

Course Levels



Level 0 Setup & Fundamentals

Level 1 HTML Basics

Level 2 Must-Use HTML Tags

Level 3 Browser Tools

Level 4 HTML and Project Structure

Level 5 List, Tables & Forms

Level Bonus Github Pages & CodeSpace

Level 0

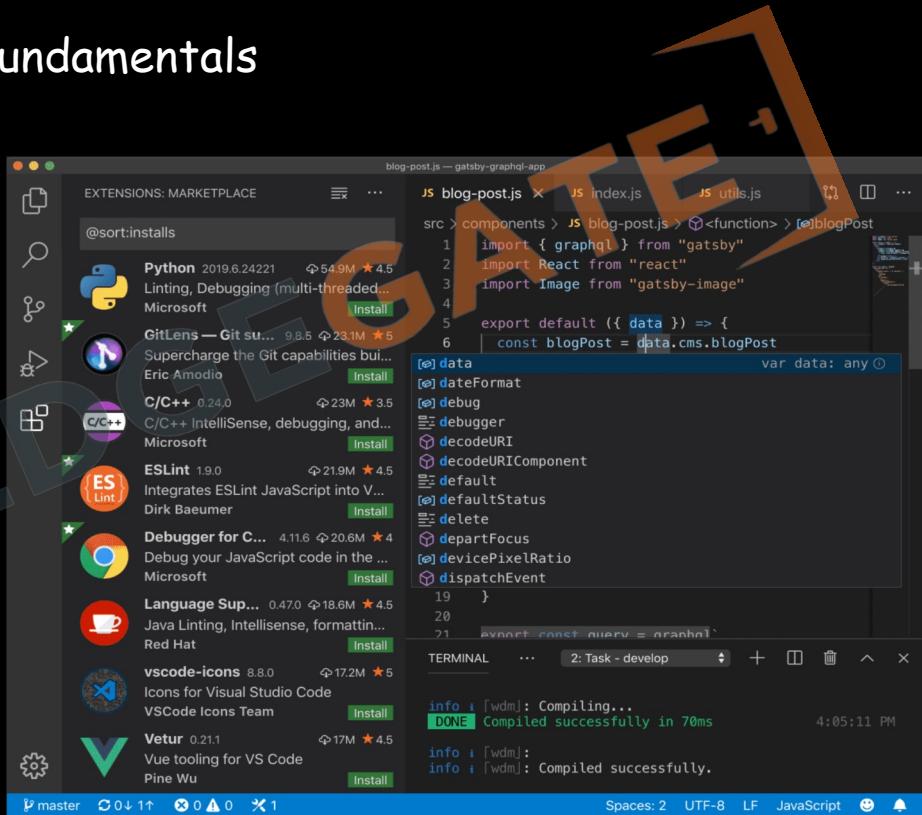
Setup & Fundamentals

1. IDE or Code Editor
 1. What is IDE
 2. Need of IDE
 3. IDE Selection
 4. Installation and Setup
 5. VsCode Extensions
2. Website Components and Fundamentals
 1. Client Side vs Server Side
 2. FrontEnd / BackEnd / FullStack
 3. Role of Browser
 4. HTML
 5. CSS
 6. JS

Level 0

Setup & Fundamentals

1. IDE OR Code Editor



1.1 What is IDE

1. IDE stands for Integrated Development Environment.
2. Software suite that consolidates basic tools required for software development.
3. Central hub for coding, finding problems, and testing.
4. Designed to improve developer efficiency.



1.2 Need of IDE

1. Streamlines development.
2. Increases productivity.
3. Simplifies complex tasks.
4. Offers a unified workspace.
5. IDE Features
 1. Code Autocomplete
 2. Syntax Highlighting
 3. Version Control
 4. Error Checking



```
 MainActivity.kt
@Composable
fun MessageCard(msg: Message) {
    Row(modifier = Modifier.padding(all = 8.dp)) {
        Image(
            painter = painterResource(R.drawable.android_studio_logo),
            contentDescription = "Profile Picture",
            modifier = Modifier
                .size(45.dp)
        )
        Spacer(modifier = Modifier.width(8.dp))
        Column (Modifier
            .background(color = Color.White)) {
            Text(text = msg.author, color = Color.Black)
            Spacer(modifier = Modifier.height(1.dp))
            Text(text = msg.body, color = Color.Black)
        }
    }
}
```



