CS 35L
Discussion 1A
Week 3

Web Application and More

Reminds

- Assignment 2 and 3 Extension by 1 Day
  - HW2: Due Oct 16 2022
  - HW3: Due Oct 21 2022
  - **11:55** pm UCLA Time
- Project Proposal and Initial Plan
  - Proposal: Due Oct 17 2022
  - Initial Plan: Due Oct 19 2022
  - Just the first version to describe your project plan - you could change it and resubmit at the end of the quarter

Last Week

0. **Programming Language**

1. **Lisp**

    a. Why Lisp
    b. Printing
    c. Arithmetic
    d. Variables and Data Type
    e. If Condition
    f. Loop
    g. Functions

2. **Python**

    a. Why Python
    b. Printing
    c. Variables and Data Type
    d. Conditions
    e. Loop
    f. Functions
    g. Module

Most of the Lisp examples in this slides are from http://ergoemacs.org/emacs/elisp_basics.html

Contents

0. **Programming Paradigms**
1. Web Application
2. HTML (DOM)
3. CSS
4. JavaScript
5. React

0. Programming Paradigm - What

- Programming paradigm is a way to **classify programming languages** based on their **features**.
  - How the code is organized
  - Style of syntax and grammar
- Programming paradigms are the **methodologies/strategies** that the programming languages follow when they are implemented.
- It is possible for one language to be classified into multiple paradigms.

Screenshots from WIKI:

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	February 20, 1991; 30 years ago[2]

C++

Logo endorsed by Standard C++

Paradigms	Multi-paradigm: procedural, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) /

JavaScript

Paradigm	event-driven, functional, imperative
Designed by	Brendan Eich of Netscape initially; others have also contributed to the ECMAScript standard
First appeared	December 4, 1995; 25 years ago[1]
Stable release	ECMAScript 2021 (12th edition)[2] / June 2021; 4 months ago
Preview release	ECMAScript 2022[3] / 22 July 2021; 2 months ago

0. Programming Paradigm - Why

- It's not required by this course
- But it could probably help you to learn faster
  - Summarize features of programming languages (pros and cons)
- When coding
  - Design first: how to solve this problem, what kind of building blocks are needed
  - Find the programming language that fit your needs the most
  - Figure out the details: Grammar/syntax/vocabulary of the selected programming language

Screenshots from WIKI:

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	February 20, 1991; 30 years ago[2]

C++

Logo endorsed by Standard C++

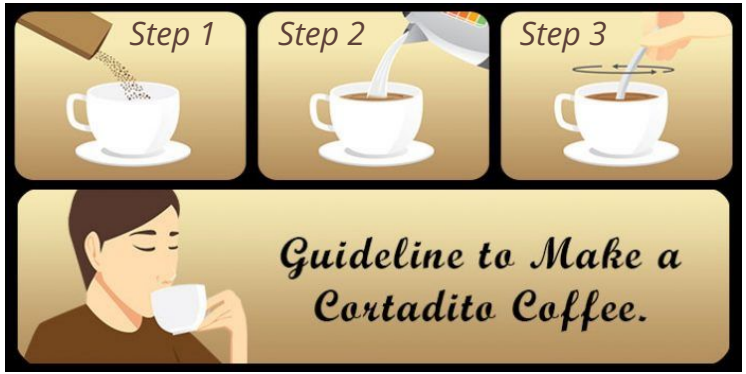
Paradigms	Multi-paradigm: procedural, functional, object-oriented, generic, modular
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) /

JavaScript

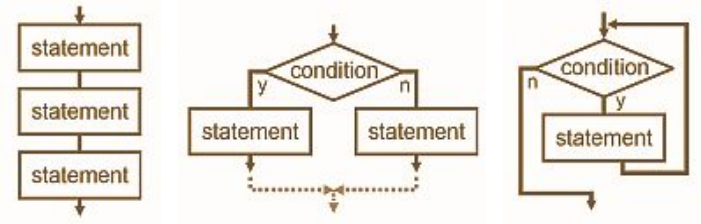
Paradigm	event-driven, functional, imperative
Designed by	Brendan Eich of Netscape initially; others have also contributed to the ECMAScript standard
First appeared	December 4, 1995; 25 years ago[1]
Stable release	ECMAScript 2021 (12th edition)[2] / June 2021; 4 months ago
Preview release	ECMAScript 2022[3] / 22 July 2021; 2 months ago

