

---

# COMP 431/531: Web Development

## Lecture 1: HTML

Mack Joyner ([mjoyner@rice.edu](mailto:mjoyner@rice.edu))

<https://www.clear.rice.edu/comp431>



# Goal: Multi-User Web Application

---

- Full Stack Development
  - Front-End (HTML, CSS, JavaScript, Framework)
  - Back-End (Web servers, Databases, Security)
- Software Engineering Design
  - Test Driven Development (Unit Testing, End-to-end testing)
  - Good design principles (e.g. encapsulation)



# Final Web Application

# Ricebook

Email or Phone


Password

☐ Keep me logged in


[Forgot your password?](#)

[Log In](#)


## Connect with friends and the world around you on Ricebook



**See photos and updates** from friends in News Feed.



**Share what's new** in your life on your Timeline.



**Find more** of what you're looking for with Facebook Search.

## Sign Up

It's free and always will be.

**Birthday**

Month ▼

Day ▼

Year ▼

Why do I need to provide my birthday?

☐ Female ☐ Male

By clicking Sign Up, you agree to our [Terms](#) and that you have read our [Data Policy](#), including our [Cookie Use](#).

[Sign Up](#)

[Create a Page for a celebrity, band or business.](#)

8/19/18

Fall 2017

3



# Assumptions/Expectations

---

- No prior HTML/CSS/JavaScript experience
- Motivated to learn about web development and design



# Administration

---

Instructor: Mack Joyner

TAs: Rex Le, Jingbin Qian, Prashithaa Balaji

Office Hours: See course website ([www.clear.rice.edu/comp431](http://www.clear.rice.edu/comp431))

Piazza: <https://piazza.com/class/mdz1yc4pqh44bl>

Github classroom (hw1): <https://classroom.github.com/a/gsvZWWoo>

Grades: Canvas (has slides)

Acknowledgment: *Special thanks to Scott Pollack and Leo Elworth*



# Git Classroom

---

- We'll use GitHub classroom
  - Create a GitHub account if you don't have one
  - Click on GitHub repo link (slide 5)
  - Link git id (name) to git repo (first repo)
- Submit code often
  - Recommend using an IDE (IntelliJ - ultimate edition)
  - `git add`
  - `git commit -m "description of change"`
  - `git push origin main` (don't forget)



# Grade Policies

---

## Course Rubric

- Homework (not including final web app, with 531 presentation) 60%
- Final Web App 20%
- Quizzes 10% (on-line quizzes on Canvas)
- Class Participation 5% (in-class worksheets)
- Peer Reviews 5%



# In-Class Exercises

---

- We'll have 1 for each class lecture
- Based on material presented in the lecture
- Grade rubric
  - 1 = submit before end of class
  - 0.5 = submit before beginning on next class
- Submit in-class exercises in Canvas





# Assignments

---

1. Simple HTML page
2. Dynamic Page
3. JavaScript Game
4. Draft Front-End
5. Front-End Application
6. Draft Back-End
7. Final Full Web App



# Late Assignment Policy

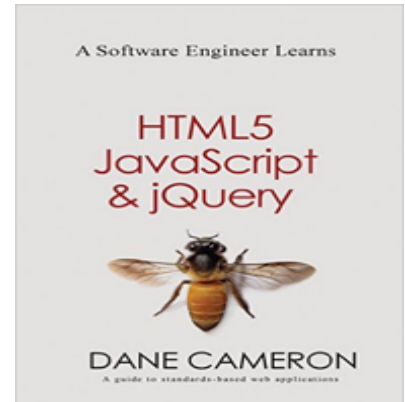
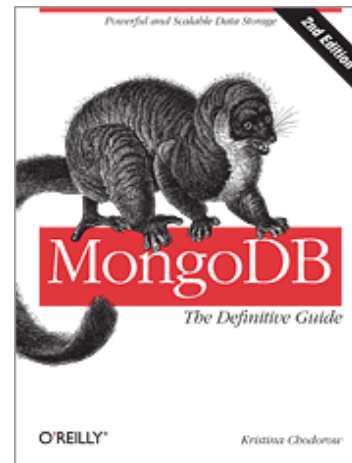
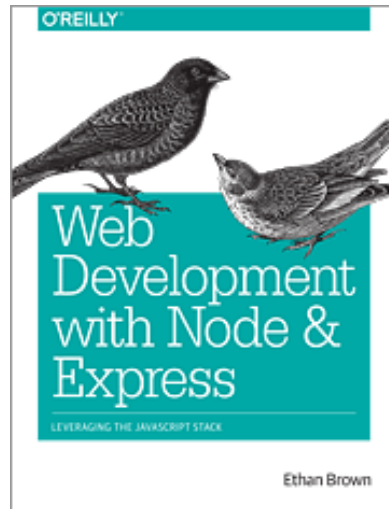
---

