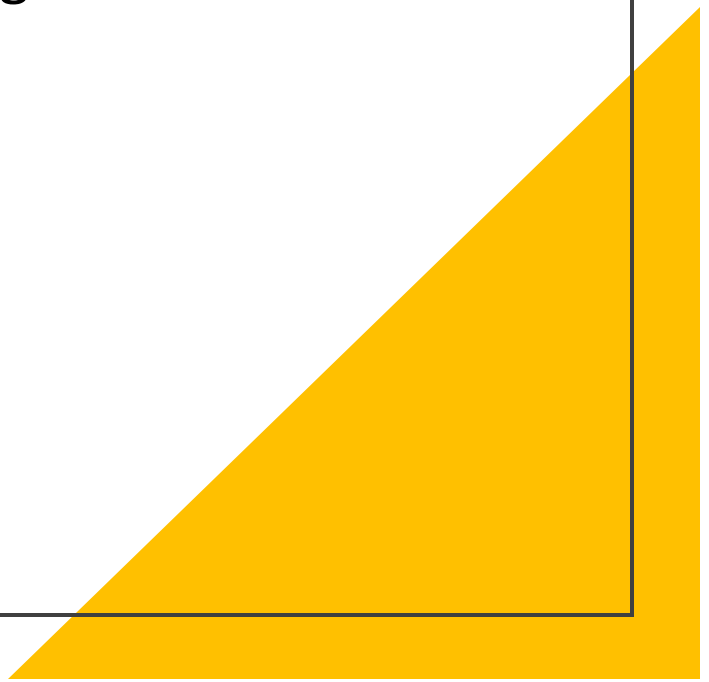
DOM

JavaScript in browser

JavaScript was originally developed to work with HTML pages in a browser application.



<script> element

- The <script> element is used to embed JavaScript codes.
- It can go anywhere in the HTML page, but by convention it is placed in the head section.

window

- When JavaScript runs on a browser, it runs inside the global environment called window (object).
  - `alert()` and `prompt()` are methods (functions) of window object for alerting output and displaying prompt for user input.

Running a JavaScript program

- The computer (browser) runs or executes JavaScript code when certain events happen.
  - When HTML document is loaded in a browser.
  - When user clicks a button
  - When user types in a textbox
  - and more…

HTML Event Attributes

```
<button onclick =
"doSomething()">Do it</button>


<script>
      function doSomething(){
            // code to do
something.
      }
</script>
```

