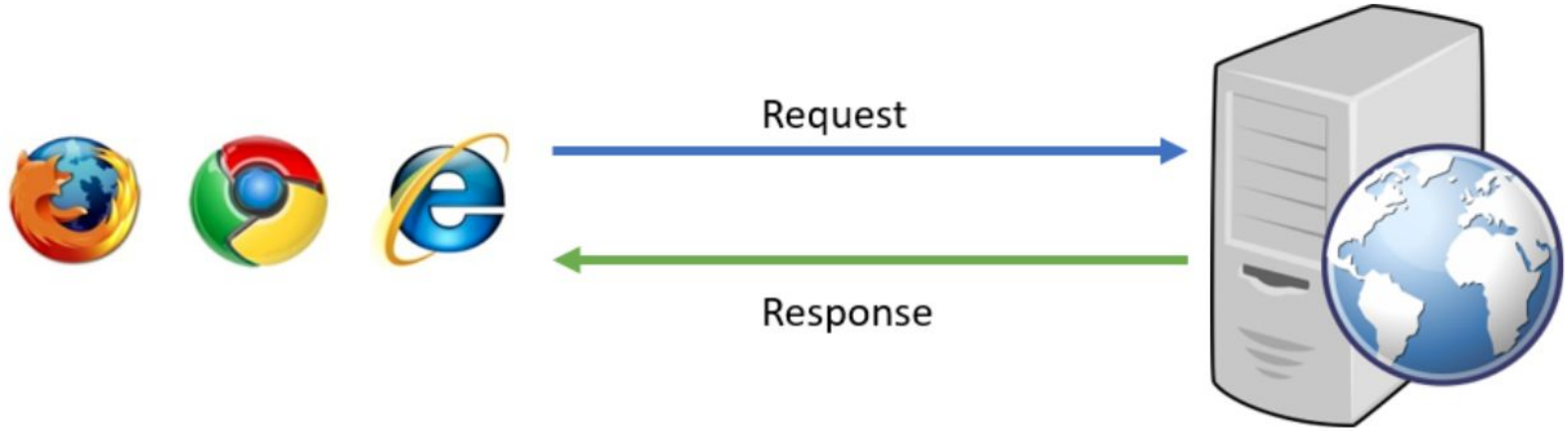


Module 3 - Lecture 10


Accessing Web APIs with JavaScript



HTTP Revisited



Live Score Tracking (2006)

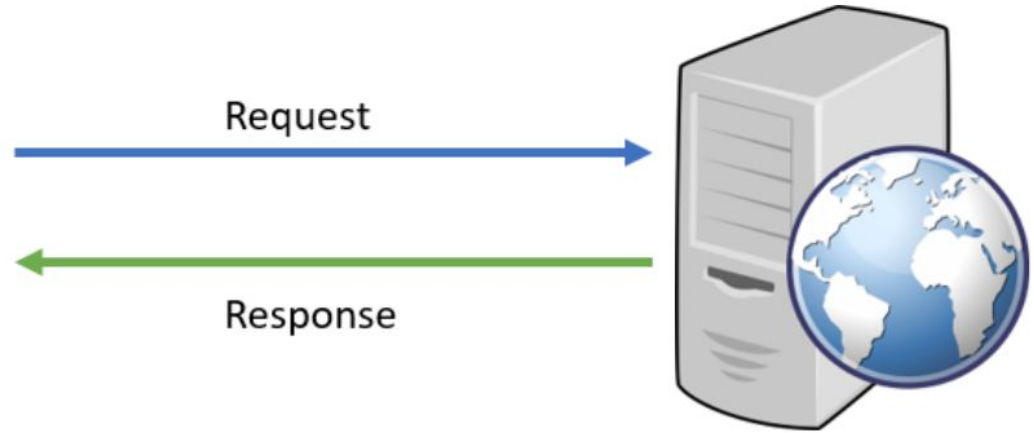
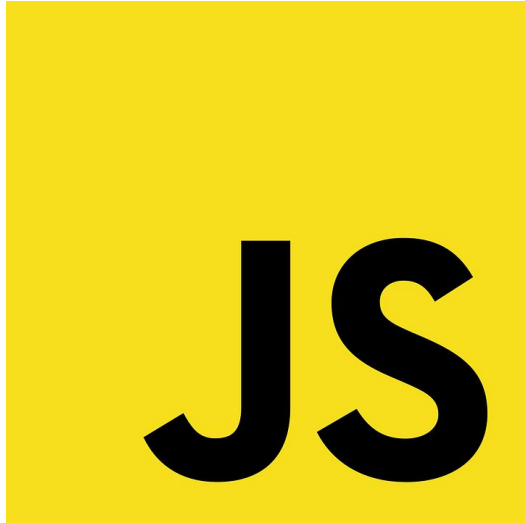


