[Welcome](#_ox9af0pgdqp5)

[Getting Started](#_5kds7mq7eio6)

[Having Fun #1 - HTTP Status Codes](#_n40c39yynir)

[Having Fun #2 - Making The News](#_1db0j3wp15v4)

[Having Fun #3 - The Dinosaur Video Game](#_juniziiykvjb)

[Websites vs Apps - Should You Learn Web Dev of App Dev?](#_a9tktb9u81vn)

[Your Pathway to a $100,000 / Year Salary](#_87cinjfranim)

[Course Overview - The Art of Building Websites](#_ioxqfhb8l74h)

[FREE PREVIEW this video](#_s5rjux8xs3tb)

[Getting The Code Used In This Course](#_f0zzlu9uwqq9)

[Stay Current, Find Jobs, Receive Discounts](#_eizznme0cgfu)

[FREE PREVIEW this video](#_d7t79obod4hj)

[Resources](#_rn1hbtlhsfyt)

[How To Succeed](#_5tpsiu3a17qn)

[Understanding The Internet & World Wide Web](#_xyq04jsgk808)

[History of the Internet](#_b6yvf3hp39zp)

[History of the World Wide Web](#_cgqo3cdkgzk2)

[Who Controls the Internet & WWW?](#_hykb7zkoszvu)

[Web Documentation](#_ntimy1duitgg)

[Highlighting Webpages & Chrome Extensions](#_tktrp5cavx7g)

[How The Web Works](#_oz9qn8894fn8)

[The Dark Web](#_2nahoimh7vre)

[BitTorrent](#_33zvbw1suhfq)

[An Introduction to HTML](#_tw8iwsywjuj3)

[Your First Webpage](#_7xu4gh1olxjs)

[Anatomy of an HTML Page](#_9x0ek71uwqz5)

[HTML Terminology](#_ufzy3f92v8h1)

[An Introduction to CSS](#_iep60twunfkp)

[A Separation of Concerns](#_wjhck8j5v2kn)

[CSS Rule-sets](#_c1sm5ka37kls)

[Linking CSS to HTML](#_hmzx0solnvkg)

[Multiple CSS Selectors](#_m3cuy8khqqvm)

[Configuring Your Environment](#_r14aghmormc)

[Integrated Development Environments](#_ehpd997yn8dt)

[WebStorm](#_j5f7pg2a626j)

[Atom.io](#_b77084my7ds1)

[Sublime](#_nfhh6hjsoigh)

[Dreamweaver](#_mn4cv5mh0kqj)

[Becoming Skilled with Github](#_6h19jnqj29mp)

[Github Overview](#_gmlup6rlaiva)

[Git & Github - The Story](#_24i1j5ppovur)

[Configuring Windows for Github](#_mv184vg49x0s)

[Configuring Mac and Linux for Github](#_i2d8twrweeg)

[Terminal Essentials](#_41ey2rw2v0gd)

[Creating a “Master” Local Git Repository](#_ylarbsqqpwp3)

[Creating an “Origin/Master” Remote Github Repository](#_au9yzwpobmwf)

[Using Github - Basic Commands](#_mkxzih7oilgf)

[Using Github - Deleting A Repo](#_8jw7ehdqy41r)

[HTML Essentials](#_oab2zslazuzd)

[Section Overview](#_9ox7eu10g9c4)

[FIle Naming Conventions](#_f6slatcfeggl)

[Folder Naming Conventions](#_8sibk7tduqq2)

[Opening A Project in Webstorm](#_w54bxr8gubl9)

[Essential Tags](#_k84kqd5nne8k)

[Paragraph element](#_y2ppsawcmc7j)

[Heading elements](#_27q47fvsn3pl)

[Unordered list element](#_7brz62kdbvp2)

[Ordered list](#_c2v90mwrgell)

[Emmet.io](#_2kks2eqjb)

[Modifying Webstorm Formatting](#_dgnz0pucqw0)

[Tag Attributes](#_d54bafu6wdlc)

[Link element](#_h99kgp6id63m)

[Image element](#_dg02q0ybmrks)

[Anchor element](#_wvumycdjwmmc)

[Paragraph element](#_rs3i9hqhk1y1)

[Relative URLs](#_pxd57v8cknqk)

[Absolute URLs](#_t3p7scb004tn)

[Comments](#_hm72sbhvragz)

[Hands-On Exercises](#_se1khzylp6dt)

[Hands-On Exercises - Solutions](#_i2vgm3vq5mp6)

[CSS Essentials](#_ve0t67llg806)

[Section Overview](#_ucy1caga5b8j)

[All HTML Tags & Attributes, All CSS Selectors & Properties](#_861t894q0q73)

[border Property](#_hew9d7aqnrns)

[border](#_p2hlwsm2dyzo)

[border-radius Property](#_grk8x3rv0mzc)

[border-radius](#_v7b8jlywgfa8)

[display Property](#_qnrmv2bu8lgl)

[display: inline](#_pnlzjx500iai)

[display: block](#_6epfvuhbvfq0)

[display: inline-block](#_3agjxznjeicd)

[display: none](#_o4ms8owbln3m)

[padding & margin Properties](#_wb17b82sv4u)

[padding](#_6e4q39ppdgsf)

[border](#_8nv3w4lx3ijp)

[margin](#_nu6a6ta1tt4c)

[Box Model](#_9pvgv0ngmo8o)

[box-sizing: border-box](#_ot0zj1ejm4qq)

[box-Sizing Property](#_xqmbtd6xbiew)

[box-sizing: border-box](#_tg2iclvftmqw)

[Review](#_lv5hzm1ujw6y)

[Hands-On Exercises](#_bjif9z942tsq)

[Hands-On Exercises - Solutions](#_x7wkc28nkaww)

[CSS Selectors](#_3tyvexgqx7h7)

[Section Overview](#_3fmwg5agr1m1)

[CSS Resets](#_wln18ksopxbq)

[Selectors: element, class, id](#_yez1b6w9cmo4)

[element](#_bf5bxx1hwepn)

[.class](#_bf5bxx1hwepn)

[#id](#_bf5bxx1hwepn)

[Selectors: attribute](#_ap8urghu7i15)

[[contenteditable]](#_u193xtwehncv)

[[src]](#_u193xtwehncv)

[[href]](#_u193xtwehncv)

[Selectors: pseudo-classes](#_wd7gpqap5rgq)

[:link](#_nqqlenk1u5jn)

[:visited](#_q5q67gn509g3)

[:hover](#_y3erg6tpga8l)

[:active](#_9ccbzdsjuo81)

[:focus](#_snymlqdnbrb6)

[Selectors: pseudo-class :focus](#_dngit5kgx65v)

[:focus](#_h48qg6n3jcc)

[Selectors: pseudo-class nth child - part I](#_vknr8sjrrlbh)

[first-child](#_en350utuj95o)

[last-child](#_en350utuj95o)

[nth-child(even)](#_en350utuj95o)

[nth-child(odd)](#_en350utuj95o)

[Selectors: pseudo-class nth child - part II](#_pk138ekidfk1)

[nth-child(n)](#_en350utuj95o)

[nth-last-child(n)](#_en350utuj95o)

[nth-child(an+b)](#_en350utuj95o)

[only-child](#_wog70fm5oyxf)

[Selectors: pseudo-class typography](#_1ljxpikwbtnj)

[first-letter](#_6u7tf39msw7q)

[first-line](#_6u7tf39msw7q)

[Selectors: nested selectors](#_oib4ct41o8ef)

[div p](#_b4ufq1syhhb4)

[div > p](#_e58mwk7504hl)

[div ~ p](#_tybwt7grnw2y)

[div + p](#_vu5emr68fu6m)

[margin: 0 auto & The Display Property](#_fx8gph3u4b15)

[Hands-On Exercise - The Surfer Page](#_maitoai2sjx5)

[Hands-On Exercise - The Surfer Page - Solution](#_dc1fo8g6l6aq)

[Hands-On Exercises](#_hms4nubipzle)

[Hands-On Exercises - Solutions I](#_9oait09gkely)

[Hands-On Exercises - Solutions II](#_jq4rzcce7ctb)

[div p](#_qhaenud5ug4b)

[div > p](#_j0vd0yjl2499)

[div ~ p](#_hhoms5kcvmmz)

[div + p](#_xo51fw6lmjgj)

[Review](#_ur4ttrjqfh4d)

[CSS Order & Specificity](#_6x0ykvjxo4eo)

[Section Overview](#_6zai7wko07ij)

[The Browser’s Application of CSS](#_2vafv1gel3fj)

[Challenge & Solution #1](#_6g6otso3q2ku)

[Challenge & Solution #2, 3, 4](#_w5xto6dq0x6)

[Challenge & Solution #5, 6](#_q2tuockx6c9i)

[Hands-On Exercise](#_lebnwzfwf4y5)

[Hands-On Exercises - Solutions](#_94it12koy0s0)

[Review](#_bjjtr1s42elu)

[Formatting Text](#_ujsqlnkwmod9)

[Section Overview](#_88g65i7z7la1)

[font Property](#_so7nheg1da85)

[font](#_q5fpbt9scoib)

[font-family](#_y2mcqaab5md)

[font-size](#_5605p9zrh3w)

[font-weight](#_9k31trau2c7t)

[font-variant](#_g2pj6xmyn29y)

[line-height](#_rdjpn08tmcpf)

[font-style](#_b00vivroqvjp)

[font-family Property](#_5adseeyc0mt3)

[font-size Property](#_4waohhd6fndt)

[font-weight & font-style Properties](#_fjxled4wn0we)

[font-variant & text-transform Properties](#_lqiyj1q3fhak)

[text-transform](#_t8jgunzc34rz)

[line-height Property](#_yauwl82jfox5)

[letter-spacing, word-spacing, & color Properties](#_zgapz0fyepaz)

[letter-spacing](#_rljx9iri7m77)

[word-spacing](#_frm8i45f3okd)

[color](#_b5wk8kwcx7g2)

[Google Fonts](#_5vzp43k8ojo8)

[text-align & text-shadow Properties](#_3ihfebnk9jt5)

[text-align](#_n3zywgcmy8lw)

[text-shadow](#_2beuf995biv3)

[text-decoration Property & <del>](#_q4t5w8xi419v)

[text-decoration](#_8clnj5vqwf9i)

[<del>](#_bb5l19k69m24)

[text-indent Property](#_4oexst2117bk)

[text-indent](#_f2j2u4ia9lvv)

[Hands-On Exercises](#_zg965u83pqs8)

[Hands-On Exercises - Solutions](#_z8sgmyaq3xl4)

[Review](#_tdu1o31lz8jy)

[FREE PREVIEW this video](#_hyay43ly6zx5)

[Structuring Documents](#_8diy9mjhdbxz)

[Section Overview](#_3y8aho3va4a4)

[FREE PREVIEW this video](#_1vwd5au2z7a5)

[Document Structure - Terminology](#_2j2gqhrm7cpq)

[Semantic HTML](#_p1fjqe36hums)

[FREE PREVIEW this video](#_ne3mblryvdsa)

[Hands-On Exercises - Semantic HTML](#_yi9cjwt1zq6b)

[FREE PREVIEW this video](#_yo9bnp6pzzn0)

[Hands-On Exercises - Semantic HTML - My Solutions](#_jdyup2mowolj)

[FREE PREVIEW this video](#_mgot63ib64ur)

[Semantic HTML - header, nav, main](#_fmgpnge0fpk)

[header](#_eew972bvcmex)

[nav](#_wjy5ulcn9bt5)

[main](#_hix843m5kpv6)

[Semantic HTML - article, section](#_ca7maq8vlu06)

[article](#_bve516gcpeer)

[section](#_t28wvoggtuub)

[Semantic HTML - aside & footer](#_wgcleixeo9je)

[aside](#_n6vszf5a9rn)

[footer](#_nlsrfqs3hhy3)

[Semantic HTML - Examples](#_4x4io14bs7dt)

[Semantic HTML - headings - h1 - h6](#_y0lfyb6z3xxt)

[h1 - h6](#_ljcwx3sqbx18)

[Semantic HTML - figure, figcaption](#_2apzolf72eht)

[figure](#_emi58jk88ht0)

[figcaption](#_n2k99591bf6i)

[address](#_o1lqa4vn7afk)

[Hands-On Exercises](#_27np3vij6n2n)

[Hands-On Exercises - Solutions](#_lb7830282jrf)

[Hands-On Exercises - Learning HTML Elements](#_yz2h9qpto8zq)

[Hands-On Exercises - Learning HTML Elements - My Solutions](#_xp9ow473928p)

[Review](#_vf0abeaj6ery)

[Layout Fundamentals - Box Model & Display](#_1dt4ps1bt7d3)

[Section Overview](#_h161pqr3o9j0)

[Layout Resources & Review](#_kuv8a25fqvbk)

[Display Property Review](#_2b9x9fb129yy)

[Hands-On Exercise - Page of Divs](#_p9w3szotr08s)

[Hands-On Exercise - Page of Divs - Solution](#_33chm44bas2x)

[Hands-On Exercise - Horizontal Menu](#_1y8es7ba1sy6)

[list-style](#_c83dbl9pgsh)

[Hands-On Exercise - Horizontal Menu - Solution](#_eq6pi0ebeaxx)

[Taco Shop - Display Property](#_o558systfukq)

[vertical-align](#_iz0dkqgrt8sv)

[Review](#_x9lxqglfjlr)

[Layout with Flexbox](#_aa2dcj504p88)

[Section Overview](#_hrshncal271r)

[Understanding Flexbox - Container & Items](#_fllh5thg2n6m)

[Container properties](#_2mauljt2yvb8)

[display](#_6vzfjaqrjg0g)

[flex-wrap](#_6vhssonljbih)

[flex-direction](#_vo51w1b01png)

[flex-flow](#_sh6wc2mnyzth)

[justify-content (main axis)](#_3194x3vw1bkd)

[align-items (cross axis)](#_7dwe10qzlf0j)

[align-content (cross axis)](#_cp78wpq0g2pu)

[Item properties](#_9d4g7f4lkues)

[align-self](#_jj1a8ieb37eo)

[order](#_nyo4cya8bg58)

[flex-grow](#_9pxvrg1vqdnm)

[flex-shrink](#_rb2g64ac08ob)

[flex-basis](#_qvbgrerm6sn)

[flex](#_j8shbs8uf2hq)

[Containers & Creating A Flex Container](#_d4nybeav6mpk)

[FREE PREVIEW this video](#_bhfqpdoijxs5)

[Container Property: flex-wrap](#_i0y8tkhxrrcc)

[Container Property: flex-direction (primary-axis & cross-axis)](#_utai51j9ja0i)

[Container Property: flex-flow](#_79uz5s2nwdw)

[Container Property: justify-content](#_ttw0l0fhp84f)

[Container Property: align-items](#_br16yw6q17wa)

[Container Property: align-content](#_ycnlhh44wicq)

[Item Property: align-self](#_hlivp1mork86)

[Item Property: order](#_tjoaqdgodgsn)

[Item Property: flex-grow](#_2a7d7c9501h2)

[Item Property: flex-shrink](#_ovedn17r0k4k)

[Item Property: flex-basis](#_osoyiu47g4yt)

[Item Property: flex](#_jd6q8nhy5n4a)

[Hands-On Exercise - Flexbox Froggy](#_byo7c5fqwo74)

[Hands-On Exercises](#_eaj1qzk47m9c)

[Hands-On Exercises - Solutions - 8 Pages](#_kqvexs5196xk)

[Hands-On Exercises - Solutions - 6 Pages](#_z4rbfo8aa68c)

[Review](#_74ac24m6jm82)

[Media Queries](#_5t0q50spaqii)

[Section Overview](#_isbi5n8dqh29)

[Understanding Media Queries](#_fj732r916l4o)

[Min-Width & Max-Width](#_29g176lpxzye)

[Print Media Query](#_7qw3xnnhhz79)

[Media Type & Expressions](#_t0m4yhr14t5)

[Taco Shop & Scaling Font - Examples](#_s4qyzw6d4b19)

[Responsive Design & Mobile First Design](#_mkic6tcdx2k4)

[Hands-On Exercises](#_7dgbci57dfxb)

[Hands-On Exercises - Solutions](#_kn4ajgguvko9)

[Review](#_vxdqgbd2ukis)

[Flexbox Design Patterns](#_7x9enqsbr62h)

[Section Overview](#_uki4mcgxg510)

[Google’s Flexbox Design Pattern #1](#_ejpxhqnwhusf)

[Google’s Flexbox Design Pattern #2](#_l293xyz299ka)

[Google’s Flexbox Design Pattern #3](#_rewzpjy5a1zn)

[Hands-On Exercises](#_ysjtre2zchlh)

[Hands-On Exercises - Solutions](#_gv9i2oj638wm)

[Flexbox Example #1 - Responsive Menu](#_krl1s6nc80n3)

[Flexbox Example #2 - Above The Fold](#_n84b8l5lmx1h)

[FREE PREVIEW this video](#_gvwc6thoc3iq)

[Flexbox Example #3 - Holy Grail Example](#_5z85wis636tv)

[Flexbox Example #4 - Holy Grail Redux](#_tsaegdmv7ykw)

[Hands-On Exercises](#_tc1cztjapder)

[Review](#_ckevoduukyav)

[Layout with Position](#_pb3p70twq48b)

[Section Overview](#_95pddqk5bm06)

[Layout Essentials - A Quick Reference](#_jjajfb4lrlez)

[FREE PREVIEW this video](#_stww3cl4t3d3)

[Position Overview](#_28gtfnpvm3kj)

[FREE PREVIEW this video](#_p04wjr8et5s)

[Position Fixed](#_y7hgyjtiyyg0)

[Position Relative](#_8cl6oy713co9)

[FREE PREVIEW this video](#_if8nlytxuxsj)

[Position Absolute](#_6dnh4ps8xraa)

[Taco Shop - Position Property](#_1vpsy8w3kkw3)

[Hands-On Exercises](#_pz4j4vey88dq)

[Hands-On Exercises - Solutions](#_ko9gabk4c3bm)

[Review](#_bwedru4mzzvn)

[Layout with Float](#_ezybp49qdfg3)

[Section Overview](#_40s9nw85p15)

[Float Layout & Clearing Floats](#_f7wvfwpq9hu5)

[Float Example - Holy Grail](#_l05dqydjy7g)

[Taco Shop - Float Property](#_cs6v6dlaoafl)

[Hands-On Exercises](#_oyse5rda7612)

[Hands-On Exercises - Solutions](#_agu2ezs989cy)

[Review](#_dwot8ejcm2r5)

[Background](#_lhqbge5et4ui)

[Section Overview](#_vsge088wzx7b)

[Background Color](#_of5271sxz5dr)

[background-color](#_u154qbrmtp04)

[background-image, background-repeat, background-size](#_ae8e9u70pooe)

[background-image](#_g3v5aeriu65y)

[background-repeat](#_rpqbzyexw77d)

[background-size](#_huw8s0yf6pop)

[background-position, background-origin](#_6vmpyocbttgh)

[background-position](#_h5e8soce70gf)

[background-origin](#_x7dy8s25ln7f)

[background-attachment](#_6k4wtl44pjea)

[background-attachment](#_7g7obdeladdv)

[background](#_jwsahjh8hsxs)

[background](#_berx9h558my2)

[Hands-On Exercises](#_kaqtejq9562d)

[Hands-On Exercises - Solutions](#_8ly1sfmcnrbr)

[Review](#_elqst493mkh5)

[Refactor Challenge](#_maxqy6aesv7l)

[Section Overview](#_w710cnb070uy)

[Analyzing The Starting Files](#_xeokw7xb62an)

[inline-block Refactor](#_1dwxxom94hwv)

[flexbox Refactor](#_1chlggxpbblq)

[!important rule](#_4kd3nay7ag9l)

[Full-Page Above-The-Fold](#_makj8my3mrrp)

[Section Overview](#_b6etn7nh1bq1)

[Step #1: HTML Structure](#_cag1uyiqtlot)

[Step #1: HTML Structure - Solution](#_wf03eqj0u5d6)

[Step #2: Layout](#_gyzlgoz2gmq5)

[Step #2: Layout - Solution](#_s9ucn9sf109m)

[Step #3: background image](#_4ayn35mfkgql)

[Step #3: background image - Solution](#_xzrpvj10wkjn)

[Step #4: Format Text](#_q1p41g9m1bsh)

[Step #4: Format Text - Solution](#_bzpeaqehxfvh)

[Step #5: Style The Button](#_4e8n61k22sly)

[Step #5: Style The Button - Solution](#_y9bsx7nn4d3c)

[Step #6: Mobile](#_n8bnj8a02s0k)

[Step #6: Mobile - Solution](#_t5ywy6539447)

[Review](#_tw92i62ucs5f)

[Expanding Your Skillset](#_s7lyfwh19bfw)

[Section Overview](#_nfyk67qhho0k)

[q, blockquote, & cite elements; cite attribute](#_qm0p6ctjynua)

[q](#_owxuc59i59af)

[blockquote](#_owxuc59i59af)

[cite](#_owxuc59i59af)

[cite](#_rvr4k9imcdap)

[pre, samp, code, kbd, abbr elements](#_mbelhnxwe2yf)

[pre](#_xayvwu1belhk)

[samp](#_xayvwu1belhk)

[code](#_xayvwu1belhk)

[kbd](#_xayvwu1belhk)

[abbr](#_xayvwu1belhk)

[HTML Entities](#_eh87tp4becsz)

[Linking Images & Bookmarks](#_5ko2baoei5rv)

[box-shadow Property](#_8g19mc7rixy2)

[box-shadow](#_dopchyy4ksf9)

[vertical-align property](#_upwcfm3c7u64)

[vertical-align](#_p57m7j4glu4)

[mobile devices - mobile history & font-boosting](#_6uk65kbfdqd0)

[mobile devices - meta viewport width=device-width](#_b4fu58iz8w1r)

[mobile devices - meta viewport initial-scale=1](#_7oxvvug2hqfe)

[meta viewport - picture example](#_cz2a5dytyk11)

[Review](#_impck7y1nmbq)

[Graphics](#_yomwepk5x0cw)

[Section Overview](#_r91yfo7hkh3)

[Colors](#_f92k4nnshc16)

[Picking Color Combinations - Adobe Color ( Kuler )](#_yu0ysvisb0x)

[Image Types - Raster/Bitmap & Vector](#_pgumr3to827)

[Font Awesome](#_md83xc8z0s8k)

[Font Awesome - Performance](#_vdk3z9t2r4h8)

[Font Awesome - Instructions on Use Review](#_8llzipoxg4ed)

[Font Awesome to SVG - Adobe Illustrator](#_o09miuncgd01)

[Understanding SVG - Basic SVG Shapes](#_2j4ktqinyyfm)

[SVG Terminology](#_f9e129xrhyxo)

[<svg>](#_qhygydonsffd)

[SVG Path Element](#_nk2gt3ewfy77)

[SVG Path - More Examples](#_9d6jwpjf7g7w)

[SVG Stroke & Fill](#_cq9g27hwaw94)

[SVG - Notes & Resources](#_ig88rfugykk6)

[SVG - Viewbox](#_5flhcu8bwp0)

[Styling SVGs with CSS](#_6rqhlhq7wg3c)

[SVG - Symbol & Use Elements](#_w996qziwop8n)

[Symbol, Use, & CSS](#_5djre8wqlkxn)

[Fieldset](#_3zuohve7k26t)

[Legend](#_3zuohve7k26t)

[Symbol with Multiple Paths](#_8djddc4vc5uo)

[Leave Room For Strokes](#_r07jrt8w9mxb)

[Stroke %](#_7a76dqxms0af)

[symbol & use vs img](#_c7uxi8f0cvaz)

[Review](#_i9a3d71ipt69)

[Full-Page Background](#_6qhgwhh45dam)

[Section Overview](#_86ii1e1gjaod)

[Fixed Header](#_l6bbrzscuenf)

[Fixed Header - Solution](#_t8lk93hzr3ml)

[Header Containers](#_hmsmlfek2jut)

[Header Containers - Solution](#_8l3awgss0arb)

[Make SVG’S](#_3hl3rw3kt0b)

[Make SVG’S - Solution](#_kjawvv0smgb)

[Insert SVGs](#_q6y1ve45fhcu)

[Insert SVGs - Solution](#_673wshkrk0qb)

[Style SVGs - Solution](#_ojnsbwlogm8n)

[Dev Tools Styling & Company Naming - Solution](#_11prr2kufdxk)

[Adding Links & Style - Solution](#_whsud6d7yh06)

[Responsive, Mobile-First](#_5gz3oa5do1ed)

[Responsive, Mobile-First - Solution](#_wd53i3fgwfh5)

[Mobile Menu](#_5p29d7sl82w)

[Mobile Menu - Solution](#_g1dau6w3yob7)

[Review](#_t06pi3sgsx4c)

[Favicon](#_163semljb287)

[Section Overview](#_lyeexssf8i20)

[Create An Image](#_hdma7c7boq5a)

[Generate Favicon](#_624baxye8k63)

[Generate Favicon - Subfolder](#_f5xwdc95cgto)

[Generate Favicon - Clears Cache](#_k401llvob7lg)

[Generate Favicon - Compression](#_sew0nvma1ufb)

[Favicon Essentials](#_6af0scj85w67)

[Full Page Background - Favicon](#_9fnu5wa0n0k9)

[Review](#_uy6qvyfjg63d)

[Flexbox Practicum - Part 1](#_3npy1amxx98m)

[Section Overview](#_k2w6k1afeypw)

[Counter](#_ecc21imeyjdo)

[Counter - Solution](#_j9egz45kf3yx)

[Tab Menu](#_h5md702vg47d)

[Tab Menu - Solution](#_gqnik06bqq9d)

[FREE PREVIEW this video](#_o5osqkoggxqa)

[Header Menu](#_h1xq9scwcbce)

[Header Menu - Solution](#_6ortro9zoptv)

[Review](#_mc62jknvuk4d)

[Transitions & Animations](#_ivn5tyf50mt8)

[Section Overview](#_1p2nwdkadwfx)

[Transitions](#_1c56i4fqzh0v)

[Transition Examples](#_zh7xhxxfbosn)

[Animations](#_f7o6pvygtmln)

[Animation Examples & Hands-On Exercise](#_r3jl3sxt8v84)

[Hands-On Exercise Solution & Another Hands-On Exercise](#_396phtq41w1c)

[Hands-On Exercise Solution](#_178pckrsm0jn)

[Transform & Animations](#_ka583us0n1ye)

[Review](#_fuqqaooydsfa)

[Flexbox Practicum - Part 2](#_v4l3ssowckf)

[Section Overview](#_js4d7iip8ry)

[Footer Saving - Solution](#_aqhco3erqny8)

[Sidebar Menu - Solution](#_jv3oqppqrgpn)

[Welcome Page - Solution](#_ch4seoi49pw7)

[Dial Icon - Solution](#_o4z4hgw38qtx)

[Feature List - Solution](#_gs2a8t17d5do)

[Card - Solution](#_makf3k7s86c8)

[Card Group - Solution](#_ifsobhw5ll32)

[You Are A Hero](#_8btt046d22pi)

[Section Overview](#_widxpl2g7eng)

[Above-The-Fold Flexbox Practice](#_dxwvov1b1qly)

[Side-Bar Fly-Out](#_8ehi0g1wwm2z)

[Linear-Gradient](#_uuuxq6am43xx)

[Audio & Video](#_6ov5a6wxkxi9)

[Markdown](#_nb6u1u7at5n)

[Vendor Prefixes, AutoPrefixer, Task-Runners Gulp & Grunt](#_r5lup4k3z1q4)

[Publishing Your Site - An Overview](#_og6sn6hzwj54)

[Publishing Your Site - Buying A Domain](#_qlqhg8tod89j)

[Publishing Your Site - Google Cloud](#_baeprphjpyx1)

[Publishing Your Site - Google Cloud II](#_6ml261hzr8c)

[Review](#_zhu3kbr7khn5)

[Forms](#_gby2anwj7vki)

[Section Overview](#_ct1ad8xc0kl)

[The Form Element](#_mymwibh7nme7)

[The Input Element](#_9sepswi4xkxt)

[The Label Element](#_2uqr49q4tfds)

[The Input Element - Types](#_kp2f6to5n42a)

[textarea, select, option, optgroup](#_xya1uzd3ylyn)

[fieldset & legend](#_o9d4w613mo0j)

[Form Examples, progress, meter](#_ttx0iw1pvzjo)

[Review](#_aid2nfysgbqw)

[Now Go Build It](#_3vq51uwd5zt0)

[Overview of Three Projects](#_lggm5jqfipv)

[Game Plan - Contractor Website](#_mdgybbg64794)

[Starting Files - Contractor Website](#_r3v88sykyt6e)

[Game Plan - Teaching Website](#_zadiwncr7sl4)

[Starting Files - Teaching Website](#_di1djjwn17bz)

[Game Plan - Travel Website](#_3yee1gdwbtyg)

[Starting Files - Travel Website](#_oqhwke7991fi)

[Farewell - Fare Thee Well](#_y1i7ppepayg)

[What’s Next? And THANK YOU FOR JOINING ME!](#_sq0fzp12hp4g)

[REFERENCE - what we have learned](#_gl8mwv30b3rc)

# Welcome

* Top ten reasons to take this course
  + My credentials
  + Wayback Machine
  + Best Stack
  + Fundamentals
  + Cutting Edge
  + Motivation & Life
  + Q&A
  + Money-back guarantee
  + Good Times
  + My kids get a stay-at-home mom
* get to this document: goo.gl/tPe4K3
* **Files**
  + [**Presentation 001 Reasons to take this course**](https://docs.google.com/presentation/d/1aIPVOYX5tjyomt8WMYLUaRSomfwHQrvUr_u26yaw2kA/edit?usp=sharing)
* **Video: 01 / 12 (scene 01 take 12)**

# 

# Getting Started

## Having Fun #1 - HTTP Status Codes

* Http status codes
  + clients & web servers
  + HTTP status code
    - [Wikipedia](https://en.wikipedia.org/wiki/List_of_HTTP_status_codes)
  + Google chrome
  + developer tools
    - network tab
* Description:

When a web server responds to a request it includes in its response a status code. You can see a list of all of the status codes on wikipedia. Interestingly, there is an HTTP status code 418 which means “I’m a teapot.” This was put into the specification as an April fool’s joke. However, if you go to [www.google.com/teapot](http://www.google.com/teapot), you can see Google’s web servers respond with the HTTP Status code 418.

* **Files**
  + [002 having fun #1 - status codes](https://docs.google.com/presentation/d/1afkjPBZykH1z4bAV62iV6yRqd4WVPMztAnWDdgS5qy8/edit?usp=sharing)
* **Video: 02 / 05 (scene 02 take 5)**

## 

## Having Fun #2 - Making The News

* Google Chrome / right click / inspect
  + developer tools / elements
    - Seeing the html of a webpage
    - Changing the html of that webpage
* Description:

In Google chrome, we are able to view all of the HTML which makes up a webpage. By right-clicking that webpage and choosing “inspect,” we are taken into the “elements” tab of developer tools. In the “elements” tab of developer tools, we can change the HTML which is there and our changes will be reflected in the webpage shown in the browser. This allows us to play practical jokes like changing a headline in the news.

* **Files**
  + [**003 having fun #2 - making news**](https://docs.google.com/presentation/d/1tYkII-nxz8eF1-52m8l9Z1UxQo8hGEJ05qoKcH-toWE/edit?usp=sharing)
* **Video: 03 / 07 (scene 03 take 7)**

## 

## Having Fun #3 - The Dinosaur Video Game

* Google Chrome / developer tools / network
  + “No throttling”
    - Change it to “offline”
    - Go to a web page
    - The dinosaur comes up
    - Hit your spacebar
* Description:

In Google chrome, if you’re offline and try to go to a webpage, you will see a dinosaur letting you know that you are offline. What most people don’t know is that this dinosaur is a video game. If you hit your spacebar, that dinosaur will start running across the desert. Your goal, if you want the dinosaur to stay alive, is to jump each cactus.

