

# Computer Science Principles

## Web Programming

## JavaScript Programming Essentials

CHAPTER 5: THE BASIC OF HTML

DR. ERIC CHOU

IEEE SENIOR MEMBER

# ELEMENTS OF

# A GOOD WEBSITE

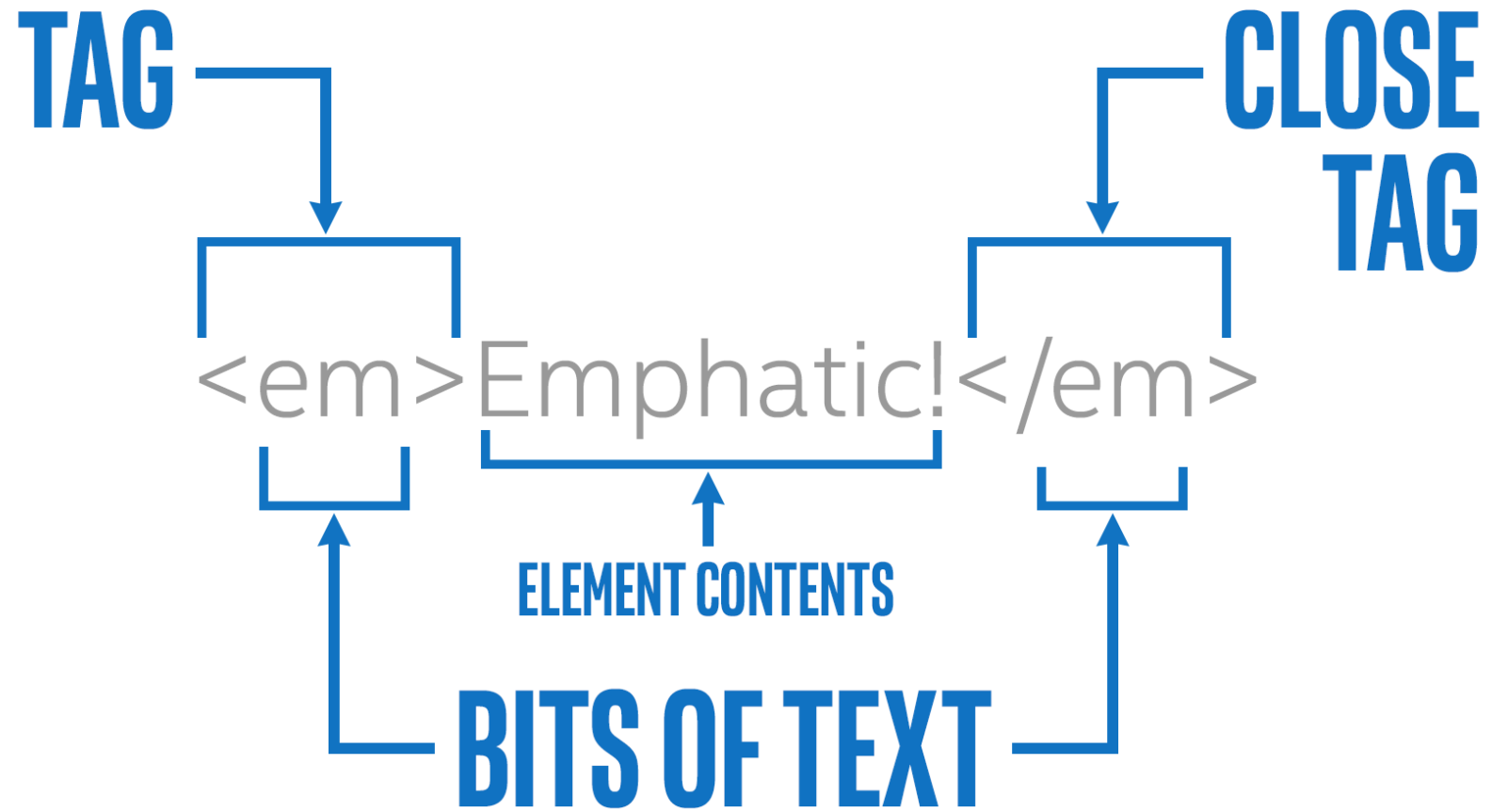




# Elements

---

LECTURE 1





# 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 `<p>` and ends with the end tag `</p>`. 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>

