Web development - Intermediate

Agenda

Day 1(How the web works):

- How the web works
- Client-server architecture
- Evolution of web(WWW vs Internet)
- Set up Developer Environment

Day 2(Bootstrap, JQuery and DOM manipulation):

- Bootstrap
- DOM
- DOM selectors and events
- jQuery

Day 3(HTTP/JSON/AJAX +Async JS):

- HTTP/HTTPS
- JSON
- AJAX
- Asynchronous JavaScript

Agenda

Day 4(Frameworks):

- Introduction about frameworks and How the frameworks work under the hood?
- Introduction to React(state, props, component)

Day 5(APIs and microservices):

- How APIs work?
- Evolution of APIs
- Micro services and Web services

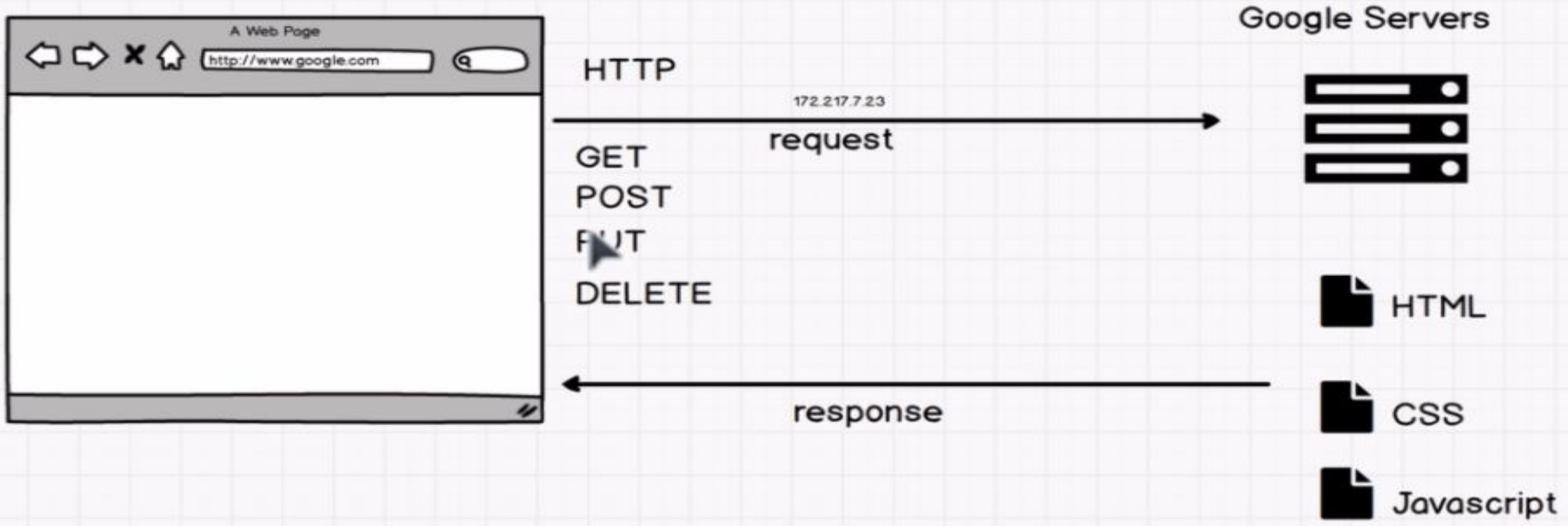
Day 6-7(Backend):

- Basics
- Introduction to NodeJs and ExpressJs

Day 3

- HTTP/HTTPS
- JSON
- AJAX
- Asynchronous JavaScript

HTTP/HTTPS


A Web Page
http://www.google.com
HTTP
172.217.7.23
request
GET
POST
PUT
DELETE
Google Servers
HTML
CSS
Javascript
response

