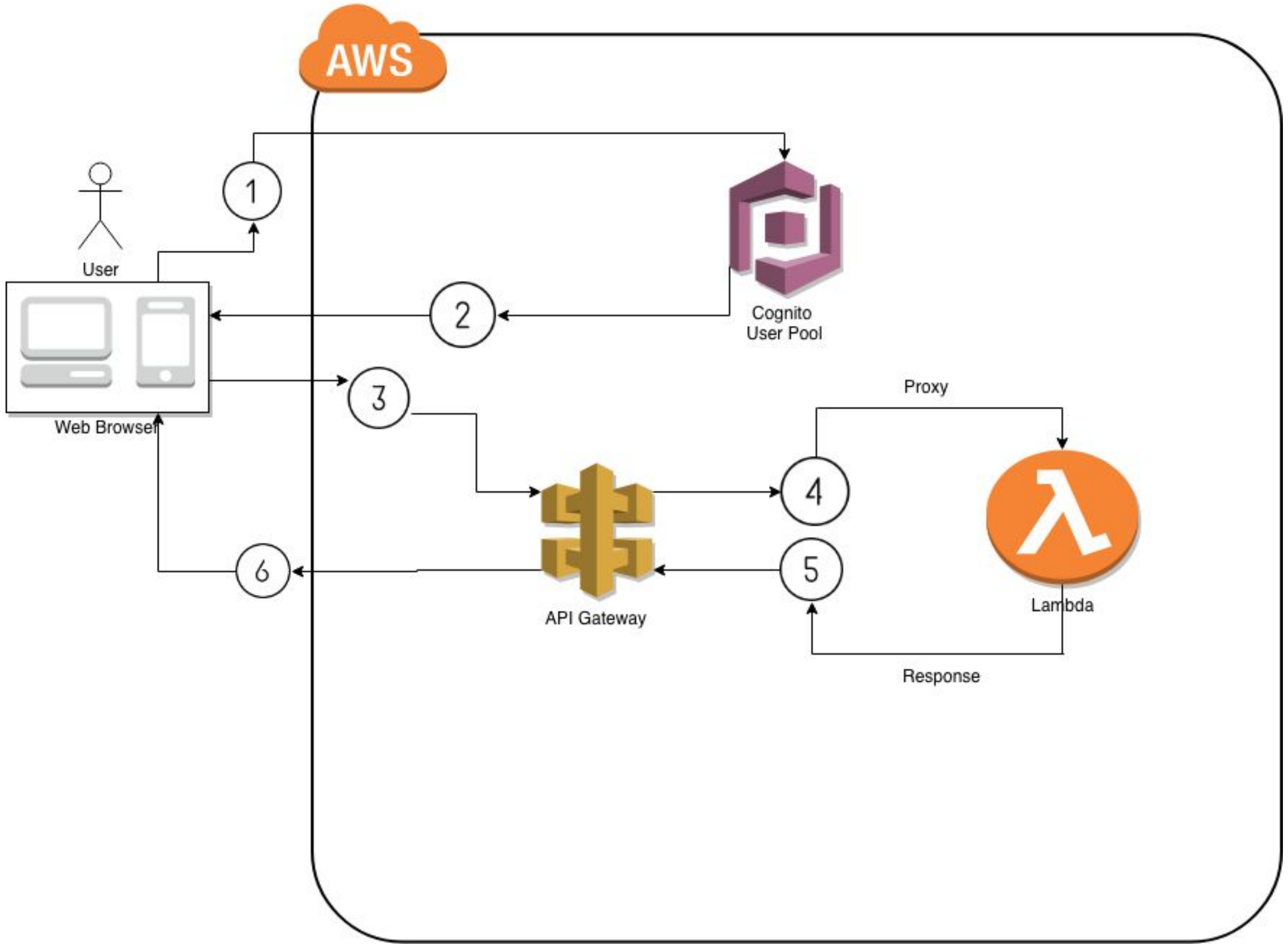
Week 14
Web services integration projects

Agenda

- **Week 14:** Web services integration projects
  - AWS Integration: Granting authorized users limited access to create VMs from a web browser


AWS
User
Web Browser
1
2
3
Cognito User Pool
Proxy
4
5
6
API Gateway
Lambda
Response