0. Programming Paradigm

- Common programming paradigms
- Imperative programming (How to do)
  - Consists of ***commands*** for the computer to perform
  - Changes a program's ***state*** with ***statements***
  - Focuses on describing ***how a program operates***
- Declarative programming (What to do)
  - Building the ***structure*** and ***elements*** of computer programs
  - Expresses the ***logic of a computation*** without describing its control flow
  - Focuses on the ***result*** (what need to be done) rather than the how it is produced



0. Programming Paradigm



- Imperative programming (How to do)

  Describing how a program operates

  - Procedural programming / Structured programming
    - The structured control flow: selection (if/then/else) and repetition (while and for), block structures, and subroutines
    - Procedural call (code reuse)

  - Object oriented programming
    - Everything is object! -- Solve real life problems
      - Data (fields, properties, attributes...)
      - Method (procedures...)
    - Class - define the data format, available procedures
    - Object - instance of classes
    - Encapsulation, inheritance, polymorphism ....

0. Programming Paradigm

- Declarative programming (What to do)

  Describing the logic of a computation

  - Functional programming
    - Programs are constructed by applying and composing functions
    - Function definitions are trees of expressions that each return a value
  - Logic programming
    - Largely based on formal logic
    - A set of sentences in logic form (express facts and rules about some problem domain)
  - Database programming
    - Based on data and its movement
    - Database program may provide file creation, data entry, update, query and reporting

Traditional Imperative Loop:

```
const numList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let result = 0;
for (let i = 0; i < numList.length; i++) {
  if (numList[i] % 2 === 0) {
    result += numList[i] * 10;
  }
}
```

Functional Programming with higher-order functions:

```
const result = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
               .filter(n => n % 2 === 0)
               .map(a => a * 10)
               .reduce((a, b) => a + b);
```

```
canfly(X) :- bird(X), not abnormal(X).
abnormal(X) :- wounded(X).
bird(john).
bird(mary).
wounded(john).
```

Goal

```
:- canfly(X).
```

Contents

0. Programming Paradigms
1. **Web Application**
2. HTML (DOM)
3. CSS
4. JavaScript
5. React

1. Web Application

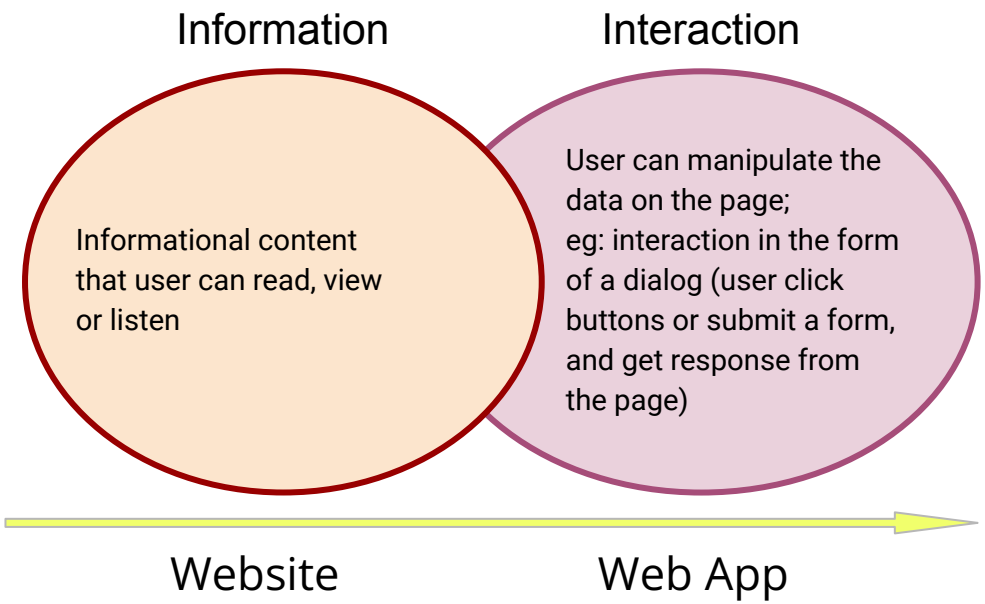
- There is a lot more that goes into a web application than what we can cover in this one-hour Discussion.
- This Discussion:
  - Concepts to start on assignment 3
- Project !?
  - More research !
  - More practice !!
- Resources:
  - ACM's Hack School: https://hack.uclaacm.com/archive/
  - Codecademy: https://www.codecademy.com/catalog

