ISSN:2790-1688 DOI: 10.56028/aetr.3.1.390

Object-oriented Design based Comprehensive Experimental Development of Document Object Model

Yanlong Wang [1, a], Jinhua Liu [2, b]

[1]School of Media Engineering, Communication University of Zhejiang, Hangzhou 310018, China;

[2] Experimental Center, Communication University of Zhejiang, Hangzhou 310018, China.

[a] wangyl@cuz.edu.cn, [b] 392127199@qq.com

Abstract. JavaScript code using Document Object Model (DOM) can realize the dynamic control of Web pages, which is the important content of the Web development technology course. The application of DOM is very flexible and includes many knowledge points, so it is difficult for students to master. In order to help students to understand each knowledge point and improve their engineering ability to solve practical problems, a DOM comprehensive experiment project similar to blind box is designed and implemented. This experimental project integrates knowledge points such as DOM events, DOM operations, and communication between objects. Practice has proved that running and debugging of the project can help students to understand and master relevant knowledge points.

Keywords: document object model; web programming; class diagram; object communication.

1. Introduction

WEB front-end development involves three major technologies: Hypertext Markup Language (HTML), Cascading Style Sheets (CSS) and JavaScript programming. HTML documents use tags to define the structure and content of Web pages. CSS, which defines the presentation of documents written in HTML, separate presentation from structure. JavaScript is a prototype-based, single-threaded, dynamic language that adds interactivity and custom behaviors to our sites, supporting object-oriented, event-driven programming model. Combining with document object model(DOM) that defines a standard for accessing HTML documents, JavaScript can manipulate the action of web pages. JavaScript process events by adding event handler on DOM object corresponding to HTML element. Accessing and manipulating the content of web pages by DOM object is the important content of the Web development technology course. The application of DOM is very flexible and involves many knowledge points, so it is difficult for students to master[1-4]. In order to enable students to effectively understand each knowledge point and improve their engineering ability to solve practical problems with the knowledge they have learned, a DOM comprehensive experiment project similar to blind box is designed and implemented. This experimental project integrates knowledge points such as DOM events, DOM operations, this pointer of function, and communication between objects. Running and debugging the project is helpful for students to understand these knowledge points[4].

2. The document object model basics

The Document Object Model (DOM) is an application programming interface for HTML and XML documents. It provides a structured map of HTML documents, as well as a set of methods to interface with the elements contained therein. It is a complete object-oriented representation of WEB pages, which can be used to find elements by their names or attributes, and then add, modify, or delete elements and their content.

2.1 The node tree

DOM represents an HTML document as a tree structure with the corresponding objects of HMTL elements as nodes, which is called a node tree. The browser builds its corresponding DOM

ISSN:2790-1688 DOI: 10.56028/aetr.3.1.390
node tree while rendering the HTML document. Typically, each attribute of an HTML element has a corresponding attribute in its corresponding DOM node object. Through the methods and properties of the DOM node object, user can access any element in the page, and perform operations such as element modification, deletion, and addition[5]. Consider the following HTML page:

```
<html>
<head>
    <title>Main</title>
</head>
<body>
    <a href="http://baidu.com">Baidu</a>
</body>
</html>
```

The DOM node tree corresponding to the above HTML document is shown in Figure 1.

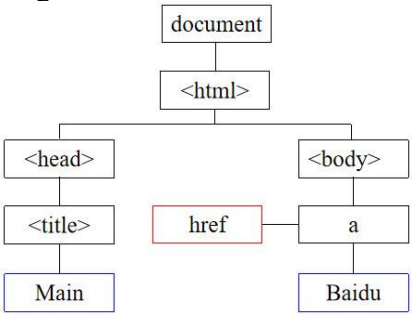


Fig. 1 A node tree corresponding to HTML document

In Figure 1, the document node which is an instance of HTMLDocument, also called document object, is the root node of the DOM node tree corresponding to the HTML document. The only child of the document node is the <html> element, which is called the document element. Each element within the page is referred to as a node in above figure. Nodes with black border are element nodes. Node with red border is attribute node. Nodes with blue border is text nodes.