* **Files**
  + [**004 having fun #3 - dinosaur**](https://docs.google.com/presentation/d/1TqCyF_AIO4zCzjDWUxJQk8coSiNr9L6WP0yZmidnQSg/edit?usp=sharing)
* **Video: 04 / 01**

## 

## Websites vs Apps - Should You Learn Web Dev of App Dev?

* [video & transcript - Rand Fishkin at Moz](https://moz.com/blog/mobile-web-mobile-apps-invest-marketing-whiteboard-friday)
  + Only the top 25 to 50 apps — and some sources show that just the top 5 apps — are responsible for 80% to 90% of all app usage.
  + the average mobile owner uses about 24 apps per month, 24 unique apps per month and visits between 10 and 30 times as many unique websites in a given month.
  + [Rand’s outline](https://drive.google.com/file/d/0B22KXlqHz6ZNTWdaTEQtU0RlNHM/view?usp=sharing)
* [What the web can do](https://whatwebcando.today/)
* **Files:**
  + [**004-02 Websites vs Apps - Should You Learn Web Dev or App Dev**](https://docs.google.com/presentation/d/1Rrok0IIN2VcZRMXpUjPvqg28DvQh2ztwy8jwSS1cX1k/edit?usp=sharing)
* **Video: 04-02 / 01**

## 

## Your Pathway to a $100,000 / Year Salary

* Code bootcamps
  + Intensive training programs
  + Place graduates in jobs
    - Strong statistics on job placement and salaries
* Description:

Not a lot of people are aware of the opportunities which are available through coding bootcamps. Code bootcamps take individuals who have a foundation in HTML, CSS, and JavaScript then train them to be job ready. When the students in a bootcamp have successfully completed the bootcamp, the bootcamp then places the students in jobs. Coding bootcamps have very compelling statistics for job placement rates and salaries. The best code bootcamps place over 95% of their graduates in jobs that earn over $100,000 / year. You are on your pathway to a $100,000 / year job.

* **Files**
  + [**005 your pathway to 100k**](https://docs.google.com/presentation/d/1H5rWfdq5bPrlKJ8FsOXddh0AenxMOeg5j-Any2ZQcqw/edit?usp=sharing)
* **Video: 05 / 01**

## 

## Course Overview - The Art of Building Websites

* My teaching style
* This is both an art and engineering
  + it is a craft which requires us to learn details and be precise
* Description:

Building websites is both an art and engineering. It is a craft which requires us to know many details. Much like cooking, we must learn all of the possible ingredients which we can add to the dish we are creating. A large part of this course will be learning these “ingredients” - the little things which, once we know them, we can then select from to build a masterpiece. My teaching style in the course will be to give you too much material rather than not enough. I’d rather have you skipping a few videos than wishing there were a

few more.

### FREE PREVIEW this video

* **Files:**
  + [**006 Getting The Code Used In The Course**](https://docs.google.com/presentation/d/1s08H6EiISt2aMhk4wc3CggGtFwjYrVn9Ft26g8dLg58/edit?usp=sharing)
* **Video: 05-02 / take 01**

## 

## Getting The Code Used In This Course

* download the zipped folder of code here:
  + <https://github.com/GoesToEleven/html-css-bootcamp>
* Description:

All of the code for this course will be stored on github. You can download all of the code in this course from the course repository on github. Often people abbreviate “repository” and just say “repo.” Later in the course, you will learn how to use github and create your own repos.

* **Files:**
  + [**006 Getting The Code Used In The Course**](https://docs.google.com/presentation/d/1s08H6EiISt2aMhk4wc3CggGtFwjYrVn9Ft26g8dLg58/edit?usp=sharing)
* **Video: 06\_01 / take 01**

## 

## Stay Current, Find Jobs, Receive Discounts

* Follow me
  + [twitter.com/todd\_mcleod](https://twitter.com/todd_mcleod)
  + [google.com/+ToddMcLeod](https://plus.google.com/+ToddMcLeod)
  + [youtube.com/user/toddmcleod](https://www.youtube.com/user/toddmcleod)
  + [github.com/GoesToEleven](https://github.com/GoesToEleven)
  + [linkedin.com/in/tamcleod](https://www.linkedin.com/in/tamcleod)
* I’m often posting discount codes or free access
  + In fact, you can go to my twitter account right now and search for access codes to my courses

### FREE PREVIEW this video

* Description

Stay current. Find great jobs. Gain discounted or even free access to my courses. Follow me on social media to realize all of these benefits.

* **Video: 06\_02 / take 02**

## 

## Resources

* Description

There is a sea of information on how to learn web programming today. I’ve looked at most of it. Here are the resources which I think are the best for learning web programming:

[these notes](https://goo.gl/YJTNed) ( <https://goo.gl/YJTNed> )

* Learning resources
  + <http://learn.shayhowe.com/>
  + [codecademy.com html/css](https://www.codecademy.com/learn/web)
  + [Google Web Fundamentals](https://developers.google.com/web/fundamentals/?hl=en)
  + [MDN Learning The Web](https://developer.mozilla.org/en-US/Learn)
* layout
  + [box model](http://s3.amazonaws.com/codecademy-blog/assets/ae09140c.png)
  + <http://learnlayout.com/>
  + <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
  + [Layout essentials](https://docs.google.com/document/d/1IjV1IlnEXKPu5IFdWtr9Fubl_RZEVJdmPgaBQOjomU0/edit?usp=sharing)
* reference
  + [MDN](https://developer.mozilla.org/en-US/)
    - [all HTML elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
    - [all HTML attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes)
    - [CSS Reference](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)
  + <http://caniuse.com/>
  + <http://www.webplatform.org/>
* css selectors
  + <http://www.w3schools.com/cssref/css_selectors.asp>
  + <https://css-tricks.com/how-css-selectors-work/>
  + <https://docs.webplatform.org/wiki/css/selectors>
  + <https://docs.webplatform.org/wiki/tutorials/using_selectors>
  + [MDN](https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Getting_started/Selectors)
* tools
  + testing
    - [Google mobile friendly test](https://www.google.com/webmasters/tools/mobile-friendly/)
    - [Google page speed](https://developers.google.com/speed/pagespeed/insights/)
  + craftsmanship
    - [Google webmaster guidelines](https://support.google.com/webmasters/answer/35769)
      * [review](https://support.google.com/webmasters/answer/35769)
  + performance
    - [My Notes: Critical Rendering Path](https://docs.google.com/document/d/1zmBC-oRIMOqccjnUdtjZJEaXcMmmuvfFLHildZNn7Hg/edit#heading=h.ljyisebk4kae)
  + favicon
    - <http://realfavicongenerator.net/>
  + tools
    - <http://codepen.io/>
    - [emmet cheat sheet](http://docs.emmet.io/cheat-sheet/)
    - [google fonts](http://www.google.com/fonts)
    - [Autoprefixer](https://css-tricks.com/autoprefixer/)
    - [remote debugging on android](https://developer.chrome.com/devtools/docs/remote-debugging)
    - [css lint](http://csslint.net/)
    - [JSON editor](https://www.jsoneditoronline.org/)
* stats
  + <http://cssstats.com/>
* staying current
  + [The Web Ahead](http://thewebahead.net/)
  + [Rand Fishkin at Moz](https://moz.com/community/users/63)
  + <http://html5weekly.com/> (front end focus)
* **Files:**
  + [**007 Resources**](https://docs.google.com/presentation/d/1XiVhdMQOe7YEfuHSSur4ROyDAHlpSCmsQPb7Zbu93X4/edit?usp=sharing)
* **Video: 07 / 01**

## 

## How To Succeed

* Keys to success
  + Focus
  + Forethought
  + Habits of effective people
  + Time on task
  + Grit
  + Bill Gates
    - get in front of what’s coming and let it hit you
* Description:

Understanding what has made others successful can help you become successful. These are principles which have helped me become successful. I learned these principles from others and from my own experience. I share these principles to help you succeed in this course and in life.

* **Files:**
  + [**008 How To Succeed**](https://docs.google.com/presentation/d/1tzRbfIaaNNuvcJrFXYvLwN9lPht-WzUV_3eBW9GwAew/edit?usp=sharing)
  + [**Seven Habits of Highly Effective People**](https://drive.google.com/file/d/0B22KXlqHz6ZNRzZUSnRsZldNUEk/view?usp=sharing)
* **Video: 08 / 01**

# Understanding The Internet & World Wide Web

## History of the Internet

* The Internet’s History
  + October 29, 1969
  + Arpanet
  + Nuclear war
  + Circuit network
  + Packet-switching network
  + TCP/IP
* Description:

The Internet was created in the 1960’s. Originally it was called the “arpanet.” The first message was sent on the Internet in 1969. The Internet was created to allow communication to continue in the event of a nuclear war. Before the Internet, people communicated using “switched” networks. The Internet allowed people to communicate using “packet switching” networks. With a “packet switching” network, message are broken up into packets and put onto the network, then “routers” send those packets to their destination via the routes which are still viable. The Internet was created by Darpa which is the Defense Advanced Research Projects Agency.

* **Files**
  + [**009 History of Internet**](https://docs.google.com/presentation/d/1Hj2ItFyW69rGDWn35biONQZelE98vxpWtuZw4cOm7p8/edit?usp=sharing)
* **Video: 09 / 01**

## 

## History of the World Wide Web

* The WWW’s History
  + Tim Berners-Lee
  + CERN
  + 1980 - Hypertext
  + 1989
  + 1990 - HTTP & HTML
  + Web browsers
  + 1991 - web switched on
  + Alex page rank
* Description:

Tim Berners-Lee, an engineer at the Center for European Nuclear Research (CERN), conceived of the World Wide Web (WWW) in 1980. The WWW was launched in 1990. In 1991, the first web server outside of CERN was launched. The WWW was created to allow people to format text shared on the Internet, and to also allow them to use hypertext. Hypertext is text that has hyperlinks. A hyperlink is a link which takes you to some other resource on the WWW. It is important to know that the Internet and the WWW are two separate things. Even though most people use “Internet” and “WWW” interchangeably, the Internet and WWW are indeed two separate things. You can think of the Internet as all of the hardware: the computers, servers, routers, modems, cables, and satellites. You can think of the WWW as a service which runs on the Internet.

* **Files**
  + [**010 History of WWW**](https://docs.google.com/presentation/d/1SeSleuYPyGorCWVaIEANDJTfIltXmm6OlM00Ci2ZOdI/edit?usp=sharing)
* **Video: 10 / 05**

## 

## Who Controls the Internet & WWW?

* The Internet and WWW are controlled by anyone
  + Several organizations influence its growth
    - Standards organizations
      * W3C
      * WHATWG
      * IETF
      * Others
        + Telecom companies
        + Innovators
    - Telecom companies
    - Legal jurisdictions
    - Innovators
* Description:

The Internet and WWW aren’t controlled by any one organization nor individual. Instead, there are many different organizations and individuals who influence the development of the Internet and WWW. The main standards organizations which influence the direction of the Internet and WWW include the W3C, the WHATWG, and the IETF.

* **Files**
  + [**011 Who controls the Internet & WWW**](https://docs.google.com/presentation/d/1wWEdt__h2b3nn1PYRfrQke0I0IfGrcey8lzmbsfMTI0/edit?usp=sharing)
* **Video: 11 / 01**

## Web Documentation

* MDN
* webplatform.org
* w3schools.com
* caniuse.com
* stackoverflow.com
* codepen.io
* csstricks.com
* The Web Ahead
* Description:

MDN is the ultimate source for web documentation. MDN stands for Mozilla Developer Network. Webplatform.org might also soon be a good source for web documentation. W3schools.com can sometimes be useful, but their material is also at times incorrect. Stackoverflow.com is a good place to ask questions and read answers to questions that have been asked in the past. Codepen.io is a good place to look for examples of code. Csstricks.com has good articles and is well liked in the industry. The web ahead offers good podcasts on creating websites.

* **Files**
  + [**012 Web documentation**](https://docs.google.com/presentation/d/1nFK-KtuvszABSjWjJ1vb4v8SxNCxDkqVybQ7EWKz43E/edit?usp=sharing)
* **Video: 12 / 01**

## 

## Highlighting Webpages & Chrome Extensions

* [Super Simple Highlighter](https://www.google.com/#q=Super+Simple+Highlighter)
* Description:

Google Chrome allows you to add extensions. Extensions extend the capabilities of chrome. You can learn about Chrome extensions by searching Google for “chrome extensions”. The “super simple highlighter” is an extension which allows you to highlight webpages. Once you have extensions, you can manage them by going to “settings” in Chrome and then “extensions.” You can check whether or not extensions are “enabled” and you can also delete extensions.

* **Files**
  + [**012-02 Highlighting Webpages & Chrome Extensions**](https://docs.google.com/presentation/d/1PuB1spaG73YkdabSHt_6QIiOIsV5PHVgPmm59hmpEtc/edit?usp=sharing)
* **Video: 12-02 / 01**

## 

## How The Web Works

* client / server architecture
* request / response pattern
  + header & body
* Description:

The web uses a client / server architecture. Clients make requests to servers, and then servers respond to clients. After the request / response cycle, the connection between the client and the server is severed. The web is known as being “stateless” because of this - state is not maintained, eg, an ongoing connection between the server and client are not maintained. For both the request and the response, each of them have both a header and a body. The header includes information pertinent to the communication between the devices, but not usually interesting to the user. The body includes the main “payload” of the communication: either data the client is sending to the server, or the webpage the server is sending to the client.

* **Files**
  + [**013 How the web works**](https://docs.google.com/presentation/d/1ZOlP9E73765QWGqckk2uPRdhCqci_YJlO-8hJCc_n7s/edit?usp=sharing)
* **Video: 13 / 01**

## 

## The Dark Web

* TOR
* TOR browser
* illegal commerce
* [Jamie Bartlett’s talk](https://www.youtube.com/watch?v=pzN4WGPC4kc)
* Description:

The dark web is an anonymous web. It is built using The Onion Router (TOR). Tor helps conceal your identity by routing requests through many routers with each router only knowing the router before it and the router after it. The dark web hosts many websites which sell illegal goods.

* **Files**
  + [**014 The Dark Web**](https://docs.google.com/presentation/d/1Qo6VFMblzmim2NzaVCC928d-lfFBuZg0GV8JjCcfoT4/edit?usp=sharing)
* **Video: 14 / 01**

## 

## BitTorrent

* P2P file sharing
* BitTorrent client
* BitTorrent trackers
* Description:

BitTorrent is peer-to-peer file sharing. Using BitTorrent you can download legal, and illegal, digital goods. To use BitTorrent, you need a BitTorrent client. Once you have a BitTorrent client, you find a torrent file that you want to download, then download it through your client. To find a file you want to download, you can browse sites known as “torrent trackers”. Remember, though, if you break the law there are consequences.

* **Files**
  + [**015 BitTorrent**](https://docs.google.com/presentation/d/185GJpJ6GlJsM6KaSFs6OAGwcutMJVYM7JYr3AVUAkL4/edit?usp=sharing)
* **Video: 15 / 01**

## 

# An Introduction to HTML

## 

## Your First Webpage

* codepen.io
* Emmet.io
  + [emmet cheat sheet](http://docs.emmet.io/cheat-sheet/)
* Integrated Development Environments (IDEs)
* github
* Description:

We will build your first webpage at codepen.io. Eventually we will use an Integrated Development Environment (IDE) to build our webpages. The IDE we will use in this course is webstorm. I will also introduce you, however, to the IDE Atom in this course as Atom is very popular. When I introduce the IDEs later in the course, I will also discuss the differences between Webstorm and Atom and share with you why I chose Webstorm. You are also able to access the code in this course, and for this exercise, through github.

* **Files**
  + [**016 Your first webpage**](https://docs.google.com/presentation/d/1BymDM1bSpWYHRiYTx8rUPy-nk_YvlgQW_VgMMtNSFNw/edit?usp=sharing)
* **Video: 16 / 01**

## 

## Anatomy of an HTML Page

* document type declaration (DTD)
  + <!DOCTYPE html>
* html tags
  + <html lang=“en”>
  + <head>
  + <meta charset=“UTF-8”>
  + <title>
  + <body>
* Description:

An HTML document is made up of a document type declaration (DTD) and different tags. The DTD is declared at the top of the page: <!DOCTYPE html> …. The different elements required for a basic HTML document: html, head, meta, title, body. We use MDN as our source for documentation to learn about the different elements.

* **Files**
  + [**017 Anatomy of an HTML Page**](https://docs.google.com/presentation/d/1dKEV-kJB93SXePK7yHNlKxXrC4mH1oLzC8QBh-3tBkc/edit?usp=sharing)
* **Video: 17 / 01**

## 

## HTML Terminology

* To do understand webpages, you must understand the terminology used to talk about webpages:
  + Elements vs tags
  + Opening and closing tags
  + Self-closing tags
    - <meta>
    - <link>
    - <hr>
    - <br>
    - <img>
    - <input>
    - <source>
    - <embed>
    - <param>
    - <wbr>
  + Nesting tags
    - parent & child
    - ancestor & descendant
    - siblings
  + Root
  + Synonyms
    - web page / html document / html page / document
    - element / tag
    - parent / ancestor
    - child / descendent
* Description:

To be specific, when we talk about an element, we are talking about the html element which helps us build an html document. When we talk about a tag, we are talking about an element being used in the html document. A tag will have brackets around it. Many tags also have a closing tag. Some tags do not have a closing tag, and we call these tags “self-closing” tags. To build an html document, we will nest tags. To discuss the relationship between nested tags, we use terms like “parent / child / ancestor / descendant / sibling.” We call the tag at the top of the document the root tag or root element.

* **Files**
  + [**018 HTML Terminology**](https://docs.google.com/presentation/d/109Zj6H0EIlFdxz6Z4qYGdMi2-w0JKn7jKcIU36fG9HA/edit?usp=sharing)
* **Video: 18 / 01**

## 

# An Introduction to CSS

## A Separation of Concerns

* Separation of concerns
  + **HTML - the structure**
    - semantic HTML
    - avoid div-itis
  + **CSS - the formatting**
    - rule-set
    - selector
    - declaration block
    - declarations
    - property
    - Value
  + **JavaScript - the functionality**
    - The functionality
      * client-side programming
    - Try not to use it
* Description:

In programming, separating concerns allows for a more stable and robust program. Separating concerns means to means to separate a program into distinct sections. In web programming, we use the concept of separating concerns to separate structure and formatting. HTML is responsible for structuring our document. CSS is responsible for formatting our document.

* **Files**
  + [**019 HTML & CSS - A Separation of Concerns**](https://docs.google.com/presentation/d/1cdyc3VfQdNlZWuqIBLLygjJVuzBSz5VO01-c931E838/edit?usp=sharing)
* **Video: 19 / 01**

## 

## CSS Rule-sets

* rule-set
  + selector
  + declaration block
    - declarations
      * property
      * value
* Description:

We use rule-sets in CSS to select html tags in our html document, and then apply fomatting rules to those selected tags. A CSS rule-set consists of a selector(s), a declaration block, and declarations. Each declaration consists of a property and value.

* **Files**
  + [**020 CSS Rule-set**](https://docs.google.com/presentation/d/1ZX8L78pA2LVXTnxOZ-9tr9iMZna21IhZhASvKBMajmA/edit?usp=sharing)
* **Video: 20 / 01**

## 

## Linking CSS to HTML

* external style sheet
  + preferred method
* internal styles
* Inline styles
* Description:

There are three ways you can link CSS to HTML. You can link an external CSS style sheet to your HTML. This is the preferred method to connect your CSS to your HTML. YOu can also link your CSS to your HTML using internal styles. To use internal styles, in between your head tags, you include an opening and closing style tag ( <style> </style> ). You then put your CSS rule-sets in between these style tags. The last way you can connect your CSS to your HTML is to place your CSS inline in the HTML tag. This method is strongly not recommended as it is not modular and it is hard to maintain.

* **Files**
  + [**021 Linking CSS to HTML**](https://docs.google.com/presentation/d/1Qf0meQ6jX7TmHaLi5S4dytE_Ul5iSVAbiHhnJsgtNe0/edit?usp=sharing)
* **Video: 21 / 01**

## 

## Multiple CSS Selectors

* Many rule-sets
* Multiple selectors for one rule-set
* The everything ( \* ) selector
* Description:

You can use many rule-sets in one style sheet. You can also use multiple selectors in one rule-set. The everything selector allows you to select all html tags in an html document.

* **Files**
  + [**022 Multiple CSS Selectors**](https://docs.google.com/presentation/d/1e_E2Lp1SGf13_GrKQD_6cE0_5AmtN1qcI57aa9-hxzI/edit?usp=sharing)
* **Video: 22 / 01**

## 

# Configuring Your Environment

## Integrated Development Environments

* Separating the ...
  + Wheat from the chaff
  + The men from the boys
  + The women from the girls
  + Those who can do this, from those who have no grit
  + Now is the time for grit
    - This can be challenging for some
    - This is the work, this is how you learn, by facing these challenges and working at them until you overcome them
* Integrated Development Environments (IDEs) allow you to write code more easily
  + code highlighting
  + code formatting
  + code completion
  + live preview
* Different IDEs
  + Webstorm
  + Atom.io
  + Sublime
  + Dreamweaver
* Github
  + Important part of any work environment
  + We will use it through the command line interface
* Description:

Choose one editor and get good at it. You don’t need to have all three of these editors, or even watch all three of the editor videos. Just choose one editor. That’s it. Just one. I use webstorm, and in the next video I will tell you why and show it to you. One thing to know: it takes awhile to get good with an IDE. I recently had the experience of driving my friend’s car. And I couldn’t figure out how to do things. I had to stop and spend time trying to find XM radio again. It’s the same way with an IDE. It takes you awhile to be able to drive it quickly and well.

* **Files**
  + [**023 Configuring Your Development Environment**](https://docs.google.com/presentation/d/1ODuDlkgBPqSSeiPO3Mpj7YqsXZ_q_JAqp7-pxyBK-mw/edit?usp=sharing)
* **Video: 23 / 01**

## WebStorm

* Live preview
* Getting your webstorm to look like my webstorm
* Description:

Webstorm is an IDE created by JetBrains. I chose Webstorm because JetBrains also creates an Intellij, and Android Studio is built on top of Intellij. By knowing Webstorm, it’s easier to understand Android Studio. So if you want to build Android Apps somewhere down the line, then using Webstorm is a good choice. I also find Webstorm to provide all of the functionality that I need.

* **Files**
  + <http://color-themes.com/> - [relax your eyes](http://color-themes.com/?view=theme&id=563a1a6e80b4acf11273ae76)
    - Install the theme: <http://color-themes.com/?view=help>
  + [font - Bitstream Vera Sans Mono](https://goo.gl/JWK8fK)
  + [my settings for webstorm which you can import](https://drive.google.com/folderview?id=0B22KXlqHz6ZNcXdtR0w1VmZodzA&usp=sharing)
    - install the font
    - use “import settings” in webstorm
  + live preview
    - you will also need this: [chrome jetbrains plugin](https://chrome.google.com/webstore/detail/jetbrains-ide-support/hmhgeddbohgjknpmjagkdomcpobmllji?hl=en)
* **Code**
  + **005\_webstorm-live-preview**
* **Video: 24 / 01**

## 

## Atom.io

* running from the command line on Mac
* opening your project folder
* live preview
* Description:

Atom.io is the new kid on the block. It is very compelling. It was created by github. The people at github know what they’re doing. It is now also open source, which means many talented people are helping it grow. There are also a lot of packages you can get for it.

* **Video: 25 / 01**

## 

## Sublime

* Sublime was the industry standard in the mid-2000’s
* Now sublime is the old-dog on the block
  + It will still work fine, however, have there are stronger arguments in favor of using atom.io and webstorm
* Description:

While this video talks about Sublime as a “has-been,” the people who make Sublime have recently updated the editor making it a viable option today.

* **Video: 26 / 01**

## 

## Dreamweaver

* Dreamweaver was the industry standard for website creation in the late 1990’s and early 2000’s.
* Dreamweaver is a WYSIWYG (weeeseeewigggeeee) tool - what you see is what you get
  + It allows you to drag-and-drap items around and dreamweaver writes the code for you
* As dreamweaver writes the code for you, it can be a big crutch that prevents you from writing and learning the code yourself
* It is strongly recommended that you do not use dreamweaver
  + Most professional website builders today do not use dreamweaver.
* Description:

I just have to mention Dreamweaver because I don’t want you to fall in its evil embrace. Dreamweaver is like the drug dealer who gives crack cocaine to people, gets them hooked, then sucks all of the money out of them as their lives are destroyed. Dreamweaver will make you feel very good at first, but ultimately it will make you a pathetic and insecure coder who doesn’t understand anything.

* **Video: 27 / 01**

# 

# Becoming Skilled with Github

## Github Overview

* The difference between git and github
  + The story - I will tell it to you in the next video.
* Description:

Github is like facebook but for code: it allows you to share your code with others. Just like you share moments of your life on Facebook, so too can you share moments from your coding on Github. Why would you want to do this? There are many reasons to use github, but here are three:

1. it allows others to have insight into who you are as a coder
2. it is Version Control Software (VCS) which allows you to collaborate easily with others
3. it is a repository for storing your code online.

There are many ways to use github, but I strongly recommend you use github through the command line. We will learn how to use the command line, which is also known as the terminal, in the next video. For coders this axiom holds true: I use github therefore I am.

* **Video: 28 / 01**

## 

## Git & Github - The Story

* The difference between git and github
  + The git story
  + [How linux is built](https://www.youtube.com/watch?v=yVpbFMhOAwE)
* Description:

Don’t mess with Linus Torvald, that’s the moral of this story. Linus created the Linux operating system, the world’s largest open-source project (see youtube video: how linux was built). To create Linux, they needed to use version control software (VCS). This type of software helps you build software: it allows many different people to smoothly work on the same project. Well, for many years, they used Bitkeeper. Then, in 2005, Bitkeeper said they wanted to charge Linux. Linus told them to go pound sand and created Git. Git and github have now effectively put Bitkeeper out of business.

* **Video: 2802 / 01**

## 

* **REMOVED LECTURES**
  + **From the numbering system, these lectures were intentionally removed as the topics were covered in the introduction video:**
    - **29**
    - **30**
    - **31**

## 

## Configuring Windows for Github

* This is only for Windows
  + Only watch this video if you are using a Windows machine
    - Which is one of the reasons why I’m making this course on a Windows machine. As most people use a Windows machine, I want to be able to show you this
  + We will need to install a terminal emulator
    - github’s software
    - <http://babun.github.io/>
* Install git
  + Here are the choices you make in the installation process:
    - “Select components” dialogue box
      * Use the default selections
    - “Adjusting your path environment” dialogue box
      * Use git and optional unix tools …
    - “Configuring the line ending conversions” dialogue box
      * Checkout as is, commit unix style
    - “Configuring the terminal emulator to use with git bash”
      * Use MinTTY …
    - “Configuring extra options …”
      * Use the default selections
* Install github desktop
  + <https://desktop.github.com/>
  + Configure the settings on github installation:
    - Login
      * You will need to have a github account
        + Create one at the github website if you don’t already have an account
    - Settings / options (click the gear icon)
      * change “DEFAULT SHELL” to “git bash”
* In Windows, search for “git bash” then run it
* Description:

To get github to work well on a windows machine, there are two pieces of software which we must install: git and github desktop. As we will be using the terminal for Github, you will need a terminal emulator on your machine. Github desktop comes with a terminal emulator.

* **Video: 32 / 01**

## 

## Configuring Mac and Linux for Github

* Should already have git
  + Try and see
    - git --version
* If it doesn’t work
  + Install git
  + Install github desktop
* Description:

Github should work on Mac, Linux, and Posix machines. Follow the videos below. If you hit issues, try this: install git, then install github desktop.

* **Video: 33 / 03**

## 

## Terminal Essentials

* basic commands
  + pwd
  + cd
    - cd /
    - cd ../
  + ls
    - ls -la
  + clear
  + rm file-name
  + rm -rf dir-name
  + mkdir dir-name
  + cat
    - prints out contents of a file
* Description:

There are a few basic commands which allow you to use the terminal CLI well.

* **Video: 34 / 02**

## 

## Creating a “Master” Local Git Repository

* Create a folder
* Add a .gitignore file
* At the terminal
  + git init
  + git status
  + git add --all
  + git commit -m “first commit”
  + git log
* Description:

To create a local git repository

* **Video: 35 / 02**

## 

## Creating an “Origin/Master” Remote Github Repository

* Create a new repo
  + name it the same as the folder name of your local git repo
    - Not required, but logical
  + copy / paste instructions
    - navigate to the root of your local git repo
    - copy / paste the code given to you - one line at a time - into the terminal
      * Your code will look something like this:

git remote add origin [git@github.com](mailto:git@github.com):GoesToEleven/webdog.git  
git push -u origin master

* [master vs origin/master](http://stackoverflow.com/questions/18137175/in-git-what-is-the-difference-between-origin-master-vs-origin-master)
  + The “master” branch
  + “origin/master” means “the master branch on the origin”, the source from which everyone is contributing
* git push
* Description:

We use github to create a remote github repository. There are many ways to do this. I am going to show you what I believe is the simplest, and best, method.

* **Video: 36 / 03**

## 

## Using Github - Basic Commands

* Here are the basic github commands which you will use 98% of the time:
  + git status
  + git add --all
  + git commit -m “some concise descriptive message”
  + git push
* Description:

As we code, we will want to “push” our code and store it remotely on github. This will keep track of all of the code we are writing. To do this, we follow a few simple commands: git status, git add --all, git commit -m “some message”, git push.

* **Video: 37 / 02**

## 

## Using Github - Deleting A Repo

* Delete locally
* Delete remote origin on github
* Description:

As we code, we will want to “push” our code and store it remotely on github. This will keep track of all of the code we are writing. To do this, we follow a few simple commands: git status, git add --all, git commit -m “some message”, git push.

* **Video: 38 / 01**

# 

# HTML Essentials

## Section Overview

* FIle Naming Conventions
* Folder Naming Conventions
* Opening a Project in Webstorm
* Essential Tags
  + Heading elements
  + Paragraph element
  + Unordered list element
  + Ordered list
* Emmet.io
* Tag Attributes
  + Link element
  + Image element
  + Anchor element
  + Paragraph element
* Relative vs Absolute URLs
* Comments
* Hands-On Exercises
* Description

In this section, we will learn some of the essentials of working with HTML.

* **File:** 
  + [**039 File Naming Conventions**](https://docs.google.com/presentation/d/1Z2LAz73oJ5OOjqFgpMw7vsauwOrzuXgG4eB93FEkBJI/edit?usp=sharing)
* **Code:**
* **Video: 39-01 / take 01**

## 

## FIle Naming Conventions

* [HTML File Naming Conventions](http://thenewcode.com/30/HTML-naming-conventions-and-file-management)
  + HTML pages should be saved with the **.html extension**
  + 90% of the time the home page of your site will be named **index.html** in order to be picked up by the browser as the default document.
  + Use **alphanumerics** only in page names. That is, a-z, 0-9. The only exceptions are: **-(dash), \_(underscore)** and ~ (tilde).
  + **Never use spaces** in the file name of anything destined for the web, as spaces will be encoded by the server, producing long and ugly URLs. Replace spaces with hyphens.
  + Use **lowercase** exclusively when naming files, as some web servers are sensitive to case.
* Description

It is important to know how to name your files. There are certain standard naming conventions which are used. You should always use lowercase alphanumeric characters to name files, eg, a-z, 0-9. You should never use spaces in your file names. The main page for any website should always be index.html.

* **File:** 
  + [**039 File Naming Conventions**](https://docs.google.com/presentation/d/1Z2LAz73oJ5OOjqFgpMw7vsauwOrzuXgG4eB93FEkBJI/edit?usp=sharing)
* **Code:**
* **Video: 39-02 / take 01**

## 

## Folder Naming Conventions

* [HTML Folder Structure](https://css-tricks.com/forums/topic/web-folder-structure/)
  + css
  + js
  + imgs
  + pics
* Description

Folders help you organize your code. You will want to organize your code into logical groupings. You can think about this like you think about the separation of concerns. Just like we want to separate our structure from our formatting from our functionality - our HTML, our CSS, our JS - so too we could create folders to hold those separate files. When creating folder names, always use lowercase alphanumeric characters, and never use spaces.

* **Files:**
* **Video: 40 / take 03**

## 

## Opening A Project in Webstorm

* Opening a project in webstorm
  + Ignoring webstorm’s request to manage your git VCS commits and pushes
* Description

Since we’ve already created a folder to hold our code and initialized it as a git repository, all we need to do now is open that folder in webstorm. When you start webstorm, choose “open” and then open that folder. If webstorm prompts you to manage all of your git commits and pushes, just close this dialogue box. I will show you how to use webstorm to do this at the end of the course. For now, however, I want you using the terminal to do all of your commits and pushing to github.