1. Web Application V.S. Website

- Common points:
  - Run in browsers, require access to the internet, have a front end and a back end written in the same programming languages ....
- Difference
  - In general: web app is more complex than website
  - Interactivity
  - Integration
  - Authentication

1. Web Application V.S. Website

- Interactivity



1. Web Application V.S. Website

- Integration
  - I.e., bring together different components to build a more comprehensive system
  - Developers can integrate web app and websites with other software
    - Mostly choose apps to integrate
    - Support more complex features, or require additional information from extra systems
- Authentication
  - I.e., enter user data to gain access to the site or the system
  - User accounts must be secured!
  - Most of the web apps require authentication
  - After logining in, web app can offer a broader scope of options than website
  - Eg: read news only v.s. leave a comment

1. Build a Web App

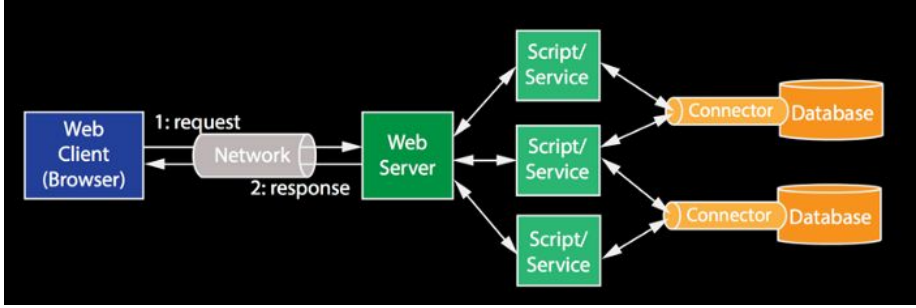


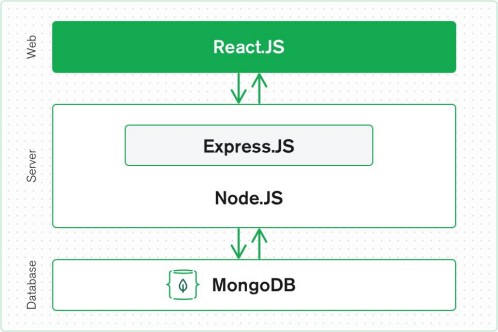
Image Credit: G.L. Heileman, Coursera MOOC, "Web Application Architectures"

- Front-end Layer (Blue):
  - Design: create the look and feel of the application
  - What the user would see in the client's web browser
    - HTML, CSS, Javascript
- Back-end Application Layer (Green):
  - Develope: services and presentation logic => deliver the response back to the user
- Back-end Database Layer (Orange):
  - Data: store the data, support a data-based application

- All together => a "Full Stack"
  - E.g.: MERN (***M***ongoDB, ***E***xpress, ***R***eact, ***N***ode)

MongoDB - document database;
Express(.js) - Node.js web framework;

React(.js) - a client-side JavaScript framework;
Node(.js) - the premier JavaScript web server;



Contents

0. Programming Paradigms
1. Web Application
2. **HTML (DOM)**
3. CSS
4. JavaScript
5. React

2. HTML - *H*yper*T*ext *M*arkup *L*anguage

- HTML
  - The foundation of all web pages
  - Give content to a web page and instructs web browsers on how to structure that content
    - Allows us to organize text, add images or videos to our web pages!
- Key Concepts
  - Element / Element content
    - The content of an HTML element is the information between the opening and closing tags of an element
    - `<h1> Hello world! </h1>`
  - Structure
    - HTML is organized into a tree structure
    - HTML elements can have parents, siblings, childrens, etc

