Computer Science Principles
Web Programming

JavaScript Programming Essentials

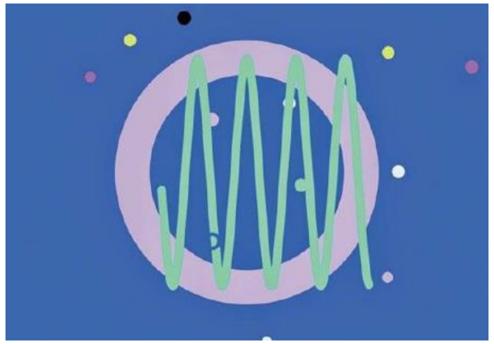
CHAPTER 1: INTRODUCTION

DR. ERIC CHOU

IEEE SENIOR MEMBER

What Is JavaScript?

- Computers are incredibly powerful machines, capable of performing amazing feats like playing competitive chess, serving thousands of web pages, or making millions of complex calculations in less than a few seconds.
- But deep down, computers are actually pretty dumb. Computers can only do exactly what we humans tell them to do. We tell computers how to behave using computer programs, which are just sets of instructions for the computers to follow. Without programs, computers can't do anything at all!



Meet JavaScript

- JavaScript lets you play music and create amazing visual effects.
  - http://lights.helloenjoy.com/
- JavaScript lets you build tools for others to make their own art.
  - https://www.patatap.com/
- JavaScript lets you play fun games.
  - https://www.cubeslam.com/

Overview

LECTURE 1

Writing Some JavaScript

Console Mode

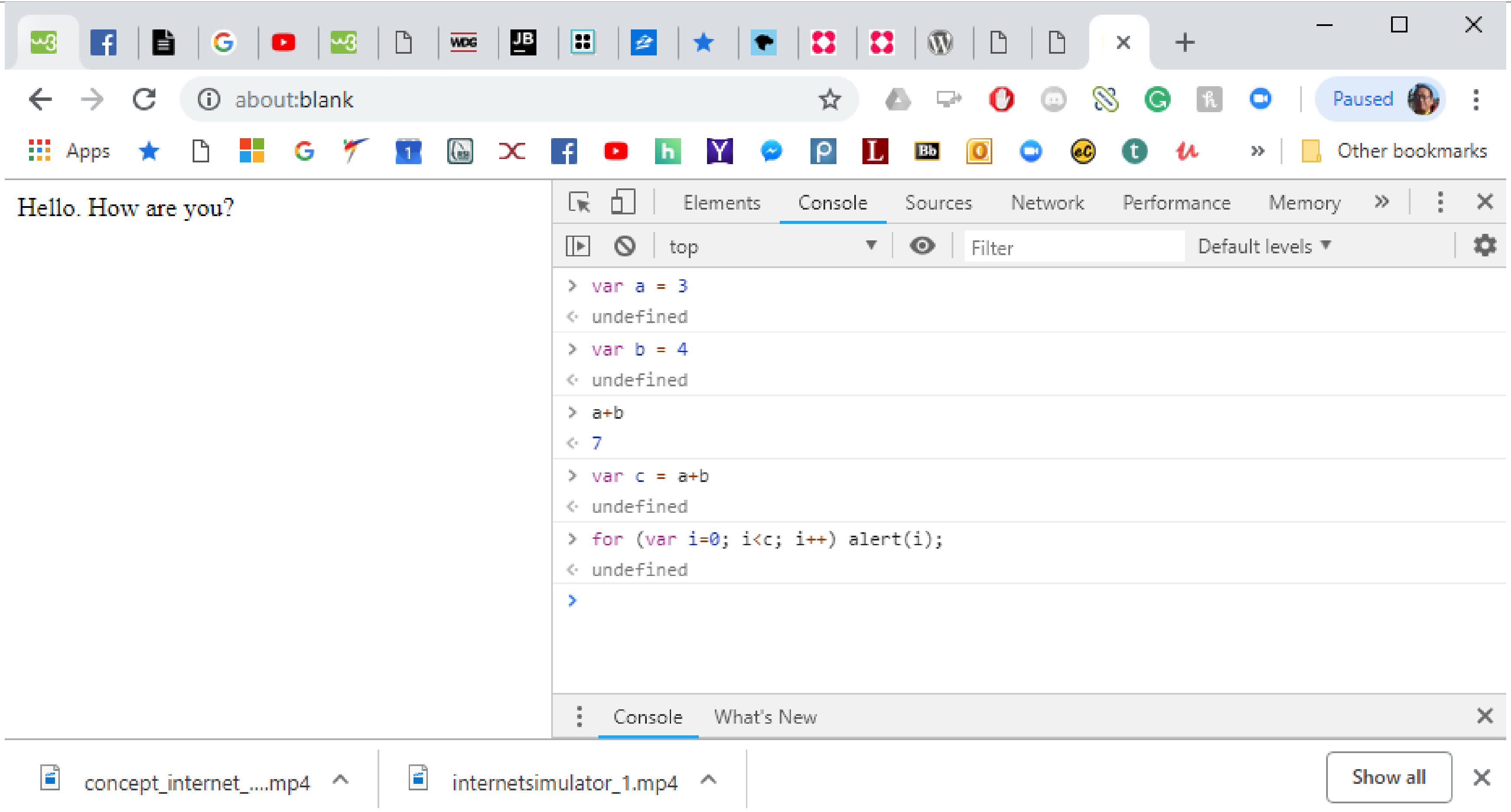
Install Chrome on your computer (if it's not already installed), and then open it and type about:blank in the address bar. Now press ENTER and you'll see a blank page.

We'll begin by coding in Chrome's JavaScript console, which is a secret way programmers can test out short JavaScript programs. On Microsoft Windows or Linux, hold down the CTRL and SHIFT keys and press J. On Mac OS, hold down the COMMAND and OPTION keys and press J.

If you've done everything correctly, you should see a blank web page and, beneath that, a blinking cursor (|) next to a right angle bracket (>). That's where you'll write JavaScript!

