Hospital Information Systems

Unit 2: Information Presentation
Part 2 – Information Presentation

Master in Biomedical Engineering

Rodrigo García Carmona

Universidad CEU San Pablo
Escuela Politécnica Superior
Departamento de Tecnologías de la Información

Table of Contents

1. Introduction
2. HTML
3. CSS
4. JavaScript
5. Ajax

INTRODUCTION

Presentation Tier

- Web-based software deployments, that use the browser as the user interface, have become one of the most popular ways of providing software applications to customers.
- The presentation tier (presentation logic and client tiers) produces what users see and interact with via the browser in a web application.
- It's more than a bunch of hyperlinks, the "stickiness" and usability can make or break a business.
- A high-quality website has:
  - Useful functionality.
  - A visually appealing layout.
  - An intuitive navigation structure.

User Interface Design

- Designing a good layout requires graphic designing skills.
- The complexity on the client side has increased significantly over the years:
  - Cookies for session management.
  - Client-side validations for data entry errors.
  - Client-side JavaScript for interactivity.
  - Frameworks that provide modern layout and responsive design.
- Even if you aren't involved in user interface design, it's important to have knowledge of the design patters and principles that underlie client-side technologies.
- Languages:
  - HTML, used for specifying web pages.
  - CSS, used for styling web pages.
  - JavaScript, used for adding behavior to web pages.
  - Ajax, to provide dynamic content.

Good and bad web design

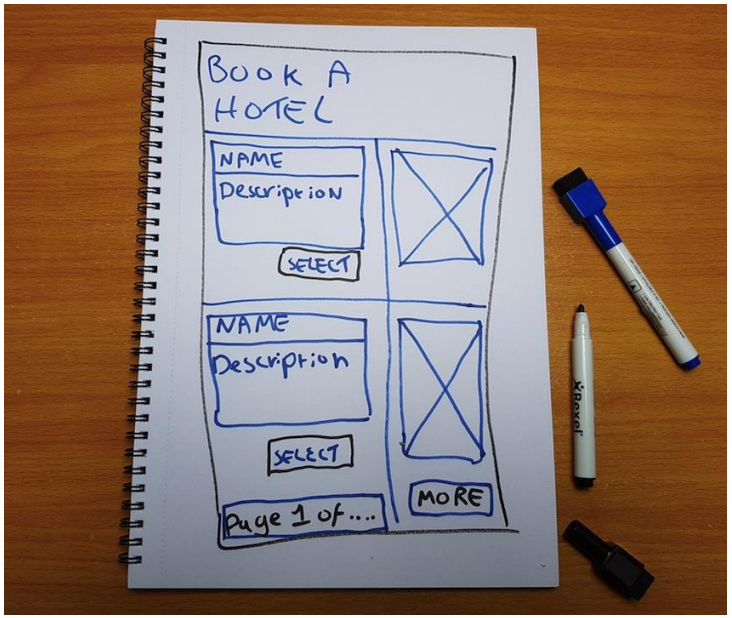
- The good:
  - Big text
  - Grid design
  - High-quality images
  - Responsive design
- The bad:
  - http://www.theworldsworstwebsiteever.com/
- Tasks for next class:
  - Find a web that you like.
  - Find a web that you dislike.
  - Explain why.