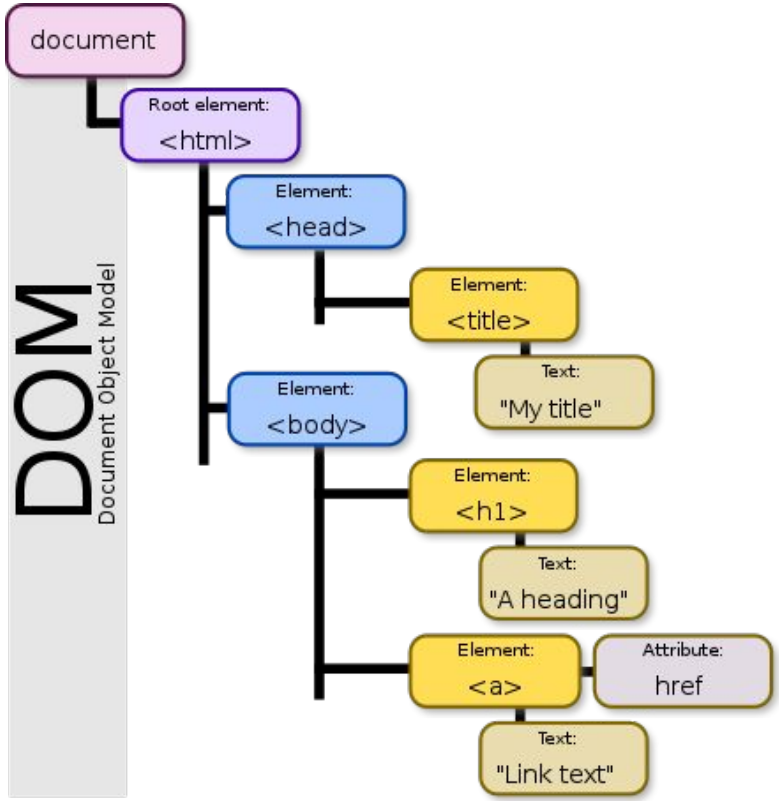
2. HTML - *H*yper*T*ext *M*arkup *L*anguage

- HTML Elements (common examples)
  - `<div> </div>` create a division, a container to divide an HTML into sections
  - `<h1>-<h6> </h1>-</h6>` different levels of heading
  - `<p> </p>` paragraph element, contains and displays a block of text
  - `<a> </a>` anchor, to create the hyperlinks
  - `<li> </li>` list item element create list items inside
    - `<ol>` ordered/numbered lists; `<ul>` unordered/bulleted lists
  - `<br>` line break, create a line break in text, no closing tag needed
  - `<em> <strong> <span>` change text style, need closing tag
  - `<img> </img>` embed images
  - `<video> </video>` embed media player for video playback
  - ...

2. HTML - DOM (Document Object Model)

- The DOM - treats an HTML document as a tree structure
- Each branch of a tree ends in a node
- Each node is an object representing a part of the document

- Programming
  - Access to the tree: Change the structure / content / style of a document
  - Nodes can have event handlers attached to them



Contents

0. Programming Paradigms
1. Web Application
2. HTML (DOM)
3. **CSS**
4. JavaScript
5. React

3. CSS - Cascading Style Sheets

- Without CSS
  - Every web page would be plain text and images that flowed straight down the page
- With CSS
  - Add color and background images
  - Change the layout of the page
  - Change text font
  - Align content
  - Create borders, margins, paddings
  - ....
- Make your pages works of art!

