AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

Outline

DOM Manipulation

- Introduction to HTML DOM
- DOM Methods
- DOM Document
- DOM Elements
- DOM HTML & CSS
- DOM Events
- DOM Event Handlers

What is HTM DOM?

The HTML DOM is a standard object model and programming interface for HTML. It defines:

- The HTML elements as objects
- The properties of all HTML elements
- The methods to access all HTML elements
- The events for all HTML elements

In other words: The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

Animations

JavaScript animations are done by programming gradual changes in an element's style.

The changes are called by a timer. When the timer interval is small, the animation looks continuous.

Basic Animation Code

AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

```
var id = setInterval(frame, 5);

function frame( ) {
  if (/* test for finished */) {
    clearInterval(id);
  } else {
    /* code to change the element style */
  }
}
```


SUPER
MARIO
™

Super Mario Animation


```
<!DOCTYPE HTML>
<html>
 <head>
  <style>
    #mario {
     position: relative;
     cursor: pointer;
     width: 150px;
    }
  </style>
 </head>
<body>
 <img id="mario"
src="https://live.staticflickr.com/4077/4887414974_e742d80cb1_b.jpg">
</body>
</html>
```

```
mario.onclick = function() {
 let start = Date.now();

 let timer = setInterval(superMario, 20);

 function superMario() {
  let timePassed = Date.now() - start;

  if (timePassed > 2000) {
   clearInterval(timer);
  } else {
   mario.style.left = timePassed / 5 + 'px';
  }
 }
}
```

DOM Events

An event is a signal that something has happened.

All DOM nodes generate such signals (but events are not limited to DOM).

List of the most useful DOM events

Mouse events:

- click – when the mouse clicks on an element (touchscreen devices generate it on a tap).
- contextmenu – when the mouse right-clicks on an element.
- mouseover / mouseout – when the mouse cursor comes over / leaves an element.
- mousedown / mouseup – when the mouse button is pressed / released over an element.
- mousemove – when the mouse is moved.

AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

Form element events:

- submit – when the visitor submits a <form/>.
- focus – when the visitor focuses on an element, e.g. on an <input/>.
- change – when the value of an element changes, e.g. on an <input/>

Keyboard events:

- keydown and keyup – when the visitor presses and then releases the button.

Document events:

- DOMContentLoaded – the browser fully loaded HTML, and the DOM tree is built, but external resources like pictures <img/> and stylesheets may be not yet loaded.
- load – not only HTML is loaded, but also all the external resources: images, styles etc.
- beforeunload/unload – the user is leaving the page.

CSS events:

- transitionend – when a CSS-animation finishes.

AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

DOM Event Handlers

To react on events we can assign a handler – a function that runs in case of an event.

Handlers are a way to run JavaScript code in case of user actions.

Ways to assign a handler to a DOM element:

- HTML-attribute
- DOM property
- addEventListener

HTML-attribute

A handler can be set in HTML with an attribute named on<event>.

For instance, to assign a click handler for an input, we can use onclick, like here:

<input value="Click me" onclick="alert('Click!')" type="button">

On mouse click, the code inside onclick runs.

Write code to display your name from an input field when a button is clicked.

DOM property

We can assign a handler using a DOM property on<event>.

For instance, elem.onclick:

```
<input id="elem" type="button" value="Click me">
<script>
  elem.onclick = function() {
    alert('Thank you');
  };
</script>
```

If the handler is assigned using an HTML-attribute then the browser reads it, creates a new function from the attribute content and writes it to the DOM property.

The handler is always in the DOM property: the HTML-attribute is just one of the ways to initialize it.

Note: With this approach, **we can’t assign more than one event handler.**

Therefore, these two code pieces work the same:

Example 1

```
<input type="button" onclick="alert('Click!')" value="Button">
```

Example 2

```
<input type="button" id="button" value="Button">
<script>
  button.onclick = function() {
    alert('Click!');
  };
</script>
```

addEventListener

The addEventListener() method attaches an event handler to the specified element.

It also attaches an event handler to an element without overwriting existing event handlers.

This implies that:

- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.