Static vs Dynamic site

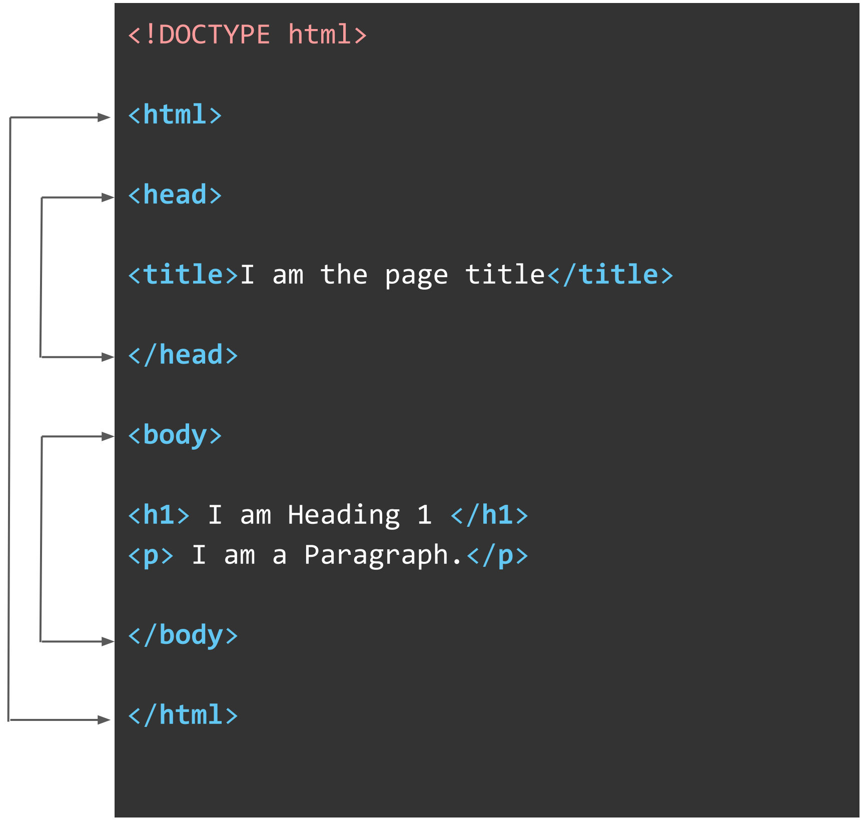
Static is more secure and faster. Why ?

On dynamic site the client contact the server, the server run a programming code which might need access to a database before response

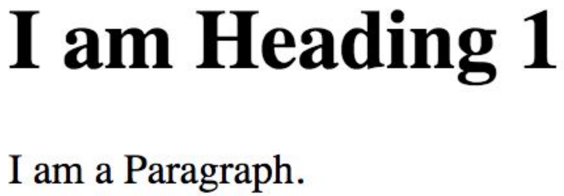
Components

- HTML
- HTML JavaScript
- AWS Lambda
- API Gateway
- AWS EC2
- AWS Cognito

HTML



```
<!DOCTYPE html>

<html>

<head>

<title>I am the page title</title>

</head>

<body>

<h1> I am Heading 1 </h1>
<p> I am a Paragraph.</p>

</body>

</html>
```

