# CS 50 Web Design

## APCSP Module 2: Internet

## Unit 3: JavaScript

LECTURE 11: DOM AND JQUERY

DR. ERIC CHOU

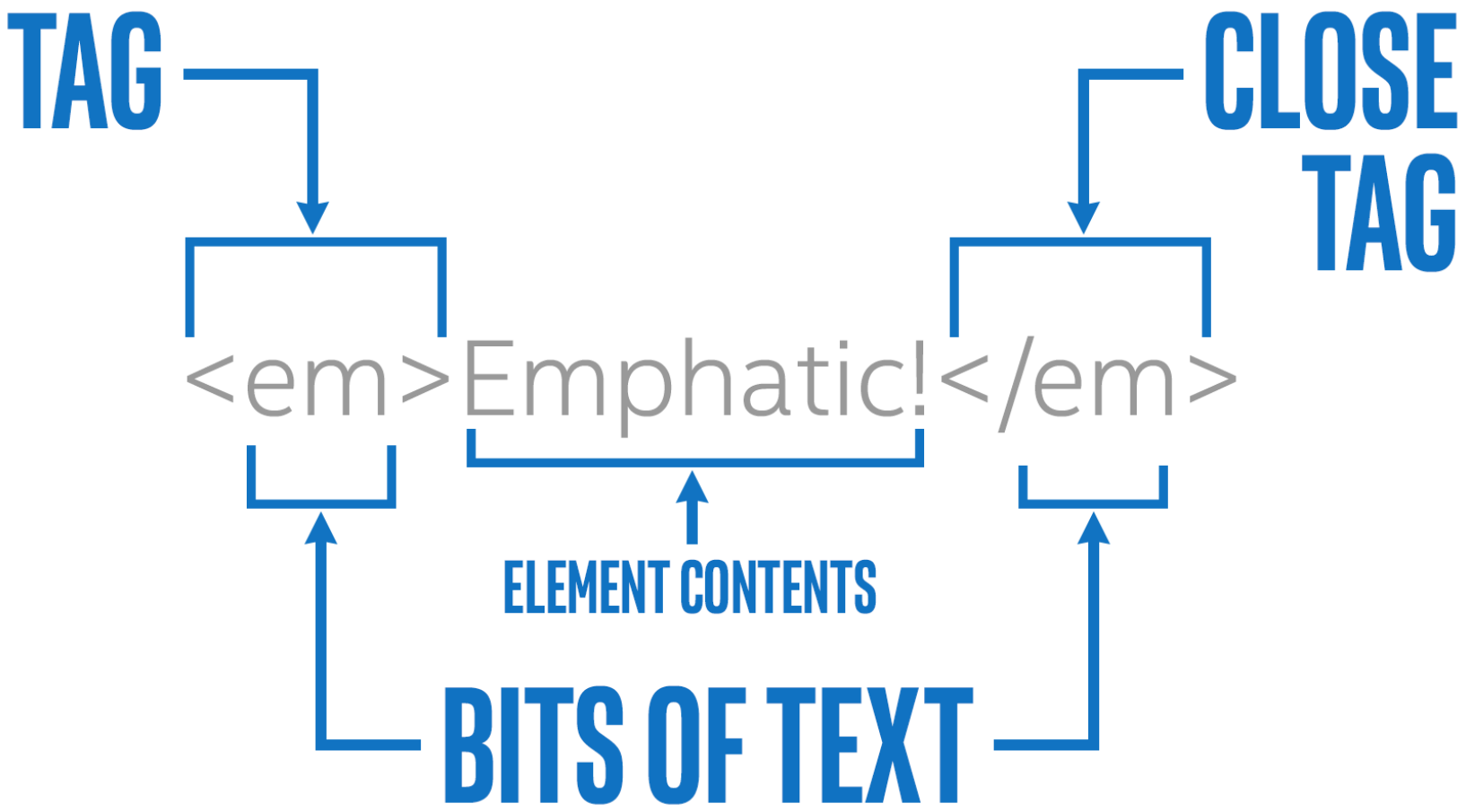
IEEE SENIOR MEMBER

# Objectives

- Elements and Tags in HTML
- Static pages and DOM
- Dynamics Pages
- HTML DOM, CSS CSSOM, and JavaScript
- jQuery on DOM/CSSOM

# Elements and Tags

SECTION 1


TAG
CLOSE TAG
Emphatic!
ELEMENT CONTENTS
BITS OF TEXT

# Tags and Elements

- HTML documents are made up of **elements**. An element starts with a **start tag** and ends with an **end tag**.
- For example, in our document so far we have two elements: h1 and p. The h1 element starts with the start tag <h1> and ends with the end tag </h1>. The p element starts with the start tag  and ends with the end tag . Anything between the opening and closing tags is the content of the element.
- Start tags consist of the element name surrounded by angle brackets: < and >. End tags are the same, but they have a forward slash (/) before the element name.

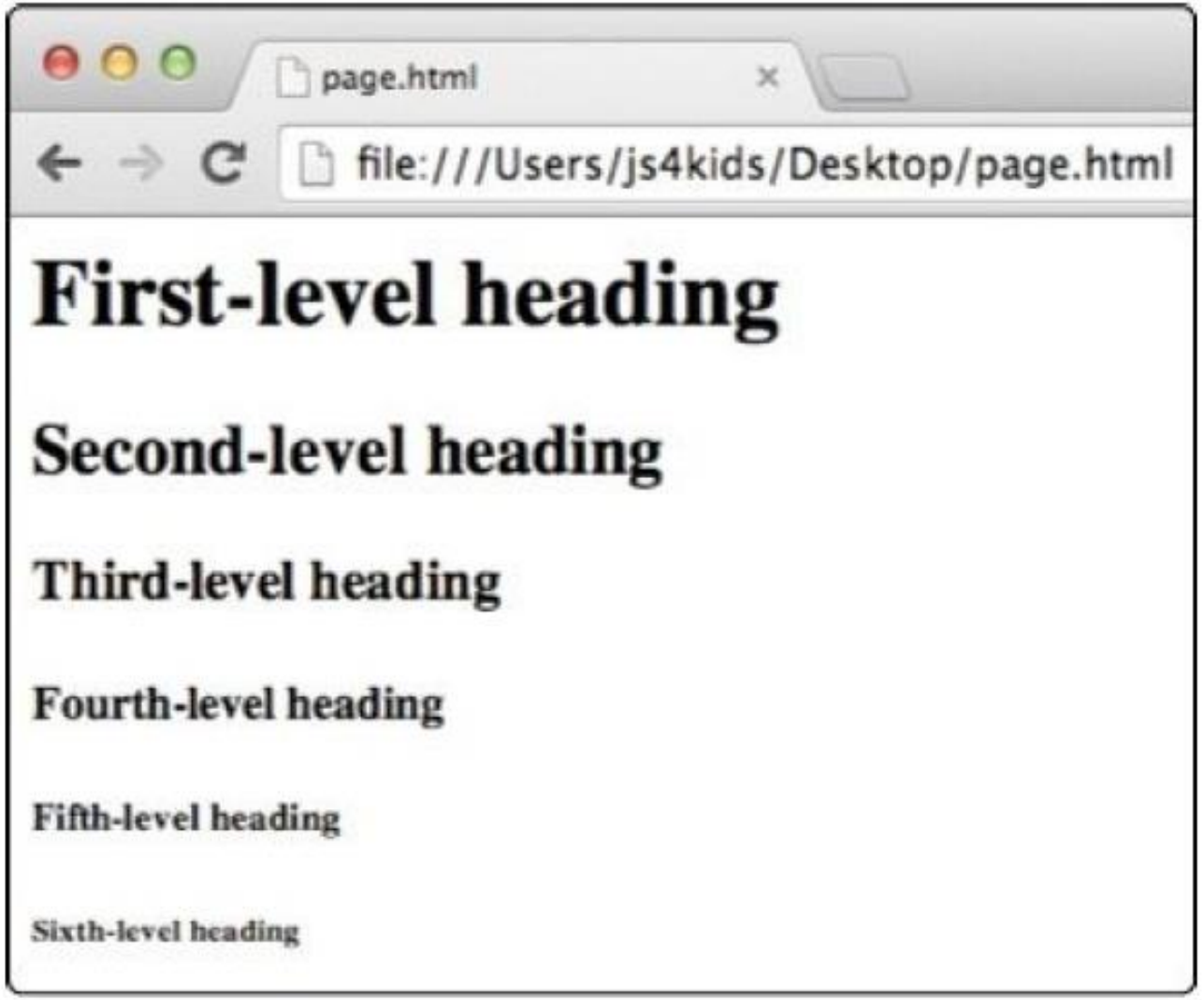
# HTML block level and inline elements

- In general, HTML elements can be divided into two categories : block level and inline elements.
  1. HTML block level elements can appear in the body of an HTML page.
  2. It can contain another block level as well as inline elements.
  3. By default, block-level elements begin on new lines.
  4. block level elements create larger structures (than inline elements).

# Heading Elements

- Each element has a special meaning and use. For example, the h1 element means “This is a top-level heading.” The content you put in between the opening and closing <h1> tags is displayed by the browser on its own line, in a large, bold font.
- There are six levels of heading elements in HTML: h1, h2, h3, h4, h5, and h6. They look like this:

```
<h1>First-level heading</h1>

<h2>Second-level heading</h2>

<h3>Third-level heading</h3>

<h4>Fourth-level heading</h4>

<h5>Fifth-level heading</h5>

<h6>Sixth-level heading</h6>
```


page.html
file:///Users/js4kids/Desktop/page.html
First-level heading
Second-level heading
Third-level heading
Fourth-level heading
Fifth-level heading
Sixth-level heading

# The p Element

- The p element is used to define separate paragraphs of text. Any text you put between <p> tags will display in a separate paragraph, with some space above and below the paragraph. Let's try creating multiple p elements.
- Add this new line to your page.html document (the old lines are shown in gray):

```
<h1>Hello world!</h1>
<p>My first web page.</p>
<p>Let's add another paragraph.</p>
```


page.html
file:///Users/js4kids/Desktop/page.html
Hello world!
My first web page.
Let's add another paragraph.

# Whitespace in HTML and Block-Level Elements

- What would our page look like without the tags? Let’s take a look:
  - Hello world!
  - My first web page.
  - Let's add another paragraph.
- Oh no! Not only have we lost the formatting, but everything’s on one long line! The reason is that in HTML, all *whitespace* is collapsed into a single space. Whitespace means any character that results in blank space on the page — for example, the space character, the tab character, and the newline character (the character that is inserted when you press ENTER or RETURN).
- Any blank lines you insert between two pieces of text in an HTML document will get collapsed into a single space.
- The p and h1 elements are called *block-level* elements because they display their content in a separate block, starting on a new line, and with any following content on a new line.

# List of block level elements

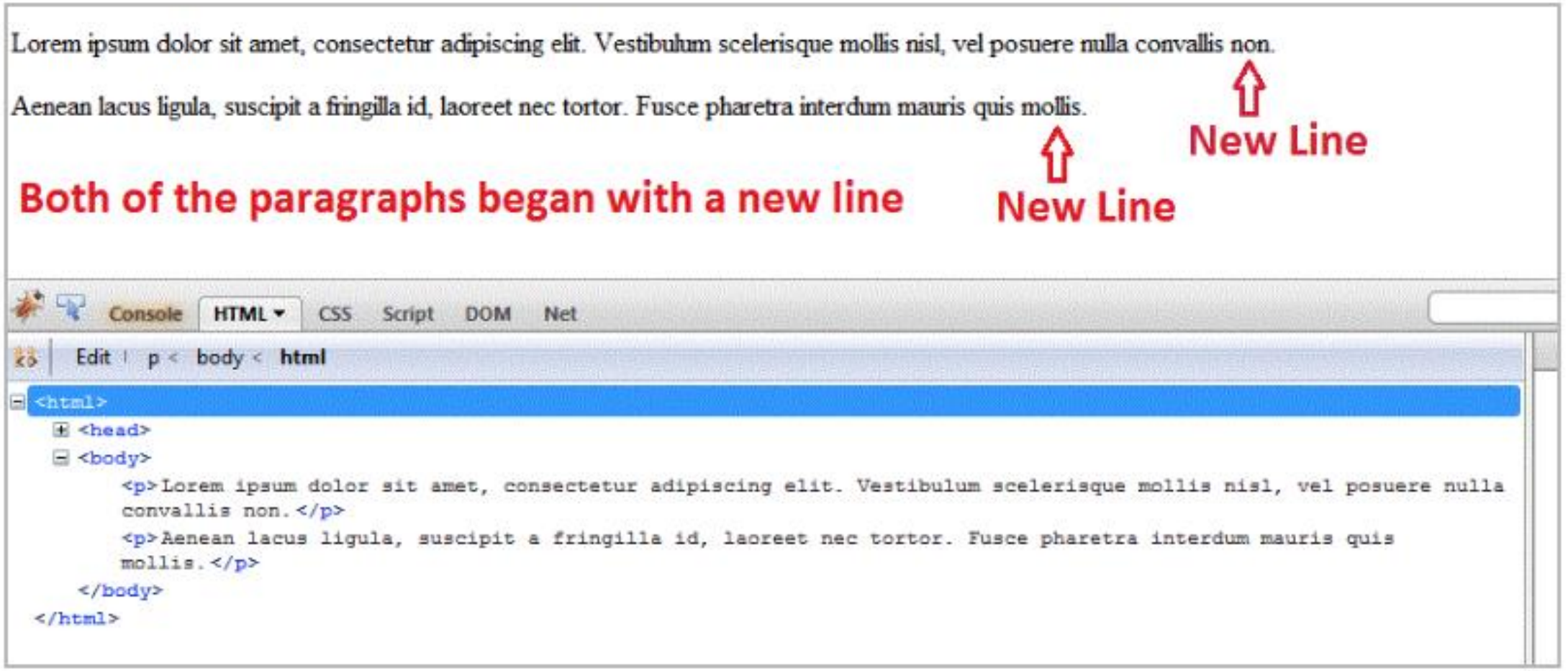
- p
- h1, h2, h3, h4, h5, h6
- ol, ul
- pre
- address
- blockquote
- dl
- div
- fieldset
- form
- hr
- noscript
- table

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Example of HTML block level element</title>
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum scelerisque mollis nisl,
vel posuere nulla convallis non.</p>
<p>Aenean lacus ligula, suscipit a fringilla id, laoreet nec tortor. Fusce pharetra interdum mauris quis mollis.</p>
</body>
</html>
```

Copy

# Demo Program: block.html

## Pictorial presentation



# HTML inline elements

1. **HTML inline level elements** can appear in the body of an HTML page.
2. It can contain data and other **inline** elements.
3. By default, **inline elements** do not begin on new lines.
4. **inline elements** create shorter structures (than block level elements).

**List of inline elements**

- b, big, i, small, tt
- abbr, acronym, cite, code, dfn, em, kbd, strong, samp, var
- a, bdo, br, img, map, object, q, script, span, sub, sup
- button, input, label, select, textarea

# Inline Elements

- Let's add two more elements to our document, em and strong:

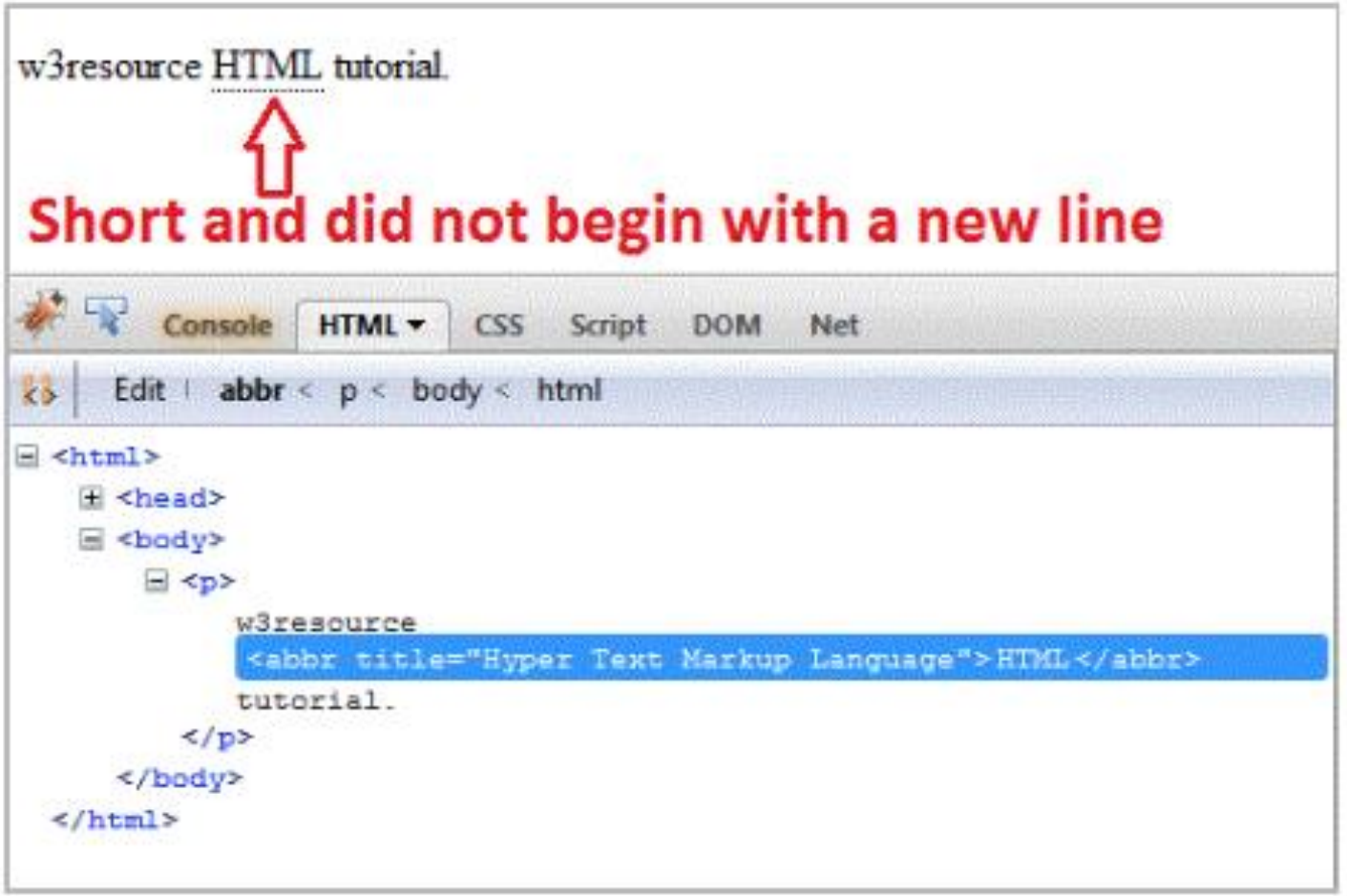
```
<h1>Hello world!</h1>
<p>My <em>first</em> <strong>web page</strong>.</p>
<p>Let's add another <strong><em>paragraph</em></strong>.</p>
```

- The em element makes its content italic. The strong element makes its content bold. The em and strong elements are both inline elements, which means that they don't put their content onto a new line, as block-level elements do.

```
 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Example of HTML block level element</title>
</head>
<body>
<p>w3resource <abbr title="Hyper Text Markup Language">HTML</abbr> tutorial.</p>
</body>
</html>
```

# Demo Program

## Pictorial presentation



# Static pages

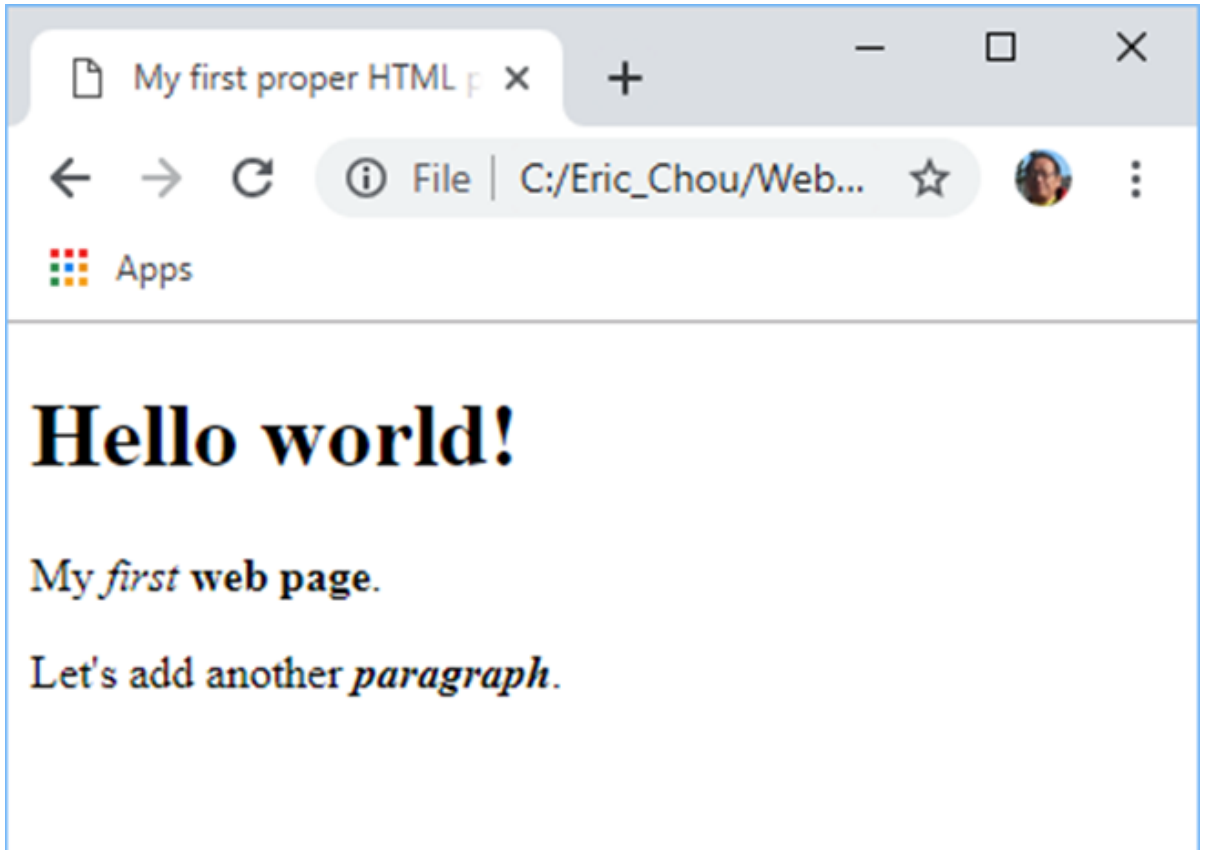
SECTION 2

- What we've looked at so far is really just a snippet of HTML. A full HTML document requires some extra elements. Let's take a look at an example of a complete HTML document and what each part means. Update your *page.html* file with these new elements:

# A Full HTML Document

Demo Program: static1.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>My first proper HTML page</title>
5  </head>
6  <body>
7  <h1>Hello world!</h1>
8  <p>My <em>first</em> <strong>web page</strong>.</p>
9  <p>Let's add another <strong><em>paragraph</em></strong>.</p>
10 </body>
11 </html>
```




# A Full HTML Document

Demo Program: static1.html

# A Full HTML Document

Demo Program: static1.html

- Let's take a walk through the elements in our static1.html file. The <!DOCTYPE html> tag is just a declaration. It simply says, "This is an HTML document." Next comes the opening <html> tag (the closing </html> tag is at the very end). All HTML documents must have an html element as their outermost element.