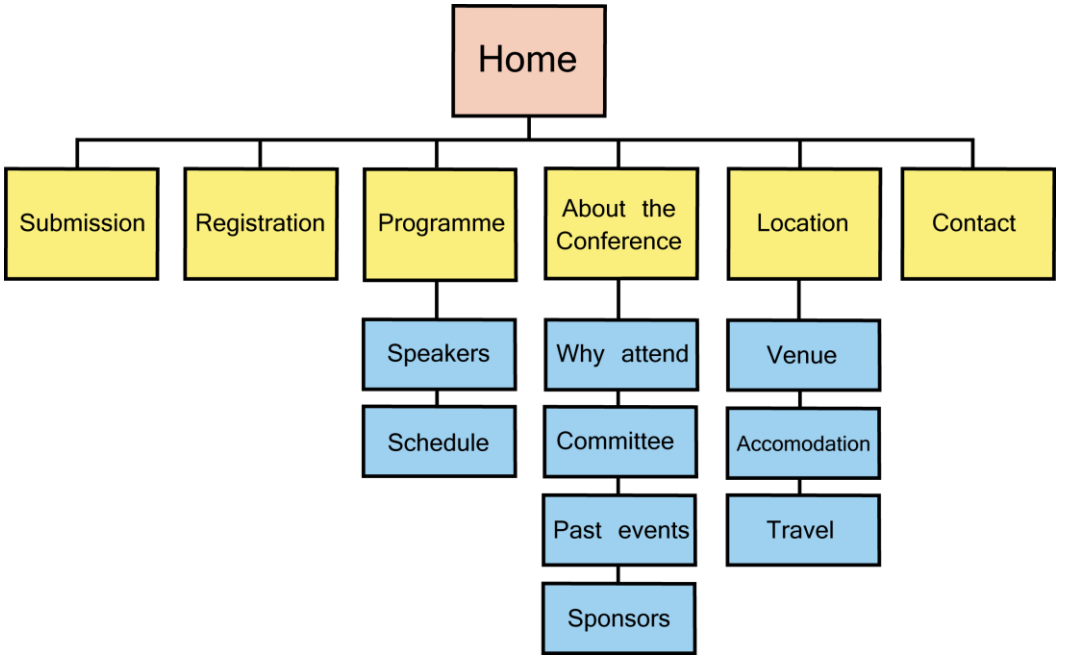
Layout

- Layout is how your content appears on screen.
- The trend now is to put a lot of things that you can find by scrolling down.
- Responsive design:
  - The goal is to provide an optimal viewing and interaction experience across a wide range of devices, from desktop monitors to mobile phones.
  - Adapt the layout to support easy reading and navigation with a minimum of resizing, panning and scrolling. No matter in which platform the content is being watched on.
  - The design uses fluid, proportion-based grids, flexible images, and CSS3 media queries.
  - More and more people are accessing webs through a phone. In fact, a good responsive web application can completely substitute a mobile app.

Layout – Wireframing and Structure

- Wireframing: Ideal for showing to the customer what the web page is going to end looking like and get feedback from them.
- Structure: The customer gets the sense of how things flow from one page to the next.





HTML

HTML, History and Philosophy

I just had to take the hypertext idea and connect it to the TCP and DNS ideas and – ta-da! – the World Wide Web.

