Client-side Scripting

Databases and Interfaces

Matthew Pike & Yuan Yao

University of Nottingham Ningbo China (UNNC)

Overview

This Lecture

- Understanding the Necessity of Client-Side Scripting
- Introduction to JavaScript fundamentals.
- Practical JavaScript Applications - basic client-side scripts using JavaScript.
- Understand the concept of event-driven programming.

! DBI Exam

For the DBI exam, you are not required to write JavaScript code. However, it's important to understand the concepts of Client-Side Scripting and Event-Driven Programming. If you are interested in learning more about JavaScript, you can find a number of resources online. We recommend the Mozilla Developer Network as a good starting point.

Recap: What makes a web page?

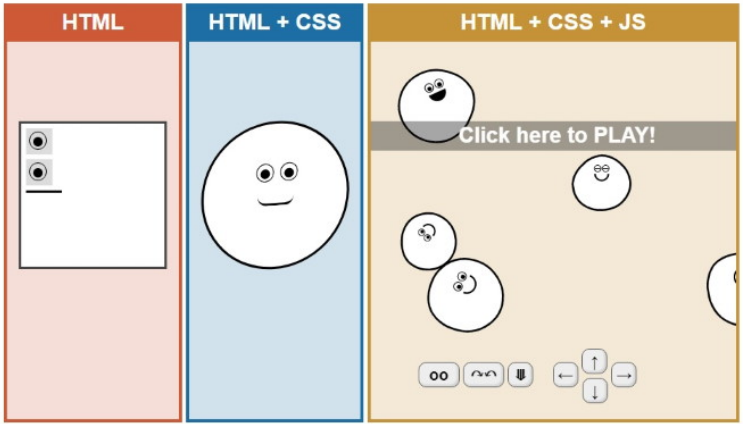

Figure 1: The role of HTML, CSS and JavaScript in a webpage.

What is Client-side Scripting?

- **Definition**: Execution of scripts in the user's web browser, enhancing web page interactivity.
- **Benefits**: Speeds up user interactions, reduces server load, allows real-time content updates.
- **Use Cases**: Form validation, interactive elements (like menus and sliders), dynamic content loading.
- **Considerations**:
  - Dependent on user's browser capabilities;
  - Security concerns as code is client-exposed;
  - Performance concerns as code is executed on the client's machine;
  - Possibility of users disabling scripts (no guarantee of execution).

Client-side Scripting vs Server-side Scripting

Aspect	Client-side Scripting	Server-side Scripting
Execution	In user's web browser	On web server
Languages	JavaScript	PHP, Python, Ruby, Java, .NET
Use	Enhancing user interface	Backend processes, databases
Load on Server	Reduces load	Increases load
User Experience	Immediate feedback, dynamic content	May require page reloads
Security	Code visible to user	Code hidden from user

Table 1: Client-side Scripting vs Server-side Scripting

When to Use Client-side Scripting?

Ideal Scenarios

- **Interactive User Interfaces**: Use for adding responsive elements like drop-down menus and modal pop-ups.
- **Instant Feedback**: Ideal for validating user inputs in forms in real-time.
- **Dynamic Content Updates**: Suitable for updating parts of a web page without a full page reload. Sometimes referred to as AJAX (Asynchronous JavaScript and XML).

Avoid in These Cases

- **Access to Secure Data**: Not suitable for tasks requiring access to server-side databases or sensitive information.
- **Dependence on User's System**: Avoid for critical functions that can be impacted if users have disabled JavaScript.
- **High-Reliability Operations**: Not recommended for processes that need to run reliably irrespective of the client's browser capabilities.

JavaScript

What is JavaScript?

- Lightweight programming language primarily used in web pages.
- Not the same as Java.
- First released by Netscape in 1995.
- Abbreviated as JS.
- Supported by all major web browsers and is a web standard.

What can JavaScript do?

- Add interactivity to a web page, reacting to user, page, and state events.
- Update web page content without reloading the page using the Document Object Model (DOM).
- Change content, style, and attributes of HTML elements.
- Send and receive data from a web server without reloading the page.
- Create cookies to store information about the user on their computer.
- Detect the user's browser and operating system.
- Validate user input before sending it to the server.

JavaScript: Hello World

- Embedded in HTML with `<script>` tag.
- Included directly in HTML or via external file with `<script src="myScript.js">`.

```
<!DOCTYPE html>
<html>
    <head>
        <title>JavaScript: Hello World</title>
    </head>
    <body>
        <h1>JavaScript: Hello World</h1>
        <script>
            alert("Hello World!");
        </script>
    </body>
</html>
```

Event-driven Programming

What is Event-driven Programming?