# A Full HTML Document

## Demo Program: static1.html

- There are two elements inside the html element: head and body. The head element contains certain information about your HTML document, such as the title element, which contains the document's title. For example, notice that in Figure 5-6, the title in the browser tab — "My first proper HTML page" — matches what we entered in the title element. The title element is contained inside the head element, which is contained inside the html element.
- The body element contains the content that will be displayed in the browser. Here, we've just copied the HTML from earlier in the chapter.

# HTML Hierarchy

- HTML elements have a clear hierarchy, or order, and can be thought of as a kind of upside-down tree.
- You can see how our document would look as a tree in Figure 5-7.

# HTML Hierarchy

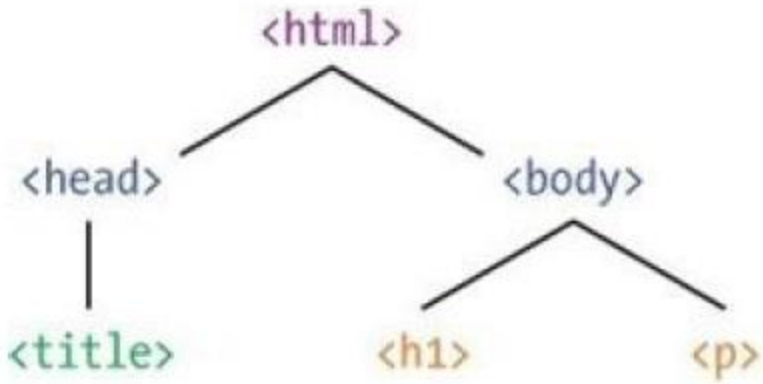




*Figure 5-7. The elements from Figure 5-6, shown as a tree*

# HTML Hierarchy

- The top element is the html element. It contains the head and body elements. The head contains the title element, and the body contains the h1 and p elements. The browser interprets your HTML according to this hierarchy. We'll look at how to change the document structure later, in Chapter 9.
- Figure 5-8 shows another way of visualizing the HTML hierarchy, as a set of nested boxes.

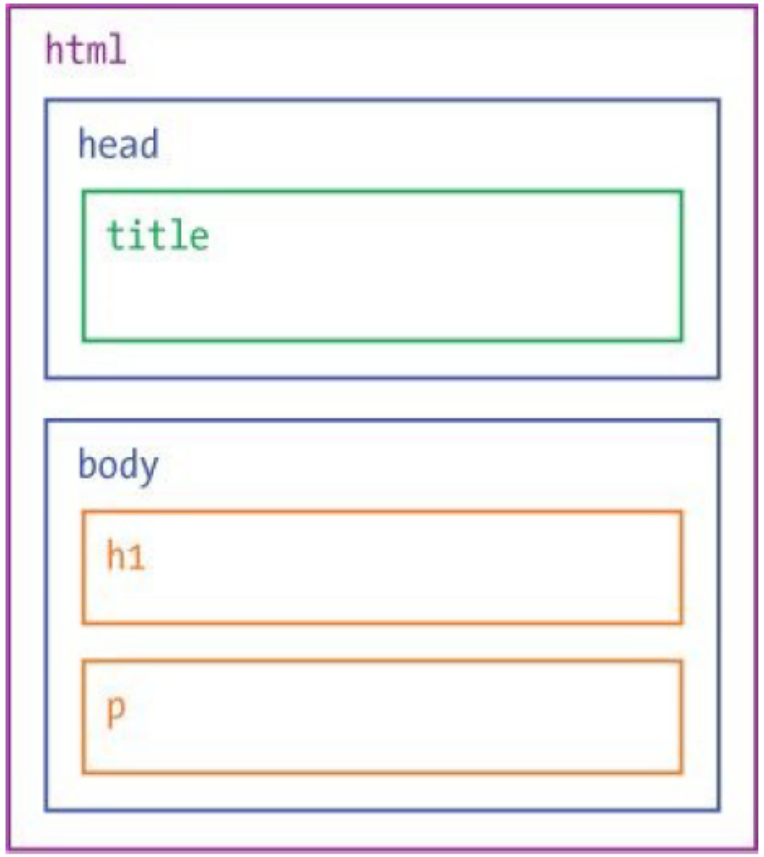


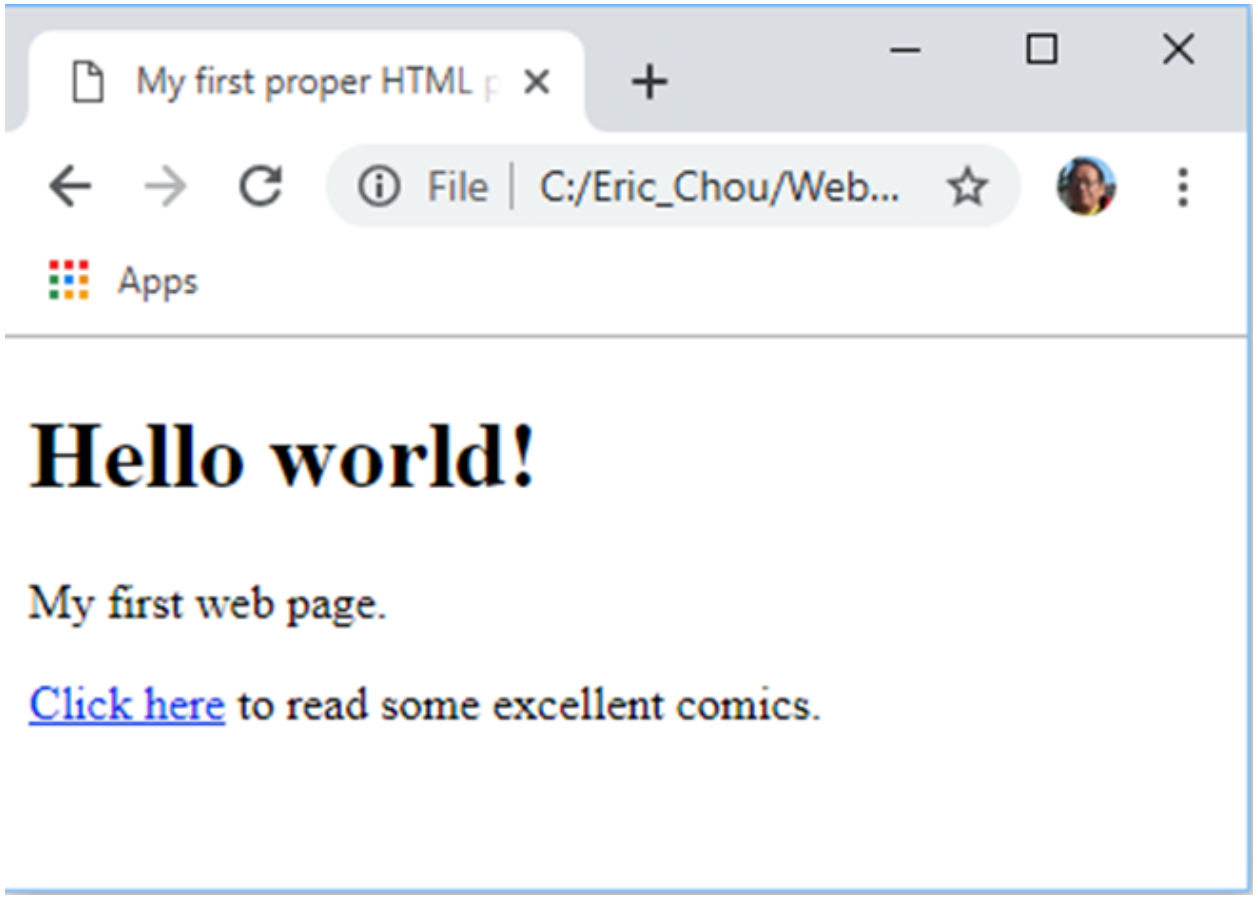

*Figure 5-8. The HTML hierarchy, shown as nested boxes*

# Adding Links to Your HTML

## Demo Program: static2.html

- Earlier in this chapter, we learned that the *HT* in *HTML* stands for HyperText, or linked text. HTML documents can contain *hyperlinks* (*links* for short) that take you to other web pages. The a element (for *anchor*) creates a link element.
- Modify your HTML document to match the following example: delete the second p element and the  and  tags, and then add the new colored code to create a link to *http://xkcd.com/*:

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>My first proper HTML page</title>
5  </head>
6  <body>
7  <h1>Hello world!</h1>
8  <p>My first web page.</p>
9  <p><a href="http://xkcd.com">Click here</a> to read some excellent
10 comics.</p>
11 </body>
12 </html>
```




# Adding Links to Your HTML

## Demo Program: static2.html

# Link Attributes

- Let’s take a closer look at how we created that HTML link. To tell the browser where to go when you click the a element, we added something called an *attribute* to the anchor element. Attributes in HTML elements are similar to key-value pairs in JavaScript objects. Every attribute has a name and a value.
- Here’s the xkcd link we created again:

```
<a href="http://xkcd.com">Click here</a>
```

- In this case, the attribute name is href and the attribute value is "http://xkcd.com". The name href stands for *hypertext reference*, which is a fancy way of saying “web address.”
- Figure 5-10 shows all the parts of the link.

# Link Attributes


*Figure 5-10. The basic syntax for creating a hyperlink*

# Title Attributes

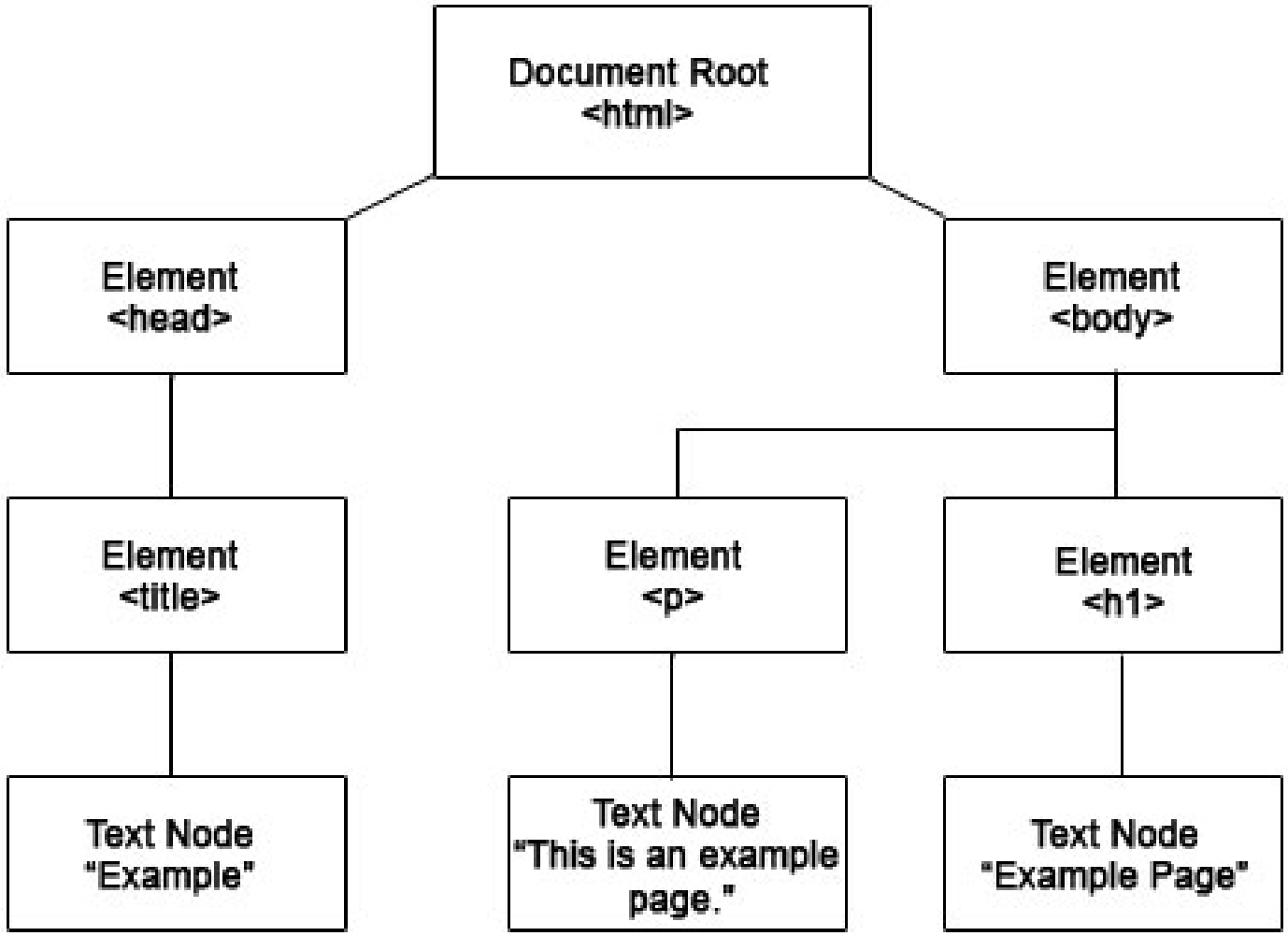
- Another attribute we can add to links is the title attribute. This attribute sets the text you see when you hover your mouse over a link. For example, change the opening <a> tag so it looks like this:

```
<a href="http://xkcd.com" title="xkcd: Land of geeky comics!">Click here</a>
```

- Now reload the page. When you hover your cursor over the link, you should see the text “xkcd: Land of geeky comics!” floating above the page.

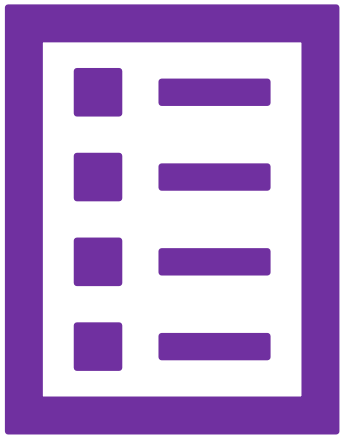
# Dynamic pages

SECTION 3


Document Root
<html>
Element
<head>
Element
<body>
Element
<title>
Element
<p>
Element
<h1>
Text Node
"Example"
Text Node
"This is an example
page."
Text Node
"Example Page"

# 10 Essential DOM Methods & Techniques for Practical JavaScript

1. **getElementById**
2. **getElementsByTagName**
3. **Node Methods**
4. **createElement**
5. **appendChild**
6. **removeChild**
7. **getAttribute**
8. **setAttribute**
9. **document.forms**
10. **innerHTML**



# Access of Elements

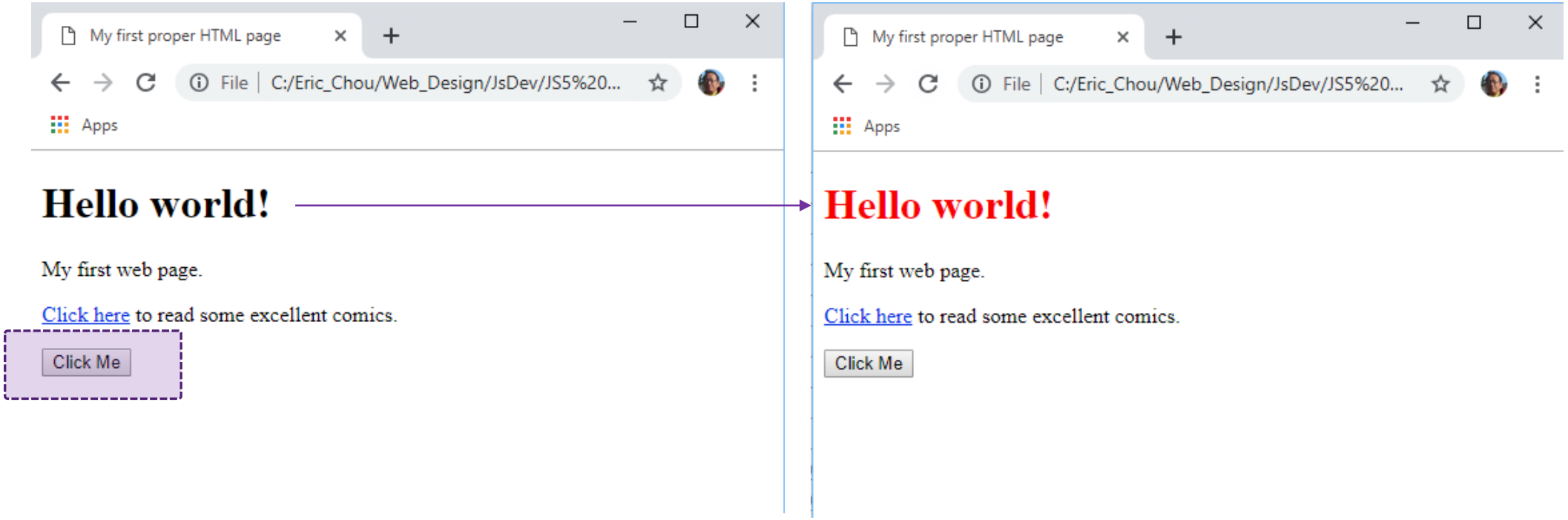
# getElementById("id")

## Demo Program: dynamic1.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>My first proper HTML page</title>
5  </head>
6  <body>
7  <h1 id="h1">Hello world!</h1>
8  <p>My first web page.</p>
9  <p><a href="http://xkcd.com">Click here</a> to read some excellent
10 comics.
11 </p>
12 <button onclick="change()">Click Me</button>
13 <script>
14     function change(){
15        const elem = document.getElementById("h1");
16        elem.style.color="red"; // change color to red
17     }
18 </script>
19 </body>
20 </html>
```

# getElementById("id")

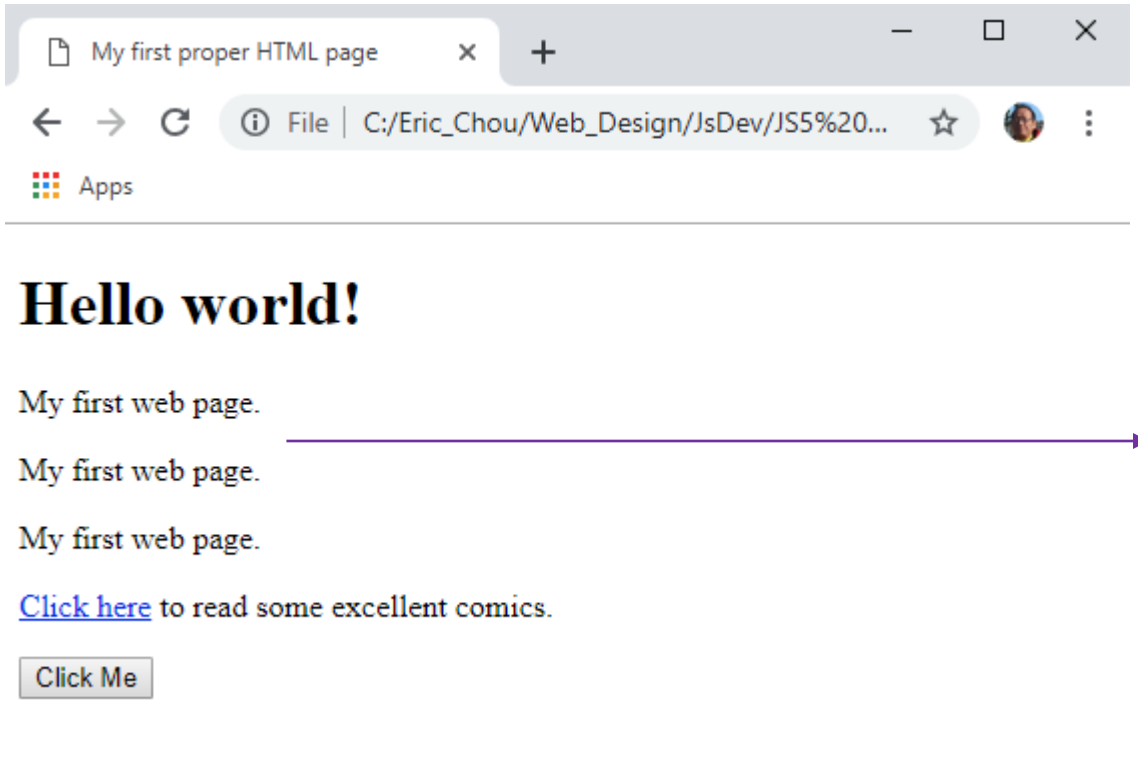
## Demo Program: dynamic1.html

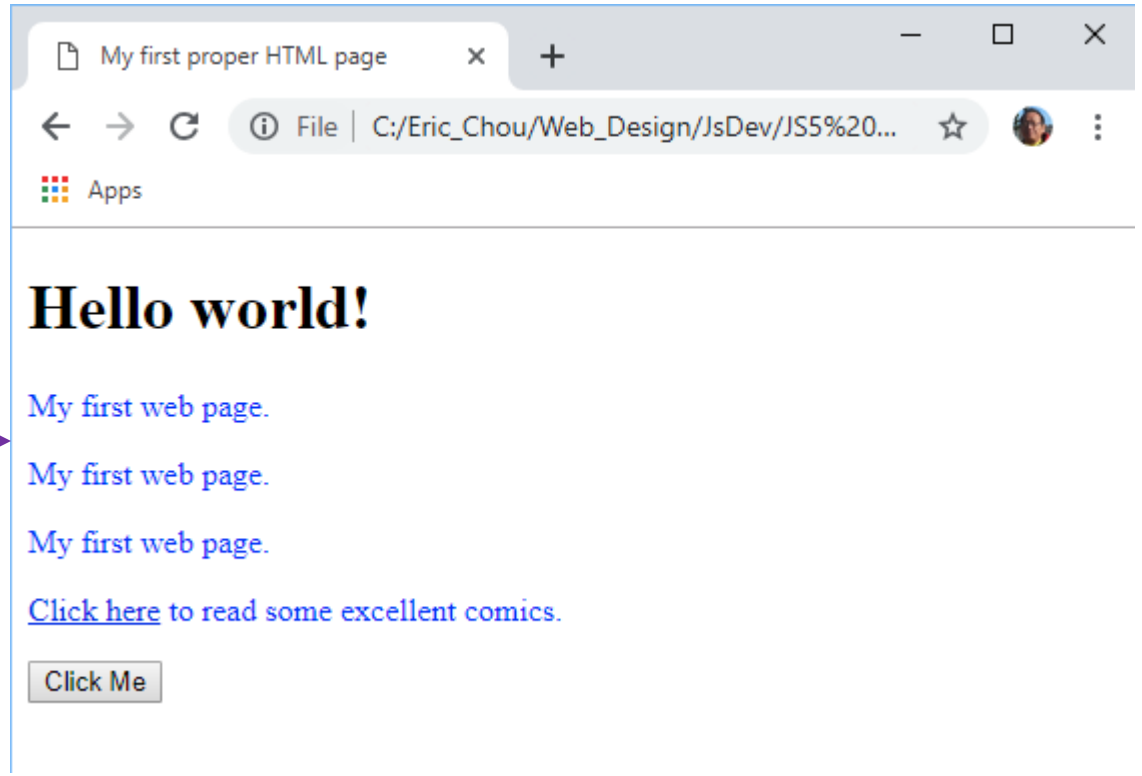


# getElementsByTagName("tn")

## Demo Program: dynamic2.html

```
<!DOCTYPE html>
<html>
<head>
<title>My first proper HTML page</title>
</head>
<body>
<h1 id="h1">Hello world!</h1>
<p>My first web page.</p>
<p>My first web page.</p>
<p>My first web page.</p>
<p><a href="http://xkcd.com">Click here</a> to read some excellent
comics.
</p>
<button onclick="change()">Click Me</button>
<script>
    function change(){
        var elems = document.getElementsByTagName("p");
        for (var i=0; i<elems.length; i++)
            elems[i].style.color="blue"; // change color to red
    }
</script>
</body>
</html>
```

