CS472 WAP

DOM API

JavaScript for Modern Web Apps

Maharishi International University Fairfield, Iowa © 2020

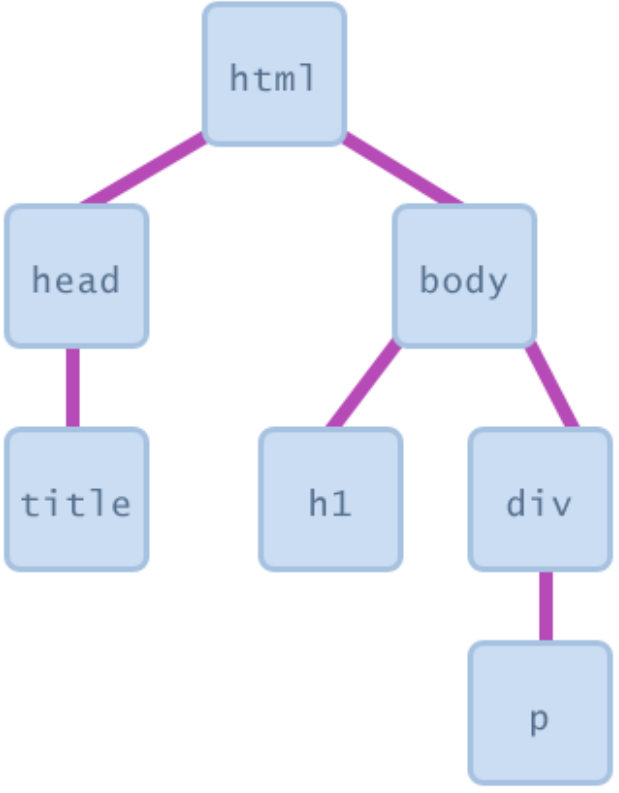
Main Point Preview

- The DOM is an API so the JavaScript programmer can conveniently access and manipulate the HTML elements in code.

- *Science of Consciousness: The TM and TM-Sidhi programs are techniques that allow anyone to conveniently contact and act at the level of the Unified Field.*

Document Object Model (DOM)

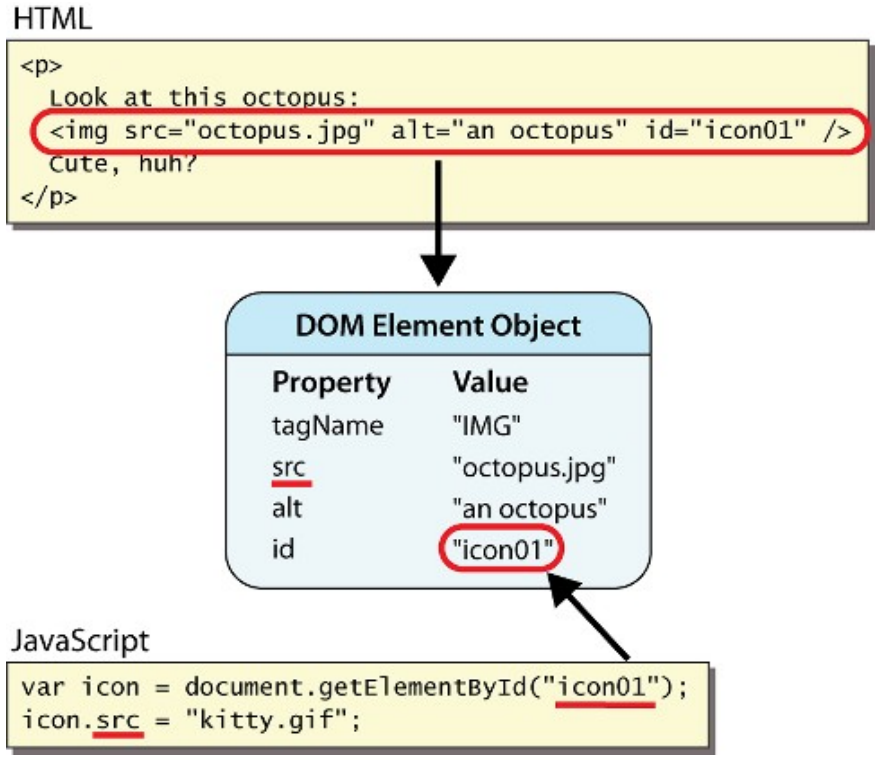
- most JS code manipulates elements on an HTML page
- we can examine elements' state
  - e.g. see whether a box is checked
- we can change state
  - e.g. insert some new text into a div
- we can change styles
  - e.g. make a paragraph red



DOM element objects

- every element on the page has a corresponding DOM object

- access/modify the attributes of the DOM object with *objectName.attributeName*



Accessing elements: document.getElementById

```
const name = document.getElementById("someId");

<button onclick="changeText();">Click me!</button>
<input id="output" type="text" value="replace me" />

function changeText() {
 const textbox = document.getElementById("output");
 textbox.value = "Hello, world!";
}
```

- document.getElementById returns the DOM object for an element with a given id
- can change the text in most *form controls* by setting the value property
- Browser automatically updates the screen when any DOM object is changed

