A. Tables
   a. Presentation of tabular data
   b. Data that can be represented as a 2 dimensional display of columns and rows
   c. Tables were used for content layout, should only be used for tabular data now
   d. Caption
      i. caption/title of a table
      ii. Always comes after <table> (first child)
   e. Thead
      i. set of rows defining the head of the columns
   f. Tbody
      i. Set of rows defining the body of the table
   g. Tfoot
      i. set of rows summarizing the columns of the table
      ii. Example: totals in a spreadsheet
   h. Tr
      i. row of cells in a table
      ii. Container for combination of <th> and <td> cells
   i. Th
      i. Header of a group of table cells
   j. Td
      i. cell of a table that contains data
   k. Rowspan/colspan
      i. Allows a single table cell to span the width or height of more than one cell or column
         1. Picture "merge cell" in spreadsheet programs
      ii. Rowspan: Allows a single table cell to span the height of more than one cell or row
      iii. Colspan: Allows a single table cell to span the width of more than one cell or column
      iv. might be used for a header cell that titles a group of columns or a side-bar that groups rows of entries.
      v. colspan= and rowspan= are attributes <th> and <td>
      vi. The value of either attribute must be a positive integer (a whole number)
         1. specifies the number of columns or rows that the cell fills

B. Forms
   a. document section that contains interactive controls
      i. submitting information to a web server
      ii. capturing/handling interactive actions in a human-usable way
      iii. Action = URI of a program that processes the form information
         1. Often not used for React app
            a. Variables -> state -> API calls, etc.
         2. Would be used to send data to form processing script on a server (ruby, php, perl, etc.)
      iv. Method = HTTP verb method of sending data. POST, GET, etc.
      v. In modern single-page apps (eg, react), inputs are often used independent of full forms in order to add hooks for interactivity via handlers on elements (buttons and inputs)
   b. Fieldset
      i. Used to group multiple related fields/controls/labels within a form
      ii. Example: first/middle/last name
      iii. Browser default is to put border around, can be removed with CSS
   c. Legend

- i. Caption/title for content of its parent fieldset

d. Label
   - i. title/caption for interactive form element
   - ii. Best practice: associate label + input with “for”
     1. allows screen readers and other non-visual browsers to make link between label and input
     2. allows input to be activated when label is activated, especially on very small inputs (eg, checkbox, radio)
   - iii. Best practice: 1 label per input
     1. can have multiple labels
     2. Screen readers can have problems with them

e. Input
   - i. create interactive controls for web-based forms in order to accept data from the user
   - ii. Multiple control widgets
     1. Listing most common, several more depending on application you need
     2. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input
     3. Not all supported by all browsers
     4. [type=button]: button widget with no default behavior
        - a. Use for any action where you need to capture javascript action and trigger code
     5. [type=text]: default text inputs
     6. [type=checkbox]: allows multiple values to be selected for an input response
        - a. Value: not seen in the browser, corresponds to the value to be given to as the input response
          - i. If value is omitted, default value is “on”
        - b. Multiple checkboxes
          - i. Server will receive values separated by “&”
     7. [type=radio]: allows only 1 value to be selected for an input response
     8. [type=submit]: button widget with default behavior of submitting the form
     9. [type=password]: password inputs
     10. [type=number]: number inputs
         - a. Mobile browsers will launch number-only keypad
         - b. Browser provides automatic validation entered text is a number
         - c. set of up and down buttons to step the value up and down
     11. [type=tel]: telephone inputs
         - a. Mobile browsers will launch telephone keypad
         - b. makes adding custom validation and handling of phone numbers more convenient
         - c. the input value is not automatically validated to a particular format
   - iii. Disabled
     1. State where user cannot interact with the control
        - a. Not clickable
        - b. User cannot activate/input value
   - iv. Readonly
     1. User cannot modify the value of the input
     2. Different than disabled: user can still click on/interact with control
   - v. Required
     1. Indicates form is invalid if left empty (will not submit)

f. Select
    i. control that provides a menu of options
    ii. Multiple: allows multiple options to be selected by cmd/ctrl clicking
g. Optgroup
    i. Group options in a select
    ii. Label displayed is not selectable
h. Option
    i. Defines items contained in a select or optgroup
    ii. Value: value to be sent to the form
    iii. Selected: indicates default selected option
        1. If none specified, defaults to first in the options list
        2. If multiple specified, multiple can be selected

