Web Advanced: Javascript APIs

“We will learn JavaScript properly. Then, we will learn useful design patterns. Then we will pick up useful tools for making cool things better.”

FALL 2018

SESSION #6

THE DOCUMENT OBJECT MODEL & EVENT HANDLING

jaink@newschool.edu

https://canvas.newschool.edu/courses/1407281

https://classroom.github.com/classrooms/42809645-parsons-web-advanced-javascript-fall-2018

RECAP

WEB STANDARDS

- ➔ HTML standard (current HTML5)
- ➔ W3C.org

- ➔ ECMAscript (current ES6)
- ➔ ecma-international.org

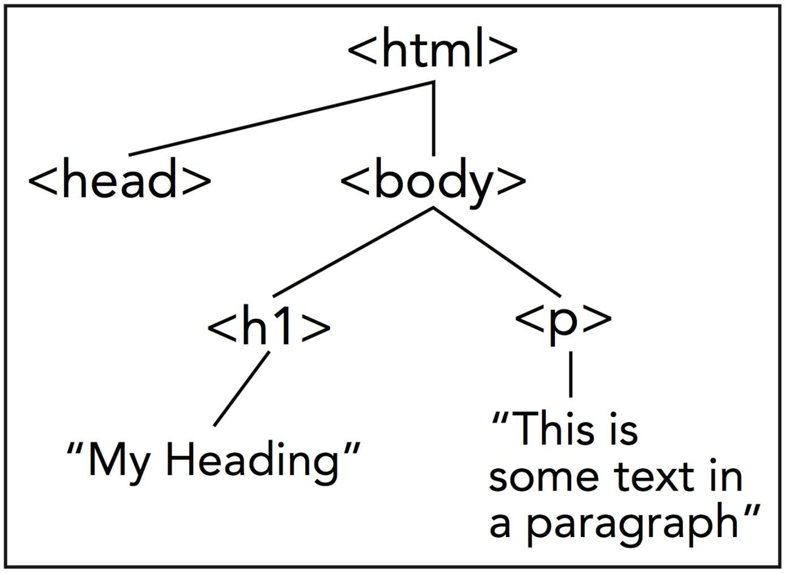
- ➔ CSS (current CSS 3)
- ➔ W3C.org

DOCUMENT OBJECT MODEL

- ➔ Abstract and standardized representation of the HTML document independent of the browser
- ➔ DOM Standards and Levels https://caniuse.com/#search=dom
- ➔ Some browsers do add in variations to the model eg. https://quirksmode.org/dom/html
- ➔ Basically a collection of Node objects and NodeLists

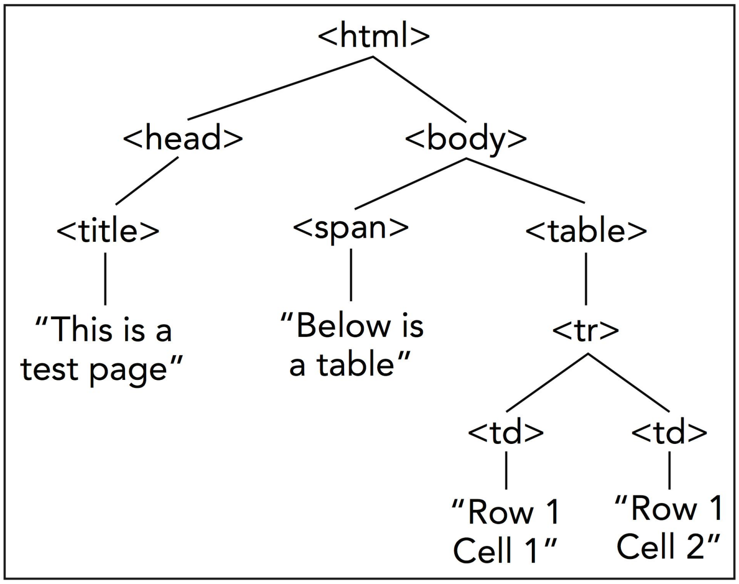
THE DOM TREE

```
<!DOCTYPE html>
<html lang="en">
<head></head>
<body>
   <h1>My Heading</h1>
   <p>This is some text in a paragraph.</p>
</body>
</html>
```