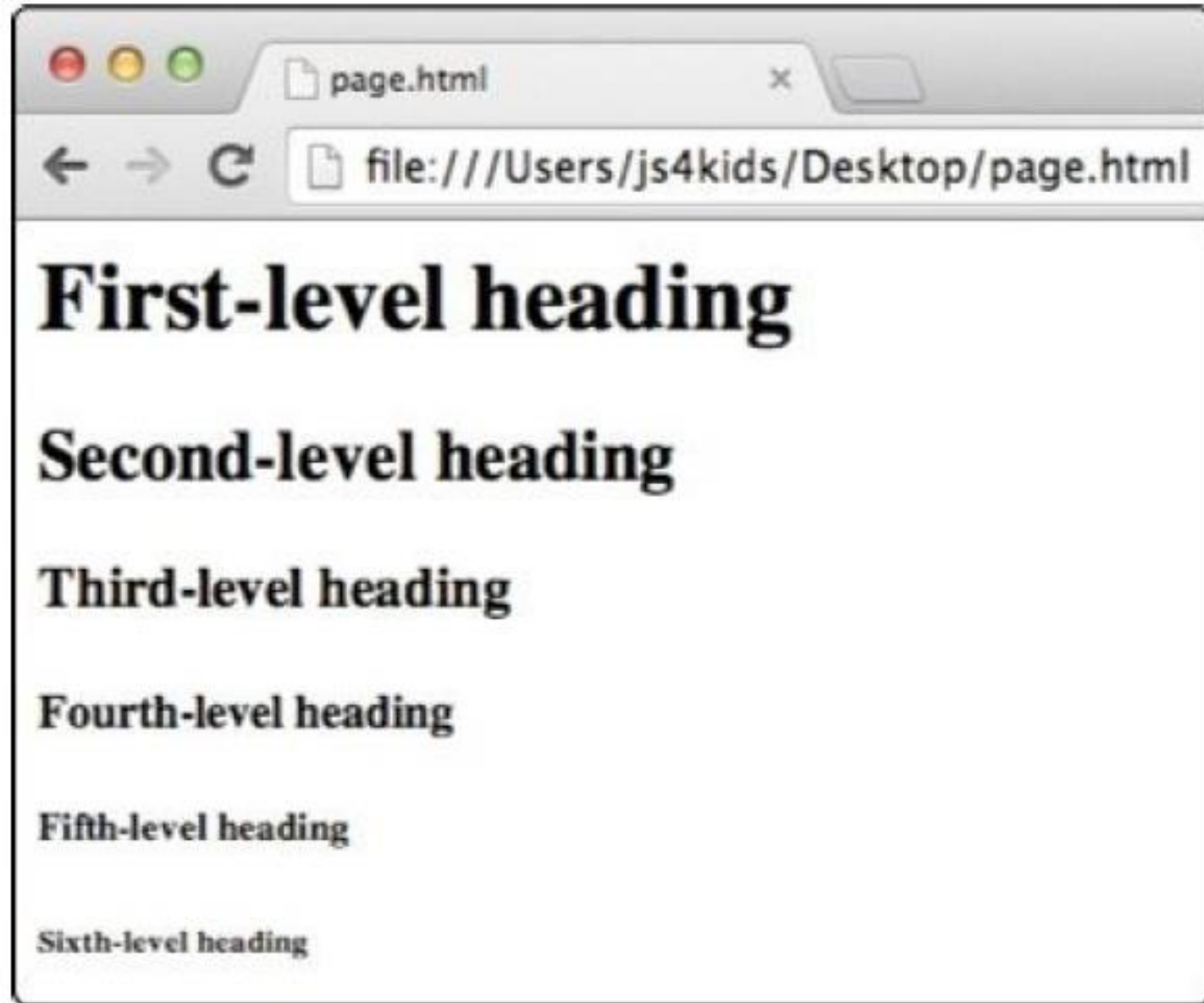
<h2>Second-level heading</h2>

<h3>Third-level heading</h3>

<h4>Fourth-level heading</h4>

<h5>Fifth-level heading</h5>

<h6>Sixth-level heading</h6>







# 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>
```





# 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



# Demo Program: block.html

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

[Copy](#)

## Pictorial presentation

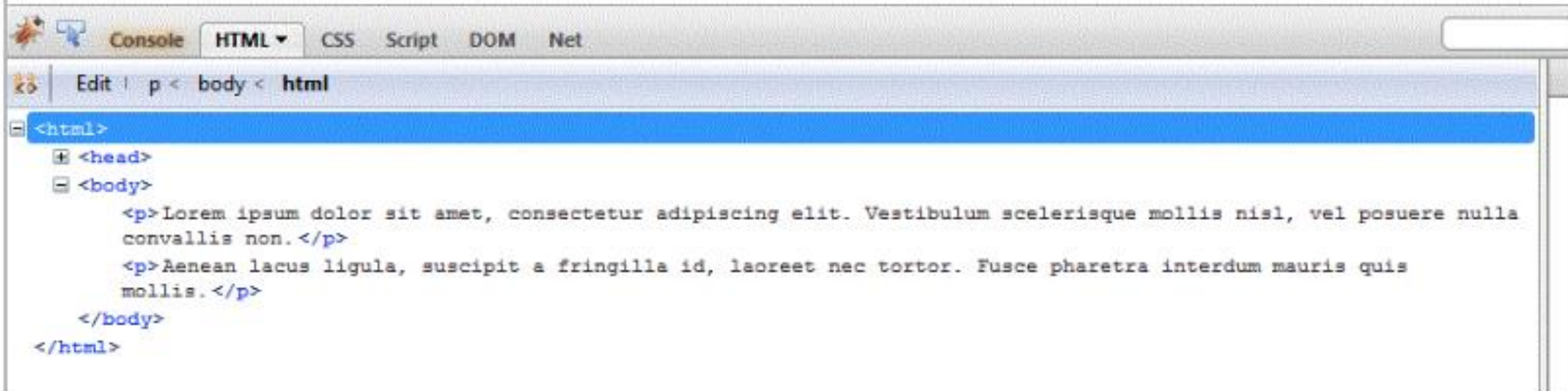
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum scelerisque mollis nisl, vel posuere nulla convallis non.

Aenean lacus ligula, suscipit a fringilla id, laoreet nec tortor. Fusce pharetra interdum mauris quis mollis.

**Both of the paragraphs began with a new line**

↑  
**New Line**

↑  
**New Line**



```

Edit | p < body < html
<html>
  <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>
```



# 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.





# 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.



# Demo Program

---

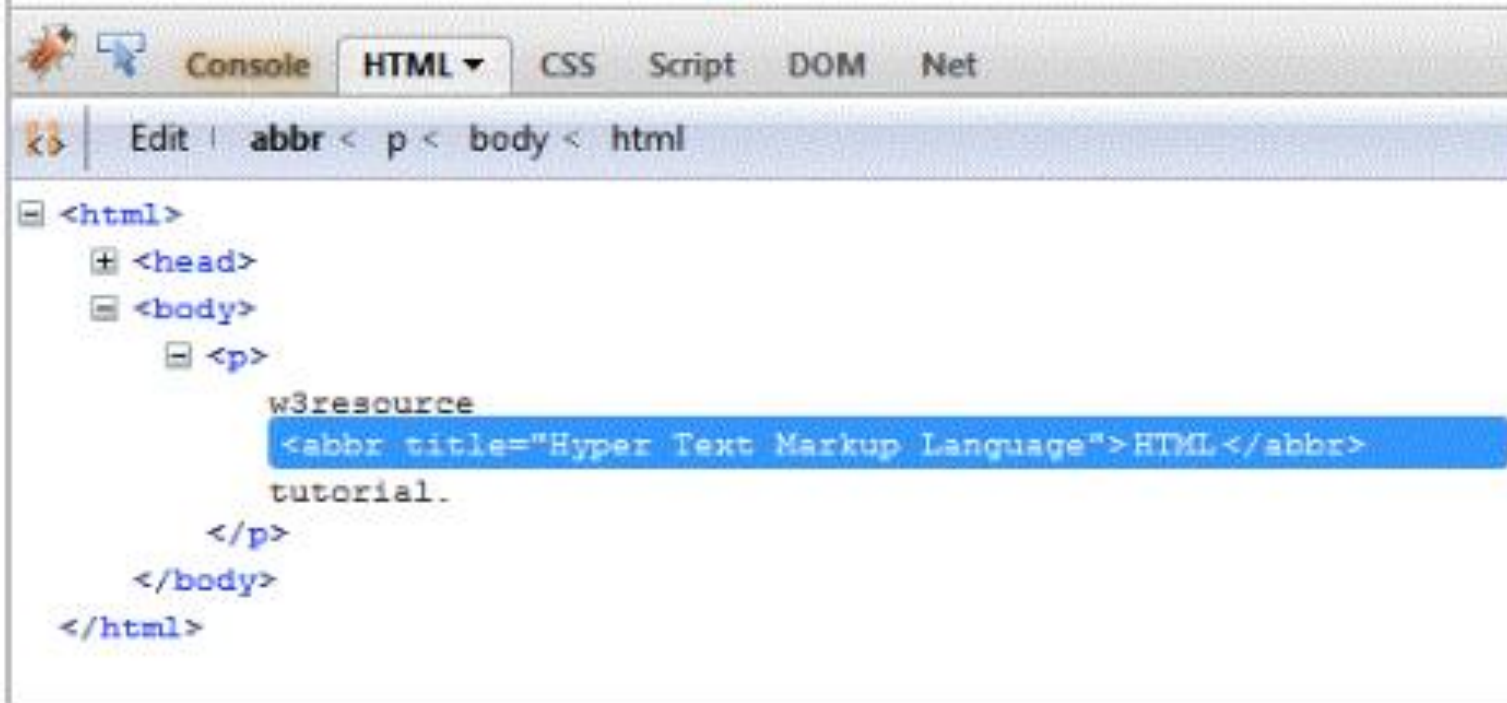
```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
5  <title>Example of HTML block level element</title>
6  </head>
7  <body>
8  <p>w3resource <abbr title="Hyper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

## Pictorial presentation

w3resource HTML tutorial.



**Short and did not begin with a new line**



```
<html>
  <head>
  <body>
    <p>
      w3resource
      <abbr title="Hyper Text Markup Language">HTML</abbr>
      tutorial.
    </p>
  </body>
</html>
```



# Static Pages

---

LECTURE 2

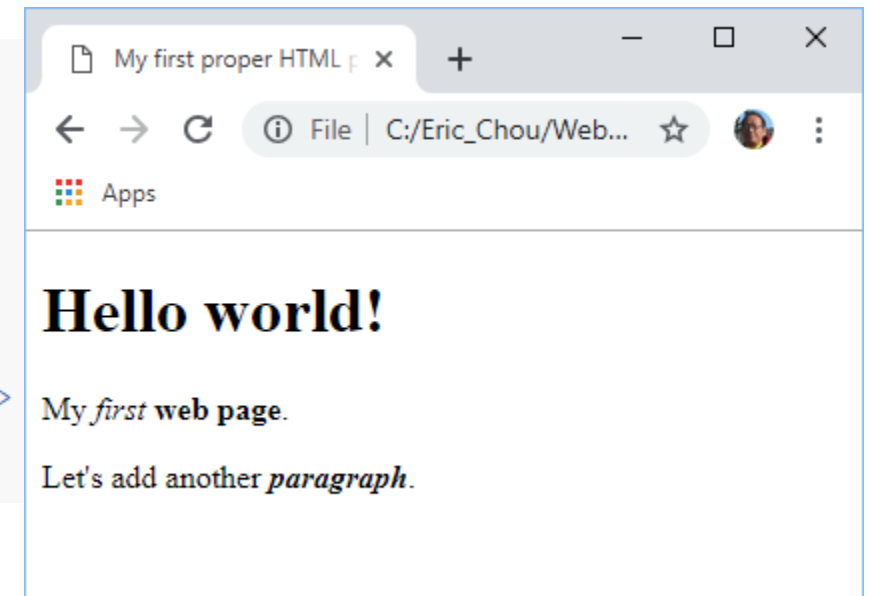


# A Full HTML Document

## Demo Program: static1.html

- 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:

```
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>
```