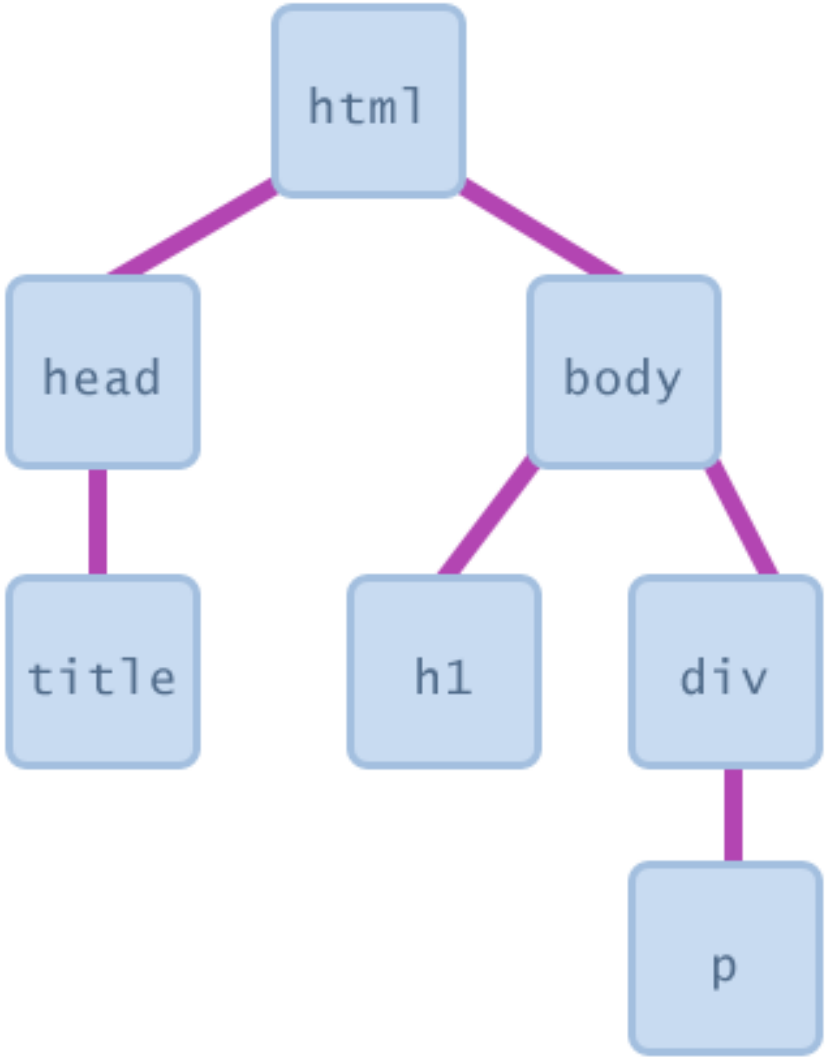
Form Input Elements

Tag	Purpose
<input type="text">	Make a text field (single row)
<input type="button" value="Click Me">	Makes a button that user can click with label "Click Me"
<button>Click Me</button>	Makes a button that user can click with label "Click Me"
<textarea></textarea>	Makes text area (multiple row)

DOM Interactions

we use form input tags for getting user inputs and update the webpage based on some events to change outputs.

For being able to do this we need to learn to use something called **DOM**.



Document Object Model (DOM)

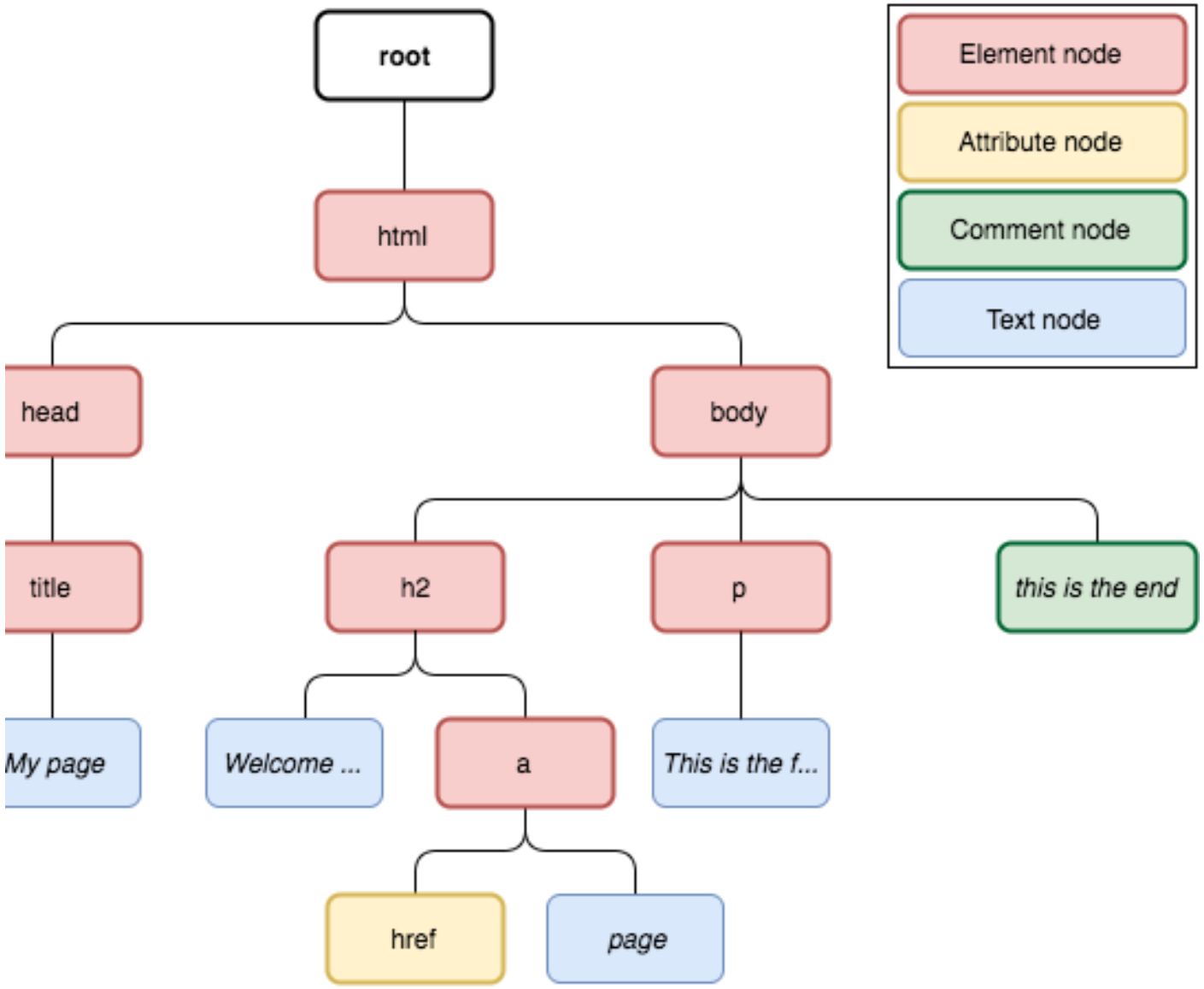
- All HTML elements are represented in browsers as objects
- All objects are nested together in one tree (DOM tree)
- Elements can have parents, siblings and children
- Most JS code manipulates elements (objects) on the DOM
  - it can change state (insert some new text into a span)
  - it can change styles (make a paragraph red)

Types of Nodes

A node is **any DOM** object.