- Event-driven programming is a paradigm in which the flow of the program is determined by events such as user actions (mouse clicks, key presses), sensor outputs, or messages from other programs or threads
- Differs from procedural programming, where the flow follows a sequence of statements.
- In JavaScript, event handling typically involves assigning functions to event handlers.
- Example in JavaScript:

```
# Assign a function to the onclick event handler of a button
button = document.getElementById("myButton");
# When the button is clicked, the function is executed
button.onclick = function() {
    alert("Hello World!");
}
```

Event-driven Programming: Example

```
<!DOCTYPE html>
<html>
    <head> <title>Event-driven Programming: Example</title> </head>
    <body>
        <h1>Event-driven Programming: Example</h1>
        <button id="myButton">Click Me!</button>
        <script>
            button = document.getElementById("myButton");
            button.onclick = function() {
                alert("Hello World!");
            }
        </script>
    </body>
</html>
```

Form Validation with JavaScript

Form Validation

- JavaScript can validate user input before sending it to the server, enhancing user experience and reducing server load.
- Server-side validation is also necessary as JavaScript can be disabled in browsers; client-side validation alone is insufficient.
- For form validation, JavaScript can use the `onsubmit` event for validation before submission or `oninput` for real-time validation while typing.
- JavaScript can also be used to change the contents of the form based on user input. For example:
  - Showing or hiding form elements based on user input;
  - Highlighting invalid form elements in red;
  - Showing a message to the user when the form is submitted.

Form Validation: Example (HTML)

- The `onsubmit` event is triggered when the form is submitted
- The `validateForm()` function is called when the form is submitted - if it returns `true`, the form is submitted, otherwise it is not

```
<h1>Form Validation</h1>
<form id="myForm" onsubmit="return validateForm()">
    <input type="text" id="name" placeholder="Name">
    <input type="email" id="email" placeholder="Email">
    <input type="submit" value="Submit">
</form>
```

Form Validation: Example (JavaScript)

```javascript
function validateForm() {
    var name = document.getElementById("name").value;
    var email = document.getElementById("email").value;
    if (name.length < 3) {
        alert("Name is too short");
        return false;
    }
    if (email.length < 3 || !email.includes("@")) {
        alert("Enter a valid email");
        return false;
    }
    return true;
}
```

Interacting with the DOM

Recap: The Document Object Model (DOM)

- DOM is a programming interface for HTML/XML documents.
- Represents documents as nodes and objects.
- Allows programming languages (JavaScript) to dynamically change document structure, style, and content.
- Enables real-time content updates without reloading the page.

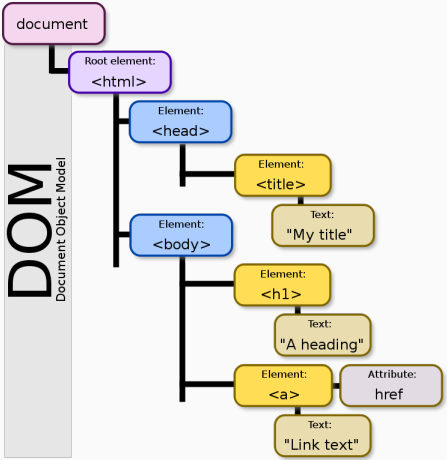

Figure 2: A visual representation of the DOM

DOM Interaction with JavaScript

- JavaScript interaction with the DOM can change content, style, or attributes of HTML elements.
- It allows for adding new HTML elements and introducing event handlers.
- JavaScript finds HTML elements by id, class name, or tag name using the `document` object, e.g.:
  - `document.getElementById(id)`: returns the element with the given id
  - `document.getElementsByClassName(className)`: returns a list of elements with the given class name
  - `document.getElementsByTagName(tagName)`: returns a list of elements with the given tag name

DOM Interaction with JavaScript: Example

```
<p id="myParagraph">Original text.</p>
<script>
    // Function to change the text of the paragraph
    function changeText() {
        // Retrieve the paragraph element by its ID
        var paragraph = document.getElementById("myParagraph");
        // Update the inner HTML of the paragraph
        paragraph.innerHTML = "Updated text after 5 seconds.";
    }
    // Call changeText function after a delay of 5 seconds
    setTimeout(changeText, 5000);
</script>
```

Resources

- Mozilla Developer Network
  - https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Learn JavaScript Online
  - https://learnjavascript.online
- https://javascript.info