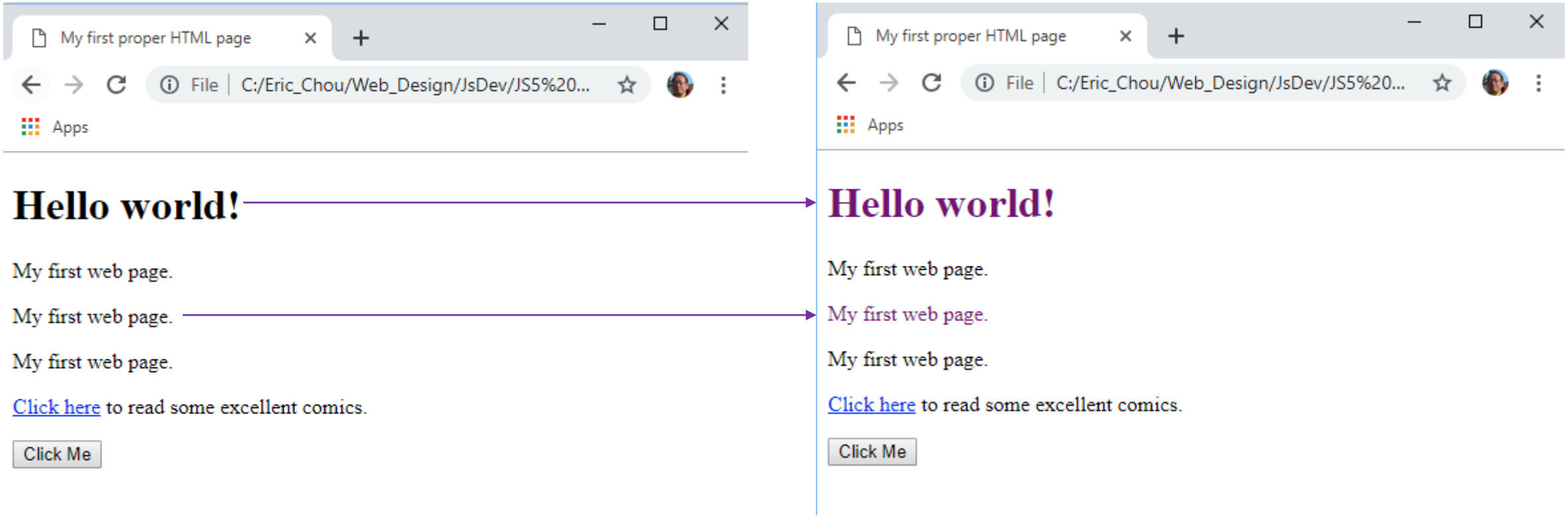

My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me


My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me

# getElementsByClassName("tn")

## Demo Program: dynamic3.html

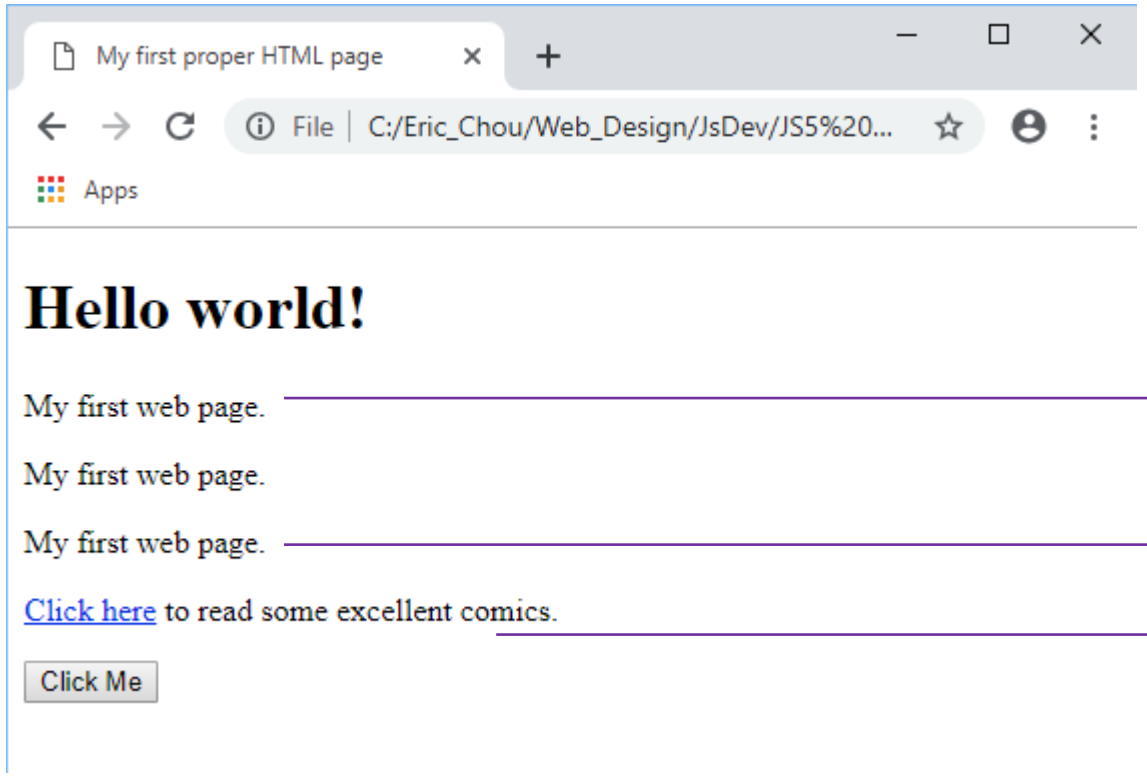
```
<!DOCTYPE html>
<html>
<head>
<title>My first proper HTML page</title>
</head>
<body>
<h1 id="h1" class="important">Hello world!</h1>
<p>My first web page.</p>
<p class="important">My first web page.</p>
<p>My first web page.</p>
<p><a href="http://xkcd.com">Click here</a> to read some excellent
comics.
</p>
<button onclick="change()">Click Me</button>
<script>
    function change(){
       var elems = document.getElementsByClassName("important");
       for (var i=0; i<elems.length; i++)
           elems[i].style.color="purple"; // change color to red
    }
</script>
</body>
</html>
```

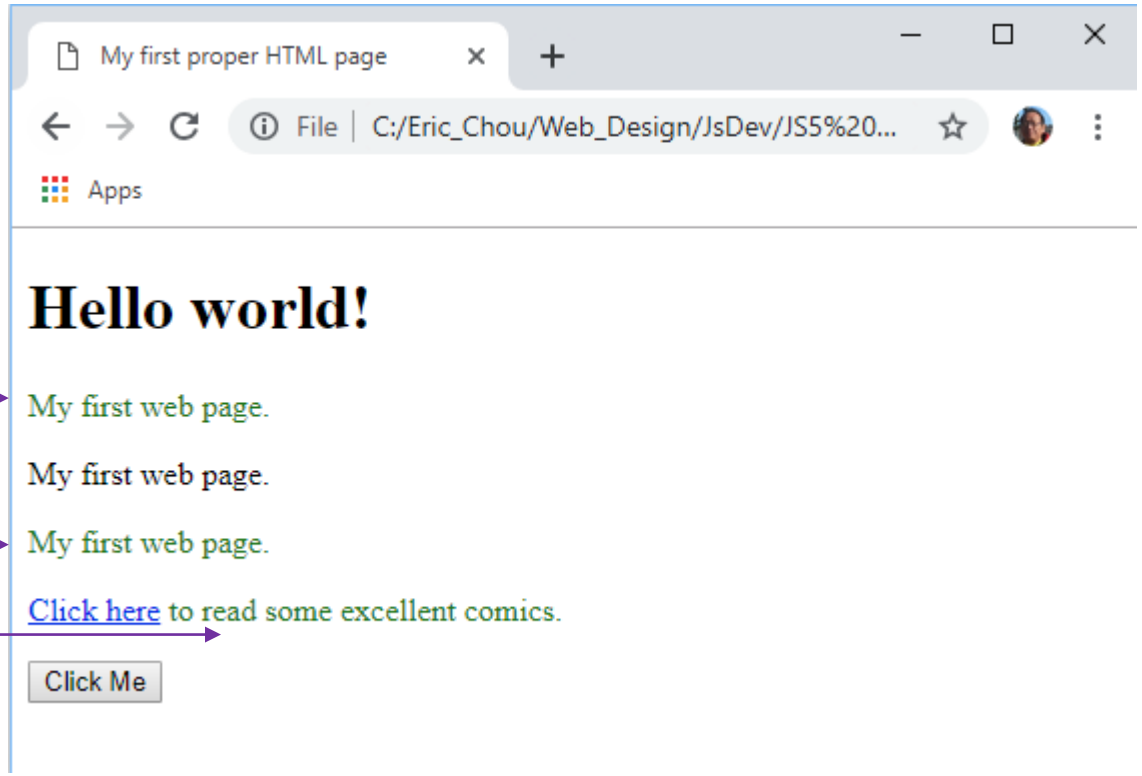

My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me
My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me

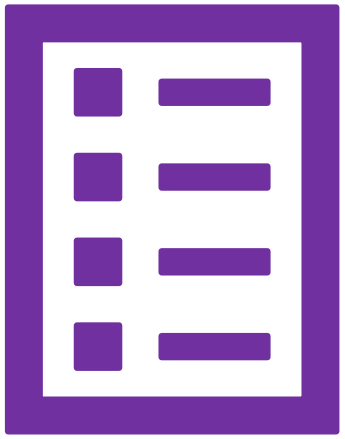
# getElementsByName("name")

## Demo Program: dynamic4.html

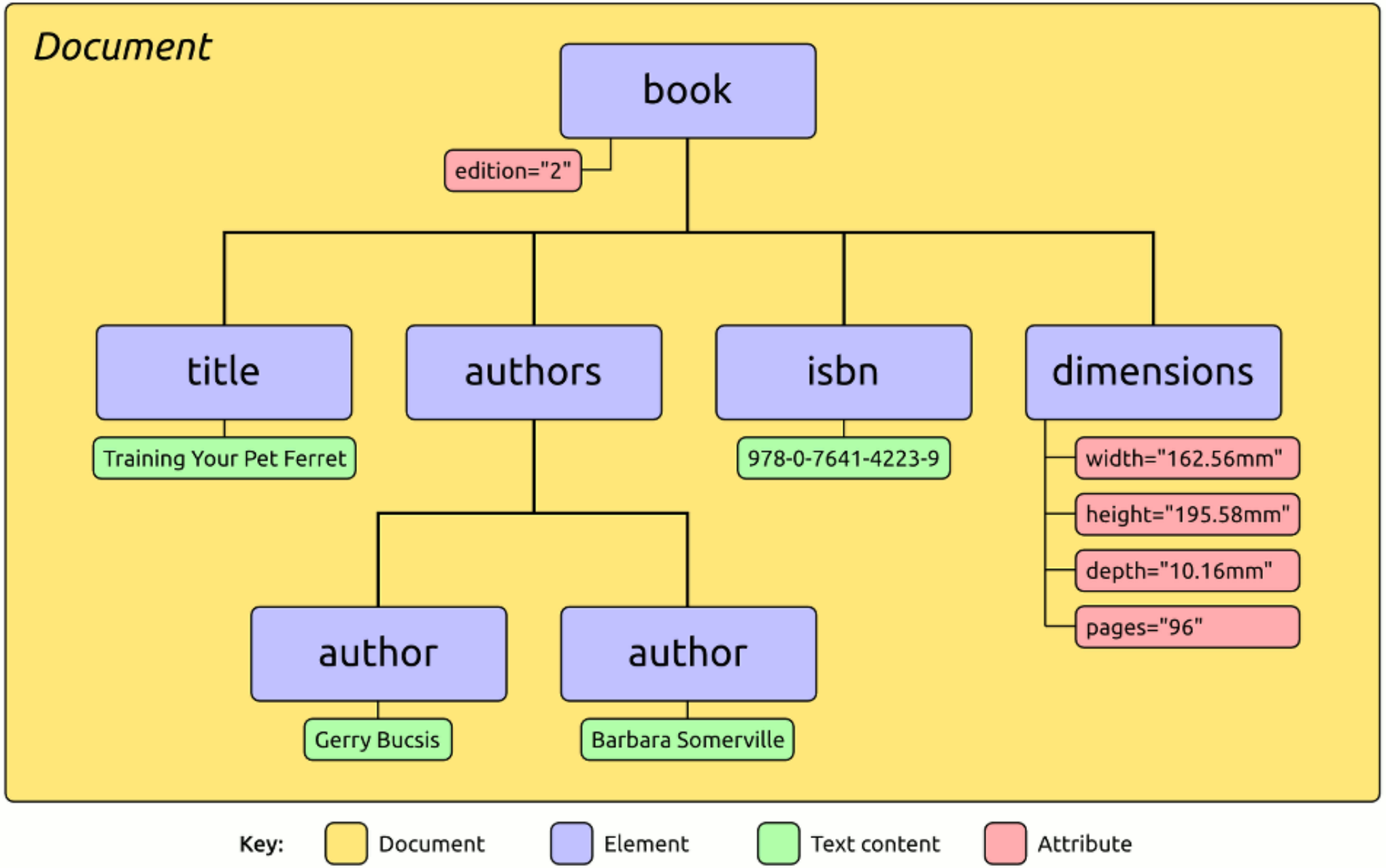
```
<!DOCTYPE html>
<html>
<head>
<title>My first proper HTML page</title>
</head>
<body>
<h1 id="h1" class="important">Hello world!</h1>
<p name="green">My first web page.</p>
<p class="important">My first web page.</p>
<p name="green">My first web page.</p>
<p name="green"><a href="http://xkcd.com">Click here</a> to read some
excellent
comics.
</p>
<button onclick="change()">Click Me</button>
<script>
    function change(){
       var elems = document.getElementsByName("green");
       for (var i=0; i<elems.length; i++)
           elems[i].style.color="green"; // change color to red
    }
</script>
</body>
</html>
```


My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me


My first proper HTML page
File | C:/Eric_Chou/Web_Design/JsDev/JS5%20...
Apps
Hello world!
My first web page.
My first web page.
My first web page.
Click here to read some excellent comics.
Click Me

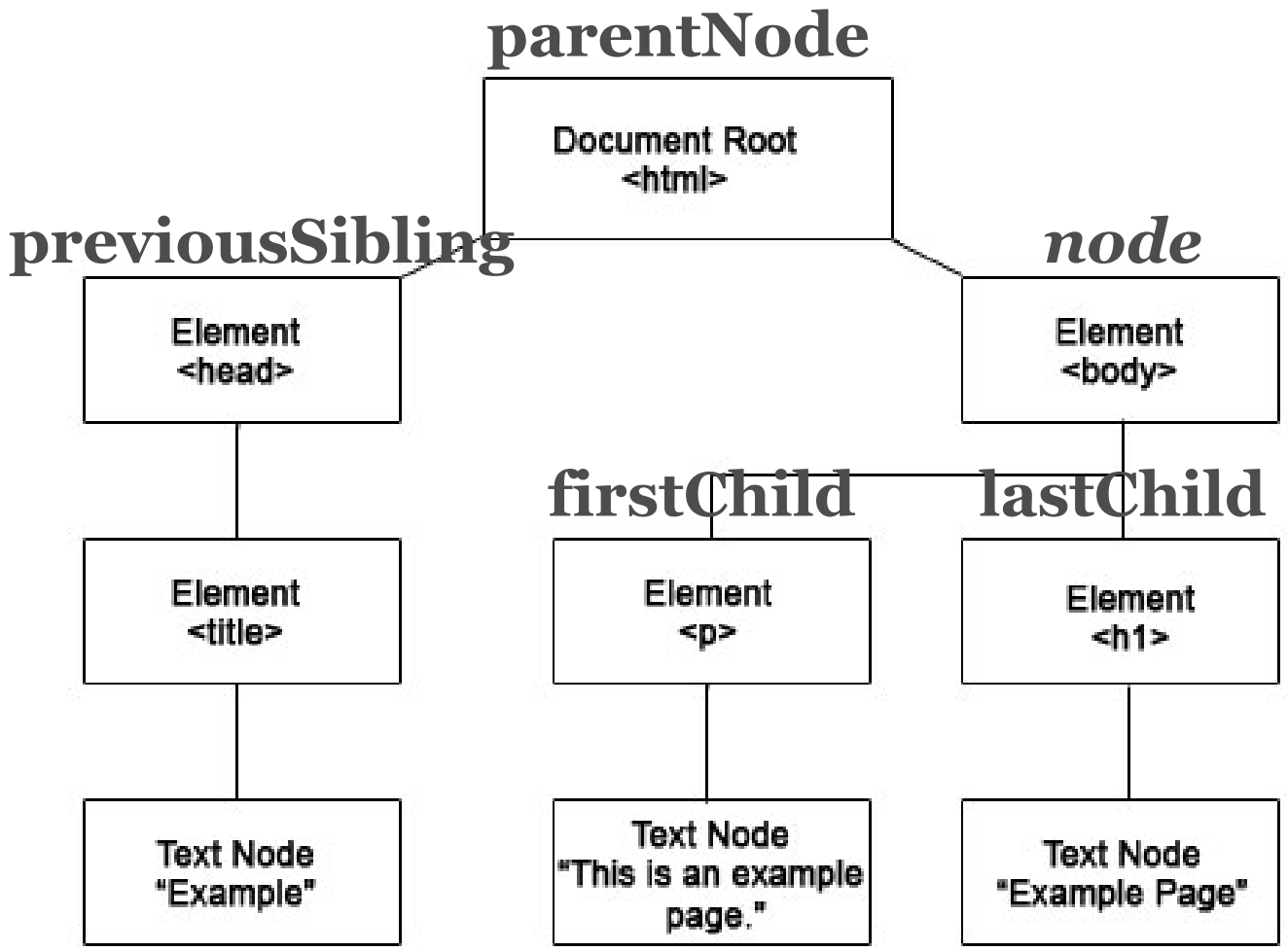


# Using Node to Access Neighbors


Document
book
edition="2"
title
Training Your Pet Ferret
authors
author
Gerry Bucsis
author
Barbara Somerville
isbn
978-0-7641-4223-9
dimensions
width="162.56mm"
height="195.58mm"
depth="10.16mm"
pages="96"
Key:
Document
Element
Text content
Attribute

The different node methods available through DOM manipulation are as follows:

***node*.childNodes**
***node*.firstChild**
***node*.lastChild**
***node*.parentNode**
***node*.nextSibling**
***node*.previousSibling**



