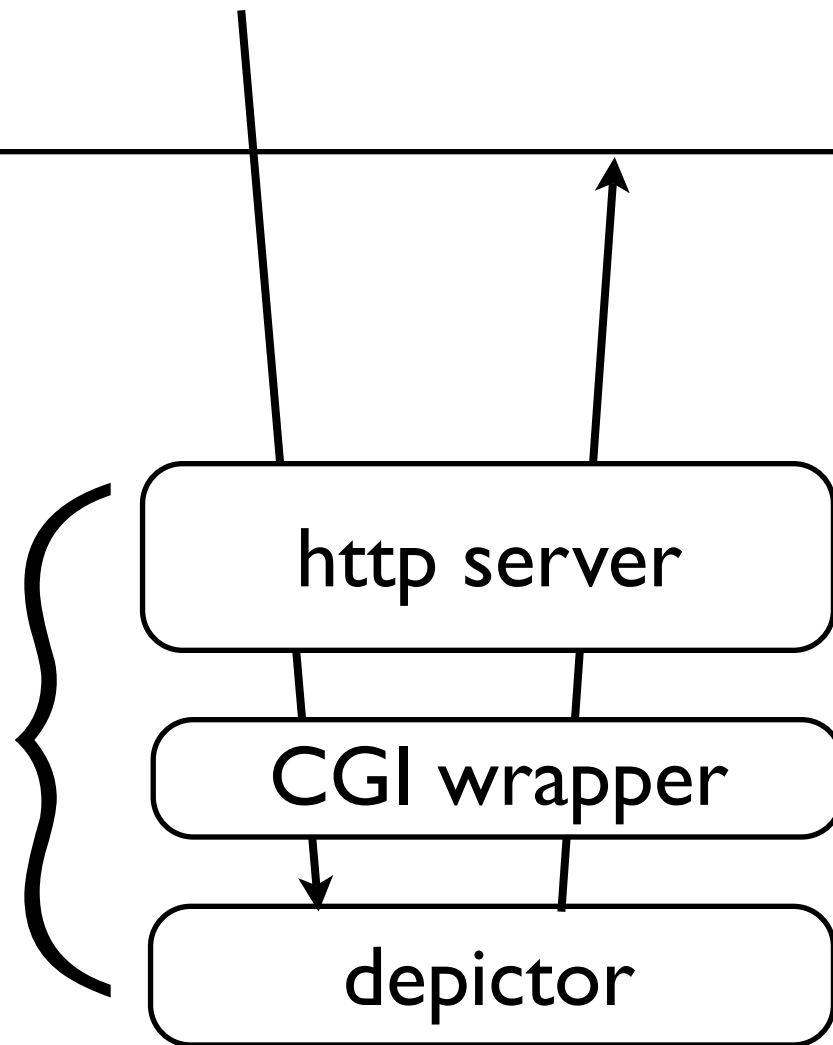


```
<html>
...

...
</html>
```

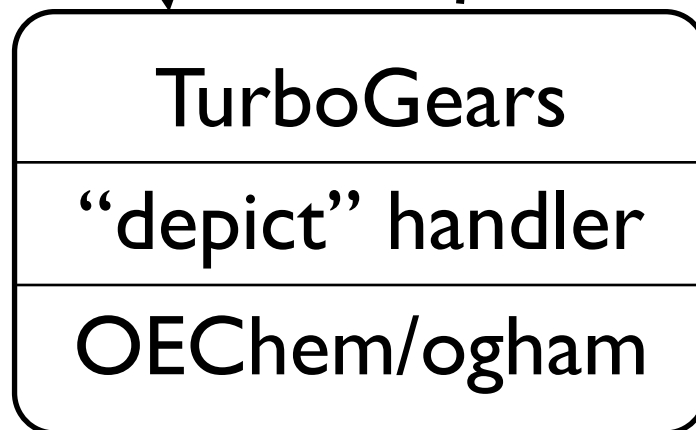
“Web
Application
Stack”