- You will have 3 slip days
  - When you use a slip day, you'll have an additional 24 hours to turn in assignment
  - Use README to indicate when you are using a slip day
- No late submissions (other than those using slip days) will be accepted



# Textbooks

- No required texts
- You may find these helpful...



# Editors

---

Atom



IntelliJ



Sublime



# Advice

---

- Start assignments early
- Ask for help often (check Piazza for questions/answers)
  - most of the TAs all went through course last semester
- Finish homework assignments
  - assignments build on each other (challenge to catch-up)
- Use people outside the course to test your web application



# History of HTML

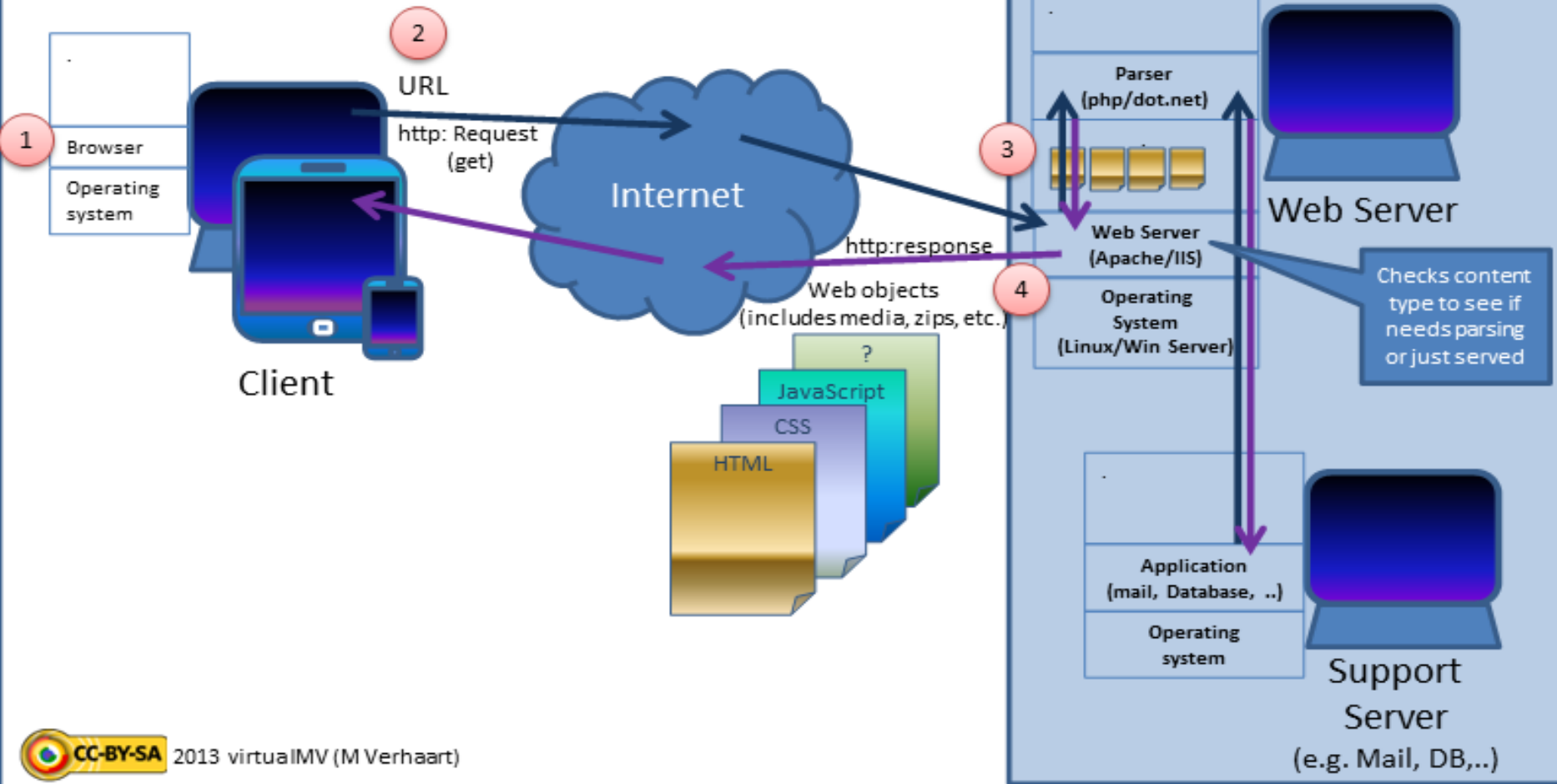
---

- 1989 - World Wide Web
- 1991 - HTML tags formally introduced
- 1993 - HTML specification [ Netscape]
- 1995 - HTML 2.0 [ Internet Explorer ]
- 1997 - HTML 4.0
- 2000's - Browser wars [Firefox, Safari, Chrome, ...]
- 2014 - HTML5