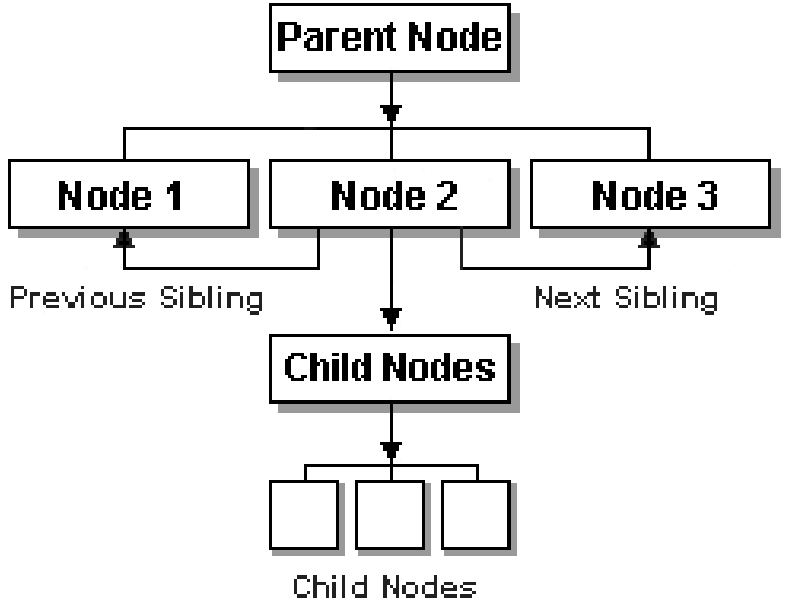
# Node Accessing

## Demo Program: dynamic5.html

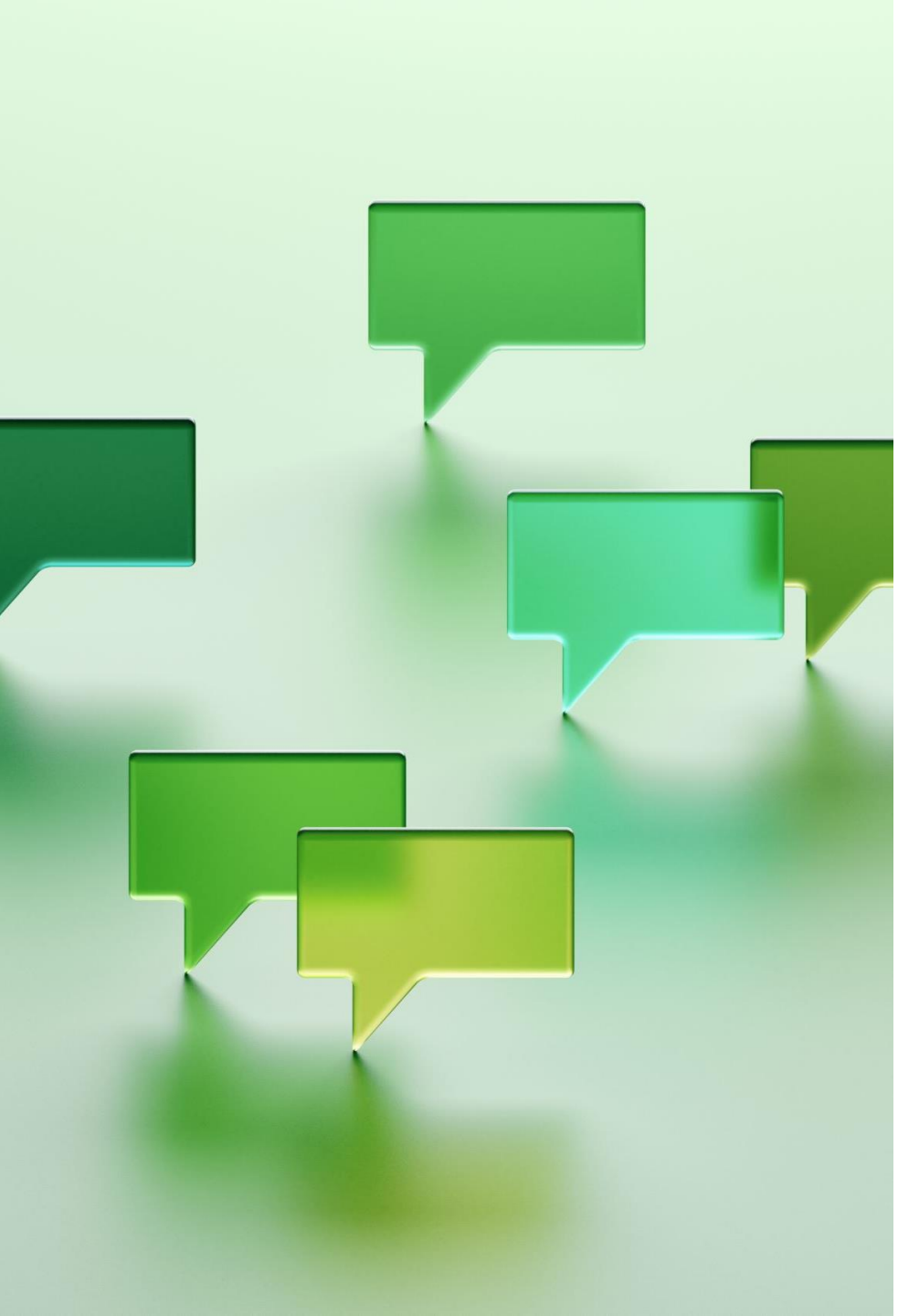
These are Data Fields

- node.childNodes
- node.firstChild
- node.lastChild
- node.parentNode
- node.nextSibling
- node.previousSibling

Be aware that there might be hidden nodes.



```
<!DOCTYPE html>
<html id="html">
<head id="head">
<title id="title">My first proper HTML page</title>
</head>
<body id="body">
<h1 id="h1">Hello world!</h1>
<p id="p1">My first web page.</p>
<p id="p1"><a href="http://xkcd.com" id="a1">Click here</a> to read some excellent
comics.
</p>
<button onclick="names()" id="button">Click Me</button>
<script id="script">
    function names(){
        var node = document.getElementById("body");
        document.write("<h1>Print out node's neighbors: </h1>");
        document.write("Node: "+node.id+"<br>");
        document.write("Parent: "+node.parentNode.id+"<br>");
        document.write("PreviousSibling's nextSibling:"
                       +node.previousSibling.nextSibling.id+"<br>");
        var children = node.childNodes;
            for (var i=0; i<children.length; i++){
                if (children[i]) document.write("Child["+i+"]="+children[i].id+"<br>");

            }
    }
</script>
</body>
</html>
```

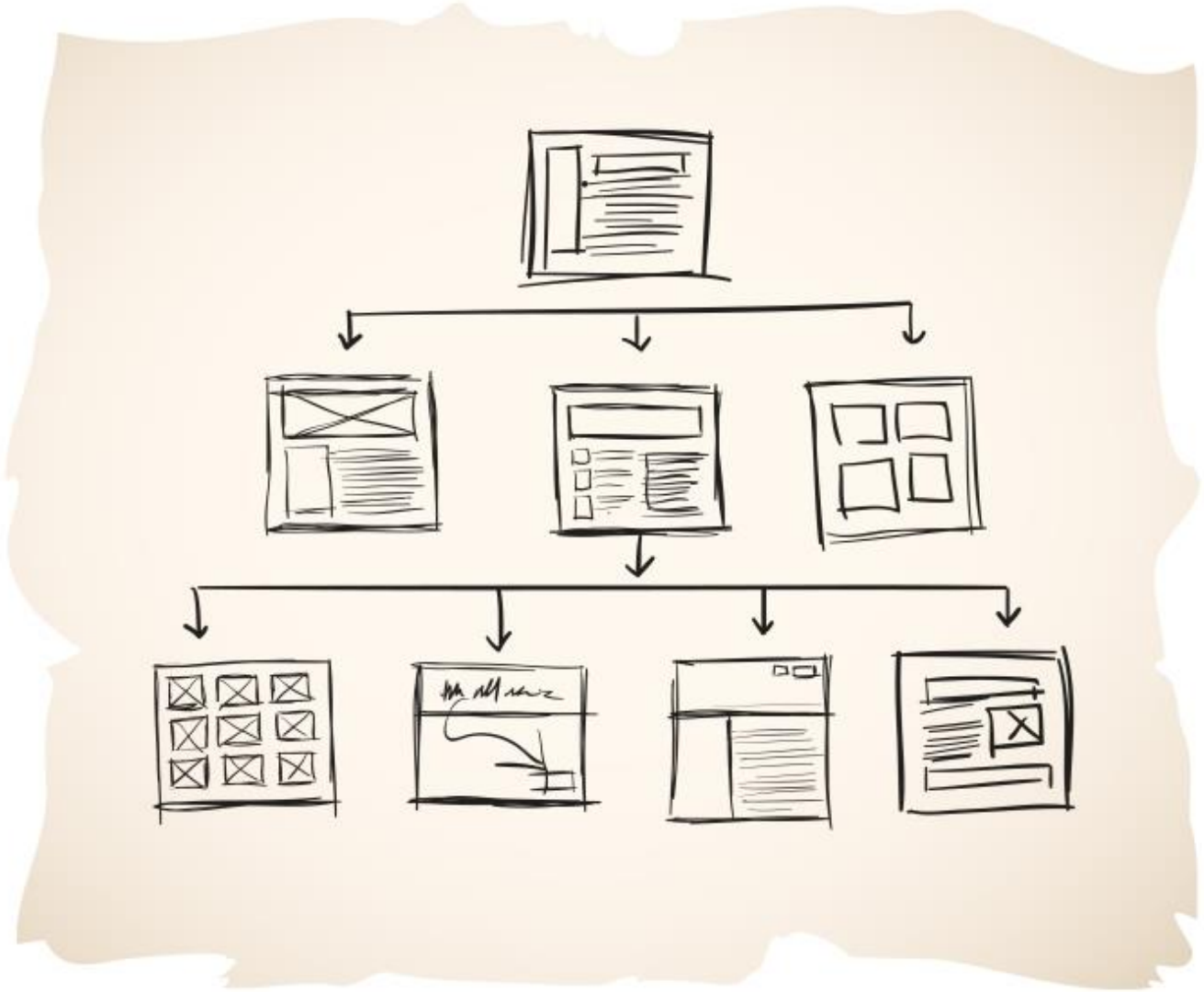


# All Other Methods

- For all other methods, we will introduce them when there is an application.

# Site

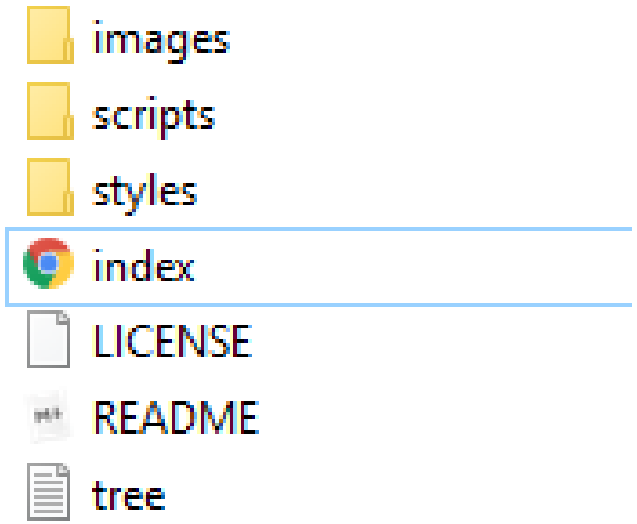
SECTION 4

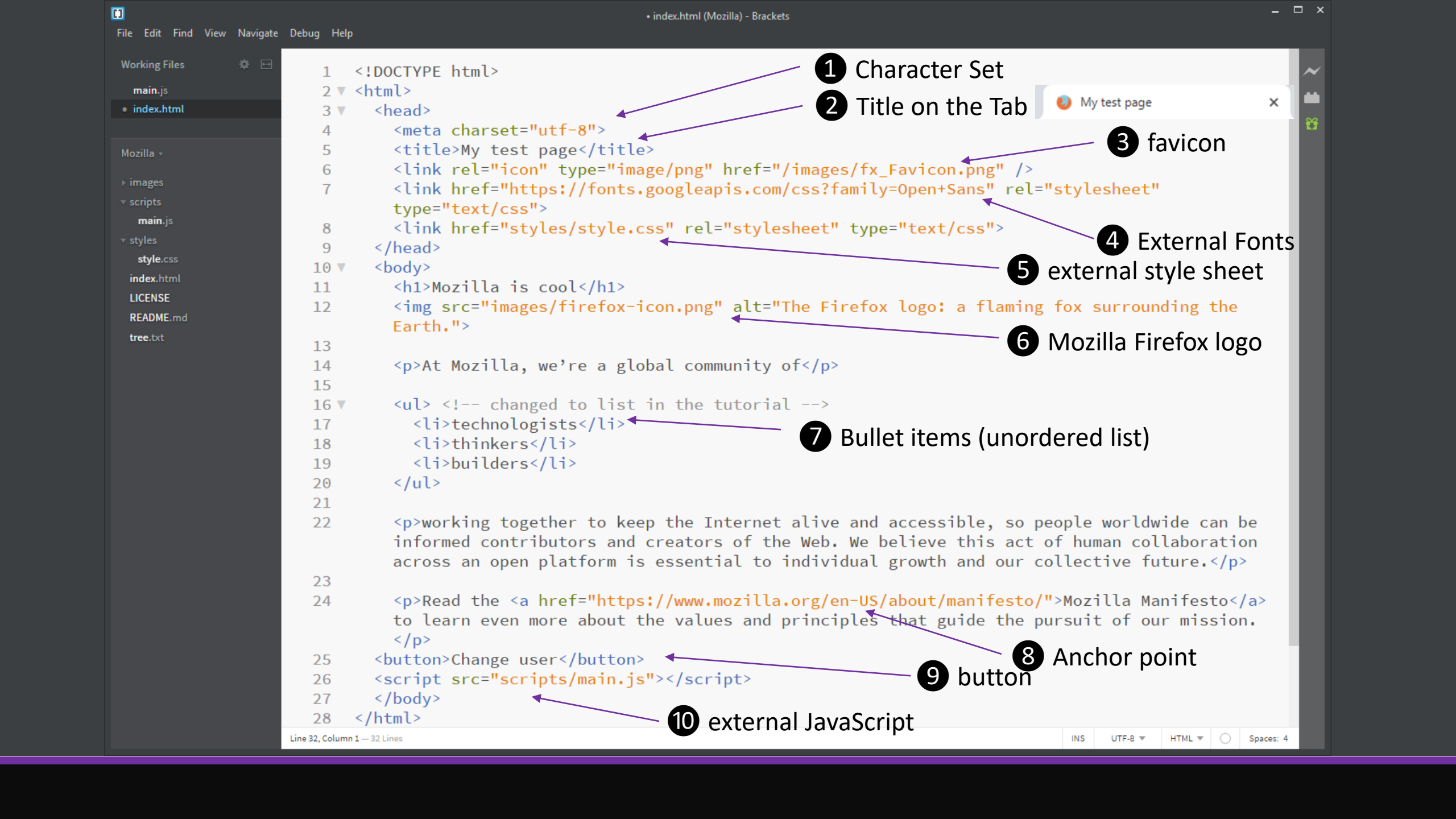


# Site Architecture Diagram

## c:>tree /f > tree.txt

```
C:\Eric_Chou\Web_Design\JsDev\JS5 HTML\Mozilla>type tree.txt
Folder PATH listing for volume Sugarcane C:
Volume serial number is 14C0-078B
C:.
    index.html
    LICENSE
    README.md
    tree.txt

---images
        appicns_Firefox.png
        firefox-icon.png
        firefox2.png
        fx.svg
        fx_Favicon.png

---scripts
        main.js

---styles
        style.css
```



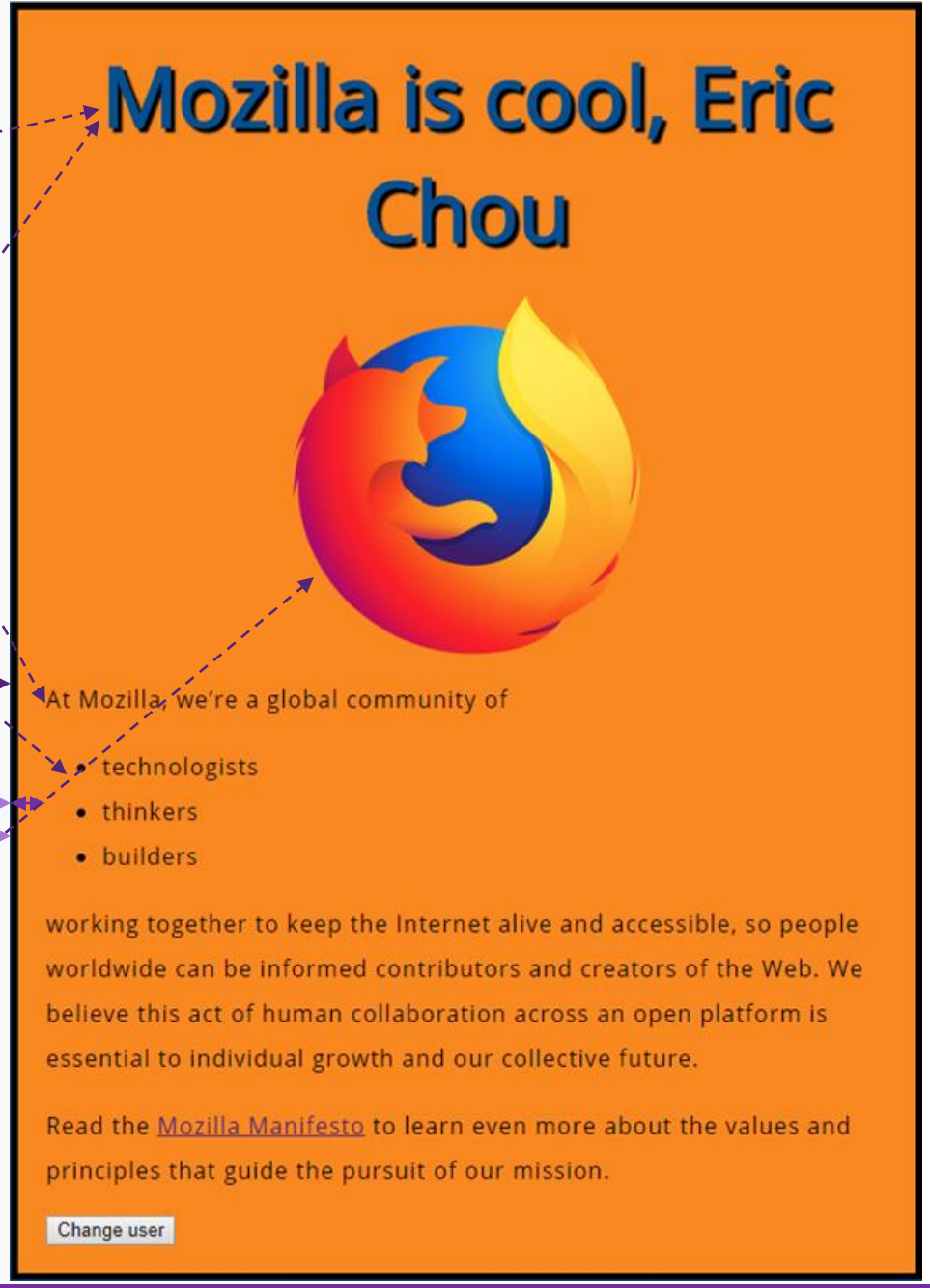

• index.html (Mozilla) - Brackets
File Edit Find View Navigate Debug Help
Working Files
main.js
index.html
Mozilla
images
scripts
main.js
styles
style.css
index.html
LICENSE
README.md
tree.txt
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>My test page</title>
6 <link rel="icon" type="image/png" href="/images/fx_Favicon.png" />
7 <link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet" type="text/css">
8 <link href="styles/style.css" rel="stylesheet" type="text/css">
9 </head>
10 <body>
11 <h1>Mozilla is cool</h1>
12 <img src="images/firefox-icon.png" alt="The Firefox logo: a flaming fox surrounding the Earth.">
13
14 At Mozilla, we're a global community of
15
16 <ul> <!-- changed to list in the tutorial -->
17 <li>technologists</li>
18 <li>thinkers</li>
19 <li>builders</li>
20 </ul>
21
22 working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.
23
24 Read the <a href="https://www.mozilla.org/en-US/about/manifesto/">Mozilla Manifesto</a> to learn even more about the values and principles that guide the pursuit of our mission.

25 <button>Change user</button>
26 <script src="scripts/main.js"></script>
27 </body>
28 </html>
Line 32, Column 1 — 32 Lines
INS
UTF-8
HTML
Spaces: 4
My test page
1 Character Set
2 Title on the Tab
3 favicon
4 External Fonts
5 external style sheet
6 Mozilla Firefox logo
7 Bullet items (unordered list)
8 Anchor point
9 button
10 external JavaScript

```
1  // Image switcher code
2
3  var myImage = document.querySelector('img');
4
5  myImage.onclick = function() {
6    var mySrc = myImage.getAttribute('src');
7    if(mySrc === 'images/firefox-icon.png') {
8      myImage.setAttribute ('src','images/firefox2.png');
9    } else {
10     myImage.setAttribute ('src','images/firefox-icon.png');
11   }
12 }
13
14 // Personalized welcome message code
15
16 var myButton = document.querySelector('button');
17 var myHeading = document.querySelector('h1');
18
19 function setUserName() {
20   var myName = prompt('Please enter your name.');
21   localStorage.setItem('name', myName);
22   myHeading.innerHTML = 'Mozilla is cool, ' + myName;
23 }
24
25 if(!localStorage.getItem('name')) {
26   setUserName();
27 } else {
28   var storedName = localStorage.getItem('name');
29   myHeading.innerHTML = 'Mozilla is cool, ' + storedName;
30 }
31
32 myButton.onclick = function() {
33   setUserName();
34 }
```

1. Use querySelector(‘img’) to get an image (by TagName).
2. Connect image and click handler.
3. get the src attribute (used to swap image)
4. Swap the image
5. Select the button node
6. setUserName() and the click hanlder for button
7. turn on the name flag (name was entered.)
8. change the content of the myHeading (h1 header)
9. first read of the name by calling setUserName()
10. Tie button the the handler setUserName()

```css
html {
  font-size: 10px;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  font-size: 60px;
  text-align: center;
}
p, li {
  font-size: 16px;
  line-height: 2;
  letter-spacing: 1px;
}
html {
  background-color: #00539F;
}
body {
  width: 600px;
  margin: 0 auto;
  background-color: #FF9500;
  padding: 0 20px 20px 20px;
  border: 5px solid black;
}
h1 {
  margin: 0;
  padding: 20px 0;
  color: #00539F;
  text-shadow: 3px 3px 1px black;
}
img {
  display: block;
  margin: 0 auto;
}
```



# Mozilla is cool, Eric Chou

At Mozilla, we're a global community of

- technologists
- thinkers
- builders

working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

Change user

# Mozilla is cool, Eric Chou

At Mozilla, we're a global community of

- technologists
- thinkers
- builders

working together to keep the Internet alive and accessible, so people worldwide can be informed contributors and creators of the Web. We believe this act of human collaboration across an open platform is essential to individual growth and our collective future.

Read the Mozilla Manifesto to learn even more about the values and principles that guide the pursuit of our mission.

Change user

# Homework

- Build a site with similar features as this Mozilla site from scratch.
- Practice all site design skills that you have learned so far.