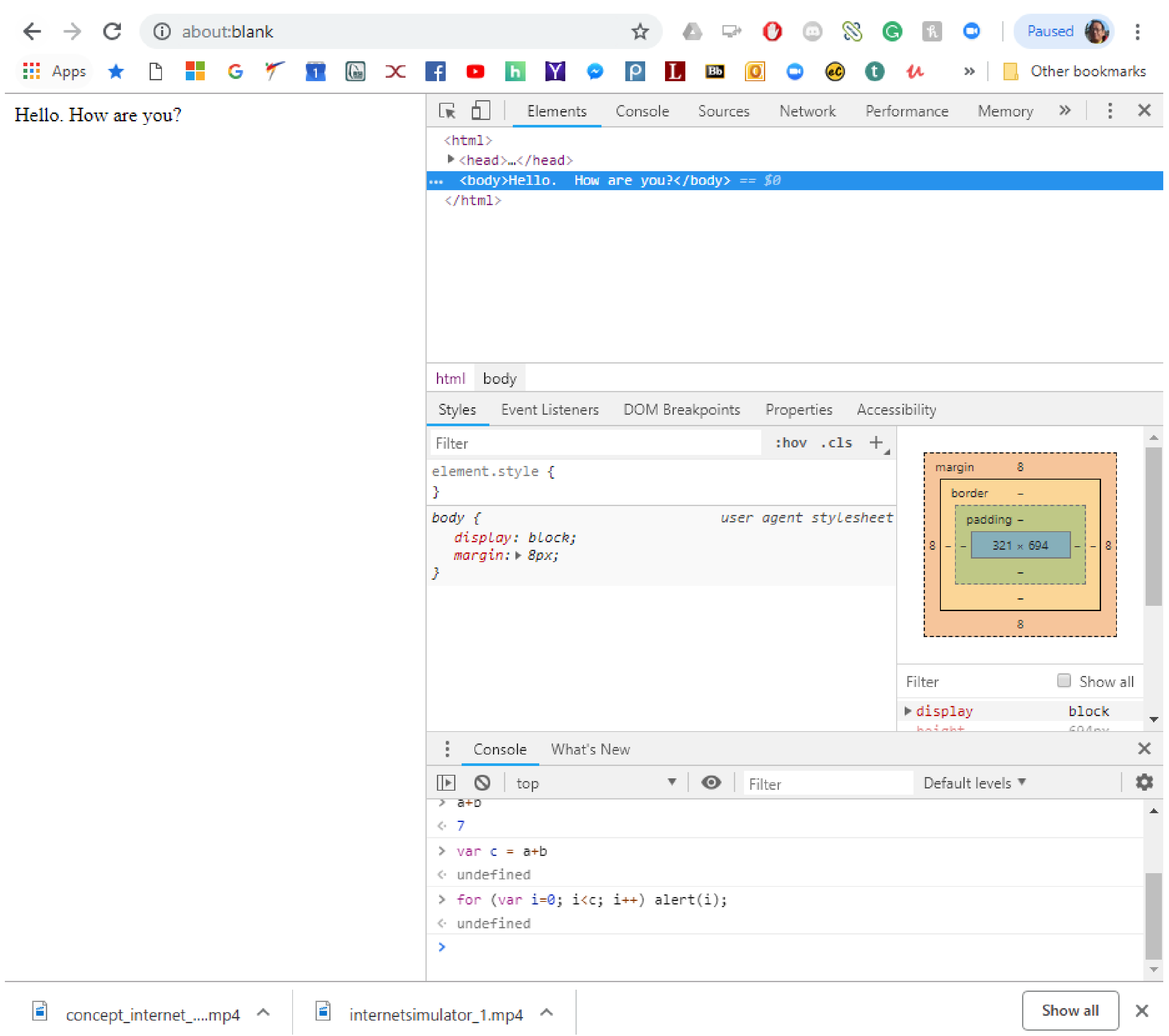
Google Chrome JavaScript Console Mode

about:blank page as console mode and html editor



Hello. How are you?

```
> var a = 3
< undefined
> var b = 4
< undefined
> a+b
< 7
> var c = a+b
< undefined
> for (var i=0; i<c; i++) alert(i);
< undefined
>
```



Hello. How are you?

```
<html>
▶<head>…</head>
… <body>Hello.  How are you?</body> == $0
</html>
```

```
element.style {
}
body {                                    user agent stylesheet
    display: block;
    margin: ▶ 8px;
}
```

```
▶ display        block
```

```
> a+b
< 7
> var c = a+b
< undefined
> for (var i=0; i<c; i++) alert(i);
< undefined
>
```

Basic JavaScript Language

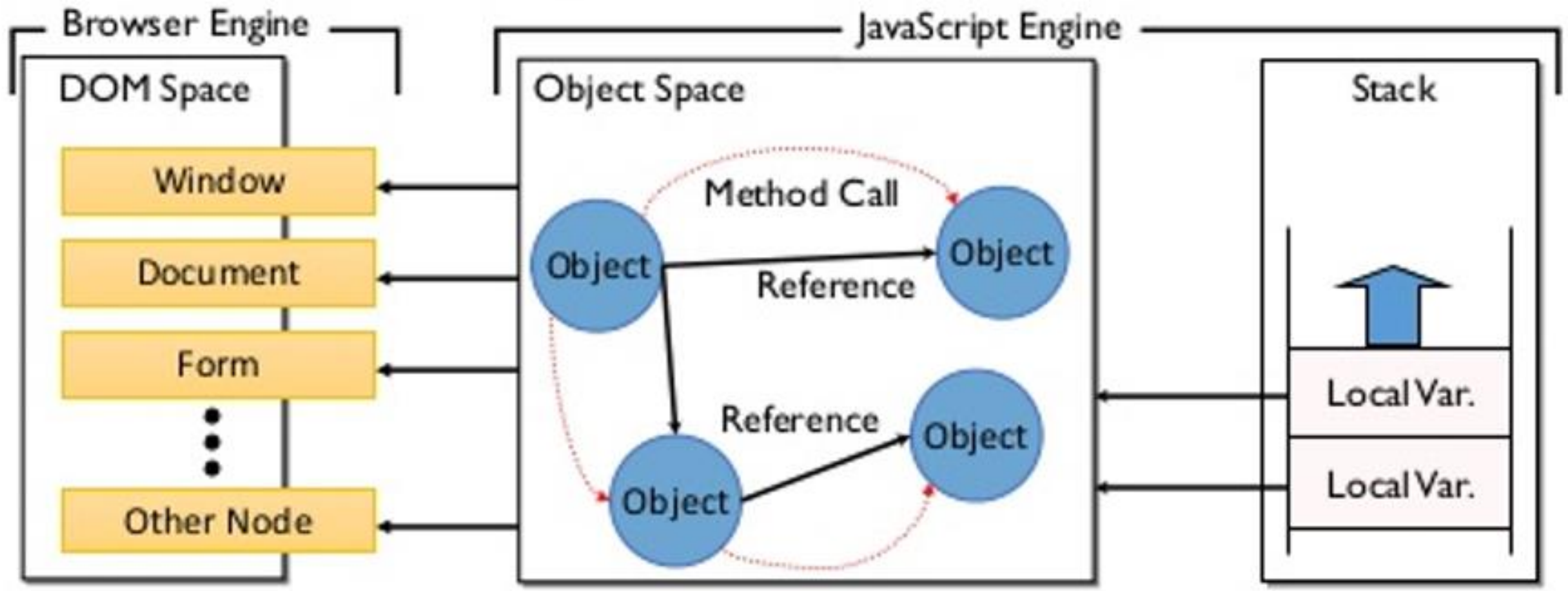
LECTURE 2

JavaScript

- Interpreter Based Scripting Language
- Everything is an object.
- No data type (Type Inference)
- Single Threading
- Event-Driven Programming

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



What is Javascript?

- A *scripting* programming language
  - Cannot be run without a browser
  - Embedded in most web browsers
- A way to create web pages that respond dynamically to user action
- A tool for server-side development

JavaScript

High-Level vs. Low Level

Compiled vs. Interpreted

Structured vs. Object Oriented

Scripting vs. Programming

What can JavaScript do?

- It is a full programming language
  - API (application programming interface) is specific to working with browsers
- Restrictions:
  - Security-based limitations
    - No networking
    - No access to local file system
  - Limited UI toolkit and graphics
    - (This is changing with HTML5)

What can JavaScript do?

- Benefits:
  - Close integration with the browser
    - Access the webpage and all elements within
    - Adjust or create HTML
    - Open and resize browser windows
    - Run animations, play sounds

The Structure of a JavaScript Program

LECTURE 3

Where do scripts go?

