

Learnings

- **position: sticky [CSS]**
 - similar to position: fixed where both maintain their position on the screen, even as the user scrolls up or down the page.
 - difference is that sticky element remains confined to the parent container it is in
- **What does a doctype do? [HTML]**
 - purpose is to prevent a browser from switching into “quirks mode” when rendering a document
 - doctype ensures that the browser makes a best effort at following relevant specifications, rather than using a different rendering mode that is incompatible with some specifications
- **What does the data-* attribute do? [HTML]**
 - allow us to store extra information on HTML elements
 - can access those information without extra properties on DOM ... etc

```
<article id="electric-cars" data-columns="3" data-index-number="12314" data-parent="cars"> ... </article>
```

- **Difference between cookie, sessionStorage, and localStorage? [HTML]**
 - They are all client-side storage options in a nutshell. Unlike database ... , which are server side storage
 - *cookie*
 - same as below, but can be trivially tempered
 - can have a degree of protection like Cross Site Scripting (XSS)
 - prevents access from javascript
 - all cookies valid for a page are sent from the browser to the server for every request to the same domain. therefore not fit for large amount of info
 - *localStorage & sessionStorage*
 - both the same with the exception of persistence
 - sessionStorage is available only for the duration of the browser session (and is deleted upon tab close or window close)
 - localStorage survives page reloads
 - perfect for storing non-sensitive data, i.e preferences, scores in games
 - HTTP is a stateless protocol - web apps have no way of identifying a user from previous visits on returning to the web site
- **Differences between script, script async, script defer? [HTML]**
 - when browser load HTML and comes across a script tag, it can't continue building the DOM, must execute the script first.
 - when leads to two issues
 - scripts can't see DOM elements below them, so can't add handlers
 - if there is a bulky script, it will block the user

- Several ways to fix
 - put the script at the end so that it doesn't block the previous html elements to render
 - but defer and async better
 - defer
 - tells the browser that it should go on working with the page, and load the script "in background", then run the script when it loads
 - async
 - script is completely independent
 - the page doesn't wait for async scripts, the contents are processed and displayed

| | Order | DOMContentLoaded |
|-------|---|--|
| async | Load-first order. Their document order doesn't matter – which loads first | Irrelevant. May load and execute while the document has not yet been fully downloaded. That happens if scripts are small or cached, and the document is long enough. |
| defer | Document order (as they go in the document). | Execute after the document is loaded and parsed (they wait if needed), right before DOMContentLoaded . |

- **What is progressive rendering? [HTML]**
 - Lazy loading: i.e loading an image when it comes into the browsers viewport instead of loading all images at page load
 - prioritizing visible content: including the minimal css/contents/script for the page to render
- **Describe Floats and how they work? [CSS]**
 - places an element on the left or right side of its container, allowing text and inline elements to wrap around it
 - it is taken out of the normal float of the document (though still remaining part of it, i.e text will flow around floated elements), unlike absolute positioning elements, which are removed from the flow of the page
 - use *clear* to force an item to move below any floated item
- **How to fix browser specific issues? [CSS]**
 - use a separate style sheet that only loads when that specific browser is being used. requires server-side rendering though
 - use libraries like bootstrap to handle them
- **Different ways to visually hide elements? [CSS]**
 - visibility: hidden
 - width: 0; height: 0;
 - position it outside of the screen. absolutely
- **Explain how a browser determines what elements match a CSS selector. [CSS]**

- browsers match selectors from rightmost to left. Browsers filter out elements in the DOM according to the key selector, and traverse up its parent elements to determine matches
- the shorter the length of the selector chain, the faster the browser can determine if that element matches the selector
- **Describe pseudo-elements and discuss what they are used for? [CSS]**
 - a keyword added to a selector that lets you style a specific part of the selected element i.e
 - :first-line, :first-letter, content:, :before, :after
- **Describe CSS box model. [CSS]**
 - when layout document, browser's rendering engine represents each element as a rectangular box. CSS determines the size, position, and properties of these boxes
 - each box is determined by padding, margin, border, and content edge

OLIA EVANS
@bork

the box model

>>

every HTML element is in a box

```
<div class="1">
  <div class="2" />
  <div class="3" />
</div>
```

boxes have padding, borders, and a margin

Width doesn't include margin/border/padding by default

margins are allowed to overlap sometimes

box-sizing: border-box; includes border + padding in the width

inline elements ignore other inline elements' vertical padding

`span` ← you can set vertical padding but the other span won't move

- **What does `* { box-sizing: border-box }` do? [CSS]**
 - by default, elements have that applied.
 - box-sizing border box changes how the width and height are calculated, where border and padding are also being included in the calculation, where
 - height = content height + border height + padding height
- **Difference between inline and in-block? [CSS]**

| | |
|--------|---|
| inline | - respects left & right margins and padding, but not top and bottom |
|--------|---|