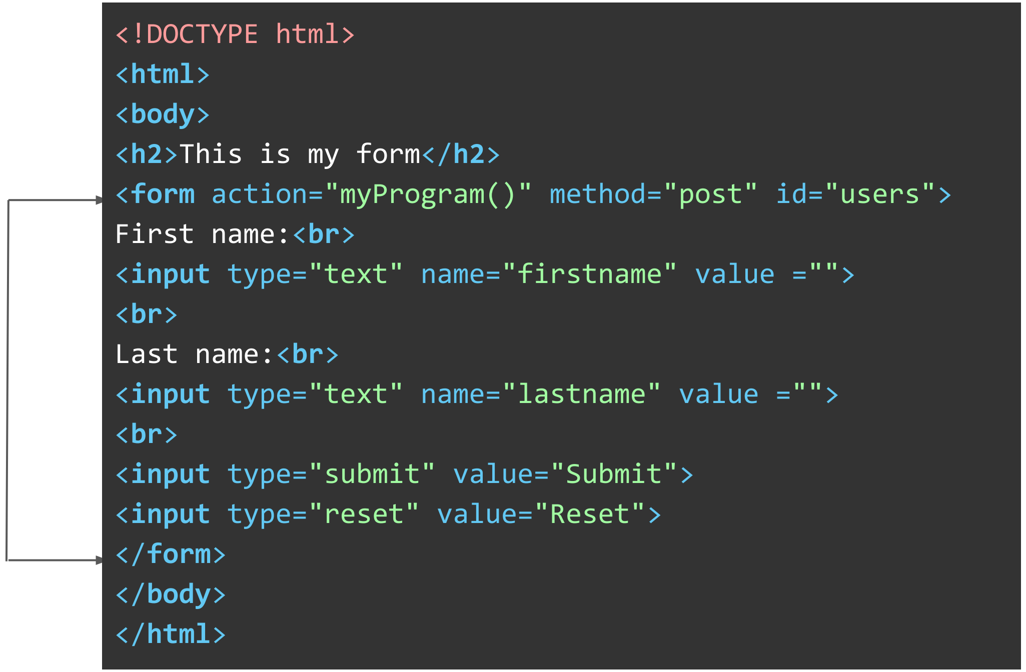


I am Heading 1

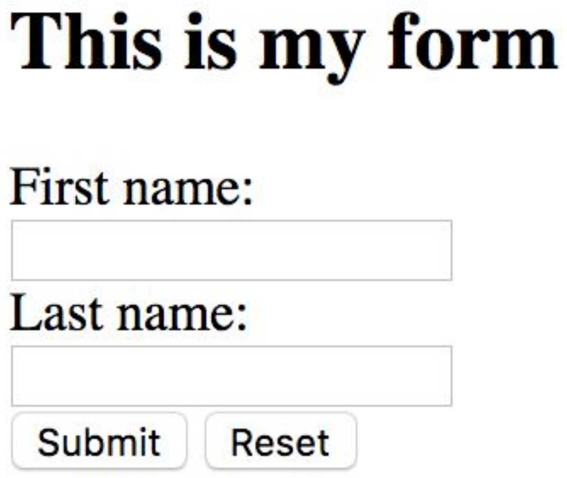
I am a Paragraph.

HTML FORMS

- A section that contains interactive controls to enable a user to submit information to a web server.
- Method → URL of the “myProgram” that processes the information submitted via form.
- Method → The HTTP method that the browser uses to submit the form (get / post).



```
<!DOCTYPE html>
<html>
<body>
<h2>This is my form</h2>
<form action="myProgram()" method="post" id="users">
First name:<br>
<input type="text" name="firstname" value ="">
<br>
Last name:<br>
<input type="text" name="lastname" value ="">
<br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>
</body>
</html>
```



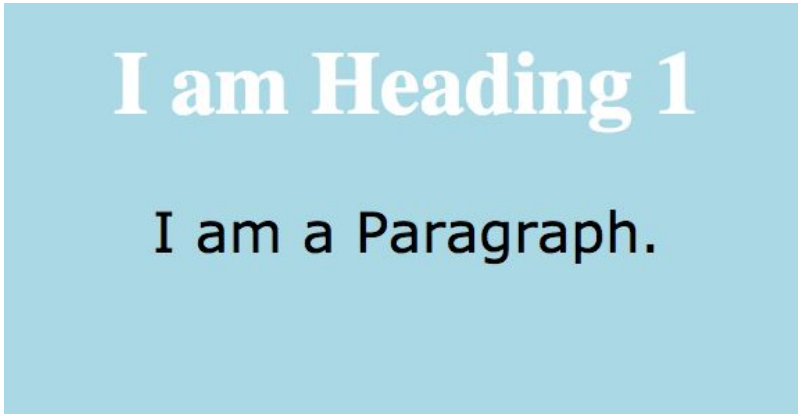
This is my form

First name:

Last name:

CSS

```css
body {
	background-color: lightblue;
}
h1 {
	color: white;
	text-align: center;
}
p {
	font-family: verdana;
	font-size: 20px;
}
```



I am Heading 1

I am a Paragraph.