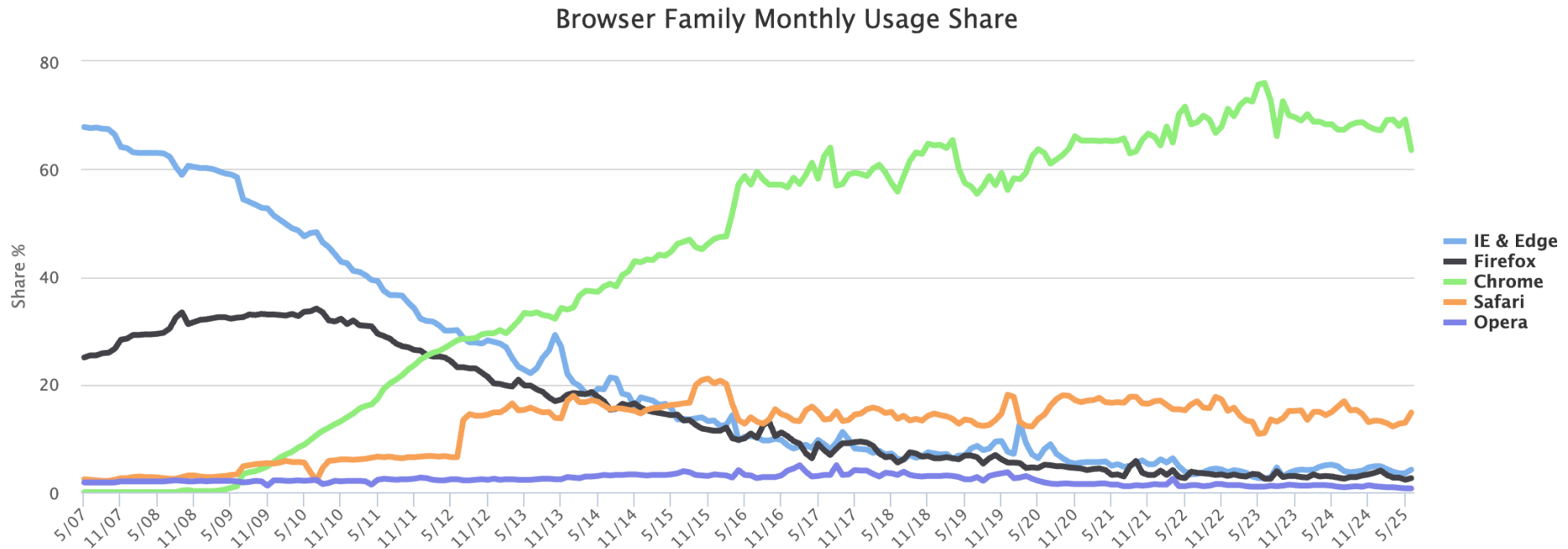
# World Wide Web Ecosystem

## Client-server ecosystem



# Web Browser Usage Trends

<https://www.w3counter.com/trends>





# HTML 5

- First line of document defines document type

## Current HTML 5

```
<!DOCTYPE html>
<html lang="en-US">
<!-- language is helpful for screen
      readers and search engines -->
```

- HTML 4.01 Strict (old way)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

- HTML body by default
  - Could omit `<html>`, `<head>`, `<body>` tags
  - We won't omit the tags, write "complete" HTML

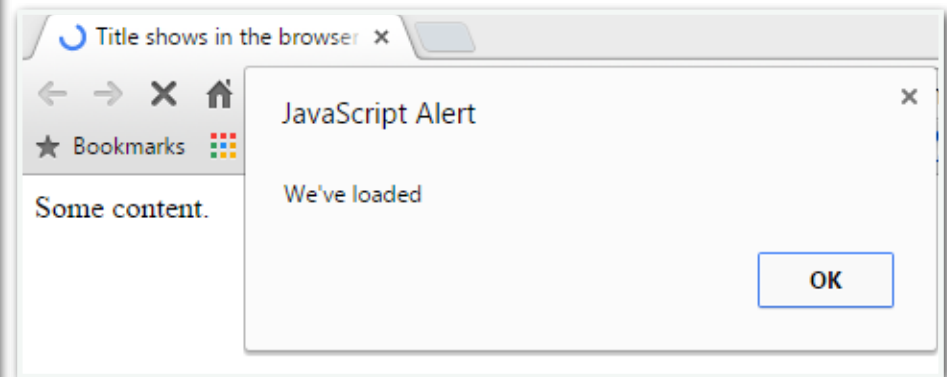
## Initial file created in IntelliJ

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Title</title>
6  </head>
7  <body>
8
9  </body>
10 </html>
```



# Basic structure of an HTML page

```
1 <!DOCTYPE html>
2 <html lang="en-US">
3 <head>
24 </head>
25
26 <body onload="doSomething()">
27 Some content.
28 </body>
29 </html>
30
31 <script>
32 function doSomething() {
33     alert("We've loaded");
34 }
35 </script>
36
```



W3Schools is a good resource ([www.w3schools.com](http://www.w3schools.com))

# jsfiddle.net

The screenshot shows the jsfiddle.net web application interface. At the top, there is a navigation bar with a cloud icon, a 'Run' button, a 'Save' button, a 'Collaborate' button, a 'Change' button, and a notification 'Fiddle listing shows latest version'. On the right side of the navigation bar are links for 'Settings' and 'Sign in'.

The main interface is divided into several sections:

- Fiddle meta**: A sidebar on the left containing 'Untitled fiddle', 'No description', a 'Private fiddle' toggle (currently off), 'Groups', 'Resources' (with 'URL' and 'cdnjs' buttons), 'Async requests', and 'Other (links, license)'.
- HTML**: A central editor area showing a code editor with the following HTML code:

```
1 <html>
2 <h1>
3   Welcome to COMP 431/531
4 </h1>
5 </html>
```
- CSS**: A panel on the right showing a single line of CSS code.
- JavaScript + No-Library (pure JS)**: A panel at the bottom left showing a single line of JavaScript code.
- Preview**: A large area on the right showing the rendered output of the code, which is 'Welcome to COMP 431/531'.

At the bottom left, there is a DigitalOcean advertisement with the text: 'Provision MongoDB clusters in minutes. Get \$100 free credit.'



# Markup

