Web Programming

JavaScript 11

Prof. Josué Obregón
Department of Industrial Engineering- ITM
Seoul National University of Science and Technology

Objectives

- Identify and describe more complex JavaScript statements that are used for manipulating local storage and handling asynchronous server calls.
- Apply the acquired knowledge of JavaScript to construct dynamic web pages.

Agenda

- Create DOM elements
- Local Storage
- JSON
- AJAX

DOM element creation: `createElement`

```JS
const li = document.createElement('li');
li.innerHTML = project;
document.querySelector('#projects').append(li);
```

- The `createElement()` method creates an element node.
- It creates the HTML element specified by *tagName*

Accessing local storage: `localStorage`

```JS
localStorage.getItem('counter');
localStorage.setItem('counter’,0);
localStorage.removeItem('counter’);
localStorage.clear();
```

- The `localStorage` object allows you to save key/value pairs in the browser.
- The `localStorage` object stores data with no expiration date
  - The data is not deleted when the browser is closed, and are available in future sessions
- The `sessionStorage` object stores data for only one session
  - The data is deleted when the browser is closed

JavaScript object syntax

```JS
let person = {
            name: "Philip J. Fry",                          // string
            age: 23,                                        // number
            "weight": 172.5,                                // number
            friends: ["Farnsworth", "Hermes", "Zoidberg"],  // array
            };

person.age;
person["weight"];
person.friends[2];
```

- in JavaScript, you can create a new object without creating a class
- can refer to the fields with .fieldName or ["fieldName"] syntax

JavaScript Object Notation (JSON)

- JavaScript Object Notation (JSON): Data format that represents data as a set of JavaScript objects
  - natively supported by all modern browsers (and libraries to support it in old ones)
  - ~~not yet as popular as XML, but steadily rising due to its simplicity and ease of use~~
  - JSON is a lightweight substitute for XML and it is more popular than XML because of JavaScript's dominance as the most widely used language of today.

Example of JSON

```
{
    "origin": "New York",
    "destination": "London",
    "duration": 415
}
```