2.2 Creating and adding elements

New element can be created by using the createElement() method of document object. This method accepts a single argument, which is the tag name of the element to create. To create a <img> element and set its src attribute, the following code can be used:

```
const image = document.createElement('img');
image.src = 'book.jpg';
```

Variable image is a element object corresponding to <img> element, since it is not part of the document tree, it doesn't affect the browser's display. The DOM also allows developers to change the document structure itself by adding and removing nodes. The following code adds the newly created <img> element to the document's <body> element:

```
document.body.appendChild(img);
```

After the above code is executed, the <img> element node is inserted into the DOM node tree, the browser will render the image immediately, and the web page will display the image.

2.3 Event and event handler

DOM events are responses to user input and other actions, and JavaScript interacts with HTML pages through events. DOM events involve three elements: event source (eg, button), event name (eg, mouse click) and event handler (eg, custom function). Element objects (event source) that can fire events have two methods to deal with the assignment and removal of event handlers: addEventListener() and removeEventListener(). These methods accept three arguments: the event name to handle, the event handler function, and a Boolean value indicating whether to call the event handler during the capture phase (true) or during the bubble phase (false).

ISSN:2790-1688 DOI: 10.56028/aetr.3.1.390

When an event is triggered in the DOM, all information related to the event is stored in an object called event. Normally, the event object is the only parameter passed to the event handler. The event handler function obtains various information related to the event from it[5].

2.3 Communication between objects

Generally, there are three ways to communicate between objects: 1) The bidirectional reference between communication objects is used to realize object communication. From a software engineering point of view, that's not good. Because this kind of including relation is sometimes contrary to common sense. For example, it is normal for a car to have a motor, but it is incredible that there is a car in the motor. 2) Fire custom events to implement object communication. The object registers the event listener for the custom event. After the event that is listened is triggered, the object can receive and handle it. 3) Use callback function to implement object communication. The object that includes another object creates a callback function. The method of included object calls that callback function, when object communication is needed.

3. Design of DOM Comprehensive Experiment Project

3.1 Function of DOM comprehensive experimental project

This comprehensive experimental project implements a simple function similar as blind box. The user can set the number of blind boxes, and the program generates a specified number of blind boxes and randomly puts gifts into each blind box. The user clicks on the selected blind box, and the gift in the blind box and its value will be displayed.

3.2 Interface design for DOM comprehensive experimental project

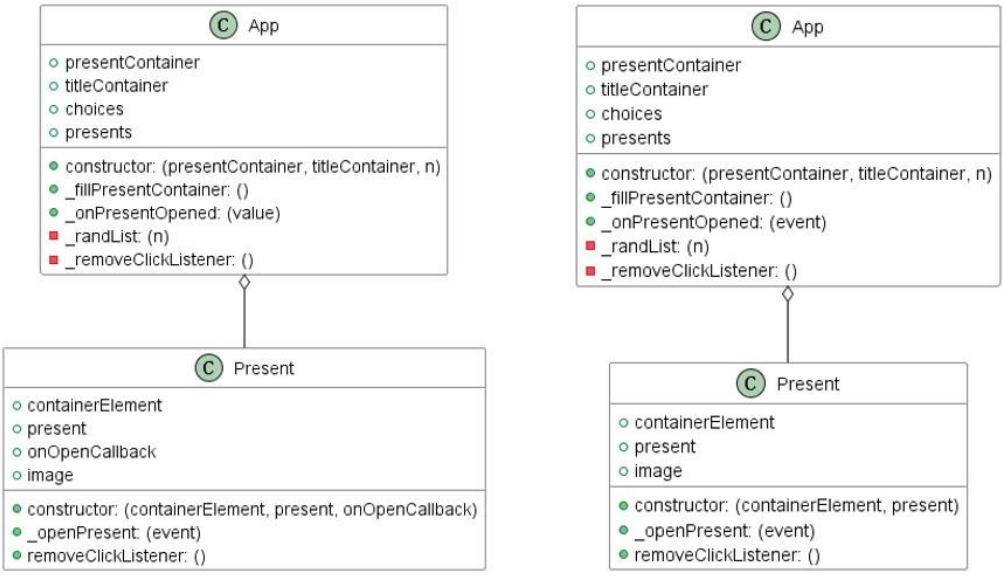
According to the function of the comprehensive experimental project, the user interface of the system should include: a text input box for inputting the number of blind boxes; a button to reload the blind boxes; pictures representing the blind boxes and hint for operation. The interface is shown in Figure 2.