C. Button
    a. Clickable button element
    b. Can be used either inside or outside forms
    c. Presented as same style as OS button by default with no styling
    d. Value: initial value of the button.
    e. Button content: enter between tags

D. Image
    a. Embeds image into a document
    b. Src attribute = path for images
        i. Relative
        ii. Absolute
    c. Width & height attributes inherent on tag to set layout for stable page layout
        i. CSS for width & height
    d. Each browser supports different set of image formats
        i. Eg Firefox: JPEG; GIF, including animated GIFs; PNG; APNG; SVG; BMP; BMP ICO; PNG ICO
    e. Alt attribute: alternate text displayed
        i. Eg while loading, loading error

E. DOM
    a. Document Object Model
    b. Even though it has functions, not a programming language
    c. DOM is an in-memory, object-oriented representation of web pages, HTML documents, XML documents, and their component parts as objects and nodes which can be modified with a scripting language
        i. The page content is stored in the DOM and may be accessed and manipulated via JS
    d. DOM as an object
        i. describes how every element in HTML page relates
            1. HTML document = the topmost structure
            2. each element on the page = separate object that with own relationship to the document
        ii. Allows interaction w/individual elements and page as a whole
            1. Modify elements
            2. retrieve and set element properties
            3. add and remove elements or objects
            4. capture and respond to user or browser actions/events
    e. Viewing DOM

            i. HTML in devtools = visual representation of the DOM
            ii. created directly from your HTML, but it's often not the same, possible reasons:
                1. mistakes/looseness in HTML → browser has fixed them for you
                    a. Eg table
                        i. browser will insert missing <tbody>
                        ii. Viewable in DOM
                2. Manipulated DOM via JS
                    a. Eg empty container injected with content via js (react app node)
        f. Structure
            i. Live DOM viewer: http://software.hixie.ch/utilities/js/live-dom-viewer/
            ii. every element of the document = object or a node
                1. organized in a hierarchical fashion
                2. has a function and an identity
                3. each node can also have any number of child nodes
                    a. may be other elements or text nodes
            iii. browsers read an HTML page and turn data into objects logically arranged as a data structure tree (DOM tree)
                1. parent node (the node right above it)
                2. child nodes (the nodes below it)
                3. siblings nodes (other nodes belonging to the same parent)
            iv. Types of nodes
                1. 12 node types: https://dom.spec.whatwg.org/#node
                2. Element Nodes
                    a. individual tags or tag pairs in HTML
                3. Text Nodes
                    a. content in between the open/close HTML tags
                    b. usually have a parent node and sometimes sibling nodes
                    c. cannot have their own child nodes.
                4. Comment Node
                    a. Doesn't affect presentation
                    b. Everything in HTML becomes DOM
        g. Accessing and using DOM
            i. As soon as JS is loaded, can immediately use API for the document or window elements
                1. manipulate the document itself
                2. get at the children of document (elements in the web page)
            ii. Document and window objects used most often
                1. window object = the browser
                    a. global scope
                    b. all functions and methods built into JS are built off the window object
                    c. JS checks window for any variables we haven't defined
                2. document object = root of the document itself
                    a. DOM representation is a property of window (window.document)
    F. React
        a. JS crash course
            i. Variables
                1. A box to store data in
                2. const/let
                    a. Const -- value won't change

