UDEMY FRONTENED

Current lesson:

<https://www.udemy.com/course/the-advanced-web-developer-bootcamp/learn/lecture/7598306#content>

FlexItem Properteis

Curriculum walkthrough

* CSS animations
* CSS flexbox (layouts)
* Special project section (company website)
* Asynchronous javascript (event loop and queue, stack and heap)
* Ajax (xml request api, axios stuffs and jquery)
* Testing (jasmine testing library, test writing)
* Advanced array methods (map, filter, reduce, every, some)
* Closures and keyword ‘this’ and ‘new’(‘use strict’ implicit and explicit binding, call(), apply() and bind())
* OOP(object oriented programming, constructors, prototypes, chain, inheritance)
* Creating own apps(own api using node, mongodb, and express, routes that respond with json content, restful routing api)
* Single page apps with node(full crud on top of node json api, then without framework all from scratch)
* Es2015 (ins and outs, let const, arrow functions, rest and spread etc…)
* Guess the password app refactor (refactor complete app using ES2015 features)
* ES2016, 17(includes, async functions, object rest and spread)
* D3 and the dom(manipulate dom data visualization, add event listeners, refactor using d3 notes app)
* D3 and data(use d3 and dom elements and data together, build project)
* SVG and D3(draw using svg, refactor using d3 to make pie charts/line graphs)
* Intermediate d3(extreme values and scale data with d3, scatterplot, histogram)
* Advanced d3(manage async code, tooltips, maps and map data, force directed graphs)
* D3 climate dashboard(c02 data emissions)
* Intro to react and jsx
* Create react app and props(webpack and other tools, create app, recipe demo app)
* React and state(compare props and state, work with dev tools, component architecture, other exercises)
* Virtual dom and react events(handle events, work with forms, build a couple apps)
* The component life cycle(make ajax calls, using componentdidmount, geography app)
* React and authentication(auth topics, json web tokens, hash and cookies, implement auth on nodejs server with bcrypt, authentication vs authorization, start warbler project)
* React router(add router to projects, manage auth with routing, warbler)
* React and redux(overview, actions, reducers, pure functions, dev tools, redux-thunk middleware, finish warbler

Css animations

Pseudo classes

-commonly used as triggers ( :hover, :active, :checked, :focus)

Hover - on mouse hover.

Focus - on focus, tab selection, mouse click, persists

Active – being activated by user, mouse click / hold

h1:hover{

background-color: pink;

}

button:focus{

border-radius: 10px;

background-color: #2ecc71;

}

button:active{

background-color: orange;

font-size: 30px;

}

Mini exercise – button animation

Alter box shadow on hover, click

Adding top drops element downward, illusion of sinking into page, eventually eliminating the shadow on click

.btn:hover{

box-shadow: 0 4px #efa424;

top: 2px;

}

.btn:active{

box-shadow: none;

top: 6px;

}

Transforms – move, wrap, rotate and scales elements

Translation – move something around (translate works with origin only, 2 translates sucks)

Origin – sets origin target of element

p.move-right{

background-color: #3498db;

transform: translatex(100px);

}

.move-down{

background-color: #f1c40f;

transform: translatey(20px);

}

p:hover{

x/y

transform: translate(20px, 20px);

}

Scaling – alter size of element (takes initial origin, scaling evenly)

//scale from top left, helps uniform elements

.transformed {

background-color: #3498db;

transform: scale(2);

transform-origin: 0 0; // transform-origin: top left

}

//scale toward bottom, 0.5 scale to bottom, shrink keeping bottom position

.transformed {

background-color: #3498db;

transform: scale(2, 0.5);

transform-origin: left bottom

}

Rotations – spin stuffs

//TRANSFORMS MUST BE DECLARED ONCE ELSE OVERWRITTEN

.transformed {

background-color: #9b59b6;

transform: rotate(90deg // 0.25turn)

scale(0.5);

transform-origin: right bottom;

}

Browser Prefixes – to work with every browser, some animations wont work without

-webkit-transform: rotate();

Auto prefixer auto adds requirements

CSS Transitions – allows us to control animation speed when changing css properties (double size of div ‘over 3 seconds’)

4 properties

* Transition-duration (length of change)
  + Transition-duration: 1s;
  + Transition-duration: 1s, 3s; (can have multiple)