Ruby on Rails
Django
TurboGears



```
<html>
...

...
</html>
```

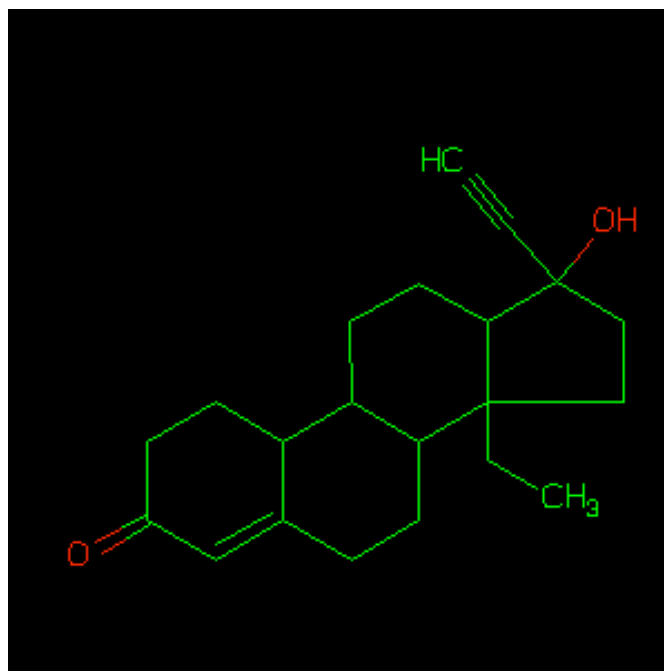


Google “OE8BitImage to PNG” for pointers
of how to use Ogham’s Python interface

```
<form action="/depict">  
Enter SMILES: <input type="text" name="smiles">  
</form>
```

Enter SMILES: CCC12CCC3C(C1CCC2(C#C)O)CCC4=CC(=O)CCC34

submit form,
server responds with an image:



```
<form action="/show_depict">  
Enter SMILES: <input type="text" name="smiles">  
</form>
```

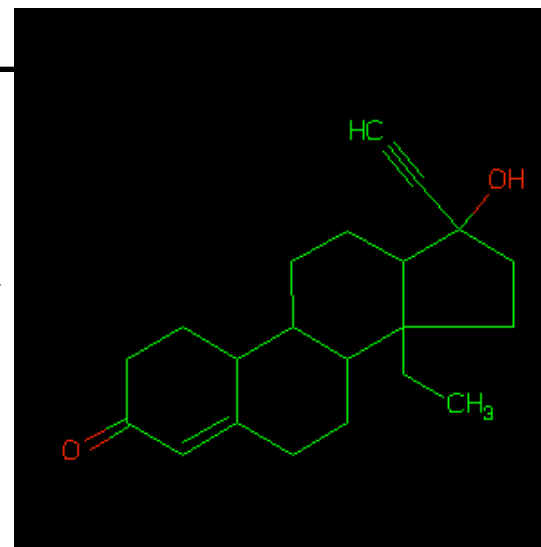
Enter SMILES:

server could respond with HTML ↓

```
<form action="show_depict">  
Enter SMILES: <input type="text" name="smiles">  
</form>  

```

client requests the
image from the server



Improved interactivity

Use Javascript to change the image src URL instead of doing two round-trips to the server.

Enter SMILES:



```
<form action="http://localhost:8080/depict"
  onsubmit="..Javascript code to update the image..">
Enter SMILES: <input type="text" size="40" name="smiles"
  id="smiles">
</form>
```

```

```

```
<html> <head>
<script type="text/javascript">
var update_image = function (x) {
    var smiles = document.getElementById("smiles").value;
    var depiction_ele = document.getElementById("depiction");
    depiction_ele.src = "http://localhost:8080/depict?smiles=" +
        escape(smiles);
    return false;
}
</script>
</head>

<body>
<form action="http://localhost:8080/depict"
    onsubmit="return update_image()">
Enter SMILES: <input type="text" size="40" name="smiles"
    id="smiles">
</form>



</body> </html>
```

```
<html> <head>
<script type="text/javascript">
var update_image = function (x) {
  var smiles = document.getElementById("smiles").value;
  var depiction_ele = document.getElementById("depiction");
  depiction_ele.src = "http://localhost:8080/depict?smiles=" +
    escape(smiles);
  return false;
}
</script>
</head>

<body>
<form action="http://localhost:8080/depict"
  onsubmit="return update_image()">
Enter SMILES: <input type="text" size="40" name="smiles"
  id="smiles">
</form>



</body> </html>
```

This is out-dated!

