DBI-016
Client-Side Scripting

COMP1048: Databases and Interfaces

Matthew Pike (matthew.pike@nottingham.edu.cn)

Dylan Shen (linlin.shen@nottingham.edu.cn)

Today

- Identify when Client-Side scripting is necessary
- Introduce JavaScript
- Write (simple) client-side scripts using JavaScript
- Understand Event-Driven Programming

[Review] What makes a webpage?

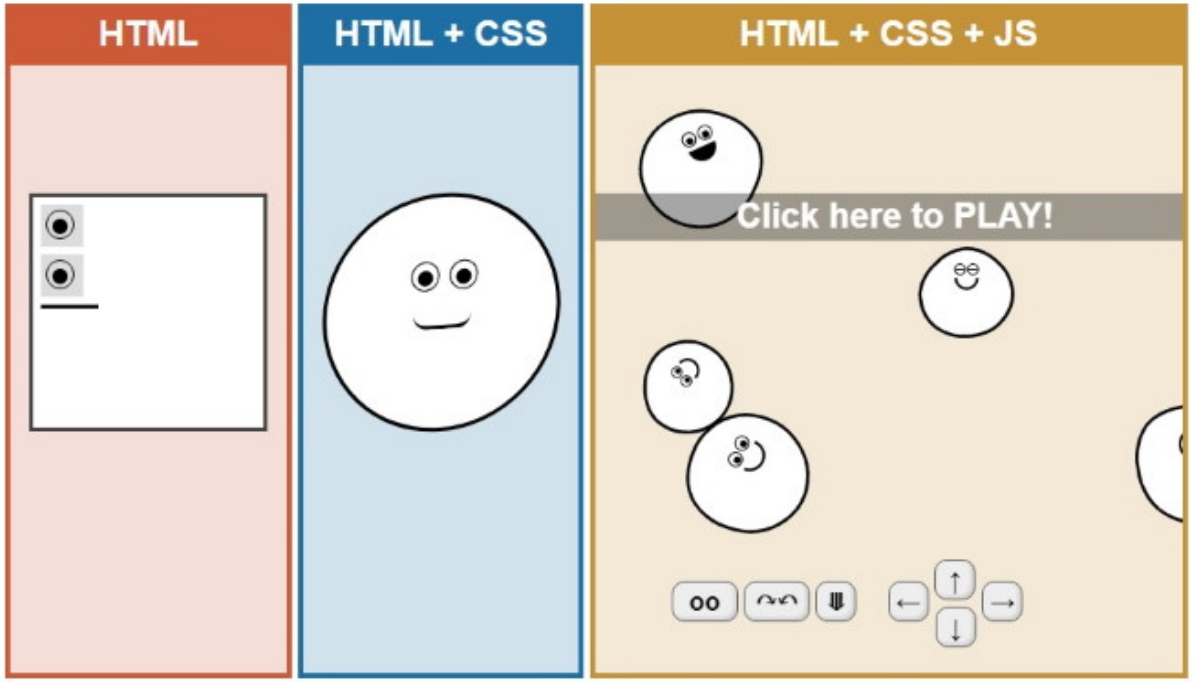

Image Source: https://html-css-js.com/images/og.jpg

Client-Side vs Server-Side

- A client-side script runs in the user's browser **when particular events occur**
  - Useful when we don't want to send information to the server, but still perform some processing of the data
- A server-side script runs on the web server **when the user requests information**
  - Necessary when interacting with privileged or authenticated resources e.g. Server File-Systems or Database Servers

When to use Client-Side Scripting

- Yes
  - Providing quick feedback to users on their (form) input
  - Dynamic content loading/update
  - Introduce interactive components to pages
- No
  - Tasks which require privileged access to remote databases
  - If access to user's hard drive is required (also, server-side scripting is not appropriate here)
  - Operations which require some guarantee of running (users may disable client-side scripting)

JavaScript

- JavaScript is a lightweight scripting language
- Originally developed by Netscape
- JavaScript != Java
- Often abbreviated as JS
- It is a web-standard, but not supported equally by all browsers

What can JavaScript do?

- **React to events** - Do something when the user clicks a button.
- **Update without refreshing** - AJAX enables developers to retrieve new data in the background.
- **Validate data** - Check form data is valid **without** sending to the server.
- **Create Cookies** - Useful for saving user preferences.
- **Detect the user's browser** - Some browsers support different functionalities.
- **Read and Write to the DOM** - Dynamically change the content and structure of a page.

