IS216
Web
Application
Development II

Session 5

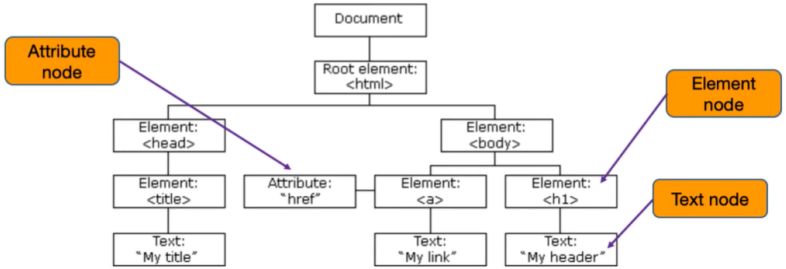
JavaScript – Part 2

(DOM & Event Handling)

K. J. Shim

Sections: G1 / G2 / G3 / G11

Agenda



Introduction to DOM

Event Handling – onclick event

Accessing HTML Elements

Accessing HTML Attributes

Modifying HTML

Adding and Removing HTML Elements

More event handling

Sessions 5/6/7 & Mini Lab Test

- **Sessions 5/6/7**
  - Continue with **ONLINE lesson delivery** (same lesson day/time) via **Zoom**
  - *My sincere apology to those students who really looked forward to meeting on campus*
  - **[Pair Programming]** Choose your own partner. Pick an empty Breakout Room and enter on your own.

- **Mini Lab Test**
  - Date/time: **Week 8**, **6-October-2021 (Wednesday)**, **2-3 PM SG Time**
  - Scope: *All topics* covered in **Sessions 1 through 6** – namely, **CSS** and **JavaScript**
  - Location
    - **[Local Students]** SMU campus (specific venues TBA)
    - **[Overseas Students]** IS216 course coordinator will reach out to you for online proctored test
  - **IMPORTANT**: You need to arrive at the test venue by **1:40 PM SG Time**
  - *In case of any changes, we will let you know ASAP – latest by **Week 7 Monday 9AM SG Time***

Source Code Files

eLearn → Content → Session 5 → In Class → **Week5.zip**

- **Unzip** it into your **webroot** (*any meaningful sub-directory), for example:*
  - (WAMP) C:\wamp64**www**\is216\…**Week5**
  - (MAMP) /Applications/MAMP/**htdocs**/is216/…/**Week5**

- You don't have to follow the above path – it's just an example.
  - But we DO strongly encourage you to keep source code files **organized** so that you can easily **search** them during **Lab Tests**

Follow Me (Code Together)

- **home.html**
  1. Accessing HTML ***elements*** (by **id**, **tag**, **class**), ***attributes***, ***styles***
  2. Event handling ***onclick***, ***onmouseover***, ***onmouseout***
  3. Modifying HTML ***elements***, ***attributes***, ***styles***
  4. (Pair Programming) **Activity 1**
  5. Adding new HTML ***elements***
  6. Removing HTML ***elements***
  7. Event handling using ***Event Listener***
  8. (Pair Programming) **Activity 2**
  9. (Extra Challenge) **Activity 3**
  10. (Extra Challenge) **Activity 4**

Activity 1: Girls & Boys (★★)

1. Go to **FollowMe** → **exercise1.html**
2. Complete the implementation of function **`show_next_photo()`** such that:

exercise1.html (Loaded in web browser for the **first time**)	exercise1.html (**AFTER** clicking "Show Next Photo" button)
Girl ← Default heading  Show Next Photo Click	Boy ← Heading changed  Show Next Photo

Default photo (a girl, "chloe.jpg")

Photo changed (randomly selected "boy" photo)

Back to Menu

Activity 1: Girls & Boys (★★) *continued…*

- Next, the user clicks “Show Next Photo” button again.