Fig. 2 The system user interface

3.3 Class design for DOM comprehensive experimental project

The main purpose of the comprehensive experiment project is to help the students to understand key knowledge points, so as to cultivate the students' ability to comprehensively use the knowledge they have learned to solve engineering problems. In order to reduce the difficulty for students to understand key knowledge points, the classes should be as few and simple as possible. According to the functions of the comprehensive experimental project, two class, App and Present, are designed. The class App is responsible for creating the user interface, and the class Present is responsible for creating the blind box and handling the click event of the blind box. The class diagram of the system that communication between instance of class App and class Present is implemented by calling callback function is shown in Figure 3a. The class diagram of the system that communication between instance of class App and class Present is implemented by firing event is shown in Figure 3b[6,7].

ISSN:2790-1688 DOI: 10.56028/aetr.3.1.390



(a)Object communication with callback function (b)Object communication with event

Fig. 3 The class diagram of the system

In figure 3(a), the _onPresentOpened method of class App is a callback function acting as a real parameter, which is passed to constructor function of class Present and called after the blind box is opened. In figure 3(b), the _onPresentOpened method of class App is a event handler, which is called when the custom event named 'opened' is triggered. Statements in _onPresentOpened method involve many knowledge points such as manipulating HTML elements with DOM, DOM events and communication between objects. Statements in constructor method of class Present involve knowledge points such as adding elements with DOM, DOM events and binding pointer this for function. The _openPresent method of class Present is a event handler that is responsible for displaying present, which is called when blind box is clicked and opened. The difference between the method _openPresent in figure 3(a) and figure 3(b) is action that will be done after gift has been displayed. The action of method _openPresent in figure 3(a) is to execute callback function that is received by augment onOpenCallback. The action of method _openPresent in figure 3(b) is to create 'opened' event and fire it. Codes in method _openPresent include knowledge points such as manipulating HTML elements with DOM, DOM event and communication between objects. In above methods, each key knowledge point appears repeatedly, which is facilitate to students understand and master them.

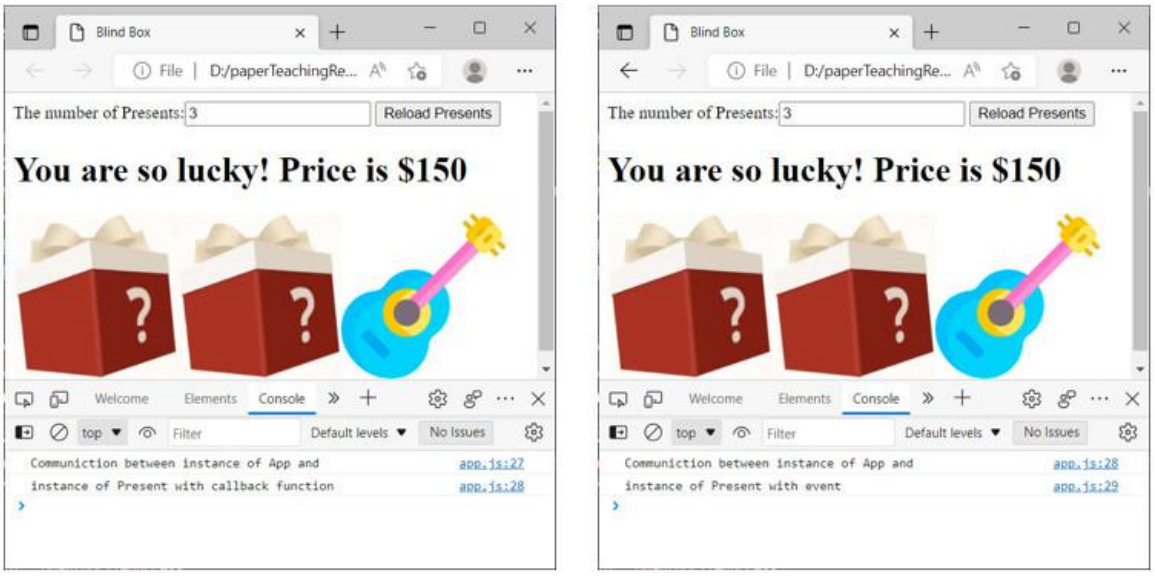
3.4 Interface implementation and comparison of the two project running result

