

## Web Applications

### CSE183

Fall 2020

## Document Object Model



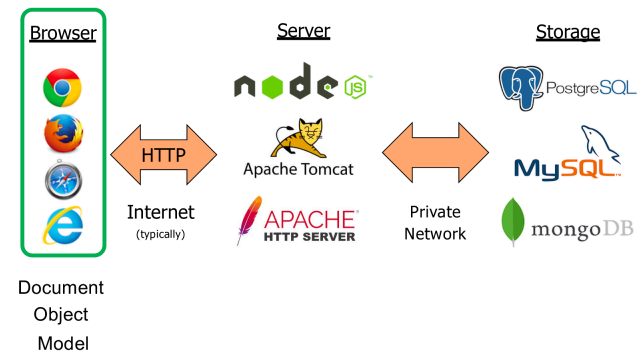
### Today's Lecture

- Document Object Model (DOM)
  - Hierarchy
  - Node Properties, Methods, Mutators
  - DOM and CSS
  - Coordinates & Dimensions
- DOM Events
  - Handling
  - Event Objects
  - Precedence
  - Timer Events
- Questions

### Notices

- **Assignment 3** due 23:59 **Thursday October 22**
- **Assignment 1** grades posted

### Full Stack Web Applications



## Document Object Model (DOM)

- HTML **document** exposed as a collection of JavaScript **objects**
- JavaScript can query and modify the HTML document via DOM
- Accessed via the JavaScript global scope:  
   window when using 'use strict'  
   this otherwise

UCSD BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

6

## DOM Hierarchy

- Starts at `window.document`
  - Can be shortened to simply `document`
- Follows HTML structure:  
   `document.head`  
   `document.body`
- DOM Objects have hundreds of properties
  - Though most are private
- DOM Objects have a common set of properties and methods known as the DOM “Node”

UCSD BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

7

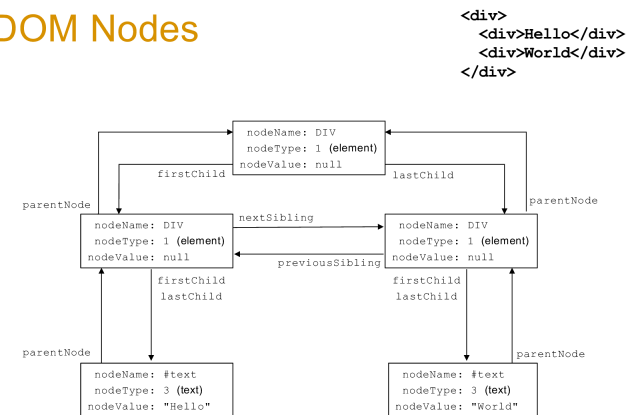
## The DOM Node

- Properties  
   `nodeName`
  - Element tag (P, DIV, etc.) always in UPPER CASE`parentNode`, `nextSibling`, `previousSibling`,  
   `firstChild`, `lastChild`
  - Codification of the document hierarchy
- Methods  
   `getAttribute`, `setAttribute`, ...
  - Accessors and mutators

UCSD BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

8

## DOM Nodes



UCSD BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

[https://www.w3schools.com/jsref/dom\\_node\\_properties.asp](https://www.w3schools.com/jsref/dom_node_properties.asp)

9

## Finding DOM Nodes

- Walk the hierarchy ( not recommended )

```
let element = document.body.firstChild.nextSibling;
```

- Use a lookup method

- Several of these [Self Study](#)

- E.g. by element id

```
<div id="mydiv">...</div>
```

```
let myDiv = document.getElementById('mydiv');
```

- E.g. by tag name

```
let allDivs = document.getElementsByTagName('div');
```

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

10

## Common Properties & Methods

**textContent** All the text in node and its descendants

```
document.getElementById('container').textContent
is 'Hello World'
```

**innerHTML** HTML of node's descendants

```
document.getElementById('container').innerHTML
is '<div>Hello</div><div>World</div>'
```

**outerHTML** HTML of node and its descendants

```
document.getElementById('container').outerHTML
is '<div id="container"><div>Hello</div><div>World</div>'
```

**getAttribute()** / **setAttribute()**

get and set attributes of an element

```
<div id='container'>
  <div>Hello</div>
  <div>World</div>
</div>
```

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

11

## Common Mutators

- Change the content of an element

```
element.innerHTML = '<p>A Paragraph</p><div>A DIV</div>'
```

- Replaces content but retains attributes

- Change the **src** attribute of an image tag

```
img.src = 'foo.png'
```

- Set visibility of elements

```
element.style.display = "none";           // not visible and uses no space
```

```
element.style.visibility = "hidden";       // invisible, but uses space
```