* **Video: 41 / take 02**

## 

## Essential Tags

### Paragraph element

* + p

### **Heading elements**

* + h1 - h6

### Unordered list element

* + ul
  + li

### Ordered list

* + ol
  + li
* emmet
  + Sibling selector
    - +
  + Child selector
    - >
  + Multiplier
    - \*
* Description

Using webstorm, we’ll create a new folder “005\_essential-tags”. Inside this folder, we will create four new folders to hold each of the new tags we’re going to learn about: p, h1-h6, ul, ol, li. We will use emmet to quickly write this code for us. We will also use several webstorm shortcuts which can be referenced on webstorm’s keymap reference under help inside webstorm. Finally, we will commit and push all of our new code to github.

* **files:**
  + [**005\_essential-tags**](https://github.com/GoesToEleven/html-css-bootcamp/tree/master/005_essentials-tags)
* **Video: 42 / take 02**

## 

## Emmet.io

* [emmet cheat sheet](http://docs.emmet.io/cheat-sheet/)
  + Sibling selector
    - +
  + Child selector
    - >
  + Multiplier
    - \*
  + Text
    - {text content here}
  + Numbering
    - $
  + HTML 5 document
    - !
* Description:

Emmet can help us write code more quickly. The emmet cheat sheet provides us with examples we can use to quickly create different code.

* **File:** [**006\_emmet**](https://github.com/GoesToEleven/html-css-bootcamp/tree/master/006_emmet)
* **Video: 43 / take 01**

## 

## Modifying Webstorm Formatting

* How to change webstorm formatting preferences
* Description:

My teaching style: I’d rather give you too much than not enough; I’d rather you be able to skip a video than to be wishing there was a video. To that end, here is how you change the settings in webstorm to format html differently.

* **Video: 44 / take 01**

## 

## Tag Attributes

* tags may have attributes
* attributes have values
  + [MDN - Getting Started > HTML Basics](https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/HTML_basics)
* examples:

### Link element

* + - <link rel="stylesheet" href="main.css">
    - self-closing

### Image element

* + - <img src="puppy.jpeg" alt="a puppy with a stick">
    - self-closing

### Anchor element

* + - <a href=”[www.google.com](http://www.google.com)” target=”\_blank”>go to google</a>

### Paragraph element

* + - <p class="happiness">I love puppies!</p>
* Description

HTML tags can also have attributes. Some HTML tags require certain attributes to work. Attributes are additional pieces of information which the browser uses. Attributes help us provide additional functionality to our HTML tags.

* **Files:** [**007\_tag-attributes**](https://github.com/GoesToEleven/html-css-bootcamp/tree/master/007_tag-attributes)
* **Video: 45 / take 05**

## 

## Relative URLs

* relative
  + within one domain / website: relative to one resource, where is the other resource
  + examples:
    - pic/anatomy-of-an-html-element.png
    - ../pic/anatomy-of-an-html-element.png
* [MDN documentation on relative and absolute URLs](https://developer.mozilla.org/en-US/Learn/Common_questions/What_is_a_URL#Absolute_URLs_vs_relative_URLs)
* **File:** [**008\_relative-urls**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 46 / take 02**

## 

## Absolute URLs

* absolute
  + full URL
    - https://developer.mozilla.org/en-US/docs/Learn
  + from the root of a site, where is the resource
    - *“If the path part of the URL starts with the "/" character, the browser will fetch that resource from the top root of the server, without reference to the context given by the current document.” (source:* [*MDN*](https://developer.mozilla.org/en-US/Learn/Understanding_URLs#Absolute_URLs_vs_relative_URLs)*)*
    - /pic/anatomy-of-an-html-element.png
* [MDN documentation on relative and absolute URLs](https://developer.mozilla.org/en-US/Learn/Common_questions/What_is_a_URL#Absolute_URLs_vs_relative_URLs)
* **File:** [**009\_absolute-urls**](https://github.com/GoesToEleven/html-css-bootcamphttps://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 47 / take 01**

## Comments

* HTML Comments
* CSS Comments
* Description

It is important to know how to add comments to code. Code with comments is better code. Comments allow you, and others, to understand what your code is doing. There is an art to writing code comments: you don’t want too many, nor too few.

* **Files:**
  + [010\_code-comments](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 48 / take 01**

## 

## Hands-On Exercises

* Challenges
  + (1) Create two files: an html file and a css file. Name the files using standard naming conventions. Link the css file to the html file.
  + (2) Take the css file you created in the previous challenge and put that css file in a folder. Use standard naming conventions for the folder. Make sure the index.html file and the main.css file are still linked.
  + (3) [Create this page.](https://github.com/GoesToEleven/html-css-bootcamp/blob/master/011_hands-on-exercsises-solutions/3.PNG) Use emmet to build your html structure.
  + (4) [Create this page.](https://github.com/GoesToEleven/html-css-bootcamp/blob/master/011_hands-on-exercsises-solutions/4.PNG) Use emmet to build your ordered list and unordered list.
  + (5) [Create this page](https://github.com/GoesToEleven/html-css-bootcamp/blob/master/011_hands-on-exercsises-solutions/5.PNG) which demonstrates the use of the image element and the anchor element. Have the anchor element launch a new browser tab.
  + (6) Add comments to the html document you created in the previous example.
  + (7) Identify which of these URLs are relative URL’s and which are absolute URLs:
    - [www.google.com](http://www.google.com)
    - ../pic/dog.jpg
    - /pic/dog.jpg
    - chapter07/index.html
* **File:** 
  + [**011\_hands-on-exercises-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 49 / take 01**

## 

## Hands-On Exercises - Solutions

* Description

This video presents the solutions to the hands-on exercises.

* **File:** 
  + [**011\_hands-on-exercises-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 50 / take 02**

## 

# CSS Essentials

## Section Overview

* This is what we will learn in this section
  + Section Overview
  + All HTML Tags & Attributes, All CSS Selectors & Properties
  + Border
  + Border Radius
  + Padding & Margin
  + Box Model
  + Box-Sizing: Border-Box
  + Review
  + Challenges
  + Solutions
* Description

We’ve learned some great material so far, and we have more great material coming. In this section we are going to learn some of the CSS essentials. We’re going to learn how to add a border to elements, and then how to round that border with the border-radius property. We are also going to learn how to add spacing between different elements on a page with the padding and margin properties. After that, we are going to learn about the box model. The box model is a super important concept in web programming. Every element on a web page is a box. Those boxes then get stacked on top of each other. We’ll see how this all works, and then see how the box-sizing property helps us when creating our layouts. Finally, we’ll have some challenges to reinforce everything we’re learning, and then I will show you my solutions to those challenges. We’ve learned some great things so far. Let’s take a moment to appreciate this. Here is a high-level review of what we’ve covered so far.

* **Video: 51 / take 01**

## 

## All HTML Tags & Attributes, All CSS Selectors & Properties

* All HTML tags
  + [all HTML elements](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
  + [all HTML attributes](https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes)
* All CSS selectors
  + [all CSS selectors](http://www.w3schools.com/cssref/css_selectors.asp)
* All CSS properties
  + [all CSS properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Reference)
* Description

These resources are super-valuable as they list all of the HTML tags & attributes, as well as all of the CSS selectors & properties. You will use these resources over and over as you create web pages. These resources will allow you to reference the many different tags, selectors, and properties you can use to build your web pages. This is a super-valuable resource.

* **Video: 52 / take 03**

## 

## border Property

### border

* Description

The border property is super fun. Now we’re going to apply the border to several different elements and explore what we can do with it.

* **File:** [**012\_border**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 53 / take 01**

## 

## border-radius Property

### border-radius

* Description

The border-radius css property allows us to round the corners of different elements. The border-radius property can take a value which is measured in either pixels (px) or as a percentage (%). Pixels (px) and percentages (%) are units of measurement. We will learn more about units of measurement later in the course. Once the code is written in this example, we then commit and push the code to github.

* [**File: 013\_border-radius**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 54 / take 01**

## 

## display Property

### display: inline

### display: block

### display: inline-block

### display: none

* Description

The display property allows us to control how an element is displayed on a page. Elements have a default display property. Some elements are block level elements. Some elements are inline level elements. You can also use display: none to not display an element.

* [**File: 014\_display**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 55 / take 01**

## padding & margin Properties

### padding

### border

### margin

* Description

The padding property adjust the amount of space between the content and the border. The margin property adjusts the amount of space between the border and other elements.

* [**File: 015\_padding-margin**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 56 / take 01**

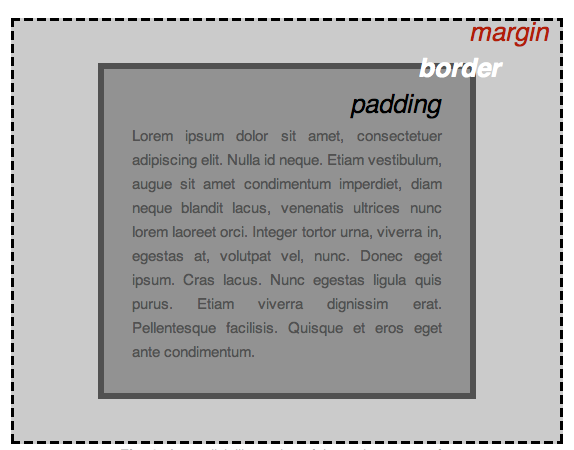
## 

## Box Model

* everything on a web page is a square box
* we stack and arrange those boxes on the page
* the is a visual analogy:



* all content is in a box
* there is **padding** around the content
  + the space between the content and the border
* there is a **border** around the content
* there is a **margin** around the content/padding/border
  + this is the space between this box of content and other boxes of content



* here’s the **problem**
  + if we’re arranging boxes on our page …
  + … and then we change the border or the padding …
  + … then the size of the box changed …
  + … and this results in other boxes shifting their location …
* here’s the **question**
  + how can we change border and padding …
  + … and not have the size of the box change?
* here’s the **solution**:
  + we use a special property:value …
  + box-sizing: border-box

### box-sizing: border-box

* + mnemonic device: “box” is on the outside of “box-sizing: border-box”
  + now we can change padding and border …
  + … and the size of the box doesn’t change …
  + … thus the location of other boxes doesn’t shift.
* Description

Each element on a web page is a box. Some elements have a default display property of “block level” and some elements have a default display property of “inline”. These boxes stack up on the page. Understanding this “box model”, and understanding that elements have a default display property of either “block level” or “inline”, is the first step in understanding layout. Additionally, each element holds “content” which is surrounded by “padding” which is surrounded by a “border” which is surrounded by a “margin”. You can see the “box model” layers of an element in Google Chrome’s developer tools under the “elements” tab.

* [**File: 016\_box-sizing**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 57 / take 01**

## box-Sizing Property

* here’s the **problem**
  + if we’re arranging boxes on our page …
  + … and then we change the border or the padding …
  + … then the size of the box changed …
  + … and this results in other boxes shifting their location …
* here’s the **question**
  + how can we change border and padding …
  + … and not have the size of the box change?
* here’s the **solution**:
  + we use a special property:value …
  + box-sizing: border-box

### box-sizing: border-box

* + mnemonic device: “box” is on the outside of “box-sizing: border-box”
  + now we can change
  + padding and border …
  + … and the size of the box doesn’t change …
  + … thus the location of other boxes doesn’t shift.
* Description

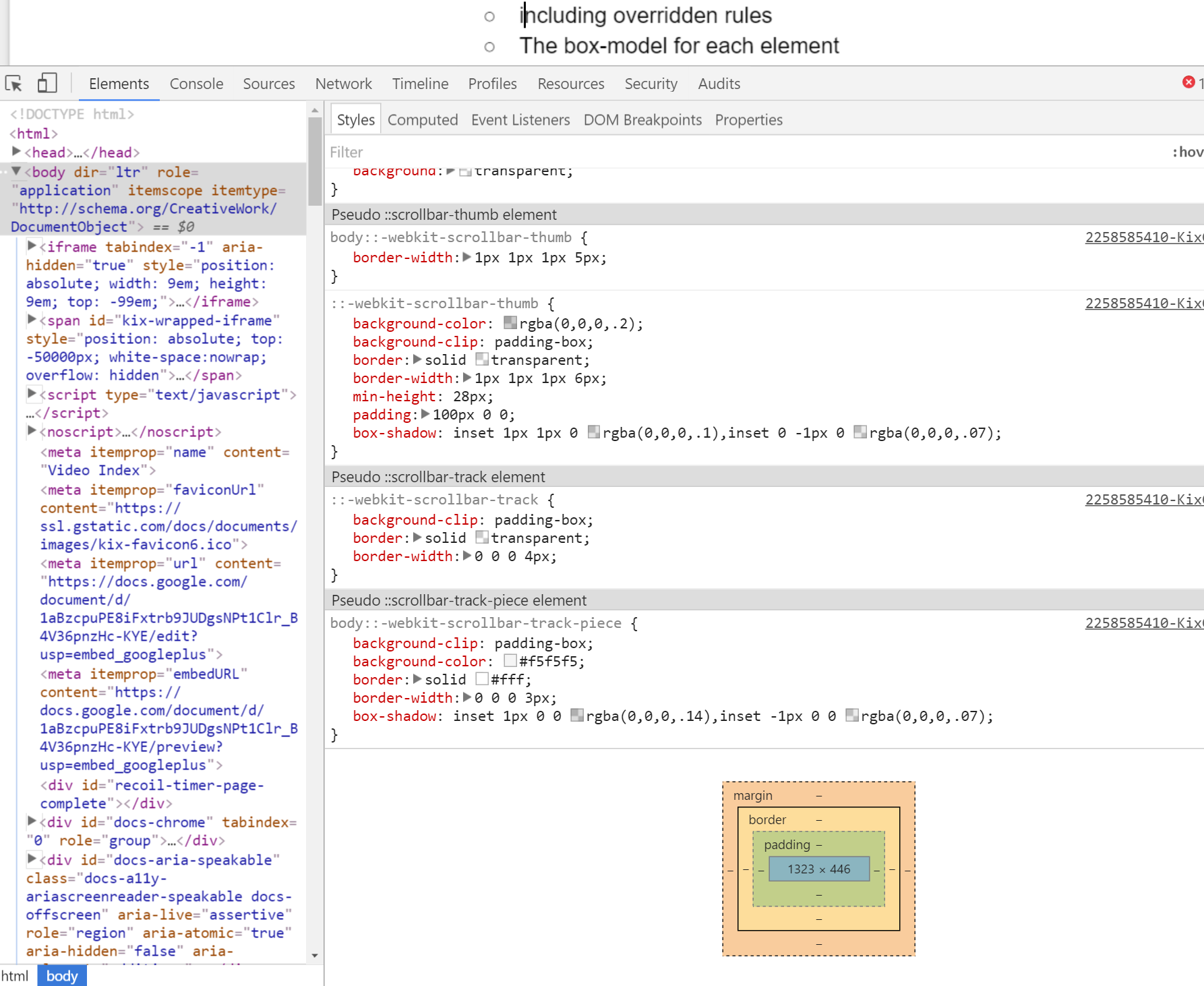
The CSS box-sizing property is important to helping us create layouts. When the width is set on an HTML element, the box-sizing property holds the width of that element to its setting even when the padding and border are changed on the element. When the “box-sizing: border-box” property and value are set on an element, we can increase the size of that element’s padding and border without changing the set width of that element. This helps us format elements without breaking our layout. If we didn’t have “box-sizing: border-box” set on elements, as we increased the padding and border, we would push other elements to the right which could disrupt our layout.

* [**File: 016\_box-sizing**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 58 / take 02**

## 

## Review

* All HTML Tags & Attributes, All CSS Selectors & Properties
* border Property
  + border
* border-radius Property
  + border-radius
* display Property
  + display: inline
  + display: block
  + display: inline-block
  + display: none
* padding & margin Properties
  + padding
  + border
  + margin
* Box Model
* box-sizing Property
  + box-sizing: border-box
* **DTD**
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + \*
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
* **General knowledge**
  + http status codes
    - Developer tools / network
  + Developer tools
    - network tab / throttling
    - right-click / inspect
    - elements tab
      * all of the html
      * all of the css
        + including overridden rules
        + The box-model for each element



* + HTML Terminology
    - Opening tags
    - Closing tags
    - Self-closing tags
    - Nesting tags
    - Parent / child / ancestor / descendant / sibling
    - Tags vs elements
  + CSS Terminology
    - rule-set
      * selector
      * declaration block
        + declarations

property

Value

* + Separation of concerns
  + Linking CSS to HTML
    - External
    - Internal
    - Inline
  + Integrated Development Environments
    - WebStorm
    - Atom.io
    - Sublime
    - Dreamweaver
  + Github & VCS
    - “Master” Local Git Repository
    - “Origin/Master” Remote Github Repository
  + Using Github
    - Basic Commands
      * git status
      * git add --all
      * git commit -m “some concise descriptive message”
      * git push
    - Deleting A Repo
  + HTML & CSS Naming Conventions
    - Files
    - Folders
  + Webstorm shortcuts
    - Keymap reference
  + Modifying Webstorm Formatting
  + Emmet
  + Tag Attributes
  + Relative URLs vs Absolute URLs
  + Comments
  + Box model
* Description

Studies show that reviewing material, and gaining a high-level perspective on course material, significantly increases the speed at which students learn the material. We have learned many great things so far. All of the HTML tags we have learned up to this point are reviewed. Additionally, all of the CSS properties we have learned up to this point are also reviewed. A high-level overview of everything we’ve learned in this section is also presented. In addition, a high-level overview of everything we’ve learned in this course is presented.

* **Video: 59 / take 01**

## 

## Hands-On Exercises

* (1) Place a div on a page. Format the div to have a width, height, and background color.
* (2) Take the code from “Hands-On Exercise (1)” and round the corners of the div.
* (3) Take the code from “Hands-On Exercise (2)” and give the div a grooved border which is 20px.
* (4) Create a new HTML document. Place two divs on the HTML document.
  + Format both divs with the following declarations:
    - width: 400px;
    - height: 200px;
    - border: 10px dashed red;
    - margin: 20px auto;
      * TRBL
      * TB RL
      * T R B L
  + Also format the second div with this declaration:
    - box-sizing: border-box;
* (5) Take the code from “Hands-On Exercise (3)” and give the div a background image by using the following CSS declarations:
  + background-image: url("path/to/some/image.png");
  + background-size: cover;
  + background-repeat: no-repeat;
* Description

Hands-on exercises help reinforce what you are learning. Complete these exercises, then watch the solutions in the next video.

* **Video: 60 / take 01**

## 

## Hands-On Exercises - Solutions

* Description

Here are the solutions to the hands-on exercises. These solutions are live-coded in webstorm. The solutions will show you how to solve the exercises. There is one adjustment to the exercises which is covered in the video. On the fourth exercise, the border is changed to 10px.

* **Files:** [**017\_hands-on-exercises-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 61 / take 01**

# 

# CSS Selectors

## Section Overview

* CSS Resets
* Selectors:
  + element
  + class
  + id
  + attribute
  + pseudo-class
    - link
    - visited
    - hover
    - active
    - focus
    - nth child
    - first-child
    - last-child
    - nth-child(even)
    - nth-child(odd)
    - nth-child(n)
    - nth-last-child(n)
    - nth-child(an+b)
    - only-child
    - first-letter
    - first-line
  + nested selectors
    - div p
    - div > p
    - div ~ p
    - div + p
* Hands-On Exercise - [The Surfer Page](https://drive.google.com/file/d/0B22KXlqHz6ZNOVRBMGVMSGRjaU0/view?usp=sharing)
* Description

CSS selectors allow us to target elements on an HTML page for formatting. Knowing how to use CSS selectors is essential to being able to write effective CSS. This section will focus on many of the different selectors we can use. In addition, we will learn about CSS resets. At the end of this section, we will complete a hands-on exercise to build “The Surfer Page”.”

* **Video: 62 / take 01**

## 

## CSS Resets

* [meyer css reset](http://meyerweb.com/eric/tools/css/reset/) (oldest)
* [normalize.css](http://necolas.github.io/normalize.css/) (newer)
* [sanitize.css](http://jonathantneal.github.io/sanitize.css/) (newest)
* [Mcleod-reset.css](https://github.com/GoesToEleven/html-css-bootcamp/tree/master/018_css-reset/04_mcleod-reset) (best)
* Description

By default, browsers add styling to elements on a web page. We can see this with the heading tags. To complicate issues even further, different browsers style different elements in different ways. A “css reset” allows us to create a uniform starting point from which we can then style our pages. You can think of a “css reset” “zero’ing out” all default browser styling. There are different css resets that you can use. The best practice is to create your own reset. Understand ALL of your code. Have nothing in your code which is extraneous.

* **File:** [**018\_css-reset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 63 / take 06**

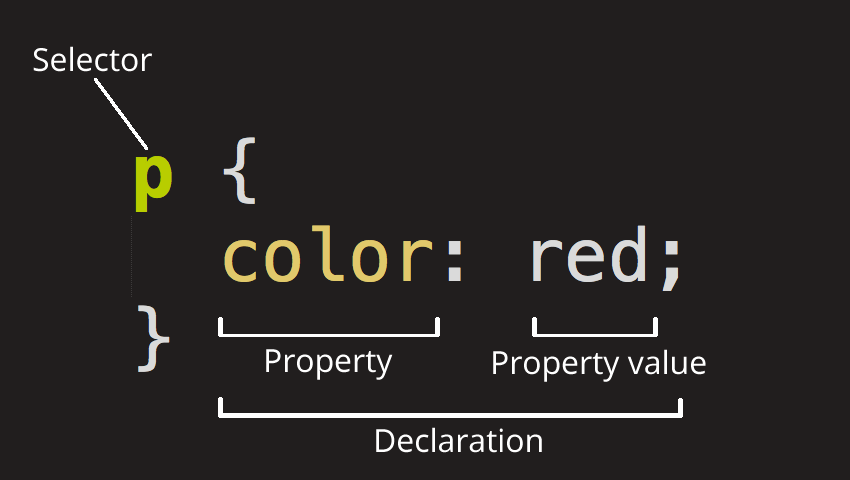
## 

## Selectors: element, class, id

### element

### .class

### #id



* Description

To apply formatting to html elements on an html page, we use CSS. Formatting with CSS entails creating rule-sets. Each rule set begins with a selector. The selector will select which element, or elements, on the html page will be formatted by the declarations which follow. Some of the selectors we may use to select elements on an HTML page include element selectors, class selectors, and id selectors. An id selector can only be applied to one element on an HTML page.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 64 / take 02**

## 

## Selectors: attribute

### [contenteditable]

### [src]

### [href]

* Description

CSS attribute selectors allow us to select an element on an HTML page based upon an element’s attributes.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 65 / take 01**

## 

## Selectors: pseudo-classes

* CSS pseudo-classes that provide interactivity - LVHA order:

### :link

* + - default color for a link

### :visited

* + - clicked a link previously

### :hover

* + - hover over element

### :active

* + - click a link or element that is active

### :focus

* + - an element that can receive focus (an element in a form)
* Description

A CSS pseudo-class is another way we can select elements on an HTML page. A pseudo-class is called a pseudo-class because it is like a class selector - it is able to select multiple elements on a single HTML page. A pseudo-class allows us to add interactivity and visual confirmation to our webpage.

The **:link** CSS pseudo-class lets you select links. This will select any link which has not yet been visited, even those already styled using a selector with other link-related pseudo-classes like :hover, :active or :visited. In order to appropriately style links, you need to **put the :link rule before the other ones.**

The **:visited** CSS pseudo-class lets you select only links that have been visited. This style may be overridden by any other link-related pseudo-classes, that is :link, :hover, and :active, appearing in subsequent rules. In order to style appropriately links, you need to **put the :visited rule after the :link rule but before the other ones.**

The **:hover** CSS pseudo-class matches when the user designates an element with a pointing device, but does not necessarily activate it. This style may be overridden by any other link-related pseudo-classes, that is :link, :visited, and :active, appearing in subsequent rules. In order to style appropriately links, you need to **put the :hover rule after the :link and :visited rules but before the :active one.**

The **:active** CSS pseudo-class matches when an element is being activated by the user. It allows the page to give a feedback that the activation has been detected by the browser. When interacting with a mouse, this is typically the time between the user presses the mouse button and releases it. The :active pseudo-class is also typically matched when using the keyboard tab key. It is frequently used on <a> and <button> HTML elements, but may not be limited to just those.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 66 / take 06**

## 

## Selectors: pseudo-class :focus

* CSS properties

### :focus

* Description

The **:focus** CSS pseudo-class is applied when an element has received focus, either from the user selecting it with the use of a keyboard or by activating with the mouse (e.g. a form input).

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 67 / take 01**

## 

## Selectors: pseudo-class nth child - part I

* CSS properties

### first-child

### last-child

### nth-child(even)

### nth-child(odd)

* Zebra-striping
* Description

We can use a CSS pseudo-class to also target a certain element on a page based upon its position on the page. For instance, if we want to select the first item, or the last item, or some specific number in a list of items we can do so by using the CSS nth child pseudo-class. There are several different CSS properties related to the nth child.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 68 / take 01**

## 

## Selectors: pseudo-class nth child - part II

* CSS properties

### nth-child(n)

### nth-last-child(n)

### nth-child(an+b)

### only-child

* Zebra-striping
* Description

We can use a CSS pseudo-class to also target a certain element on a page based upon its position on the page. For instance, if we want to select the first item, or the last item, or some specific number in a list of items we can do so by using the CSS nth child pseudo-class. There are several different CSS properties related to the nth child.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 69 / take 01**

## 

## Selectors: pseudo-class typography

* CSS properties

### first-letter

### first-line

* Description

Two of my favorite CSS properties are the first-letter and first-line properties. With these two properties, we can achieve very nice typography effects.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 70 / take 01**

## 

## Selectors: nested selectors

### div p

* + All p tags beneath a div

### div > p

* + All p tags **immediately** beneath a div

### div ~ p

* + All p tags that are a sibling following a div

### div + p

* + All p tags that are an **immediate** sibling following a div
* Description

We can further refine which elements we target on an HTML page by using CSS nested CSS selectors.

* **File:** [**019\_css-selectors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 71 / take 01**

## 

## margin: 0 auto & The Display Property

* margin: 0 auto;
* The display property
* debugging with border: 2px solid hotpink;
* Description

Before moving onto the next “hands-on exercise,” it will help you to understand a little bit about one debugging technique: setting a border around all of the elements on an html page. By seeing how much space an element occupies on an html page, you can visually determine if the element is acting like a “block level” element or an “inline” element or an “inline-block” element. In order to center an element across a space, we will need that element to be taking up the entire space. This will be demonstrated with the image element which is an inline element. By changing this element to a “block level” element, we will be able to use “margin: 0 auto” to center the image.

* **File:** [**020\_center-an-image**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Videos: 72 / take 02**

## 

## Hands-On Exercise - The Surfer Page

* Challenge
  + [Create this page.](https://drive.google.com/file/d/0B22KXlqHz6ZNOVRBMGVMSGRjaU0/view?usp=sharing) Make sure you use a stylesheet to format your headings. Include a css reset stylesheet. Link your stylesheets to your html document. Push your code to github.
* Description

Hands-on exercises are designed to help you learn the material. For the vast majority of people, the best way to learn a certain subject matter is by actually applying what they are learning.

* **File:** [Create this page](https://drive.google.com/file/d/0B22KXlqHz6ZNOVRBMGVMSGRjaU0/view?usp=sharing)
* **Videos: 73 / take 03**

## 

## Hands-On Exercise - The Surfer Page - Solution

* Solution to “The Surfer Page” hands-on exercise.
* Description:

To build this solution, we first have to build our html.

* **File:** [**021\_the-surfer-page**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Videos: 74 / take 06**

## 

## Hands-On Exercises

* (1) Create an html page with an ordered list of 20 list items. Use an ID to select the third list item. Style the third list item red.
* (2) Create an html page with an ordered list of 20 list items. Use a class to select the third, fourth, and fifth list item. Style these items red.
* (3) Create an html page with 2 anchor tags providing links to different websites. Give one of the anchor tags the target attribute. Use an attribute selector to select the anchor tag with the target attribute. Style this selected anchor tag to have a border around it.
* (4) Create an html page with 1 anchor tag. Provide styling for this anchor tag including different colors for these css pseudo-class properties: link, visited, hover, active.
* (5) Create a paragraph of text. Style the first letter of that paragraph and the first line to be different from the rest of the paragraph.
* (6) Use this emmet to create this html within an html page’s body tag ... p{$}+div>p{$$}\*3+article>p>lorem^^^p{$$$}\*3 … and then, once the html is created, do the following: select all p tags that are children of a div and make them red.
* (7) Use this emmet to create this html within an html page’s body tag ... p{$}+div>p{$$}\*3+article>p>lorem^^^p{$$$}\*3 … and then, once the html is created, do the following: select all p tags that are **immediate** children of a div and make them red.
* (8) Use this emmet to create this html within an html page’s body tag ... p{$}+div>p{$$}\*3+article>p>lorem^^^p{$$$}\*3 … and then, once the html is created, do the following: select all p tags that are siblings following a div and make them red.
* (9) Use this emmet to create this html within an html page’s body tag ... p{$}+div>p{$$}\*3+article>p>lorem^^^p{$$$}\*3 … and then, once the html is created, do the following: select all p tags that are an **immediate** sibling following a div and make it red.
* **Videos: 75 / take 03**

## 

## Hands-On Exercises - Solutions I

* Description

Here are the solutions to the hands-on exercises 1 - 5. These solutions are live-coded in webstorm. The solutions will show you how to solve the exercises.

* **Files:** [**022\_hands-on-exercise-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 76 / take 02**

## 

## Hands-On Exercises - Solutions II

### div p

* + All p tags beneath a div

### div > p

* + All p tags **immediately** beneath a div

### div ~ p

* + All p tags that are a sibling following a div

### div + p

* + All p tags that are an **immediate** sibling following a div
* Description

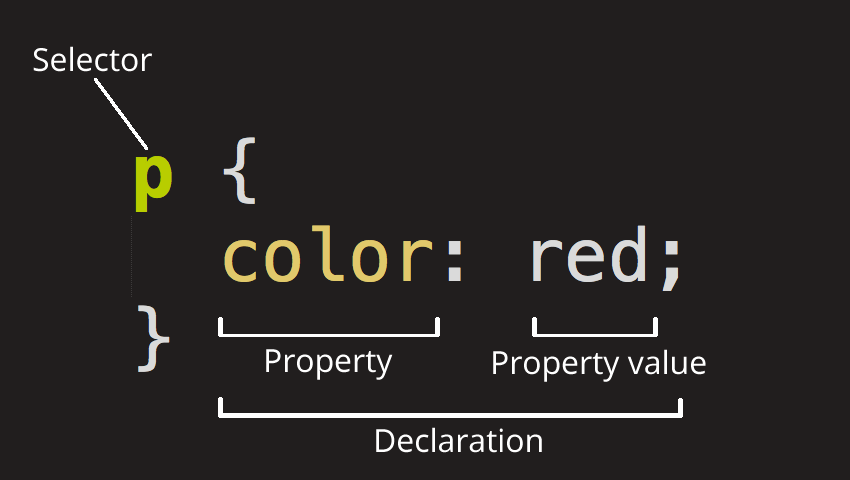
Here are the solutions to the hands-on exercises 6 - 9. These solutions are live-coded in webstorm. The solutions will show you how to solve the exercises.