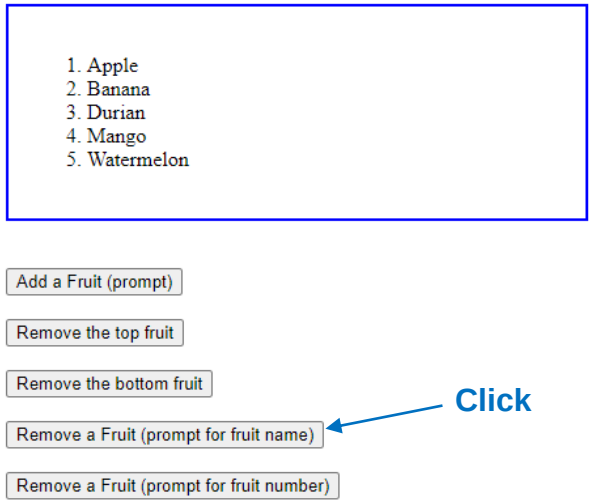
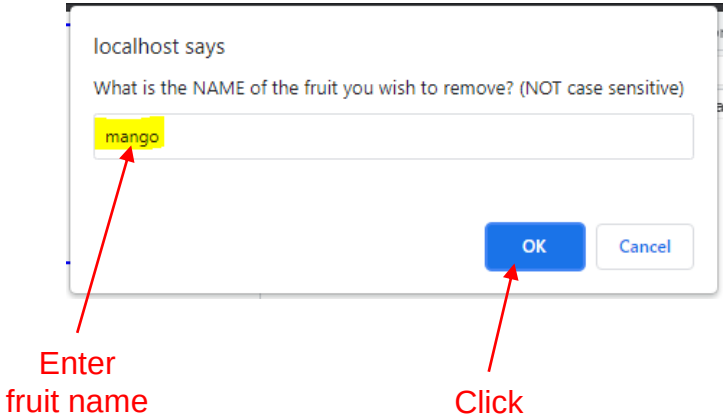
exercise1.html	exercise1.html (**AFTER** clicking “Show Next Photo” button)
Boy  Show Next Photo Click	**Girl** ← Heading changed  Show Next Photo Photo changed (randomly selected “girl” photo)

- In summary, **upon each “click” of the button**, the function must draw and display the next “opposite gender” celebrity photo and update the heading accordingly.

Back to Menu

Activity 2: Fruits (★★)

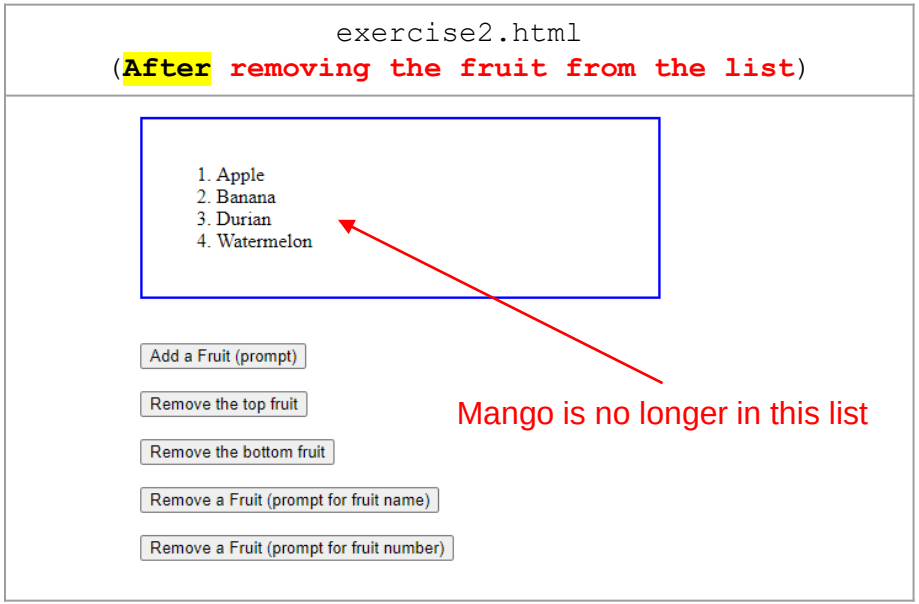
1. Go to **FollowMe** → **exercise2.html**
2. Complete the implementation of function **`remove_fruit_prompt_name()`** such that:

exercise2.html (Loaded in web browser for the **first time**)	exercise2.html (**Prompt** pop-up appears)
	

Back to Menu

Activity 2: Fruits (★★) *continued...*

- Notice how the fruit **mango** is no longer in the list. It's been ***removed***.