THE DOM TREE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>This is a test page</title>
</head>
<body>
<!-- this is a comment -->
  <h1 id="title">Main Title</h1>
  <span>Below is a <strong>table</strong></span>
  <table class="primary_table">
         <tr>
             <td>Row 1 Cell 1</td>
             <td>Row 1 Cell 2</td>
         </tr>
  </table>
  <span>Above is a table</span>
</body>
</html>
```



DOM NODES

DOM has different types of nodes. Some common ones:

- ➔ Element node: 1
- ➔ Text node: 3
- ➔ Comment node: 8
- ➔ The top level node, which is document: 9

DOM METHODS

the document object: document

```
let body = document.body      // returns node
object - level 1

console.log( typeof(body) );  // returns "object"

console.log( body.nodeType);  // returns integer

console.log( body.nodeName);  // returns name
"BODY"

console.log( body.nodeValue); // returns text
value for text, comment, and CDATA nodes

console.log( document.documentElement);  //
returns top level (usually HTML)

let images = document.images; // returns a list
of image nodes

let forms = document.forms;   // returns a list
of forms

let links = document.links;   // returns a list
of links inside anchors
```

DOM METHODS

getting elements

```
let title =
document.getElementById('title');
// returns node object

let table =
document.getElementsByClassName('primary_ta
ble');    // list of table objects by
classname

let spans =
document.getElementsByTagName('span');
   // list of span objects
```

Element list - similar to array (has length property) and can be iterated. eg.

```
for (let i = 0; i < spans.length; i++) {
  let item = spans[i];
  console.log( item.nodeType );
}
```

DOM METHODS

querySelector: returns first match on any element based on the css query

```
let spans =
document.querySelector('.primary_table');
// first span objects
```

querySelectorAll - returns the node list

```
let spans =
document.querySelectorAll('span');
// list of span objects
```

DOM NAVIGATION

childNodes returns ALL nodes that are children of a DOM object node:

```
let tables =
document.querySelector('.primary_table');
// list of table objects by class
```

```
tables.childNodes;
```

children returns all ELEMENT nodes that are children of a DOM object node:

```
tables.children; // returns only child
element Nodes
```

parentNode returns the parent node of a DOM object node:

```
tables.parentNode;
```

DOM NAVIGATION

textContent/innerHTML returns values inside a DOM object node:

```
let tables =
document.querySelector('.primary_table');
// list of objects inside table
```

```
tables.textContent;
```

```
tables.innerHTML;
```

nodeValue returns value inside a DOM object node containing text:

```
tables.nodeValue;   //returns null
```

```
tables.getElementsByTagName('td')[0].childN
odes[0].nodeValue; //returns content
```

```
document.querySelector('h1').childNodes[0].
nodeValue; //returns content
```

DOM ATTRIBUTES

All HTML elements have attributes such as class, id, src, and href:

```
tables.getAttribute("class"); // return
"primary_table"
tables.getAttribute("id");    // return null


let metas =
document.getElementsByTagName("meta");
for (let i=0; i<metas.length; i++) {
    if (metas[i].getAttribute("charset")) {
        console.log (
metas[i].getAttribute("charset") );
      }
}

console.log(document.querySelector("meta[charset]
").getAttribute("charset"));
```

DOM ATTRIBUTES

Class attributes:

```
tables.className;      // return
"primary_table"


tables.classList;   // returns a object
list of all classes

tables.classList.contains("primary_table");
```

SETTING DOM ATTRIBUTES

Using generic attributes:

```
tables.className;       // return
"primary_table"
```

```
tables.setAttribute("class", "sec_table");
```

```
tables.setAttribute("id", "FirstTable");
```

```
document.querySelector("meta[charset]").set
Attribute("charset","ASCII");
```

Using Class attributes:

```
tables.className = tables.className +
"sec_table";     // return "primary_table"
```