| | |
|--------------|---|
| | <ul style="list-style-type: none"> - cannot have width and height set - allow other elements to sit to their left and right |
| block | <ul style="list-style-type: none"> - respect all left & right & top & down margins and padding - force a line break after the block element - acquires a full width if width is not defined |
| inline-block | <ul style="list-style-type: none"> - all elements sit to left or right - respect top and bottom margin and padding - respect height and width <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 2px;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 2px;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 2px;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 2px;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 2px;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> </div> <div style="border: 1px solid green; padding: 2px; display: inline-block; margin: 10px 0;"> <div style="background-color: #e0ffe0; padding: 2px;"><div class="box"></div> <div style="background-color: #e0ffe0; padding: 2px;">I'm floating!</div> <div style="background-color: #e0ffe0; padding: 2px;"></div></div> </div> <div style="border: 1px solid orange; padding: 2px; margin-top: 10px;"> <div style="background-color: #fff9c4; padding: 2px;"><div class="after-box"></div> <div style="background-color: #fff9c4; padding: 2px;">I'm using clear so I don't float next to the above boxes.</div> <div style="background-color: #fff9c4; padding: 2px;"></div></div> </div> <ul style="list-style-type: none"> - the yellow box can be in its own line, and won't be forced to the right of the last box </div></div></div></div></div> |

JULIA EVANS
@bork

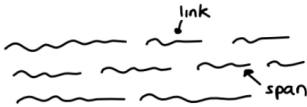
inline vs block

»

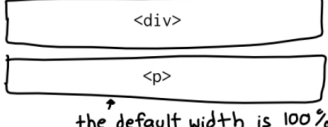
HTML elements default to inline or block

| example block elements | example inline elements |
|------------------------|-------------------------|
| <p> <div> | <a> |
| | <i> |
| <h1> - <h6> | <button> <input> |
| <table> <form> | <small> <abbr> |
| <article> <nav> | <textarea> |

inline elements are laid out horizontally



block elements are laid out vertically by default



inline elements ignore width & height

Setting the width is impossible, but you can use line-height to change the height

also, inline elements ignore the vertical padding of other inline elements


display can force an element to be inline or block

display determines 2 things:

- ① whether the element itself is inline, block, inline-block, etc
- ② how child elements are laid out (grid, flex, table, default, etc)

display: inline-block;

inline-block makes a block element that's laid out horizontally like an inline element



- **Difference between nth-of-type() and nth-child() selectors?**
 - **p:nth-of-type(2):** selects the second matched p element ← this does not take into consideration of the other sibling tags
 - **p:nth-child(2):** selects the second child from the parent ← this does take into account of other same tags

```
<div>
  <h1>Hello</h1>
  <p>Paragraph</p> ←- selected by nth-of-type(2)
  <p>Target</p> ← selected by nth-child(2)
</div>
```

- **Difference between relative, fixed, absolute, and statically positioned element?**
 - *Static:* default. element flows into the page as it normal would
 - *Relative:* position is adjusted relative to itself. without changing layout
 - *Absolute:* removed from the flow of page and positioned at a specified position relative to its closest positioned ancestor
 - *Fixed:* removed from flow of page and positioned at specified position relative to the viewport
 - *Sticky:* treated as relative position to its parent
- **Any reason why you want to use translate() instead of absolute positioning?**
 - changing transform or opacity does not trigger browser reflow or repaint, only compositions, whereas changing absolute positioning triggers reflow.
 - Transform causes browser to create a GPU layer
 - Absolute positioning uses CPU.
 - translate it more efficient and will result in shorter paint times for smoother animation

- Implement Vue.js

[SandBox](#)

```
<div id="app">
  {{message}}
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue!'
  }
})
```

- Goal is to make message appear
- 1. Get access to the element first. Do this by using document.querySelector(), and do String.replace() on its innerHTML property

```

Class Vue {
  constructor(options) {
    const el = document.querySelector(options.el)
    const data = options.data

    Object.keys(data).forEach(key => {
      el.innerHTML = el.innerHTML.replace(`{{key}}`, data[key])
    })
  }
}

```

2. If the div has something nested inside (i.e a p tag), will need to traverse the DOM to retrieve the element. Use the **element.childNodes** to get the element

- **Implement async.series**

- run the collection of functions in **series**, each one running once the previous one is done. and if any function in the series pass an error to its callback, no more functions run, and the callback is immediately called with the value of the error

```

const async = {
  series: (tasks, callback) => {
    let i = 0
    const results = []
    const _callback = (err, result) => {
      results[i] = result
      if(err || ++i >= tasks.length) {
        callback(err, result)
        return;
      }
      tasks[i](_callback)
    }
    tasks[0](_callback)
  }
}

```

- run the collection of functions in parallel

```

const async = {
  parallel: (tasks, callback) => {
    let done = false
    let count = 0
    const results = []
    const _callback = (err, result) => {
      count++
      results[i] = result
      if(!done && (err || count === tasks.length)) {

```

```

        callback(err, result)
        done = true
        return;
    }
    tasks[i](_callback)
    }
    tasks.forEach((task, i) => {
    task((err, result) => _callback(i, err, result))
    })
    }
}

```

- **Implement slide-out transition without using CSS**
 - use `window.requestAnimationFrame()` to stimulate a translation

```

const slideOut = (element, duration) => {
const initial = 0
const target = window.innerWidth;
const start = new Date();
const loop = () => {
    const time = (new Date().getTime() - start.getTime()) / 1000
    const value = (time * target) / duration + initial
    box.style.transform = `translate(${value}px)`
    if(value >= target) {
        box.style.transform = ""
        return;
    }
    window.requestAnimationFrame(loop)
}
}

```

- everytime something is clicked, the function is called. it initializes the initial and target values for the `translateX` transform. The loop will continue to execute until the the end of the screen