Syntax

element.addEventListener(event, handler[, options]);

- The first parameter is the type of the event (like "click" or "mousedown").
- The second parameter is the function we want to call when the event occurs.
- The third parameter is options. An additional optional object with properties:
  - once: if true, then the listener is automatically removed after it triggers.
  - capture: the phase where to handle the event, specifying whether to use event bubbling or event capturing. For historical reasons, options can also be false/true, that's the same as {capture: false/true}.
  - passive: if true, then the handler will not preventDefault()

Note that you don't use the "on" prefix for the event; use "click" instead of "onclick".

For example,

```
element.addEventListener("click", function(){ alert("Hello World!"); });
```

To add Many Event Handlers to the Same Element:

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
```

You can add events of different types to the same element:

```
element.addEventListener("mouseover", myFunction);
element.addEventListener("click", mySecondFunction);
element.addEventListener("mouseout", myThirdFunction);
```

To add an Event Handler to the window Object

```
window.addEventListener("resize", function(){
  document.getElementById("demo").innerHTML = sometext;
});
```

The above adds an event listener that fires when a user resizes the window.

removeEventListener

The removeEventListener() method removes event handlers that have been attached with the addEventListener() method:

For example:

```
element.removeEventListener("mousemove", myFunction);
```

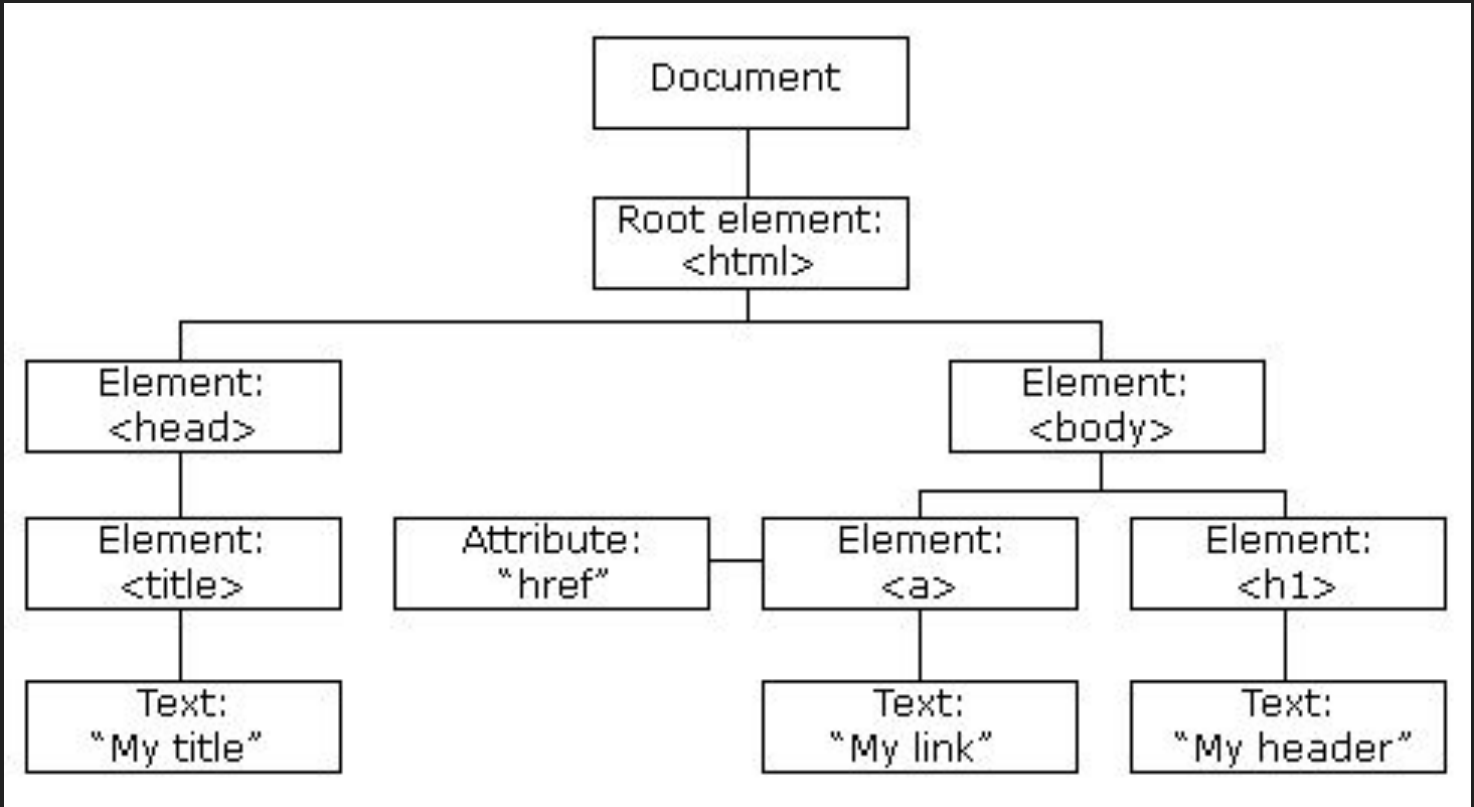
AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

DOM Nodes

According to the W3C HTML DOM standard, everything in an HTML document is a node:

- The entire document is a document node
- Every HTML element is an element node
- The text inside HTML elements are text nodes
- Every HTML attribute is an attribute node (deprecated)
- All comments are comment nodes

All nodes in the HTML DOM node tree can be accessed by JavaScript. Thus, nodes can be created, modified or deleted.



Source: https://www.w3schools.com/js/pic_htmltree.gif

Creating New HTML Elements (Nodes)

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