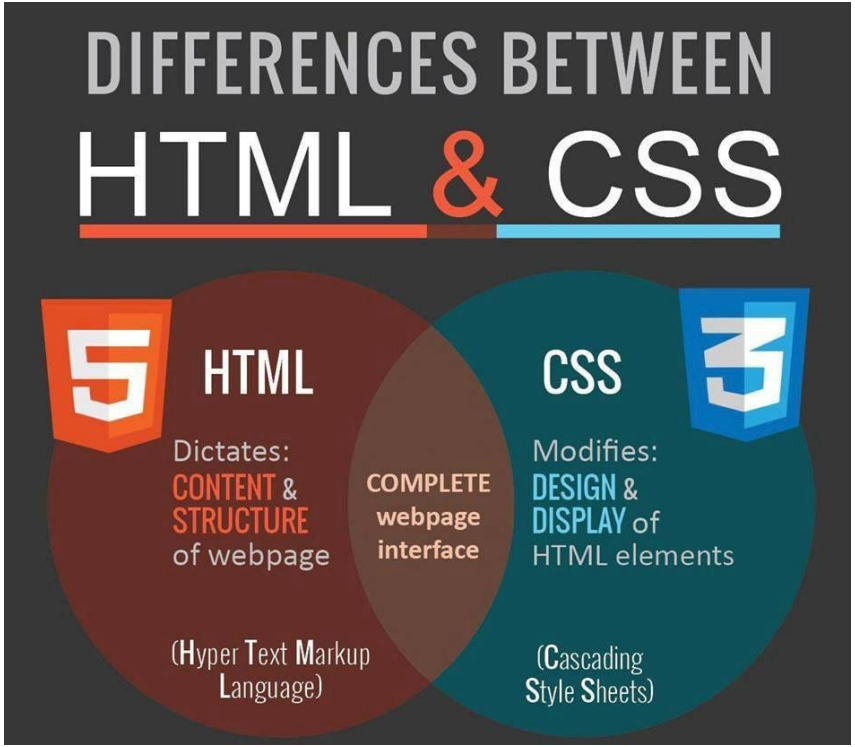


Image cited from the Internet

Contents

0. Programming Paradigms
1. Web Application
2. HTML (DOM)
3. CSS
4. **JavaScript**
5. React

4. JavaScript

Image cited from the Internet

- So far:
  - HTML - Content / Structure
  - CSS - Color / Style
  - Still static pages
- We need more !!!
  - Interaction !
- JavaScript
  - A powerful and flexible programming language of the web
  - Powers the dynamic behavior on most website
  - Paradigm: event-driven[1], functional, imperative
- Tutorial E.g.: https://javascript.info/

[1] Event-driven

- The flow of the program is determined by events such as user actions, sensor outputs or messages from other programs

4. JavaScript Basics

- Supports much of the structured programming syntax
  - Selections (if-else, switch)
  - Loops (while, for, etc)
  - Functions
- Weakly typed
  - Certain types are implicitly cast depending on the operation used
- Object-orientation
  - Prototype-based: JavaScript use prototypes while many other languages use classes for inheritance
    - Prototype objects => make new instances

- Object <= (properties, methods)
  - Properties:
    collection of `name: value` pairs

```
let user = {
  name: "John",
  age: 30,
  // multiword property name must be quoted
  "likes birds": true
};

// set
user["likes birds"] = true;
// get
alert(user["likes birds"]); // true
// delete
delete user["likes birds"];

user.sayHi = function() {
  alert("Hello!");
};

user.sayHi(); // Hello!
```

4. JavaScript - Manipulate the DOM

- Use JavaScript to manipulate the Page Structure
- General way:
  - Get the object(s) using some identifier
  - Manipulate the element

```
// get the element
let elem = document.getElementById("p1");

// make its background red
Elem.style.background= 'red';
```

There are 6 main methods to search for nodes in DOM:

Method	Searches by...	Can call on an element?	Live?
`querySelector`	CSS-selector	✔	-
`querySelectorAll`	CSS-selector	✔	-
`getElementById`	`id`	-	-
`getElementsByName`	`name`	-	✔
`getElementsByTagName`	tag or `'*'`	✔	✔
`getElementsByClassName`	class	✔	✔

This figure is from https://javascript.info/searching-elements-dom

4. JavaScript - Event Handlers

- Want some program piece only runs once a user perform actions
  - Mouse events: click, double click, drag/drop ...
  - Keyboard: keydown, keyup...
  - Pointer events
  - Modifiers: shift, alt, ctrl ...
  - ....
- Use an Event Handler
  - Bind some function to some specific event

```
<button id="button">Alt+Shift+Click on me!</button>

<script>
  button.onclick = function(event) {
    if (event.altKey && event.shiftKey) {
      alert('Hooray!');
    }
  };
</script>
```

Contents

0. Programming Paradigms
1. Web Application
2. HTML (DOM)
3. CSS
4. JavaScript
5. **React**

5. React

- Front-end JavaScript Library
  - For building user interfaces or UI components
  - Make it more easier and convenient to create our web app!
- Maintained by Facebook and a community of individual developers and companies

- React is only concerned with state management and rendering that state to the DOM
- Creating React applications usually requires the use of additional libraries

- React code is made of entities called components which can be rendered to a particular element in the DOM
  - Props: the value/data that parents can pass to child components
  - State: data that maintained by the component itself