Events

- **Event** → Notification that something has occurred
- Example of situations that make the web browser generate an event
  - Browser finishes loading a document
  - When the user clicks on a button
  - When the user moves the mouse
  - Others
- **Event handler** (also known as event listener)
  - JavaScript function or code fragment that is executed when a particular event occurs
- **Event handler registration**
  - Associating an event handler with a particular event

Event-driven Programming

- **Normal (control flow-based) programming**
  - Approach
    - Start at main()
    - Continue until end of program or exit()
- **Event-driven programming**
  - Start at main()
  - Register event handlers
  - Await events & perform associated computation
- **GUIs (Graphical User Interfaces)**
  - Example of event-driven software

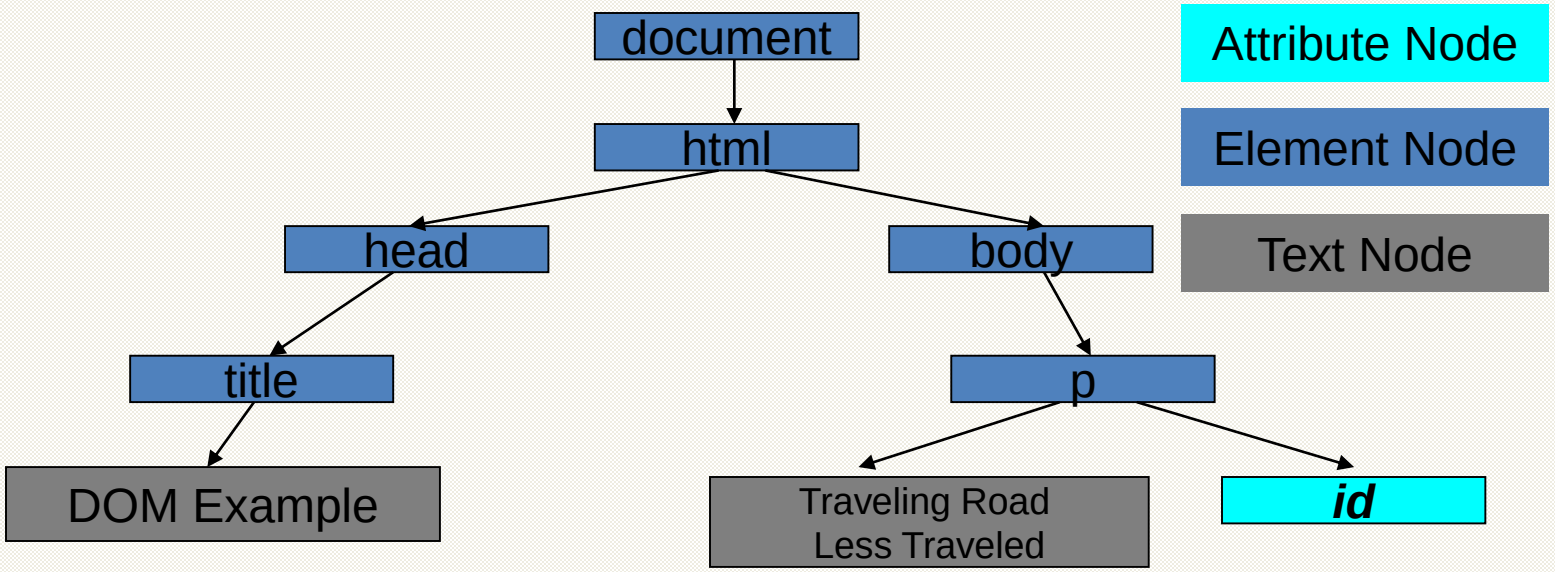
Event Handler Attributes for Most HTML

- ***Mouse Related***
  - **onclick** → mouse button is pressed and released
  - **ondblclick** → mouse button is double-click over element
  - **onmouseover** → mouse moves over element
  - **onmouseout** → mouse moves off element
  - **onmousemove** → mouse pointer is moved
  - **onmousedown** → mouse is pressed down while cursor is over the element
  - **onmouseup** → mouse is released while the cursor is over the element
- ***Keyboard Related***
  - **onkeypress** → key pressed and released
  - **onkeydown** → key is pressed
  - **onkeyup** → key is released
- ***Other***
  - Keep in mind that there additional handlers that are specific to certain tags. We will address those later on

DOM (Document Object Model)

- **DOM** → representation of the elements of a web page (e.g., headings, lists, paragraphs, styles, etc.) used by a JavaScript program to manipulate web page elements
- DOM represents elements of a web page as a tree structure consisting of nodes (next slide)

Example DOM for HTML File




```
<html>
    <head><title>DOM Example</title></head>
        <body>
            <p id="message">Traveling the road less traveled. </p>
         </body>
</html>
```

DOM (Document Object Model)

- To access any element of your web page you could traverse the tree
- Easier approach:
  - **document.getElementById** function
  - **document.getElementsByTagName** function
- **Example:** AverageComputationMVC.html