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

12

## DOM and CSS

- Change an element's class ( i.e. change its style completely )

```
element.className = 'active';
```

- Update an element's style ( not recommended approach )

```
element.style.color = '#ff000';
```

- Query DOM using a CSS selector

```
document.querySelector()
document.querySelectorAll()
```

Details? [Self Study](#)

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

13

## Modifying the DOM Hierarchy

- Create a new element  
`element = document.createElement('DIV');`
- Clone an existing element  
`element = document.getElementById('myDiv').cloneNode();`
- Add to an existing Node  
`parent.appendChild(element);`  
`parent.insertBefore(element, sibling);`
- Remove Nodes  
`parent.removeChild(element);`
- Yet most developers find setting `innerHTML` easier
  - Yet this is, arguably, poor practice

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

14

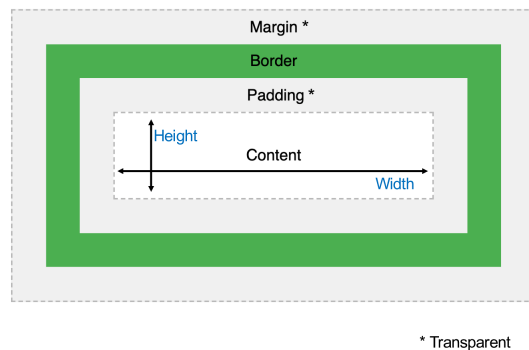
## Miscellaneous DOM Operations

- Redirect to a new page  
`window.location.href = "newPage.html";`
  - Warning: interrupts JavaScript execution
- Communicate with the user  
`alert('Hello World!');`  
`confirm('Are you quite sure?');`
- Developer messages  
`console.log();`  
`console.debug();`  
`console.error();`
  - And many more [Self Study](#)

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

15

## The CSS Box Model



UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

[https://www.w3schools.com/css/css\\_boxmodel.asp](https://www.w3schools.com/css/css_boxmodel.asp)

16

## DOM Coordinates

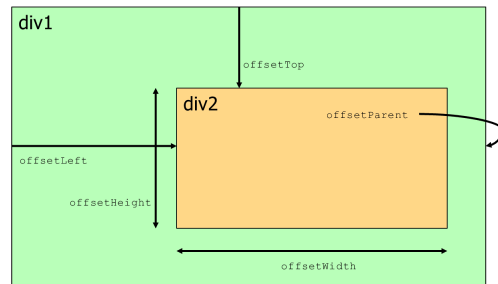
- Top left corner is the origin (0,0)
- Element position defined by the top left corner of its margin
- Read with:  
`element.offsetLeft`  
`element.offsetTop`
- Coordinates are relative to:  
`element.offsetParent`
  - Not necessarily the same as `element.parentNode`

UCSB BRIDE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

17

## DOM Coordinates

```
<div class='div1'><div class='div2'></div></div>
```



UCSC BSOE CBE183 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

18

## Element Dimensions

- **Reading:**  
`element.offsetWidth` and `element.offsetHeight`
  - Includes contents, padding, border but not margin
- **Updating:**  
`element.style.width` and `element.style.height`

UCSB BSOE CBE183 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

19

## DOM Events

- DOM “talks to” JavaScript via Events
  - JavaScript manipulates DOM via methods and mutators
- Principal Event Types:
  - Mouse ( movement, button clicks, enter / leave an element )
  - Keyboard ( key up, down, pressed )
  - Focus
  - Input Field Traversal
  - Timer
- Other Event Types:
  - Element content changed
  - Page loaded or unloaded
  - Image loaded
  - Uncaught exceptions
  - And many more... **Self Study**

UCSC BSCOE CSE183 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

20

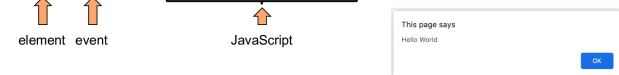
## Event Handling

- Create an **Event Handler**
- Specify:
  - The event of interest ( what happened )
  - The element of interest ( which element it happened to )
  - JavaScript to execute when event happens to the element

- Options:

- In HTML:

```
<div onclick = "alert('Hello World');">Hello World</div>
```



- In JavaScript via DOM

```
let mouseClick = function() { alert('Hello World'); }
element.onclick = mouseClick;
Or element.addEventListener('click', mouseClick);
```

UCSC B50E C8E1B3 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

21

## The Event Object

- Event listener functions passed an `Event` object
- Typically sub-classed `MouseEvent`, `KeyboardEvent`, etc.
- Selected Event properties:
  - `type` name of the event ('click', 'mousedown', 'keyup', ...)
  - `timeStamp` when the event was created
  - `currentTarget` element listener was registered on
  - `target` element that dispatched the event