- In the HMTL page
- Like styles, can be **external**, **internal**, or **inline**
  - Use these for different situations

Body example

Demo Program: bodyexample.html

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("This message written by JavaScript");
</script>
</body>
</html>
```

Use of Document Write

- Use document as output terminal, you may debug JavaScript
- Until all JavaScript code is code, then, we may optimize the html/css to make the page more attractive.

Internal example

```
<html>
<head>
<script type="text/javascript">
function message()
{
  alert("This alert was called with the onload event");
}
</script>
</head>
<body onload="message()">
</body>
</html
```

Use of Alert

- Use alert as printf.
- Alert can be commented out when the development is finished.

External example

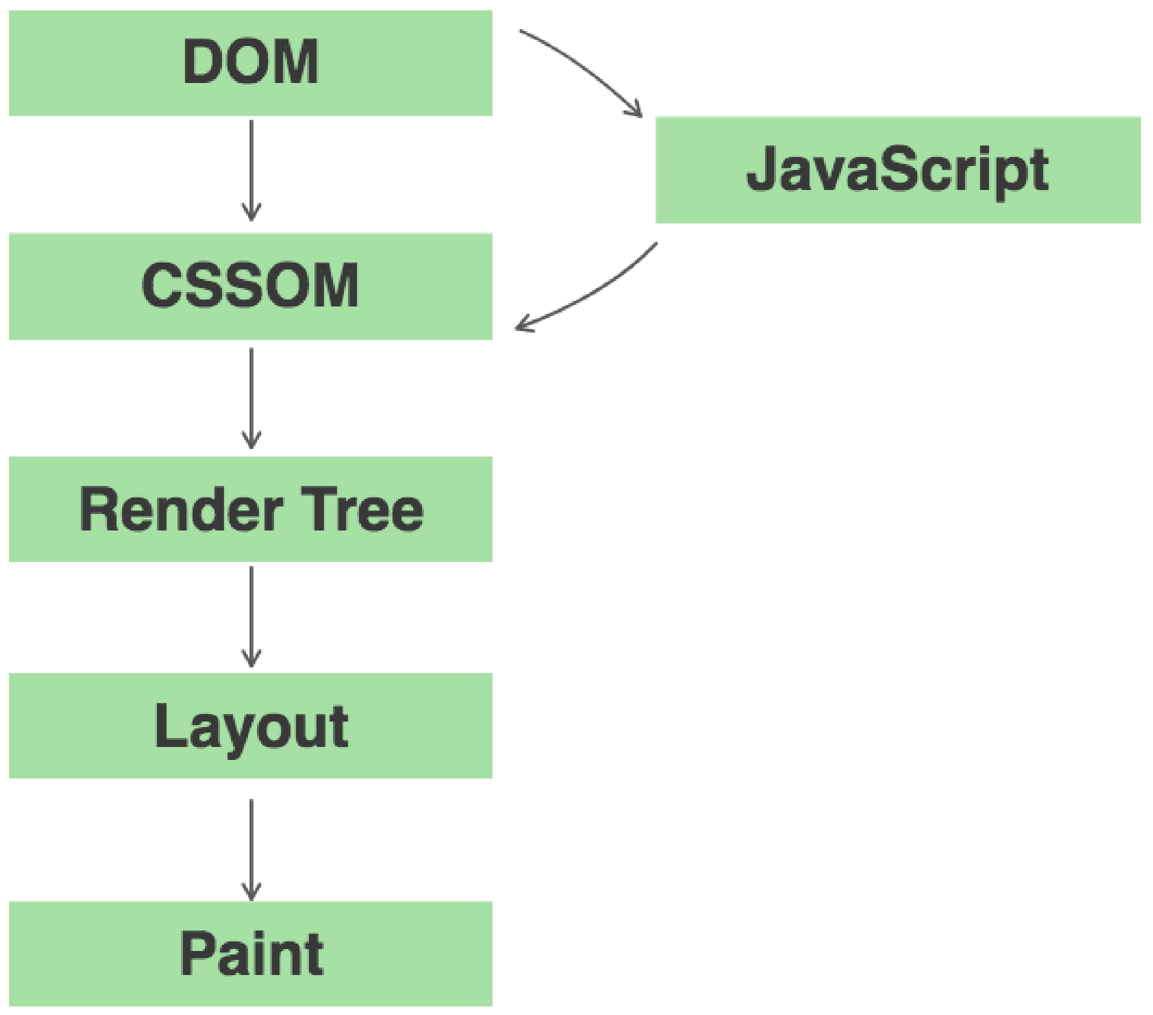
```
<html>
<head>
 <script type="text/javascript" src="xyz.js"></script>
</head>
<body>
</body>
</html>
```

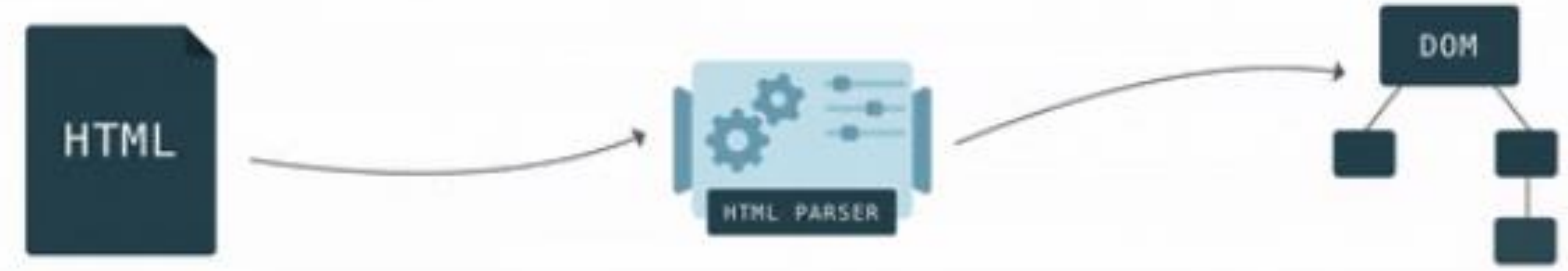
Embedded JavaScript

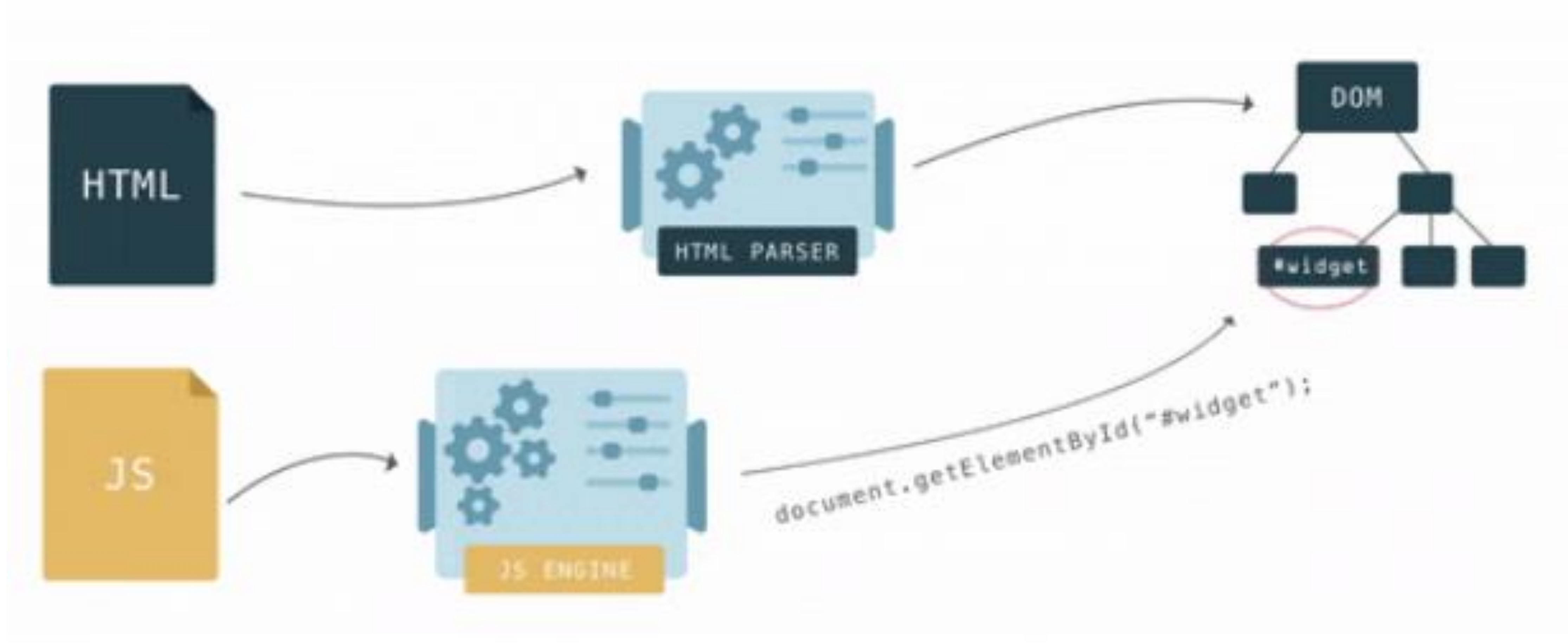
- JavaScript Library
- JavaScript API (processing.js and other API)
- Put the library at proper location.

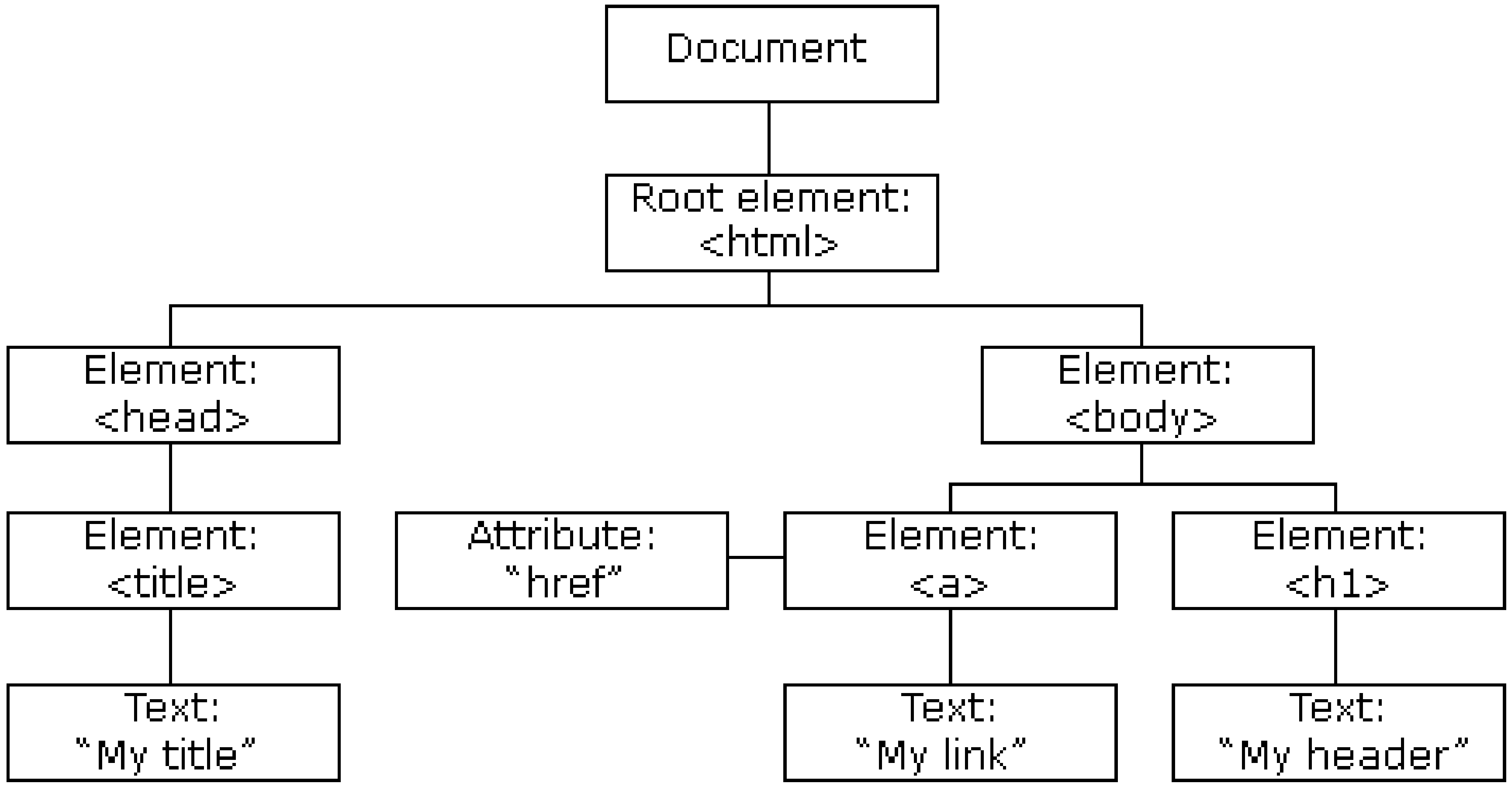
Execution of HTML/CSS/ JavaScript

LECTURE 3


DOM
CSSOM
Render Tree
Layout
Paint
JavaScript


HTML
HTML PARSER
DOM

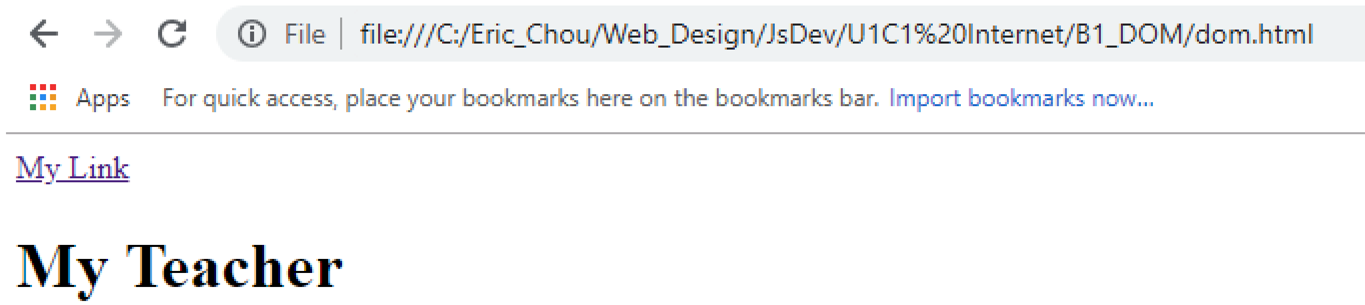

HTML
HTML PARSER
DOM
#widget
JS
JS ENGINE
document.getElementById("#widget");


Document
Root element: <html>
Element: <head>
Element: <body>
Element: <title>
Attribute: “href”
Element: <a>
Element: <h1>
Text: “My title”
Text: “My link”
Text: “My header”

DOM and html

Demo Program: dom.html