Request methods/verbs

- GET requests a specific resource in its entirety
- HEAD requests a specific resource without the body content
- POST adds content, messages, or data to a new page under an existing web resource
- PUT directly modifies an existing web resource or creates a new URI if need be
- DELETE gets rid of a specified resource
- TRACE shows users any changes or additions made to a web resource
- OPTIONS shows users which HTTP methods are available for a specific URL
- CONNECT converts the request connection to a transparent TCP/IP tunnel
- PATCH partially modifies a web resource

JSON

```
{
 "employees"
  [
   { "firstName":"John", "lastName":"Doe" },
   { "firstName":"Anna", "lastName":"Smith" },
   { "firstName":"Peter", "lastName":"Jones" }
  ]
}
```

XML

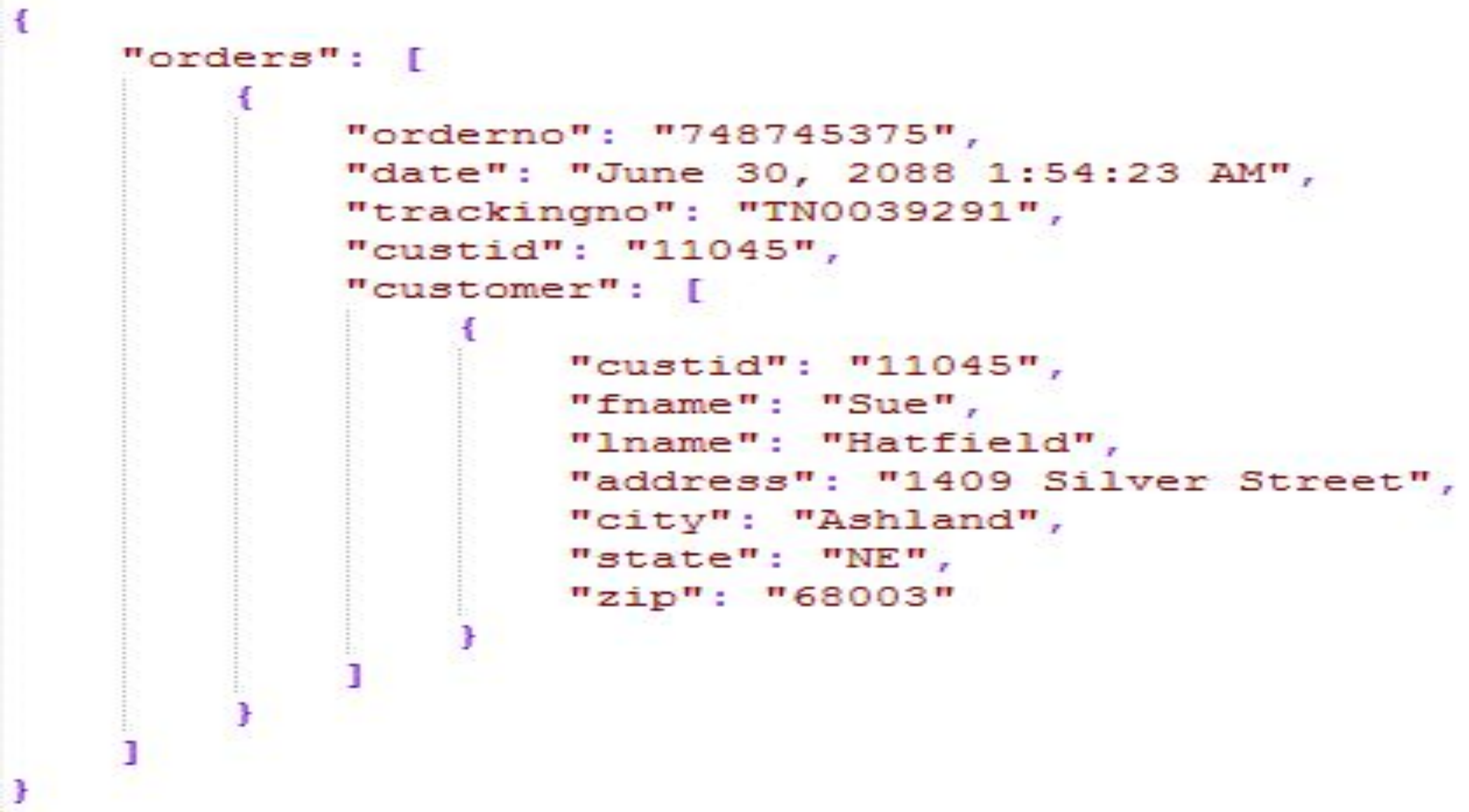
```
<employees>
   <employee>
      <firstName>John</firstName> <lastName>Doe</lastName>
   </employee>
   <employee>
      <firstName>Anna</firstName> <lastName>Smith</lastName>
   </employee>
   <employee>
      <firstName>Peter</firstName> <lastName>Jones</lastName>
   </employee>
</employees>
```