* **Files:** [**022\_hands-on-exercise-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 77 / take 02**

## 

## Review

* CSS resets
* CSS selectors
  + CSS rule sets



* CSS Selectors:
  + element
  + class
  + id
  + attribute
  + pseudo-class
    - link
    - visited
    - hover
    - active
    - focus
    - nth child
    - first-child
    - last-child
    - nth-child(even)
    - nth-child(odd)
    - nth-child(n)
    - nth-last-child(n)
    - nth-child(an+b)
    - only-child
    - first-letter
    - first-line
  + nested selectors
    - div p
    - div > p
    - div ~ p
    - div + p
* margin: 0 auto & display
* [The Surfer Page](https://drive.google.com/file/d/0B22KXlqHz6ZNOVRBMGVMSGRjaU0/view?usp=sharing)
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
* All CSS selectors we have learned so far
  + element
  + class
  + id
  + \*
  + attribute
  + link
  + visited
  + hover
  + active
  + focus
  + first-letter
  + first-line
  + div p
    - All p tags beneath a div
  + div > p
    - All p tags **immediately** beneath a div
  + div ~ p
    - All p tags that are a sibling following a div
  + div + p
    - All p tags that are an **immediate** sibling following a div
* **Video: 78 / take 04**

# 

# CSS Order & Specificity

## 

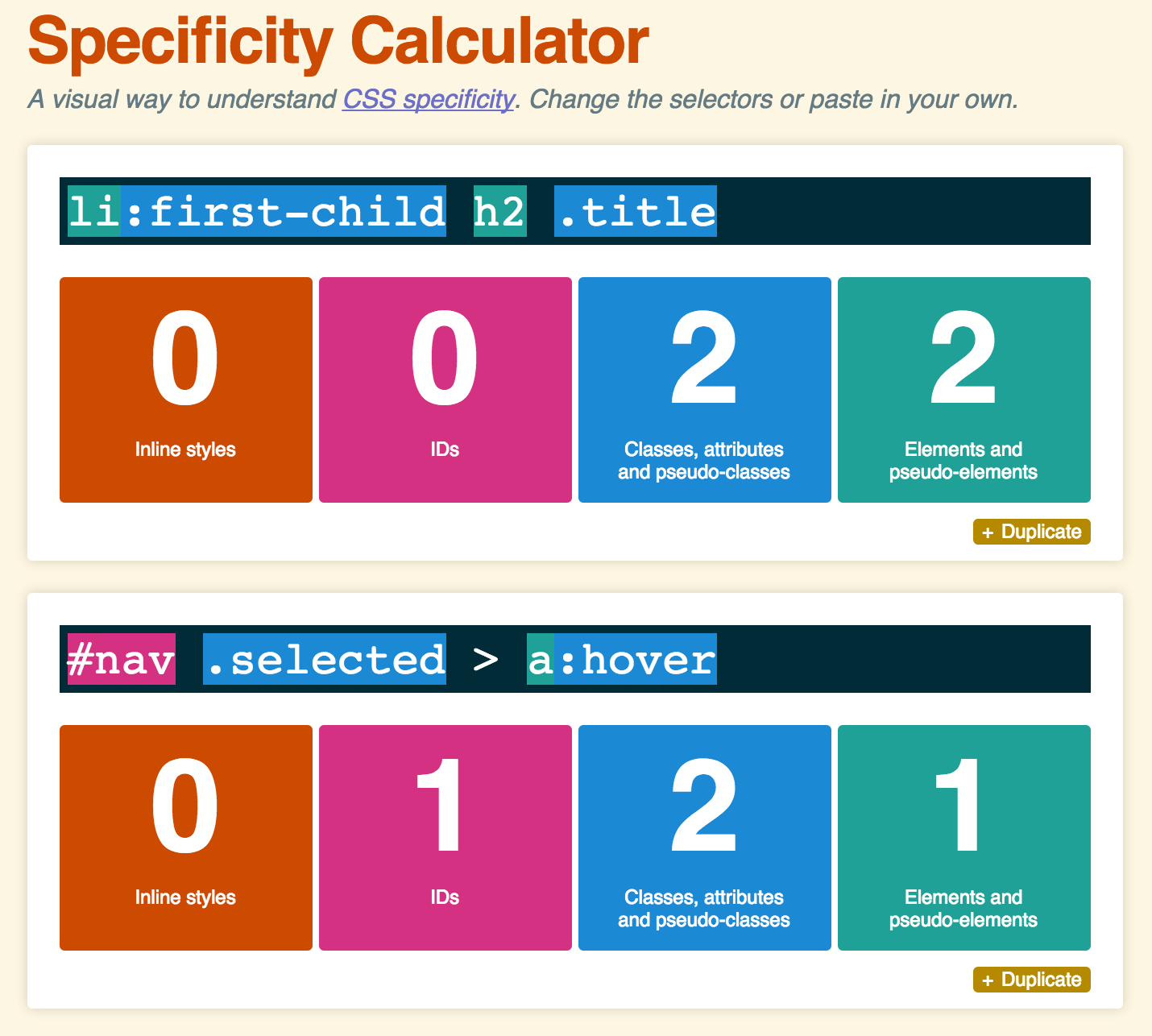
## Section Overview

* When more than one CSS rule-set is applied to the same element, what determines which declarations will be applied?
  + CSS specificity and order determine which declarations are applied
    - CSS specificity
      * inline style
      * id
      * class, attribute, pseudo-class
      * element, pseudo-element
    - order
      * when specificity is equal, last declaration wins
* Applying multiple classes to the same element
* 9 challenges
* Hands-on exercises
* **File:** [**023\_css-specificity**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 79 / take 01**

## 

## The Browser’s Application of CSS

* [CSS Specificity Calculator](https://specificity.keegan.st/)



* Different browsers, different [rendering engines](https://en.wikipedia.org/wiki/Web_browser_engine)
  + **Webkit**
    - Safari
    - **Blink**
      * Chrome
      * Opera
      * Amazon Silk
  + **Gecko**
    - Firefox
  + **Trident**
    - Internet Explorer
  + **EdgeHTML**
    - Microsoft Edge
* As a developer …
  + Order matters
  + Specificity matters
    - Keep selectors as simple as possible
    - Prefix selectors
      * Provides scope to your page
* CSS is applied top to bottom
* If multiple rule-sets target the same element
  + CSS specificity and order determine which declarations are applied
    - CSS specificity
      * inline style
      * id
      * class, attribute, pseudo-class
      * element, pseudo-element
    - order
      * when specificity is equal, last declaration wins
* When selectors have an **unequal** specificity value ...

… the more specific rule is the one that counts.

* When selectors have an **equal** specificity value …

… the last declaration is the one that wins.

* The universal selector ( \* ) has a specificity of 0, 0, 0, 0.
* Description

As you add more CSS rule-sets to format an html page, and as your CSS files become larger, you might start having conflicts between different CSS selectors. In the event of two CSS selectors targeting the same element and conflicting, there CSS specificity rules which are followed to determine which CSS rule-set should be applied.

* **File:** [**023\_css-specificity**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 80 / take 01**

## 

## Challenge & Solution #1

* Description

A challenge which demonstrates the calculation of CSS specificity.

* **File:** [**023\_css-specificity**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 81 / take 01**

## 

## Challenge & Solution #2, 3, 4

* Description

A challenge which demonstrates the calculation of CSS specificity.

* **File:** [**023\_css-specificity**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 82 / take 01**

## 

## Challenge & Solution #5, 6

* Description

A challenge which demonstrates the calculation of CSS specificity.

* **File:** [**023\_css-specificity**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 83 / take 01**

## 

## Hands-On Exercise

* (1) Write some CSS that has two rules applying to the same element. The first rule should use a **class** and the second rule should use an **ID.** Calculate the CSS specificity of each rule to explain which rule won.
* (2) Write some CSS that has two rules applying to the same element. The first rule should use an **element** selector and the second rule should use an **ID.** Calculate the CSS specificity of each rule to explain which rule won.
* (3) Write some CSS that has two rules applying to the same element. The first rule should use an **element** selector and the second rule should use a **class.** Calculate the CSS specificity of each rule to explain which rule won.
* (4) Write some CSS that has two rules applying to the same element. The first rule should use an **element** selector and the second rule should use an **attribute.** Calculate the CSS specificity of each rule to explain which rule won.
* (5) Write some CSS that has two rules applying to the same element. The first rule should use an **class** selector and the second rule should use an **attribute.** Calculate the CSS specificity of each rule. Explain which rule won and why.
* **Video: 84 / take 01**

## Hands-On Exercises - Solutions

* Description

Here are the solutions to the hands-on exercises. These solutions are live-coded in webstorm. The solutions will show you how to solve the exercises.

* **Files:** [**024\_hands-on-exercises-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 85 / take 01**

## 

## Review

* When more than one CSS rule-set is applied to the same element, what determines which element will be applied?
  + CSS Specificity
    - inline style
    - id
    - class, attribute, pseudo-class
    - element, pseudo-element
  + Order: last declaration wins
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + font-style
    - font-style: italic;
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* **Video: 86 / take 02**

# Formatting Text

## 

## Section Overview

* font
  + font-family
  + font-size
  + font-weight
  + font-variant
  + line-height
  + font-style
* serif vs sans-serif fonts
* text-transform
  + uppercase
  + lowercase
  + capitalize
* letter-spacing
* word-spacing
* color
* Google fonts
* text-align
* text-shadow
* text-decoration
* striking text
  + CSS
    - text-decoration: line-through;
  + HTML elements
    - del
    - s
* text-indent
* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 87 / take 03**

## 

## font Property

* [MDN font](https://developer.mozilla.org/en-US/docs/Web/CSS/font)
* CSS properties

### font

### font-family

* + - * a prioritized list of font family names
      * Values
        + <family-name>
        + <generic-name>

serif

sans-serif

monspace

cursive

fantasy

* + - Serif vs. Sans-Serif
      * Current web design favors sans-serif
      * Article: [Serif vs. Sans: the final battle](http://www.webdesignerdepot.com/2013/03/serif-vs-sans-the-final-battle/)
      * Article: [Serif vs. Sans Serif Fonts: Is One Really Better Than the Other?](https://designshack.net/articles/typography/serif-vs-sans-serif-fonts-is-one-really-better-than-the-other/)

### font-size

* + - * Must understand user’s default font-size
        + Usually 16px
        + You can see this in your browser
        + Some people change this
        + font-size values can be based upon user’s default font-size
      * font-size values
        + xx-small, x-small, small, medium, large, x-large, xx-large

User’s default font size is medium

**relative to default font size ( root font size )**

* + - * + larger, smaller

**relative to parent element’s font size**

* + - * + <length>

[MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)

em

**relative to parent element’s font size**

rem

**relative to default font size ( root font size )**

vh

**1/100th of the height of the viewport.**

vw

**1/100th of the width of the viewport.**

* + - * + <percentage>

A percentage of the parent element’s font size

**relative to parent element’s font size**

### font-weight

* + - * normal
        + same as 400
      * bold
        + same as 700
      * lighter
      * bolder
      * 100, 200, 300, 400, 500, 600, 700, 800, 900

### font-variant

* + - * normal
      * small-caps
      * titling-caps
      * unicase
      * [see other values](https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant#Values)
      * Also see: [text-transform](https://developer.mozilla.org/en-US/docs/Web/CSS/text-transform) 
        + uppercase
        + lowercase
        + capitalize

### line-height

* + - * [MDN line height](https://developer.mozilla.org/en-US/docs/Web/CSS/line-height)
      * On block level elements, the line-height property specifies the minimum height of line boxes within the element.
      * **Leading**: space between lines
      * **Kerning**: space between characters
      * Values
        + normal
        + <number>
        + <length>
        + <percentage>

### font-style

* + - * normal
      * Italic
      * Oblique
* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 88 / take 01**

## 

## font-family Property

* a prioritized list of font family names
* Values
  + <family-name>
  + <generic-name>
    - serif
    - sans-serif
    - monspace
    - cursive
    - fantasy
* Description:

The **font-family** CSS property lets you specify a **prioritized list** of font family names and/or generic family names for the selected element. Values are separated by a comma to indicate that they are alternatives. The browser will select the first font on the list that is installed on the computer or that can be downloaded. **Web authors should always add at least one generic family in a font-family list, since there's no guarantee that a specific font is installed on the computer or can be downloaded.** The generic family lets the browser select an acceptable **fallback** font when needed. It is often convenient to use the shorthand property font to set font-size and other font related properties all at once.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 89 / take 01**

## 

## font-size Property

* Must understand user’s default font-size
  + Usually 16px
  + You can see this in your browser
  + Some people change this
  + font-size values can be based upon user’s default font-size
* font-size values
  + xx-small, x-small, small, medium, large, x-large, xx-large
    - User’s default font size is medium
    - **relative to default font size ( root font size )**
  + larger, smaller
    - **relative to parent element’s font size**
  + <length>
    - [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
    - em
      * **relative to parent element’s font size**
    - rem
      * **relative to default font size ( root font size )**
    - vh
      * **1/100th of the height of the viewport.**
    - vw
      * **1/100th of the width of the viewport.**
  + <percentage>
    - A percentage of the parent element’s font size
    - **relative to parent element’s font size**
* Description:

The font-size CSS property specifies the size of the font.

The [<length>](https://developer.mozilla.org/en-US/docs/Web/CSS/length) **CSS data type** denotes distance measurements. It is a <number> immediately followed by a length unit (px, rem, …). Like for any CSS dimension, there is no space between the unit literal and the number.

The length unit is optional after the <number> 0.

Many **CSS properties** take <length> values, such as width, margin, padding, font-size, border-width, text-shadow, … For some properties, using negative lengths is a syntax error, but for some properties, negative lengths are allowed.

Please note that although <percentage> values are also CSS dimensions and are accepted by some CSS properties that accept <length> values, they are not themselves, <length> values.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 90 / take 08**

## 

## font-weight & font-style Properties

* font-weight
  + normal
    - same as 400
  + bold
    - same as 700
  + lighter
  + bolder
  + 100, 200, 300, 400, 500, 600, 700, 800, 900
* font-style
  + normal
  + Italic
  + oblique
* Description:

The **font-weight** CSS property specifies the weight or boldness of the font. Some fonts are only available in normal and bold. The **font-style** CSS property lets you select italic or oblique faces within a font-family. Italic forms are generally cursive in nature, usually using less horizontal space than their unstyled counterparts, while oblique faces are usually just sloped versions of the regular face. Both italic and oblique faces are simulated by artificially sloping the glyphs of the regular face.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 91 / take 02**

## 

## font-variant & text-transform Properties

* font-variant
  + normal
  + small-caps
  + titling-caps
  + unicase
  + [see other values](https://developer.mozilla.org/en-US/docs/Web/CSS/font-variant#Values)

### text-transform

* + [MDN text-transform](https://developer.mozilla.org/en-US/docs/Web/CSS/text-transform)
  + uppercase
  + lowercase
  + Capitalize
* Description:

The **font-variant** property acts as a shorthand for the longhand properties: font-variant-caps, font-variant-numeric, font-variant-alternates, font-variant-ligatures, and font-variant-east-asian. You can also set the CSS Level 2 (Revision 1) values of font-variant, (that is, normal or small-caps), by using the font shorthand. The **text-transform** CSS property specifies how to capitalize an element's text. It can be used to make text appear in all-uppercase or all-lowercase, or with each word capitalized.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 92 / take 02**

## 

## line-height Property

* [MDN line height](https://developer.mozilla.org/en-US/docs/Web/CSS/line-height)
* On block level elements, the line-height property specifies the minimum height of line boxes within the element.
* Values
  + normal
  + <number>
  + <length>
    - CSS data type
  + <percentage>
* Description:

On block level elements, the line-height property specifies the minimum height of line boxes within the element.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 93 / take 04**

## 

## letter-spacing, word-spacing, & color Properties

### [letter-spacing](https://developer.mozilla.org/en-US/docs/Web/CSS/letter-spacing)

* + values
    - normal
    - [<length> css data type](https://developer.mozilla.org/en-US/docs/Web/CSS/length)

### [word-spacing](https://developer.mozilla.org/en-US/docs/Web/CSS/word-spacing)

* + values
    - normal
    - [<length> css data type](https://developer.mozilla.org/en-US/docs/Web/CSS/length)
    - percentage

### [color](https://developer.mozilla.org/en-US/docs/Web/CSS/color)

* + values
    - [<color> css data type](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)
* Description:

The **letter-spacing** CSS property specifies spacing behavior between text characters. In typography, letter spacing is known as kerning. The **word-spacing** CSS property specifies the spacing behavior between words. The **color** property sets the foreground color of an element's text content, and its decorations. It doesn't affect any other characteristic of the element; it should really be called text-color and would have been named so, save for historical reasons. Note that the color value must be a uniform color, which may include a transparency value from CSS3 onwards.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 94 / take 05**

## 

## Google Fonts

* Different fonts
* Performance considerations
* Using Google fonts
* Developer tools / network tab
  + Type
  + Size
  + Click the name of the item downloaded:
    - Headers
    - Preview
* Description:

Google fonts allows us to use different fonts on our websites. Google fonts is used extensively. Loading a font on a user’s computer does have a download cost. Popular fonts are cached and this can help with performance.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 95 / take 02**

## 

## text-align & text-shadow Properties

### [text-align](https://developer.mozilla.org/en-US/docs/Web/CSS/text-align)

* + margin: 0 auto;

### [text-shadow](https://developer.mozilla.org/en-US/docs/Web/CSS/text-shadow)

* + accepts a comma-separated list: x, y, blur radius, color
    - offset-x | offset-y | blur-radius | color
    - text-shadow: 1px 1px 2px black;
* Description:

The **text-align** CSS property describes how inline content like text is aligned in its parent block element. text-align does not control the alignment of block elements, only their inline content. The **text-shadow** property adds shadows to text.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 96 / take 01**

## 

## text-decoration Property & <del>

### [text-decoration](https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration)

* + [text-decoration-line](https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration-line) 
    - none
    - underline
    - overline
    - line-through
  + [text-decoration-color](https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration-color) 
    - <color> CSS data type
  + [text-decoration-style](https://developer.mozilla.org/en-US/docs/Web/CSS/text-decoration-style) 
    - solid
    - double
    - dotted
    - dashed
    - wavy

### [<del>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/del)

* [<s>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/s)
  + do not use
  + use text-decoration: line-through;
* Description:

The **text-decoration** CSS property is used to set the text formatting to underline, overline, or line-through. The HTML Deleted Text Element (**<del>**) represents a range of text that has been deleted from a document. This element is often (but need not be) rendered with strike-through text. The HTML Strikethrough Element (**<s>**) renders text with a strikethrough, or a line through it. Use the <s> element to represent things that are no longer relevant or no longer accurate. However, <s> is not appropriate when indicating document edits; for that, use the <del> and <ins> elements, as appropriate. The CSS text-decoration property is to be used to achieve the former visual aspect of the <s> element.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 97 / take 01**

## 

## text-indent Property

### [text-indent](https://developer.mozilla.org/en-US/docs/Web/CSS/text-indent)

* + <length> CSS data type
  + percentage
* Description:

The **text-indent** property specifies the amount of indentation (empty space) should be left before lines of text in a block. By default, this controls the indentation of only the first formatted line of the block, but the hanging and each-line keywords can be used to change this behavior. Horizontal spacing is with respect to the left (or right, for right-to-left layout) edge of the containing block element's box.

* **Files:** [**025\_formatting-text**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 98 / take 01**

## 

## Hands-On Exercises

* (1) Use a generic serif font in an HTML document.
* (2) Use a generic san-serif font in an HTML document.
* (3) Use a font from Google fonts that has different font-weights. Use the different font-weights.
* (4) Using CSS, transform all of the text of a sentence to uppercase and italic.
* (5) Set spacing between characters in a short phrase.
* (6) Add shadow to the text in a H1 tag.
* (7) Remove the underline on an anchor tag. And then add the underline in when someone hovers over the link.
* (8) Show text that is no longer relevant by putting a line through it.
* (9) Indent the first line of a paragraph.
* **Video: 99 / take 01**

## 

## Hands-On Exercises - Solutions

* Description

Here are the solutions to the hands-on exercises. These solutions are live-coded in webstorm. The solutions will show you how to solve the exercises.

* **Files:** [**026\_hands-on-exercises-solutions**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 100 / take 01**

## Review

* Formatting text
* Serif vs. Sans-Serif
  + Current web design favors sans-serif
* Google fonts
  + Performance
  + Developer tools / network
  + Avg page is 2400KB ~ 2.4MB
* striking text
  + CSS
    - text-decoration: line-through;
  + HTML elements
    - del
    - s
* Font sizes
  + rem
  + vw
  + px
* <length> css data type
* <color> css data type
* shorthand properties
* fallback settings
  + like with font-family
* leading and kerning
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite

### FREE PREVIEW this video

* **Video: 101 / take 01**

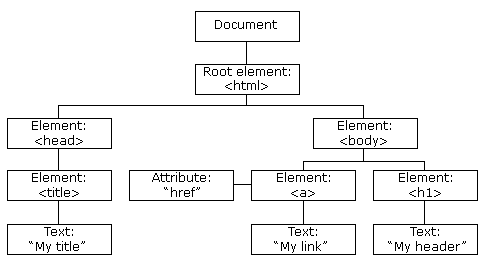
# 

# Structuring Documents

## 

## Section Overview

* document structure
  + the importance of knowing your craft
  + precision in engineering
  + a master must know her or his tools
  + separate the skilled from the unskilled
    - know the tools, know the materials
      * wood worker analogy
    - be precise
* importance of good document structure
  + maintainability
  + readability
    - using correct tags = greater readability
    - avoid “div soup” aka “divitis”
  + accessibility
    - screen readers
  + search engine rank
    - SEO - search engine optimization
    - [alexa website statistics](http://www.alexa.com/)
      * my speculation about Google’s page ranking
        + Google wants to deliver quality to its customers
        + Google looks to see how the page is built
        + pages built well reflect quality
* document structure terminology
  + semantic HTML
  + structure vs layout
  + document object model (DOM)



* + document flow
  + Content sectioning
    - reading and understanding documentation
  + Content categories
  + HTML5 Outline algorithm - don’t use
    - Use headings instead
* hands-on exercises - semantic html
  + my solutions
* elements / tags
  + header
  + nav
  + main
  + article
  + section
  + aside
  + footer
  + h1 - h6
  + figure
  + figcaption
  + address
* Take-away:
  + don’t just use div’s
  + use semantic html
    - the most specific and applicable element
* Description:

Being a craftsperson requires that you learn about both the tools and materials with which you will be working. Learning the craft of building web pages requires us to know semantic html. Semantic html will make our documents better: more maintainable, more readable, more accessible, and more likely to be ranked well by search engines. The main goal of this section is to introduce you to several semantic elements which are used for structuring html documents. In addition, we will learn about the Document Object Model.

### FREE PREVIEW this video

* **Video: 102 / take 01**

## 

## Document Structure - Terminology

* semantic HTML
* structure vs layout
* document object model (DOM)
* document flow:
  + default display: block-level & inline elements
  + how they stack up & line up on the page
    - “arrangement of page elements as defined by positioning statements and the order of html statements; that is, how the different elements take up space and arrange themselves around each other.” ([src](https://www.codecademy.com/forum_questions/4fb56b2c9bcf1e00030635b2))
    - “how a page is presented when you do nothing to it with regard to structural styling. Browsers display content in the order that it's encountered; top stuff first, proceeding content lower down.” ([src](http://webdesign.tutsplus.com/articles/quick-tip-utilizing-normal-document-flow--webdesign-8199))
    - “the normal way that the elements will display as they're written in the HTML file. Which, as you know, is from top to bottom -- as in "A block level element takes up the entire width of the page and the next element coming after it within the document, then gets displayed beneath it, etc.. etc..etc.. and so on and so forth" So when he says it removes it from the normal document flow it's like... Hmm, Ok, think of it like this. Aside from inline elements, all the elements stack on top of each other, right? Well, instead of stacking on top of each other, I like to think of an HTML doc as one big tetris game” ([src](https://teamtreehouse.com/community/what-is-the-normal-document-flow))
* Content sectioning
  + [content sectioning](https://developer.mozilla.org/en-US/docs/Web/HTML/Element)
* Content categories
  + [content categories](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Content_categories)
* HTML5 Outline algorithm
  + don’t use
    - [document outline -](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Sections_and_Outlines_of_an_HTML5_document) [using HTML sections and outlines](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Using_HTML_sections_and_outlines)
    - some resources could make you think you need to use it (don’t use it):
      * + [HTML5 Doctor Document Outlines](http://html5doctor.com/outlines/)
        + [HTML5 Outliner](https://chrome.google.com/webstore/detail/html5-outliner/afoibpobokebhgfnknfndkgemglggomo?hl=en)
  + use headings (h1-h6) instead
* Description:

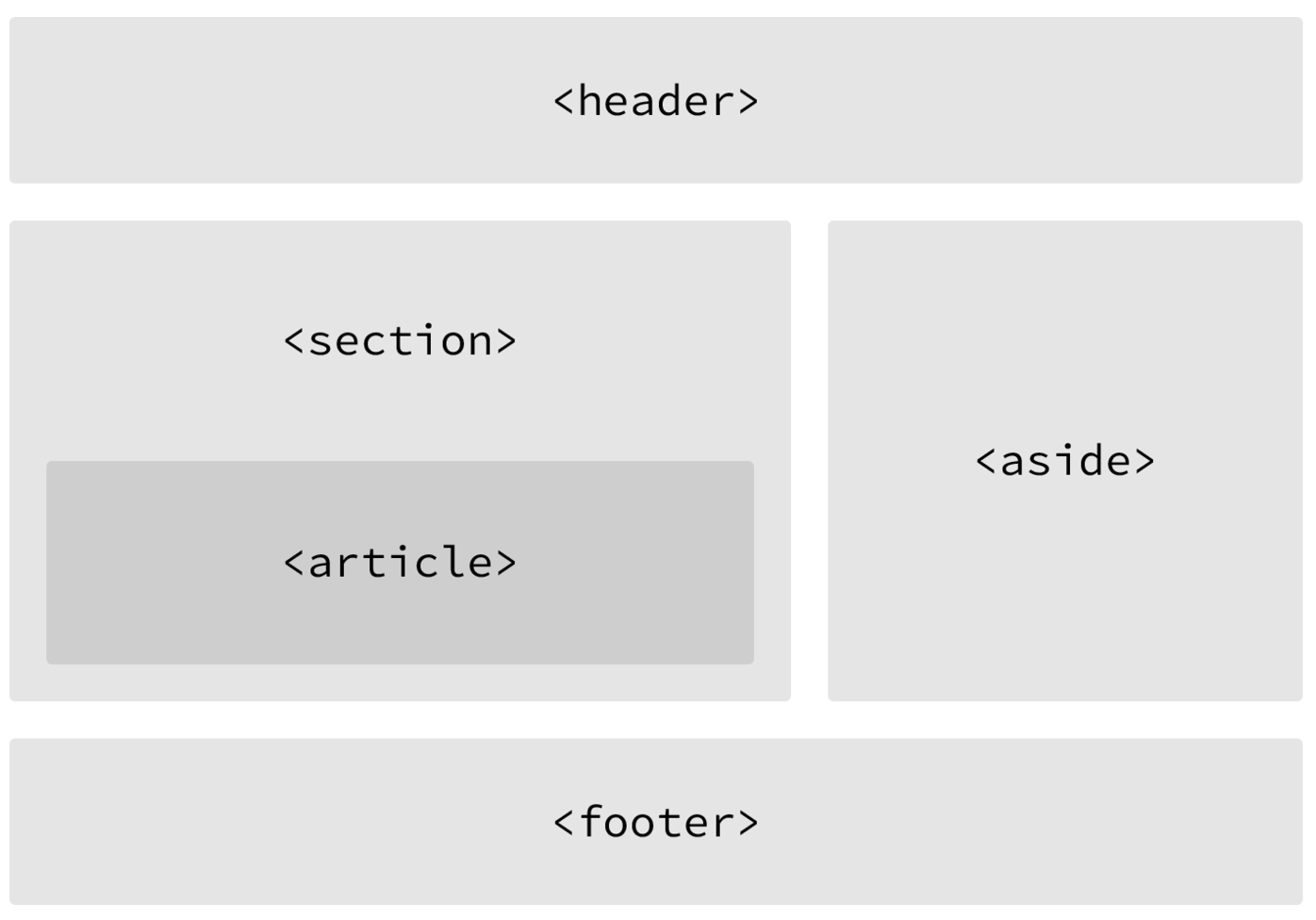
The “HTML element reference” page lists all of the HTML elements. They are grouped by function to help you find what you have in mind easily. Although this guide is written for those who are newer to coding, it is intended to be useful for anyone. In addition, we will learn about **content models.** Each HTML element must abide by rules defining what kind of content it can have. These rules are grouped into content models common to several elements. Each HTML element belongs to zero, one, or multiple content models, each setting rules that the element's content must follow in an HTML-conformant document. Understanding content models will help us read and understand documentation. This, in turn, will help us use HTML correctly.

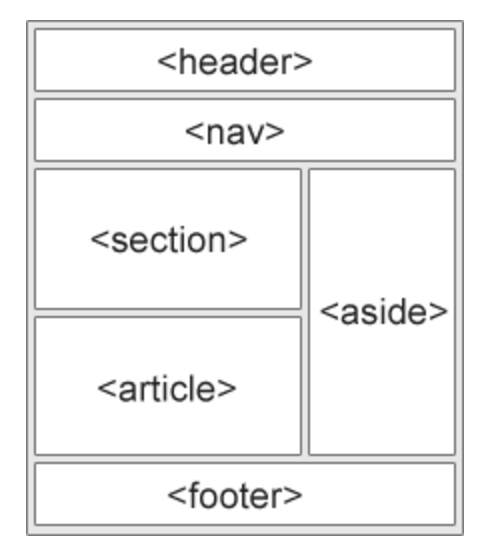
* **Video: 103 / take 01**

## 

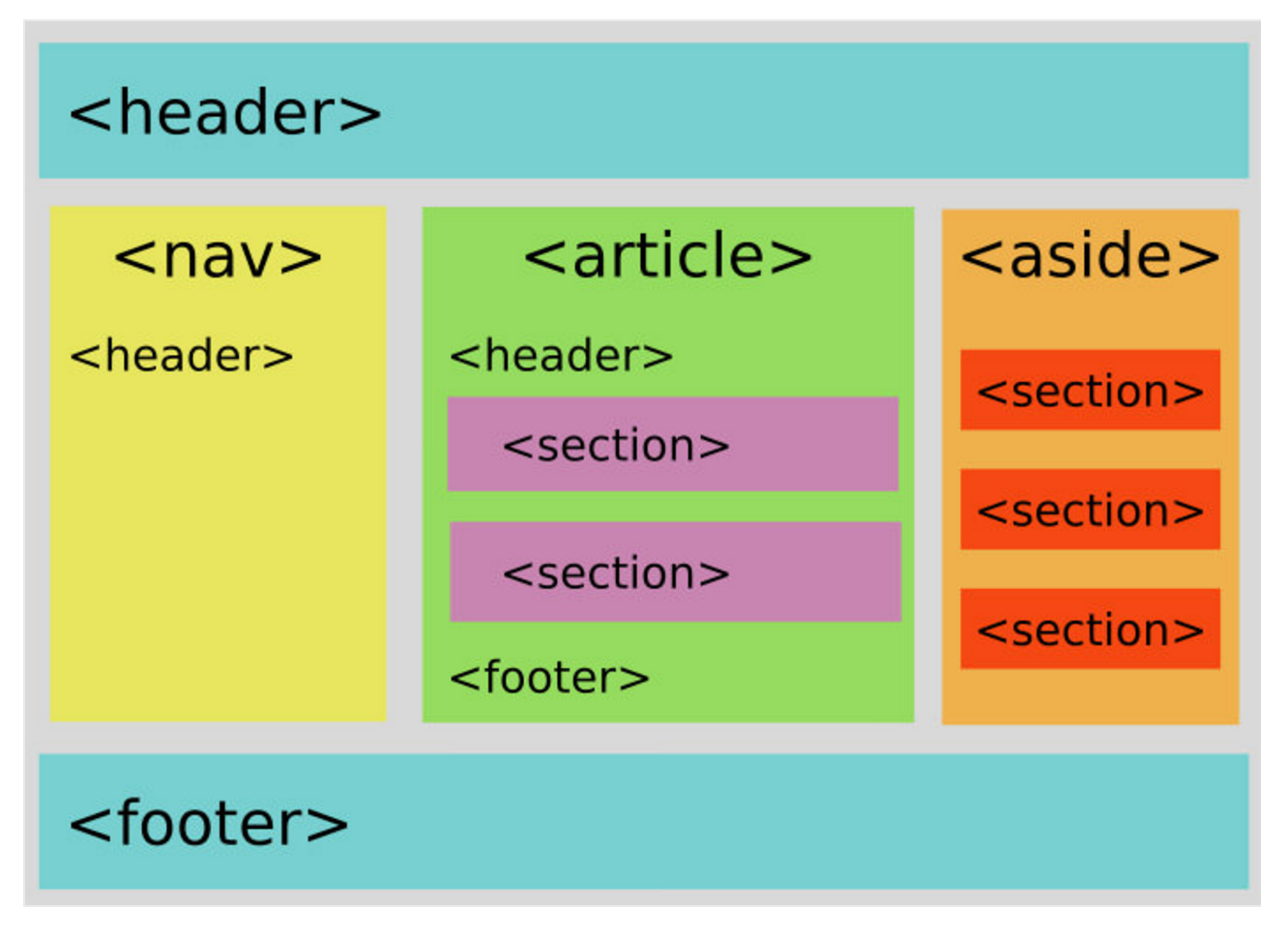
## Semantic HTML

* Semantic tags
  + semantics: conveying meaning
  + some tags are semantic, some aren’t
  + As much as possible, use tags which convey meaning
* Examples of **content sectioning**:

  
source: [Shay Howe](http://learn.shayhowe.com/html-css/getting-to-know-html/)

  
source: [W3 Schools](http://www.w3schools.com/html/html5_semantic_elements.asp)

  
source: [Jennifer Marsman](http://blogs.msdn.com/b/jennifer/archive/2011/08/01/html5-part-1-semantic-markup-and-page-layout.aspx)

  
source & **code example**: [hongkiat](http://www.hongkiat.com/blog/html-5-semantics/)

* section vs. article
  + think about newspapers: What is a section and what is an article?
    - example:
      * The business section can have many articles
      * A stock market article can have many sections
    - example:
      * The global section can have many articles
      * An article on African development can have many sections
  + take away #1
    - articles can have sections
    - sections can have articles
  + take away #2
    - articles can stand on their own
      * you could take the article and put it anywhere and it would make sense
* Description:

Semantic HTML is the use of HTML markup to reinforce the semantics, or meaning, of the information in webpages and web applications rather than merely to define its presentation or look. Semantic HTML is processed by traditional web browsers as well as by many other user agents. CSS is used to suggest its presentation to human users. As an example, recent HTML standards discourage use of the tag <i> (italic, a typeface) in preference of more accurate tags such as <em> (emphasis); the CSS stylesheet should then specify whether emphasis is denoted by an italic font, a bold font, underlining, slower or louder audible speech etc. This is because italics are used for purposes other than emphasis, such as citing a source; for this, HTML provides the tag <cite>. Another use for italics is foreign phrases. Marking emphasis and citations in different ways makes it easier for web agents such as search engines and other software to ascertain the significance of the text.