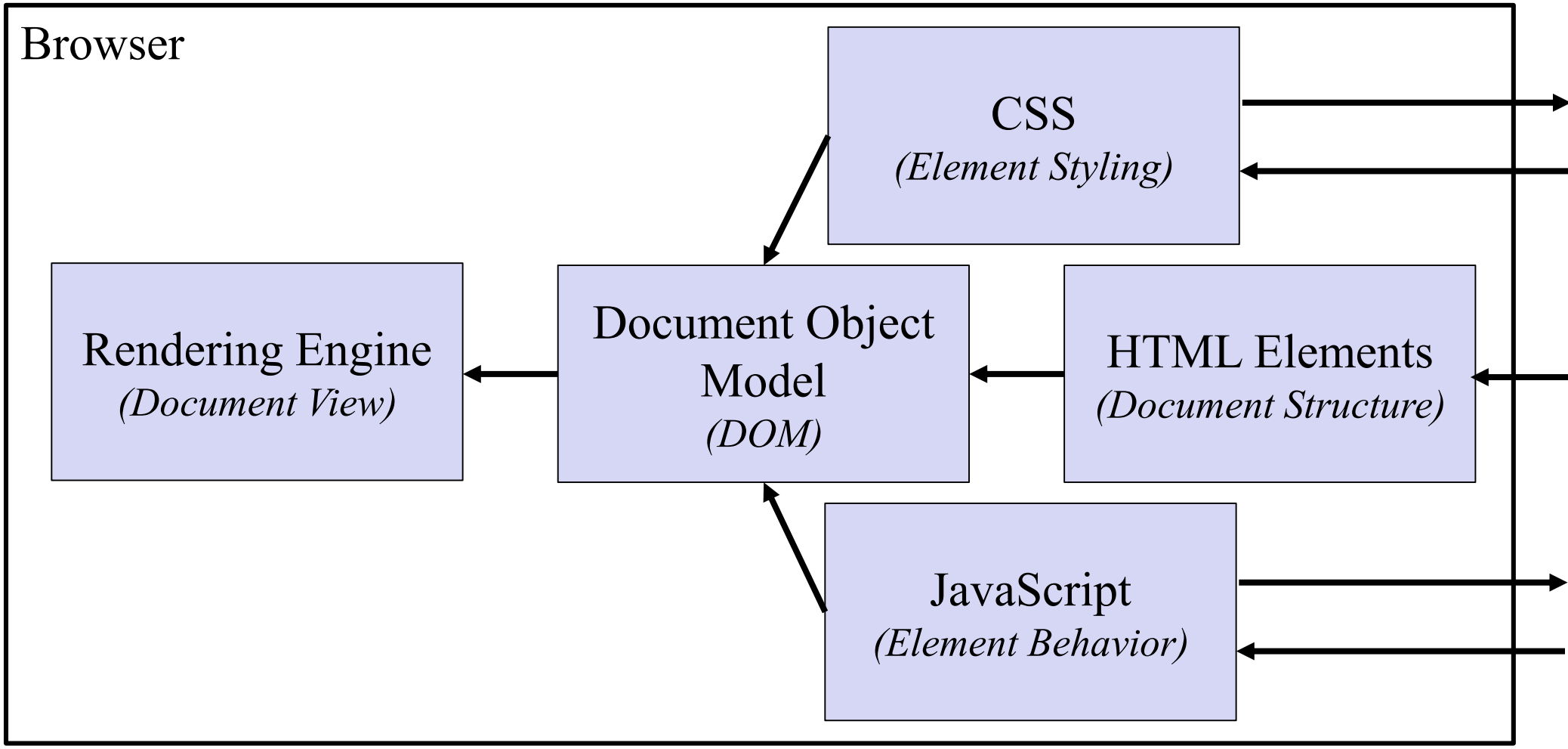
Tim Berners-Lee

- The Hypertext Markup Language (HTML) was present in the document sharing system that Berners-Lee developed at CERN in the early 1990's.
- The HTML Working Group, created by the IETF released HTML 2.0 as a standard in 1995.
- Additional versions of this standard have since been created under the auspices of the W3C (World Wide Web Consortium).
- HTML5, released in 2008, incorporates new features such as video playback and drag-and-drop capabilities.
- If created properly, HTML documents should follow the separation of content and presentation principle.

HTML

- HTML is XML. Uses the same rules we have already talked about.
- All HTML elements support the **id** and **class** attributes. Each **id** must have a unique value within an HTML document. Many elements can share the same **class**.
- Semantic HTML:
  - There is a design principle related to publishing entitled separation of concern and presentation: how content is presented should be separated from the content itself.
  - This is a special case of the more general concept of separation of concerns.
  - Advantages of this design principle:
    - The same content can be rendered differently, depending upon the needs of the user (text, braille, etc.)
    - You can change in one place how a recurrent item in the document is styled. You can modify the whole look of a web page without changing its content.
    - By capturing the meaning of a document, further intelligent machine processing of the document is supported. You can do a semantic analysis.

HTML, CSS and JavaScript



The Anchor Element

- One of the most important HTML elements is the **anchor**: <a>
- It is used to create hyperlinks.
- Example:

```
<a href="http://www.uspceu.es">CEU</a>
```

- The **href** attribute is used to especifiy the hypertext reference, that is, the link. This will produce a link that looks like:

CEU

HTML Forms

- Forms are the most basic way the users can supply input in a web application.
- An HTML form is a section of a document that may contain normal markup, as well as special elements called **controls** (checkboxes, radio buttons, drop-down lists, file selection, etc.), as well as **labels** for the controls.
- Users "complete" a form by modifying its controls (entering text or making selections).
- When a completed form is submitted, its data is first processed by a **user agent**, running as part of the browser, before it is actually submitted to a **processing agent** (like a web server o a mail server) on the server side.
- Basic structure:

```
<form action="http://example.com/log" method="get">
    <!-- form controls and other HTML markup -->
    <input type="submit" value="Log In">
</form>
```

- The **action** attribute is used to specify the URL of the processing agent on the server side that will receive the data collected in the form.
- The **method** attribute is used to specify the HTTP request method that the user agent will use to send the data.