JSON

- JavaScript Object Notation
- Syntax for storing and exchanging data
- Actually text, written with Javascript Object Notation
- "self-describing" and easy to understand

Further reading:

https://medium.com/omarelgabrys-blog/json-in-a-nutshell-7d638dfea7cc



```
{
    "orders": [
        {
            "orderno": "748745375",
            "date": "June 30, 2088 1:54:23 AM",
            "trackingno": "TN0039291",
            "custid": "11045",
            "customer": [
                {
                    "custid": "11045",
                    "fname": "Sue",
                    "lname": "Hatfield",
                    "address": "1409 Silver Street",
                    "city": "Ashland",
                    "state": "NE",
                    "zip": "68003"
                }
            ]
        }
    ]
}
```

JSON.parse()

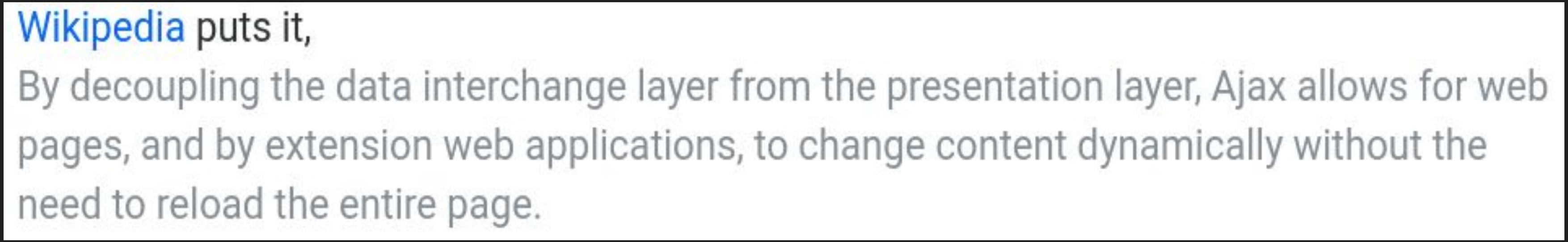
```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

JSON.stringify()