# Upload

SECTION 5

# Upload the Your Site to Student's Directory

- Destination Directory: (YYYY is a 4-digit year code)

```
/public_html/Students/CSP_YYYY/FirstnameLastName/ProjectName
```

- Upload Tool (WinSCP or CoffeeCup)

# Overview of DOM and JavaScript

SECTION 6

# Objectives

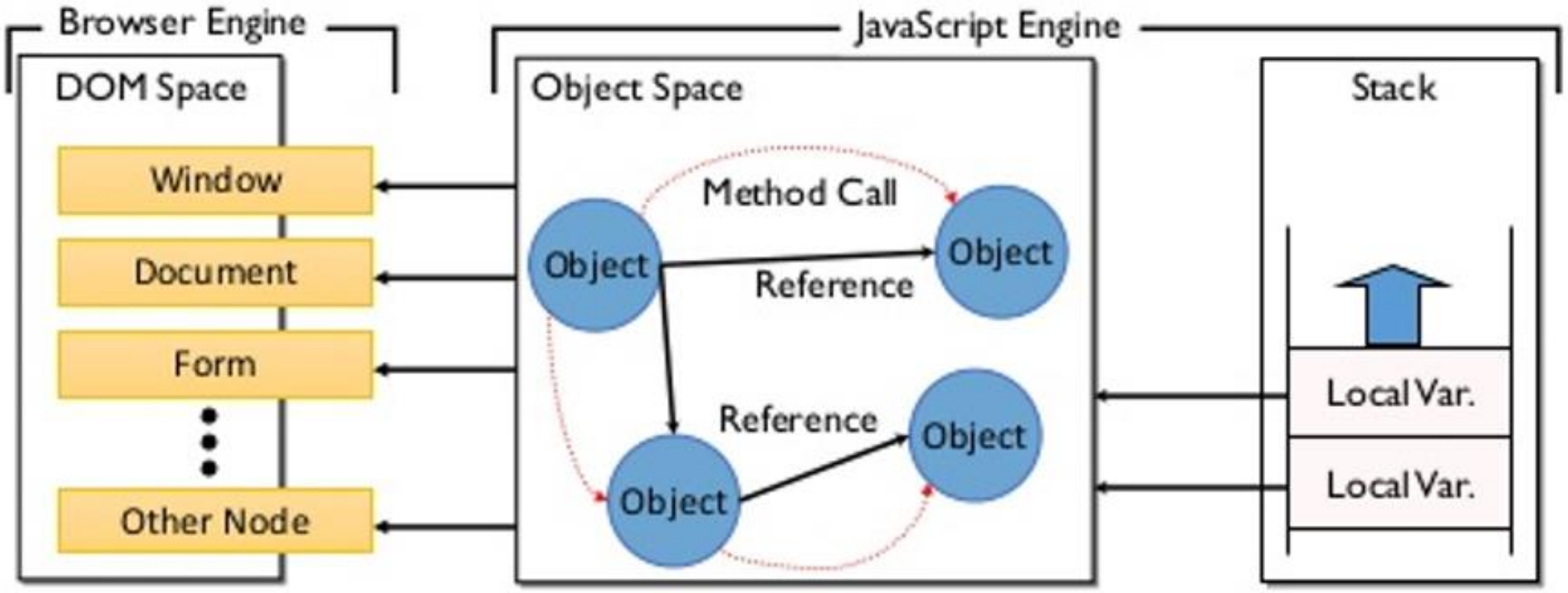
- So far, we've been using JavaScript to do relatively simple things like print text to the browser console or display an alert or prompt dialog.
- But you can also use JavaScript to manipulate (control or modify) and interact with the HTML you write in web pages. In this chapter, we'll discuss two tools that will allow you to write much more powerful JavaScript: the **DOM** and **jQuery**.

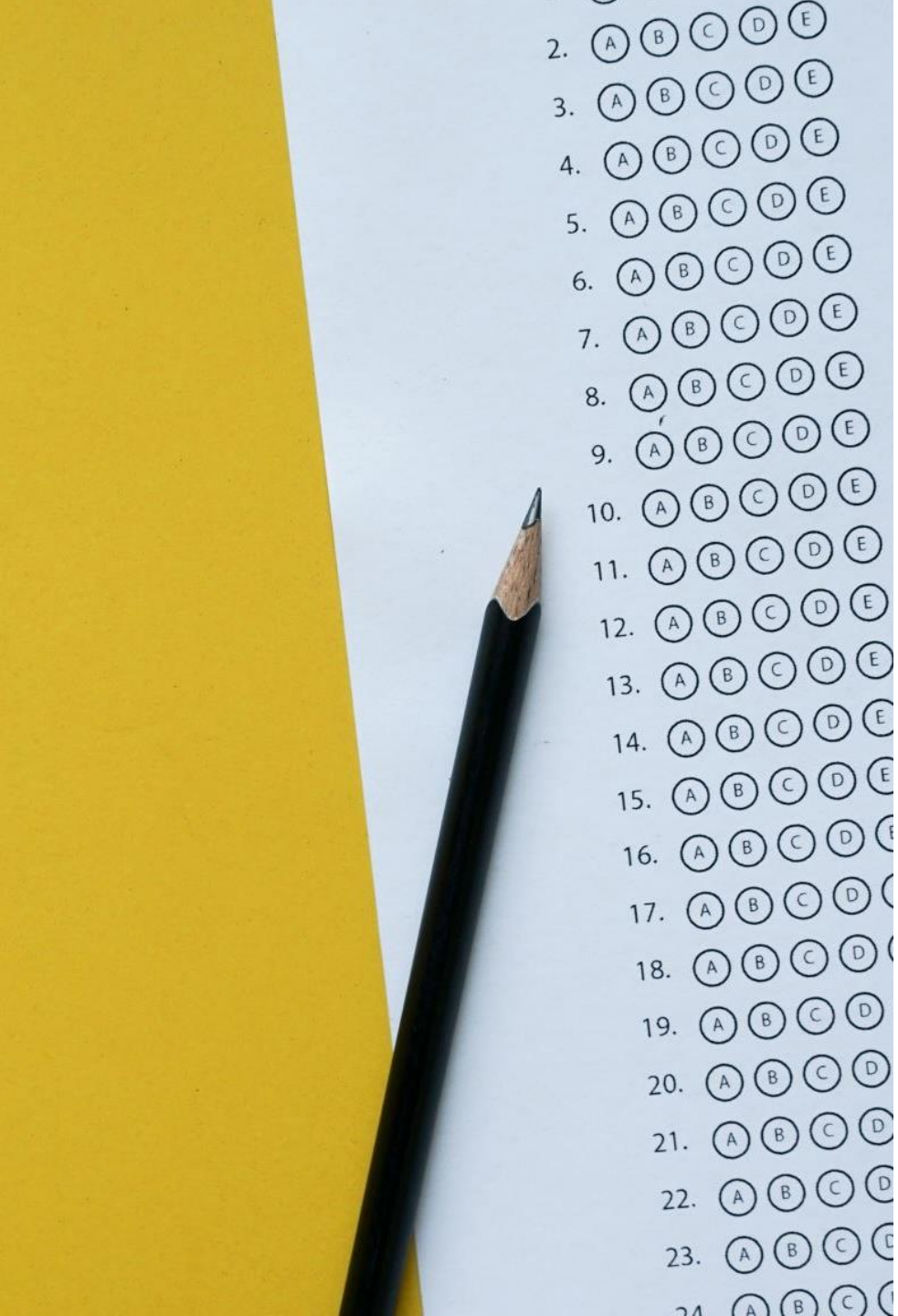
# DOM

- The **DOM**, or document object model, is what allows JavaScript to access the content of a web page. Web browsers use the **DOM** to keep track of the elements on a page (such as paragraphs, headings and other **HTML** elements), and JavaScript can manipulate **DOM** elements in various ways.
- For example, you'll soon see how you can use JavaScript to replace the main heading of the **HTML** document with input from a prompt dialog.

# JavaScript Memory Model

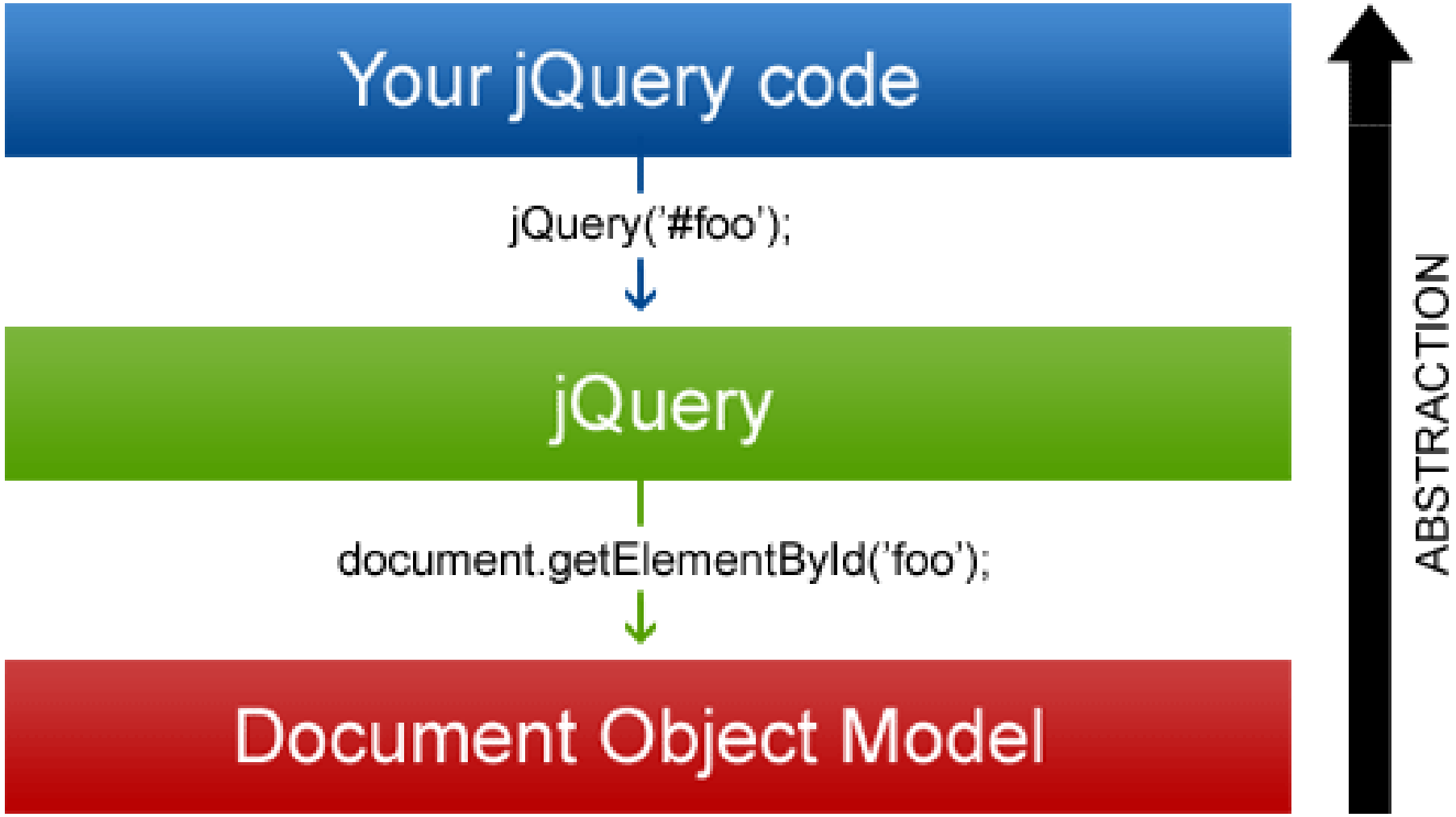
- DOM Space: the space where the Document Object Model representing the HTML's layered structure is represented.
- Object Space: the space where all JavaScript objects are located.
- Stack: short-term memory





# jQuery

- We'll also look at a useful tool called **jQuery**, which makes it much easier to work with the **DOM**.
- **jQuery** gives us a set of functions that we can use to choose which elements to work with and to make changes to those elements.
- In this chapter, we'll learn how to use the **DOM** and **jQuery** to edit existing **DOM** elements and create new **DOM** elements, giving us full control over the content of our web pages from JavaScript. We'll also learn how to use **jQuery** to animate **DOM** elements — for example, fading elements in and out.


Your jQuery code
jQuery('#foo');
jQuery
document.getElementById('foo');
Document Object Model
ABSTRACTION

# Selecting DOM Elements

SECTION 7

# Selecting DOM Elements

- When you load an **HTML** document into a browser, the browser converts the elements into a tree-like structure. This tree is known as the **DOM** tree.
- Figure 9-1 shows a simple **DOM** tree — the same tree we used in Chapter 5 to illustrate the hierarchy of **HTML**.
- The browser gives JavaScript programmers a way to access and modify this tree structure using a collection of methods called the **DOM**.

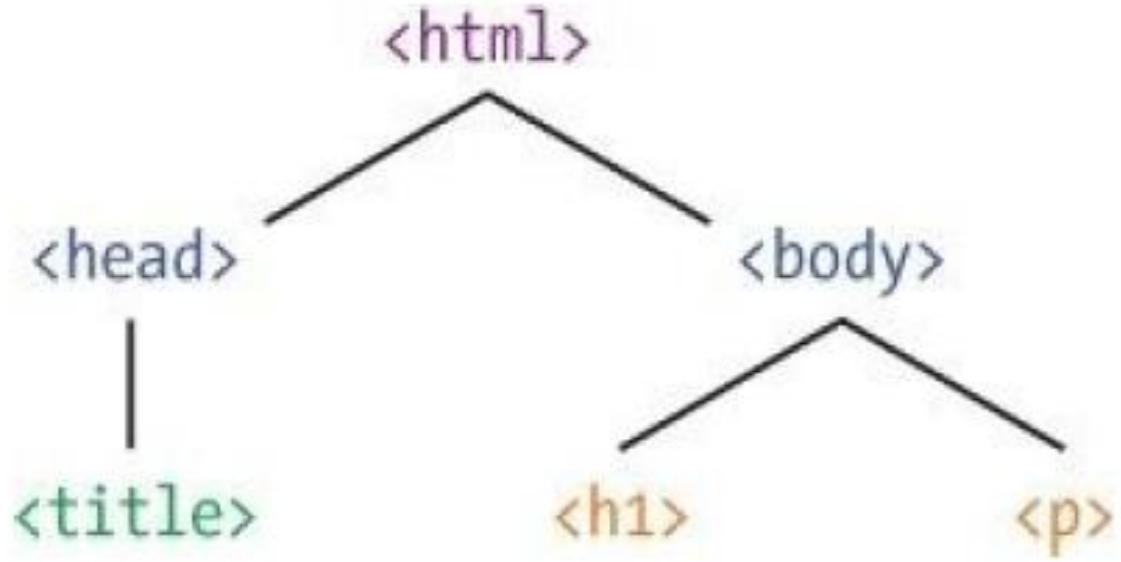



*Figure 9-1. The DOM tree for a simple HTML document, like the one we made in Chapter 5*

# Selecting DOM Elements

# Using id to Identify Elements

- The HTML id attribute lets you assign a unique name, or **identifier**, to an HTML element. For example, this h1 element has an id attribute:

```
<h1 id="main-heading">Hello world!</h1>
```

- In this example, the id of "main-heading" will let us identify, and eventually change, this particular heading without affecting other elements or even other h1 headings.

# Selecting an Element Using getElementById

- Having uniquely identified an element with id (each id must have a unique value), we can use the **DOM** method document.getElementById to return the "main-heading" element:

```
var headingElement =

document.getElementById("main-heading");
```

- By calling document.getElementById("main-heading"), we tell the browser to look for the element with the id of "main-heading". This call returns a DOM object that corresponds to the id, and we save this DOM object to the variable **headingElement**.

# Selecting an Element Using getElementById

- Once we've selected an element, we can manipulate it with JavaScript. For example, we can use the innerHTML property to retrieve and replace the text inside the selected element:

```
headingElement.innerHTML;
```

- This code returns the HTML contents of **headingElement** — the element we selected using **getElementById**. In this case, the content of this element is the text **Hello world!** that we entered between the <h1> tags.

# Replacing the Heading Text Using the DOM

Here's an example of how to replace heading text using the DOM. First, we create a new HTML document called *dom.html* containing this code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Playing with the DOM</title>
  </head>
  <body>
  <h1 id="main-heading">Hello world!</h1>
    <script>
❶     var headingElement = document.getElementById("main-heading");
❷     console.log(headingElement.innerHTML);
❸     var newHeadingText = prompt("Please provide a new heading:");
❹     headingElement.innerHTML = newHeadingText;
    </script>
  </body>
</html>
```

# Replacing the Heading Text Using the DOM

- At ❶ we use document.getElementById to get the h1 element (with the id of "main-heading") and save it into the variable headingElement. At ❷ we print the string returned by headingElement.innerHTML, which prints Hello world! to the console. At ❸ we use a prompt dialog to ask the user for a new heading and save the text the user enters in the variable newHeadingText.
- Finally, at ❹ we set the innerHTML property of headingElement to the text saved in newHeadingText.
- When you load this page, you should see a prompt dialog like the one shown in Figure 9-2.

# Demonstration Program

HEADING.HTML

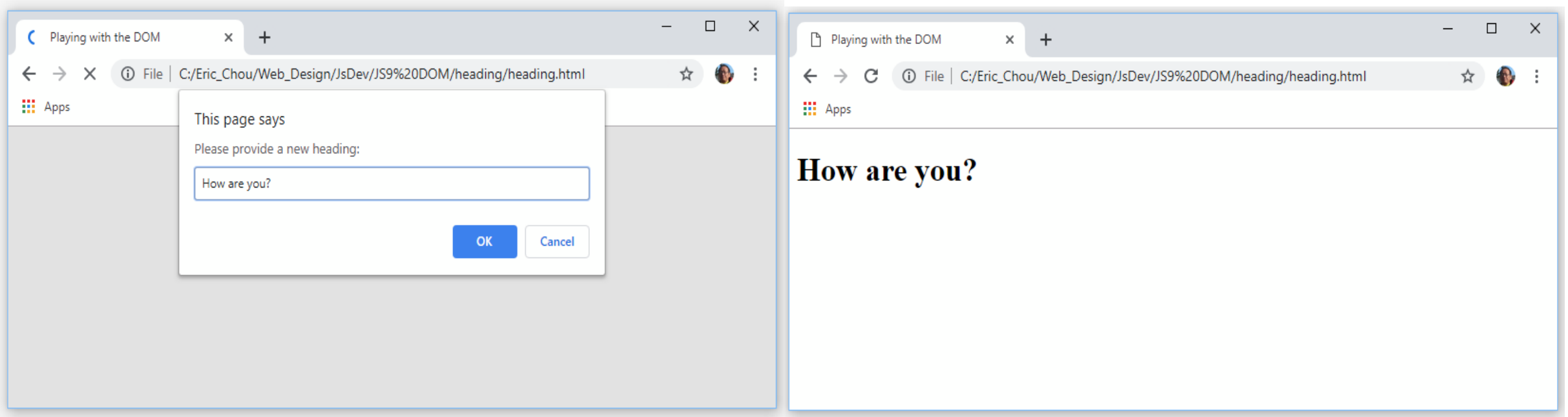



Figure 9-2. Our page with the dialog open

# Replacing the Heading Text Using the DOM

- Enter the text **JAVASCRIPT IS AWESOME** into the dialog and click **OK**. The heading should update instantly with the new text, as shown in Figure 9-3.


- *Figure 9-3. Our page after the heading change*

- Using the innerHTML property, we can change the content of any DOM element using JavaScript.

# Work with the DOM Tree

SECTION 8

# Using jQuery to Work with the DOM Tree

- The built-in **DOM** methods are great, but they're not very easy to use. Because of this, many developers use a set of tools called **jQuery** to access and manipulate the **DOM** tree. **jQuery** is a JavaScript library
  — a collection of related tools (mostly functions) that gives us, in this case, a simpler way to work with
- **DOM** elements. Once we load a library onto our page, we can use its functions and methods in addition to those built into JavaScript and those provided by the browser.

# Loading jQuery on Your HTML Page

- To use the jQuery library, we first tell the browser to load it with this line of HTML:

```
<script
src="https://code.jquery.com/jquery-
2.1.0.js">
</script>
```

- Notice that the <script> tag here has no contents, and it has a src attribute. The src attribute lets us insert a JavaScript file into our page by including its *URL* (web address). In this case, *https://code.jquery.com/jquery-2.1.0.js* is the URL for a specific version of jQuery (version 2.1.0) on the jQuery website.

- To see the jQuery library, visit that URL; you'll see the JavaScript that will be loaded when this <script> tag is added. The entire library is over 9,000 lines of complicated JavaScript, though, so don't expect to understand it all right now!

# Replacing the Heading Text Using jQuery

- In Replacing the Heading Text Using the DOM, you learned how to replace text using the built-in **DOM** methods. In this section, we'll update that code to use **jQuery** to replace the heading text instead. Open **dom.html** and make the changes shown.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Playing with the DOM</title>
    </head>
    <body>
        <h1 id="main-heading">Hello world!</h1>
❶       <script src="https://code.jquery.com/jquery-2.1.0.js"></script>
        <script>
            var newHeadingText = prompt("Please provide a new heading:");
❷           $("#main-heading").text(newHeadingText);
        </script>
    </body>
</html>
```

# Demonstration Program

HEADING2.HTML

# Replacing the Heading Text Using jQuery

- At ❶ we add a new **<script>** tag to the page to load **jQuery**. With **jQuery** loaded, we use the **jQuery** function **$** to select an **HTML** element.
- The **$** function takes one argument, called a *selector string*, which tells **jQuery** which element or elements to select from the **DOM** tree. In this case, we entered **"#main-heading"** as the argument. The **#** character in a selector string means **“ID,”** so our selector string **"#main-heading"** means **“the element with an id of main-heading.”**

# Replacing the Heading Text Using jQuery

- The $ function returns a **jQuery** object that represents the elements you selected. For example,

  `$("#main-heading")` returns a jQuery object for the **h1** element (which has an **id** of "mainheading").

- We now have a **jQuery** object representing the h1 element. We can modify its text by calling the text method on the **jQuery** object at ❷, passing in the new text for that element, and replacing the text of the heading with the user input saved to the variable **newHeadingText**.

- As before, when you load this page, a dialog should prompt you to enter replacement text for the old text in the **h1** element.