Data Validation

- Data validation is the process of ensuring a web application operates on clean, correct and useful data.
  - The user provides a valid email address, phone number, and so on.
  - Inputs like 646-67-67-67, +34646676767 and 646676767 are treated the same.
  - The “business rules” are not violated. For instance, you cannot assign a patient to a ward without doctors.
- Data input also provides an attack vector.
- Failure to validate client-side input (common error) enables:
  - SQL injection attacks.
  - Cross-site scripting attacks.
  - Buffer overflow attacks.
- You can catch this even before the data is sent, using JavaScript or HTML5. For instance, if the email address is valid. **Forms must be treated with special care**.
- You can also perform validations in the server, in the controller or in the model (makes more sense in the latter). For instance, if the email address is already in use.
- Finally, validation can be performed in the database as stored procedures. This are database dependent and not portable, so it’s only useful when several web apps use the same database. It is not recommended.

HTML Styling

- In earlier versions of HTML, formatting elements found their way into the specification. Eg: you could specify the font that should be used in a particular **h1** heading.
- Example:

```
<h1>
    <font face="arial" size="20" color="#ffffff">
        Introduction
    </font>
</h1>
```

- This clearly breaks the separation of content and presentation principle.
- **Don't ever do this anymore.** You should use CSS and JavaScript
- Styling rules (CSS) and/or behaviors (JavaScript) are attached to particular HTML elements according to their **id** and/or **class**.
- To learn more, visit: **w3schools.com**

HTML Document Structure

- Every HTML document has the following structure:

```
<!DOCTYPE html>
<html>
     <head>
          <!-- The document head -->
     </head>
     <body>
          <!-- The document body -->
     </body>
</html>
```

- The DOCTYPE declaration must be the first thing that appears in an HTML document.
- This declaration is not an HTML tag, it lets the web browser know the version of HTML the page is using. The one we showed before is for HTML 5. Older versions provide more information in the DOCTYPE.
- The <html> tag marks the opening of the outermost element associated with every HTML document.
- The <html> element is the root of the document. It is the container for all of the HTML elements in a document.
- The other HTML elements are organized as a tree relative to the root <html> element.

HTML Head (I)

- The <head> element is the container for head elements.
- A required head element is the document title, specified using the <title> element.
- Other common head elements include: <link>, <script>, <meta>.
  - <link> is used to specify the location of an external resource. This element is always empty, it can only contain attributes.
    - Example: `<link rel="stylesheet" href="theme.css">`
    - The **rel** attribute specifies the relationship between the document and the linked resource.
    - The **href** attribute specifies the location (URL) of the external resource. The URL may be:
      - Absolute: pointing to another web site.
      - Relative: pointing to a file within a web site.

HTML Head (II)

- <script> is used to define a client-side script (typically JavaScript) or to include one from a external source.
  - Example:

```
<script>
      Function message() {
            Document.write("Hello World")
      }
</script>
```

  - If **src** is present, the <script> element must be empty. Ex: <script src="myscript.js"></script>
- <meta> is used to provide metadata.
  - Metadata is data about data. It is machine parsable, but will not be displayed in the webpage.
  - Metadata can be used by browsers (e.g: when to refresh the webpage), search engines (keywords) or web services.
  - Example:

```
<head>
      <meta charset="UTF-8">
      <meta name="description" content="My page">
      <meta name="keywords" content="biomedical engineering">
      <meta name="author" content="Rodrigo Garcia">
</head>
```

HTML Body

- The <body> element contains the HTML elements that will actually be rendered in the browser, the actual “content” of the HTML document.
- Every HTML body element is classified as being either a **block-level** or a **text-level** (**inline**) element.
- Block-level elements define the major structure of a Web Page: headings, paragraphs, and so on. They always produce a new line in the document. is a block-level element.
- Inline elements define the minor structure of a Web Page: bold or emphasized text, and the like. They do not produce a new line in the document.

CSS

CSS

- Cascading Style Sheets (CSS) are the language for specifying the presentation semantics, that is, the style, of an HTML document.
- It involves creating rules that specify how particular HTML elements should appear.
- A CSS rule has the following syntax:
  - Selector {declaration}
  - The **selector** specifies the elements the rule applies to.
  - The **declaration** is a semicolon separated list of **property:value** pairs.
  - Example:

```
h1, h2 {
        font-weight:bold;
        font-family:arial;
        color:black
}
```

- Errors in CSS, like in HTML, are ignored.
- They are called cascading sheets because when a rule affects an element, it also affects its children elements.