```
<!DOCTYPE html>
<html>
<head><title>My Title</title></head>
<body>
<a href="http://www.eCodeHacker.com">My Link</a>
<h1>My Teacher</h1>
</body>
</html>
```



File | file:///C:/Eric_Chou/Web_Design/JsDev/U1C1%20Internet/B1_DOM/dom.html

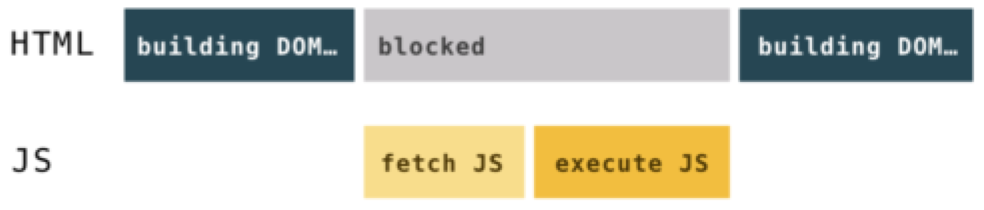
Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

My Link

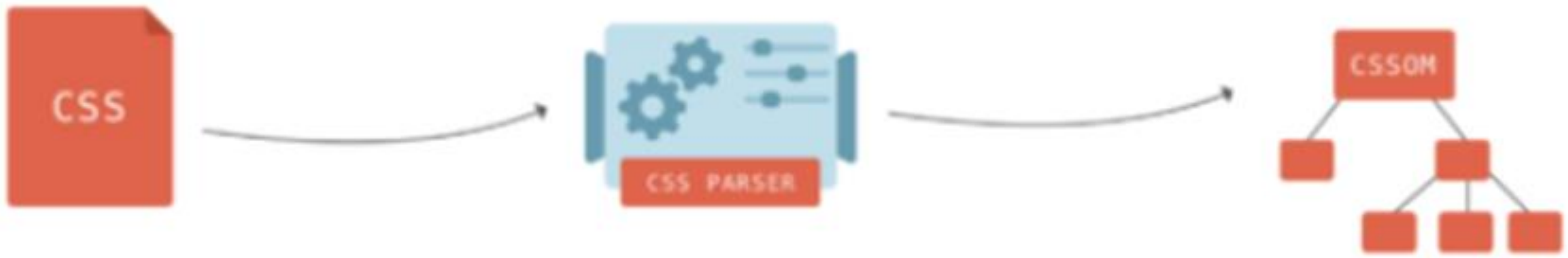
My Teacher

The history of the **<script>** tag

- As the browser is constructing the DOM, if it comes across a **<script>...</script>** tag in the HTML, it must execute it right away. If the script is external, it has to download the script first.
- Back in the old days, in order to execute a script, parsing had to be paused. It would only start up again after the JavaScript engine had executed code from a script.
- Why did the parsing have to stop? Well, scripts can change both the HTML and its product—the DOM. Scripts can change the DOM structure by adding nodes with **document.createElement()**. To change the HTML, scripts can add content with the notorious **document.write()** function. It's notorious because it can change the HTML in ways that can affect further parsing.

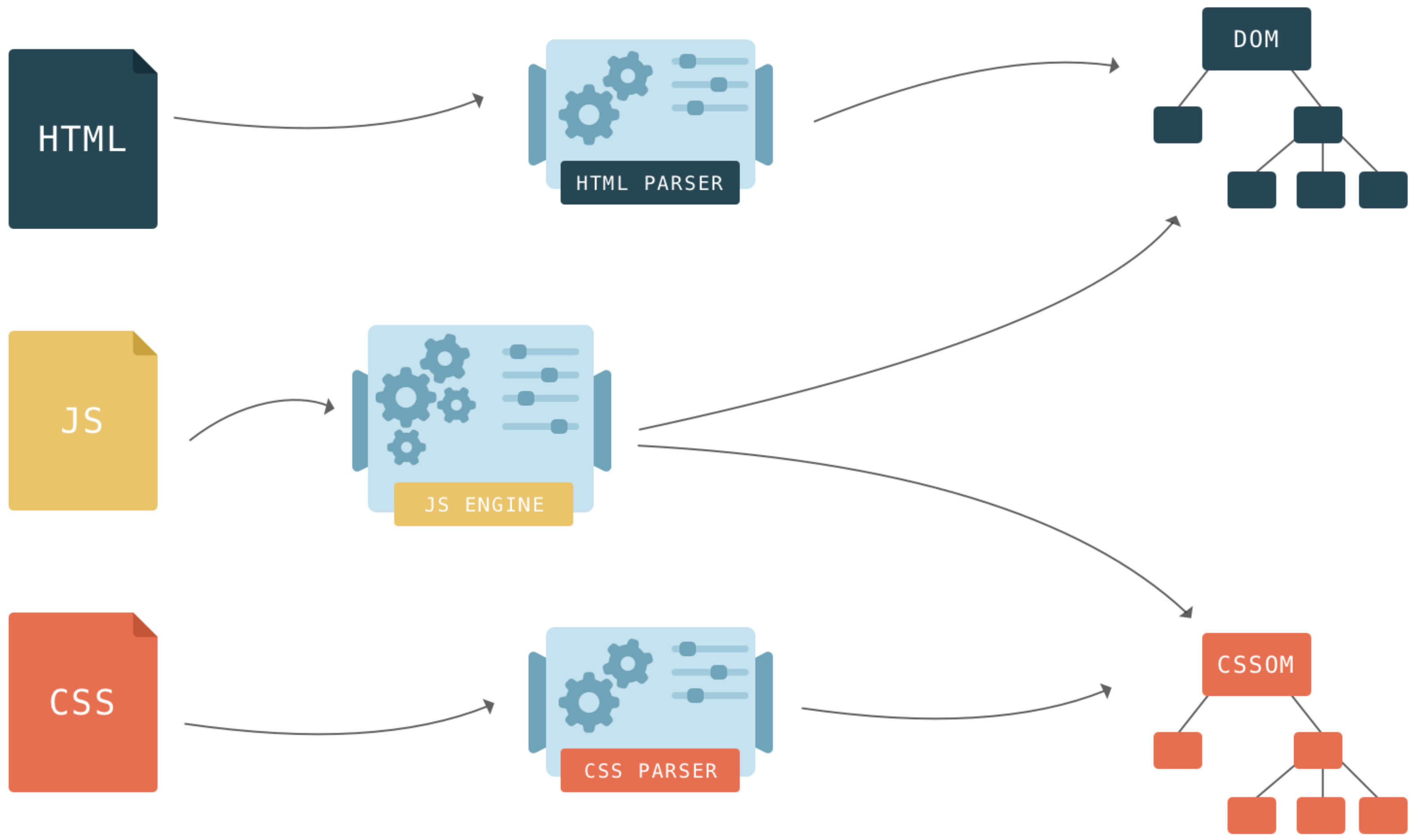


Parser blocking script


CSS
CSS PARSER
CSSOM

What about CSS?

- JavaScript blocks parsing because it can modify the document. CSS can’t modify the document, so it seems like there is no reason for it to block parsing, right?
- However, what if a script asks for style information that hasn’t been parsed yet? The browser doesn’t know what the script is about to execute—it may ask for something like the DOM node’s background-color which depends on the style sheet, or it may expect to access the **CSSOM** directly.