* **File:** [**027\_structuring-documents / 01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **File:** [**027\_structuring-documents / 02**](https://github.com/GoesToEleven/html-css-bootcamp)

### FREE PREVIEW this video

* **Video: 104 / take 01**

## 

## Hands-On Exercises - Semantic HTML

* For each of the challenges below, use your current knowledge of **semantic HTML** elements to create appropriate **content sections**
  + generally sketch out what tags could be used where
    - you do not have to have mastery in your answer
    - you do not have to have intricate depth
    - you do not have to sketch out every single item on the page
* [challenge #1](https://drive.google.com/file/d/0B22KXlqHz6ZNZlpVdWFjTW1KSW8/view?usp=sharing)
* [challenge #2](https://drive.google.com/file/d/0B22KXlqHz6ZNaGw4bTZhcjM4NDA/view?usp=sharing)
* [challenge #3](https://drive.google.com/file/d/0B22KXlqHz6ZNSy12T2dMVklMd2c/view?usp=sharing)
* [challenge #4](https://drive.google.com/file/d/0B22KXlqHz6ZNSnVrVlNWc3JUeG8/view?usp=sharing)
* [challenge #5](https://drive.google.com/file/d/0B22KXlqHz6ZNQmhPd1dST3NJU2s/view?usp=sharing)
* [challenge #6](https://drive.google.com/file/d/0B22KXlqHz6ZNOXhYV1pIVWdpS0k/view?usp=sharing)
* Description:

These hands-on exercises will help you learn how to structure content in an html document using semantic html elements.

### FREE PREVIEW this video

* **Video: 105 / take 01**

## 

## Hands-On Exercises - Semantic HTML - My Solutions

* My solutions
  + [challenge #1](https://drive.google.com/file/d/0B22KXlqHz6ZNZlpVdWFjTW1KSW8/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNaTNnUG1ieHVVRFU/view?usp=sharing)
  + [challenge #2](https://drive.google.com/file/d/0B22KXlqHz6ZNaGw4bTZhcjM4NDA/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNdEEweTV0YnM3dDg/view?usp=sharing)
  + [challenge #3](https://drive.google.com/file/d/0B22KXlqHz6ZNSy12T2dMVklMd2c/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNQlRHRnY1NTBoRUE/view?usp=sharing)
  + [challenge #4](https://drive.google.com/file/d/0B22KXlqHz6ZNSnVrVlNWc3JUeG8/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNM2F3S283cWt3UU0/view?usp=sharing)
  + [challenge #5](https://drive.google.com/file/d/0B22KXlqHz6ZNQmhPd1dST3NJU2s/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNZW1SdnV3YVN4bTg/view?usp=sharing)
  + [challenge #6](https://drive.google.com/file/d/0B22KXlqHz6ZNOXhYV1pIVWdpS0k/view?usp=sharing) & [my possible solution](https://drive.google.com/file/d/0B22KXlqHz6ZNMUpkdVRIVVpHWms/view?usp=sharing) ([inspired from here](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article#Examples))
* Description:

These are my solutions to the semantic html hands-on exercises.

### FREE PREVIEW this video

* **Video: 106 / take 01**

## 

## Semantic HTML - header, nav, main

* not semantic
  + div
  + span
* semantic
  + There are more semantic elements than those shown here
  + These are the semantic elements primarily used for structure:
    - [header](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/header)
    - [nav](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/nav)
    - [main](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main)
    - [article](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article)
    - [section](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section)
    - [aside](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside)
    - [footer](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/footer)
    - [h1 - h6](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/Heading_Elements)
    - [figure](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figure)
    - [figcaption](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figcaption)
    - [address](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/address)

### header

* + a group of “introductory content” or “navigational aids”
    - may contain some heading elements but also other elements like a logo
  + must **not** be a descendent of
    - * header
      * footer
      * address

### nav

* + a section with navigation links
    - the links may link to other pages or to parts within a page
    - not all links within a document must be in a nav element
    - **intended for major block of navigation links**
      * the footer element often has a list of links that don’t need to be in a nav element
      * “it is common for footers to have a list of links to various key parts of a site, but the footer element is more appropriate in such cases, and no nav element is necessary for those links.” (src: [W3](https://www.w3.org/html/wg/drafts/html/master/semantics.html#the-nav-element))
  + a document may have several nav elements, for example, one for site navigation and one for intra-page navigation

### main

* + the main content of the <body> of a document or application
    - content that is the central topic of a document, or the central functionality of an application.
  + should be unique to the document
    - excludes content that is repeated across a set of documents
      * sidebars, navigation links, copyright information, site logos, and search forms (unless the document's main function is as a search form)
  + must **not** be a descendent of:
    - <article>, <aside>, <footer>, <header>, or <nav>
  + Only one main element can be used per document.
  + [see MDN example](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/main)
* Description:

The HTML **<header>** element represents a group of introductory or navigational aids. It may contain some heading elements but also other elements like a logo, wrapped section's header, a search form, and so on. The HTML **<nav>** element (HTML Navigation Element) represents a section of a page that links to other pages or to parts within the page: a section with navigation links. The HTML **<main>** element represents the main content of the <body> of a document or application. The main content area consists of content that is directly related to, or expands upon the central topic of a document or the central functionality of an application. This content should be unique to the document, excluding any content that is repeated across a set of documents such as sidebars, navigation links, copyright information, site logos, and search forms (unless the document's main function is as a search form).

* **Video: 107 / take 02**

## 

## Semantic HTML - article, section

### article

* + a **self-contained composition**
    - **independently distributable or reusable**
  + examples:
    - forum post
    - magazine or newspaper article
    - blog entry
    - user review
    - an object
    - any other independent item of content
  + typically includes a heading
    - h1 - h6
  + Usage notes:
    - When an <article> element is nested, the inner element represents an article related to the outer element. For example, the comments of a blog post can be <article> elements nested in the <article> representing the blog post.
    - Author information of an <article> element can be provided through the <address> element, but it doesn't apply to nested <article> elements.
  + [see MDN example](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/article)

### section

* + a thematic grouping of content
  + typically includes a heading
    - h1 - h6
  + Usage notes :
    - do not use the <section> element as a generic container
      * this is what <div> is for, especially when the sectioning is only for styling purposes.
    - If something can stand on its own, use an <article> element
  + [see MDN example](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/section)
* Additional reading
  + [W3C Article & Section](https://www.w3.org/TR/html5/sections.html#the-article-element)
  + [HTML5 Doctor: Article vs Section vs Div](http://html5doctor.com/the-article-element/)
* Description:

The HTML **<article>** element represents a self-contained composition in a document, page, application, or site, which is intended to be independently distributable or reusable (e.g., in syndication). This could be a forum post, a magazine or newspaper article, a blog entry, an object, or any other independent item of content. Each <article> should be identified, typically by including a heading (<h1>-<h6> element) as a child of the <article> element. The HTML **<section>** element represents a generic section of a document, i.e., a thematic grouping of content, typically with a heading. Each <section> should be identified, typically by including a heading (<h1>-<h6> element) as a child of the <section> element.

* **Video: 108 / take 03**

## 

## Semantic HTML - aside & footer

### aside

* + content related to something
  + examples:
    - sidebars
    - inserts
    - advertisements
    - biography of the author
    - related links on the blog
  + see examples: [MDN](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/aside), [W3Schools](http://www.w3schools.com/tags/tag_aside.asp), [HTML5Doctor](http://html5doctor.com/aside-revisited/)

### footer

* + examples:
    - information about the author of the section
    - copyright data
    - links to related documents
  + Usage notes:
    - Enclose information about the author in an <address> element that can be included into the <footer> element
* Description:

The HTML **<aside>** element represents a section of the page with content connected tangentially to the rest, which could be considered separate from that content. These sections are often represented as sidebars or inserts. They often contain the definitions on the sidebars, such as definitions from the glossary; there may also be other types of information, such as related advertisements; the biography of the author; web applications; profile information or related links on the blog. The HTML **<footer>** element represents a footer for its nearest sectioning content or sectioning root element. A footer typically contains information about the author of the section, copyright data or links to related documents.

* **Video: 109 / take 01**

## 

## Semantic HTML - Examples

* Description:

These examples demonstrate how to structure html documents using semantic html.

* **File:** [**027\_structuring-documents / 01 - 06**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 110 / take 01**

## 

## Semantic HTML - headings - h1 - h6

### h1 - h6

* + headings
  + structure content on a page by topic & importance
    - headings are important for document outline & SEO
* Description:

Heading elements implement six levels of document headings, <h1> is the most important and <h6> is the least. A heading element briefly describes the topic of the section it introduces. Heading information may be used by user agents, for example, to construct a table of contents for a document automatically.

* **File:** [**027\_structuring-documents / 07**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 111 / take 01**

## 

## Semantic HTML - figure, figcaption

### figure

* + self-contained content, frequently with a caption (<figcaption>)
  + While it is related to the main flow, its position is independent of the main flow; it could be moved to another page or an appendix without affecting the main flow.
  + typically referenced as a single unit
  + examples:
    - image
    - illustration
    - diagram
    - code snippet
  + Usage notes:
    - A caption can be associated with the <figure> element by inserting a <figcaption> inside it (as the first or the last child).
  + [see MDN examples](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figure#Examples)

### figcaption

* + a caption for a <figure>
  + the first or last element inside a <figure> block
  + optional
    - if not provided, <figure> will have no caption

### address

* + The HTML <address> element supplies contact information for its nearest <article> or <body> ancestor; in the latter case, it applies to the whole document.
  + Usage notes:
    - To represent an arbitrary address, one that is not related to the contact information, use a <p> element rather than the <address> element.
    - Typically an <address> element can be placed inside the <footer> element of the current section, if any.
  + [see MDN example](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/address#Example)
* Description:

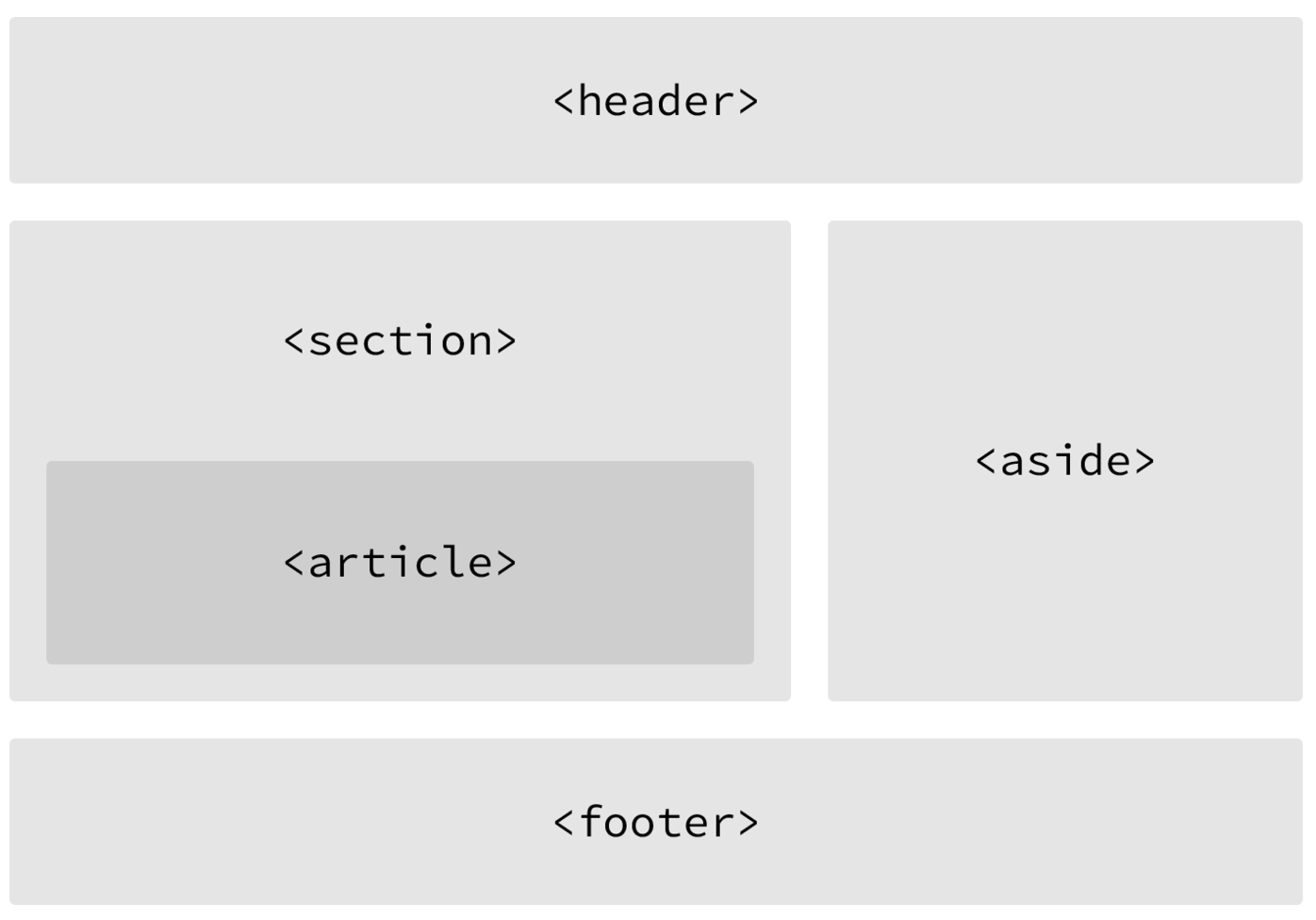
The HTML **<figure>** element represents self-contained content, frequently with a caption (<figcaption>), and is typically referenced as a single unit. Usually this is an image, an illustration, a diagram, a code snippet, or a schema that is referenced in the main text. The HTML **<figcaption>** element represents a caption or a legend associated with a figure or an illustration described by the rest of the data of the <figure> element which is its immediate ancestor which means <figcaption> can be the first or last element inside a <figure> block. Also, the HTML Figcaption Element is optional; if not provided, then the parent figure element will have no caption. The HTML **<address>** element supplies contact information for its nearest <article> or <body> ancestor; in the latter case, it applies to the whole document.

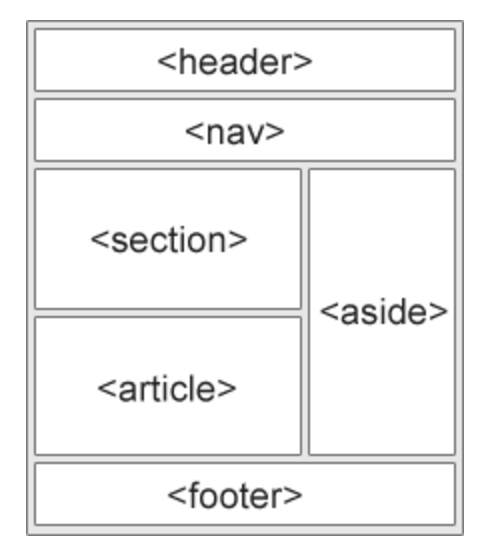
* **Video: 112 / 01**

## 

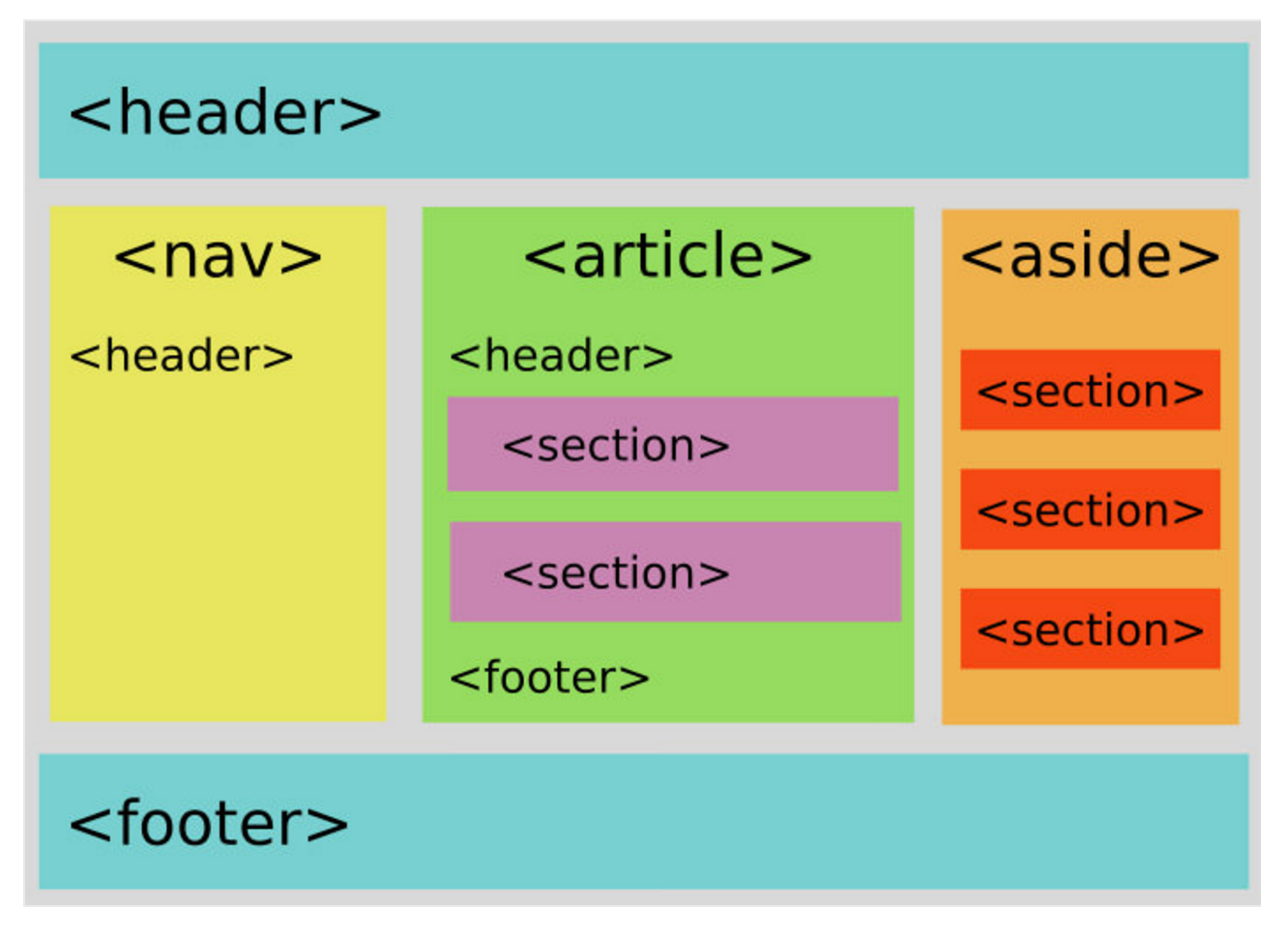
## Hands-On Exercises

* Create each of the following pages. Only create the HTML structure. Do not style the pages.

  
source: [Shay Howe](http://learn.shayhowe.com/html-css/getting-to-know-html/)

  
source: [W3 Schools](http://www.w3schools.com/html/html5_semantic_elements.asp)

  
source: [Jennifer Marsman](http://blogs.msdn.com/b/jennifer/archive/2011/08/01/html5-part-1-semantic-markup-and-page-layout.aspx)

  
source & **code example**: [hongkiat](http://www.hongkiat.com/blog/html-5-semantics/)

* Description:

This hands-on example will help you learn how to write semantic html into a document.

* **Video: 113 / take 01**

## 

## Hands-On Exercises - Solutions

* Description:

These are my solutions to the hands-on exercise showing you how to write semantic html into a document.

* **File:** [**028\_structuring-documents**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 114 / take 01**

## 

## Hands-On Exercises - Learning HTML Elements

* [Content Sectioning & Text Content](http://goo.gl/forms/GN1X7lA5Py)
* Description:

Click the link above. Answer the questions. Please note, these questions are not graded. To see the correct answers, watch my solution in the next video.

* **Video: 115 / take 01**

## 

## Hands-On Exercises - Learning HTML Elements - My Solutions

* Description:

Here are my answers to the questions in this hands-on exercise.

* **Video: 116 / take 01**

## 

## Review

* document structure
* importance of good document structure
  + maintainability
  + readability
  + accessibility
  + search engine rank
* terminology
  + semantic HTML
  + structure vs layout
  + document object model (DOM)
  + document flow
  + content categories
  + HTML5 Outline algorithm - don’t use
    - Use headings instead
* take-away:
  + don’t just use div’s
  + use semantic html
    - the most specific and applicable element
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

In this section, we have learned more about structuring content in an html document. In particular, we focused on using semantic html. There are many benefits to semantic html which include: maintainability, readability, accessibility, and higher search ranking. We learned about many semantic elements for structuring content.

* **Video: 117 / take 01**

# 

# Layout Fundamentals - Box Model & Display

## 

## Section Overview

* layout vs structuring documents
* layout resources & review
  + box model
  + box-sizing: border-box
  + rounding & centering
* display property
  + block
  + inline
  + inline-block
  + none
* hands-on exercises
* Description:

There are certain fundamentals that must be known when it comes to working with the layout of objects on an html page: the box model, the box-sizing property, the border-radius property, the margin property, and the display property. In this section, we will review and reinforce concepts to which we have already been exposed and we will also learn some new concepts important to working with layout well.

* **Video: 118 / take 01**

## 

## Layout Resources & Review

* resources
  + <http://learnlayout.com/>
  + [css tricks flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)
* box model reminder
  + everything on a web page is a box
  + **File:** [**029\_layout / 01\_box-model\_reminder**](https://github.com/GoesToEleven/html-css-bootcamp/)
* box-sizing: border-box
  + Prevents padding and border from increasing size of box
  + **File:** [**029\_layout / 02\_box-sizing\_border-box**](https://github.com/GoesToEleven/html-css-bootcamp)
* rounding & centering
  + border-radius property
    - 100%; (TRBL)
    - 10% 30%; (TB RL)
    - 10% 50% 10% 35%; (T R B L)
  + margin property
    - 0 auto; (TB RL)
    - 20px auto; (TB RL)
* Description:

Two valuable resources are given for learning about, and working with, layout. The box-model, box-sizing: border-box, border-radius, and margin property are all reviewed.

* **Video: 119 / take 02**

## 

## Display Property Review

* display property
  + block vs Inline
    - block
      * begins a new line
      * takes up the entire line; takes up as much width as possible
      * you can nest inside a block level element:
        + inline elements
        + block level elements
    - inline
      * does not begin a new line
      * takes up as little width as possible
      * you can nest inside an inline level element:
        + inline elements
  + the display property may have other values
    - [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/display#Syntax)
  + we will primarily use
    - block
    - inline
    - inline-block
    - none
    - flex
      * more on this soon
* Description:

The display property is reviewed. New examples are given for some of the display properties different settings.

* **Files:** [**030\_display / 01\_display-property**](https://github.com/GoesToEleven/html-css-bootcamp)
  + 01\_display
  + 02\_display\_block
  + 03\_display\_inline
  + 04\_display\_inline-block
  + 05\_display\_none
* **Video: 120 / take 01**

## 

## Hands-On Exercise - Page of Divs

* display property
* Description

Create a page filled with divs. Change the default display property of divs (block-level) to a setting which allows the divs to be displayed one after another horizontally. [Your page should look like this.](https://drive.google.com/file/d/0B22KXlqHz6ZNVkl1MDM1cGVuWmc/view?usp=sharing)

* **Files:** [**030\_display / 02\_hands-on-display**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 121 / take 02**

## 

## Hands-On Exercise - Page of Divs - Solution

* display property
* Description

This is my solution to creating a page with divs that line up, one next to the other, in an inline fashion.

* **Files:** [**030\_display / 02\_hands-on-display**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 122 / take 01**

## 

## Hands-On Exercise - Horizontal Menu

* ul & display property

### list-style

* Description

For this hands-on exercise, create a menu using the ul element and the display property. Have that menu displayed in a horizontal fashion. Use the list-style css property to remove the bullet points.

* **Files:** [**030\_display / 03 hands-on-menu**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 123 / take 01**

## 

## Hands-On Exercise - Horizontal Menu - Solution

* ul & display property
* block, inline, inline-block
  + Inline elements:
    - respect left & right margins and padding, but **not top & bottom**
    - **cannot have a width and height set**
    - allow other elements to sit to their left and right.
  + Block elements:
    - respect all of those
    - force a line break after the block element
  + Inline-block elements:
    - allow other elements to sit to their left and right
    - respect top & bottom margins and padding
    - respect height and width



* Description

Here is my solution to the hands-on challenge of creating a horizontal menu.

* **Files:** [**030\_display / 03 hands-on-menu**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 124 / take 01**

## 

## Taco Shop - Display Property

### vertical-align

* Description

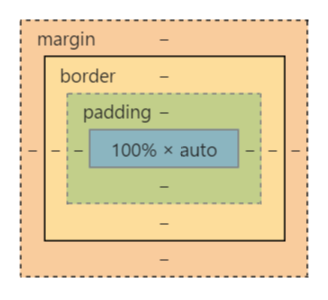
We can create a basic layout with a menu on the left and content on the right by using the display properties inline-block value. We also learned about the CSS vertical-align property: The vertical-align CSS property specifies the vertical alignment of an inline or table-cell box.

* **Files:** [**030\_display / 04\_taco-shop\_and-no-space**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 125 / take 01**

## 

## Review

* box model



* box-sizing: border-box
* rounding & centering
  + border-radius
    - TRBL
    - TB RL
    - T R B L
  + margin: 0 auto;
    - TRBL
    - TB RL
    - T R B L
* display property
  + block
  + inline
  + inline-block
  + none
* resources
  + <http://learnlayout.com/>
  + [css tricks flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)
* block, inline, inline-block
  + Inline elements:
    - respect left & right margins and padding, but **not top & bottom**
    - **cannot have a width and height set**
    - allow other elements to sit to their left and right.
  + Block elements:
    - respect all of those
    - force a line break after the block element
  + Inline-block elements:
    - allow other elements to sit to their left and right
    - respect top & bottom margins and padding
    - respect height and width



* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + vertical-align
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

We’ve learned new aspects of layout fundamentals and deepened our understanding of other aspects of layout fundamentals which we already knew. There are certain fundamentals that must be known when it comes to working with the layout of objects on an html page: the box model, the box-sizing property, the border-radius property, the margin property, and the display property. We also deepened our understanding of the display property values block, inline, and inline-block.

* **Video: 126 / take 2**

# 

# Layout with Flexbox

## 

## Section Overview

* We have learned about
  + box model
  + box-sizing: border-box
  + border-radius
    - TRBL
    - TB RL
    - T R B L
  + margin: 0 auto;
    - TRBL
    - TB RL
    - T R B L
  + display property
    - block
    - inline
    - inline-block
    - none
    - flex - this is what we’re going to learn about now!
* CSS properties & values we will use for layout
  + Display
    - Block
    - Inline
    - Inline-block
    - None
    - Flex
  + Position
    - Static
    - Relative
    - Absolute
    - Fixed
  + Float
    - Left
    - Right
* <http://caniuse.com/#search=flexbox>
* Description:

Flexbox is a powerful tool for layout. We activate and use flexbox by setting the display property to the value of flex. We have already seen the css display property in use with the values of block, inline, and inline-block. After we have learned flexbox, we will also learn a few other tools for creating layouts. We will learn the css position property and the css float property. Examples are shown of what we will learn to do with flexbox. There is also a fun game which we will use to learn flexbox called “flexbox froggy”.

* **Video: 127 / take 02**

## 

## Understanding Flexbox - Container & Items

* Container & Items
  + Create a container
  + That container’s immediate children ( > ) are flex items
* [css tricks flexbox](https://css-tricks.com/snippets/css/a-guide-to-flexbox/)

### Container properties

### display

* + - flex
    - inline-flex

### flex-wrap

* + - Nowrap (default)
    - Wrap
    - Wrap-reverse

### flex-direction

* + - Row (default)
    - Row-reverse
    - Column
    - Column-reverse

### flex-flow

* + - Shorthand: Flex-direction Flex-Wrap

### justify-content (main axis)

* + - Flex-start (default)
    - Flex-end
    - Center
    - Space-between
    - Space-around

### align-items (cross axis)

* + - Flex-start
    - Flex-end
    - Center
    - Stretch (default)
    - Baseline

### align-content (cross axis)

* + - Flex-start
    - Flex-end
    - Center
    - Stretch (default)
    - Space-between
    - Space-around

### Item properties

### align-self

* + - auto
    - flex-start
    - flex-end
    - center
    - baseline
    - Stretch

### order

* + - Numeric value

### flex-grow

* + - Numeric value
    - 0 (default)

### flex-shrink

* + - Numeric value
    - 1 (default)

### flex-basis

* + - Some width value
    - Auto (default)

### flex

* + - Flex-grow, flex-shrink, flex-basis
    - 0 1 auto (default)
* Learn about flexbox by playing with properties
* Description:

To use flexbox, we must understand the concepts of container, items (which are all immediate children of the container), primary axis, and cross-axis. Once you know how to use flexbox, an entire world of layout design possibilities open up to you.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 128 / take 01**

## 

## Containers & Creating A Flex Container

### FREE PREVIEW this video

* Start to think in terms of containers, even when not using flexbox
  + Containers hold things
  + The html container holds head and body
  + The body container holds other tags
  + The header container holds other elements you want in the header
  + You use containers to structure your content
  + You use containers also for laying out your content
* You make any element a flex container by applying this css property value to it
  + display: flex;
* Learn about flexbox by playing with properties
* Description:

To use flexbox, we must understand the concepts of container and items (which are all immediate children of the container). When talking about containers, however, I need to first speak generally about the importance of thinking in terms of containers as you create your layouts. You need to think in terms of containers when creating your layout. Build your pages, and your layouts, by breaking the different areas of a page up into containers. Now that everything is in a container, you can make containers as necessary “flex containers” and use flexbox to arrange what is in those containers. Once you learn to think in terms of containers, and once you know how to use flexbox containers, an entire world of layout layout possibilities open up to you.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 129 / take 02**

## 

## Container Property: flex-wrap

* **Flex-wrap**
  + Nowrap (default)
  + Wrap
  + Wrap-reverse
* Description:

Flex-wrap is a property which you can apply to an element which is a flex container. Another way to say that: flex-wrap is a flex container property. Flex-wrap lets you determine whether or not your flex items wrap when they reach the end of their row or column. The CSS flex-wrap property specifies whether flex items are forced into a single line or can be wrapped onto multiple lines. If wrapping is allowed, this property also enables you to control the direction in which lines are stacked.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 130 / take 02**

## 

## Container Property: flex-direction (primary-axis & cross-axis)

* **Flex-direction**
  + Row (default)
  + Row-reverse
  + Column
  + Column-reverse
* Primary axis
  + The axis which is parallel to flex-direction’s value
  + Whatever your flex-direction is set to, that is your primary axis
* Cross axis
  + The axis which is perpendicular to the primary axis
* examples:
  + flex-direction: row;
    - primary axis: horizontal
    - cross axis: vertical
  + flex-direction: column;
    - primary axis: vertical
    - cross axis: horizontal
* Description:

Flex-direction allows us to set the direction in which our flex items are laid out. “row” and “column” are the values to which we will usually set the flex-direction property. It is important for you to understand the concepts of primary axis and cross axis. Other flex container properties operate on either the primary or cross axis. To use those other flex properties, you will need to determine the direction of your primary and cross axis. The flex-direction CSS property specifies how flex items are placed in the flex container defining the main axis and the direction (normal or reversed).