.animated{

Properties are list order for duration

transition-property: background-color, border-radius, transform;

transition-duration: 5s, 5s, 5s;

}

* Transition-property (selection of particular property such as color)
  + Transition-property: background;
  + Transition-property: color, opacity;
* Transition-timing-function (acceleration, linear, custom curves)

Easings.net for curves

* + Transition-timing-function: ease-in/out, linear
  + Transition-timing-function: cubic-bezier(0.950, 0.05, 0.795, 0.035);
* Transition-delay (delay before animation starts)
  + Transition-delay: 4s;
  + Transition-delay: 5ms, 10s;

Background-color transition length 5s, delayed for 2s after trigger

Border-radius transition length 1s, delayed for 2s after trigger

.animated{

transition-property: background-color, border-radius;

transition-duration: 5s, 1s;

transition-delay: 2s, 2s;

}

* Shorthand Transition (single line transition statements)

Passing - (property, duration, timing function, delay)

* + Transition: background 1.5s ease-in 1;

.animated{

transition:

transform 2s ease-out 1s,

background-color 0.5s linear;

/\* transision-property: transform;

transition-duration: 2s;

transition-timing-function: cubic-bezier(0.68, -0.6, 0.32, 1.6);

transition-delay: 1s; \*/

}

2 Important Questions (Performance)

1. What can be transitioned
2. What should be transitioned
3. Tons, not all
4. Recommended if concerned about performance
   1. Transform: translate( npx, npx );
   2. Transform: scale( n );
   3. Transform: rotate( ndeg );
   4. Opacity: 0…1;

Make something Exercise

Target image for css, apply transition properties to all item classes in div

div.item:hover img{

opacity: 0.3;

}

div.item \* {

transition: all 1s ease-in-out;

}

Set non hover properties, then hover

div.item i{

transform: translate(-50%, -50%) scale(0);

transition: transform 300ms 100ms cubic-bezier(0.175, 0.885, 0.32, 1.275);

}

div.item:hover i{

transform: translate(-50%, -50%) scale(1);

transition: transform 300ms 100ms cubic-bezier(0.175, 0.885, 0.32, 1.275);

}

Img hover settings

div.item:hover img{

opacity: 0.3;

filter: grayscale(100%);

}

Keyframe Animations

* Transitions allow single state change

Keyframes allow for much more complex multistate animations

Sunrise to sunset, add some stuff in between

Hello! Animation task

0%

* Color: red
* Font-size: 20px

25%

* Color: orange

50%

* Color: yellow
* Font-size: 40px

75%

* Color: green

100%

* Color: blue
* Font-size: 20px

Step 1 – Syntax (should use scale instead of font)

@keyframes rainbowtext{

0%{

color: red;

font-size: 20px;

}

25%{

color: orange;

}

50%{

color: yellow;

font-size: 40px;

}

75%{

color: green;

}

100%{

color: blue;

font-size: 20px;

}

}

Step 2 – Apply

p{

animation-name: rainbowtext;

animation-duration: 3s;

animation-timing-function: linear;

animation-delay: 0s;

animation-iteration-count: infinite;

}

Keyframe properties

* Animation-name
* Animation-duration
* Animation-timing-function
* Animation-delay

Newer properties

* Animation-iteration-count

How many times should it repeat?

* + Animation-iteration-count: 1
  + Animation-iteration-count: 7
  + Animation-iteration-count: infinite
* Animation-fill-mode (usually only use forwards)

How an animation should apply styles before starting and after animation

* + Animation-fill-mode: forwards

Takes end state(100%), keeps after ending

* + Animation-fill-mode: backwards

Takes start state(0%), keeps after ending

* + Animation-fill-mode: both

Combines forward and back

* + Animation-fill-mode: none
* Animation-direction

Defined animation 0-100 could be reverse, alternated

* + Animation- direction -count: forward
  + Animation- direction -count: reverse
  + Animation- direction -count: alternate
* Animation-play-state

Specifies whether the animation is running or paused

* + Animation-play-state: running

Starts animation

* + Animation-play-state: paused

Pauses animation

Rising and setting sun exercise

Step1 – plan

Skycolor{

0% grey

10% red/orange

30% blue

70% blue

90% red/orange

100% grey

}

Sun{

0, 25, 50, 75, 100%

Transform: translate

}

Shorthand Animations – combines animations into single line (maybe try not to use first one)(order kind of matters)