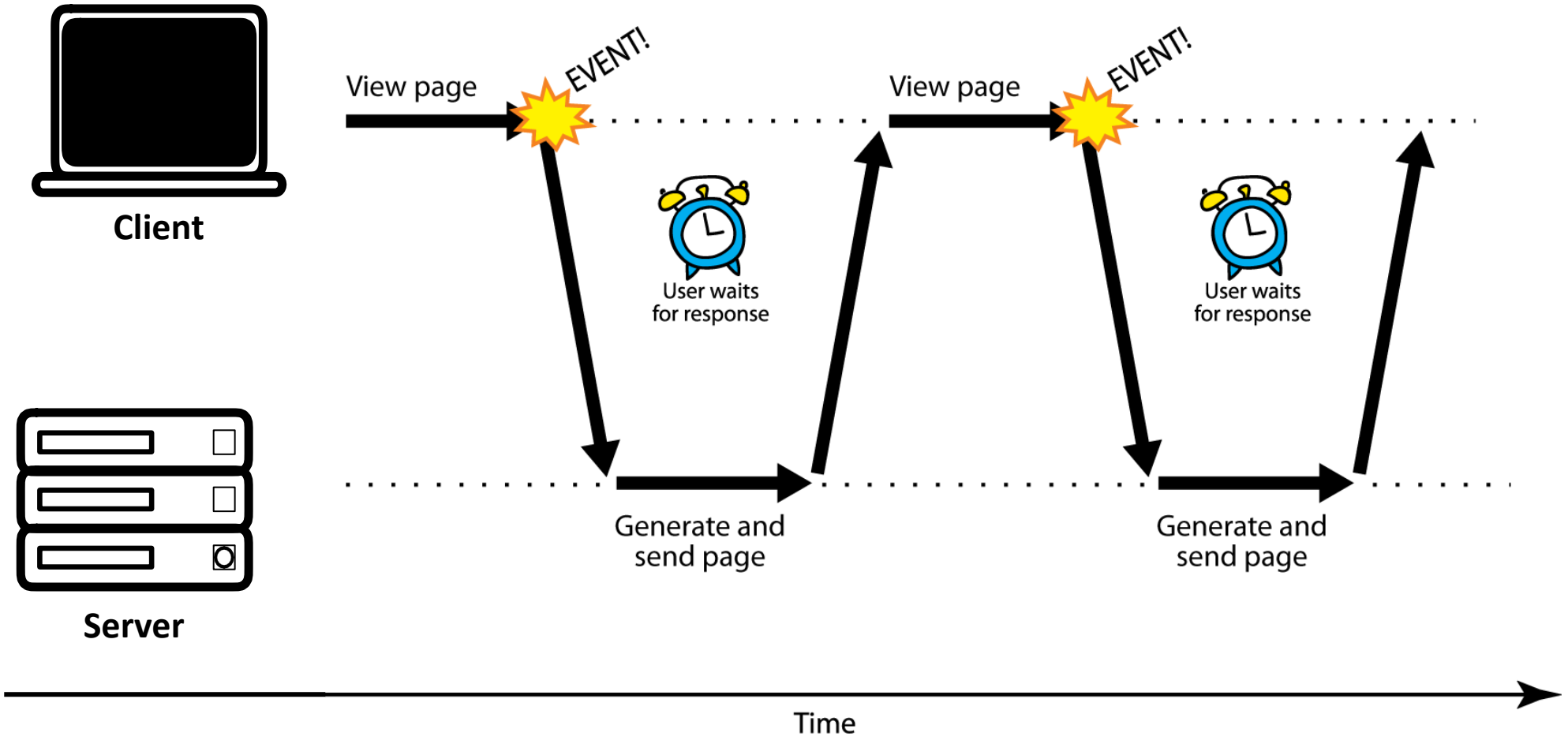
```
{
    "origin": {
        "city": "New York",
        "code": "JFK"
    },
    "destination": {
        "city": "London",
        "code": "LHR"
    },
    "duration": 415
}
```

JSON format for exchange rate API

```
{
    "rates": {
        "EUR": 0.907,
        "JPY": 109.716,
        "GBP": 0.766,
        "AUD": 1.479,

    },
    "base_code": { "USD"
}
```