# 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.
- 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.

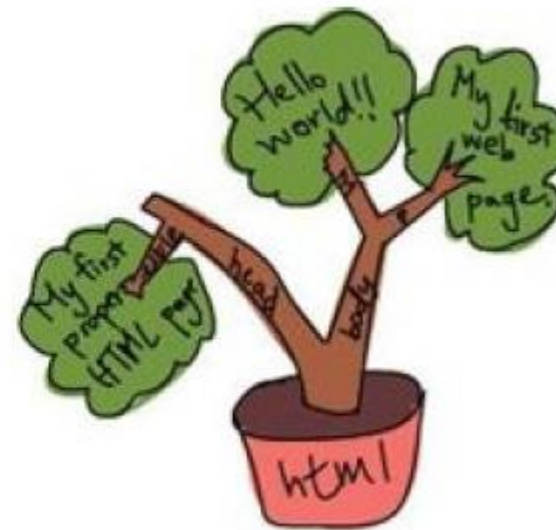
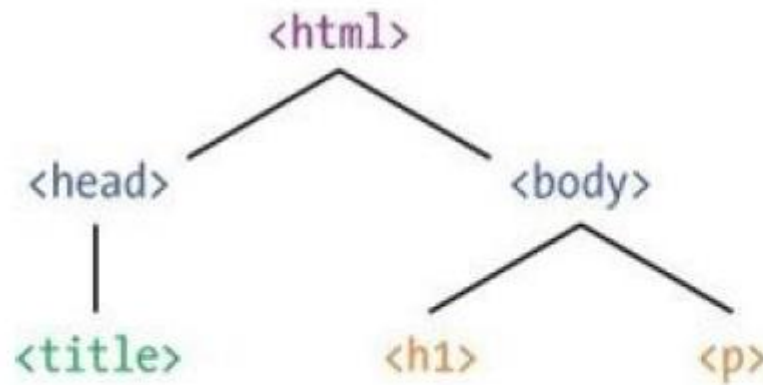