CSS Rules

- Remember that the **class** attribute can be applied to almost any HTML element:

```
<div class="main"> … </div>
```

- This attribute can be used to associate CSS rules with HTML elements using the **class selector**:

```
div.main {
    background-color:white;
    font-family: arial, verdana, sans-serif
}
```

- Selectors allow you to specify the HTML elements that a particular CSS rules should apply to:
  - **Type selector:** applies to elements of a given type.

```
h1 { color:purple} /* h1 elements purple */
h2, b {color: red} /* h2 and b elements red */
```

  - **Class selector:** applies to elements of a given class.

```
p.main {font-style:normal} /* p elements of class main */
.main {color:red} /* all elements of class main */
```

  - **Id selector:** applies to elements with a given id.

```
#chapter1 {text-align:center } /* elements with chapter1 as id */
```

JAVASCRIPT

JavaScript

- JavaScript is a lightweight, interpreted programming language that was designed to be embedded within, and provide scripting capabilities to, any application.
- **Client-side JavaScript** combines the scripting capabilities of the JavaScript interpreter with the document object model (DOM) defined in the web browser, enabling executable content to be distributed over the web.
- All major browsers include a JavaScript interpreter, also called engine.

JavaScript Code

- You are not going to learn how to program with JavaScript in this course, but to give you a general idea, here is a very brief example.
- Fibonacci numbers script:

```
<script>
document.write(“<h2>Fibonacci Numbers</h2>”);
for (i=0, j=1, k=0, f=0; i<50; i++, f=j+k, j=k,
k=f) {
   document.write(“Fibonacci (“+ i + “) = “ + f);
   document.write(“<br>”);
}
</script>
```

JavaScript and Security

- Client-side JavaScript opens up the possibility for authors to deliver malicious scripts to the browser.
- Browsers guard against this using two strategies:
  - JavaScript code is run in a **sandbox** that only allows web-related actions to be performed, not general-purpose programming tasks, like writing to disk, creating files and so on.
  - JavaScript code is constrained by the **same origin policy**: scripts from one website do not have access to information such as usernames, passwords or cookies from other websites.

JavaScript Libraries

- Numerous JavaScript libraries, containing pre-written JavaScript code, have been developed for the purpose of making the development of JavaScript-based web applications easier.
- The most popular JavaScript library is **jQuery**.
- An entire ecosystem has been built up around jQuery, including companion libraries (like jQuery UI) and plug-ins.
- jQuery supports the notion of **unobtrusive JavaScript**; the separation of behavior from document structure.
- With unobtrusive JavaScript, you never embed any JavaScript expressions or statements within the body of an HTML document, either as attributes of HTML elements (such as onclick) or in script blocks. Instead, you put those in separate files, like with CSS.

jQuery

- The focus in jQuery is on retrieving elements from HTML pages, and performing operations over them.
- Elements are retrieved via selectors (same selectors as in CSS).
- To collect a group of elements, pass a selector to the jQuery function:

  ```
  jQuery(<selector>)
  ```

- Or just:

  ```
  $(<selector>)
  ```

- The jQuery function returns a JavaScript object containing an array of DOM elements matching the selector.
- The returned elements are “wrapped” with additional methods, and these elements are therefore referred to as the **wrapped set**.
- Example: `$(“div.byebye”).hide();`
  - This will hide all the div elements in the document that belongs to the class byebye.
- Many of the jQuery methods also return a wrapped set, so it’s common to chain methods in jQuery:
- Example: `$(“div.byeby”).hide().addClass(“removed”);`
  - This will add a new class, called removed, to the hidden div elements.