Codes in the HTML document (index. html) that implement the system user interface shown in Figure 2 are as follows:

```
<body>
    <div>
        <label>The number of Presents:<input id="number" width="10" value="3"></label>
        <button onclick="reload()">Reload Presents</button>
    </div>
    <h1 id="title">Click a present to open it:</h1>
    <div id="presents"></div>
    <script src="present-source.js" defer></script>
    <script src="present.js" defer></script>
    <script src="app.js" defer></script>
    <script src="main.js" defer></script>
</body>
```

Running projects implemented based on class diagram figure 3 (a) and (b) by opening above index.hmtl file in browser, Figure 4 (a) and (b) respectively shows web page rendered by browser.

ISSN:2790-1688 DOI: 10.56028/aetr.3.1.390



(a) result based on class diagram in figure 3 (a) (b) result based on class diagram in figure 3 (b)

Fig. 4 web pages rendered by browser

According to figure 4, the two web pages are the same except the display of console window of browser, which shows that object communication is implemented with different method.

4. Summary

The document object model is one of the important contents of the WEB front-end development course. It involves many knowledge points, and it is difficult for students to master it. In this paper, a comprehensive experimental project that integrates multiple knowledge points is similar to blind box is designed and implemented. The realization and debugging of this project can train students' practical ability. Based on this project, students can effectively improve their engineering ability to solve practical problems by adding the functions of the system.

Acknowledgments. The research presented in the paper is supported by the research grant No. AB08190 of CUZ and No. 201902303021 of Industry-Univ. Cooperation Collaborative Education.

References

[1] Robert F, Dugan, Jr. A single semester web programming course model. Journal of Computing Sciences in Colleges, 2013, 29(1):26-34.

[2] Wang Y D, Zahadat N. Teaching Web Development in the Web 2.0 Era. The 10th ACM Conference on SIG- information Technology Education. Fairfax, USA: Association for Computing Machinery, 2009.

[3] Connolly R. Facing Backwards While Stumbling Forwards: The Future of Teaching Web Development. The 50th ACM Technical Symposium on Computer Science Education. Minneapolis, USA: Association for Computing Machinery, 2019.

[4] Kar S, Islam M M, Rahaman M. State-of-the-Art Reformation of Web Programming Course Curriculum in Digital Bangladesh. International Journal of Advanced Computer Science and Applications, 2020, 11(3):193-201.

[5] Matt Frisbie. Professional JavaScript for Web Developers, 4th Edition. Indianapolis：John Wiley & Sons, Inc., 2019.

[6] Zeaaraoui A, Bougroun Z, Belkasmi M G, et al. Object-oriented Analysis and Design Approach for the Requirements Engineering. Journal of Electronic Systems, 2012, 2(4):147-153.

[7] Kulkarni R N, Prasad P Pani Rama. Abstraction of UML Class Diagram from the Input Java Program. International Journal of Advanced Networking and Applications, 2021, 12(4): 4644-4649.