```
tables.classList.add("third_table"); //
returns a object list of all classes
```

```
tables.classList.remove("sec_table");
```

```
tables.classList.toggle("enabled");
```

UPDATING DOM

Create a new element:

```
let new_para = document.createElement('p');
let text = document.createTextNode('This is
the end.');
new_para.appendChild(text);
```

OR

```
let new_para = document.createElement('p');
new_para.textContent = "This is the end.";
```

UPDATING DOM

Add to Page:

```
document.body.appendChild(new_para); //adds to
end of body


let top_head =
document.getElementsByTagName('h1')[0];

top_head.appendChild(new_para); //adds to end of
h1 tag


let sect =
document.getElementsByTagName('section')[0];

let title = sect.getElementsByTagName('h1')[0];

sect.insertBefore(new_para,title); // //adds
before h1 tag
```

UPDATING DOM

Removing elements:

```
sect.removeChild(new_para);
// reference still exists so it can be
reinserted.
sect.insertBefore(new_para,title);
```

Replacing elements:

```
h1 = document.getElementById("title");
let oldText = h1.firstChild;
let newText = document.createTextNode("New
Title");
h1.replaceChild(newText,oldText)

// alternative non-standard but accepted
approach
h1.innerHTML = "New Title";
```

UPDATING DOM

Playing with Styles:

```
let h1 = document.getElementById("title");


h1.style.border = "2px solid red";


h1.style.backgroundColor = "lightgrey";
//props are in lower camel case


h1.style.display = "none";
```

EVENTS

- ➔ Connects user interactions with the DOM.
- ➔ Events occur whenever user clicks, types, moves the mouse.
- ➔ Custom events can also be defined in javascript.
- ➔ Event listeners informs javascript when the event happens.

EVENT LISTENERS

Event listener is a DOM method that listens out for any specified event on the page:

```
document.addEventListener(event_type,
callback_function);


let an =
document.getElementsByTagName('a')[0];

an.addEventListener("click", function() {
alert('xx')});
```

Event listeners get added to the node object or root (document):

```
addEventListener("click", function() {
alert("yy")}); //click anywhere on the
document
```

Events list: http://devdocs.io/dom_events/

EVENT OBJECT

An event object is sent to the callback function:

```
let an =
document.getElementsByTagName('a')[0];

let clickFunction = function(e) {

   console.log(e.type); // e contains all
properties of the event that occurs

}

an.addEventListener("click",
clickFunction);
```

MOUSE EVENTS

The mouseover event occurs when the mouse pointer is placed over the element to which the event listener is attached.
The mouseout event occurs when the mouse pointer moves away from an element.
The mouseover event occurs when the mouse pointer is placed over the element
The mouseout event occurs when the mouse pointer moves away from an element.

```
let an = document.getElementsByTagName('a')[1];

an.addEventListener("click",function(e){
console.log("click") });

an.addEventListener("mousedown",function(e){
console.log("down") });

an.addEventListener("mouseup",function(e){
console.log("up") });

an.addEventListener("mouseover",function(e){
console.log("over") });

an.addEventListener("mouseout",function(e){
console.log("out") });
```

BLOCK DEFAULT BEHAVIOUR

EG. Prevent redirecting to a link on clicking an anchor:

```
let an =
document.getElementsByTagName('a')[1];
an.addEventListener("click",function(e){
    console.log("click");
    e.preventDefault();
});
```

EVENT PROPAGATION

An event is inherited by all child nodes of the node the listener is added to.

Bubbling - when event goes up the tree, from more specific event handler to less specific - default

```
let an =
document.getElementsByTagName('a')[1];

an.addEventListener("click",function(e){

   console.log("click");

   e.preventDefault();

});
```

To stop propagation:

```
e.stopPropagation();
```

EVENT DELEGATION

An event is inherited by all child nodes of the node the listener is added to.

```
let list =
document.getElementsByTagName('ul')[0];
list.addEventListener("click",function(e){
    console.log("click");
    e.preventDefault();
});
```

EXAMPLES

EXAMPLES

Next Steps

➔ Forms and AJAX