# New Elements

SECTION 9

# Creating New Elements with jQuery

- In addition to manipulating elements with **jQuery**, we can also use jQuery to create new elements and add them to the **DOM** tree. To do so, we call append on a jQuery object with a string containing **HTML**.
- The append method converts the string to a **DOM** element (using the **HTML** tags in the string) and adds the new element to the end of the original one.
- For example, to add a p element to the end of the page, we could add this to our **JavaScript**:

```
$("body").append("<p>This is a new paragraph</p>");
```

# Creating New Elements with jQuery

- The first part of this statement uses the **$** function with the selector string "body" to select the body of our **HTML** document. The selector string doesn't have to be an id. The code **$("body")** selects the body element. Likewise, we could use the code **$("p")** to select all the p elements.
- Next, we call the append method on the object returned by **$("body")**. The string passed to append is turned into a **DOM** element, and it is added inside the body element, just before the closing tag. Figure 9-4 shows what our revised page would look like.


*Figure 9-4. Our document with a new element*

# Creating New Elements with jQuery

- We could also use append to add multiple elements in a for loop like this:

```
for (var i = 0; i < 3; i++) {
 var hobby = prompt("Tell me one of your hobbies!");
 $("body").append("<p>" + hobby + "</p>");
}
```

- This loops three times. Each time through a loop, a prompt appears, asking users to enter one of their hobbies. Each hobby is then put inside a set of **<p>** tags and passed to the append method, which adds the hobby to the end of the body element. Try adding this code to your **dom.html** document, and then load it in a browser to test it. It should look like Figure 9-5.

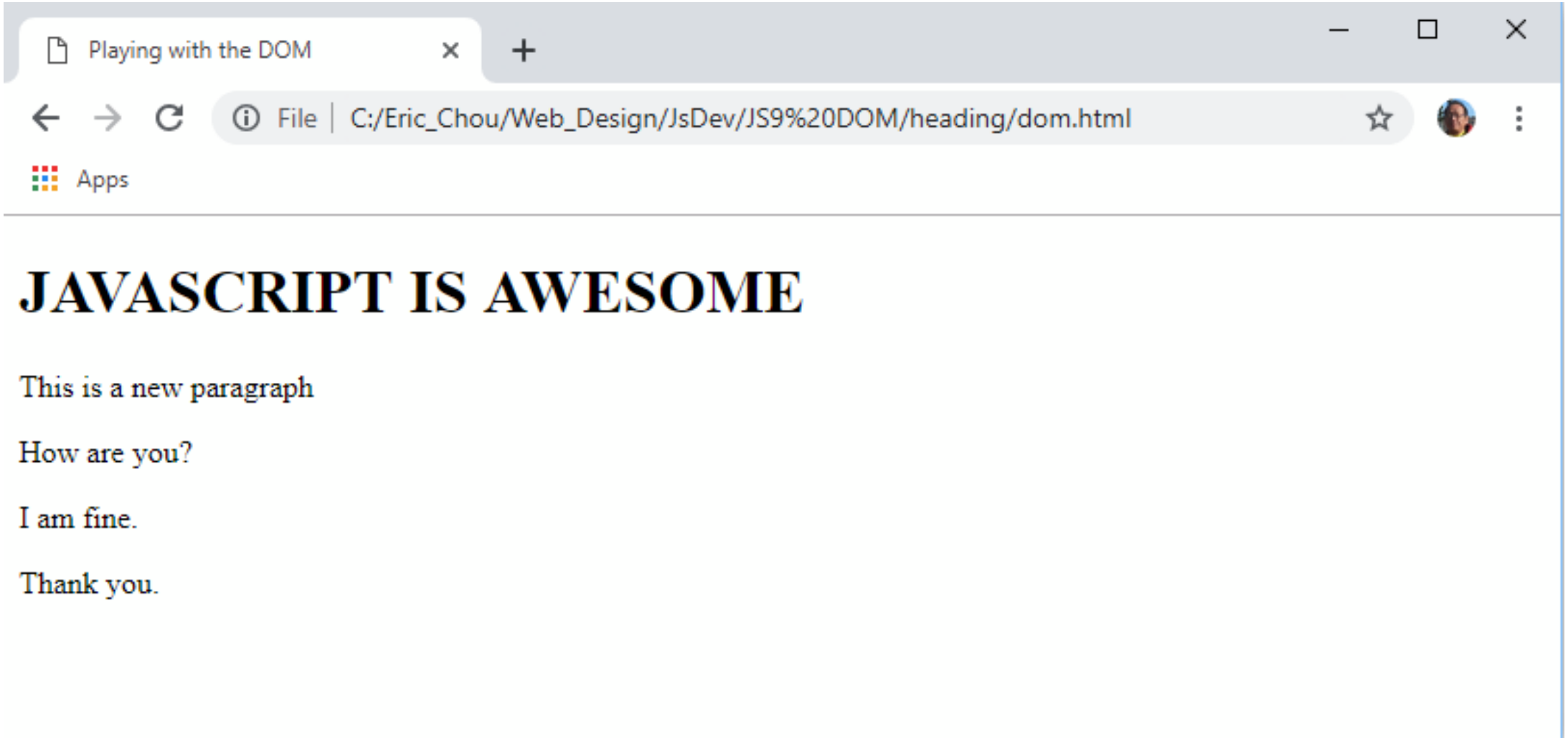



Figure 9-5. Extra elements added in a loop

# Animating Elements

SECTION 10

# Animating Elements with jQuery

- Lots of websites use animations to show and hide content. For example, if you were adding a new paragraph of text to your page, you might want to fade it in slowly so it doesn't appear all of a sudden.
- **jQuery** makes it easy to animate elements. For example, to fade an element out, we can use the fadeout method. To test this method, replace the contents of the second script element in **dom.html** with this:

```
$("h1").fadeOut(3000);
```

# Animating Elements with jQuery

- We use the **$** function to select all h1 elements. Because **dom2.html** has only one **h1** element (the heading containing the text Hello world!), that heading is selected as a **jQuery** object. By calling **.fadeOut(3000)** on this **jQuery** object, we make the heading fade away until it disappears, over the course of 3 seconds. (The argument to **fadeOut** is in milliseconds, or thousandths of a second, so entering 3000 makes the animation last 3 seconds.)
- As soon as you load the page with this code, the **h1** element should start to fade away.

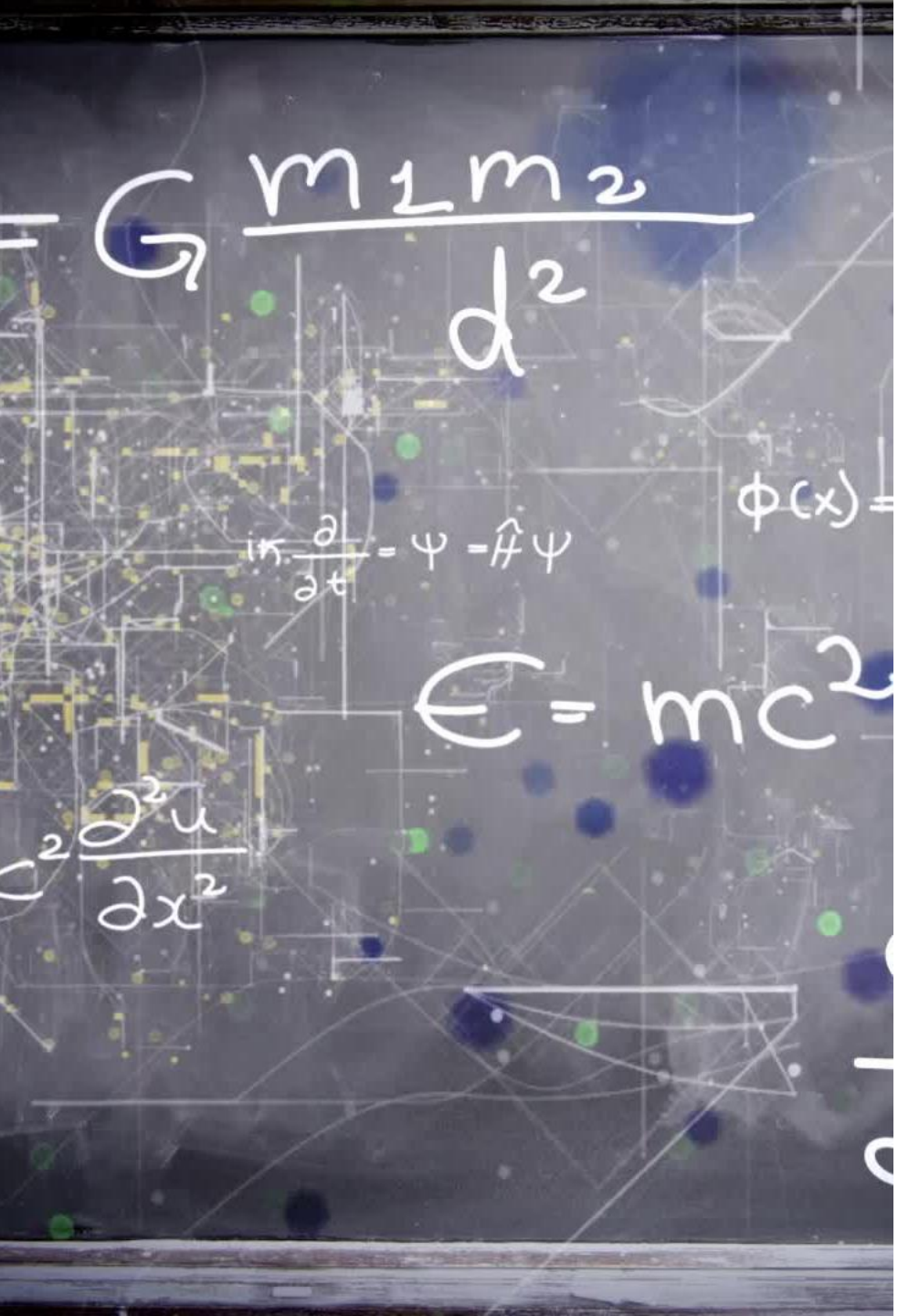
# Demonstration Program

DOM2.HTML

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Playing with the DOM</title>
5  </head>
6  <body>
7      <h1 id="main-heading">Hello world!</h1>
8      <script src="https://code.jquery.com/jquery-3.4.0.js"></script>
9      <script>
10       $("h1").fadeOut(3000);
11     </script>
12   </body>
13 </html>
```

# Chaining jQuery Animations

SECTION 11




# Chaining jQuery Animations

- When you call a method on a **jQuery** object, the method usually returns the original object that it was called on. For example, $("h1") returns a **jQuery** object representing all **h1** elements, and
- $("h1").fadeOut(3000) returns the *same* **jQuery** object representing all **h1** elements. To change the text of the h1 element and fade it out, you could enter:

```
$("h1").text("This will fade out").fadeOut(3000);
```

- Calling multiple methods in a row like this is known as ***chaining***.

# Chaining jQuery Animations

- We can chain multiple animations on the same element. For example, here's how we could chain a call to the **fadeOut** and **fadeIn** methods to fade an element out and then immediately fade it in again:

```
$("h1").fadeOut(3000).fadeIn(2000);
```

- The **fadeIn** animation makes an invisible element fade back in. **jQuery** is smart enough to know that when you chain two animations in a row like this, you probably want them to happen one after the other. Therefore, this code fades the **h1** element out over the course of 3 seconds and then fades it back in over 2 seconds.

# Chaining jQuery Animations

- **jQuery** provides two additional animation methods similar to fadeOut and fadeIn, called **slideUp** and **slideDown**. The slideUp method makes elements dis appear by sliding them up, and **slideDown** makes them reappear by sliding them down. Replace the second script element in the **dom.html** document with the following, and reload the page to try it out:

```
$("h1").slideUp(1000).slideDown(1000);
```

- Here we select the **h1** element, slide it up over 1 second, and then slide it down over 1 second until it reappears.

# Activity

- We use **fadeIn** to make invisible elements visible. But what happens if you call **fadeIn** on an element that's already visible or an element that comes *after* the element you're animating?
- For example, say you add a new p element to your **dom.html** document after the heading. Try using **slideUp** and **slideDown** to hide and show the h1 element, and see what happens to the p element. What if you use **fadeOut** and **fadeIn**?
- What happens if you call **fadeOut** and **fadeIn** on the same element without chaining the calls? For example:

```
$("h1").fadeOut(1000);
$("h1").fadeIn(1000);
```

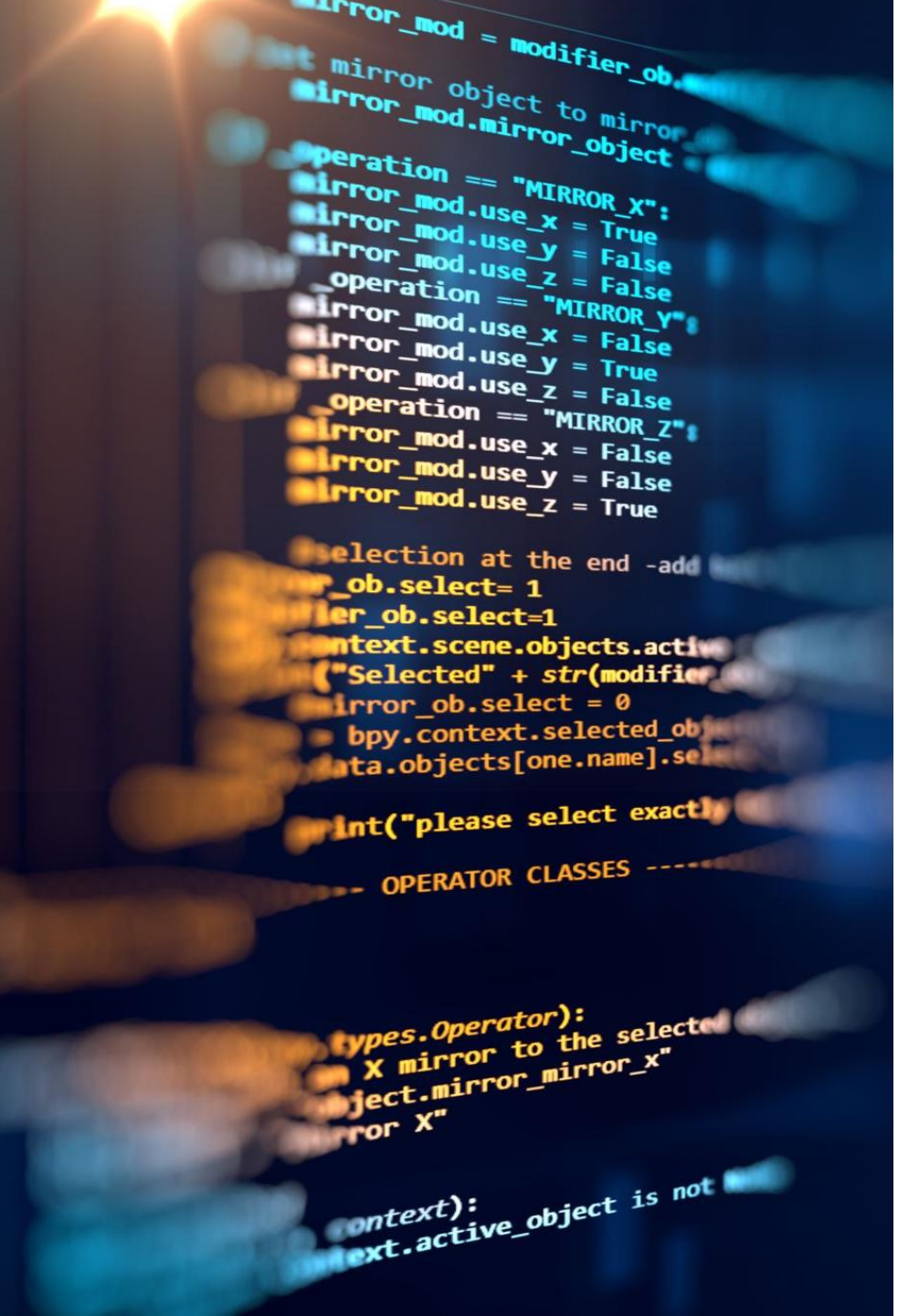
- Try adding the preceding code inside a for loop set to run five times. What happens?
- What do you think the show and hide **jQuery** methods do? Try them out to see if you're right. How could you use hide to fade in an element that's already visible?

# Demonstration Program

DOM3.HTML / DOM4.HTML

# Summary of DOM and JavaScript

SECTION 12



# Summary

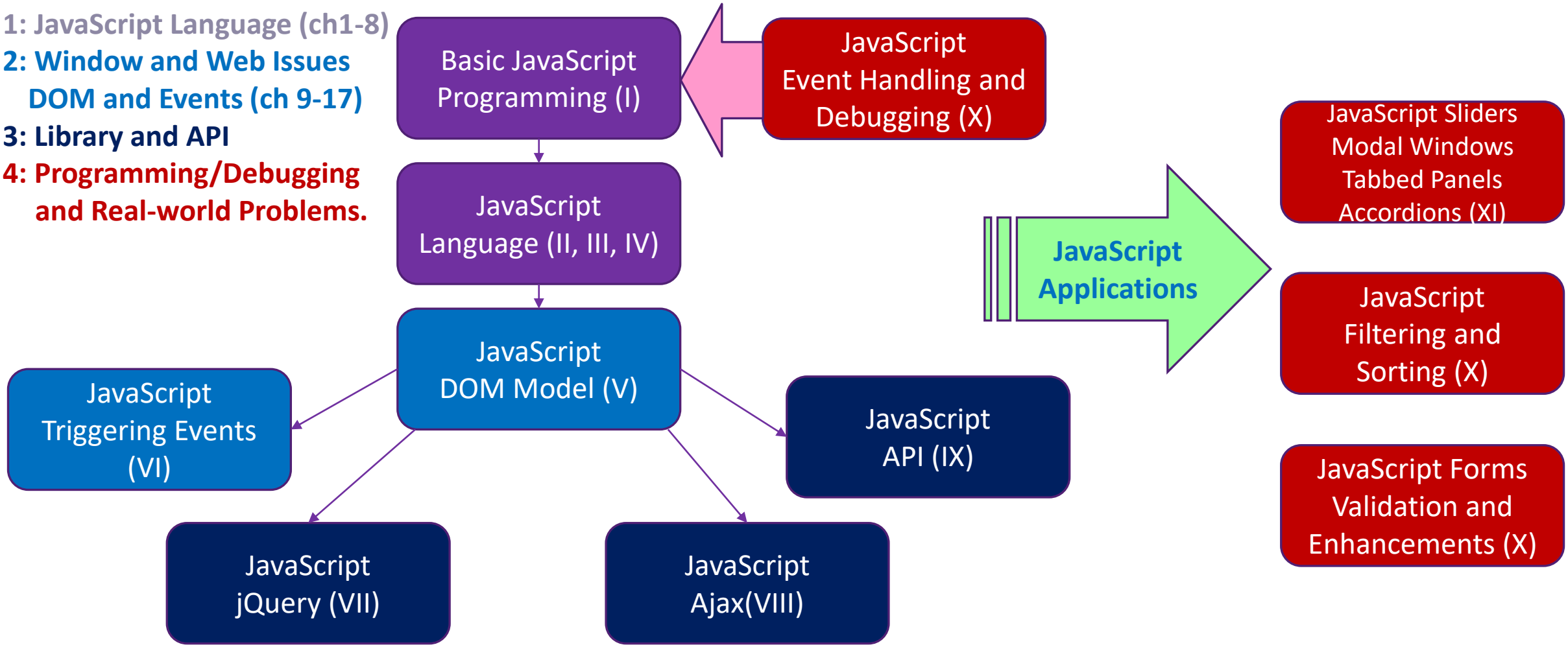
- In this chapter, you learned how to update **HTML** pages using JavaScript by manipulating **DOM** elements. As you've seen, **jQuery** gives us even more powerful ways to select elements and change or even animate them. You also learned a new **HTML** attribute, id, which allows you to give an element a unique identifier.
- In the next chapter, you'll learn how to control when your JavaScript is run — for example, once a timer has run out or when you click a button. We'll also look at how to run the same piece of code multiple times with a time delay in between — for example, updating a clock once every second.

# jQuery on DOM [Optional]

SECTION 13

# JavaScript and jQuery Course Work

Part 1: JavaScript Language (ch1-8)
Part 2: Window and Web Issues
DOM and Events (ch 9-17)
Part 3: Library and API
Part 4: Programming/Debugging
and Real-world Problems.



# jQuery Version

**Download Site:**

https://jquery.com/download/

**Versions:**

- Regular version.
- Compressed version.
- Online version

# Objectives

**Applied**

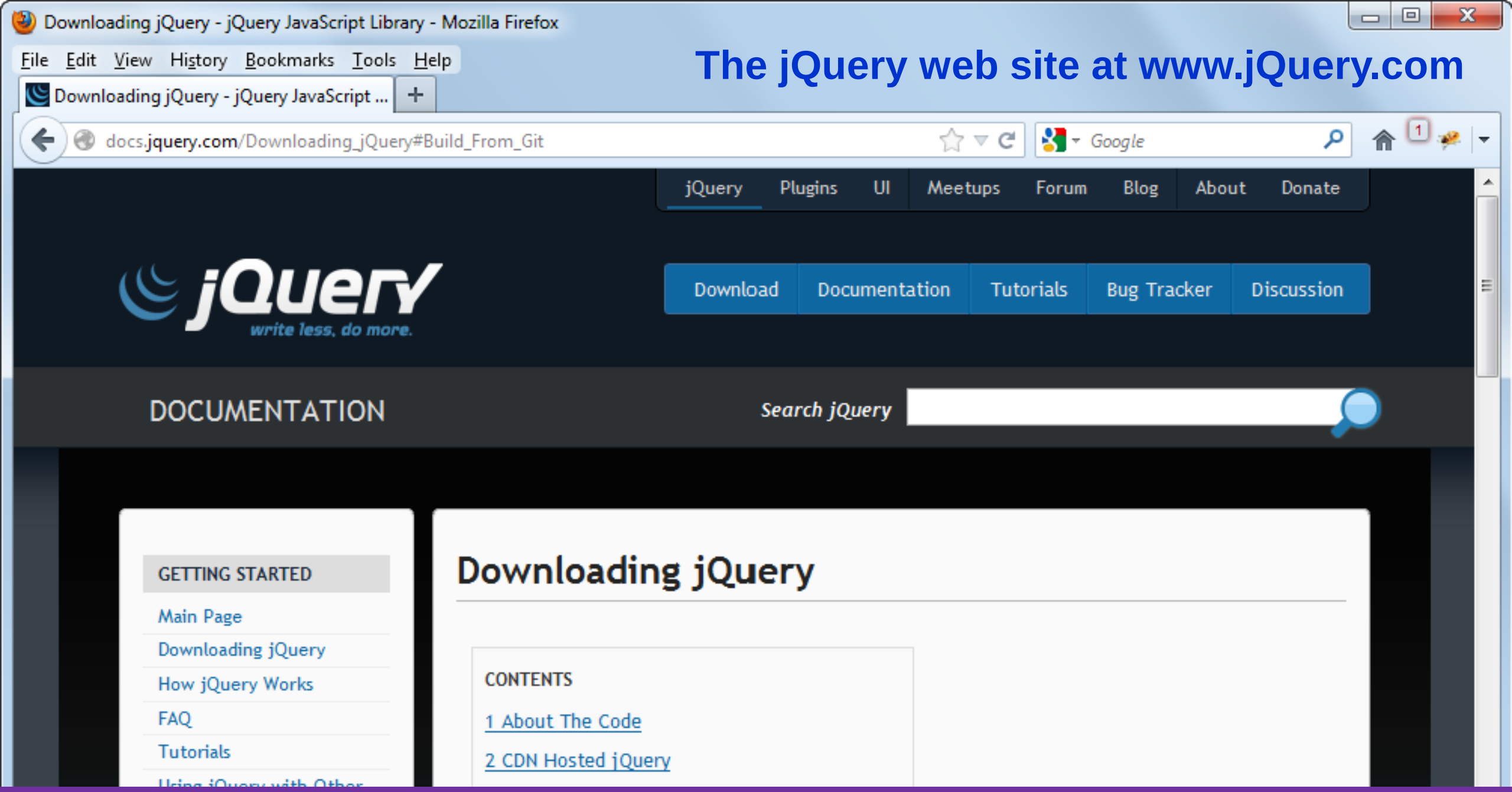
Use jQuery to develop common DOM scripting applications like the Email List, FAQs, Image Swap, and Image Rollover applications that are presented in this chapter.

**Knowledge**

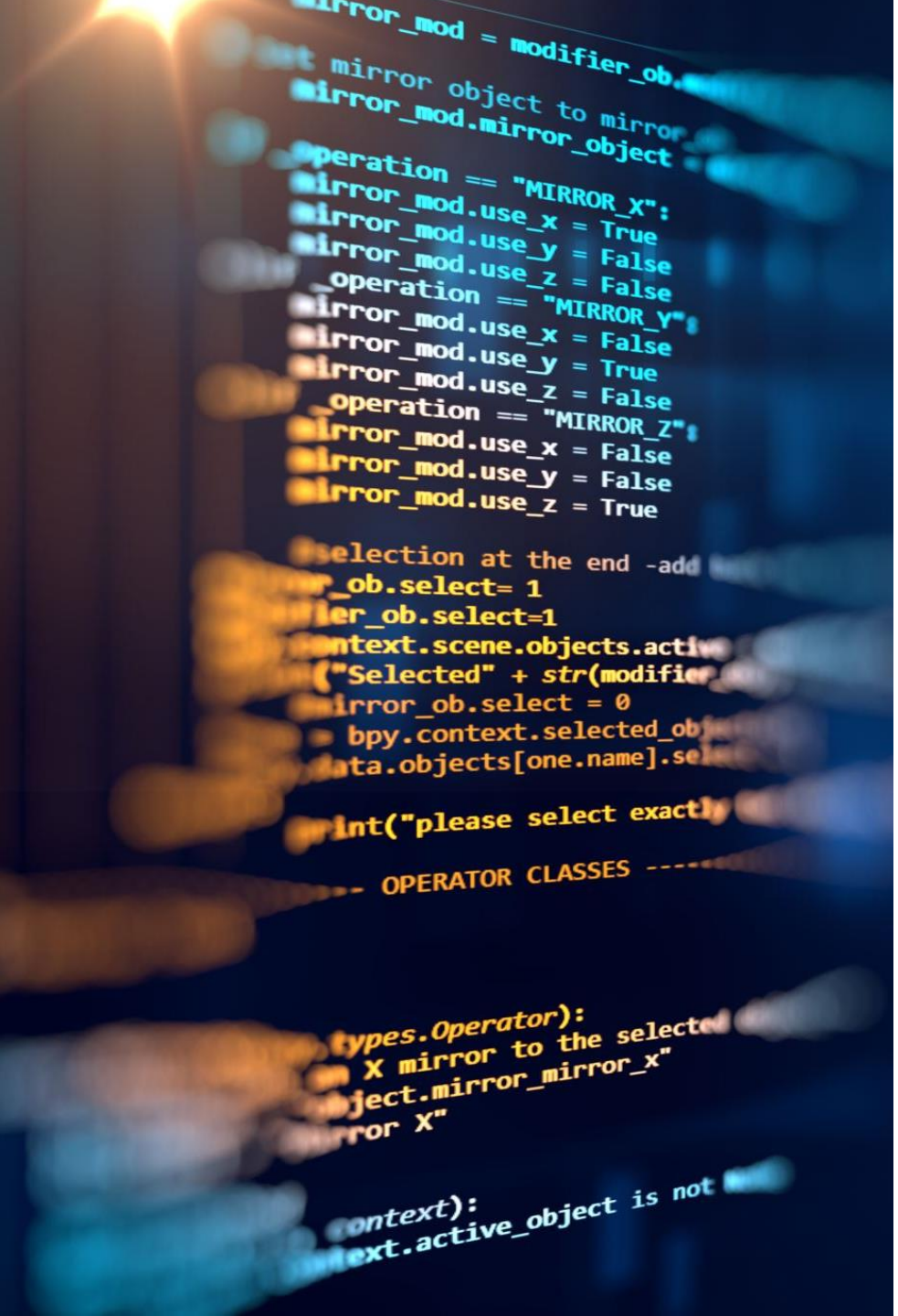
1. Describe jQuery
2. Describe two ways to include the jQuery library in your web pages.
3. In general terms, describe the use of jQuery selectors, methods, and event methods
4. Describe the syntax for a jQuery selector.
5. Describe the use these methods: val, next, prev, text, attr, css, addClass, removeClass, toggleClass, hide, show and each.
6. Describe object chaining.
7. Describe the use of these jQuery event methods: ready, click, toggle, mouseover, and hover.
8. Describe the use of this keyword withing a function for an event method.

# Introduction to jQuery [Optional]

SECTION 14




# The jQuery web site at www.jQuery.com



# What jQuery offers

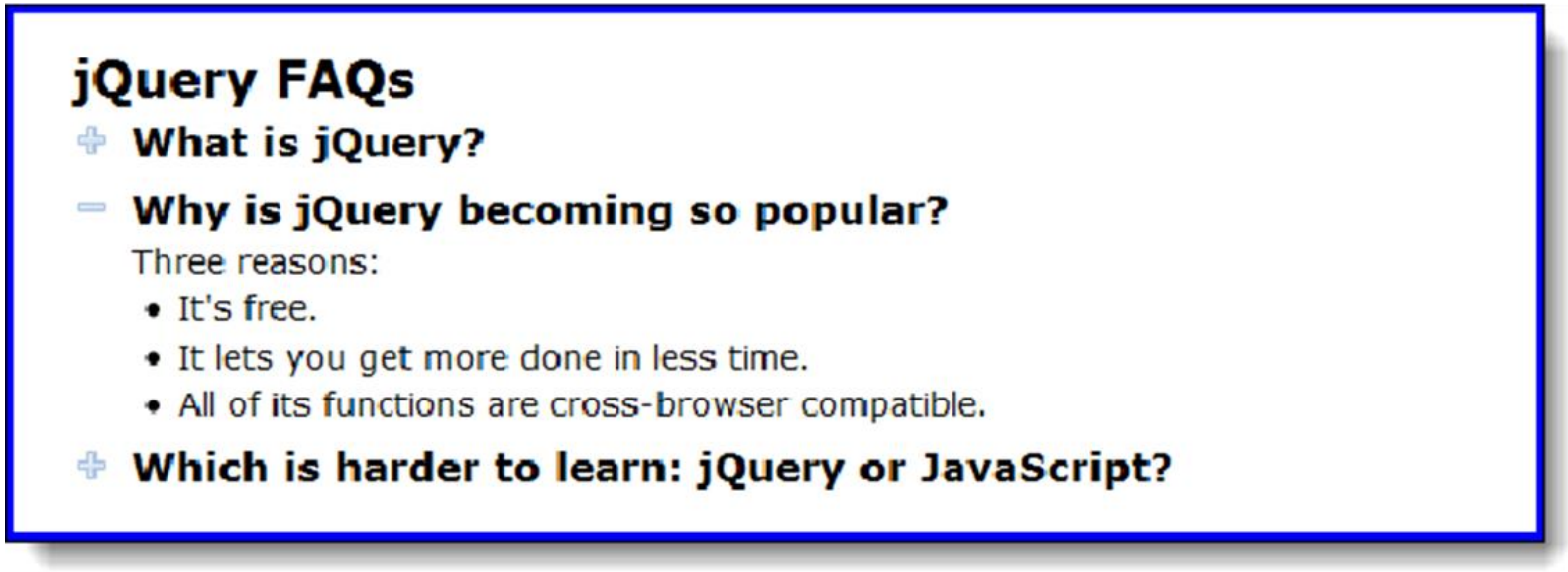
- Dozens of methods that make it easier to add JavaScript features to your web pages
- Methods that are tested for cross-browser compatibility
- How to include the jQuery file from your computer?