```
<div id="div1">
  <p id="p1">This is a paragraph.</p>
  <p id="p2">This is another paragraph.</p>
</div>
```

```
var para = document.createElement("p");

var node = document.createTextNode("This is new.");

para.appendChild(node);

var element = document.getElementById("div1");

element.appendChild(para);
```

The appendChild() method in the previous example, appended the new element as the last child of the parent.

If you don't want that you can use the insertBefore() method:

```
var element = document.getElementById("div1");

var child = document.getElementById("p1");

element.insertBefore(para, child);
```

Removing Existing HTML Elements

```
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.removeChild(child);
```

Replacing HTML Elements

To replace an element to the HTML DOM, use the replaceChild() method:

```
var parent = document.getElementById("div1");
var child = document.getElementById("p1");
parent.replaceChild(para, child);
```

AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

DOM Collections & Node Lists

The HTMLCollection Object

An HTMLCollection object is an array-like list (collection) of HTML elements.

The getElementsByTagName() method returns an HTMLCollection object. For example:

```
var x = document.getElementsByTagName("p");
```

The elements in the collection can be accessed by an index number.

To access the second <p> element you can write:

```
y = x[1];
```

The HTML DOM NodeList Object

A NodeList object is a list (collection) of nodes extracted from a document.

A NodeList object is almost the same as an HTMLCollection object.

Some (older) browsers return a NodeList object instead of an HTMLCollection for methods like getElementsByClassName().

All browsers return a NodeList object for the property childNodes.

Most browsers return a NodeList object for the method querySelectorAll().

For example:

```
var myNodeList = document.querySelectorAll("p");
```

The elements in the NodeList can be accessed by an index number.

To access the second <p> node you can write:

```
y = myNodeList[1];
```

Note: *A node list is not an array! A node list may look like an array, but it is not. You can loop through the node list and refer to its nodes like an array. However, you cannot use Array Methods, like valueOf(), push(), pop(), or join() on a node list.*

Assignment

AkiraChix
SHE BUILDS • SHE SERVES • SHE LEADS

```
<!DOCTYPE HTML>
<html>
<head>
 <title>Greens Kiosk</title>
</head>
<body>
 <h1 id="title">Welcome to Greens Kiosk</h1>
 <p>We sell fruits and vegetables</p>
 <h3>Fruits</h3>
 <ul id="fruList">
  <li>Mangoes</li>
  <li>Bananas</li>
  <li>Water Melons</li>
 </ul>
 <h3>Vegetables</h3>
 <ul id="vegList">
  <li>Onions</li>
  <li>Tomatoes</li>
  <li>Kales</li>
 </ul>
</body>
</html>
```

Questions:

1. Animate the title to toggle between green and silver font color after each second
2. Animate fruits list to be collapsible
3. Animate vegetables list to be collapsible
4. Add an input field to append a fruit to the fruits list on clicking a button submit fruit
5. Add an input field to append a vegetable to the vegetables list on pressing enter button on the keyboard. Hint: https://www.w3schools.com/howto/howto_js_trigger_button_enter.asp

Next class

- Introduction to:
  - REST
  - JSON
  - AJAX
- Introduction to JS Frameworks
  - React
  - Vue
- Introduction to JS Tools & Libraries
  - NPM
  - NodeJS

Video the week

https://www.youtube.com/watch?v=hUCT4b4wa-8