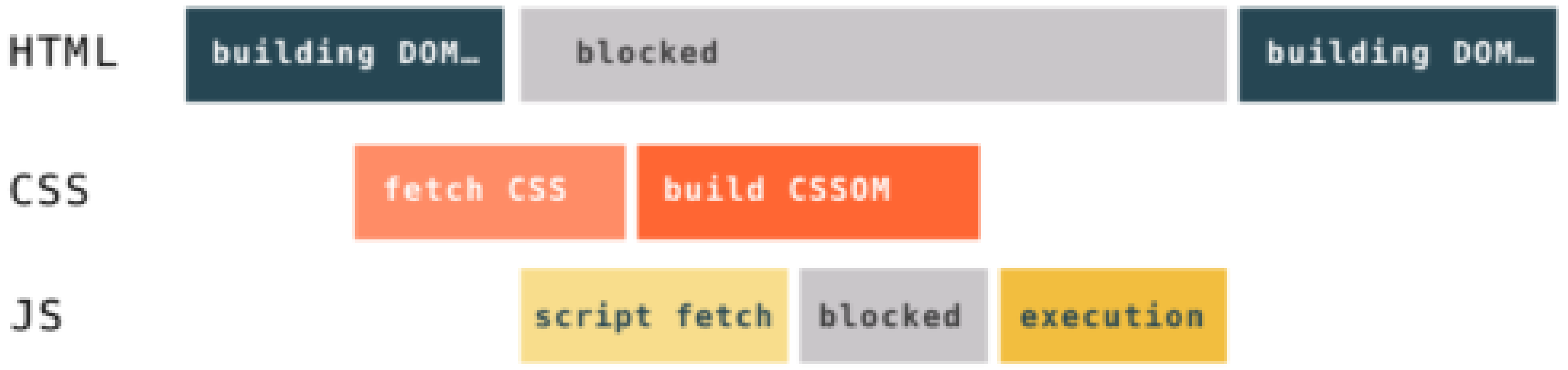

HTML
HTML PARSER
DOM
JS
JS ENGINE
CSS
CSS PARSER
CSSOM

Parsing CSS

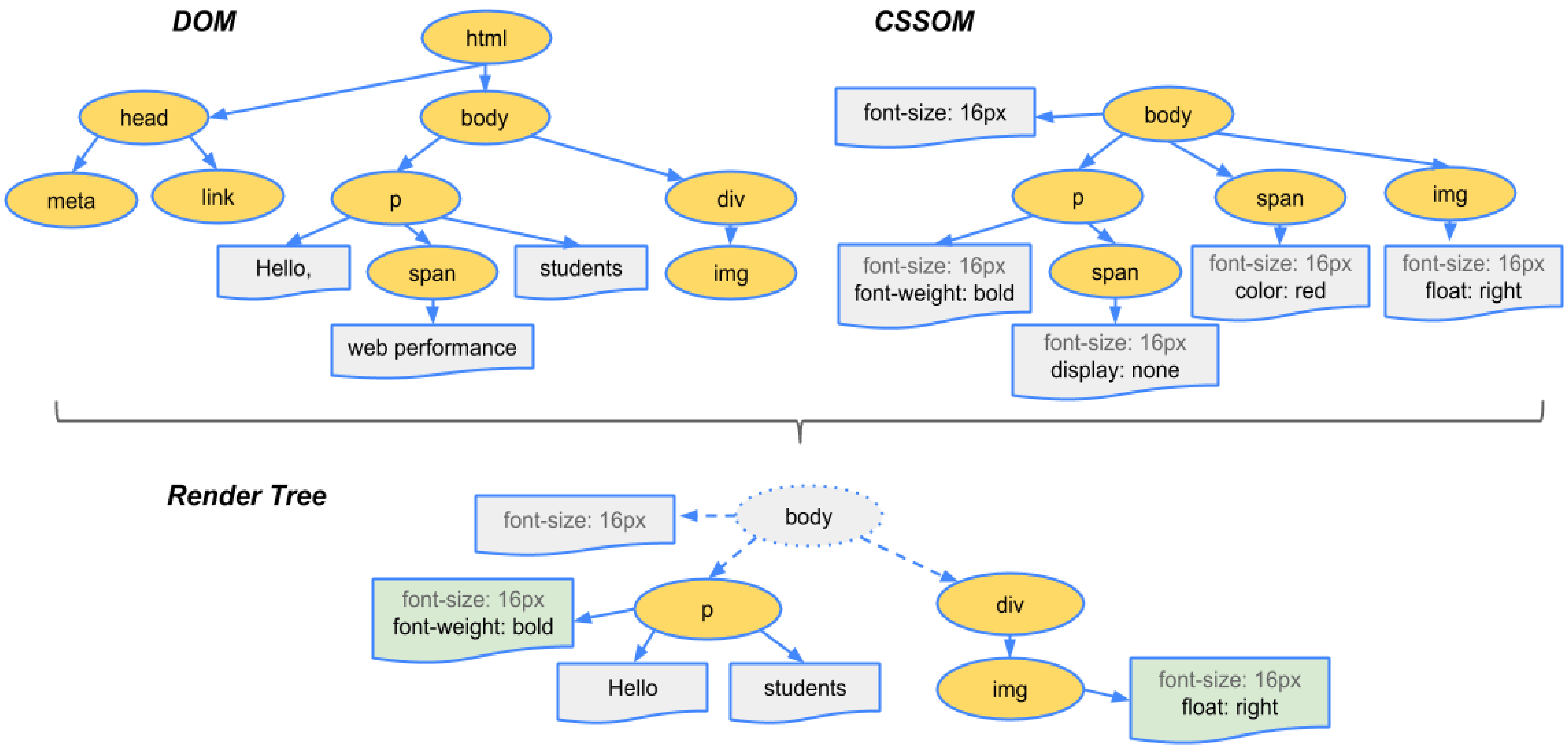
- Because of this, **CSS** may block parsing depending on the order of external style sheets and scripts in the document. If there are external style sheets placed before scripts in the document, the construction of **DOM** and **CSSOM** objects can interfere with each other.
- When the parser gets to a script tag, **DOM** construction cannot proceed until the JavaScript finishes executing, and the JavaScript cannot be executed until the **CSS** is downloaded, parsed, and the **CSSOM** is available.

Parsing CSS

- Another thing to keep in mind is that even if the CSS doesn’t block DOM construction, it blocks rendering. The browser won’t display anything until it has both the DOM and the CSSOM.
- This is because pages without CSS are often unusable. If a browser showed you a messy page without CSS, then a few moments later snapped into a styled page, the shifting content and sudden visual changes would make a turbulent user experience.



Parser blocking CSS


DOM
html
head
body
meta
link
p
div
Hello,
span
students
img
web performance
CSSOM
font-size: 16px
body
p
span
img
font-size: 16px
font-weight: bold
span
font-size: 16px
color: red
font-size: 16px
float: right
font-size: 16px
display: none
Render Tree
font-size: 16px
body
font-size: 16px
font-weight: bold
p
div
Hello
students
img
font-size: 16px
float: right

Internal CSS <style> tag

- Style tag and placed at many locations.
- Each <style></style> can include css script in the html section <head> or <body>

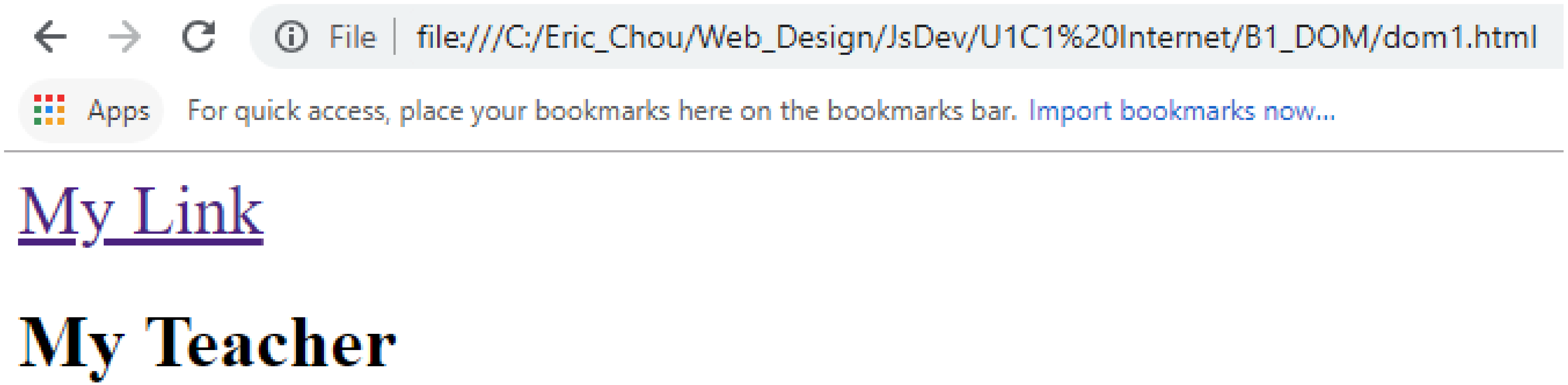
DOM and html

Demo Program: dom1.html