ESPN - NHL RealTime Scoreboard									
ESPN REALTIME		NHL Scoreboard		Thursday, January 12, 2006		Schedule Standings NHL Insider			
Montreal	0		Detroit	1	10:35	Vancouver	10:30	Toronto	4
Boston	3	4 Final	Colorado	2	2nd Ptd.	Phoenix	PM ET	Philadelphia	5
									4 Final
San Jose	10:30		Buffalo	2	9:36	Florida	11:35	Columbus	3
Los Angeles	PM ET		Pittsburgh	1	3rd Ptd.	Nashville	3rd Ptd.	Chicago	3
									3:22
									2nd Ptd.
Feedback About ESPN RealTime Message Board Copyright © 2004 ESPN.com									

How does this work?



HTTP (2006)



Fetch API

- The Fetch API provides an interface for fetching resources.
- It uses a Request / Response model.
- It can work with other protocols other than HTTP, but we'll ignore those for now.
- Fetch API is somewhat equivalent to Spring's RestTemplate.



Asynchronous Programming

“Have a seat. We’ll bring out your order when it’s finished”



What's wrong with this?

```
function takeOrder() {  
    // take down the order and return it  
}  
  
function cookOrder(orderRequest) {  
    // prepare order per the request and return it|  
}  
  
function serveOrder(customer, cookedOrder) {  
    // serve cooked order to customer  
}  
  
function doRestaurant() {  
    customers.forEach(customer => {  
        const orderRequest = takeOrder(customer);  
        const cookedOrder = cookOrder(orderRequest);  
        serveOrder(customer, cookedOrder);  
    });  
}
```



Promises

- A promise to supply a value at some later point.
- Allows you to associate handlers with an asynchronous action's eventual success value or failure reason.
- 3 states of a Promise
 - ***pending***: initial state, neither fulfilled nor rejected.
 - ***fulfilled***: meaning that the operation was completed successfully.
 - ***rejected***: meaning that the operation failed.
 -
- .then() for accessing returned Promise
- .catch() for handling errors



Asynchronous Approach

```
function takeOrder() {  
    // take down the order and return it  
}  
  
function cookOrder(orderRequest) {  
    // prepare order per the request and return it  
}  
  
function serveOrder(customer, cookedOrder) {  
    // serve cooked order to customer  
}  
  
function doRestaurant() {  
    customers.forEach(customer => {  
        takeOrder(customer)  
            .then((orderRequest) => {  
                return cookOrder(orderRequest);  
            })  
            .then((cookedOrder) => {  
                serveOrder(customer, cookedOrder);  
            });  
    });  
}
```



Cross Origin Resource Sharing (CORS)

- Your browser enforces a policy that prevents requests from going to a different domain than the current one.
- The web server has influence over this. They can whitelist domains to permit them through.
- In Spring this can be done using the annotation **@CrossOrigin**
 - This can be added to the Controller or Method Handler.

```
@CrossOrigin(origins = { "http://127.0.0.1:5500", "http://localhost:5500" })
```



QUESTIONS?