An element is one specific type of node as there are many other types of nodes (text nodes, comment nodes, document nodes, etc...).

The DOM consists of a hierarchy of nodes where each node can have a parent, a list of child nodes and a nextSibling and previousSibling.



HTML element's id attribute

- The id attribute specifies a unique identifier for a HTML element (the value must be unique within the HTML document)
- The id attribute is used to target elements in JavaScript (via. the HTML DOM)

Getting a DOM element using its id

- `document.getElementById("id")`
  - Get the element with the specified id.

- Example, Program to get first name and last name from the input fields and display full name inside span element on button click.

Other DOM selection APIs

- `getElementsByName("name")`
  - Get all the elements with the specified name
- getElementsByTagName("tag")
  - Get all the elements in the document with the specified tag name
- querySelector("selector")
  - Get the first element in the document that matches the specified CSS selector(s) in the document.
- querySelectorAll("selector")
  - Returns all the elements in the document that matches a specified CSS selector(s)

Query Selectors

```
//class selector
document.querySelector(".example").style.backgroundColor = "red";

//Tags selector
document.querySelector("p,h1").style.backgroundColor = "red";

//Id selector
document.querySelector("#myId").style.backgroundColor = "red";

//All Input Texts
document.querySelectorAll('input[type="text"]')

//Get the first <p> element in the document where the parent is a <div>
element.
document.querySelector("div >p").style.backgroundColor = "red";

//Get the first <a> element in the document that has a "target" attribute:
document.querySelector("a[target]");
```

QuerySelector Example

```
<!DOCTYPE html>
<html>
<body>
<ul>
  <li>The</li>
  <li>test</li>
</ul>
<ul>
  <li>has</li>
  <li>passed</li>
</ul>
<input id="myText" type="text" value="" />
<script>
  let elements = document.querySelectorAll('ul > li:last-child');

  for (let elem of elements) {
    alert(elem.innerHTML); // "test", "passed"
  }
  let inputText = document.getElementById("myText");
  alert(inputText.value,inputText.type);

</script>
</body>
</html>
```

```
// 1. Create <div> element
let div = document.createElement('div');

// 2. Set its class to "alert"
div.className = "alert";

// 3. Fill it with the content
div.innerHTML = "<strong>Hi there!</strong>an important message.";
```

Create DOM Nodes

DOM modification is the key to creating “live” pages.

To create DOM nodes,

`document.createElement(tag)`

Creates a new *element node* with the given tag:

Add DOM Elements

- `node.append(...nodes or strings)` – append nodes or strings *at the end* of node,
- `node.prepend(...nodes or strings)` – insert nodes or strings *at the beginning* of node,
- `node.before(...nodes or strings)` –- insert nodes or strings *before* node,
- `node.after(...nodes or strings)` –- insert nodes or strings *after* node,

```
ol.before('before'); // insert string "before" before <ol>
ol.after('after'); // insert string "after" after <ol>

let liFirst = document.createElement('li');
liFirst.innerHTML = 'prepend';
ol.prepend(liFirst); // insert liFirst at the beginning of <ol>
```

Changing HTML Elements

Property	Description
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element
Method	Description
element.setAttribute(attribute, value)	Change the attribute value of an HTML element

Finding HTML Elements

Method	Description
document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name

Adding and Deleting Elements

Method	Description
document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(new, old)	Replace an HTML element
document.write(text)	Write into the HTML output stream

JavaScript in a separate file

- JS code can be placed directly in the HTML file's body or head
  - but this is not a good practice.
- script code should be stored in a separate .js file
  - script tag in HTML should be used to link the .js files

```
<script src="filename" type="text/javascript"></script>
```

- When more than one script file is included
  - interpreter treats them as a single file;
  - share global context.
  - order in which file files are loaded matters
    - interpreter executes code as soon as it hits the `<script>` tag.

Main point

- JavaScript programs run on events, which makes JavaScript an event driven programing language. We also perform our actions based on events that are either external or internal. *When we establish our self at the field of pure consciousness, we perform conscious actions rather than the compulsive ones.*

Readings

- JavaScript HTML DOM (w3schools.com)
- JavaScript DOM Methods (w3schools.com)
- JavaScript DOM Document (w3schools.com)
- JavaScript DOM Elements (w3schools.com)
- JavaScript DOM HTML (w3schools.com)