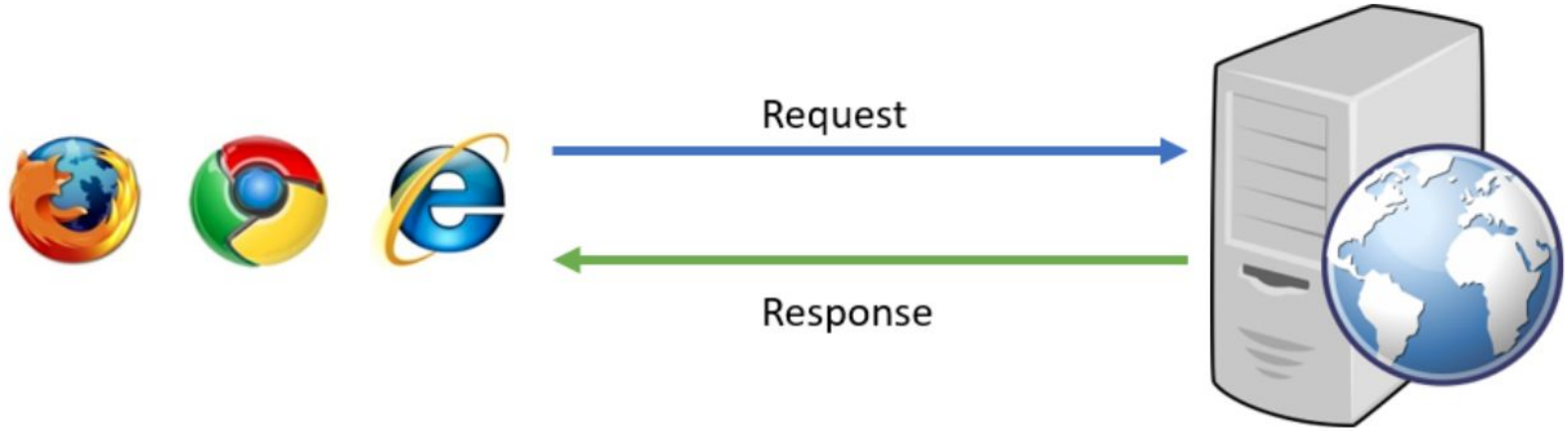


Module 3

Accessing Web APIs with JavaScript



HTTP Revisited



Live Score Tracking (2006)



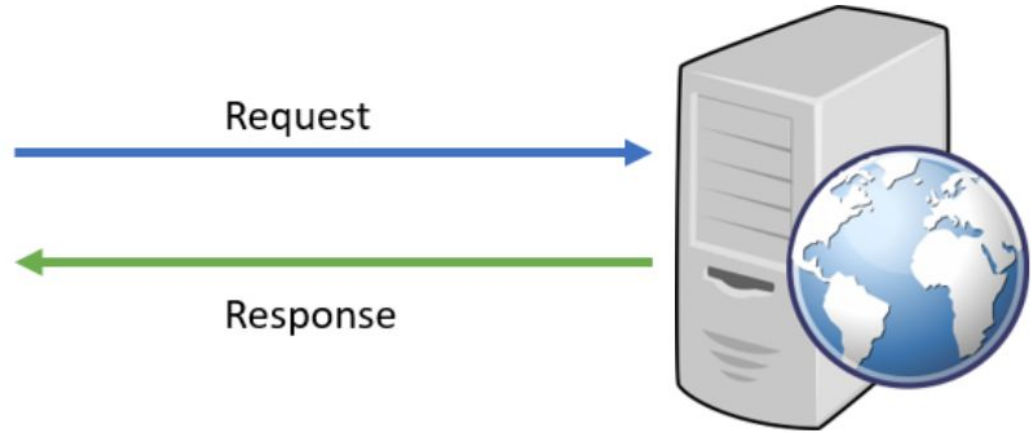
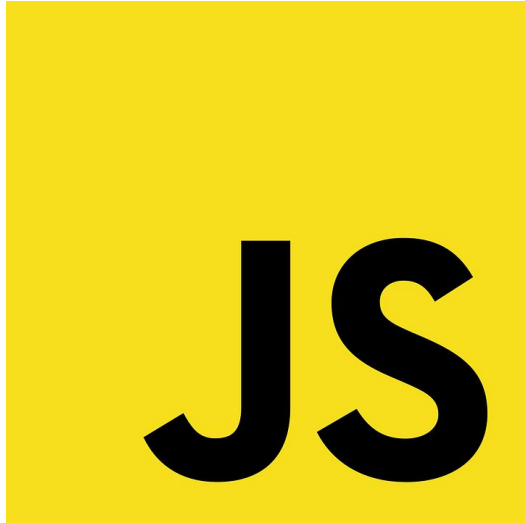
The screenshot shows a web browser window titled "ESPN - NHL RealTime Scoreboard". The page features the ESPN logo and "REALTIME" text. The main heading is "NHL Scoreboard" followed by the date "Thursday, January 12, 2006". Navigation links include "Schedule", "Standings", and "NHL Insider". The scoreboard is organized into a grid of game boxes, each showing the home team, away team, score, and game status. Some games are in progress, indicated by a blue highlight and a clock. The footer contains links for "Feedback", "About ESPN RealTime", and "Message Board", along with a copyright notice for 2004 ESPN.com and the "insider" logo.

Game	Home Team	Away Team	Score	Status	Time
1	Montreal	Boston	0 - 3	4 Final	
2	Detroit	Colorado	1 - 2	2nd Per.	10:35 PM ET
3	Vancouver	Phoenix			10:30 PM ET
4	Toronto	Philadelphia	4 - 5	4 Final	
5	San Jose	Los Angeles			10:30 PM ET
6	Buffalo	Pittsburgh	2 - 1	3rd Per.	9:36 PM ET
7	Florida	Nashville	3 - 0	3rd Per.	11:35 PM ET
8	Columbus	Chicago	3 - 3	3:22 2nd Per.	

How does this work?



HTTP (2006)



Fetch API

- The Fetch API provides an interface for fetching resources.
- It uses a Request / Response model.
- 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?