Figure 5-7. The elements from Figure 5-6, shown as a tree



# 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.

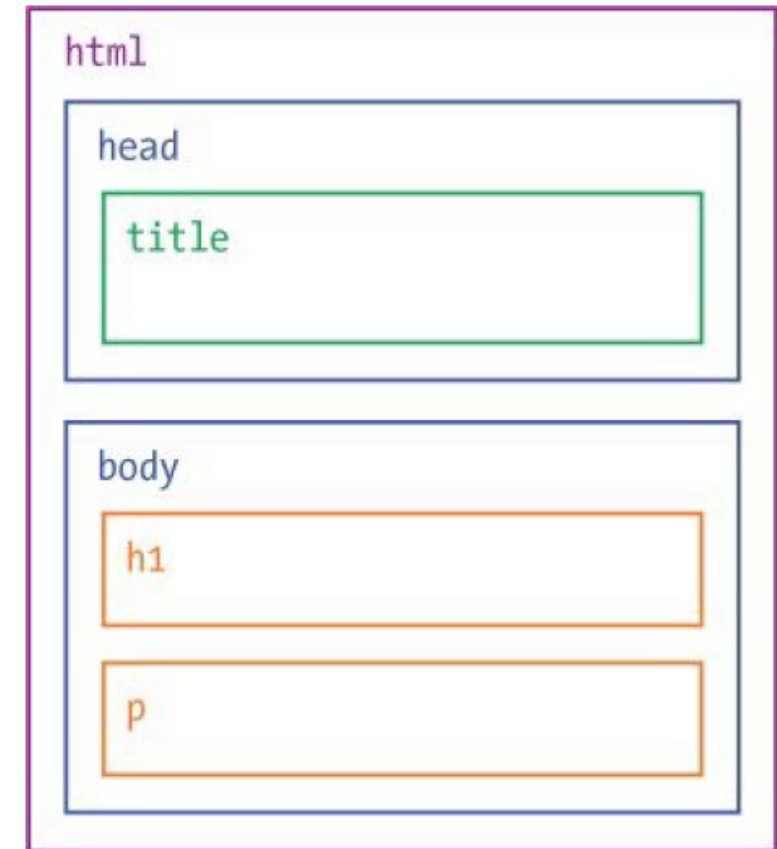


Figure 5-8. The HTML hierarchy, shown as nested boxes



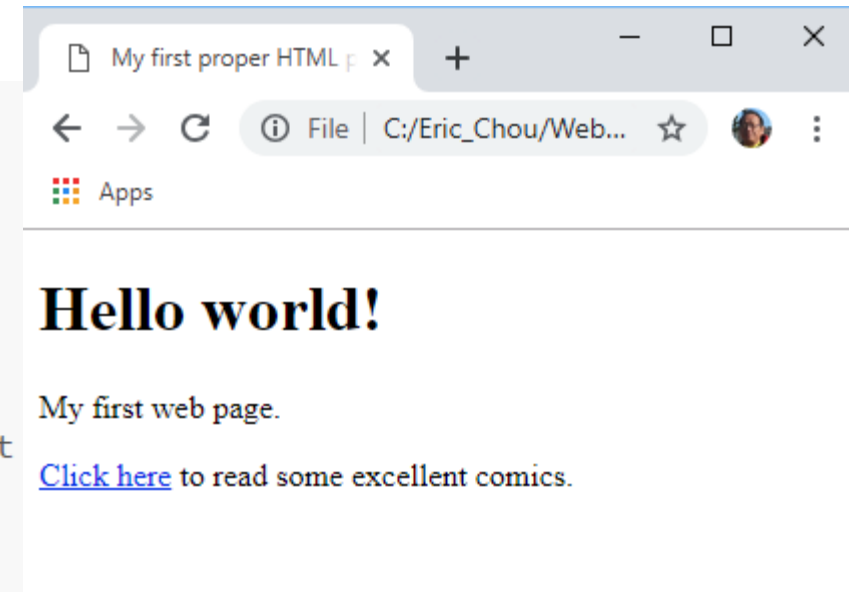


# Adding Links to Your HTML

## 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 `<em>` and `<strong>` 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>
```



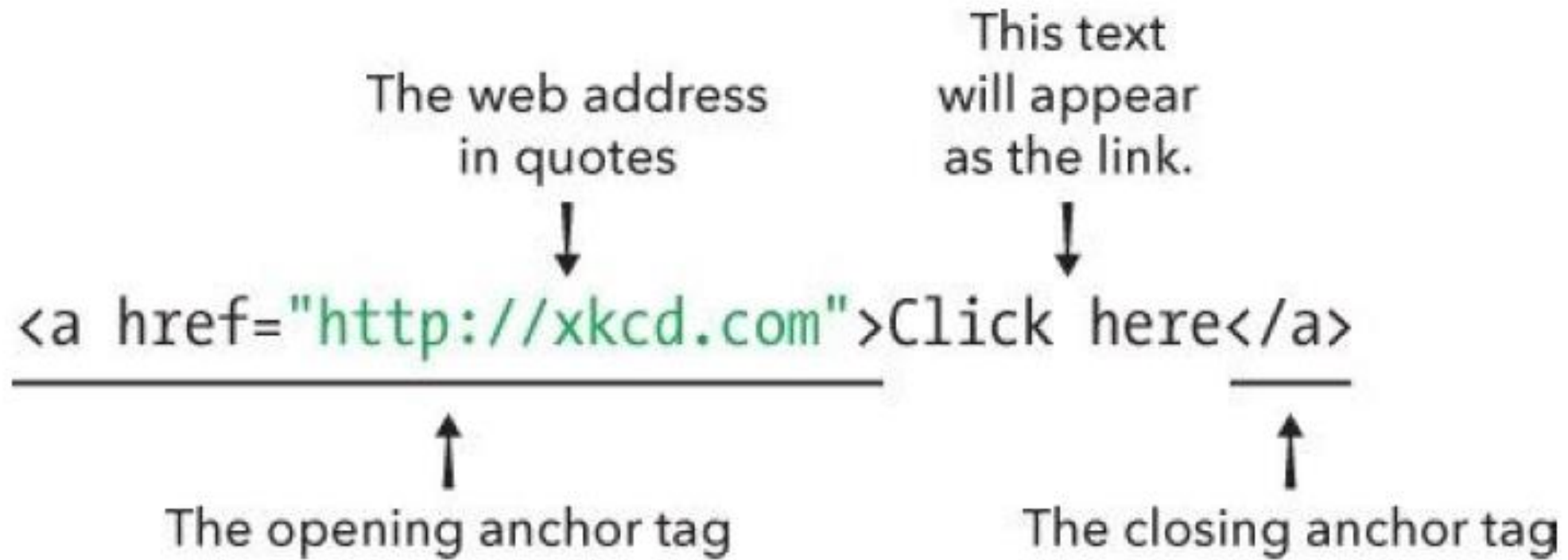


# Link Attributes

---

- 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.



*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:  

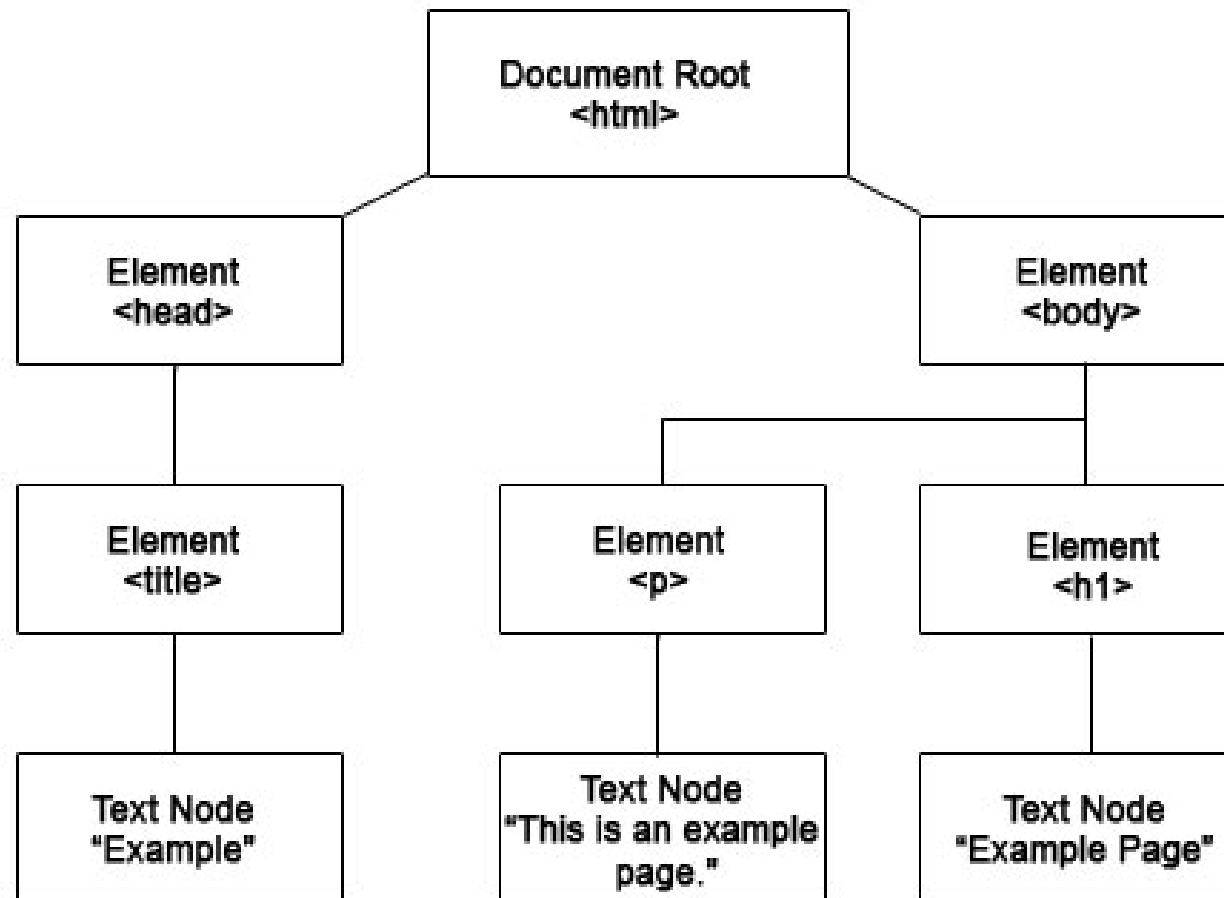
```
<a href="http://xkcd.com" title="xkcd: Land of geeky  
comics!">Click here</a>
```
- Now reload the page. When you hover your cursor over the link, you should see the text “xkcd: Land of geeky comics!” floating above the page.



# Dynamic Pages

---

LECTURE 3





# 10 Essential DOM Methods & Techniques for Practical JavaScript

---

**1. getElementById**

**2. getElementsByTagName**

**3. Node Methods**

**4. createElement**

**5. appendChild**

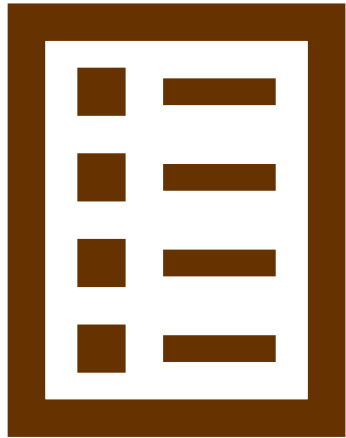
**6. removeChild**

**7. getAttribute**

**8. setAttribute**

**9. document.forms**

**10. innerHTML**



# Access of Elements

---





# getElementById("id")

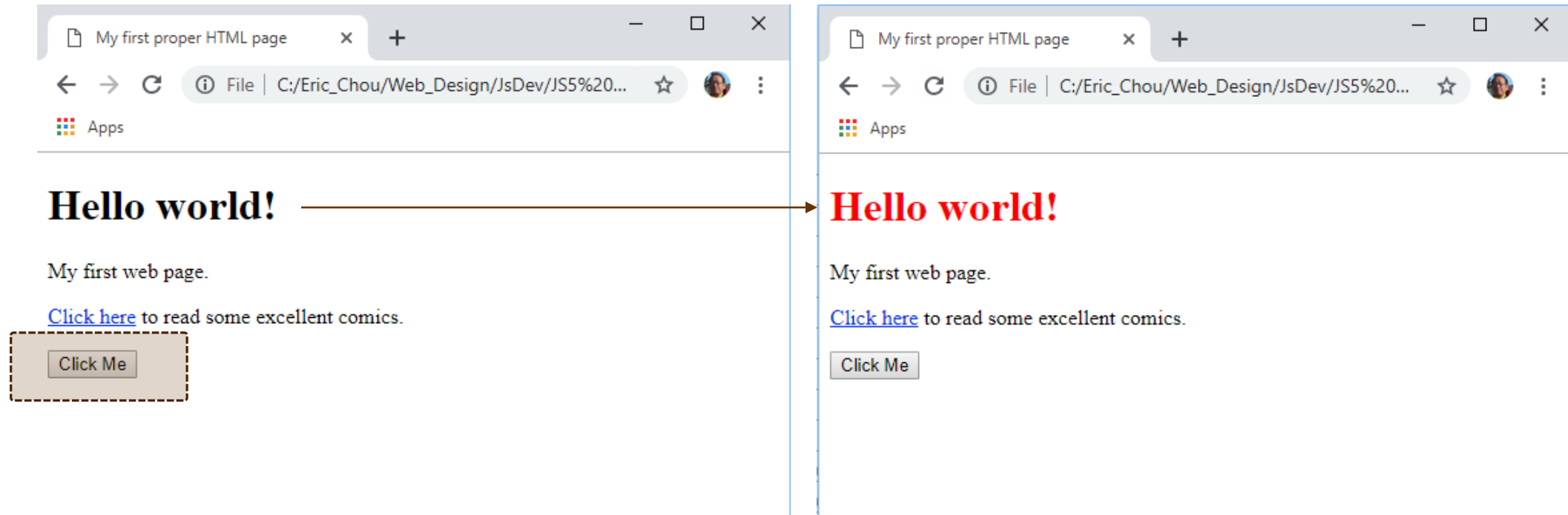
Demo Program: dynamic1.html