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 131 / take 04**

## 

## Container Property: flex-flow

* Flex-flow: Flex-direction & Flex-wrap
  + **Flex-direction** 
    - Determines our primary axis
    - Values:
      * Row (default)
      * Row-reverse
      * Column
      * Column-reverse
  + **Flex-wrap**
    - Values:
      * Wrap
      * Nowrap (default)
      * Wrap-reverse
* examples:
  + flex-flow: row wrap;
  + flex-flow: row nowrap;
  + flex-flow: column wrap;
  + flex-flow: column nowrap;
* Description:

The flex-flow CSS property is a shorthand property for flex-direction and flex-wrap individual properties.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 132 / take 01**

## 

## Container Property: justify-content

* Allows you to position items along the **main axis**
  + Flex-direction: row
    - Main axis is horizontal
  + Flex-direction: column
    - Main axis is vertical
  + Values:
    - Flex-start
    - Flex-end
    - Center
    - Space-between
    - Space-around
* Description:

The justify-content property allows us to arrange items along the primary axis. The primary axis is also sometimes called the main axis. What is the primary axis? The primary axis is whatever you’ve set in the flex-direction property. If the flex-direction property is row, then the primary axis is row. If the flex-direction property is column, then the primary axis is column. We can use 5 different values when setting the justify-content property. The CSS justify-content property defines how the browser distributes space between and around flex items along the main-axis of their container.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 133 / take 01**

## 

## Container Property: align-items

* Allows you to position items along the **cross axis**
  + Flex-direction: row
    - Main axis is horizontal
    - **Cross-axis is vertical**
  + Flex-direction: column
    - Main axis is vertical
    - **Cross-axis is horizontal**
  + Values:
    - Flex-start
    - Flex-end
    - Center
    - Stretch
    - Baseline
* Description:

The CSS align-items property aligns flex items of the current flex line the same way as justify-content but in the perpendicular direction.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 134 / take 01**

## 

## Container Property: align-content

* Allows you to position all lines of content along the **cross axis. You must have multiple lines of content for align-content to have an effect.**
  + Flex-direction: row
    - Main axis is horizontal
    - **Cross-axis is vertical**
  + Flex-direction: column
    - Main axis is vertical
    - **Cross-axis is horizontal**
  + Values:
    - Flex-start
    - Flex-end
    - Center
    - Stretch
    - Space-between
    - Space-around
* Description:

The CSS align-content property aligns a flex container's lines within the flex container when there is extra space on the cross-axis. This property has no effect on single line flexible boxes.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 135 / take 01**

## 

## Item Property: align-self

* Allows you to position an individual flex item along the **cross axis,** overriding align-items
* Values:
  + auto
  + flex-start
  + flex-end
  + center
  + baseline
  + stretch
* Description:

The align-self CSS property aligns flex items of the current flex line overriding the align-items value. If any of the flex item's cross-axis margin is set to auto, then align-self is ignored.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 136 / take 01**

## 

## Item Property: order

* Arrange the order in which elements inside a flex container appear
* Description:

The CSS order property specifies the order used to lay out flex items in their flex container. Elements are laid out in the ascending order of the order value. Elements with the same order value are laid out in the order in which they appear in the source code.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 137 / take 01**

## 

## Item Property: flex-grow

* Allows flex items to grow in relation to each other inside a flex container
* Experiment with how flex-order interacts with
  + width
  + wrap / no-wrap
* Description:

The flex-grow CSS property specifies the flex grow factor of a flex item. It specifies what amount of space inside the flex container the item should take up.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 138 / take 01**

## 

## Item Property: flex-shrink

* Allows flex items to shrink in relation to each other inside a flex container
* Description:

The flex-shrink CSS property specifies the flex shrink factor of a flex item.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 139 / take 01**

## 

## Item Property: flex-basis

* Sets the initial size of a flex item
* Description:

The flex-basis CSS property specifies the flex basis which is the initial main size of a flex item. This property determines the size of the content-box unless specified otherwise using box-sizing.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 140 / take 01**

## 

## Item Property: flex

* Shorthand property for **flex-grow, flex-shrink, flex-basis**
* Description:

The flex CSS property is a shorthand property specifying the ability of a flex item to alter its dimensions to fill available space. Flex items can be stretched to use available space proportional to their flex grow factor or their flex shrink factor to prevent overflow.

* **FIles:** [**031\_flexbox**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 141 / take 01**

## 

## Hands-On Exercise - Flexbox Froggy

* <http://flexboxfroggy.com/>

Flexbox froggy is a game on a website that asks you to position frogs on lily pads using flex-box properties.

* **FIles:** [**031\_flexbox/03\_hands-on-exercises/**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 142 / take 01**

## 

## Hands-On Exercises

* Shorthand property for **flex-grow, flex-shrink, flex-basis**
* [Two challenges](https://docs.google.com/document/d/1smxcWloGucIXll-qlrKkrMNBxgXyBo4vpWzhgJiq-nk/edit?usp=sharing)
* Description:

These hands-on exercises will help reinforce everything you are learning about flexbox.

* **FIles:** [**031\_flexbox/03\_hands-on-exercises/**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 143 / take 01**

## 

## Hands-On Exercises - Solutions - 8 Pages

* Description:

My solutions to the first part of the hands-on exercise where you are requested to build 8 different pages by setting properties on the flex container.

* **FIles:** [**031\_flexbox/03\_hands-on-exercises/**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 144 / take 01**

## 

## Hands-On Exercises - Solutions - 6 Pages

* Description:

My solutions to the second part of the hands-on exercise where you are requested to build 6 different pages by setting properties on flex items.

* **FIles:** [**031\_flexbox/03\_hands-on-exercises/**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 145 / take 02**

## 

## Review

* Flexbox
* Containers & Items
  + Thinking in terms of “containers” when doing layout
* Main axis
* Cross axis
* Container properties & values
  + **Display**
    - flex
    - inline-flex
  + **Flex-direction**
    - Row (default)
    - Row-reverse
    - Column
    - Column-reverse
  + **Flex-wrap**
    - No-wrap (default)
    - Wrap
    - Wrap-reverse
  + **Justify-content (main axis)**
    - Flex-start (default)
    - Flex-end
    - Center
    - Space-between
    - Space-around
  + **Align-items (cross axis)**
    - Flex-start
    - Flex-end
    - Center
    - Stretch (default)
    - Baseline
  + **Align-content (cross axis)**
    - Flex-start
    - Flex-end
    - Center
    - Stretch (default)
    - Space-between
    - Space-around
* Item properties & values
  + **Align-self**
    - auto
    - flex-start
    - flex-end
    - center
    - baseline
    - Stretch
  + **Order**
    - Numeric value
  + **Flex-grow**
    - Numeric value
    - 0 (default)
  + **Flex-shrink**
    - Numeric value
    - 1 (default)
  + **Flex-basis**
    - Some width value
    - Auto (default)
  + **Flex**
    - Flex-grow, flex-shrink, flex-basis
    - 0 1 auto (default)
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

Flexbox is the main tool we will use for layout. When using flexbox, you have to think about the flex container and the flex items that container holds. There are various properties which we can set for both flex containers and flex items. We also learned the importance of understanding the main axis and the cross axis. Certain properties operate on the main axis exclusively, or the cross axis exclusively, We practice our skills using the fun online “flexbox froggy” game.

* **Video: 146 / take 1**

# 

# Media Queries

## Section Overview

* Understanding media queries
  + media type
  + expression
* media type
  + screen
  + print
* expressions
  + min-width
  + max-width
* viewport
* breakpoints
* Description

Media queries allow us to apply certain CSS in certain conditions. Media queries are used to tailor the formatting of output to certain device characteristics. This section will show you how to use media queries.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 147 / take 02**

## 

## Understanding Media Queries

* Media queries
  + Allow us to apply certain css in certain conditions
  + composed of a media type **and/or** a number of media features
* Media type (optional)
  + All
  + Screen
  + Print
  + Speech
* Media feature (“expression” - “zero or more”)
  + Min-width
  + Max-width
  + … and there are more …
* Examples
  + **In CSS**

@media (min-width: 900px)

@media (max-width: 900px)

@media print

@media screen and (min-width: 900px)

* + **In HTML link to stylesheet Preferred Method**

<link rel="stylesheet" href="mq-900-plus.css" media="(min-width: 600px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* + - **Preferred Method - HTML link to stylesheet**
      * Allows your code to be more modular
      * Only pulls down the code that is needed for a page
      * With http2, making multiple requests to a server no longer has a negative impact on performance
* Description

Media queries allow us to apply certain CSS in certain conditions. Media queries are used to tailor the formatting of output to certain device characteristics. The preferred method for applying a media query is in the HTML link to a stylesheet.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 148 / take 01**

## 

## Min-Width & Max-Width

* [MDN documentation](https://www.google.com/#q=media+queries+mdn)
  + optional media type
    - screen
    - print
  + zero or more expressions
    - min-width
    - max-width
* min-width
  + “From this minimum point and upwards …”
* max-width
  + “From this maximum point and downwards …”
* Viewport
  + The port through which you’re viewing a web page
  + Different than the screen
  + The viewport is pretty much the browser window
* Examples
  + **css applied when viewport is 600px and higher**

<link rel="stylesheet" href="mq-600-plus.css" media="(min-width: 600px)">

* + **css applied when viewport is 900px and under**

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

* Description

The media query expressions “min-width” and “max-width” allow us to apply certain CSS based upon the width of the viewport. The viewport is the port through which the web page is being viewed - this is also commonly known as the browser’s window, or just the browser. The “min-width” and “max-width” expressions can be counter-intuitive for people. “Min-width” means from this minimum point upwards. “Max-width” means from this maximum point downwards. You can also think of it as the “max-width” at which this css will be applied is this one point. For instance, if “max-width” was set to 700px, then the max-width at which the css in the media query would be applied would be 700px.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 149 / take 03**

## 

## Print Media Query

* Example
  + <link rel="stylesheet" href="mq-900-plus.css" media="print">
* Description

The print media query allows us to specify formatting for a page only when it is printed.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 150 / take 01**

## 

## Media Type & Expressions

* **Example**

<link rel="stylesheet" href="mq\_600-899.css" media="screen and (min-width: 600px) and (max-width: 899px)">

* Description

In CSS media queries, you can create a query that combines one media type and one or multiple expressions. Notice the title of this lecture: media type and expressions. You can have one media type and multiple expressions in a media query.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 151 / take 01**

## 

## Taco Shop & Scaling Font - Examples

* Examples of media queries
  + taco-shop
  + scaling font
* Description

The print media query allows us to specify formatting for a page only when it is printed.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 152 / take 02**

## 

## Responsive Design & Mobile First Design

* Responsive Design
  + Your web page responds to the size of the viewport
* Mobile First
  + Build your pages to look good on mobile first
  + Then add in break-points so that your pages work at larger resolutions
* Breakpoints based upon content
  + Create your breakpoints based upon your content
  + Have your content look good, and be readable, at different sizes
* 70 - 80 characters of text per line
  + Classic readability theory suggests that an ideal column should contain 70 to 80 characters per line (about 8 to 10 words in English)
  + When the width of a text block grows past about 10 words, a breakpoint should be considered
* Readability
  + [write like Hemingway](http://www.hemingwayapp.com/)
  + [measure readability - Flesch-Kincaid](https://readability-score.com/text/)
* Use
  + min-width
  + max-width
  + min-height
  + max-height
* Don’t use
  + min-device-width
  + max-device-width
  + min-device-height
  + max-device-height
* Description

With Web users primarily using mobile devices to browse web sites, developers need to be sure that their creations look as good and work as well on mobile devices as on traditional desktop computers. In the past, this was often accomplished by building separate sites: a mobile site and a desktop site. But a number of problems with this approach led to **responsive design**: making your page respond to the browser's viewport size. When creating responsive websites, it is good to build "**mobile first.**" Whether you design for mobile devices as a primary target or as a nice extra, you can use the power of CSS to ensure that the same content can be accessed across all platforms, from mobile phones to wide-screen high-resolution displays.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 153 / take 01**

## 

## Hands-On Exercises

* Description:

These hands-on exercises will help reinforce everything you are learning about media queries.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 154 / take 01**

## 

## Hands-On Exercises - Solutions

* Description:

My solutions to the hands-on exercises.

* **FIles:** [**032\_media-queries**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 155 / take 01**

## 

## Review

* Understanding media queries
  + media type
  + expression
* media type
  + screen
  + print
* expressions
  + min-width
  + max-width
* viewport
* breakpoints
* responsive design
* mobile first
* readability
  + 8 - 10 words per line (70 - 80 characters)
  + [write like Hemingway](http://www.hemingwayapp.com/)
  + [measure readability - Flesch-Kincaid](https://readability-score.com/text/)
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

Media queries allow us to apply certain CSS in certain situations. We learned that a media query can have a media type like screen or print, an expression like (min-width: 600px), or both. We talked more about the definition of the viewport. We also learned new terms like breakpoints responsive design, and mobile first. In addition, we covered some best practices for readability.

* **Video: 156 / take 1**

# 

# Flexbox Design Patterns

## 

## Section Overview

* Google’s flexbox design patterns
  + “Responsive web design patterns are quickly evolving, but there are a handful of established patterns that work well across the desktop and mobile devices”
  + Google’s Web Fundamentals
    - Flexbox design patterns
* Example layouts built with flexbox
* [Google’s Web Fundamentals](https://developers.google.com/web/fundamentals/?hl=en)
  + [Flexbox design patterns](https://developers.google.com/web/fundamentals/design-and-ui/responsive/patterns/?hl=en)
* Description:

We now understand how flexbox and media queries work. We can take that knowledge and use it to build different layouts. The best way to learn how to build layouts with flexbox and media queries is to look at various design patterns and examples. Google has design patterns just for the purpose: to help people learn how to build layouts with flexbox. Several other example layouts built with flexbox will also be provided and analyzed.

* **Files:** 
  + [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
  + [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 157 / take 3**

## 

## Google’s Flexbox Design Pattern #1

* Take-aways
  + Design:
    - Build mobile first
    - Use:
      * **display: flex;**
      * **flex-flow: row wrap;**
    - Each row:
      * **What you want on each row adds up to 100%**
* Description:

This is the first flexbox design pattern created by Google. You can see all of the flexbox design patterns on Google’s Web Fundamentals website.

* **Files:** [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 158 / take 1**

## 

## Google’s Flexbox Design Pattern #2

* Description:

This is the second flexbox design pattern from Google’s Web Fundamentals website.

* **Files:** [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 159 / take 1**

## 

## Google’s Flexbox Design Pattern #3

* Description:

This is the third flexbox design pattern from Google’s Web Fundamentals website.

* **Files:** [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 160 / take 1**

## 

## Hands-On Exercises

* Description:

These hands-on exercises will help reinforce Google’s Web Fundamentals flexbox design pattern.

* **Files:** [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 161 / take 1**

## 

## Hands-On Exercises - Solutions

* Description:

My solutions to the hands-on exercises.

* **FIles:** [**033\_flexbox-google**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 162 / take 01**

## 

## Flexbox Example #1 - Responsive Menu

* Responsive menu
* Description:

This is how you create an awesome, mobile first, responsive header menu using flexbox and media queries for your layout.

* **Files:** [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 163 / take 1**

## 

## Flexbox Example #2 - Above The Fold

* Centering content with flexbox
* Above the fold design pattern

### FREE PREVIEW this video

* Description:

The above the fold design pattern is an amazing, super great technique for getting content to always appear in the full screen of any viewport.

* **Files:** [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 164 / take 1**

## 

## Flexbox Example #3 - Holy Grail Example

* Description:

The “holy grail” layout is a layout which is commonly used when building web pages. This “holy grail” layout was achieved using the same flexbox design pattern recommended by Google Web Fundamentals.

* **Files:** [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 165 / take 1**

## 

## Flexbox Example #4 - Holy Grail Redux

* Description:

Here is yet another way you could write the CSS to create a holy grail layout.

* **Files:** [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 166 / take 2**

## 

## Hands-On Exercises

* Description:

These hands-on exercises will help reinforce everything you have been learning about flexbox design patterns.

* **Files:** [**034\_flexbox-examples**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 167 / take 1**

## 

## Review

* Google’s flexbox design patterns
  + [Google’s Web Fundamentals](https://developers.google.com/web/fundamentals/?hl=en)
    - [Flexbox design patterns](https://developers.google.com/web/fundamentals/design-and-ui/responsive/patterns/?hl=en)
  + Design:
    - Build mobile first
    - Use:
      * **display: flex;**
      * **flex-flow: row wrap;**
    - Each row:
      * **What you want on each row adds up to 100%**
* Example layouts built with flexbox
  + Menu
  + Above the fold
  + Holy grail
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

There are many different ways you create layouts with flexbox. Google’s Web Fundamentals offers one design pattern. Several other examples were presented as to how you can create layouts with flexbox. One of the greatest things covered in this section, and in the entire course, is how to create “above the fold” layouts.

* **Video: 168 / take 1**

# 

# Layout with Position

## 

## Section Overview

* Layout Essentials - A Quick Reference
* Position overview
* CSS property, and its values, to position elements
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
* CSS properties to position positioned elements
  + Top
    - <length>
    - percentage
  + Right
    - <length>
    - percentage
  + Bottom
    - <length>
    - percentage
  + Left
    - <length>
    - percentage
* Taco Shop
* Hands-on Exercises
* Description:

The CSS position property is an important tool for layout. The position property allows us to position elements on the page. We can position an element relative to the viewport, relative to itself, and relative to some other element. We can also overwrite the positioning of an element by setting the position property to the value of static. In this section, you will also receive an incredibly valuable resource - a quick reference guide to the most commonly used layout tools.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 169 / take 1**

## 

## Layout Essentials - A Quick Reference

* [Layout essentials](https://docs.google.com/document/d/1IjV1IlnEXKPu5IFdWtr9Fubl_RZEVJdmPgaBQOjomU0/edit?usp=sharing)
  + Short URL: <https://goo.gl/k7hbXq>

### FREE PREVIEW this video

* Description:

This is a great resources. My students at the college and university have found this document to be extremely helpful. This document summarizes in one location the most common tools you will use for layout. This is a great reference to print out and tape to your wall at your desk.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 170 / take 1**

## 

## Position Overview

* CSS property, and its values, to position elements
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
* CSS properties to position positioned elements
  + Top
    - <length>
    - percentage
  + Right
    - <length>
    - percentage
  + Bottom
    - <length>
    - percentage
  + Left
    - <length>
    - percentage

### FREE PREVIEW this video

* Description:

The position CSS property is good for positioning elements. A positioned element is an element whose computed position property is either relative, absolute, fixed or sticky.

A **fixed** positioned element is an element whose position is **fixed at a certain location on the viewport.** *The element is taken out of the normal flow of the DOM - said another way, the element is taken out of the normal document flow.*

A **relative** positioned element is an element whose computed position property is **relative to itself**. *The space the element formally held in the DOM is still blocked out.*

An **absolute** positioned element is *removed from the normal flow of the DOM.* An absolute positioned element is an element whose computed position property is **calculated from the next element up the DOM** which has a position property **or the html element if nothing up the DOM has a position property.**

When the **static** value is used for the position property, the element uses the **normal document flow** behavior, that is, the element is laid out in its current position in the flow. The top, right, bottom, left and z-index properties do not apply.

The **top**, **right**, **bottom**, and **left** properties **specify the position of positioned elements.**

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 171 / take 1**

## 

## Position Fixed

* **Position**
  + Fixed
* **Top**
  + <length>
  + percentage
* **Right**
  + <length>
  + percentage
* **Bottom**
  + <length>
  + percentage
* **Left**
  + <length>
  + Percentage
* Fixes an element in a fixed position in relation to the viewport
  + top, right, bottom, left
  + These properties set how far the element is located from top, right, bottom, or left
* Takes the element out of the normal document flow
* **FREE PREVIEW this video**
* Description:

A **fixed** positioned element is an element whose position is **fixed at a certain location on the viewport.** *The element is taken out of the normal flow of the DOM - said another way, the element is taken out of the normal document flow.*

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 172 / take 1**

## 

## Position Relative

* Relative to itself
* **Position**
  + Relative
* **Top**
  + <length>
  + percentage
* **Right**
  + <length>
  + percentage
* **Bottom**
  + <length>
  + percentage
* **Left**
  + <length>
  + Percentage

### FREE PREVIEW this video

* Description:

A **relative** positioned element is an element whose computed position property is **relative to itself**. *The space the element formally held in the DOM is still blocked out.*

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 173 / take 1**

## 

## Position Absolute

* **Position**
  + Absolute
* **Top**
  + <length>
  + percentage
* **Right**
  + <length>
  + percentage
* **Bottom**
  + <length>
  + percentage
* **Left**
  + <length>
* Positioned in relation to the next element up the DOM which has a position property
  + Or the body element if nothing up the DOM has a position property
* Description:

An **absolute** positioned element is *removed from the normal flow of the DOM.* An absolute positioned element is an element whose computed position property is **calculated from the next element up the DOM** which has a position property **or the html element if nothing up the DOM has a position property.**

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 174 / take 1**

## 

## Taco Shop - Position Property

* Description:

Here is how you create our taco shop example using the CSS position property.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 175 / take 1**

## 

## Hands-On Exercises

* Description:

Hands-on exercises help reinforce your skills. I strongly encourage you to do these exercises. Doing exercises tremendously helps students learn the material.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 176 / take 1**

## 

## Hands-On Exercises - Solutions

* Description:

Here are my solutions to the hands-on exercises.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 177 / take 1**

## 

## Review

* [Layout essentials](https://docs.google.com/document/d/1IjV1IlnEXKPu5IFdWtr9Fubl_RZEVJdmPgaBQOjomU0/edit?usp=sharing)
* CSS property, and its values, to position elements
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
* CSS properties to position positioned elements
  + Top
    - <length>
    - percentage
  + Right
    - <length>
    - percentage
  + Bottom
    - <length>
    - percentage
  + Left
    - <length>
    - Percentage
* Taco Shop
* Hands-on Exercises
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

The CSS position property helps us position elements on our html web page. We can position an element relative to the viewport, relative to itself, and relative to some other element. We can also overwrite the positioning of an element by setting the position property to the value of static. In this section, you will also receive an incredibly valuable resource - a quick reference guide to the most commonly used layout tools.

* **Files:** [**035\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 178 / take 1**

# 

# Layout with Float

## 

## Section Overview

* float
  + left
  + right
  + none
* Floating an element
  + Image example
* The “overflow” problem
  + the solution → overflow: auto;
* Clearing a float
* clear
  + left
  + right
  + both
* Description:

The float CSS property specifies that an element should be***taken from the normal flow***and placed along the **left or right side of its container,** where **text and inline elements will wrap around it.**

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 179 / take 1**

Float Images & Overflow: Auto

* Float an element
  + Image example
    - text and inline elements wrap around the floated image
* Overflow
  + A floated element will overflow its container if the container is too small
* overflow: auto
  + Prevents a floated element from overflowing its container
* Description:

The float CSS property specifies that an element should be taken from the normal flow and placed along the left or right side of its container, where text and inline elements will wrap around it. If the container is smaller than the element, then the element will “overflow” the container. We can prevent this overflow by using the CSS property “overflow” and setting it to “auto”. Philosophical ramblings are also presented at the end of the video.

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 180 / take 2**

## 

## Float Layout & Clearing Floats

* Clearing floats
* clear
  + left
  + right
  + both
* Description:

Once we have told an element to float, then that element has been removed from the normal document flow. Other elements will slide under the floating element. The way we prevent elements from sliding under the floated element is by “clearing” the float. For the element that we do not want sliding under the float, we apply the CSS “clear” property to it with whatever value is most appropriate: left, right, or both. That element, and all elements which follow it, will no longer slide under the floated element.

MDN says this about clearing floats: “The clear CSS property specifies whether an element can be next to floating elements that precede it or must be moved down (cleared) below them. The clear property applies to both floating and non-floating elements. When applied to non-floating blocks, it moves the border edge of the element down until it is below the margin edge of all relevant floats ....”

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 181 / take 3**

## 

## Float Example - Holy Grail

* Description:

This is the holy grail layout created with floats.

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 182 / take 1**

## 

## Taco Shop - Float Property

* Description:

Here is our taco shop layout created with float.

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 183 / take 1**

## 

## Hands-On Exercises

* [Hands-On Exercises - Descriptions](https://docs.google.com/document/d/193REAKxZgfwu_4BVsL2kuk58iU78qHqClAyAyAKAP70/edit?usp=sharing)
* Description:

These hands on exercises will help reinforce what you are learning about the CSS float property.

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 184 / take 1**

## 

## Hands-On Exercises - Solutions

* Description:

These are my solutions to the hands on exercises.

* **Files:** [**036\_float**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 185 / take 1**

## 

## Review

* Understanding float
  + The float CSS property specifies that an element should be***taken from the normal flow***and placed along the **left or right side of its container,** where **text and inline elements will wrap around it.**
* float
  + left
  + right
  + none
* Floating an element
  + Image example
* The “overflow” problem
  + the solution → overflow: auto;
* Clearing a float
* clear
  + left
  + right
  + both
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-image
    - background-image: url("../000\_img/pup.jpg");
  + background-size
    - background-size: cover;
  + background-repeat
    - background-repeat: no-repeat;
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

The CSS position property helps us position elements on our html web page. We can position an element relative to the viewport, relative to itself, and relative to some other element. We can also overwrite the positioning of an element by setting the position property to the value of static. In this section, you will also receive an incredibly valuable resource - a quick reference guide to the most commonly used layout tools.

* **Files:** [**036\_position**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 186 / take 1**

# 

# Background

## 

## Section Overview

* CSS properties
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
* Background nuances
  + Good to cover
  + Many students are confused by background until all of the individual properties are covered.
* Shorthand
  + Not recommended - do not use
  + Mixed implementation
  + sets missing properties to their initial values
    - Can overwrite previously set values
  + Some properties share the same value which causes confusion
* Description:

The background CSS property is a shorthand for setting the individual background values in a single place in the style sheet. background can be used to set the values for one or more of: background-clip, background-color, background-image, background-origin, background-position, background-repeat, background-size, and background-attachment. The background CSS shorthand property assigns explicit given values and sets missing properties to their initial values. Many students are confused by background until all of the individual properties are covered. Many students breathe a collective sigh of relief when they hear the recommendation not to use the background shorthand property.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 187 / take 1**

## 

## Background Color

### background-color

* + <color value>-
* Special keyword values
  + background-color: currentColor;
  + background-color: transparent;
* Description:

The background-color CSS property sets the background color of an element, either through a color value or the keyword transparent.

* **FREE PREVIEW this video**
* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 188 / take 1**

## 

## background-image, background-repeat, background-size

### background-image

* + background-image: url(http://www.example.com/bck.png);

### background-repeat

* + no-repeat
  + repeat-x
  + repeat-y
  + repeat

### background-size

* + cover
  + contain
* Description:

The CSS **background-image** property sets one or several background images for an element. The images are drawn on stacking context layers on top of each other. The first layer specified is drawn as if it is closest to the user. If a specified image cannot be drawn (for example, when the file denoted by the specified URI cannot be loaded), browsers handle it as they would a none value. The **background-repeat** CSS property defines how background images are repeated. A background image can be repeated along the horizontal axis, the vertical axis, both axes, or not repeated at all. The **background-size** CSS property specifies the size of the background images. Todd also slips into his southern roots in this video when talking about his dog.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 189 / take 3**

## 

## background-position, background-origin

### background-position

* + top;
  + bottom;
  + left;
  + right;
  + center;
  + 100px 10px;
  + 25% 75%;
  + 0 0;

### background-origin

* + content-box
  + padding-box
  + border-box
* Description:

The **background-position** CSS property lets you move an element (position it) around. The **background-origin** CSS property determines where the picture starts and ends.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 190\_01 / take 1**

## 

## background-attachment

### background-attachment

* + scroll
    - scrolls with the element
  + fixed
    - stays in place on the viewport (element leaves it behind with scroll)
* Description:

If a background-image is specified, the **background-attachment** CSS property determines whether that image's position is fixed within the viewport, or scrolls along with its containing block.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 190\_02 / take 1**

## 

## background

### background

* Do not use
  + Mixed implementation
  + sets missing properties to their initial values
    - Can overwrite previously set values
  + Some properties share the same value which causes confusion
* Description:

**I recommend not using the background shorthand property.** That said, here’s a description of the background property: The **background** CSS property is shorthand for setting the individual background values in a single place in the style sheet. background can be used to set the values for one or more of: background-clip, background-color, background-image, background-origin, background-position, background-repeat, background-size, and background-attachment..

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 191 / take 1**

## 

## Hands-On Exercises

* Description:

These hands on exercises will help reinforce what you are learning about the CSS background property.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 192 / take 1**

## 

## Hands-On Exercises - Solutions

* Description:

To see the solutions for the hands-on exercises, look at the files used for the lectures.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)

## 

## Review

* Be clear instead of concise and clever
  + Make your code readable
* CSS properties
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
* Didn’t talk about these
  + background-attachment
    - If a background-image is specified, the background-attachment CSS property determines whether that image's position is fixed within the viewport, or scrolls along with its containing block.
  + background-clip
    - The background-clip CSS property specifies whether an element's background, either the color or image, extends underneath its border.
* Shorthand
  + Not recommended - do not use
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

The CSS position property helps us position elements on our html web page. We can position an element relative to the viewport, relative to itself, and relative to some other element. We can also overwrite the positioning of an element by setting the position property to the value of static. In this section, you will also receive an incredibly valuable resource - a quick reference guide to the most commonly used layout tools.

* **Files:** [**037\_background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 193 / take 2**

## 

# Refactor Challenge

## Section Overview

* Starting files
* Refactoring code
  + Code refactoring is the process of restructuring existing computer code—changing the factoring—without changing its external behavior. Refactoring improves nonfunctional attributes of the software.
* Description:

We are going to start with an html page and it’s css code. We will then analyze that code to see what could be made better. We will draw upon everything we have learned in the course so far. We will then improve the code so that we have better code that conforms to today’s standards.

* **Files:** [**038\_refactor-challenge**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 194 / take 1**

## 

## Analyzing The Starting Files

* Things to fix about the page
  + style.css to main.css
  + Make the heading smaller
  + Make the menu horizontal
  + Give transparency to colors
  + div soup
    - Use semantic html
  + Don’t use background shorthand property
  + Use a reset stylesheet
  + Keep my code DRY
    - Don’t Repeat Yourself
* Description:

First we need to analyze our files to see what needs to change. Once we have determined what to change, then we can being the process of refactoring our code.

* **Files:** [**038\_refactor-challenge**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 195 / take 3**

## 

## inline-block Refactor

* inline-block
* Description:

We will use inline-block to get make our menu of list items horizontal

* **Files:** [**038\_refactor-challenge**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 196 / take 1**

## 

## flexbox Refactor

* flexbox

### !important rule

* Description:

We will use flexbox to make our menu of list items horizontal. We will also use flexbox for other layout tasks. In addition, information on debugging, the css !important rule, and specificity.