```
<script src="jquery-1.8.2.min.js"></script>
```

- How to include the jQuery file from a Content Delivery Network (CDN)?

```
<script src="http://code.jquery.com/jquery-latest.min.js"></script>
```

# The user interface for the FAQs application



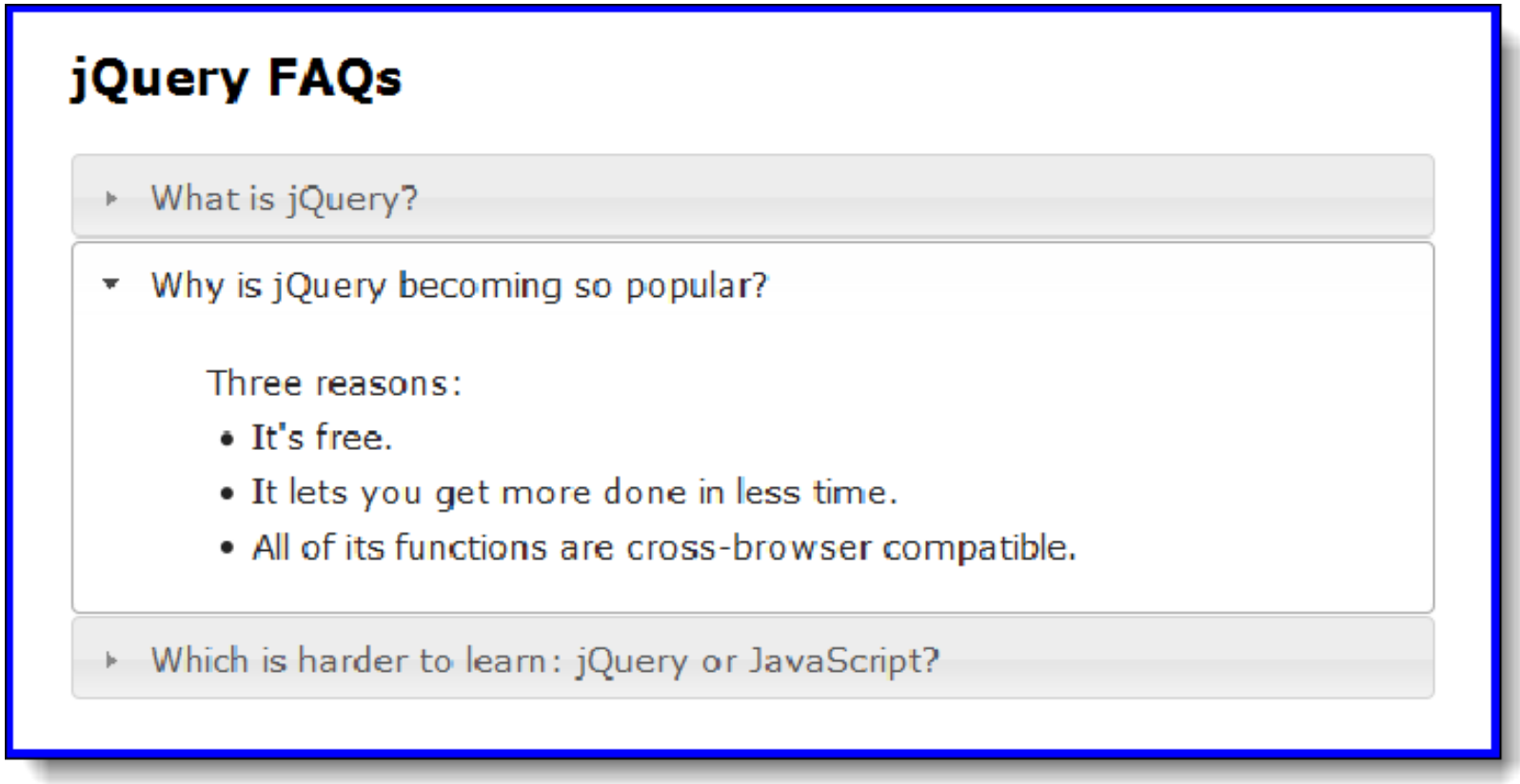
```
var $ = function (id) {
    return document.getElementById(id);
}
window.onload = function () {
    var faqs = $("faqs");
    var h2Elements = faqs.getElementsByTagName("h2");
    var h2Node;
    for (var i = 0; i < h2Elements.length; i++ ) {
        h2Node = h2Elements[i];
        // Attach event handler
        h2Node.onclick = function () {
            var h2 = this;    // h2 is the current h2Node object
            if (h2.getAttribute("class") == "plus") {
                h2.setAttribute("class", "minus");
            }
            else {
                h2.setAttribute("class", "plus");
            }
            if (h2.nextElementSibling.getAttribute("class")
                    == "closed") {
                h2.nextElementSibling.setAttribute("class",
                    "open");
            }
            else {
                h2.nextElementSibling.setAttribute("class",
                    "closed");
            }
        }
    }
}
```

# The JavaScript for the FAQs application

```
$(document).ready(function() {
    $("#faqs h2").toggle(
        function() {
            $(this).addClass("minus");
            $(this).next().show();
        },
        function() {
            $(this).removeClass("minus");
            $(this).next().hide();
        }
    );  // end toggle
}); // end ready
```

# The jQuery for the FAQs application

# The FAQs application as a jQuery UI accordion

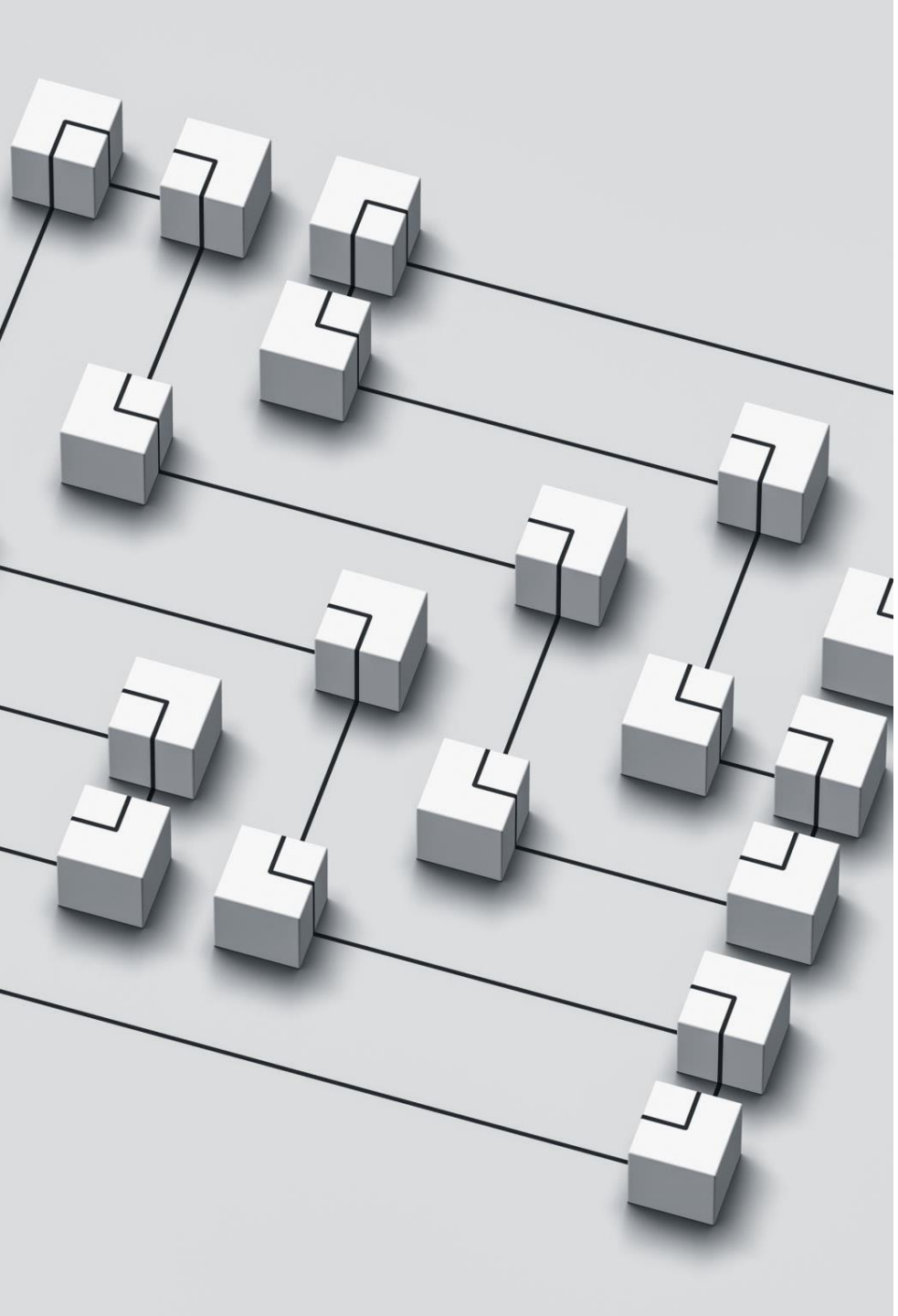


```
<div id="accordion">
  <h3><a href="#">What is jQuery?</a></h3>
  <div> <!-- panel contents --> </div>
  <h3><a href="#">Why is jQuery becoming so popular?</a>
  </h3>
  <div> <!-- panel contents --> </div>
  <h3><a href="#">Which is harder to learn: jQuery or
      JavaScript?></a></h3>
  <div> <!-- panel contents --> </div>
</div>
```

# The HTML for a jQuery UI accordion

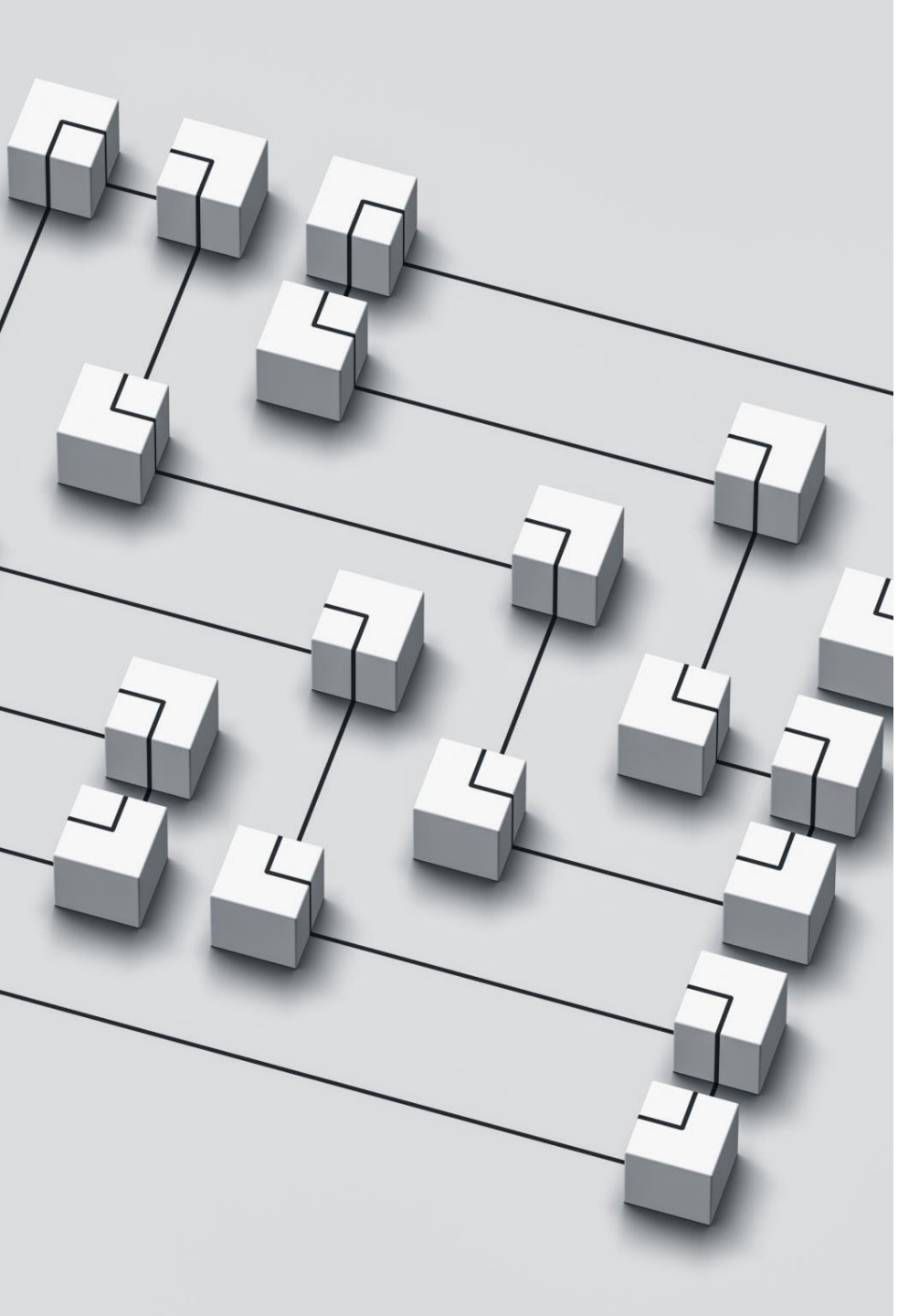
```
<script>
      $(document).ready(function()  {
            $("#accordion").accordion();
      });
</script>
```

The JavaScript code for the jQuery UI accordion



# Some typical plugin functions

- Data Validation
- Slide shows
- Carousels



# Terms

- jQuery
- Content Delivery System (CDN)
- jQueryUI
- plugin

# The basics of jQuery programming [Optional]

SECTION 15

# The syntax for a jQuery selector

$(“selector”)