Your code must work for all fruits (not just "mango") !!!

Please sufficiently TEST your code!

- In summary, **upon each "click" of the button**, the function must prompt the user to specify a **fruit name** (NOT case-sensitive) and **remove** that fruit from the list accordingly.

Back to Menu

Activity 2: Fruits (★★) *continued…*

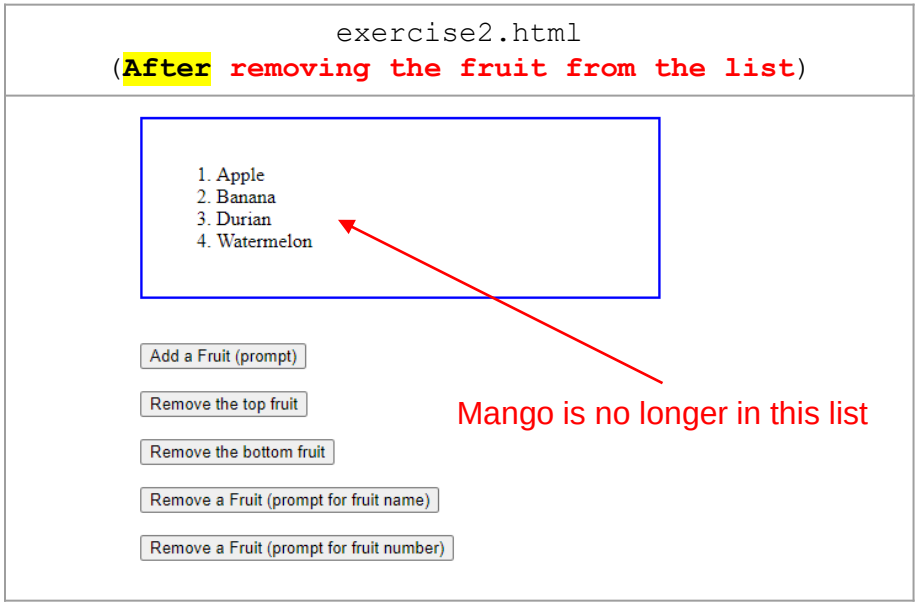
- Next, Complete the implementation of function **`remove_fruit_prompt_number()`** such that:

exercise2.html (Loaded in web browser for the **first time**)	exercise2.html (**Prompt** pop-up appears)
1. Apple 2. Banana 3. Durian 4. Mango 5. Watermelon Add a Fruit (prompt) Remove the top fruit Remove the bottom fruit Remove a Fruit (prompt for fruit name) Remove a Fruit (prompt for fruit number) ← **Click**	localhost says What is the fruit NUMBER you wish to remove? (Enter a valid number) 4 OK Cancel Enter fruit number (e.g. “4” corresponds to “Mango”) Click

Back to Menu

Activity 2: Fruits (★★) *continued...*

- Notice how the fruit **mango** (the *formely* **4th fruit**) is no longer in the list. It's been ***removed***.



Your code must work for all fruits (not just "4") !!!

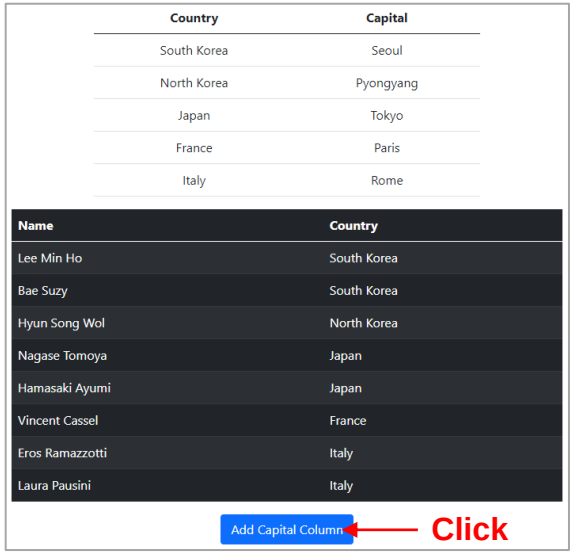
Please sufficiently TEST your code!