* **Files:** [**038\_refactor-challenge**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 197 / take 1**

# 

# Full-Page Above-The-Fold

## 

## Section Overview

* Responsive
* full page
* above-the-fold
* Background image
* Use our “[layout essentials](https://docs.google.com/document/d/1IjV1IlnEXKPu5IFdWtr9Fubl_RZEVJdmPgaBQOjomU0/edit?usp=sharing)” page
* Description:

We are going to build a responsive, full page, above the fold web page. We will do this in steps. There are starting files for you to use. I will introduce each step. You will then complete the step on your own. I will then show you how I completed the step.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 198 / take 4**

## 

## Step #1: HTML Structure

* Description:

What HTML structure would you use to create the page?

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 199 / take 1**

## 

## Step #1: HTML Structure - Solution

* above the fold
* scoping css
* button
* html entities: &nbsp;
* Description:

Here is the HTML structure I used.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 200 / take 4**

## 

## Step #2: Layout

* Description:

Use flexbox to get your text appear in the correct position.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 201 / take 1**

## 

## Step #2: Layout - Solution

* Flexbox
* Description:

Here is how I used flexbox to get the text into the correct position.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 202 / take 1**

## 

## Step #3: background image

* Description:

Add a background image to cover the above-the-fold region.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 203 / take 1**

## 

## Step #3: background image - Solution

* background-color
* background-image
* background-repeat
  + no-repeat
  + repeat-x
  + repeat-y
  + repeat
* background-position
  + top;
  + bottom;
  + left;
  + right;
  + center;
  + 100px 10px;
  + 25% 75%;
  + 0 0;
* background-size
  + cover
  + contain
* background-origin
  + content-box
  + padding-box
  + border-box
* background-attachment
  + scroll
    - scrolls with the element
  + fixed
    - stays in place on the viewport (element leaves it behind with scroll)
* Description:

Here is how I added a background image to cover the above-the-fold region.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 204 / take 1**

## 

## Step #4: Format Text

* Google fonts
  + Kotta One
  + Cantarell:400,700
* Description:

Format the text using google fonts.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 205 / take 1**

## 

## Step #4: Format Text - Solution

* Google fonts
  + Kotta One
  + Cantarell:400,700
* Description:

How I formatted the text.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 206 / take 2**

## 

## Step #5: Style The Button

* Description:

For this next step, style the button.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 207 / take 1**

## 

## Step #5: Style The Button - Solution

* Description:

This is how I styled the button.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 208 / take 1**

## 

## Step #6: Mobile

* <meta name="viewport" content="width=device-width, initial-scale=1">
* Description:

For this next step, make your site mobile-first responsive.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 209 / take 2**

## 

## Step #6: Mobile - Solution

* Description:

This is how I made the site mobile-first responsive.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 210 / take 11**

## 

## Review

* scoping css
* button element
* html entities
  + &nbsp;
* <meta name="viewport" content="width=device-width, initial-scale=1">
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

We’ve learned some great things in this section. We’re reinforcing our growing skillset. We also learned about some new items: button, html entities, !important, meta viewport. Your skillset is growing and you’re starting to be able to create some cool looking pages. These pages don’t rely on any framework or library - they are pure css and html. As such, they are very light and quick.

* **Files:** [**039\_full-page-background**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 211 / take 1**

# 

# Expanding Your Skillset

## 

## Section Overview

* html elements
  + q
  + blockquote
  + cite
  + pre
  + samp
  + code
  + kbd
  + abbr
* html element attributes
  + q, blockquote
    - cite
* HTML entities
* Linking images and bookmarks
* box-shadow
* vertical-align
* meta viewport
* Description:

We are going to learn some very important things in this section. Most importantly, we are going to learn about building well for mobile. You will learn all about “meta viewport” and how it works. You will also learn about physical pixels, device independent pixels, and css pixels. In addition to that, we will learn about linking images and linking to different places within the same document which is also known as bookmarks.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 212 / take 1**

## 

## q, blockquote, & cite elements; cite attribute

* html elements

### q

### blockquote

### cite

* html element attributes
  + q, blockquote

### cite

* Description:

The HTML Quote Element (**<q>**) indicates that the enclosed text is a short inline quotation. This element is intended for **short quotations that don't require paragraph breaks**; for long quotations use <blockquote> element. Most modern standards-aware browsers, like Mozilla Firefox, Opera, and Safari, should add quotes around text enclosed within the <q> element. Some browsers, like Internet Explorer, may not make any sort of style change for quotations, but it is possible to apply a style rule. A URL for the source of the quotation may be given using the *cite attribute*, while a text representation of the source can be given using the *<cite> element*.

The HTML **<blockquote>** Element (or HTML Block Quotation Element) indicates that the enclosed text is **a long quotation.** Usually, this is rendered visually by indentation. A URL for the source of the quotation may be given using the *cite attribute*, while a text representation of the source can be given using the *<cite> element*.

The HTML Citation Element (**<cite>**) represents **a reference to a creative work**. It must include the title of a work or a URL reference, which may be in an abbreviated form.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 213 / take 2**

## 

## pre, samp, code, kbd, abbr elements

* html elements

### pre

### samp

### code

### kbd

### abbr

* Description:

The HTML **<pre>** element (or HTML Preformatted Text) represents **preformatted text.** Text within this element is typically displayed in a **monospace** font exactly as it is laid out in the file. Whitespace inside this element is displayed as typed. Note: you will need to escape any contained '<' characters as '&lt;' to make sure enclosed code is not interpreted as markup.

The HTML **<samp>** element is an element intended to identify sample **output from a computer program.** It is usually displayed in the browser's default **monospace** font (such as Lucida Console).

The HTML Code Element (**<code>**) represents a **fragment of computer code.** By default, it is displayed in the browser's default **monospace** font.

The HTML Keyboard Input Element (**<kbd>**) **represents user input** and produces an inline element displayed in the browser's default **monospace** font.

The HTML <**abbr**> element (or HTML Abbreviation Element) represents **an abbreviation** and optionally provides a full description for it. If present, the *title attribute* must contain this full description and nothing else. The title attribute becomes a hovering tooltip.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 214 / take 3**

## 

## HTML Entities

* Reserved characters
  + &
  + <
  + >
  + “
* Description:

HTML reserves the following four characters: & < > “

Browsers interpret these characters instead of rendering them in the output.

To display these characters in your content, use the entity.

An HTML entity is a string that begins with '&' and ends with ';'.

Entities can be used to represent any character in HTML.

Use entities for reserved characters and strange characters.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 215 / take 1**

## 

## Linking Images & Bookmarks

* Linking images
* Bookmark
  + Linking to an element within a page
* Description:

We can make an image a clickable link. To do this, we surround the <img> tag with an <a> tag. We can also create links within a document to take us to different places in the document. This is known as a “bookmark.”

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 216 / take 1**

## 

## box-shadow Property

### [box-shadow](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow)

* + accepts a comma-separated list: x, y, blur radius, color
    - offset-x | offset-y | blur-radius | color
    - block-shadow: 1px 1px 2px black;
* Description:

The **box-shadow** property describes one or more shadow effects as a comma-separated list. It enables you to cast a drop shadow from the frame of almost any element. If a border-radius is specified on the element with a box shadow, the box shadow takes on the same rounded corners. [Box-shadow generator](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Box_Model/Box-shadow_generator) is an interactive tool allowing you to generate a box-shadow.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 217 / take 2**

## 

## vertical-align property

### vertical-align

* + We have already seen this
  + Just wanted to provide this example
* Description:

The vertical-align CSS property specifies the vertical alignment of an inline or table-cell box.

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 218 / take 1**

## 

## mobile devices - mobile history & font-boosting

* Mobile history
* Font boosting
* Description:

Most developers don’t understand what the meta tag with its name attribute set to viewport does. As a developer, you should understand every piece of code in your application. To understand “meta viewport”, you need to understand the history of how mobile devices displayed web pages and a concept known as “font boosting.”

* **Files:** [**041\_mobile**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 219 / take 1**

## 

## mobile devices - meta viewport width=device-width

* Description:

The meta tag with the name attribute set to viewport and the content attribute set to width=device-width. What does the “width=device-width” value do? It sets the width of the viewport equal to the width of the device. Instead of the viewport randomly being set to 980px, or some other width, the width of the viewport will now equal the device’s width. This results in fonts being displayed at a more legible size.

* **Files:** [**041\_mobile**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 220 / take 3**

## 

## mobile devices - meta viewport initial-scale=1

* pixels
  + physical pixels
  + device-independent pixels
  + css pixels
* [Google web fundamentals](https://developers.google.com/web/fundamentals/design-and-ui/responsive/fundamentals/set-the-viewport?hl=en)
* [Google page speed insights](https://developers.google.com/speed/docs/insights/ConfigureViewport)
* [MDN](https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag)
* Description:

With the “content” attribute we can set the “initial-scale=1” value. This value maps css pixels to device independent pixels. There are three categories we talk about when talking about pixels: physical pixels, device independent pixels, and css pixels. Physical pixels are the pixels that are physically built into a device. Some devices have a high pixel densities. This means that they might be 1000 physical pixels in width, but only show 500 pixels to the user. In such a device, for every “device independent pixel” that they show to the user, there are 2 physical pixels making up that 1 “pixel” being shown to the user. As developers, the pixels we use to create our websites are known as “css pixels”. We map our css pixels to device independent pixels with the “initial-scale=1” value. This means that every one of our css pixels will take up one device independent pixel. If we had an “initial-scale=2” setting, each one of our css pixels would take up 2 device independent pixels.

* **Files:** [**041\_mobile**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 221 / take 2**

## 

## meta viewport - picture example

* Including images for mobile devices
* Description:

This is an example of how to include images for mobile devices.

* **Files:** [**041\_mobile**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 222 / take 1**

## 

## Review

* html elements
  + q
  + blockquote
  + cite
  + pre
  + samp
  + code
  + kbd
  + abbr
  + table
  + tr
  + th
  + td
* html element attributes
  + q, blockquote
    - cite
* HTML entities
* Linking images and bookmarks
* box-shadow
* vertical-align
* meta viewport
  + <meta name="viewport" content="width=device-width, initial-scale=1">
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

You’re doing it, baby! You’re getting through the material. This is a large body of knowledge to master. It takes time. But by putting in the time, you acquire the skills. And I know that by now, at this point in the course, you are starting to get it. The code is making sense. You’re starting to get skills. Drop by drop, that bucket gets filled. You are learning this material in the best way possible. You are learning the fundamentals. You are not learning the cheap easy route of using libraries or frameworks like bootstrap or jquery which makes you feel good quickly but then leaves you weak and pathetic, not understanding what is truly occurring with the html and css. You are learning the html and css. This is the bedrock of web programming and it is the absolute best path you can be on for learning how to build web pages. You are in the best course in the world for web programming, and you’re making it through it. Keep up the great work!

* **Files:** [**040\_expanding-skillset**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Files:** [**041\_mobile**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 223 / take 1**

## 

# Graphics

## Section Overview

* Colors
  + Color values
    - Keywords - [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)
    - hexadecimal
    - rgb
    - rgba
    - hsl
    - hsla
  + [Color Resource - Shay Howe](http://learn.shayhowe.com/html-css/getting-to-know-css/)
  + [Color Swatch Picker - Adobe Color](https://color.adobe.com/)
* Images
* Bitmap vs Vector
* [Bitmap](https://en.wikipedia.org/wiki/Bitmap)
  + jpg, jpeg
  + png
  + gif
  + tiff
  + bmp
* [Vector](https://en.wikipedia.org/wiki/Vector_graphics)
  + Svg
  + Software
    - Bitmap - photoshop, [gimp](https://www.gimp.org/)
    - Vector - illustrator, [inkscape](https://inkscape.org/en/)
  + Deliver the correct image size to the client
* [font-awesome](http://fontawesome.io/)
  + [font awesome cheatsheet](http://fontawesome.io/cheatsheet/)
  + Installing font awesome
    - Installing fonts
  + Understanding font awesome’s terms of use
  + Using font awesome
  + here is another alternative you can use:
    - <https://feather.netlify.com/>
* Extracting SVGs from font awesome
  + Analyzing font awesome’s weight
  + Using only what you need
    - Extracting SVGs
* Using SVGs
* Description:

This is an incredibly valuable section. In this section, you are going to learn skills which will set you apart from the majority of web developers. Most web developers do not know how to work with SVGs. By the end of this section, you will understand and be able to use SVGs. You will know the difference between bitmap and vector images. You will also understand color values, how to pick great color combinations, and how to find and use great icons. Finally, by the end of this section, you will be able to utilize font awesome in the best way possible.

* **Files:** [**042\_colors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Files:** [**043\_font-awesome**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 224 / take 1**

## 

## Colors

* Colors
  + Color values
    - Keywords - [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)
    - hexadecimal
    - rgb
    - rgba
    - hsl
    - hsla
  + [Color Resource - Shay Howe](http://learn.shayhowe.com/html-css/getting-to-know-css/)
* Description:

The <color> CSS data type denotes a color in the sRGB color space. A color can be described in any of these ways: (1) using a keyword, (2) using the RGB cubic-coordinate system (via the #-hexadecimal or the rgb() and rgba() functional notations), (3) using the HSL cylindrical-coordinate system (via the hsl() and hsla() functional notations). Though CSS color values are precisely defined, they may appear differently on different output devices. Most output devices are not calibrated. Color rendering may vary a lot. Don’t use color as the only means to conveying specific information, action, or result. Some users have problems distinguishing colors and the conveyed information may not be grasped.

* **Files:** [**042\_colors**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 225 / take 4**

## 

## Picking Color Combinations - Adobe Color ( Kuler )

* Adobe Color
  + Used to be called “adobe kuler”
  + [Color Swatch Picker - Adobe Color](https://color.adobe.com/)
* Description:

Adobe Color, formerly known as Adobe Kuler, is an internet application from Adobe Systems that lets individuals try out, create and save various color schemes, each of which consists of a set of five colors.

* **Video: 226 / take 2**

## 

## Image Types - Raster/Bitmap & Vector

* [Raster](https://en.wikipedia.org/wiki/Raster_graphics) / [Bitmap](https://en.wikipedia.org/wiki/Bitmap)
  + [***“Painting with pixels”***](http://vector-conversions.com/vectorizing/raster_vs_vector.html)
  + Made up of pixels of different colors
    - Pixel = picture element
    - Resolution - number of pixels on a screen
      * 1920 x 1080 = 2,073,600
    - Color Depth - number of bits per pixel
  + File formats
    - jpg, jpeg
    - png
    - gif
    - tiff
    - bmp
  + Good for
    - Real life photographs
  + Software
    - Bitmap - adobe photoshop, [gimp](https://www.gimp.org/)
* [Vector](https://en.wikipedia.org/wiki/Vector_graphics)
  + [***“Drawing with vectors”***](http://vector-conversions.com/vectorizing/raster_vs_vector.html)
  + Mathematical formulas
  + File formats
    - SVG
  + Good for
    - Graphics
    - No color gradations
  + Software
    - Vector - adobe illustrator, [inkscape](https://inkscape.org/en/)
* Deliver the correct image size to the client
* Description:

In computer graphics, a **bitmap** gives a way to store an image in which each pixel is some color. A bitmap is a type of memory organization or image file format used to store digital images. The term bitmap comes from the computer programming terminology, meaning just a map of bits. A bitmap file format is BMP. JPEG, TIFF, PNG, and GIF are also referred to as bitmaps though they use compressed formats internally. **Vector** graphics use polygons to represent images in computer graphics. Vector graphics are based on vectors, which lead through locations called control points or nodes. Each of these points has a definite position on the x and y axes of the work plane and determines the direction of the path; further, each path may be assigned various attributes, including such values as stroke color, shape, curve, thickness, and fill.

* **Video: 227 / take 1**

## 

## Font Awesome

* Overview
* Using
* Description:

Font Awesome is a font and icon toolkit. Font Awesome allows us to put icons on our webpages. Font Awesome has a 20% market share among websites which use third-party Font Scripts, ranking it on a second place after Google Fonts. Here is another alternative you can use: <https://feather.netlify.com/>

* **Files:** [**043\_font-awesome**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 228 / take 1**

## 

## Font Awesome - Performance

* Instructions
* The performance cost of font awesome
* Description:

When you use font awesome, you are asking the user to download 92.7 KB of data, at least that was the size at the time of this recording. The average page size, at the time of this recording, is somewhere between 2,200KB and 2,400KB. Do you really want font awesome to be 1 / 22 or 1 / 24 of the data you transfer to your client? And why are you sending them all of the icons when you are only using some of them? In future videos, we will learn how to reduce the size of what we’re sending the user by only sending them the icons being used. Here is another alternative you can use: <https://feather.netlify.com/>

* **Files:** [**043\_font-awesome**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 229 / take 1**

## 

## Font Awesome - Instructions on Use Review

* License
* Instructions
* Using font awesome
* Description:

This is a review, with more detail, of how to use font awesome. Explicit instructions for use will be covered. The license of font awesome will also be covered. Here is another alternative you can use: <https://feather.netlify.com/>

* **Files:** [**043\_font-awesome**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 230 / take 1**

## 

## Font Awesome to SVG - Adobe Illustrator

* Adobe Illustrator
* Understanding Fonts
* Extracting an SVG from Font Awesome
* Illustrator - step by step
* About SVGs
  + CSS → SVG
    - border → stroke
    - background-color → fill
* Description:

This is how you extract an SVG from font awesome using Adobe Illustrator. How to install a font on your computer is also covered. You will need to install the font awesome font on your computer. The different font formats are explained: TTF, OTF, WOFF, WOFF2.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 231 / take 1**

## 

## Understanding SVG - Basic SVG Shapes

* Different types of shapes
  + Rect
  + Circle
  + Ellipse
  + Line
* Different properties
  + Stroke
  + Fill
* Description:

There are several basic shapes used for most SVG drawing. The purpose of these shapes is reflected in the name of the svg element. The svg elements also have attributes that determine their characteristics (position, size, color).

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 232 / take 1**

## 

## SVG Terminology

### <svg>

* + is an html element
* [MDN svg tutorial](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial)
* [W3C svg](https://www.w3.org/wiki/HTML/Elements/svg)
* Description:

Scalable Vector Graphics (SVG) is an XML-based markup language for describing two-dimensional vector graphics. SVG is essentially to graphics what HTML is to text.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 233 / take 4**

## 

## SVG Path Element

* Path element
* Attributes
  + d
    - “draw”
    - takes a series of commands and parameters
      * M
        + move to a point
      * L
        + draw a line to a point
      * l
        + draw a line from where you're at to a point
      * V
        + draw a vertical line to this y-coordinate
      * v
        + draw a vertical line from where you're at this distance
      * H
        + draw a horizontal line to this x-coordinate
      * h
        + draw a vertical line from where you're at this distance
      * case
        + uppercase - absolute
        + lowercase - relative
* Description:

The <path> element is the most powerful element in the SVG library of basic shapes. You can use it to create lines, curves, arcs and more. Paths create complex shapes by combining multiple straight lines or curved lines. Complex shapes composed only of straight lines can be created as polylines. While polylines and paths can create similar-looking shapes, polylines require a lot of small straight lines to simulate curves and don't scale well to larger sizes. A good understanding of paths is important when drawing SVGs. While creating complex paths using an XML editor or text editor is not recommended (instead use a graphics program), understanding how they work will allow you to identify and repair display issues in SVGs.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 234 / take 2**

## 

## SVG Path - More Examples

* path
* Description:

The "Close Path" command is called with Z. This command draws a straight line from the current position back to the first point of the path. It is often placed at the end of a path node, although not always. There is no difference between the uppercase and lowercase command.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 235 / take 2**

## 

## SVG Stroke & Fill

* Set SVG colors
  + stroke
  + Fill
* [MDN stroke & fill](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Fills_and_Strokes)
* Description:

Basic SVG coloring can be done by setting two attributes on the node: fill and stroke. Fill sets the color inside the object and stroke sets the color of the line drawn around the object. You can use the same css color naming schemes that you use in HTML, whether that's color names (that is red), rgb values (that is rgb(255,0,0)), hex values, rgba values, etc.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 236 / take 1**

## 

## SVG - Notes & Resources

* [MDN SVG Introduction](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Introduction)
* [MDN Getting Started](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Getting_Started)
* [MDN The Grid & Positions](https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Positions)
* [J Watt’s SVG Authoring Guidelines](https://jwatt.org/svg/authoring/)
* Description:

These notes summarize and highlight the essentials of what you need to know about SVGs. Links to the best articles covering SVGs are provided.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 237 / take 5**

## 

## SVG - Viewbox

* Preview
* Viewbox
  + Allows zooming in and out, and shifting, of an svg
* Description:

The viewBox attribute allows you to specify that a given set of graphics stretch to fit a particular container element. The value of the viewBox attribute is a list of four numbers min-x, min-y, width and height, separated by whitespace and/or a comma, which specify a rectangle in user space which should be mapped to the bounds of the viewport established by the given element, taking into account attribute.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 238 / take 4**

## 

## Styling SVGs with CSS

* svg is an inline element
* svg & css
* Description:

How to style SVGs with CSS - Being able to style SVGs with CSS allows us to apply what we have learned about CSS to styling SVGs. Cross-browser, cross-platform issues are covered.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 239 / take 3**

## 

## SVG - Symbol & Use Elements

* symbol
* use
* Description:

The SVG symbol and use elements allow us to modularize our code. The symbol element is used to define graphical template objects which can be instantiated by a <use> element. The use of symbol elements for graphics that are used multiple times in the same document adds structure and semantics.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 240 / take 3**

## 

## Symbol, Use, & CSS

* Symbol
* Use
* Css
* Html elements

### Fieldset

### Legend

* Description:

How to style SVG’s with CSS when the SVGs are using symbol and use.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 241 / take 7**

## 

## Symbol with Multiple Paths

* Including multiple paths in one symbol
* Styling the different paths differently
* Description:

This is how you can add multiple paths into one symbol, and then style those paths differently.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 242 / take 2**

## 

## Leave Room For Strokes

* Make sure your svg doesn’t fill the artboard
* Description:

When you create your svg, make sure the svg doesn’t fill the artboard. If your svg fills the artboard, then you will not have room to increase the svg’s stroke without the svg overflowing the container.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 243 / take 1**

## 

## Stroke %

* Heart example
* Creating a fine stroke
* Description:

We can create a fine stroke on an SVG by using a percentage unit of measurement.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 244\_01 / take 1**

## 

## symbol & use vs img

* Symbol
* Use
* Css
* img
* Description:

What is the best way to put SVG into your file? You can use symbol and use, or you can use an img tag. If you use an img tag, you cannot change the stroke and fill of the svg element with css. So your styling options are limited with the img tag, however, the img tag is very easy to use and is also able to be cached.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 244\_02 / take 2**

## 

## Review

* Colors
  + Color values
    - Keywords - [MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)
    - hexadecimal
    - rgb
    - rgba
    - hsl
    - hsla
  + [Color Resource - Shay Howe](http://learn.shayhowe.com/html-css/getting-to-know-css/)
  + [Color Swatch Picker - Adobe Color](https://color.adobe.com/)
* Images
* Bitmap vs Vector
* [Bitmap](https://en.wikipedia.org/wiki/Bitmap)
  + jpg, jpeg
  + png
  + gif
  + tiff
  + bmp
* [Vector](https://en.wikipedia.org/wiki/Vector_graphics)
  + Svg
  + Software
    - Bitmap - photoshop, [gimp](https://www.gimp.org/)
    - Vector - illustrator, [inkscape](https://inkscape.org/en/)
  + Deliver the correct image size to the client
* [font-awesome](http://fontawesome.io/)
  + [font awesome cheatsheet](http://fontawesome.io/cheatsheet/)
  + Installing font awesome
    - Installing fonts
  + Understanding font awesome’s terms of use
  + Using font awesome
  + Here is another alternative you can use: <https://feather.netlify.com/>
* Extracting SVGs from font awesome
  + Analyzing font awesome’s weight
  + Using only what you need
    - Extracting SVGs
* Understanding SVG’s
* Using SVGs
  + Symbol
  + Use
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

This has been an incredibly valuable section. You learned skills which will set you apart from the majority of web developers. Most web developers do not know how to work with SVGs. You now understand enough to be able to use SVGs. You also know the difference between bitmap and vector images. You also understand color values, how to pick great color combinations, and how to find and use great icons. Finally, you are now also able to utilize font awesome in the best way possible.

* **Files:** [**044\_svg**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 245 / take 2**

# Full-Page Background

## Section Overview

* Fixed header
  + containers at each side
    - Justify-content: space-between;
* SVGs
  + make
  + use
* Logo & Name
  + psychological considerations
    - conscious & unconscious associations
* Dev Tools
  + styling with dev tools
* Mobile-first responsive
  + refactor
* Mobile menu
* Description:

We are going to practice the skills we have learned. We will continue to build on our full page background document. We will add a header. We will add SVGs. We will create a name for our company and choose a log. We will use dev tools to help us style our page. We will make our page mobile-first responsive. We will create a menu for our mobile page. This section will teach you new skills while continuing to reinforce everything you have been learning.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 246 / take 3**

## 

## Fixed Header

* Hints:
  + position: fixed;
  + top: 0;
  + left: 0;
  + background-color: rgba(0, 0, 0, 0.6);
  + height: 2.8rem;
  + width: 100%;
* Description:

In this hands-on exercise, add a header to the top of the page that always remains at the top of the page.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 247 / take 2**

## 

## Fixed Header - Solution

* Hints:
  + header
  + position: fixed;
  + top: 0;
  + left: 0;
  + background-color: rgba(0, 0, 0, 0.6);
  + height: 2.8rem;
  + width: 100%;
* Description:

This is the solution to adding a header to the top of the page that always remains at the top of the page.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 248 / take 1**

## 

## Header Containers

* Hints:
  + display: flex;
  + justify-content: space-between;
* Description:

In this challenge, add two containers to hold content in the header. These containers should be on the left and right sides of the header.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 249 / take 1**

## 

## Header Containers - Solution

* Hints:
  + display: flex;
  + justify-content: space-between;
* Description:

This is the solution to adding two containers to hold content in the header. These containers are on the left and right sides of the header.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 250 / take 1**

## 

## Make SVG’S

* Hints
  + fa-gears / fa-leaf / fa-universal-access
  + fa-home
  + fa-briefcase
  + fa-pencil
  + fa-gear
  + <http://fontawesome.io/cheatsheet/>
* Description:

In this hands-on exercise, make the SVG’s which you will use on your site.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 251 / take 1**

## 

## Make SVG’S - Solution

* Hints
  + fa-gears / fa-leaf / fa-universal-access
  + fa-home
  + fa-briefcase
  + fa-pencil
  + fa-gear
* Description:

This is how you make the SVG’s which you will use on your site.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 252 / take 1**

## 

## Insert SVGs

* symbol
  + style="display: none;"
* use
* Microsoft browser compatible without JS polyfill
* Description:

For this hands-on exercise, insert your SVGs into your header. You will want to do this so that you can style your SVGs. This means you cannot use the img tag. You will need to use the symbol and use tags. Also, you will want your SVGs to be shown in Microsoft browsers. This means you must place your SVGs <symbol> definitions in the HTML page.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 253 / take 1**

## 

## Insert SVGs - Solution

* Important questions to ask yourself
  + **(1) Do I want to style my SVG with CSS**
    - Yes - use <symbol> & <use>
  + **(2) Do I want to make Microsoft compatible without JS**
    - Yes - put <symbol> in HTML page
* symbol
  + style="display: none;"
* use
* Microsoft browser compatible without JS polyfill
* Description:

For this hands-on exercise, insert your SVGs into your header. You will want to do this so that you can style your SVGs. This means you cannot use the img tag. You will need to use the symbol and use tags. Also, you will want your SVGs to be shown in Microsoft browsers. This means you must place your SVGs <symbol> definitions in the HTML page.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 254 / take 1**

## 

## Style SVGs - Solution

* Description:

Using CSS to style SVGs.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 255 / take 4**

## 

## Dev Tools Styling & Company Naming - Solution

* Description:

We can use dev tools to help us choose colors. When choosing the name for a company, or when choosing a logo, it’s important to think about the psychological associations which will occur both consciously and unconsciously.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 256 / take 1**

## 

## Adding Links & Style - Solution

* Touch targets
* Description:

Adding links and styles to our header; making sure our touch targets are nicely sized.

* **Files:** [**045\_full-page-background\_02 / 06\_header-style**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 257 / take 2**

## 

## Responsive, Mobile-First

* Description:

The next hands-on challenge is to refactor our code so that it uses media-queries to make the document mobile-first responsive.

* **Files:** [**045\_full-page-background\_02 / 07\_mobile-first**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 258 / take 1**

## 

## Responsive, Mobile-First - Solution

* Description:

The next hands-on challenge is to refactor our code so that it uses media-queries to make the document mobile-first responsive.

* **Files:** [**045\_full-page-background\_02 / 07\_mobile-first**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 259 / take 4**

## 

## Mobile Menu

* Hints:
  + footer
    - nav
      * anchor
        + display: block;
  + Background colors:
    - deepskyblue
    - hover - royalblue
  + Border colors:
    - border-top: 1px solid rgba(255,255,255,0.3);
    - border-bottom: 1px solid rgba(0,0,0,0.1);
    - Last - border-bottom: none;
* Description:

Build the menu for the mobile version. Display none the mobile menu on desktop.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 260 / take 4**

## 

## Mobile Menu - Solution

* Hints:
  + footer
    - nav
      * anchor
  + Background colors:
    - deepskyblue
    - hover - royalblue
  + Border colors:
    - border-top: 1px solid rgba(255,255,255,0.3);
    - border-bottom: 1px solid rgba(0,0,0,0.1);
    - Last - border-bottom: none;
* Description:

This is how I built the menu for the mobile version. Display none the mobile menu on desktop.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 261 / take 6**

## 

## Review

* Fixed header
  + containers at each side
    - Justify-content: space-between;
* SVGs
  + make
  + use
* Logo & Name
  + psychological considerations
    - conscious & unconscious associations
* Dev Tools
  + styling with dev tools
* Mobile-first responsive
  + refactor
* Mobile menu
* [Layout essentials](https://docs.google.com/document/d/1IjV1IlnEXKPu5IFdWtr9Fubl_RZEVJdmPgaBQOjomU0/edit?usp=sharing)
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

In this section, we continued to build our full page background document. We added a header. We added SVGs. We created a name for our company and chose a logo. We used dev tools to help us style our page. We made our page mobile-first responsive. We created a menu for our mobile page. This section showed you new skills while continuing to reinforce everything you have been learning.

* **Files:** [**045\_full-page-background\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 262 / take 1**

# 

# Favicon

## 

## Section Overview

* favicon
* Description:

A favicon (short for favorite icon), which is also known as a shortcut icon, website icon, tab icon, URL icon, or bookmark icon, is a small icon associated with a particular website or web page. A web designer can create such an icon and upload it to a website (or web page) by several means, and graphical web browsers will then make use of it. Browsers that provide favicon support typically display a page's favicon in the browser's address bar (sometimes in the history as well) and next to the page's name in a list of bookmarks. Browsers that support a tabbed document interface typically show a page's favicon next to the page's title on the tab, and site-specific browsers use the favicon as a desktop icon. Favicons can also be used to have a textless favourite site, saving space.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 263 / take 1**

## 

## Create An Image

* Resolution greater than
  + **600 x 600** pixels
* Use an icon from font-awesome
  + Make it bigger in illustrator
  + Save it as a PNG in illustrator
    - file / export
* Description:

To create a favicon, the first thing we need to do is create an image. The image needs to be at least **600 x 600** pixels. We will use adobe illustrator to create the image which will serve as the base image for our favicon.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 264 / take 5**

## 

## Generate Favicon

* <http://realfavicongenerator.net/>
* Description:

There are different sites which you can use to generate a favicon. I like <http://realfavicongenerator.net/> though the quality of this site could change with time. Always check the results to make sure you’re happy with them. If needed, find another site or tool which generates favicons in a better way.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 265 / take 2**

## 

## Generate Favicon - Subfolder

* <http://realfavicongenerator.net/>
* Description:

This is how to generate a favicon at <http://realfavicongenerator.net/> and place your results in a subfolder.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 266 / take 1**

## 

## Generate Favicon - Clears Cache

* <http://realfavicongenerator.net/>
* Description:

Computers will often keep files in cache. You can tell a computer (browser or mobile app) to pull down a fresh icon file by changing the link to that file - this tells the computer that there is a new file; that you’re asking for something different than what you had before.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 267 / take 1**

## 

## Generate Favicon - Compression

* <http://realfavicongenerator.net/>
* Description:

Compression allows us to remove data from something. Often compression degrades the quality of that which is compressed. There is a fine art in achieving the right balance between quality and compression.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 268 / take 1**

## 

## Favicon Essentials

* favicon
* Description:

These are the notes I have collected about favicons from various sources.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 269 / take 1**

## 

## Full Page Background - Favicon

* favicon
* Description:

Add a favicon to our full page background.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 270 / take 1**

## 

## Review

* Favicon
* Generate an image
  + **600 x 600**
* <http://realfavicongenerator.net/>
* Cache
  + Clearing cache
* Compression
  + Performance
* Staying current
  + Things change!
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

In this section, we continued to build our full page background document. We added a header. We added SVGs. We created a name for our company and chose a logo. We used dev tools to help us style our page. We made our page mobile-first responsive. We created a menu for our mobile page. This section showed you new skills while continuing to reinforce everything you have been learning.