```
25 <body>
26
27 <h1>Sample Page</h1>
28
29 <p>Paragraphs are separated as "blocks" of text.</p>
30 <p>Simple markup such as <b>bold face</b> or <em>
   emphasized</em> text.
31
32 Note that white space is ignored.
33
34 If we want to "control" the layout, we should use
   styles. If we need some small control we can
   insert a line break. For example,</p>
35 
36 <br/>
37 
38 <p>Hyperlinks are contained within <a name="anchor">
   anchors</a> which can be referenced within a page
   using another <a href="#anchor" title="go here">
   anchor</a>. In general we want links to go to
   other pages, such as <a href="https://developer.
   mozilla.org">this</a> one.</p>
39 </body>
40 </html>
```

## Sample Page

Paragraphs are separated as "blocks" of text.

Simple markup such as **bold face** or *emphasized* text. Note that white space is ignored. If we want to "control" the layout, we should use styles. If we need some small control we can insert a line break. For example,

Image Missing



RICE®

Rice Logo

Hyperlinks are contained within anchors which can be referenced within a page using another [anchor](#). In general we want links to go to other pages, such as [this](https://developer.mozilla.org) or [go here](#).



# Inline vs Block

---

- We markup to control layout.
  - Sometimes you want content “blocks” separate from other blocks.
  - Sometimes you want to inline layout or style
- Inline:
  - `<b>`
  - `<em>`
  - `<br>`
  - `<span>`
- Block:
  - `<p>`
  - `<div>`



# More HTML Fundamentals

```
40 <hr/>
41 <!-- this is a comment -->
42 <table border="1">
43   <thead>
44     <tr><th>Centered by Default</th><th>Column Header</th></tr>
45   </thead>
46   <tbody>
47     <tr><td>Row</td><td>Content</td></tr>
48     <tr><td colspan="2">span two columns to format our table better</td></tr>
49     <tr><td>Lazy...</td></tr>
50     <tr>
51       <td>Here is an ordered list
52       <ol>
53         <li>first item</li>
54         <ol>
55           <li>nested item</li>
56         </ol>
57         <li>another item</li>
58       </ol>
59     </td>
60     <td>and an unordered list
61     <ul>
62       <li>first item</li>
63       <li>another item</li>
64     </ul>
65   </td>
66 </tbody>
67 </table>
```



# More HTML Fundamentals

Centered by Default	Column Header
Row	Content
span two columns to format our table better	
Lazy...	
Here is an ordered list <ol style="list-style-type: none"><li>1. first item<ol style="list-style-type: none"><li>1. nested item</li></ol></li><li>2. another item</li></ol>	and an unordered list <ul style="list-style-type: none"><li>• first item</li><li>• another item</li></ul>

We must escape `<` and `>` in addition to `&` and a few other character. Sometimes we want to control the amount of space " " in a cell in a table, we can do that with non-breaking spaces for example.

Pre-formatted text will appear as it is written exactly (we hope)

And it *renders* some HTML!

```
function example() { var foo = "bar"; console.log(foo); }
```



# Metadata (Tag)

---

- The `<meta>` tag provides page metadata:

- details about the page (e.g. author)
- data not displayed on page

- Always put `<meta>` tag inside `<head>` tag

- Pass metadata as key/value pairs

- attribute (e.g. name, content) defines type
- need content attribute if name attribute is defined

4  
5

```
<meta charset="UTF-8">  
<meta name="author" content="Mack Joyner">
```





# Metadata (Viewport)

---

- Set the viewport to change the page based on device size:

9

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

- width=device-width sets width of page based on device width
- initial-scale=1.0 sets the initial zoom level



# Some HTML Resources

---

- W3Schools (<http://www.w3schools.com/>)
- Mozilla Developer Network (<https://developer.mozilla.org/>)
- Validate your HTML (<http://validator.w3.org>)
- Real-time HTML rendering (<http://jsfiddle.net/bpenp4dq/>)