UCSD BRIDE CSE119 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

22

## Mouse and Keyboard Events

- Selected `MouseEvent` properties ( prototype inherits from `Event` )
  - `button` mouse button that was pressed
  - `pageX`, `pageY` mouse position relative to the page origin
  - `screenX`, `screenY` mouse position in screen coordinates
- Selected `KeyboardEvent` properties (prototype inherits from `Event`)
  - `keyCode` identifier for the keyboard key that was pressed
    - Not necessarily an ASCII character
  - `charCode` integer Unicode value corresponding to keypress
    - But only if there is one

UCSD BRIDE CSE119 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

23

## Draggable Circle

### HTML

```
<body>
  <div class="draggable">
    onmousedown="mouseDown(event);"
    onmousemove="mouseMove(event);"
    onmouseup="stopDragging(event);"
    onmouseleave="stopDragging(event);">
      Drag Me!
    </div>
</body>
```

### CSS

```
.draggable{
  position: absolute;
  height: 200px;
  width: 200px;
  background-color: yellow;
  display: flex;
  justify-content: center;
  align-items: center;
  border: 3px solid green;
  border-radius: 100px;
  font-family: Arial, Helvetica, sans-serif;
  font-size: 30px;
}
```

UCSD BRIDE CSE119 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

### JavaScript

```
let dragging = false;
let prevX, prevY;

function mouseDown(event) {
  prevX = event.pageX;
  prevY = event.pageY;
  dragging = true;
  event.target.style.backgroundColor = 'green';
}

function stopDragging(event) {
  dragging = false;
  event.target.style.backgroundColor = '';
}

function mouseMove(event) {
  if (!dragging) {
    return;
  }
  let elem = event.target;
  elem.style.left = (elem.offsetLeft +
    (event.pageX - prevX)) + "px";
  elem.style.top = (elem.offsetTop +
    (event.pageY - prevY)) + "px";
  prevX = event.pageX;
  prevY = event.pageY;
}
```

24

## Event Precedence

- Complication ☹️
  - Elements can overlap with others
  - Suppose user clicks their mouse on "Hello World"
 


```
<body>
  <table>
    <tr>
      <td>Hello World</td>
    </tr>
  </table>
</body>
```
  - If we have event handlers on the `table`, `tr` and `td` elements, which handler gets called?
    - Self Study ( find out by writing the code, use "Draggable Circle" as a guide )
  - Sometimes more convenient for outer elements to handle events
  - Sometimes more convenient for inner elements to handle event

UCSD BRIDE CSE119 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

25

## Event Capture and Bubbling

- **Capture phase** (a.k.a. "trickle-down")
  - Start at the outermost element and work down to the innermost
  - Each element can stop the capture, so children never see the event: `event.stopPropagation()`
- **Bubble phase** (a.k.a. "bubble-up")
  - Invoke handlers on the innermost element that dispatches the event
    - Usually the most appropriate thing to do
  - Repeat on parent, grandparent, etc.
  - Any element can stop the bubbling: `event.stopPropagation()`
  - Can insist on Bubble only:
 



`element.addEventListener(eventType, handler, false);`
  - Handlers in bubble phase more common than capture phase

UCSD BRIDGE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

26

## Timer Events

- Used for animations, page refreshes, forced logout etc.
- **Given**

```
function foo() { ... }
```
- Run `foo` once, 5 seconds from now
 

```
let token = setTimeout(foo, 5*1000);
```
- Run `foo` every 50 milliseconds
 

```
let token = setInterval(foo, 50);
```
- Cancel a timer
 

```
clearInterval(token);
```

UCSD BRIDGE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

27

## Event Concurrency

- Events are serialized and processed one-by-one
- Event handling does not interleave with other JavaScript
  - Handlers run to completion
  - Not preemptable
  - Not multi-threaded
- Make reasoning about concurrency easier
  - Rarely need locks
- Background processing is much harder than with threads

UCSD BRIDGE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

28

## Event Based Programming

- Must wait for something to invoke your code
- Must return quickly from the handler
  - Otherwise Web App becomes unresponsive
- Key is to maintain control through events
  - Make sure you have declared enough handlers
  - Timers should be used only as a last resort
- Node.js provides an event dispatching mechanism for server-side JavaScript programming
  - We'll see this later in the class

UCSD BRIDGE CSE193 Fall 2020. Copyright © 2020 David C. Harrison. All Rights Reserved.

29

## Upcoming Lectures

- Friday: User Interfaces
  - Plus Assignment 2 Feedback & Assignment 4 Overview
- Monday: Introduction to React
- Wednesday: Single Page Web Applications
- Friday: Quiz 2

## Tasks

- **Assignment 3** due 23:59 **Thursday October 22**