JavaScript

- JavaScript is the programming language of HTML and the Web.
- It makes HTML pages more dynamic and interactive.

https://www.w3schools.com/html/html_scripts.asp

```
<!DOCTYPE html>
<html>
<body>
<h2> My Calculator </h2>
<label for="number1">Number 1:</label>
<input type="number" name="number1" id="number1">
<label for="number2">Number 2:</label>
<input type="number" name="number2" id="number2">
<script>
function multiplyBy()
{
        num1 = document.getElementById("number1").value;
        num2 = document.getElementById("number2").value;
        document.getElementById("multiply").innerHTML = num1 * num2;
}
</script>
<p id="multiply"></p>

<button type="button" onclick="multiplyBy()">
Click me to display result.
</button>
</body>
</html>
```

JQUERY

jQuery is a JavaScript Library.

```
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
```

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
    $("p").click(function(){
        $(this).hide();
    });
});
</script>
</head>
<body>

<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>

</body>
</html>
```

Calculator again with jQuery

```
<!DOCTYPE html>
<html>
 <script language="JavaScript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.0/jquery.min.js"></script>
<body>
<h2> My Calculator </h2>
<label for="number1">Number 1:</label>
<input class="form_textfield" type="number" name="number1" id="number1">
<label for="number2">Number 2:</label>
<input class="form_textfield" type="number" name="number2" id="number2">
<label for="result">result is:</label>
<input class="form_textfield" type="number" name="result" id="result">
<script>
    $(document).ready(function(){
        $('.form_textfield').keyup(multiple);
    });
    function multiple()
    {
            $('#result').val($('#number1').val() * $('#number2').val());
    }
</script>
</body>
</html>
```

AJAX

Asynchronous JavaScript And XML

AJAX is a technique for accessing web servers from a web page to

- Update
- Request
- Receive
- Send

data from/to the web servers.

GET vs POST

GET

Parameter part of the URL

- URL stay in the browser history
- Visible to everyone
- Not secure , not to send sensitive parameters
- safe for less than 2K URL length (depends on browser and web server)

ASCII characters only

Can be cached

GET vs POST cont.

POST

Parameters are in the body

not cached

not visible

- more secure
- send sensitive info

Any data format

CORS

Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served.

Certain "cross-domain" requests, notably Ajax requests, are forbidden by default by the same-origin security policy.

Cognito

For authentication

AWS console -> cognito -> create userpool ->

Pool name:

Review Default

App client -> add an app client -> deselect Generate client secret

Write down : **Pool Id , App client id**

Lambda

1. Create Role for lambda
2. Create Lambda function
3. Test the function

Lambda

```
import boto3
import json

ec2 = boto3.resource('ec2')