More advanced example

```
<button onclick="swapText();">Click me!</button>
<span id="output2">Hello</span>
<input id="textbox2" type="text" value="Goodbye" />

function swapText() {
 var span = document.getElementById("output2");
 var textBox = document.getElementById("textbox2");
 var temp = span.innerHTML;
 span.innerHTML = textBox.value;
 textBox.value = temp;
}
```

can change the text inside most elements by setting the innerHTML property
See examples: See example: lecture05_examples/click2.html
lecture05_examples/click1.html

DOM object properties

```
<div id="main" class="foo bar">
 <p>Hello, <em>very</em> happy to see you!</p>
 <img id="icon" src="greatwall.jpg" alt="GREAT WALL" />
</div>
const mainDiv = document.getElementById("main");
const icon = document.getElementById("icon");
```

See example: lecture06_examples/objectproperties.html

Property	Description	Example
tagName	element's HTML tag	mainDiv.tagName is "DIV"
className	CSS classes of element	mainDiv.className is "foo bar"
innerHTML	Content in element	mainDiv.innerHTML is "\n <p>Hello…
src	URL target of an image	icon.src is "greatwall.jpg"

DOM properties for form controls

```
<input id="sid" type="text" size="7" maxlength="7" />
<input id="frosh" type="checkbox" checked="checked" />
  Freshman?
const sid = document.getElementById("sid");
const frosh = document.getElementById("frosh");
```

Property	Description	Example
value	the text/value chosen by the user	sid.value could be "1234567"
checked	whether a box is checked	frosh.checked is true
disabled	whether a control is disabled (boolean)	frosh.disabled is false
readOnly	whether a text box is read-only	sid.readOnly is false

Abuse of innerHTML

```
// bad style!
var paragraph = document.getElementById("welcome");
paragraph.innerHTML = "<p>text and <a
 href="page.html">link</a>";
```

- innerHTML is the content between the HTML tags of an element
- innerHTML can inject arbitrary HTML content into the page
- however, this is prone to bugs and errors and is considered poor style
  - innerHTML not part of DOM standard
  - issues involving DOM rebuilding tree and maintaining node and event handler references
- Best practice: inject plain text only
  - Do not use innerHTML to inject HTML tags;

DOM elements style property

```
<button id="clickme">Color Me</button>
window.onload = function(){
  document.getElementById("clickme").onclick = changeColor;
};
function changeColor() {
 const clickMe = document.getElementById("clickme");
 clickMe.style.color = "red";
}
```

Property	Description
style	lets you set any CSS style property for an element

- contains same properties as in CSS, but with camelCasedNames
  - examples: backgroundColor, borderLeftWidth, fontFamily

Common DOM styling errors

- many students forget to write .style when setting styles

```
var clickMe = document.getElementById("clickme");
clickMe.color = "red";
clickMe.style.color = "red";
```

- style properties are capitalized likeThis, not like-this

```
clickMe.style.font-size = "14pt";
clickMe.style.fontSize = "14pt";
```

- style properties *must* be set as strings, often with units at the end

```
clickMe.style.width = 200;
clickMe.style.width = "200px";
clickMe.style.padding = "0.5em";
```

  - write exactly the value you would have written in the CSS, but in quotes

Unobtrusive styling

```
function okayClick() {
  this.style.color = "red";
  this.className = "highlighted";
}
.highlighted { color: red; }
```

- well-written JavaScript code should contain as little CSS as possible
- use JS to set CSS classes/IDs on elements
- define the styles of those classes/IDs in your CSS file
- Remember cssZenGarden?

Getting/Setting CSS classes

```
function highlightField() {
 // turn text yellow and make it bigger
 if (!document.getElementById("text").className){
   document.getElementById("text").className = "highlight";
 } else if (document.getElementById("text").className.indexOf("invalid") < 0) {
   document.getElementById("text").className += " highlight";
 }
}
```

- JS DOM's className property corresponds to HTML class attribute
- somewhat clunky with multiple space-separated classes
  - which is what getElementById(..).className returns

Common bug: incorrect usage of existing styles

```
document.getElementById("main").style.top =
document.getElementById("main").style.top + 100 + "px"; // bad
```

- the above example computes e.g. "200px" + 100 + "px" , which would evaluate to "200px100px"

- a corrected version:

```
document.getElementById("main").style.top =
  parseInt(document.getElementById("main").style.top) +
  100 + "px"; // correct
```

Main Point

- The DOM is an API so the JavaScript programmer can conveniently access and manipulate the HTML elements in code.

- *Science of Consciousness: The TM and TM-Sidhi programs are techniques that allow anyone to conveniently contact and act at the level of the Unified Field.*

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

JavaScript for Modern Web Apps: Spontaneous Right Action

1. JavaScript code comes with an HTML page and is executed on the browser.
2. JavaScript reacts to browser events and manipulates the web page using the HTML DOM API.

---

3. **Transcendental consciousness** is the source of thought and the home of all the laws of nature.
4. **Impulses within the transcendental field:** Actions arising from this level will spontaneously be in accord with all the laws of nature.
5. **Wholeness moving within itself:** In unity consciousness, all of one's perceptions and actions are grounded in the experience of pure consciousness.


MAHARISHI
HIGHER EDUCATION
FOR HIGHER CONSCIOUSNESS
INTERNATIONAL UNIVERSITY