JavaScript Hello World

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <script>
      alert("Hello World");
    </script>
  </body>
</html>
```

Integrating JavaScript + HTML

Inline

```
  <script>
    alert("Hello World");
  </script>
```

External File

```
<script src="myJs.js"></script>
```

Event-Driven Programming

- A different paradigm of programming
- Responds to events, rather than running sequentially
- Example
  - Process a form's input when the user presses the 'Submit' button
  - Necessarily reactive, does not make sense to validate at any other time

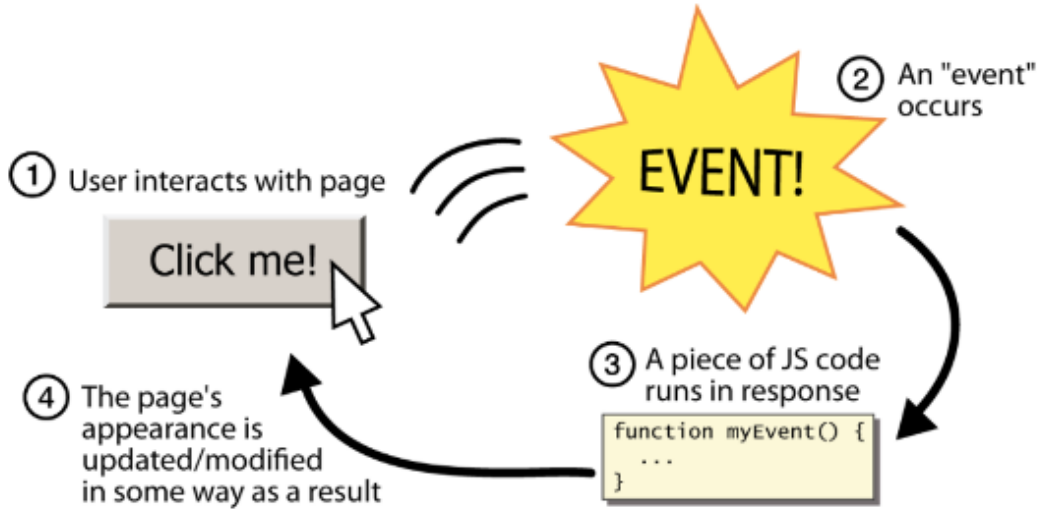

Image Credit - http://courses.cs.washington.edu/courses/cse190m/11sp/lectures/slides/images/figure_3_event.png

Event-Driven Programming in JavaScript

- We can tie JS code to user derived events
  - e.g. User interactions; browser focus/resize;
- JS can also respond to browser or system events
  - e.g. Timers; DOM mutation; Network status;
- This can also introduce interactivity to a web page

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <button onclick="Counter()">
    Click Me!
    </button>
    <script>
    count = 0;
    function Counter(){
      count++;
      alert(count);
    }
    </script>
  </body>
</html>
```

Interacting with the DOM via JavaScript

- Add, remove or update elements in the DOM using JS
- Access elements using:
  - `document.getElementById`
  - `document.getElementsByTagName`
  - `document.getElementsByClassNam e`
- Update HTML via `innerHTML`

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <p id="demo">
      JavaScript can change HTML content via the DOM.
    </p>
    <button onclick="Hello()">
    Click Me!
    </button>
    <script>
    function Hello(){
      document.getElementById('demo').innerHTML ='Hello!';
    }
    </script>
  </body>
</html>
```

Form Validation with JavaScript

- Form validation occurs on the client-side
  - Avoids sending invalid data to server
  - Fast - better user experience
- Not a replacement for server-side validation - this should still be performed.
  - Client-side scripting should not be considered 'secure'

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <label for="age">How old are you?</label>
    <input type="text" value="" id="age">
    <button onclick="Validate()">
      Validate
    </button>

    <script>
    function Validate(){
      age = Number.parseInt(document.getElementById('age').value);
      if (Number.isNaN(age)){
        alert("Invalid: Please enter a valid number");
      }
      else if (age < 0){
        alert("Invalid: Please enter a positive number");
      }
      else{
        alert(`Valid: You are ${age} years old.`);
      }
    }
    </script>
  </body>
</html>
```

Resources & Further Reading

- Mozilla Developer Network (MDN) - An Excellent Resource
  - The Basics of JavaScript
- JavaScript Specification
- Learn JavaScript Online