            b. Let -- value might change
    ii. Objects
        1. Everything in JS is made up of objects
            a. Special objects: Number, String, etc.
        2. Structure of key/value pairs
            a. Value can be any JS object: number, string, function, class, etc.
        3. Access objectName[‘foo’] or objectName.foo
        4. Used for data that doesn’t need to be in a specific order when looping
        5. Loop through for...in
    iii. Arrays
        1. Structure of ordered list of elements
            a. elements can be any JS object: number, string, function, class, etc.
        2. Access arrayName[index number of element]
        3. Loop through map/for each/for loop
    iv. Functions
        1. Params go in, result comes out
            a. In the case of React, params go in, HTML comes out
        2. Single responsibility -- small & granular
        3. Function examples
    v. Event handlers
        1. Eg. onClick, onChange
            a. Functions that are called when something on a React app triggers it
            b. Usually something like a button, form input, etc.
        2. Defining handlers as part of a class
    vi. State
        1. Way of storing variables needed to display the UI in a certain way
        2. When state is the same, UI should be the same
        3. setState AddingMachine example
b. React is view library for generating html
c. Why use React?
    i. the view is a visual representation of your application’s state
        1. For a given application state, your view will always look the same way
        2. Makes it easier to reason about how your view will behave
        3. Great for repetition (reusable components)
    ii. State
        1. stores a component or app’s dynamic data & determines the component or app’s behavior and visual representation
        2. because state is dynamic, it enables a component to keep track of changing information in between renders
        3. Examples of uses for state
            a. Which navigation item is active
            b. Whether a button is disabled or not based on external logic
            c. The value of an input
        4. React teaches us to think about business logic that might need to be shared across the app as state
            a. build your HTML using pieces of that state
            b. No one piece of the UI owns the knowledge about the app
            c. “Single source of truth”

iii. Reusable components

iv. Virtual DOM

1. memory reconciliation algorithm
2. not invented by React, but React uses it and provides it for free
3. abstraction of the HTML DOM
    a. constructs a representation of the page in a virtual memory
    b. performs the necessary updates/diffs with previous version before rendering the final into the browser
    c. Makes rendering very fast, because only the pieces that actually changed are updated
        i. Every time the DOM changes, browser needs to recalculate the CSS, do layout, and repaint the web page
        ii. Reconciliation diagram: https://medium.com/@gethylgeorge/how-virtual-dom-and-diffing-works-in-react-6fc805f9f84e
        iii. The real DOM is updated with only the actual changes, like applying a patch
        iv. you won't have to deal with the DOM directly at all, react will handle for you

G. Create-react-app

a. environment for learning React
b. Bootstraps to start building a new single-page application in React
c. sets up your development environment so that you can use the latest js features
d. provides a nice developer experience
e. optimizes your app for production
f. doesn’t handle backend logic or databases
g. creates a frontend build pipeline, so you can use it with any backend you want
h. uses Babel and webpack, but you don’t need to know anything about them

H. Components

a. Basically JavaScript functions
    i. accept arbitrary inputs (called “props”) and return React elements describing what should appear on the screen
        1. Props can be strings, numbers, other React components, functions
        2. Component can also be passed special props called children
            a. Like HTML tags, component tags can surround content
            b. Eg <Foo>this is text, an image, another component</Foo>
            c. The text, img, other component would be accessed from Foo.js via props.children
    ii. module-like pieces of code
    iii. they’re reusable and can be repeated across several web pages
    iv. single responsibility principle
        1. a component should ideally only do one thing
        2. If it ends up doing a lot, it should be broken into smaller components
b. Kinds of React components
    i. Class components
        1. Class components are ES6 classes
        2. “class Foo extends Component”
        3. only required method is render()

        4. Gives you access to lifecycle hooks
            a. methods for when the component is
                i. Rendered
                ii. added to and removed from the DOM
                iii. updated with new state or props
            b. Internal state
        5. Structure of a React class component
            a. This.props
            b. This.state
            c. Constructor
            d. Render
    ii. Functional components
        1. Simply a functions that can accept props as an argument and return valid JSX
        2. Lack of state and lifecycle methods
        3. Also called “stateless components”
        4. Small and reusable
        5. simple to both read and understand
c. Working with components
    i. Components are imported from their component files into the app or component where you want to use them
    ii. Eg import Foo from ‘Foo.js’
    iii. Referenced with <Foo />
    iv. If component can take children, then <Foo></Foo>
d. Create a component example
    i. Hello [user.name]