def lambda_handler(event, context):
    instances = ec2.instances.filter(Filters=[])
    InstanceList = []
    for instance in instances:
        InstanceList.append(instance.id)
    return {
        'statusCode': 200,
        'body': json.dumps(InstanceList),
        "headers": {
           'Content-Type': 'text/plain',
           #'Content-Type': 'application/json',
           'Access-Control-Allow-Origin': '*'
        }
   }
```

AWS API Gateway

API -> Create API -> API Name : “e91”

E91 -> Authorizers -> Create New Authorizer

Name:

Cognito -> Cognito User Pool - > e91

Token Source : Authorization

E91 -> resources -> Action -> Create Resources -> Name: ec2s , Enable API Gateway CORS

E91 -> resources -> ec2s -> Action -> Create method -> GET ->

Lambda Function , Use Lambda Proxy integration , Lambda Func:listEC2

E91 -> resources -> ec2s -> GET -> Method Request -> Authorization -> “e91”

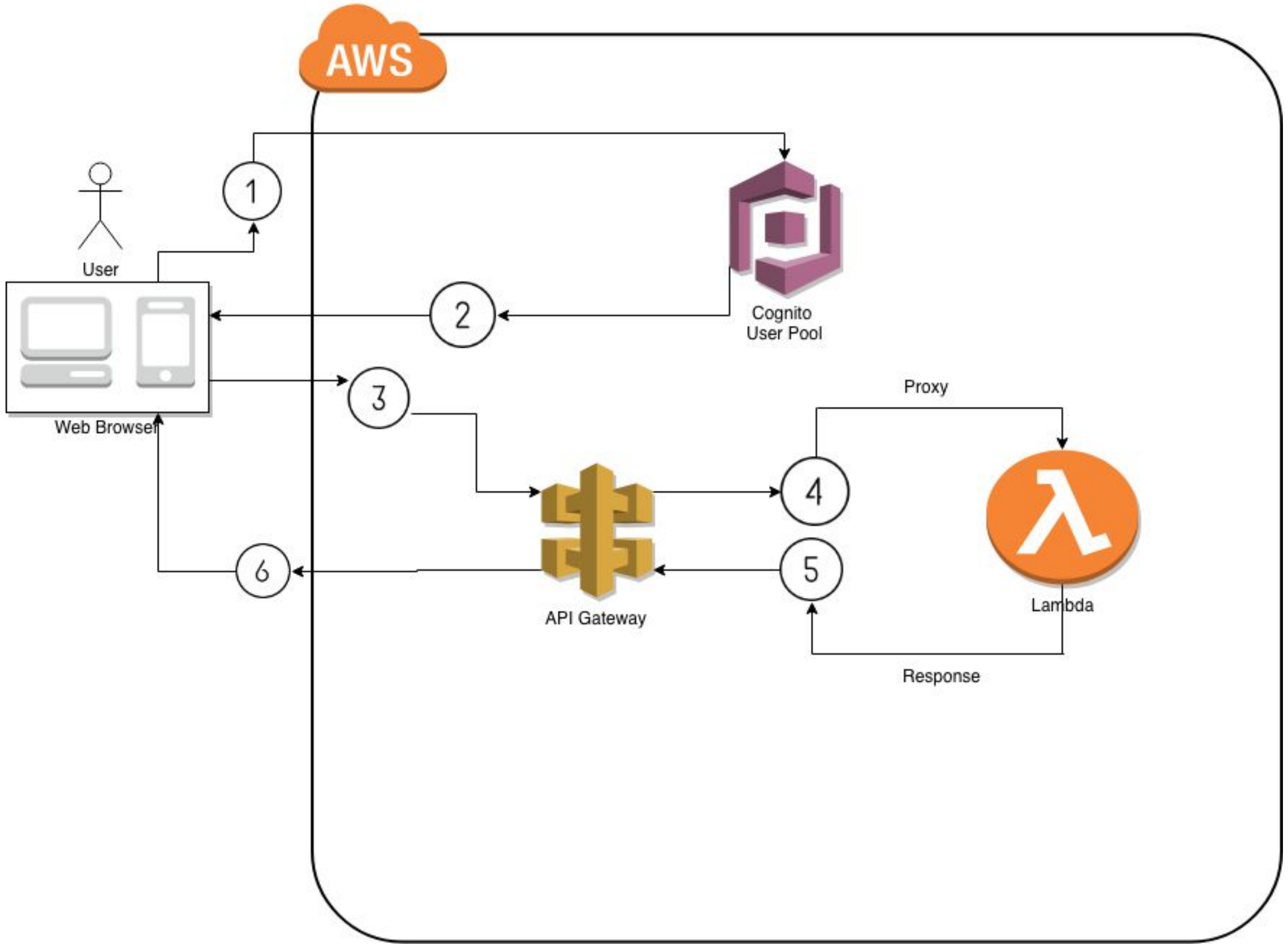
AWS API Gateway cont.

E91 -> resources -> ec2s -> Enable CORS

E91 -> resources -> ec2s -> Deploy

Write down the Invoke URL

E91 -> resources -> ec2s -> GET -> TEST


AWS
User
1
2
Cognito
User Pool
Web Browser
3
Proxy
4
5
6
API Gateway
Lambda
Response

S3

Can we put the site in S3 ?

Google API

Google API

Enable speech transcribe

Console - > AMI -> Service Account - > Create service account -> Role : Cloud Speech Service Agent -> Create Key "JSON" (download the key in safe place)

https://console.cloud.google.com/apis/credentials?project=cscie91-222920&folder&organizationId

Run the program

```
pip install google-cloud-speech

export GOOGLE_APPLICATION_CREDENTIALS="/PATH/TO/CREDENTIALS.json"

Python3 speech1.py
```

Speech1.py : https://cloud.google.com/speech-to-text/docs/quickstart-client-libraries