```
<section id="faqs">
    <h1>jQuery FAQs</h1>
    <h2 class="plus">What is jQuery?</h2>
    <div>
        <p>jQuery is a library of the JavaScript
        functions that you're most likely to need as you
        develop web sites.</p>
    </div>
    <h2 class="plus">Why is jQuery becoming so popular?
    </h2>
    <div>
        <p>Three reasons:</p>
        <ul>
            <li>It's free.</li>
            <li>It lets you get more done in less
            time.</li>
            <li>All of its functions cross-browser
            compatible.</li>
        </ul>
    </div>
</section>
```

# The HTML for the selected elements

# How to select elements by element, id, and class

**By element type: all <p> elements in the entire document**

```
$("p")
```

**By id: The element with "faqs" as its id**

```
$("#faqs")
```

**By Class: All elements with "plus" as a class**

```
$(".plus")
```

# How to select elements by relationship

**Descendants: All <p> elements that are descendants of the section element**

```
$("#faqs p");
```

**Adjacent siblings: All div elements that are adjacent siblings of h2 elements**

```
$("h2 + div")
```

**General siblings: All <p> elements that are siblings of ul elements**

```
$("ul ~ p")
```

**Children: All ul elements that are children of div elements**

```
$("div > ul")
```

# How to code multiple selectors

```
$("#faqs li, div p")
$("p + ul, div ~ p")
```

# The syntax for calling a jQuery method

## The syntax for calling a jQuery method

```
$("selector").methodName(parameters)
```

## Some common jQuery methods

```
val()
val(value)
text()
text(value)
next([type])
submit()
focus()
```

# Examples that call jQuery methods

**How to get the value from a text box**

```
var gallons = $("#gallons").val();
```

**How to set the value for an input element**

```
$("#gallons").val("");
```

**How to set the text in an element**

```
$("#email_address_error").text(
    "Email address is required");
```

**How to set the text for the next sibling with object chaining**

```
$("#last_name").next().text("Last name is required");
```

**How to submit a form**

```
$("#join_list").submit();
```

**How to move the focus to a form control or link**

```
$("#email_address").focus();
```

# jQuery for Events

## The syntax for a jQuery event method

```
$(selector).eventMethodName(function() {
    // the statements of the event handler
});
```

## Two common jQuery event methods

| Event method | Description |
|---|---|
| `ready(handler)` | The event handler runs when the DOM is ready. |
| `click(handler)` | The event handler runs when the selected element is clicked. |

# How to code an event handler for the ready event

**The long way**

```
$(document).ready(function() {
    alert("The DOM is ready");
});
```

**The short way**

```
$(function(){                    // (document).ready is assumed
    alert("The DOM is ready");
});
```

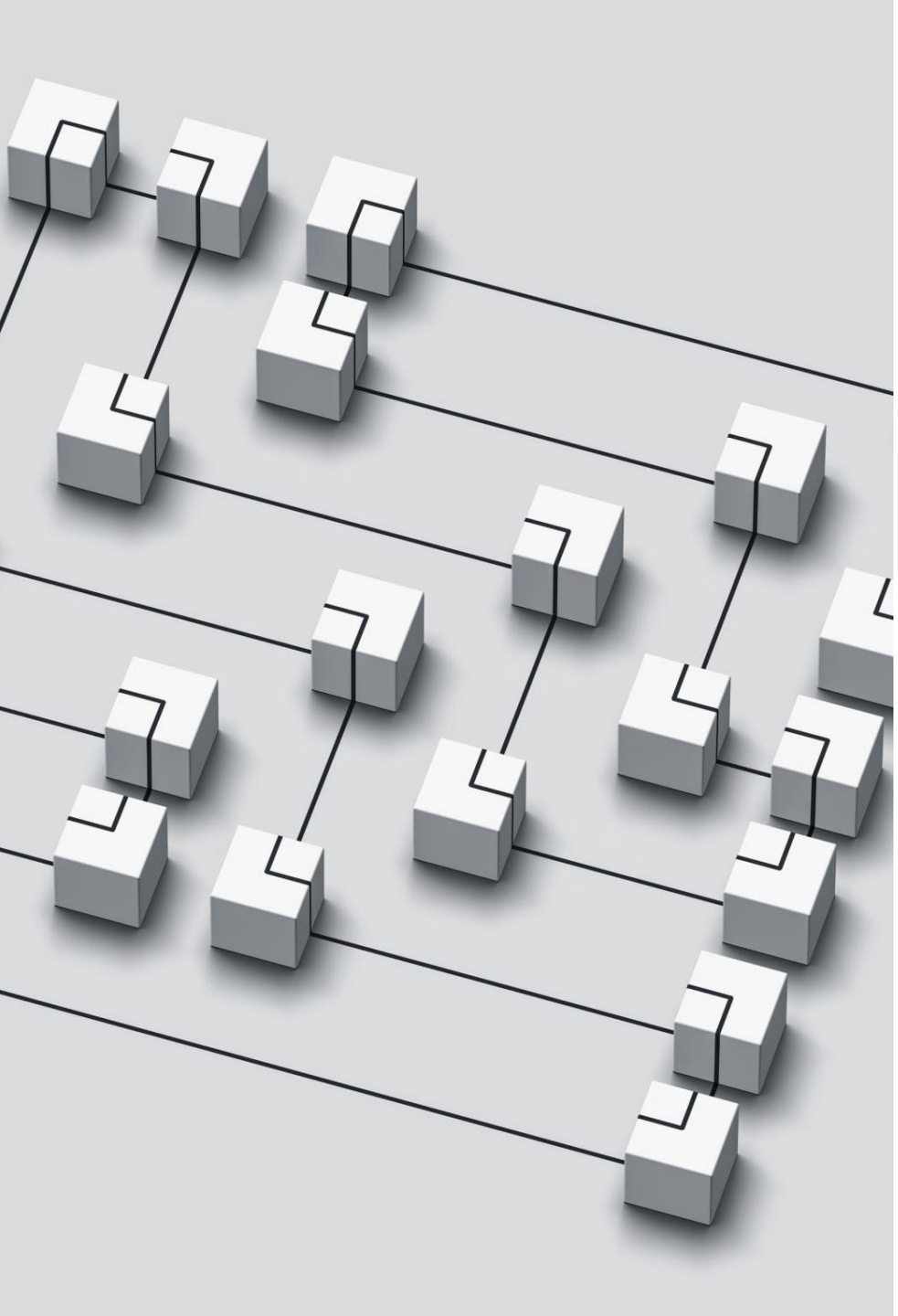
# How to code an event handler for the click event

## An event handler for the click event of all h2 elements

```
$("h2").click(function() {
    alert("This heading has been clicked");
});
```

## The click event handler within the ready event handler

```
$(document).ready(function() {
    $("h2").click(function() {
        alert("This heading has been clicked");
    }); // end of click event handler
}); // end of ready event handler
```



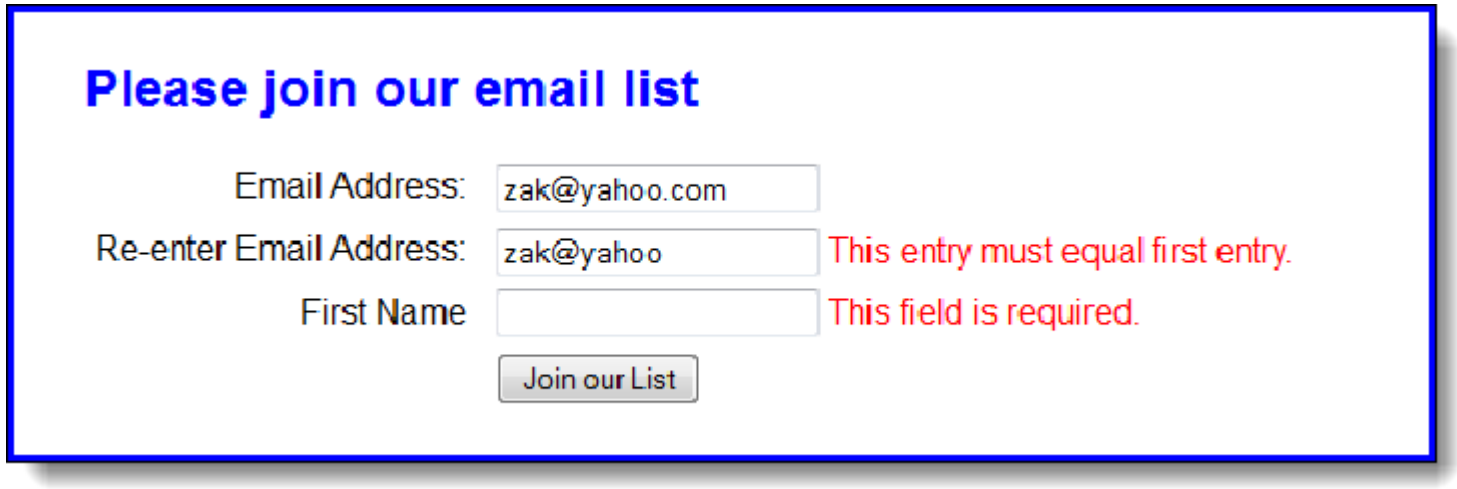
# Terms

- selector
- method
- object chaining
- event method

# The Email List application in jQuery [Optional]

SECTION 16

# The user interface for the Email List application



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Join Email List</title>
    <link rel="stylesheet" href="email_list.css">
    <script src=
        "http://html5shiv.googlecode.com/svn/trunk/html5.js">
    </script>
    <script src="http://code.jquery.com/jquery-latest.min.js">
    </script>
    <script src="email_list.js"></script>
</head>
<body>
    <section>
        <h1>Please join our email list</h1>
        <form id="email_form" name="email_form"
              action="join.html" method="get">
            <label for="email_address1">Email Address:</label>
            <input type="text" id="email_address1">
            <span>*</span><br>
            <label for="email_address2">
                  Re-enter Email Address:</label>
            <input type="text" id="email_address2">
            <span>*</span><br>

            <label for="first_name">First Name:</label>
            <input type="text" id="first_name">
            <span>*</span><br>

            <label> </label>
            <input type="button" id="join_list"
                  value="Join our List">
        </form>
    </section>
</body>
</html>
```

# The HTML for the Email List application

```
$(document).ready(function() {
    $("#join_list").click(function() {
        var emailAddress1 = $("#email_address1").val();
        var emailAddress2 = $("#email_address2").val();
        var isValid = true;

        // validate the first email address
        if (emailAddress1 == "") {
            $("#email_address1").next().text(
                "This field is required.");
            isValid = false;
        } else {
            $("#email_address1").next().text("");
        }

        // validate the second email address
        if (emailAddress2 == "") {
            $("#email_address2").next().text(
                "This field is required.");
            isValid = false;
        } else if (emailAddress1 != emailAddress2) {
            $("#email_address2").next().text(
                "This entry must equal first entry.");
            isValid = false;
        } else {
            $("#email_address2").next().text("");
        }

        // validate the first name entry
        if ($("#first_name").val() == "") {
            $("#first_name").next().text(
                "This field is required.");
            isValid = false;
        }
        else {
            $("#first_name").next().text("");
        }

        // submit the form if all entries are valid
        if (isValid) {
            $("#email_form").submit();
        }
    }); // end click
    $("#email_address1").focus();
}); // end ready
```

# The jQuery for the Email List application

# A working subset of selectors, methods, and event methods [Optional]

SECTION 17

# Some of the most useful jQuery selectors

```
[attribute]
[attribute=value]
:eq(n)
:even
:first
:first-child
:gt(n)
:header
:last
:last-child
:lt(n)
:not(selector)
:odd
:text
```

# Examples that use jQuery selectors

**Select the li elements that are the first child of their parent element**

```
$("li:first-child")
```

**Select the even tr elements of a table**

```
$("table > tr:even")      // numbering starts at 0
```

**Select the third descendant <p> element of an element**

```
$("#faqs p:eq(2)")        // numbering starts at 0
```

**Select all input elements with “text” as the type attribute**

```
$(":text")
```

# A summary of the most useful jQuery methods

```
next([selector])
prev([selector])
attr(attributeName)
attr(attributeName, value)
css(propertyName)
css(propertyName, value)
addClass(className)
removeClass([className])
toggleClass(className)
hide([duration])
show([duration])
each(function)
```

# Examples that use jQuery methods

**Get the value of the src attribute of an image**

```
$("#image").attr("src");
```

**Set the value of the src attribute of an image to the value of a variable**

```
$("#image").attr("src", imageSource);
```

**Set the value of the color property of the h2 elements**

```
$("h2").css("color", "blue");
```

**Add a class to the h2 descendants of the “faqs” element**

```
$("#faqs h2").addClass("minus");
```

**Run a function for each <a> element within an “image_list” element**

```
$("#image_list a").each(function() {
    // the statements of the function
});
```

# Some of the most useful jQuery event methods

```
ready(handler)
unload(handler)
error(handler)
click(handler)
toggle(handlerEven, handlerOdd)
dblclick(handler)
mouseenter(handler)
mouseover(handler)
mouseout(handler)
hover(handlerIn, handlerOut)
```

# Examples that use jQuery event methods

**A handler for the double-click event of all text boxes that clears the clicked box**

```
$(":text").dblclick(function () {
    $(this).val("");
}
```

**A handler for the hover event of each img element within a list**

```
$("#image_list img").hover(
    function() {
      alert("The mouse pointer has moved into an img element");
    },
    function() {
      alert("The mouse pointer has moved out of an img element);
    }
);  // end hover
```

# Other event methods that you should be aware of

| Event method | Description |
| --- | --- |
| `bind(event, handler)` | Attach an event handler to an event. |
| `unbind(event, handler)` | Remove an event handler from an event. |
| `one(event, handler)` | Attach an event handler and remove it after it runs one time. |
| `trigger(event)` | Trigger the event for the selected element. |

```
var clearClick = function () {
    // the statements for the event handler
}
```

# How to store an event handler in a variable

# How to attach an event handler to an event

## How to attach an event handler to an event

### With the bind method

```
$("#clear").bind(click, clearClick);
```

### With the shortcut method

```
$("#clear").click(clearClick);
```

## How to attach an event handler to two different events

```
$("#clear").click(clearClick);
$(":text").dblclick(clearClick);
```

# How to remove an event handler from an event

## How to remove an event handler from an event

```
$("#clear").unbind("click", clearClick);
```

## How to attach and remove an event handler so it runs only once

```
$("#clear").one("click", confirmClick);
```

# How to trigger an event

## How to trigger an event

### With the trigger method

```
$("#clear").trigger("click");
```

### With the shortcut method

```
$("#clear").click();
```

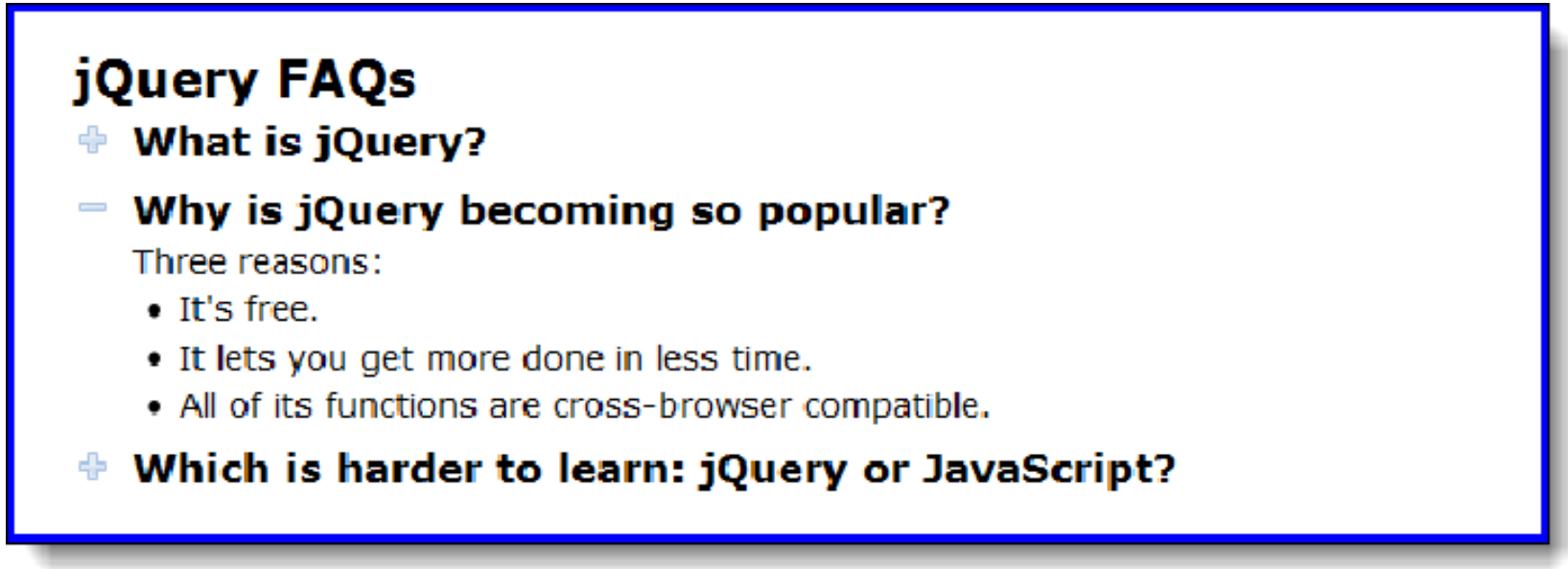
## How to use the shortcut method to trigger an event from an event handler

```
$(":text").dblclick(function () {
    $("#clear").click();      // triggers the click event
                              // of the clear button
}
```

# Three illustrative applications [Optional]

SECTION 18

# The FAQs application in a browser



```
<section id="faqs">
    <h1>jQuery FAQs</h1>
    <h2>What is jQuery?</h2>
    <div>
        <p>jQuery is a library of the JavaScript
           functions that you're most likely to need as
           you develop web sites.
        </p>
    </div>
    <h2>Why is jQuery becoming so popular?</h2>
    <div>
        <p>Three reasons:</p>
        <ul>
            <li>It's free.</li>
            <li>It lets you get more done in less
                time.</li>
            <li>All of its functions are cross-browser
                compatible.</li>
        </ul>
    </div>
    <h2>Which is harder to learn: jQuery or JavaScript?
    </h2>
    <div>
        <p>For most functions, jQuery is significantly
           easier to learn and use than JavaScript. But
           remember: jQuery is JavaScript.
        </p>
    </div>
</section>
```

# The HTML for the FAQs application

```
$(document).ready(function() {
    $("#faqs h2").toggle(
        function() {
            $(this).addClass("minus");
            $(this).next().show();
        },
        function() {
            $(this).removeClass("minus");
            $(this).next().hide();
        }
    );  // end toggle
}); // end ready
```

# The jQuery for the FAQs application

# The user interface for the Image Swap application

```
<section>
    <h1>Ram Tap Combined Test</h1>
    <ul id="image_list">
        <li><a href="images/h1.jpg" title="James Allison: 1-1">
            <img src="thumbnails/t1.jpg" alt=""></a></li>
        <li><a href="images/h2.jpg" title="James Allison: 1-2">
            <img src="thumbnails/t2.jpg" alt=""></a></li>
        <li><a href="images/h3.jpg" title="James Allison: 1-3">
            <img src="thumbnails/t3.jpg" alt=""></a></li>
        <li><a href="images/h4.jpg" title="James Allison: 1-4">
            <img src="thumbnails/t4.jpg" alt=""></a></li>
        <li><a href="images/h5.jpg" title="James Allison: 1-5">
            <img src="thumbnails/t5.jpg" alt=""></a></li>
        <li><a href="images/h6.jpg" title="James Allison: 1-6">
            <img src="thumbnails/t6.jpg" alt=""></a></li>
    </ul>
    <h2 id="caption">James Allison 1-1</h2>
    <p><img src="images/h1.jpg" alt="" id="image"></p>
</section>
```

# The HTML for the Image Swap application

# The CSS for the li elements

```
li {
    padding-right: 10px;
    display: inline;
}
```

```
$(document).ready(function() {
    // preload images
    $("#image_list a").each(function() {
        var swappedImage = new Image();
        swappedImage.src = $(this).attr("href");
    });

    // set up event handlers for links
    $("#image_list a").click(function(evt) {
        // swap image
        var imageURL = $(this).attr("href");
        $("#image").attr("src", imageURL);

        //swap caption
        var caption = $(this).attr("title");
        $("#caption").text(caption);

        // cancel the default action of the link
        evt.preventDefault();  // jQuery cross-browser method
    }); // end click

    // move focus to first thumbnail
    $("li:first-child a:first-child").focus();
}); // end ready
```

# The JavaScript for the Image Swap application

The Three images with the middle image rolled over

```
<section>
    <h1>Ram Tap Combined Test</h1>
    <ul id="image_rollovers">
        <li><img src="images/h1.jpg" alt=""
                 id="images/h4.jpg"></li>
        <li><img src="images/h2.jpg" alt=""
                 id="images/h5.jpg"></li>
        <li><img src="images/h3.jpg" alt=""
                 id="images/h6.jpg"></li>
    </ul>
</section>
```

# The HTML for the Image Rollover application

```
$(document).ready(function() {
    $("#image_rollovers img").each(function() {
        var oldURL = $(this).attr("src");
        var newURL = $(this).attr("id");
        // preload images
        var rolloverImage = new Image();
        rolloverImage.src = newURL;

        // set up event handlers
        $(this).hover(
            function() {
                $(this).attr("src", newURL);
            },
            function() {
                $(this).attr("src", oldURL);
            }
        );  // end hover
    }); // end each
}); // end ready
```

# The JavaScript for the Image Rollover application

# Summary

In this unit, we have :

- Used jQuery to develop common DOM scripting
- Described jQuery.
- Described two ways to include the jQuery library in your web pages.
- Described the use of jQuery selectors, methods, and event methods.
- Described the syntax for a jQuery selector.
- Describe object chaining.