Order is important, first value parsed as time is duration, second is delay. For distinguishing anim name from other keywords

Animation: 3s ease-in 1s 2 reverse both paused slidein;

Animation: changecolor 3s linear 1s infinite;

Animation: jiggle 4s;

old

{

animation-name: slideRight;

animation-iteration-count: infinite;

animation-delay: 0s;

animation-duration: 2s;

}

@keyframes slideRight{

100%{

transform: translatex(500px);

}

}

New

animation: slideRight 2s ease-in infinite 2s reverse;

CSS Loader Icon (likely most common occurrence of css animations)

Simple Loader Icon

A picture containing logo

Description automatically generated

.loader{

border: 16px solid #bdc3c7;

width: 120px;

height: 120px;

border-top-color: #1abc9c;

border-bottom-color: #3498db;

border-radius: 50%;

animation: Spin 2s linear infinite;

}

@keyframes Spin{

0%{

transform: rotate(0deg);

}

100%{

transform: rotate(360deg);

}

}

Keep opacity appear needs forwards

h2 {

font-size: 36px;

opacity: 0;

animation: Appear 1s 3s forwards;

}

Advanced Css Layout with flexbox

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

Css Flexbox – A more efficient way to layout, align and distribute space among items in a container, even if size is unknown (bit of syntax to get used to)

2 types of properties

* Container properties
  + Flex-direction –
    - Specifies how items are places in the flex container, defining the main axis and its direction
    - flex-direction: row;
    - flex-direction: row-reverse(mirrored);
    - flex-direction: column; (reverse)
  + Justify-content –
    - Defines how space is distributed between items in flex container along the main axis (padding)
    - Justify-content: flex-end;(keeps order, moves to end)
    - Justify-content: center:
    - Justify-content: space-between; (even space between)
    - Justify-content: space-around;
  + Flex-wrap –
    - Specifies whether items are forced onto a single line or can be wrapped into multiple lines
    - Flex-wrap: wrap (reverse); changes cross axis, builds upwards
  + Align-items
    - defines how space is distributed between items in flex container along the cross axis
    - align-items: flex-start;
    - align-items: flex-end
    - align-items: stretch
    - align-items: center;
    - align-items: baseline; (aligned with text)
  + Align-content
    - Defines how space is distributed between rows in flex container along the cross axis
    - Aling-content: stretch;
    - Aling-content: flex-start;
    - Aling-content: flex-end;
    - Aling-content: space-between
    - Aling-content: space-around
    - Aling-content: center;
* Individual items of container (Flex item properties)
* Flex: <grow><shrink><basis>
  + Order
    - Specifies the order used to lay out items in their flex container
    - All items default order 0
    - Order: 2 while all have 0 sends to end
    - Order: -1 while all have 0 sends to start
  + Flex Defines how a flex item will grow or shrink to fit the available space in a container
    - Its actually a shorthand property for 3 others
  + Flex-grow
    - Dictates how the Unused space should be spread amongst flex items
    - Its all about ratios
    - Defaults to 0
    - Shares of white space
    - Flex-grow: 1 evenly divides space if applied to all items
    - Flex-grow: 2 takes double space of element with 1
  + Flex-shrink
    - Dictates how items should shrink when there isn’t enough space in a container
    - Defaults to 1
    - Rate of shrink relative to others
  + Flex-basis
    - Sort of like width but not
    - Specifies the ideal size of a flex item BEFORE its placed into a flex container
    - Initial idea size for a flex item
    - Width for rows, height for columns
  + Align-self
    - Overrides align-items on individual flex items
    - Align-self: flex-start
    - Align- self: flex-end

Container{

Display: flex;

}

No flex flex

Icon

Description automatically generated with medium confidence

Terms

Flex container – container which has display flex

Flex item – items within flex container

Main axis – (not always) from left to right, can reverse, make up and down column maxing cross axis left to right

Cross axis – from top to bottom

NAV BAR Assignment

Flexbox sidebar

Align-items - defines how space is distributed between items in flex container along the cross axis

Lets Make A Nav Bar Exercise

Holy Grail Layout

Browser support for flexbox

Fully supported without prefixes

SECTION 5 Building a Startup Site

Project introduction

ES 20 something

Testing here

<https://www.udemy.com/course/the-advanced-web-developer-bootcamp/learn/quiz/371466#content>