Forms

- Forms → means by which information passes from the user to a server
- For now we will use forms to read values to be processed by JavaScript
- **<form> tag**
  - Defines the form
  - It has two attributes → action and method
  - **action** → indicates where the form contents will be sent when the form is submitted
  - **method** → defines how the contents will be sent (post/get)
- **<input>tag**
  - Appears inside of the <form> tag
  - Defines several input data alternatives
  - The general format is → <input type="ALTERNATIVE" />
  - ALTERNATIVE can be text, password, checkbox, radio, file, submit, button, reset, hidden, others

Data Access

- We can access data in forms by using

  document.getElementById(“elementId”) ;

- getElementById returns a reference to an element that we can use to:
  - Retrieve the value of the element (e.g., text field in a form)

    var login = document.getElementById("loginId").value;
  - Set the function to call when an element is clicked on (e.g., button)

    document.getElementById("processButton").onclick = functionDoesProcessing;
  - Get/Set Attributes

    var imageElement = document.getElementById("myImage");

    var imageName = imageElement.getAttribute("src");

    imageElement.setAttribute("src", “imageFile.jpg”);
- **Example:** AssociateButtonWithFunction.html, GetValueInTextField.html
- **Example:** UpdateValueInTextField.html, GetSetAttribute.html

Form Examples

- **Reset**
  - The functionality of the Reset button is already provided by HTML. You don't need to add any JavaScript or define a button
  - You can change the text associated with the Reset button by using the value attribute
  - **Example:** GetValueInTextField.html
- **Textfield change**
  - **Example:** GetValueInTextFieldOnChange.html
- **Animation**
  - **Example:** Animation.html

Separating Content from Behavior

- **Keep each of the following separate**
  - Content (HTML)
  - Presentation (CSS)
  - Behavior (JavaScript)
- **Separating HTML and JavaScript**
  - Use src attribute of <script>
  - **Example:** folder named *separation*

Advantages of Separation

- Simplifies HTML files (removes large segments of JavaScript code)
- Functions can be cached by the browser for efficiency
- Code sharing/reuse → Function used by several pages can be kept in a single file

Use of JavaScript in Web Pages

- Unobtrusive JavaScript → programming paradigm specifying that JavaScript should not intrude on users accessing a web page, on HTML content, or CSS Stylesheets
- Goals
  - Keep JavaScript code separate from HTML
  - Scripts are enhancements to HTML content, but the content should be available without JavaScript
- You should design web pages so they operate even if JavaScript is disabled

CheckBoxes

- Allow us to make a selection
- Defined by using type="checkbox"
- We can tell whether an entry is selected by using the "checked" property
  - true → entry has been selected
- Default selection by using checked="checked"
- **Example:** Checkboxes.html
  - It is used when submitting data to a server

Radio Buttons

- Exist in groups and only one can be checked in a group
- Defined by using type="radio"
- We defined the radio buttons to be in the same group by using the same name for all of them
- Note: Do not use the same value for name and id
- Default selection by using checked="checked"
- We access the elements using arrays
- document.getElementsByName → returns array
- **Example:** RadioButtons.html

Drop-Down/Scrollable Lists

- Defined using the <select> tag (not the <input> tag)
- <option> tag to specify possible options
- To define a default choice use:

  selected="selected"

- The ***multiple*** attribute in <select> allows for selection of multiple items (usually displayed as a multiple-selection list)
- **Example:** DropDownList.html
- **Example:** ScrollableList.html
  - Notice you can have multiple default selections

Textareas

- Allows user to input more than one line of information
- We use the <textarea> tag
- ***rows*** and ***cols*** attributes define the text area
- Default text (if any) appears between the <textarea> and </textarea> tags
- **Example:** Textarea.html

innerHTML property

- Property defining:
  - HTML code
  - Text occurring between the opening and closing tags of the element
- Allows you to modify/define HTML content
- **Example:** GetContent.html
- **Example:** ModifyContent.html
  - What happens if you keep selecting the Modify button?

Getting String Characters

- The function charAt Allows us to retrieve the character associated with a particular index position in a string. Access is similar to array indexing (first character at 0)
- Example
  - var x = "Wednesday";
  - var secondCharacter = x.charAt(1); // variable has "e";
  - var lengthOfString = x.length; // variable has 9
- This function is helpful when trying to validate data
- **Example:** charAt.html

Default Values/Prompt

- **Example:** DefaultValue.html
- Notice that prompt can have a second argument that represents a default value

Precision

- **Example:** DecimalPrecision.html

Form Validation

- **Form data validation**
  - We can validate the data associated with a form by recognizing the submit event
  - window.onsubmit = validateData;
    - validateData is a function that will check for data validity
    - It will return true or false
- Keep in mind that JavaScript can be disabled therefore the server application must also validate the data
- **Example:** formValidation example
  - Notice the organization code (HTML, CSS, JS separate)
  - This example is representative of what you need to do for the project