```
<!DOCTYPE html>
<html>
<head><title>My Title</title></head>
   <style>
      title{font-size:12px;  font-family:"sans-serif"; }
   </style>
<body>
   <style>
      a{font-size:30px;  font-family:"sans-serif"; }
   </style>
<a href="http://www.eCodeHacker.com">My Link</a>
<h1>My Teacher</h1>
</body>
</html>
```

DOM and html

Demo Program: dom1.html


File | file:///C:/Eric_Chou/Web_Design/JsDev/U1C1%20Internet/B1_DOM/dom1.html

Apps For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...

My Link

My Teacher

External Style Sheet

- With an external style sheet, you can change the look of an entire website by changing just one file!
- Each page must include a reference to the external style sheet file inside the **<link>** element. The **<link>** element goes inside the **<head>** section:

Example

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

DOM and html

Demo Program: dom2.html

dom2.html

```html
<!DOCTYPE html>
<html>
<head><title>My Title</title></head>
   <link rel="stylesheet" href="mystyle.css">
<body>
<a href="http://www.eCodeHacker.com">My Link</a>
<h1>My Teacher</h1>
</body>
</html>
```

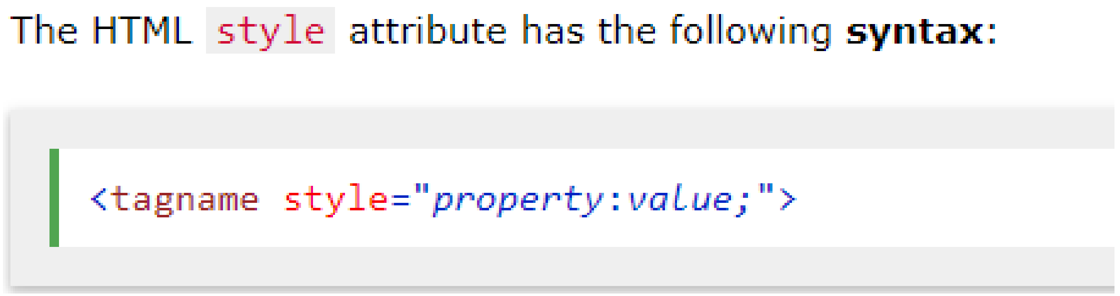
mystyle.css

```css
body { background-color: #f2f218; }
```

The HTML Style Attribute

Inline (per-element) Style Properties

- Setting the style of an HTML element, can be done with the style attribute.
- The HTML style attribute has the following syntax:



The HTML `style` attribute has the following **syntax**:

```
<tagname style="property:value;">
```

Example:

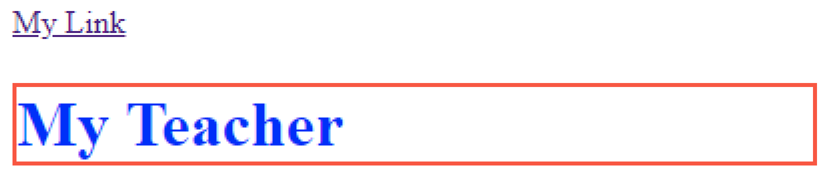
<h1 style="color: blue; border:2px solid Tomato;">My Teacher</h1>

DOM and html

Demo Program: dom3.html

dom3.html

```
<!DOCTYPE html>
<html>
<head><title>My Title</title></head>
<body>
<a href="http://www.eCodeHacker.com">My Link</a>
<h1 style="color: blue; border:2px solid Tomato;">My Teacher</h1>
</body>
</html>
```


My Link

My Teacher

First Simple JavaScript Program

LECTURE 4

Use JavaScript as other programming language:

Just For Computing Purpose

- Use internal <script></script> and put the script on top of body.
- Use only document.write() as printf function to deposit all output to the document.

Write to Page (Dynamic Document)

Demo Program: cats.html and cats2.html

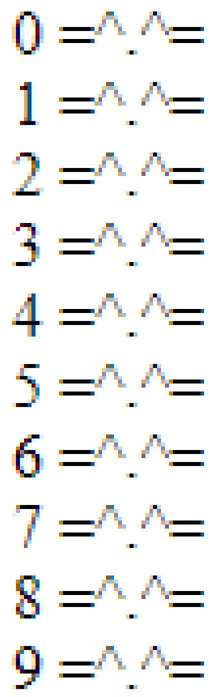
```
<html>
<head>
<script>
// Draw as many cats as you want!
var drawCats = function (howManyTimes) {
for (var i = 0; i < howManyTimes; i++) {
 document.write(i + " =^.^= <br>");
}
};
drawCats(10);
</script>
</head>
<body>

</body>
</html>
```

```
<html>
<head>

</head>
<body>
<script>
// Draw as many cats as you want!
var drawCats = function (howManyTimes) {
for (var i = 0; i < howManyTimes; i++) {
 document.write(i + " =^.^= <br>");
}
};
drawCats(10);
</script>
</body>
</html>
```

Execution Results:



```
0 =^.^=
1 =^.^=
2 =^.^=
3 =^.^=
4 =^.^=
5 =^.^=
6 =^.^=
7 =^.^=
8 =^.^=
9 =^.^=
```

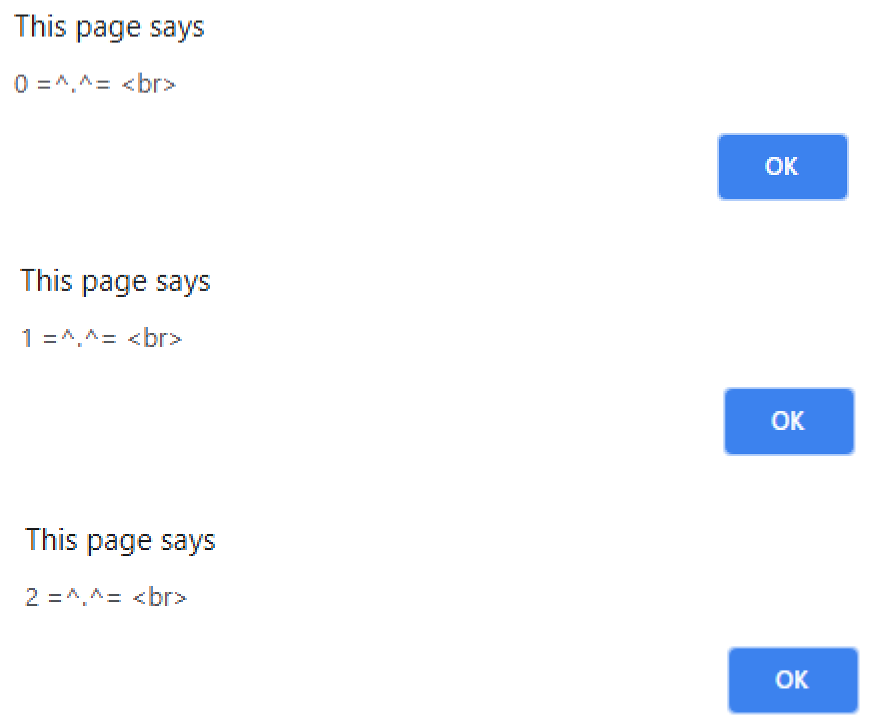
Trouble: Modify the page content

Alert

Demo Program: cats3.html

```
<html>
<head>

</head>
<body>
<script>
// Draw as many cats as you want!
var drawCats = function (howManyTimes) {
for (var i = 0; i < howManyTimes; i++) {
 alert(i + " =^.^= <br>");
}
};
drawCats(3);
</script>
</body>
</html>
```



This page says

```
0 =^.^= <br>
```

This page says

```
1 =^.^= <br>
```

This page says

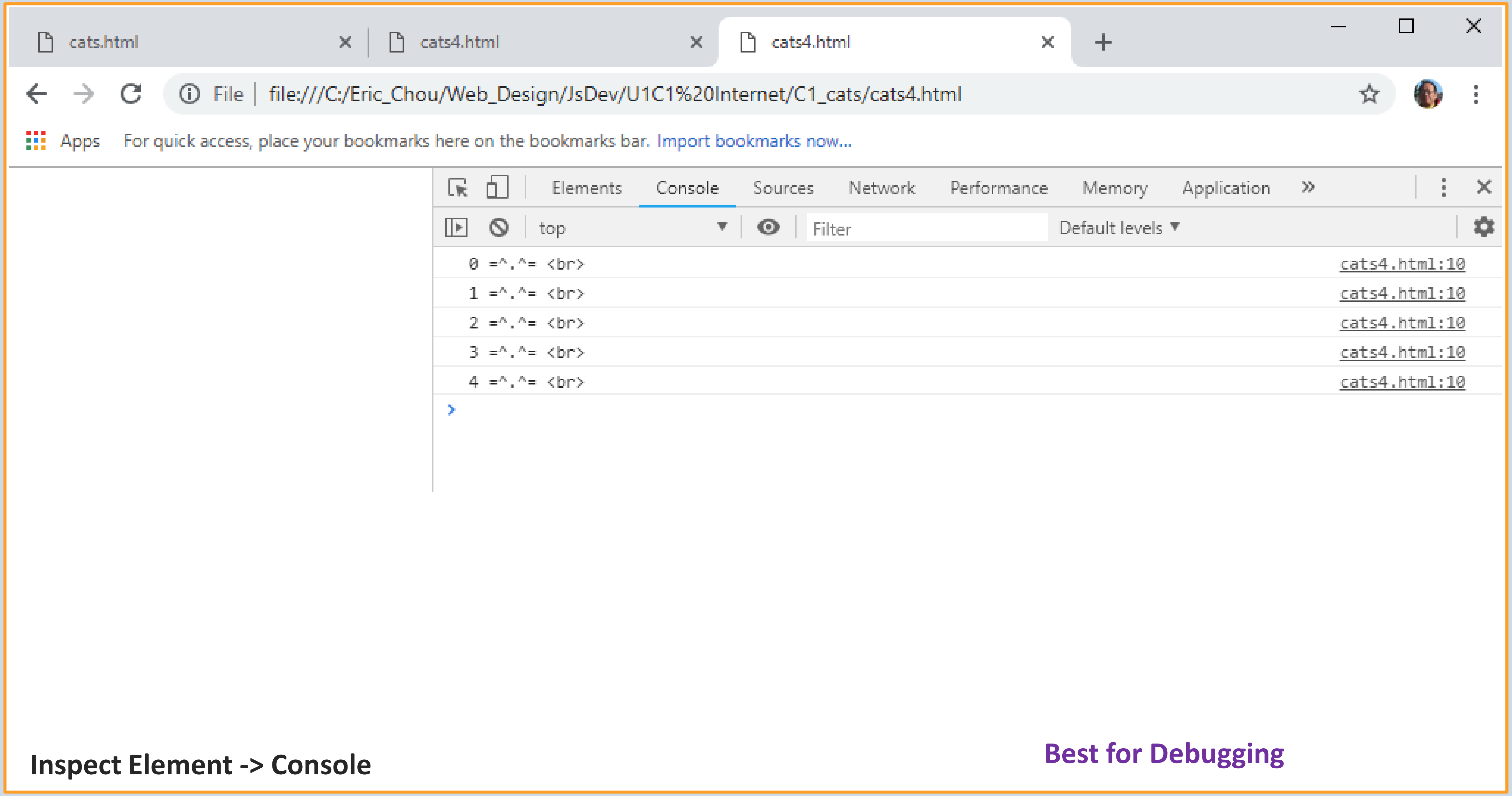
```
2 =^.^= <br>
```

Trouble: Busy alerting

Console Log

Demo Program: cats4.html

```
<html>
<head>
</head>
<body>
<script>
// Draw as many cats as you want!
var drawCats = function (howManyTimes) {
for (var i = 0; i < howManyTimes; i++) {
 console.log(i + " =^.^= <br>");
}
};
drawCats(5);
</script>
</body>
</html>
```


cats.html
cats4.html
cats4.html
File
file:///C:/Eric_Chou/Web_Design/JsDev/U1C1%20Internet/C1_cats/cats4.html
Apps
For quick access, place your bookmarks here on the bookmarks bar. Import bookmarks now...
Elements
Console
Sources
Network
Performance
Memory
Application
top
Filter
Default levels

```
0 =^.^= <br>                                   cats4.html:10
1 =^.^= <br>                                   cats4.html:10
2 =^.^= <br>                                   cats4.html:10
3 =^.^= <br>                                   cats4.html:10
4 =^.^= <br>                                   cats4.html:10
>
```

Inspect Element -> Console

Best for Debugging

Syntax

- Our program includes lots of symbols, including parentheses **()**, semicolons **;**, curly brackets **{}**, plus signs **+**, and a few words that might seem mysterious at first (like var and console.log). These are all part of JavaScript’s syntax — that is, JavaScript’s rules for how to combine symbols and words to create working programs.
- When you’re learning a new programming language, one of the trickiest parts is getting used to the rules for how to write different kinds of instructions to the computer. When you’re first starting out, it’s easy to forget when to include parentheses, or to mix up the order in which you need to include certain values. But as you practice, you’ll start to get the hang of it.
- In this course, we’ll go slow and steady, introducing new syntax little by little so that you can build increasingly powerful programs.

Comments

// comment out the rest of the line

/*comment out a code block */