```
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

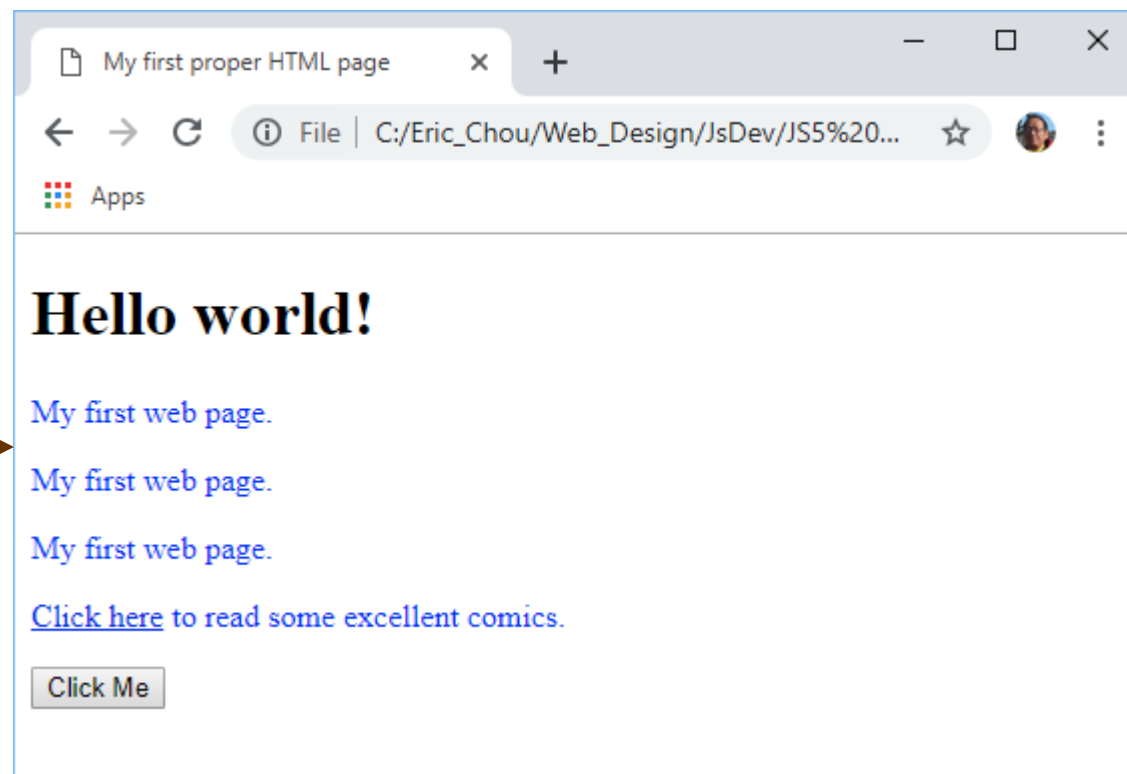
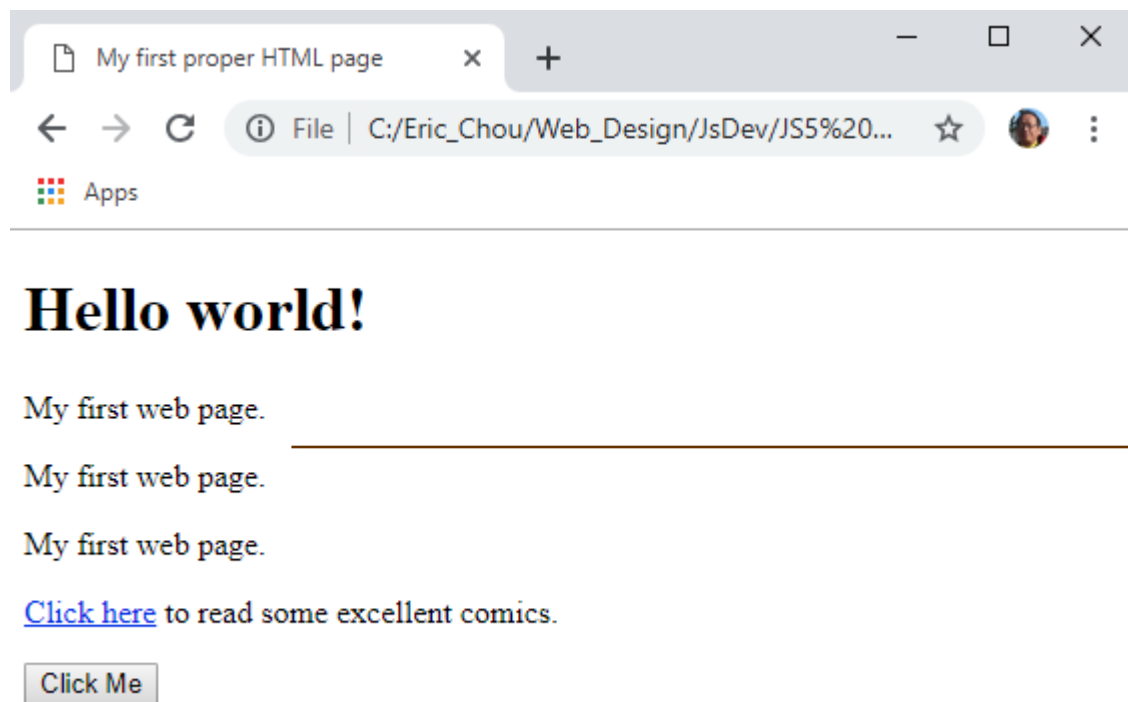




# getElementsByTagName("tn")

Demo Program: dynamic2.html

```
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>My first web page.</p>
10 <p>My first web page.</p>
11 <p><a href="http://xkcd.com">Click here</a> to read some excellent
12 comics.
13 </p>
14 <button onclick="change()">Click Me</button>
15 <script>
16     function change(){
17         var elems = document.getElementsByTagName("p");
18         for (var i=0; i<elems.length; i++)
19             elems[i].style.color="blue"; // change color to red
20     }
21 </script>
22 </body>
23 </html>
```