New things in the last 10 years:

- better support for HTML components (MVC)
- separation of content (HTML) from actions (Javascript)
- mostly standardized cross-browser support for events
- Javascript libraries to simplify development
(and work around browser bugs)

jQuery, MochiKit, YUI, Dojo, Prototype, Script.aculo.us ...

```
<html> <head>
<script type="text/javascript" src="jquery-1.2.3.js"></script>
<script type="text/javascript">
$(document).ready(function() {
  $("form").submit(function() {
    $("#depiction").attr("src",
      "http://localhost:8080/depict?smiles=" +
        escape($("#smiles").val()));
    return false;
  });
});
</script>
</head>

<body>
<form action="http://localhost:8080/depict">
Enter SMILES: <input type="text" size="40" name="smiles"
  id="smiles">
</form>


</body> </html>
```

jQuery

More Interactive

Update after every key event

```
<html> <head>
<script type="text/javascript" src="static/javascript/Mochikit.js"></script>
<script type="text/javascript" src="static/javascript/jquery.js"></script>
<script type="text/javascript">
$(document).ready(function() {
    $("#smiles").keydown(function() {
        callLater(0, update_image);
    });
});
}

var update_image = function () {
    $("#depiction").attr("src",
        "http://localhost:8080/depict?smiles=" +
        escape($("#smiles").val()));
}
</script>
</head>

<body>
<form action="http://localhost:8080/depict">
Enter SMILES: <input type="text" size="40" name="smiles" id="smiles">
</form>


</body> </html>
```

Called for each
“key down” event

“callLater” from MochiKit

JSON requests

Also show the IUPAC name
as the SMILES is being typed

Javascript can request a new document from the server

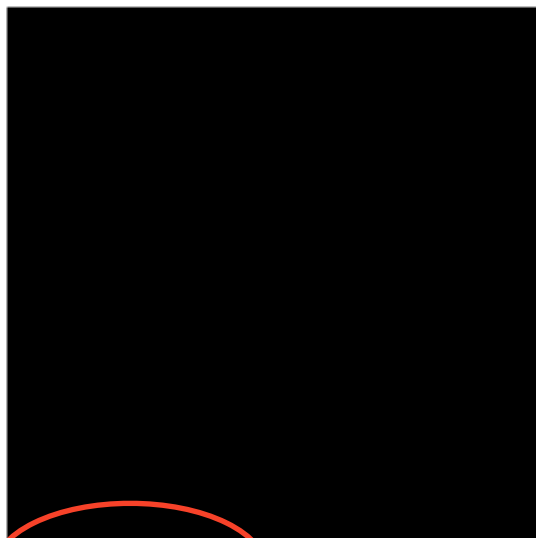
`http://localhost:8080/smi2name?smiles=c1ccccc10`



```
Content-Type: text/javascript
```

```
{"status": "valid", "name": "phenol"}
```

Enter SMILES:



Name:

```
<body>
<form action="http://localhost:8080/depict">
Enter SMILES: <input type="text" size="40"
  name="smiles" id="smiles">
</form>

<br>
Name: <span id="compound_name"></span>

</body>
```

```
$(document).ready(function() {
    $("#smiles").keydown(function() {
        callLater(0, update_display);
    });
});

var update_display = function () {
    var smiles = $("#smiles").val();
    $("#depiction").attr("src",
        "http://localhost:8080/depict?smiles=" + escape(smiles));

    var d = loadJSONDoc("http://localhost:8080/smi2name",
        {smiles: smiles});
    d.addCallback(show_compound_name);
}

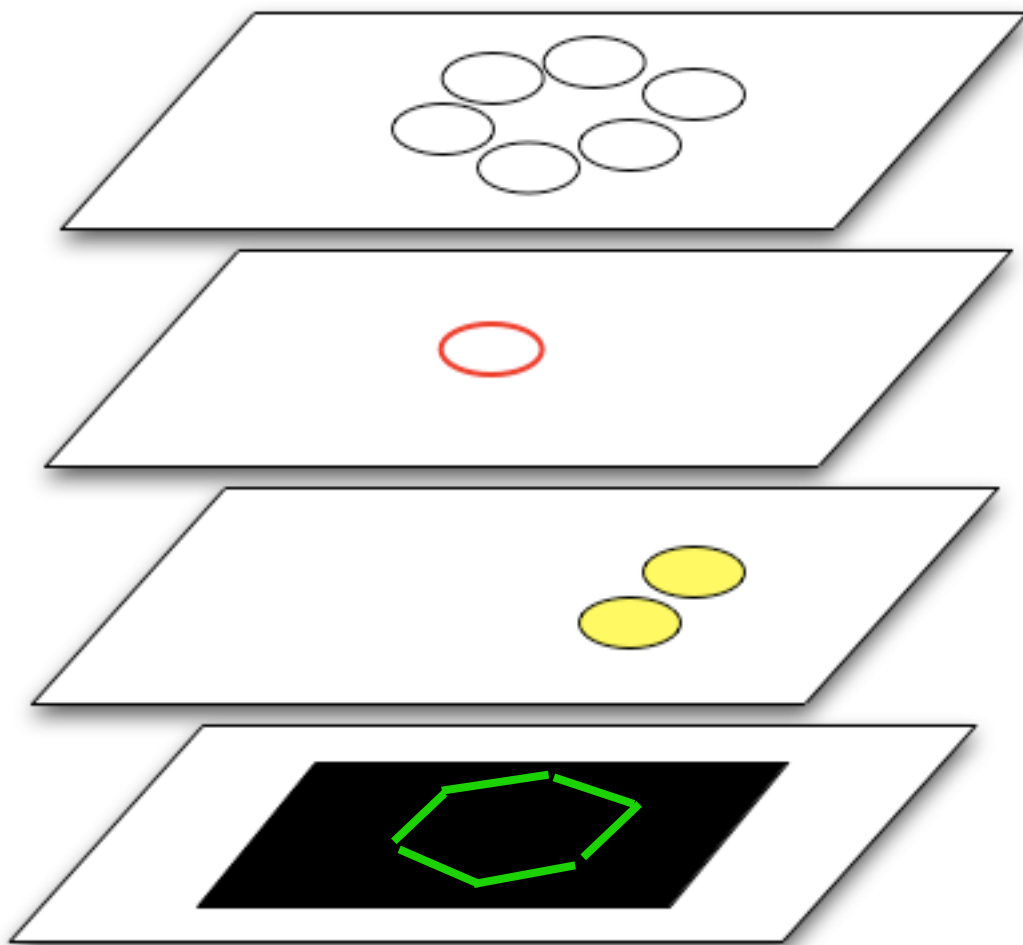
var show_compound_name = function(smi2name_result) {
    var color = "black";
    if (smi2name_result.status == "invalid") {
        color = "red";
    }
    $("#compound_name").text(smi2name_result.name).css(
        "color", color);
}
```

smi2name in TurboGears

```
@expose(format="json")
def smi2name(self, smiles):
    mol = OEGraphMol()
    if OEParseSmiles(mol, smiles):
        status = "valid"
    else:
        status = "invalid"
    name = oeiupac.OECreateIUPACName(mol)
    return dict(status=status, name=name)
```

Demo application

Layout / CSS



Transparent image map
(updated via AJAX)

Atom region mouseover
moves the highlight image

Atom selections drawn
with a translucent image

Ogham depiction