- In summary, **upon each "click" of the button**, the function must prompt the user to specify a **fruit number** and **remove** that fruit from the list accordingly.

Back to Menu

Activity 3: Countries & Capitals (⭐⭐½)

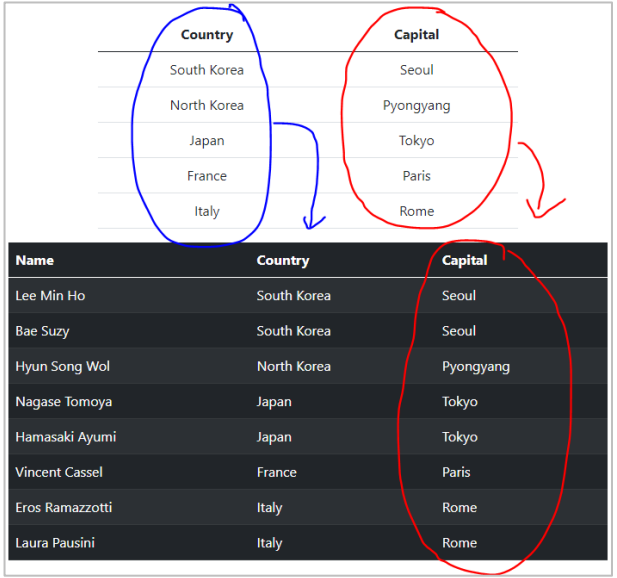
1. Go to **FollowMe** → **exercise3.html**
2. Complete the implementation of function `add_capital_column()` such that:
   - Upon clicking the **button**, the bottom (**black**) table gets an additional 3rd column containing the corresponding **capital**.
   - You should NOT hard-code the capital – you need to **dynamically** retrieve it from the top (white) table.
   - Also, the **button** must disappear.
   - DO NOT modify the HTML !!!



Country	Capital
South Korea	Seoul
North Korea	Pyongyang
Japan	Tokyo
France	Paris
Italy	Rome

Name	Country
Lee Min Ho	South Korea
Bae Suzy	South Korea
Hyun Song Wol	North Korea
Nagase Tomoya	Japan
Hamasaki Ayumi	Japan
Vincent Cassel	France
Eros Ramazzotti	Italy
Laura Pausini	Italy

Add Capital Column ← Click



Country	Capital
South Korea	Seoul
North Korea	Pyongyang
Japan	Tokyo
France	Paris
Italy	Rome

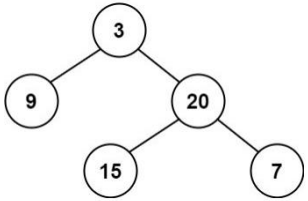
Name	Country	Capital
Lee Min Ho	South Korea	Seoul
Bae Suzy	South Korea	Seoul
Hyun Song Wol	North Korea	Pyongyang
Nagase Tomoya	Japan	Tokyo
Hamasaki Ayumi	Japan	Tokyo
Vincent Cassel	France	Paris
Eros Ramazzotti	Italy	Rome
Laura Pausini	Italy	Rome

Back to Menu

Activity 4: Binary Tree (★★½)