* **Files:** [**046\_favicon**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 271 / take 1**

# 

# Flexbox Practicum - Part 1

## 

## Section Overview

* Hands on exercises
  + Counter
  + Tab menu
  + Header menu
* Credit
  + <http://html5weekly.com/>
  + <http://www.flexboxpatterns.com/home>
* Description:

These hands-on exercises will help you learn how to build many common components used today. They will also reinforce your growing skillset.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 272 / take 1**

## 

## Counter

* Description:

The “counter” hands on exercise is introduced.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 273 / take 1**

## 

## Counter - Solution

* Description:

This is the solution to the “counter” hands-on exercise.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 274 / take 1**

## 

## Tab Menu

* Description:

The “tab menu” hands on exercise is introduced.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 275 / take 1**

## 

## Tab Menu - Solution

* button
* Description:

This is the solution to the “tab menu” hands-on exercise. The button element is further explained.

### FREE PREVIEW this video

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 276 / take 4**

## 

## Header Menu

* Description:

The “header” hands on exercise is introduced.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 277 / take 3**

## 

## Header Menu - Solution

* Description:

This is the solution to the “header” hands-on exercise.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 278 / take 14**

## 

## Review

* Hands on exercises
  + Counter
  + Tab menu
  + Header menu
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

The flexbox practicum part 1 has given us three hands-on exercises to continue building our skills with flexbox: counter, tab menu, header menu. Each of these examples showed us how to build commonly used designs using flexbox.

* **Files:** [**047\_flexbox-practicum\_01**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 279 / take 1**

# 

# Transitions & Animations

## Section Overview

* **Transitions**: do it between two points.
* **Animations**: do it between two or more points.
* **Transition properties**
  + transition-property: all;
  + transition-duration: 500ms;
  + transition-timing-function: ease-in-out;
  + transition-delay: 1s;
* **Animation properties**
  + animation-name: none
  + animation-duration: 0s
  + animation-iteration-count: 1
  + animation-direction: normal
  + animation-timing-function: ease
  + animation-delay: 0s
  + animation-fill-mode: none
  + animation-play-state: running
* Description:

Transitions and animations do the same thing. Transitions: do it between two points. Animations: do it between two or more points. The result of a transition, or animation, are the same: a gradual change between two (or more) points. Animations are good for changes which occur between more than two points. In addition, animations give you more options on what is being changed.

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 280 / take 1**

## 

## Transitions

* **Transitions**: do it between two points.
* **Transition properties**
  + [**transition-property**](https://developer.mozilla.org/en-US/docs/Web/CSS/transition-property): all;
  + [**transition-duration**](https://developer.mozilla.org/en-US/docs/Web/CSS/transition-duration): 500ms;
  + [**transition-timing-function**](https://developer.mozilla.org/en-US/docs/Web/CSS/transition-timing-function): ease-in-out;
  + [**transition-delay**](https://developer.mozilla.org/en-US/docs/Web/CSS/transition-delay): 1s;
* Description:

CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized. CSS transitions let you decide which properties to animate (by listing them explicitly), when the animation will start (by setting a delay), how long the transition will last (by setting a duration), and how the transition will run (by defining a timing function, e.g. linearly or quick at the beginning, slow at the end).

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 281 / take 1**

## 

## Transition Examples

* Some examples showing how to use CSS transitions
* Description:

We will look at two examples which show us how to use transitions.

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 282 / take 1**

## 

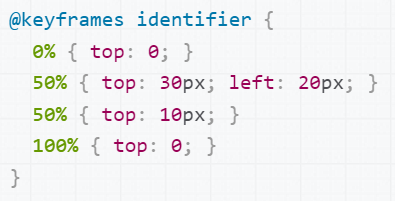
## Animations

* **Animations**: do it between two or more points.
* **Animation properties**
  + [**animation-name**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-name): myCustomAnima;



you can also use: from, to

* + [**animation-duration**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-duration): 500ms
  + [**animation-timing-function**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-timing-function): ease
  + [**animation-delay**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-delay): 300ms
  + [**animation-iteration-count**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-iteration-count): infinite
  + [**animation-direction**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-direction): alternate
  + [**animation-fill-mode**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-fill-mode): forwards
  + [**animation-play-state**](https://developer.mozilla.org/en-US/docs/Web/CSS/animation-play-state): running
* @keyframes
  + The @keyframes CSS at-rule lets authors control the intermediate steps in a CSS animation sequence by establishing keyframes (or waypoints) along the animation sequence that must be reached by certain points during the animation. This gives you more specific control over the intermediate steps of the animation sequence than you'd get when letting the browser handle everything automatically.



* Description:

CSS animations make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components: an **animation-name** and the associated **@keyframes** that indicate the states of the animated CSS properties.

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 283 / take 1**

## 

## Animation Examples & Hands-On Exercise

* Animation examples
* A hands-on exercise
* Description:

We start out looking at examples of animations by using the following CSS properties: animation-name, animation-duration, animation-iteration-count, animation-direction, animation-timing-function

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 284 / take 4**

## 

## Hands-On Exercise Solution & Another Hands-On Exercise

* Solution to hands-on exercise
* An additional hands-on exercise
* Description:

The solution to the hands on exercise is to animate the body’s css “justify-content” property from flex-start to flex-end.

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 285 / take 2**

## 

## Hands-On Exercise Solution

* Solution to hands on exercise
* Description:

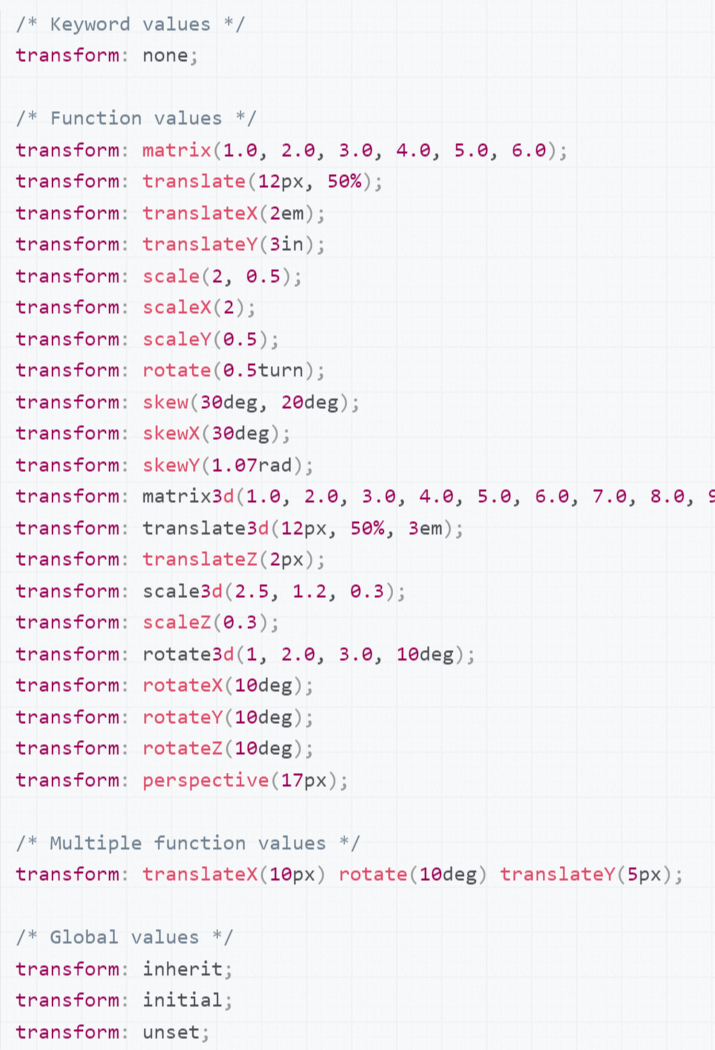
The solution to this hands on exercise involved animating the font-size.

* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 286 / take 1**

## 

## Transform & Animations

* Transform
  + the [**transform**](https://developer.mozilla.org/en-US/docs/Web/CSS/transform) property takes a [**transform function**](https://developer.mozilla.org/en-US/docs/Web/CSS/transform-function)



* Description:
  + CSS transforms change the shape and position of content without disrupting normal document flow. CSS transforms are implemented using a set of CSS properties that let you apply transformations to HTML elements. These transformations include rotation, skewing, scaling, and translation both in the plane and in the 3D space.
* **Files:** [**048\_transitions-animations**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 287 / take 1**

## Review

* Transitions & Animations
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
  + transition-property: all;
  + transition-duration: 500ms;
  + transition-timing-function: ease-in-out;
  + transition-delay: 1s;
  + animation-name: none
  + animation-duration: 0s
  + animation-iteration-count: 1
  + animation-direction: normal
  + animation-timing-function: ease
  + animation-delay: 0s
  + animation-fill-mode: none
  + animation-play-state: running
  + transform
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* Description:

Transitions and animations do the same thing. Transitions: do it between two points. Animations: do it between two or more points. The result of a transition, or animation, are the same: a gradual change between two (or more) points. Animations are good for changes which occur between more than two points. In addition, animations give you more options on what is being changed.

* **Files:**
* **Video: 288 / take 1**

# 

# Flexbox Practicum - Part 2

## Section Overview

* Hands on exercises
  + Footer - Saving
  + Sidebar Menu
  + Welcome Page
  + Dial Icon
  + Feature List
  + Card
  + Card Group
* Credit
  + <http://html5weekly.com/>
  + <http://www.flexboxpatterns.com/home>
* Description:

These hands-on exercises will help you learn how to build many common components used today. They will also reinforce your growing skillset.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 289 / take 1**

## 

## Footer Saving - Solution

* Description:

This is the solution to the “footer” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 290 / take 2**

## 

## Sidebar Menu - Solution

* Description:

This is the solution to the “side bar menu” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 291 / take 1**

## 

## Welcome Page - Solution

* Description:

This is the solution to the “welcome page” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 292 / take 1**

## 

## Dial Icon - Solution

* Description:

This is the solution to the “dial icon” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 293 / take 1**

## 

## Feature List - Solution

* Description:

This is the solution to the “feature list” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 294 / take 1**

## 

## Card - Solution

* Description:

This is the solution to the “card” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 295 / take 1**

## 

## Card Group - Solution

* Description:

This is the solution to the “card group” hands-on exercise.

* **Files:** [**049\_flexbox-practicum\_02**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 296 / take 1**

# 

# You Are A Hero

## Section Overview

* Hands on exercises
  + Flexbox & ATF
* Side-bar fly-out
* gradients
* audio & video
* markdown
* vendor prefixes
* publishing your site
  + buying a domain
    - <https://domains.google/#/>
  + deploying to Google Cloud
    - install appengine sdk for Go
    - install python 2.7.x
    - configure environment PATH variables
    - google cloud developer console
      * create a project
      * get the project ID
    - update the app.yaml file with your project ID
    - deploy to that project
      * **appcfg.py -A <YOUR\_PROJECT\_ID> -V v1 update .**
    - view your project
      * http://<YOUR PROJECT ID>.appspot.com/
      * example
        + <http://temp-145415.appspot.com/>
    - change DNS info at google domains
      * point your domain to your appengine project
* Description:

You are now officially a web hero. With the skills you have acquired, you qualify for web hero status. There are just a few last tools to learn about. In this section, we will practice some of what we have already learned, and also learn some new things like how to create a sidebar menu so that it flies out, how to use gradients, what markdown is and how to use it, understanding vendor prefixes, and publishing your site.

* **Files:** [**050\_hands-on-exercise**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 297 / take 1**

## 

## Above-The-Fold Flexbox Practice

* Hands on exercises
  + Flexbox & ATF
* **FREE PREVIEW this video**
* Description:

In just 20 minutes, we will code out a perfect above-the-fold web page. We will do this using today’s best practices including flexbox.

* **Files:**[**050\_hands-on-exercise**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 298 / take 1**

## 

## Side-Bar Fly-Out

* Side-bar fly-out
* **FREE PREVIEW this video**
* Description:

I am going to show you one of the greatest secrets in web development today - how to create a fly out menu with only HTML and CSS.

* **Files:** [**051\_side-bar**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 299 / take 2**

## 

## Linear-Gradient

* How to apply a gradient to an element
* Description:

The CSS linear-gradient() function creates an <image> which represents a linear gradient of colors. Like any other gradient, a CSS linear gradient is not a CSS <color> but an image with no intrinsic dimensions; that is, it has neither natural or preferred size, nor ratio. Its concrete size will match the size of the element it applies to.

* **Files:** [**052\_linear-gradient**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 300 / take 2**

## 

## Audio & Video

* Audio
* Video
* Description:

The HTML <audio> element is used to embed sound content in documents. Use the HTML <video> element to embed video content in a document.

* **Files:** [**053\_audio-video**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 301 / take 1**

## 

## Markdown

* markdown
* Description:

Markdown is a lightweight markup language with plain text formatting syntax designed so that it can be converted to HTML and many other formats using a tool by the same name. Markdown is often used to format readme files, for writing messages in online discussion forums, and to create rich text using a plain text editor.

* **Files:** [**054\_markdown**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 302 / take 4**

## Vendor Prefixes, AutoPrefixer, Task-Runners Gulp & Grunt

* vendor prefixes
* Description:

Browser vendors sometimes add prefixes to experimental or nonstandard CSS properties, so developers can experiment but changes in browser behavior don't break the code during the standards process. Developers should wait to include the unprefixed property until browser behavior is standardized. Browser vendors try to get rid of vendor prefix for experimental features. They noticed that Web developers were using them on production Web sites, polluting the global space, and making it more difficult for underdogs to perform well (prefixed version are only added for major browsers and the unprefixed version is often forgotten, if not broken). Lately, the trend is to add experimental feature behind user-controlled flags and to work on smaller specifications that reaches stability much quicker.

* **Video: 303 / take 2**

## 

## Publishing Your Site - An Overview

* publishing your site
  + buying a domain
    - <https://domains.google/#/>
  + deploying to Google Cloud
    - install appengine sdk for Go
    - install python 2.7.x
    - configure environment PATH variables
    - google cloud developer console
      * create a project
      * get the project ID
    - update the app.yaml file with your project ID
    - deploy to that project
      * **appcfg.py -A <YOUR\_PROJECT\_ID> -V v1 update .**
    - view your project
      * http://<YOUR PROJECT ID>.appspot.com/
      * example
        + <http://temp-145415.appspot.com/>
    - change DNS info at google domains
      * point your domain to your appengine project
* Description:

A preview of how to purchase a domain and publish a website to it.

* **Video: 304-1 / take 1**

## 

## Publishing Your Site - Buying A Domain

* <https://domains.google/#/>
* Description:

Purchase your domain at https://domains.google/#/

* **Video: 304-2 / take 4**

## Publishing Your Site - Google Cloud

* <https://cloud.google.com/>
  + install google appengine
    - <https://cloud.google.com/appengine/docs/go/download>
  + make sure python is installed VERSION 2.7.x
    - <https://www.python.org/downloads/release/python-2712/>
* Description:

How to publish a website to google cloud. First we must install google’s appengine sdk. This also means we must install Python version 2.7.x. This is all covered in this video.

* **Video: 305 / take 1**

## Publishing Your Site - Google Cloud II

* <https://cloud.google.com/>
  + install google appengine
    - <https://cloud.google.com/appengine/docs/go/download>
  + make sure python is installed VERSION 2.7.x
    - <https://www.python.org/downloads/release/python-2712/>
* <https://console.cloud.google.com>
  + create a project → get a project ID
  + change the APP.YAML file to your project ID
  + appcfg.py -A <YOUR\_PROJECT\_ID> -V v1 update .
* Description:

How to publish a website to google cloud. First we must install google’s appengine sdk. This also means we must install Python version 2.7.x. This is all covered in this video.

* **Files:** [**055\_appengine**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 306 / take 1**

## Review

* Hands on exercises
  + Flexbox & ATF
* Side-bar fly-out
  + label
  + input
    - checkbox
* gradients
  + linear-gradient
* audio & video
  + audio
  + video
* markdown
* vendor prefixes
* Task runners
  + gulp
  + grunt
* publishing your site
  + buying a domain
    - <https://domains.google/#/>
  + deploying to Google Cloud
    - install appengine sdk for Go
    - install python 2.7.x
    - configure environment PATH variables
    - google cloud developer console
      * create a project
      * get the project ID
    - update the app.yaml file with your project ID
    - deploy to that project
      * **appcfg.py -A <YOUR\_PROJECT\_ID> -V v1 update .**
    - view your project
      * http://<YOUR PROJECT ID>.appspot.com/
      * example
        + <http://temp-145415.appspot.com/>
    - change DNS info at google domains
      * point your domain to your appengine project
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
  + input
  + label
  + audio
  + video
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
  + transition-property: all;
  + transition-duration: 500ms;
  + transition-timing-function: ease-in-out;
  + transition-delay: 1s;
  + animation-name: none
  + animation-duration: 0s
  + animation-iteration-count: 1
  + animation-direction: normal
  + animation-timing-function: ease
  + animation-delay: 0s
  + animation-fill-mode: none
  + animation-play-state: running
  + transform
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* CSS functions
  + linear-gradient
* Description:

You are now officially a web hero. With the skills you have acquired, you qualify for web hero status. There are just a few last tools to learn about. In this section, we will practice some of what we have already learned, and also learn some new things like how to create a sidebar menu so that it flies out, how to use gradients, what markdown is and how to use it, understanding vendor prefixes, and publishing your site.

* **Video: 307 / take 1**

# Forms

## Section Overview

* back-end developer
  + (#1) Go
  + (#3) node.js
  + (#3) python
  + (#4) ruby
  + (#5) php
* front-end developer
  + (#1) HTML
  + (#2) CSS
  + (#3) JavaScript
* HTTP Method
  + Get
  + Post
* Description:

We are starting to bridge the territory between front-end developers and back-end developers. In this section, we will look over the wall into the next classroom of back-end development. A front-end developer needs to understand a few technical items to be able to talk with a back-end developer. We will learn these terms and ideas in this section. We will also learn the elements and attributes used for creating a form. Forms allow us to receive data from a user. We use a form to create an area on a webpage that allows the user to enter data. When the form is submitted, the entered information is sent to the server. The server can then process the submitted information.

* **Video: 308 / take 7**

## The Form Element

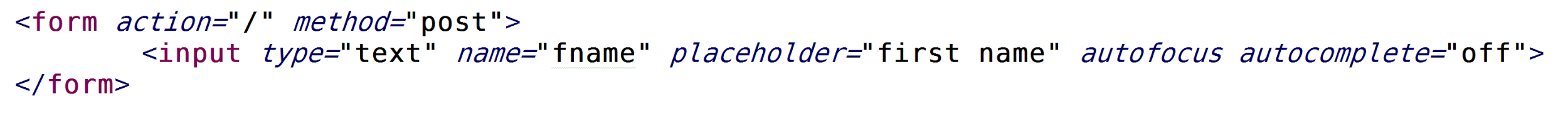
* new HTML **elements**, attributes, values
  + [**form**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/form) 
    - action
      * <URL>
    - method
      * post
      * get
* HTTP Methods
  + Get
  + Post
* IETF
* RFC 7230
* Description:

The HTML <form> element represents a document section that contains interactive controls to submit information to a web server.

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 309 / take 2**

## The Input Element

* new HTML **elements**, attributes, values
  + [**input**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input)
    - type
      * checkbox
      * color
      * date
      * datetime
      * datetime-local
      * email
      * file
      * hidden
      * image
      * month
      * number
      * password
      * radio
      * range
      * reset
      * search
      * submit
      * tel
      * text
      * time
      * url
      * week
    - title
      * *provides the tooltip text*
    - placeholder
      * *the text shown before a user enters anything*
    - name
      * *the name of the variable holding the submitted value*
      * *The name of the control, which is submitted with the form data.*
    - autofocus
    - autocomplete
      * off
      * on
    - maxlength
    - minlength
    - required
    - value
      * *the initial value*



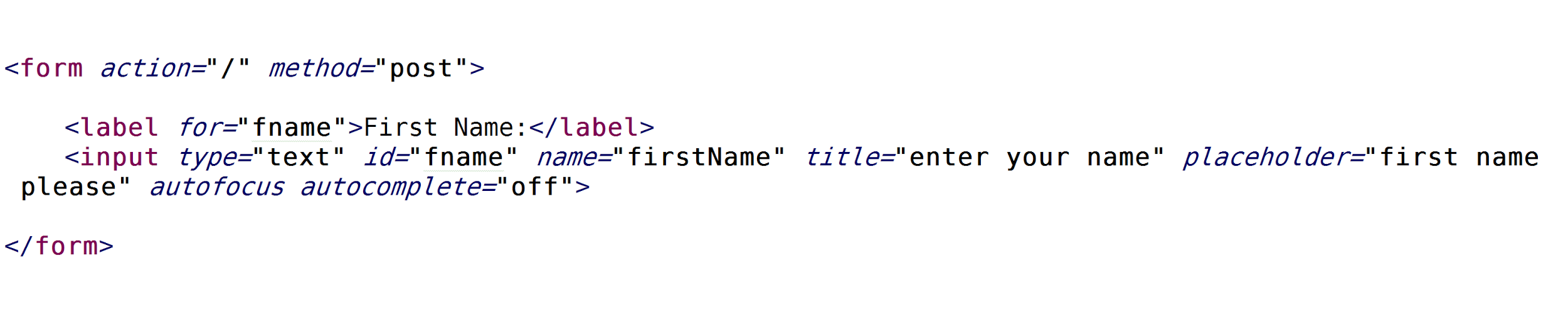
* Description:

The HTML element <input> is used to create interactive controls for web-based forms in order to accept data from the user. How an <input> works varies considerably depending on the value of its type attribute.

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 310 / take 1**

## The Label Element

* new HTML **elements**, attributes, values
  + **label**



* Description:

*The HTML Label Element (<label>) represents a caption for an element. It can be associated with an element either by placing the element inside the <label> element, or by using the* ***for*** *attribute.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 311 / take 1**

## The Input Element - Types

* new HTML **elements**, attributes, values
  + [**input**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input)
    - type
      * checkbox
      * color
      * date
      * datetime
      * datetime-local
      * email
      * file
      * hidden
      * image
      * month
      * number
      * password
      * radio
      * range
      * reset
      * search
      * submit
      * tel
      * text
      * time
      * url
      * week
* Description:

*Here are the most common input element types.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 312 / take 4**

## textarea, select, option, optgroup

* new HTML **elements**, attributes, values
  + textarea
    - rows
  + [**select**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/select)
    - rows
  + optgroup
    - label
  + option
* Description:

*The* ***textarea*** *element allows users to enter text using multiple lines. The* ***select*** *element allows users to choose an option from a drop-down menu. The select element requires options. Each option is created with the* ***option*** *element. We can group options together using the* ***optgroup*** *element.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 313 / take 1**

## fieldset & legend

* review **elements**
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
* new HTML **elements**, attributes, values
  + fieldset
  + legend
* Description:

*The <fieldset> element is used to group several controls and labels <label> within a web form. The <legend> element represents a caption for the content of its parent <fieldset>.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 314 / take 1**

## Form Examples, progress, meter

* review **elements**
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
  + **fieldset**
  + **legend**
* we also learned about these elements
  + progress
  + meter
* Description:

*Some form examples.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 315 / take 1**

## Review

* form **elements**
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
  + **fieldset**
  + **legend**
* html elements
  + progress
  + meter
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
  + input
  + label
  + audio
  + video
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
  + **fieldset**
  + **legend**
  + progress
  + meter
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
  + transition-property: all;
  + transition-duration: 500ms;
  + transition-timing-function: ease-in-out;
  + transition-delay: 1s;
  + animation-name: none
  + animation-duration: 0s
  + animation-iteration-count: 1
  + animation-direction: normal
  + animation-timing-function: ease
  + animation-delay: 0s
  + animation-fill-mode: none
  + animation-play-state: running
  + transform
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* CSS functions
  + linear-gradient
* Description:

*Forms allow us to accept input from a user.*

* **Files:** [**056\_forms**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 316 / take 1**

# Now Go Build It

## Overview of Three Projects

* fresnoradon.com
* learning website
* longbikeride.com homage
* Description:

The best way to learn is to do. In this section, we are going to build three sites. I will show you the sites, then give you an overview of the approach to building each site.

* **Files:** [**058\_contractor-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Files:** [**059\_teaching-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Files:** [**060\_travel-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 317 / take 4**

## Game Plan - Contractor Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**058\_contractor-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 318 / take 1**

## Starting Files - Contractor Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**058\_contractor-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 319 / take 1**

## Game Plan - Teaching Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**059\_teaching-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 320 / take 1**

## Starting Files - Teaching Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**059\_teaching-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 321 / take 1**

## Game Plan - Travel Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**060\_travel-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 322 / take 1**

## Starting Files - Travel Website

* putting together our game plan
* Description:

Before beginning a project, always get a game plan together about how you are going to approach that project.

* **Files:** [**060\_travel-website**](https://github.com/GoesToEleven/html-css-bootcamp)
* **Video: 323 / take 1**

# Farewell - Fare Thee Well

## What’s Next? And THANK YOU FOR JOINING ME!

* Next Learning Steps
  + JavaScript for web developers
    - JSON
    - Manipulate the DOM
  + Go programming language
  + Go programming language - web development
* Good to know
  + [Google webmaster learning](https://www.google.com/webmasters/learn/)
  + [Google’s webmaster Guidelines](https://support.google.com/webmasters/answer/35769)
  + [W3C HTML Validator](https://validator.w3.org/)
  + [W3C CSS Validator](https://jigsaw.w3.org/css-validator/)
  + [MDN - to avoid](https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Obsolete_things_to_avoid)
* Description:

The next steps for you are to (1) use what you have learned and (2) continue learning. There are also a few additional links which will be useful to you: Google’s webmaster stuff, validators, and MDN’s recommendations as to what to avoid.

* **Video: 324 / take 1**

# REFERENCE - what we have learned

* form **elements**
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
  + **fieldset**
  + **legend**
* html elements
  + progress
  + meter
* All HTML tags we have learned so far
  + Html
  + Head
  + Body
  + Meta
  + Title
  + Paragraph
  + Heading
  + Unordered list
  + Ordered list
  + List item
  + Link
  + Image
  + Anchor
  + Paragraph
  + Div
  + Del
  + S
  + Header
  + Nav
  + Main
  + Article
  + Section
  + Aside
  + Footer
  + h1 - h6
  + Figure
  + Figcaption
  + Address
  + Button
  + q
    - cite
  + blockquote
    - cite
  + cite
  + pre
  + samp
  + code
  + kbd
  + Abbr
  + Table
  + Tr
  + Th
  + Td
  + Svg
  + Fieldset
  + legend
  + input
  + label
  + audio
  + video
  + **form**
  + **input**
  + **label**
  + **textarea**
  + **select**
  + **optgroup**
  + **option**
  + **fieldset**
  + **legend**
  + progress
  + meter
* HTML tag attributes
  + Applying two classes to the same element
    - class=”authorize emphasize”
  + **q & blockquote elements**
    - **cite attribute**
  + **meta tag (meta viewport)** 
    - <meta name="viewport" content="width=device-width, initial-scale=1">
  + **link tag media attribute**
    - media type
      * all
      * screen
      * print
      * speech
    - media expression
      * min-width
      * max-width
      * … and there are more …

<link rel="stylesheet” href="mq-900-plus.css" media="(min-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="(max-width: 900px)">

<link rel="stylesheet" href="mq-900-plus.css" media="print">

<link rel="stylesheet" href="mq-900-plus.css" media="screen and (min-width: 900px)">

* HTML Entities
  + &lt;
  + &gt;
  + &nbsp;
* All CSS properties we have learned so far
  + width
  + height
  + background-color
  + color
  + font-size
  + display
    - display: inline
    - display: block
    - display: inline-block
    - display: none
  + padding
  + border
  + margin
    - margin: 20px auto;
    - margin: 0 auto;
      * TRBL
      * TB RL
      * T R B L
  + box-sizing: border-box
  + border-radius
  + background-color
  + background-image
  + background-position
  + background-size
  + background-repeat
  + background-origin
  + background-attachment
  + text-align
    - text-align: center;
  + cursor
    - cursor: pointer;
  + Font
    - font-family
    - font-size
    - font-weight
    - font-variant
    - line-height
    - font-style
  + font-family
    - <family-name>
    - <generic-name>
      * serif
      * sans-serif
      * monspace
      * cursive
      * fantasy
  + font-size
    - xx-small, x-small, small, medium, large, x-large, xx-large
      * User’s default font size is medium
      * **relative to default font size ( root font size )**
    - larger, smaller
      * **relative to parent element’s font size**
    - <length>
      * [MDN length units](https://developer.mozilla.org/en-US/docs/Web/CSS/length#Units)
      * px
      * em
        + **relative to parent element’s font size**
      * rem
        + **relative to default font size ( root font size )**
      * vh
        + **1/100th of the height of the viewport.**
      * vw
        + **1/100th of the width of the viewport.**
    - <percentage>
      * A percentage of the parent element’s font size
      * **relative to parent element’s font size**
  + font-weight
    - normal
      * same as 400
    - bold
      * same as 700
    - lighter
    - bolder
    - 100, 200, 300, 400, 500, 600, 700, 800, 900
  + font-variant
    - normal
    - small-caps
    - titling-caps
    - unicase
  + text-transform
    - uppercase
    - lowercase
    - capitalize
  + line-height
    - normal
    - <number>
    - <length>
    - <percentage>
  + font-style
    - normal
    - italic
    - oblique
  + letter-spacing
  + word-spacing
  + text-align
  + text-shadow
  + text-decoration
  + text-indent
  + max-width
  + min-width
  + list-style
  + Vertical-align
  + FLEXBOX - Container properties
    - Flex-direction
    - Flex-wrap
    - Justify-content (main axis)
    - Align-items (cross axis)
    - Align-content (cross axis)
  + FLEXBOX - Item properties
    - Align-self
    - Order
    - Flex-grow
    - Flex-shrink
    - Flex-basis
    - Flex
  + Position
    - Fixed
    - Relative
    - Absolute
    - Static
  + Top
  + Right
  + Bottom
  + Left
  + float
    - left
    - right
    - none
  + clear
    - left
    - right
    - both
  + !important
  + transition-property: all;
  + transition-duration: 500ms;
  + transition-timing-function: ease-in-out;
  + transition-delay: 1s;
  + animation-name: none
  + animation-duration: 0s
  + animation-iteration-count: 1
  + animation-direction: normal
  + animation-timing-function: ease
  + animation-delay: 0s
  + animation-fill-mode: none
  + animation-play-state: running
  + transform
* All CSS selectors we have learned so far
  + **general**
    - element
    - class
    - id
    - \*
    - attribute
  + **link**
    - link
    - visited
    - hover
    - active
  + **form** 
    - focus
  + **text**
    - first-letter
    - first-line
  + **nested**
    - div p
      * All p tags beneath a div
    - div > p
      * All p tags **immediately** beneath a div
  + **following siblings**
    - div ~ p
      * All p tags that are a sibling following a div
    - div + p
      * All p tags that are an **immediate** sibling following a div
  + **compound**
    - examples:
      * ul#summer-drinks li
      * ul#summer-drinks li.favorite
      * html body ul#summer-drinks li.favorite
* CSS functions
  + linear-gradient