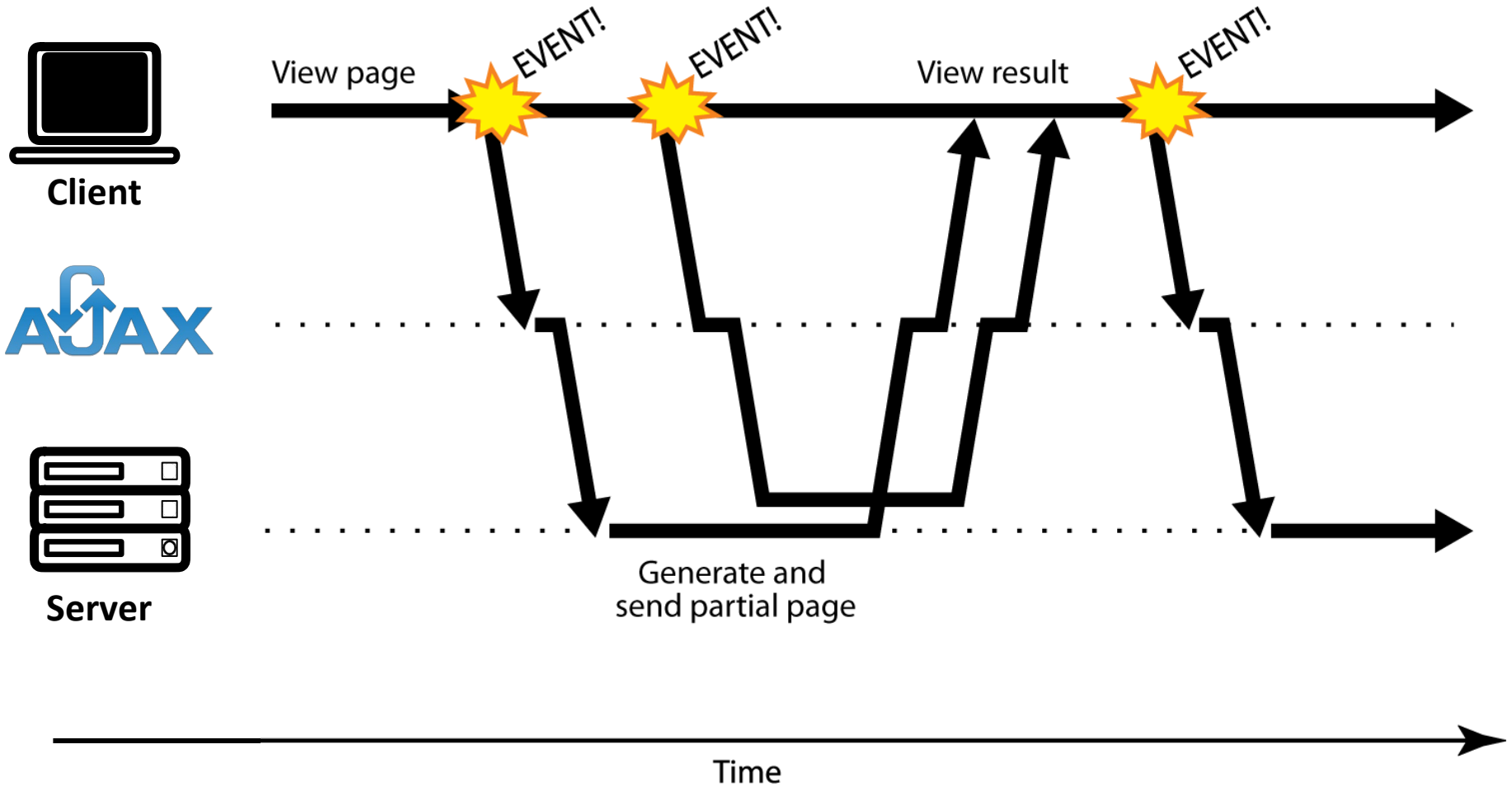
https://open.er-api.com/v6/latest/USD



- **Synchronous**: user must wait while new pages load
  - The typical communication pattern used in web pages (click, wait, refresh)

Web applications and AJAX

- **web application**: a dynamic web site that mimics the feel of a desktop app
  - presents a continuous user experience rather than disjoint pages
  - examples: Gmail, Facebook, etc.
- **AJAX**: Asynchronous JavaScript and XML
  - not a programming language; a particular way of using JavaScript
  - downloads data from a server in the background
  - allows dynamically updating a page without making the user wait
  - avoids the "click-wait-refresh" pattern



- Asynchronous: user can keep interacting with page while data loads
  - communication pattern made possible by AJAX
  - we can ask additional requests to our own server or third-party servers

How AJAX Works

- XMLHttpRequest (callback-based)
  - XMLHttpRequest (XHR) objects are used to interact with servers. You can retrieve data from a URL without having to do a full-page refresh.
  - This enables a Web page to update just part of a page without disrupting what the user is doing.
- Fetch API (promise-based)
  - It provides a global `fetch()` method that provides an easy, logical way to fetch resources asynchronously across the network.
  - Unlike XMLHttpRequest that is a callback-based API, Fetch is promise-based and provides a better alternative

Web Programming

Some of the content of this slide are based or adapted from the chapter 8 slides of the book "Web Programming Step by Step" by Marty Stepp, Jessica Miller and Victoria Kirst.