AJAX

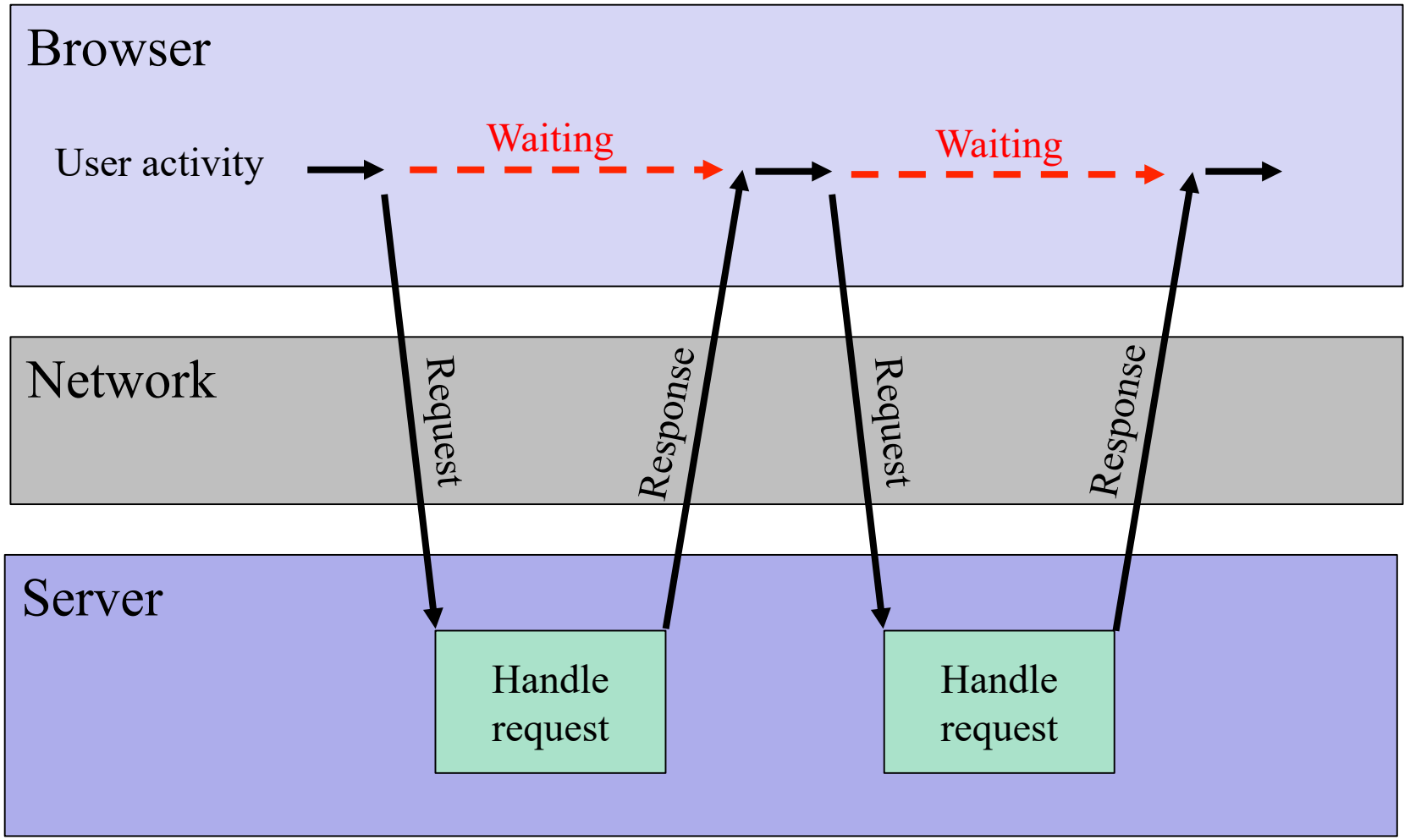
Dynamic Content and Ajax

- Technologies that enable dynamic content are the most important for turning the web into a platform for software applications.
- Ajax was originally an acronym that stood for Asynchronous JavaScript and XML (AJAX). Nowadays other technologies are also used.
- Ajax deals with how various technologies come together in order to provide a more interactive web browsing experience. It’s not a single technology, but a group of them working together.

Before Ajax

- Each time the browser made a request, it was forced to wait until it received a response.
- While the browser was waiting, the user could not interact with it.
- Each response lead to a reloading of a web page, even if only a single HTML element was changed from the one already shown in the browser.

Before Ajax


Browser
User activity
Waiting
Waiting
Network
Request
Response
Request
Response
Server
Handle request
Handle request

How Ajax Works

- **XMLHttpRequest** (XHR) is an API available to web browser scripting languages such as JavaScript. It's used to send asynchronous HTTP requests to a web server and then load the server response data back into the script.
- The user no longer has to wait for the response..
- Despite its name, an XMLHttpRequest is not limited to retrieve only XML texts from the server, any text format can be used. Nowadays it's common to retrieve JSON files, HTML, JavaScript or just plain text data.
- The data in the script can then be used to alter the current document shown in the browser (through the DOM), without loading a new web page.

How Ajax Works