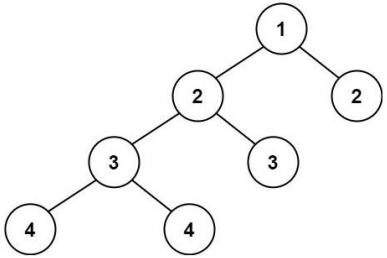
1. Go to **FollowMe** → **exercise4.html**
2. Complete **Part 1** and **Part 2** to determine whether a given **binary tree** is **height-balanced**.
   - Function `is_height_balanced()` must return **Boolean true** (tree is height-balanced) or **Boolean false** (tree isn't height-balanced)
   - For each **test case**, display either **"Height Balanced"** or **"NOT Height Balanced"** in the corresponding **<h1>** element using **JavaScript DOM**.

```
// TEST CASE 1
// DO NOT MODIFY THIS TREE
const tree1 = {
    3: {
        9: null,
        20: {
            15: null,
            7: null
        }
    }
}
```



This is a height-balanced binary tree

```
// TEST CASE 2
// DO NOT MODIFY THIS TREE
const tree2 = {
    1: {
        2: {
            3: {
                4: null,
                4: null
            },
            3: null
        },
        2: null
    }
}
```



This is NOT a height-balanced binary tree

Back to Menu

Things to Finish Before Session 6

1. Complete:
   - **JavaScript – Part 2 (DOM & Event Handling) – Challenges 10, 11, 12**
   - **Resource files** (***Bootstrap 5.1 version !!!***) can be downloaded from **this link**
2. "My Profile" GitHub assignment ***Don't forget to update README.md with your own summary***

> ## (Week 5) Add JavaScript to your website
>
> The goal this week is for you to apply what you learned about **JavaScript** to your website. Instead of using Bootstrap's pre-built components (which come with JavaScript code behind the scene), think of your own examples and write your own HTML and JavaScript code.
>
> - A good starting point might be... ***Google Searching*** ... "cool JavaScript examples", "fun JavaScript sample code" and try them.
> - You can also look for "JavaScript animation examples" and try them. Pick what you like and integrate into your ***My Profile*** website.

3. eLearn → Content → Session 6 → Before Class
   - Watch the video
   - Complete [Session 6] Pre-Class Quiz (JavaScript – Part 3)

What is DOM?

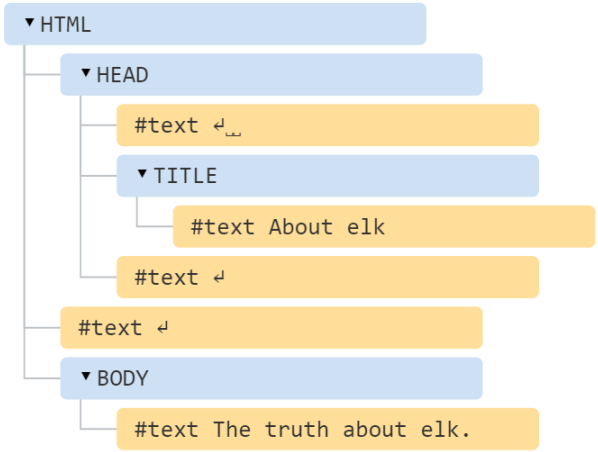
The browser represents an HTML document as Document Object Model (DOM).

In DOM, every HTML entities – elements, attributes, texts are represented as nodes (objects) in a hierarchical data structure.

This data structure makes it easy for JavaScript to access and change the entities in an HTML document.

With DOM, we can access and modify the HTML document through Javascript.

```
<!DOCTYPE HTML>
<html>
<head>
  <title>About elk</title>
</head>
<body>
  The truth about elk.
</body>
</html>
```



What is DOM?

Reference

[Document Object Model (DOM) - Web APIs | MDN]

Node is more general than Element

Important Concepts

Document: an interface represents any web page loaded in the browser.

Node: an abstract base class upon which many other DOM API objects are based.

Element: the most general base class from which all element objects (i.e. objects that represent elements).

Event: an interface represents an event which takes place in the DOM.

Event Handling

JavaScript can be used to listen to events and react (by changing the tree nodes) based on what the user does.

