CLIENT SIDE JAVASCRIPT

Stephen Schaub

Client Side JavaScript

- HTML pages can contain JavaScript code that interacts with elements on the page
- Interpreter in browser executes JavaScript code when user triggers events

```
<html>
   <body>
      <input type="button" value="Click me" onclick="alert('hello, world!')">
   </body>
</html>
```

JavaScript in Web Pages

Three approaches:

- Embed JavaScript directly in event handlers
- Event handlers call functions defined in <head>
- Event handlers call functions defined in external JavaScript file (preferred)

hello_jshead.html

```
<html>
<head>
   <script language="javascript">
      function hello() {
         alert("Hello, world!");
      }
   </script>
</head>
<body>
   <input type="button" value="Click me"
      onclick="hello()">
</body>
</html>
```

External JavaScript

hello_extjs.html

```
<html>
<head>
    <script src="hello_extjs.js"></script>
</head>
<body>
    <input type="button" value="Click me" onclick="hello()">
</body>
</html>
```

hello_extjs.js

```
function hello() {
    alert("Hello, world!");
}
```

JavaScript Can Manipulate Elements

- Assign id to element in HTML
- Use document.getElementById in event handler to access HTML Element
- Manipulate HTML Element object via its properties and methods

hello_dom.html

```
<html>
<head>
   <script language="javascript">
      function hello() {
         var btn = document.getElementById("btn")
         btn.value = "You clicked me!";
      }
   </script>
</head>
<body>
   <input id="btn" type="button" value="Click me"
         onclick="hello()">
</body>
</html>
```

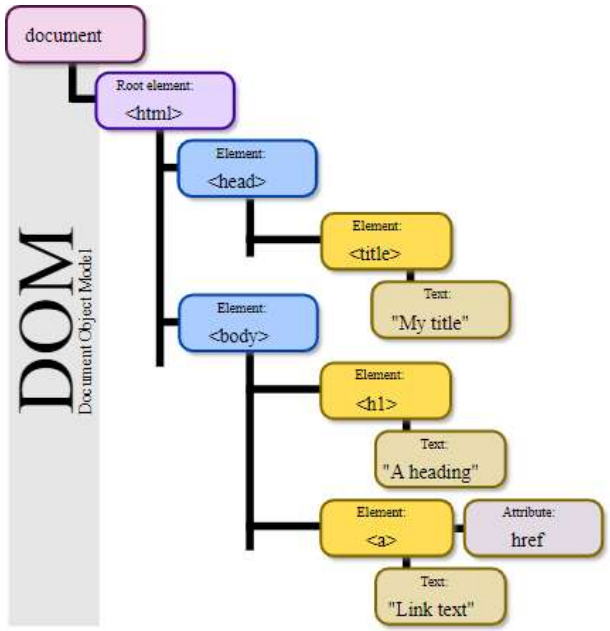
Demo

- Create examples/jscript/greet_finished.html
- Notes:
  - <p> element has an innerHTML property
  - <input> elements have value property

Document Object Model

- Browser constructs an object model that reflects the structure of the HTML page
- Each element in the HTML page has a corresponding object in the DOM
- JavaScript can access / manipulate elements by id

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="...">Link text</a>
  </body>
</html>
```



DOM Objects

- Important DOM objects include
  - window
  - document
  - HTML elements
- Properties and methods are defined by DOM API
  - https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

window object

- Provides access to the document
- Allows you to manipulate browser
- Properties and methods can be used with or without window prefix:
  - window.alert(“Hello”)

    or
  - alert(“Hello”)

window object

Key Properties

- document
- location

Key Methods

- alert()
- open()

document object

Key Properties

- cookie
- title
- forms[]

Key Methods

- getElementById()

Working with HTMLElement Objects

HTMLElement Objects

- Key properties:
  - style – allows you to manipulate style of element
    - var mypara = document.getElementById(“mypara”);
    - mypara.style.color = "red";
    - mypara.style.fontSize = 36;
    - See examples/jscript/style_manip.html
  - innerHTML – allows you to manipulate content of element
    - mypara.innerHTML = “Hello, world!”;

Accessing HTML Element Form Objects

Two ways to obtain a reference to an HTML form element:

- Use document.getElementById()
  - var txtFName = document.getElementById('txtFName')
- Access form elements by name using document.forms[] array
  - var txtFName = document.forms[0].fname

```
<html>
<body>
<form>
<input type="text" id="txtFName" name="fname">
...
```

HTMLElement Form Objects

Key Properties

- value
- disabled

Key Methods

- focus()

Event Handling

Important Events

- <input type="button" onclick="...">
  - Respond to a click event on a button
- <input type="text" onchange="...">
  - Respond when the user changes a value in a text control
- <body onload="...">
  - Perform work when page first loads
- <form onsubmit="return validateForm(this)">
  - Validate data on the form before it is submitted
- Other events: See DOM Event Reference
  - https://developer.mozilla.org/en-US/docs/Web/Events

Page Initialization Event

- Set focus on a form field when page loads:
  - <body onload="document.forms[0].txtFName.focus();">
- The DOM is fully constructed only after the document has finished loading
- Attempts to access DOM outside an event handler are unreliable

Ajax

Ajax

- Refers to techniques used to dynamically load data from a remote server into a web page displayed in the browser
- Browsers provide Fetch API that enables loading data into current page

How Ajax Works

1. Browser loads HTML page from server
2. JavaScript on page initiates HTTP request to server for additional data (called an "Ajax" request)
   - Data may be downloaded from a static file on server or generated by server-side logic
3. JavaScript receives result from server and injects downloaded information into page using DOM manipulation techniques

Fetch API

- Use window.fetch() send an HTTP request for data from server
  - let result = await window.fetch('ajaxdata.json')
  - let data = await result.json()
- Downloaded data is typically in JSON format
  - Other formats possible: Plain text, HTML, XML
- See examples/ajax/basicajax
  - index.html – Web page that initiates Ajax request
  - ajaxdata.json – Static file containing data downloaded by web page

Node.js, Express, and Ajax

- Express applications can conveniently produce JSON data to be consumed by client Ajax calls
- In a route callback function, use res.send() to send arrays and objects
  - Encodes arrays / objects to JSON
  - Sets Content-type header to application/json
- See examples/ajax/bootcloset

Ajax Security

- Browser security rules restrict where JavaScript on a page can send Ajax requests
- JavaScript on a page loaded from xyz.com can send Ajax requests to xyz.com
- Ajax requests to other servers are called "Cross-Origin" requests
  - Permitted only if the target server is configured to allow the Ajax request
  - See https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS for details