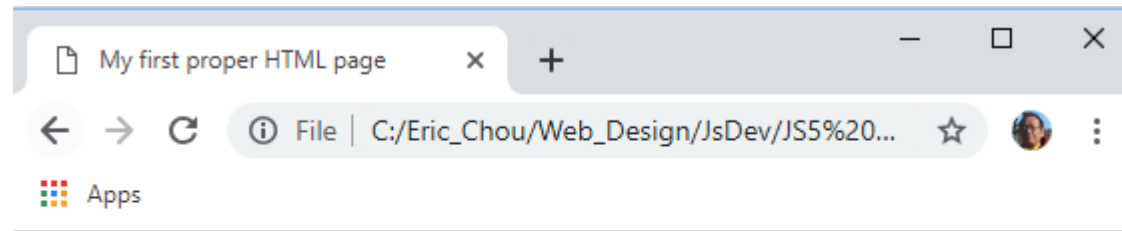




# getElementsByClassName("tn")

Demo Program: dynamic3.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>My first proper HTML page</title>
5  </head>
6  <body>
7  <h1 id="h1" class="important">Hello world!</h1>
8  <p>My first web page.</p>
9  <p class="important">My first web page.</p>
10 <p>My first web page.</p>
11 <p><a href="http://xkcd.com">Click here</a> to read some excellent
12 comics.
13 </p>
14 <button onclick="change()">Click Me</button>
15 <script>
16     function change(){
17         var elems = document.getElementsByClassName("important");
18         for (var i=0; i<elems.length; i++)
19             elems[i].style.color="purple"; // change color to red
20     }
21 </script>
22 </body>
23 </html>
```



**Hello world!**

My first web page.

My first web page.

My first web page.

[Click here](#) to read some excellent comics.

Click Me



**Hello world!**

My first web page.

My first web page.

My first web page.

[Click here](#) to read some excellent comics.

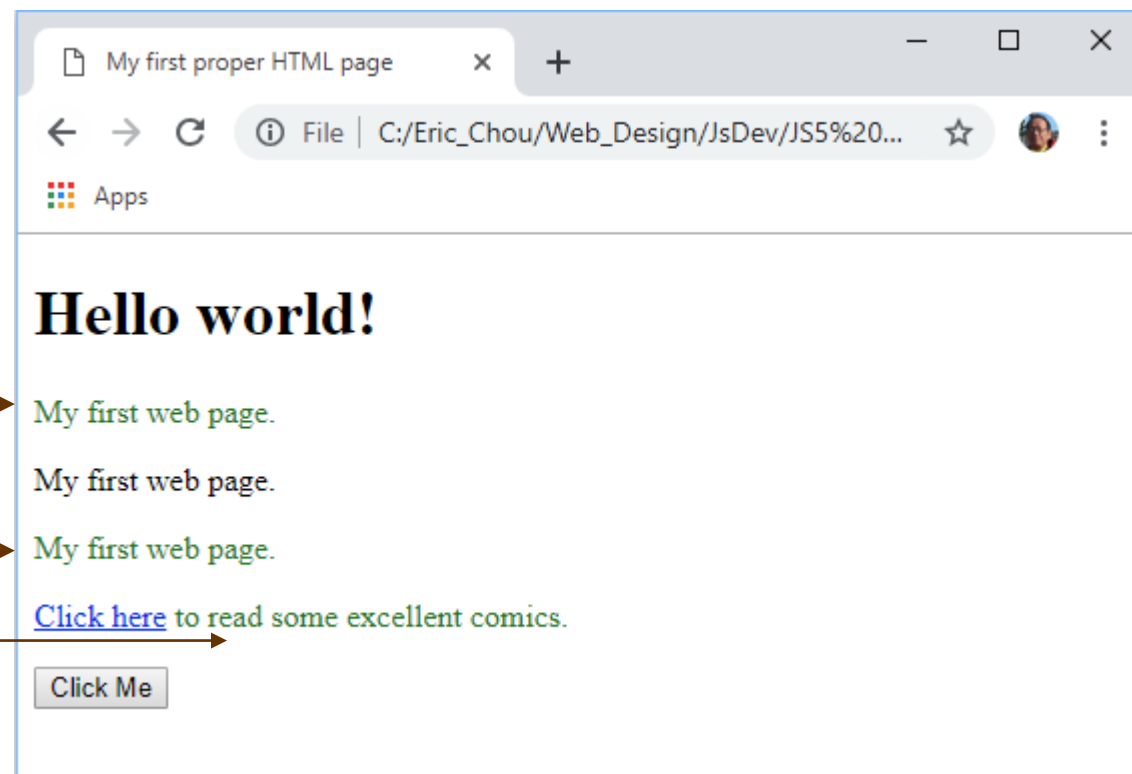
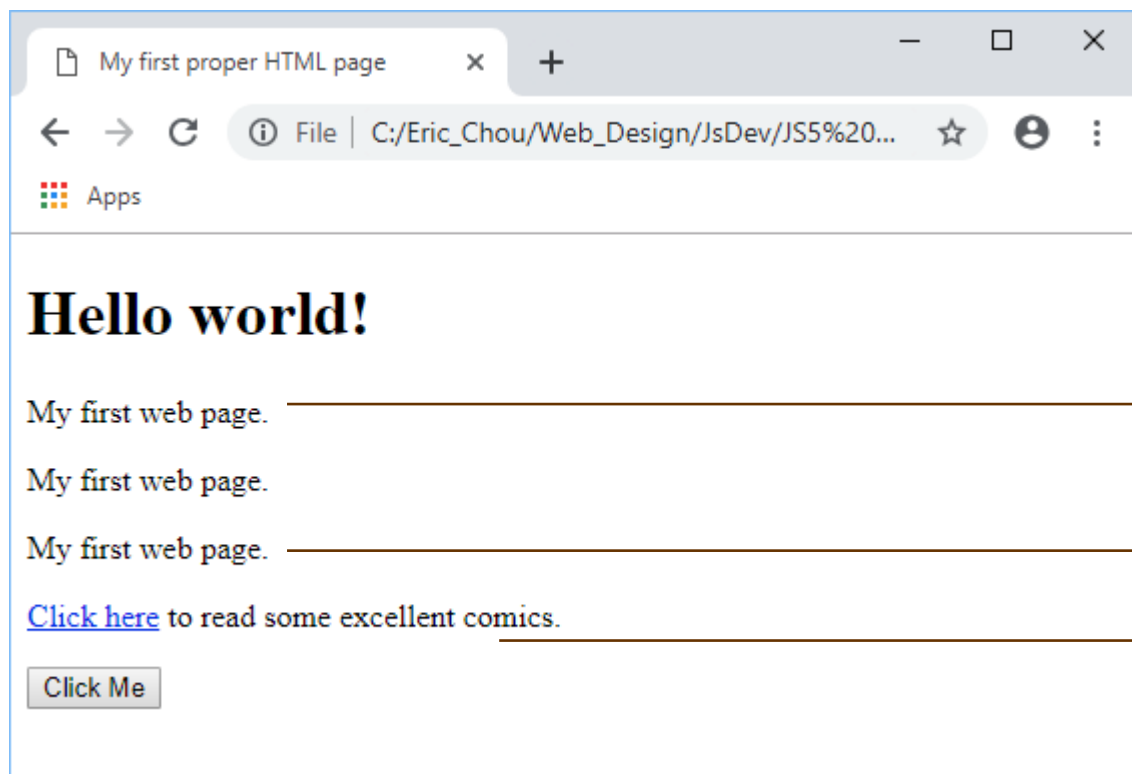
Click Me



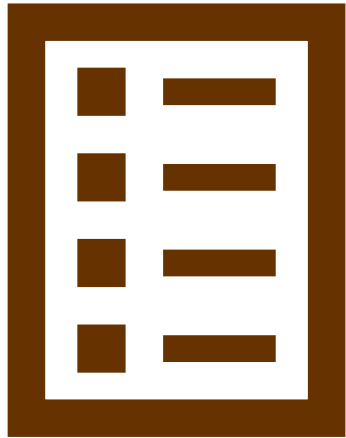
# getElementsByName("name")

Demo Program: dynamic4.html

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <title>My first proper HTML page</title>
5  </head>
6  <body>
7  <h1 id="h1" class="important">Hello world!</h1>
8  <p name="green">My first web page.</p>
9  <p class="important">My first web page.</p>
10 <p name="green">My first web page.</p>
11 <p name="green"><a href="http://xkcd.com">Click here</a> to read some
    excellent
12 comics.
13 </p>
14 <button onclick="change()">Click Me</button>
15 <script>
16   function change(){
17     var elems = document.getElementsByName("green");
18     for (var i=0; i<elems.length; i++)
19       elems[i].style.color="green"; // change color to red
20   }
21 </script>
22 </body>
23 </html>
```