```
var myJSON = JSON.stringify(obj);
```

AJAX

- Asynchronous JavaScript And XML
- exchange data in the background without actually disturbing the user experience



Wikipedia puts it,

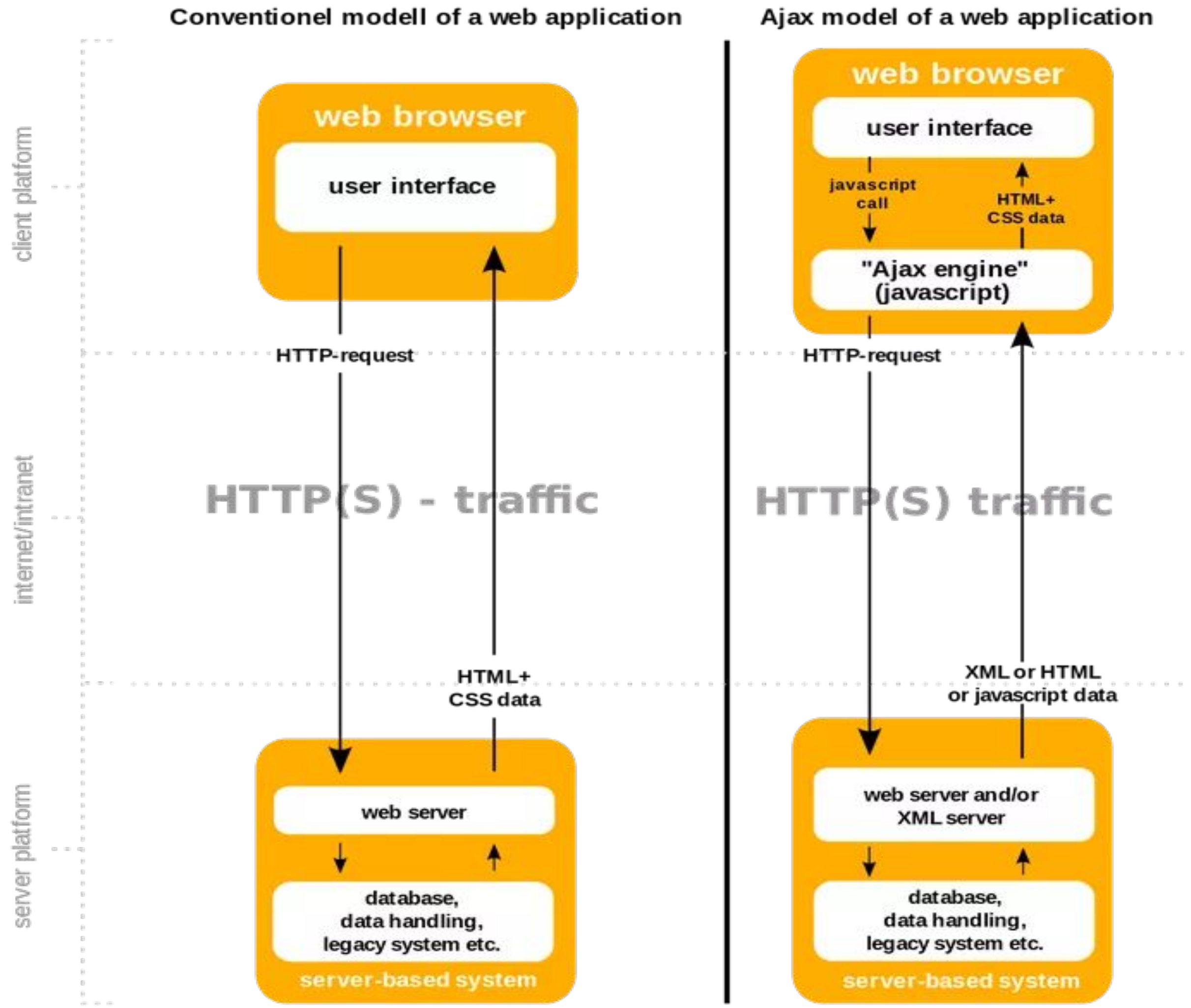
By decoupling the data interchange layer from the presentation layer, Ajax allows for web pages, and by extension web applications, to change content dynamically without the need to reload the entire page.

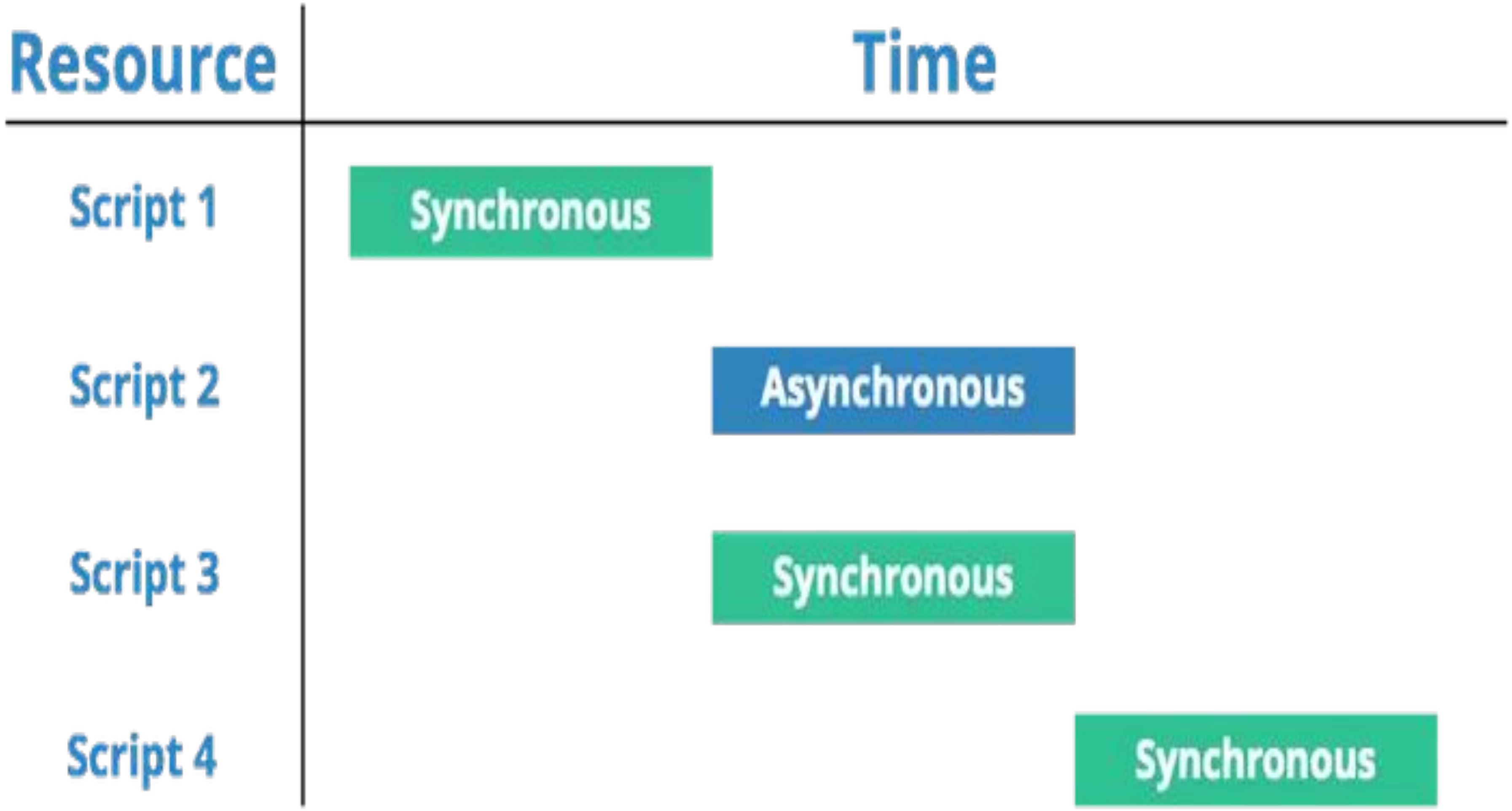
Resources:

https://www.keycdn.com/support/ajax-programming

https://api.jquery.com/jquery.ajax/

How AJAX works?




Resource
Time
Script 1
Synchronous
Script 2
Asynchronous
Script 3
Synchronous
Script 4
Synchronous

Ajax programming example

```
<script src="example.js" async></script>
```

```
$.ajax(url [, settings])
```

```
$.ajax('demo.html', {
    success: function(result) {
        $('#testdiv').html(result);
    }
});
```

Async Javascript

Resources:

https://bit.ly/3aBWtTx

https://www.w3schools.com/js/js_callback.asp

Thank you!