Events such as

- Clicking a mouse
- Hovering over an image
- Typing something in a search bar

Examples: *wk5example1.html*

```
<p id="time"></p>

<button
  onclick="document.getElementById('time').innerHTML= Date()">
      What time is it?
</button>
```

Events

Events reference: Event reference | MDN

Event	Description
onclick	The user clicks an HTML element
onchange	An HTML element has been changed
oninput	The user inputs something in an HTML element
onmouseover	The user moves the mouse over an HTML element
onmouseout	The user moves the mouse away from an HTML element
onkeypress	The user presses a key on the keyboard
onload	The browser has finished loading the page

DOM Manipulation

Manipulating the DOM, JavaScript can create dynamic HTML

- change all the HTML elements, attributes, and CSS styles in the page
- change all the HTML elements, attributes, and CSS styles in the page
- remove existing HTML elements and attributes
- add new HTML elements and attributes
- react to HTML events in the page
- create new HTML events in the page

Examples: *wk5example1.html*

```
<p id="time"></p>

<button onclick=
"document.getElementById('time').innerHTML=
Date()">What time is it? </button>
```

In this example, we change one HTML element in the page.

You can do whatever you want to the tree.

Access HTML Elements

Through ID

document.getElementById(“id”)

Returns the unique element which is associated with the provided ID.

Example

```
<p id="hello"></p>
<p id="hello"></p>
<script>
  document.getElementById("hello").innerHTML = "Hello World!";
</script>
```

What happens then?

Access HTML Elements

Through class name

```
document.getElementsByClassName("name")
```

Returns an “array” of all elements which are associated with the given class name(s).

Example: *wk5example1.html*

```
<p class="hello">To be updated</p>
<p class="hello">To be updated</p>
<button onclick= "sayHello()">Say Hello </button>
<script>
  function sayHello() {
     var x = document.getElementsByClassName("hello");
        for (ele of x) {
          ele.innerHTML = '<p>Hello~~~</p>';
        }
     }
</script>
```

Access HTML Elements

Through class name

```
element.getElementsByClassName("name")
```

You may also call getElementsByClassName() on any element; it will return only elements which are descendants of the specified root element with the given class name(s).

Examples

```
var x = document.getElementById("div1");
var y = x.getElementsByClassName("p");
```

The following finds the element with id="div1", and then finds all <p> elements inside "div1".

Access HTML Elements

Through tag name

```
getElementsByTagName("name")
```

returns a live HTMLCollection of elements with the given tag name.

*** “live” means that the collection changes if it so happens that elements are being updated

Examples

```
First Name: <input name="fname" type="text" >
Last Name: <input name="lname" type="text” >

<p id="name"></p>
<script>
  var n = document.getElementsByTagName("input")[0].value;
  document.getElementById("name").innerHTML = n;
</script>
```

Access HTML Elements

Through tag name

```
getElementsByTagName("name")
```

returns a live HTMLCollection of elements with the given tag name.

Examples: *wk5example2.html*

```
<a href="ex1.html">ex1</a>
<a href="ex2.html">ex2</a>
<a href="ex3.html">ex3</a>
<script>
  var nodes = document.getElementsByTagName("a");
  for (node of nodes) {
     console.log(node); // log the same <a> list above
  }
</script>
```

Access Properties

Properties

Use .value, .type, .id, .name and so on to access various properties of an input element

Example

wk5example3.html

Access Properties

Properties

Use document.forms to access various forms.

Example

wk5example4.html

Access Attributes

Attributes

Use node.getAttribute(“something”) to access attributes.

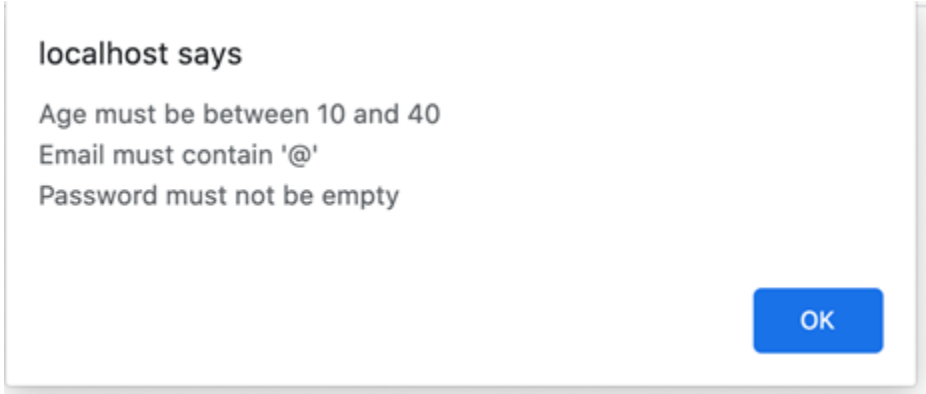
Example

wk5example5.html

Exercise 1

Given ex1.html which contains a form for multiple user inputs, write form validation code that checks:

- ID has a maximum length of 10 characters
- Email contains the character ‘@’
- Minimum age = 10, Maximum age = 40
- Password field is non-empty



Modifying Content

Two options to modify the content of HTML elements like <h1>, <p>, .

- innerText retrieves and sets the content of the tag as plain text (safer way)
- innerHTML retrieves and sets the content in HTML format (no encoding and can include HTML tags)

Examples: *wk5example6.html*

```
<div id="div">Hi…</h1>
<script>
  var element = document.getElementById("div");
  element.innerText = "Hello there…";
</script>
```

Modifying Attributes

Modifying attributes by setting its values.

Example:

wk5example7.html

Modifying Style

Syntax

```
document.getElementById(id).style.property = new style
```

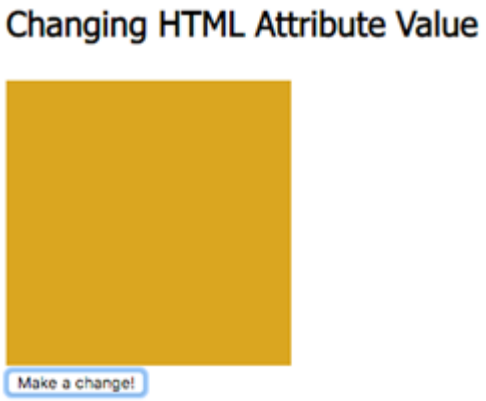
Example:

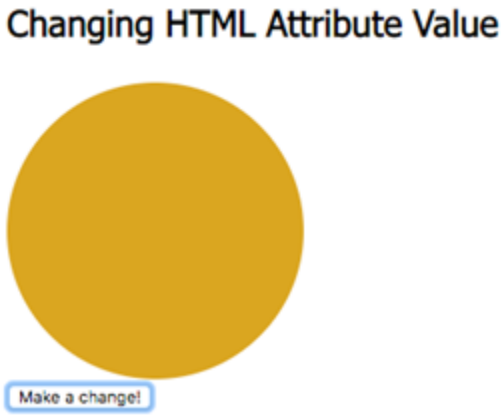
```
<p id="para">Hello Style!</p>

<script>
  document.getElementById("para").style.color = "red";
</script>
```

Exercise 2

Given that ex2.html initially shows a square box as shown in the figure below. Add CSS and Javascript code in ex2.html so that:

- When the button is clicked, the page shows a circle
- And when the button is clicked again, the page shows a square





Add Elements

JavaScript can create dynamic HTML content

The write() method writes HTML expressions or JavaScript code to a document.

The write() method is mostly used for testing: If it is used after an HTML document is fully loaded, it will delete all existing HTML.

Example: *wk5example8.html*

```
<!DOCTYPE html>
<html>
<body>
  <script>
    document.write("<h2>Hello</h2>");
  </script>
</body>
</html>
```

Add Elements

Create the element first and then append it to an existing element (e.g.).