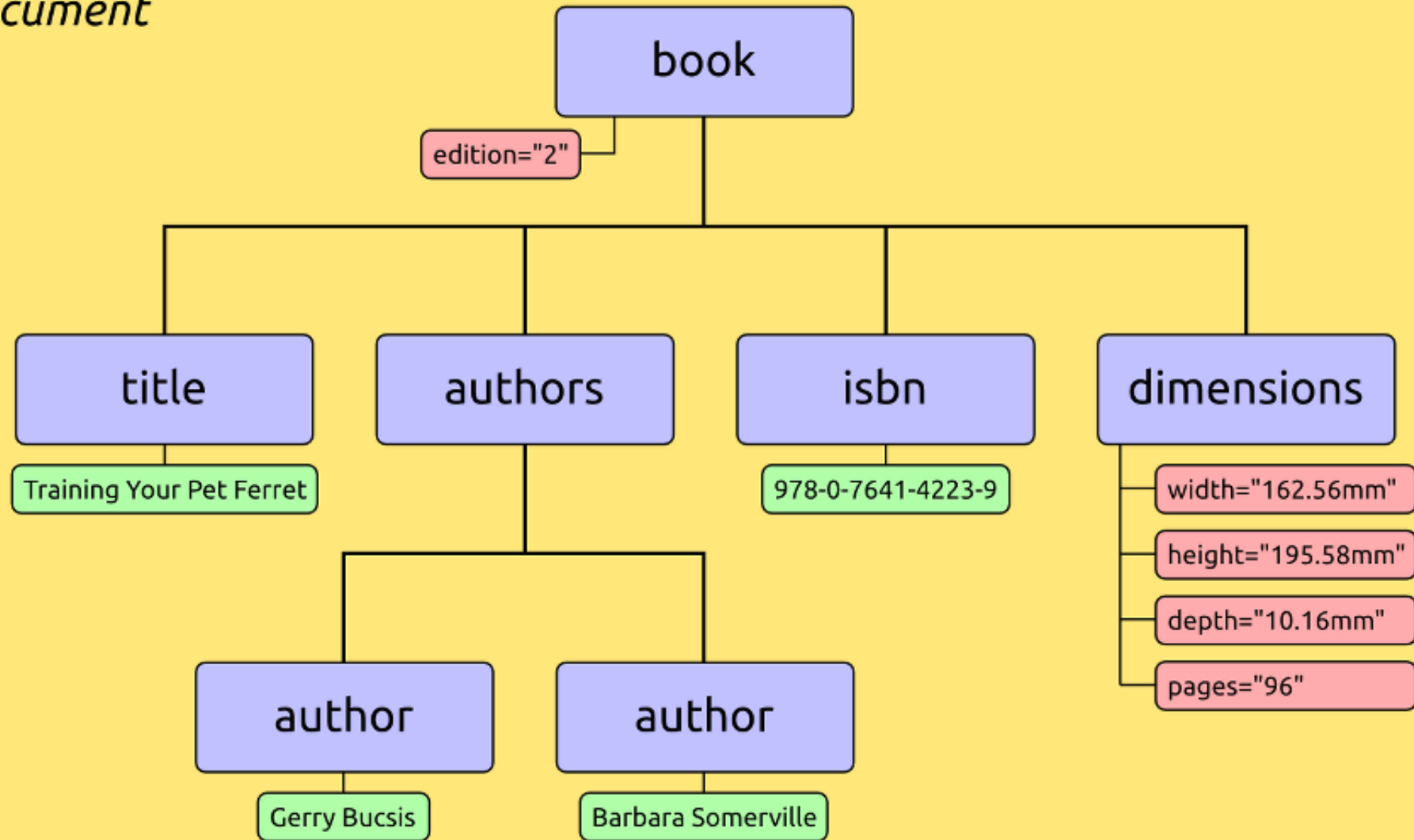




# Using Node to Access Neighbors

---

## Document



Key:



Document



Element



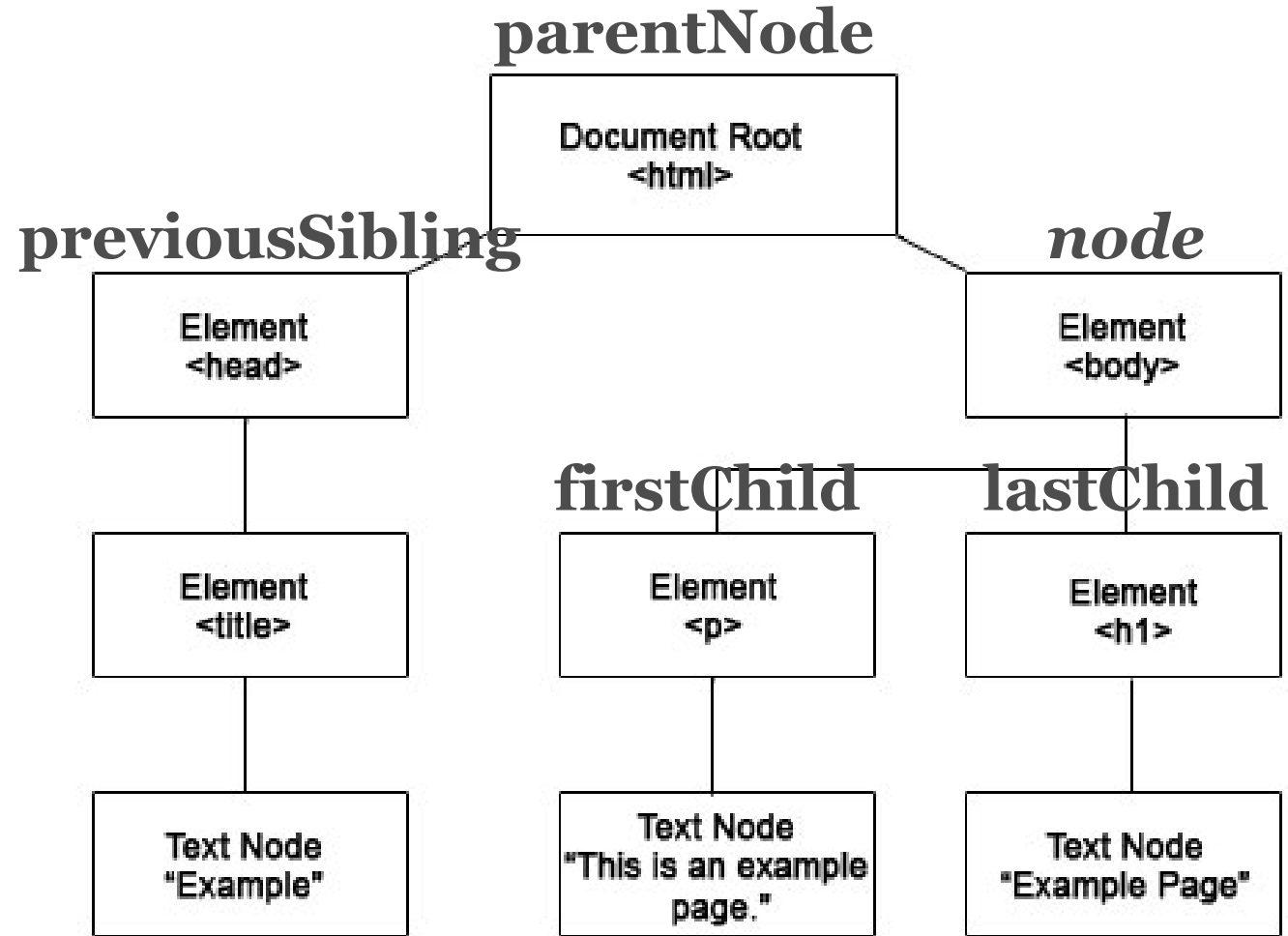
Text content



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`

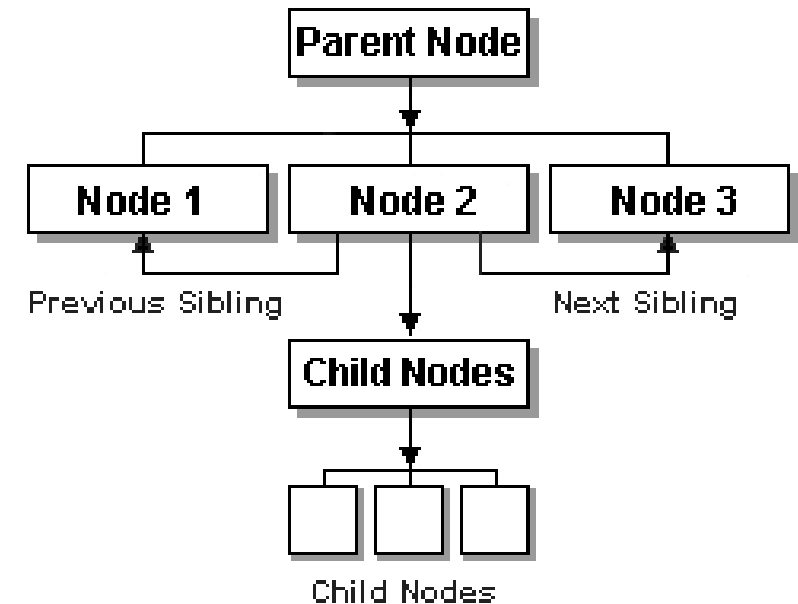
`node.lastChild`

`node.parentNode`

`node.nextSibling`

`node.previousSibling`

Be aware that there might be hidden nodes.



```

1  <!DOCTYPE html>
2  <html id="html">
3  <head id="head">
4    <title id="title">My first proper HTML page</title>
5  </head>
6  <body id="body">
7    <h1 id="h1">Hello world!</h1>
8    <p id="p1">My first web page.</p>
9  <p id="p1"><a href="http://xkcd.com" id="a1">Click here</a> to read some excellent
10 comics.
11 </p>
12 <button onclick="names()" id="button">Click Me</button>
13 <script id="script">
14   function names(){
15     var node = document.getElementById("body");
16     document.write("<h1>Print out node's neighbors: </h1>");
17     document.write("Node: "+node.id+"<br>");
18     document.write("Parent: "+node.parentNode.id+"<br>");
19     document.write("PreviousSibling's nextSibling:"
20                   +node.previousSibling.nextSibling.id+"<br>");
21     var children = node.childNodes;
22     for (var i=0; i<children.length; i++){
23       if (children[i]) document.write("Child["+i+"]="+children[i].id+"<br>");
24     }
25   }
26 </script>
27 </body>
28 </html>

```



# All Other Methods

---

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

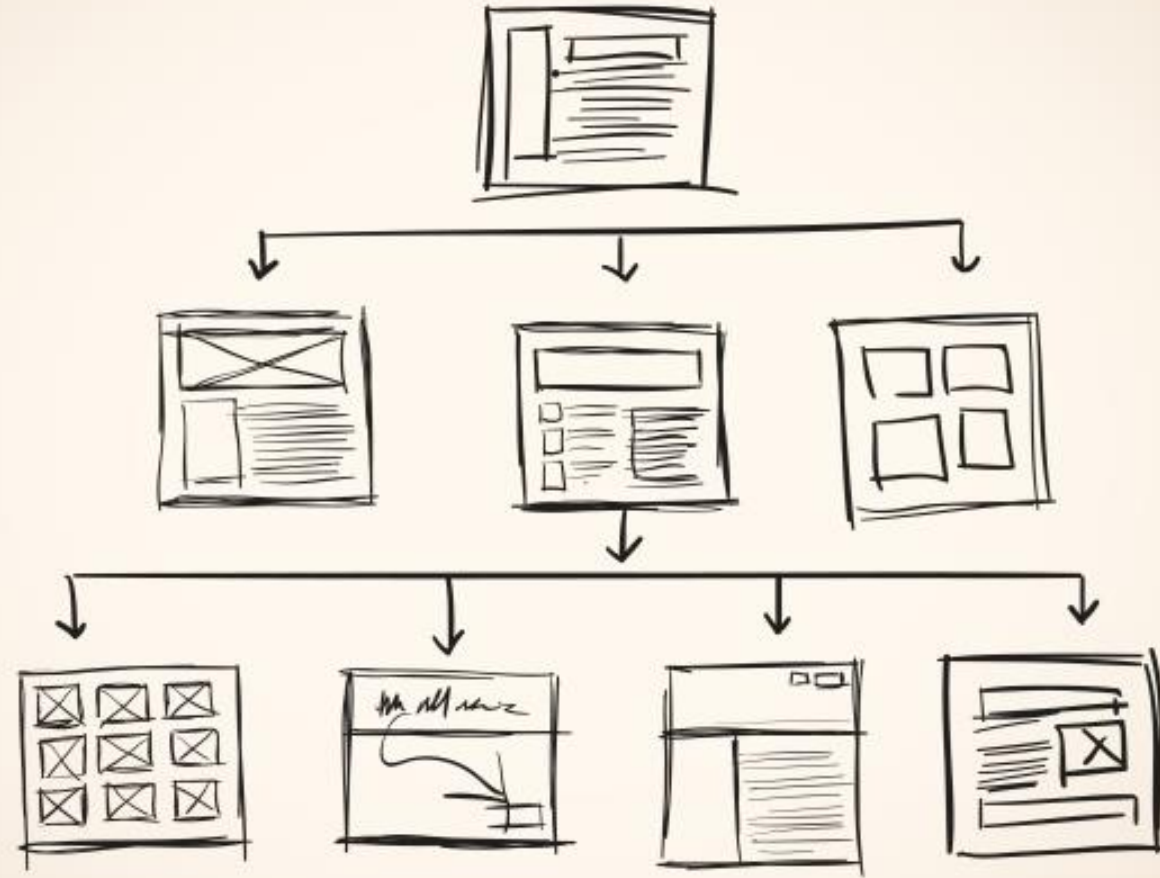


# Site

Putting Things Together

---

LECTURE 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
```

images

scripts

styles

index

LICENSE

README

tree

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   
13
14   <p>At Mozilla, we're a global community of</p>
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   <p>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.</p>
23
24   <p>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.
   </p>
25   <button>Change user</button>
26   <script src="scripts/main.js"></script>
27 </body>
28 </html>
```

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

My test page

Line 32, Column 1 — 32 Lines

INS UTF-8 HTML Spaces: 4

```
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 handler 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()`

```

1▼ html {
2    font-size: 10px;
3    font-family: 'Open Sans', sans-serif;
4 }
5▼ h1 {
6    font-size: 60px;
7    text-align: center;
8 }
9▼ p, li {
10    font-size: 16px;
11    line-height: 2;
12    letter-spacing: 1px;
13 }
14▼ html {
15    background-color: #00539F;
16 }
17▼ body {
18    width: 600px;
19    margin: 0 auto;
20    background-color: #FF9500;
21    padding: 0 20px 20px 20px;
22    border: 5px solid black;
23 }
24▼ h1 {
25    margin: 0;
26    padding: 20px 0;
27    color: #00539F;
28    text-shadow: 3px 3px 1px black;
29 }
30▼ img {
31    display: block;
32    margin: 0 auto;
33 }

```

# 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](#)

# 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

---

LECTURE 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)