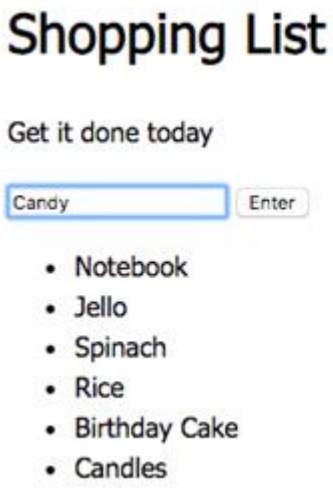
Example:

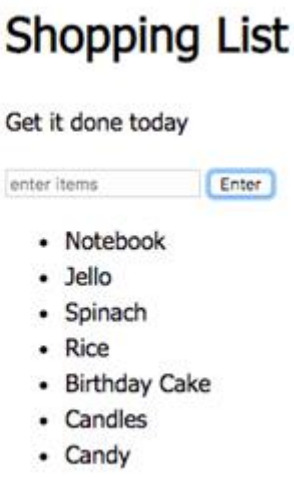
wk5example9.html

Exercise 3

Add HTML and JavaScript code in ex3.html such that when a user enters a new item in the input field and clicks “Enter” button, the new item should be added into the existing list as shown below.

You can assume that the user is no evil.





Remove Elements

Get the element node and use remove()

Example

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

<script>
  document.getElementById("div1").lastElementChild.remove();
</script>
```

Exercise 4

Continue with exercise 3, add the following functionality.

- Add a button next to each list item named “delete”
- Delete the item when the corresponding delete button is clicked.

Hint: You can use “this” or something related as a parameter when you call the function associated with the delete button (to identify the right item).

Event Handling

When an event occurs, all information about that event are stored in a type of object called event (e.g., the time that the event occurs, and the key that is pressed).

JavaScript can access this information (i.e., the event object) and react to it accordingly.

Example: *wk5example11.html*

```
<button onclick="myFunc()">Click!</button>
<script>
   function myFunc() {
      console.log(event);
   }
</script>
```

Example: “exercise *solution”/ex4.html*

We can find out all the information of the event in the debugger.

Event Handling

More mouse events

- onmouseover
- onmouseout
- onfocus

The MouseEvent

Example: *wk5example12.html*

```
<div onmouseover="mOver(this)" onmouseout="mOut(this)" >
   Mouse Over Me
</div>
<script>
   function mOver(obj) {
      obj.innerHTML = "Thank You"
   }

   function mOut(obj) {
      obj.innerHTML = "Mouse Over Me"
   }
</script>
```

This refers to the containing node, i.e., the div node.

Exercise 5

Add code in ex5.html such that

- the shape changes to square on mouse over event
- And it changes back to circle on mouse out event.

Add Event Listener

Instead of adding event listeners in HTML codes (e.g., <button onclick="myFunc()">), JavaScript provides a way to listen to events all from JavaScript code.

Use addEventListener() method to attach an event handler to the element that you are interested in.

Examples

wk5example13.html

Add Event Listener

Instead of adding event listeners in HTML codes (e.g., <button onclick="myFunc()">), JavaScript provides a way to listen to events all from JavaScript code.

Use addEventListener() method to attach an event handler to the element that you are interested in

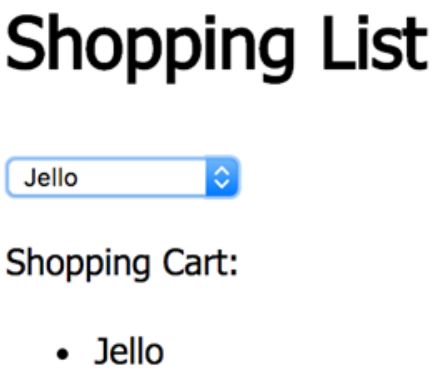
You can add many event handlers to one element. You can add many event handlers of the same type to one element, i.e., two "click" event handlers.

Examples

wk5example14.html

Exercise 6

Add code in ex6.html such that when an item is selected from the drop down list, the item is added into the shopping cart list as shown below. (Use addEvenetListener on the “change” event).



DOM: More Methods and Events

DOM Navigation:

https://www.w3schools.com/js/js_htmldom_navigation.asp

Events:

https://developer.mozilla.org/en-US/docs/Web/Events

Examples

```
//given focus to an element
document.getElementById("demo").focus();

//remove focus from an element
document.getElementById("demo").blur();
```

Take Away Message

To have a dynamic page, monitor events and change the DOM tree using Javascript.

Please be reminded to attempt Quiz 2 by this week.