1.3 IDE Selection

1. Sublime Text
2. Atom
3. VS Code
4. Github CodeSpaces



The screenshot shows the Visual Studio Code interface. On the left, the Extensions Marketplace is open, displaying various extensions like Python, GitLens, C/C++, ESLint, Debugger for C#, Language Support, vscode-icons, and Vetur. On the right, a code editor window is open with files blog-post.js, index.js, and utils.js. The code editor shows a snippet of JavaScript code related to GraphQL and Gatsby. At the bottom, the terminal window shows the output of a compilation process.

```
blog-post.js — gatsby-graphql-app
JS blog-post.js x JS index.js JS utils.js
src > components > JS blog-post.js > <function> > blogPost
1 import { graphql } from "gatsby"
2 import React from "react"
3 import Image from "gatsby-image"
4
5 export default ({ data }) => {
6   const blogPost = data.cms.blogPost
7
8   return (
9     <div>
10       <h1>{ blogPost.title }</h1>
11       <p>{ blogPost.description }</p>
12       <img alt={ blogPost.image.alt } data-image={ blogPost.image } />
13     </div>
14   )
15 }
16
17 const query = graphql`query {
18   site {
19     siteMetadata {
20       title
21       description
22       author
23     }
24   }
25   cms {
26     blogPost {
27       id
28       title
29       description
30       image {
31         alt
32         file {
33           childImageSharp {
34             fluid {
35               ...GatsbyImageSharpFluid
36             }
37           }
38         }
39       }
40     }
41   }
42 }
44
45 export const query = graphql`query {
46   site {
47     siteMetadata {
48       title
49       description
50       author
51     }
52   }
53   cms {
54     blogPost {
55       id
56       title
57       description
58       image {
59         alt
60         file {
61           childImageSharp {
62             fluid {
63               ...GatsbyImageSharpFluid
64             }
65           }
66         }
67       }
68     }
69   }
70 }
72
73 const query = graphql`query {
74   site {
75     siteMetadata {
76       title
77       description
78       author
79     }
80   }
81   cms {
82     blogPost {
83       id
84       title
85       description
86       image {
87         alt
88         file {
89           childImageSharp {
90             fluid {
91               ...GatsbyImageSharpFluid
92             }
93           }
94         }
95       }
96     }
97   }
98 }
100
101 const query = graphql`query {
102   site {
103     siteMetadata {
104       title
105       description
106       author
107     }
108   }
109   cms {
110     blogPost {
111       id
112       title
113       description
114       image {
115         alt
116         file {
117           childImageSharp {
118             fluid {
119               ...GatsbyImageSharpFluid
120             }
121           }
122         }
123       }
124     }
125   }
126 }
128
129 const query = graphql`query {
130   site {
131     siteMetadata {
132       title
133       description
134       author
135     }
136   }
137   cms {
138     blogPost {
139       id
140       title
141       description
142       image {
143         alt
144         file {
145           childImageSharp {
146             fluid {
147               ...GatsbyImageSharpFluid
148             }
149           }
150         }
151       }
152     }
153   }
154 }
156
157 const query = graphql`query {
158   site {
159     siteMetadata {
160       title
161       description
162       author
163     }
164   }
165   cms {
166     blogPost {
167       id
168       title
169       description
170       image {
171         alt
172         file {
173           childImageSharp {
174             fluid {
175               ...GatsbyImageSharpFluid
176             }
177           }
178         }
179       }
180     }
181   }
182 }
184
185 const query = graphql`query {
186   site {
187     siteMetadata {
188       title
189       description
190       author
191     }
192   }
193   cms {
194     blogPost {
195       id
196       title
197       description
198       image {
199         alt
200         file {
201           childImageSharp {
202             fluid {
203               ...GatsbyImageSharpFluid
204             }
205           }
206         }
207       }
208     }
209   }
210 }
212
213 const query = graphql`query {
214   site {
215     siteMetadata {
216       title
217       description
218       author
219     }
220   }
221   cms {
222     blogPost {
223       id
224       title
225       description
226       image {
227         alt
228         file {
229           childImageSharp {
230             fluid {
231               ...GatsbyImageSharpFluid
232             }
233           }
234         }
235       }
236     }
237   }
238 }
240
241 const query = graphql`query {
242   site {
243     siteMetadata {
244       title
245       description
246       author
247     }
248   }
249   cms {
250     blogPost {
251       id
252       title
253       description
254       image {
255         alt
256         file {
257           childImageSharp {
258             fluid {
259               ...GatsbyImageSharpFluid
260             }
261           }
262         }
263       }
264     }
265   }
266 }
268
269 const query = graphql`query {
270   site {
271     siteMetadata {
272       title
273       description
274       author
275     }
276   }
277   cms {
278     blogPost {
279       id
280       title
281       description
282       image {
283         alt
284         file {
285           childImageSharp {
286             fluid {
287               ...GatsbyImageSharpFluid
288             }
289           }
290         }
291       }
292     }
293   }
294 }
296
297 const query = graphql`query {
298   site {
299     siteMetadata {
300       title
301       description
302       author
303     }
304   }
305   cms {
306     blogPost {
307       id
308       title
309       description
310       image {
311         alt
312         file {
313           childImageSharp {
314             fluid {
315               ...GatsbyImageSharpFluid
316             }
317           }
318         }
319       }
320     }
321   }
322 }
324
325 const query = graphql`query {
326   site {
327     siteMetadata {
328       title
329       description
330       author
331     }
332   }
333   cms {
334     blogPost {
335       id
336       title
337       description
338       image {
339         alt
340         file {
341           childImageSharp {
342             fluid {
343               ...GatsbyImageSharpFluid
344             }
345           }
346         }
347       }
348     }
349   }
350 }
351
352 const query = graphql`query {
353   site {
354     siteMetadata {
355       title
356       description
357       author
358     }
359   }
360   cms {
361     blogPost {
362       id
363       title
364       description
365       image {
366         alt
367         file {
368           childImageSharp {
369             fluid {
370               ...GatsbyImageSharpFluid
371             }
372           }
373         }
374       }
375     }
376   }
377 }
378
379 const query = graphql`query {
380   site {
381     siteMetadata {
382       title
383       description
384       author
385     }
386   }
387   cms {
388     blogPost {
389       id
390       title
391       description
392       image {
393         alt
394         file {
395           childImageSharp {
396             fluid {
397               ...GatsbyImageSharpFluid
398             }
399           }
400         }
401       }
402     }
403   }
404 }
406
407 const query = graphql`query {
408   site {
409     siteMetadata {
410       title
411       description
412       author
413     }
414   }
415   cms {
416     blogPost {
417       id
418       title
419       description
420       image {
421         alt
422         file {
423           childImageSharp {
424             fluid {
425               ...GatsbyImageSharpFluid
426             }
427           }
428         }
429       }
430     }
431   }
432 }
434
435 const query = graphql`query {
436   site {
437     siteMetadata {
438       title
439       description
440       author
441     }
442   }
443   cms {
444     blogPost {
445       id
446       title
447       description
448       image {
449         alt
450         file {
451           childImageSharp {
452             fluid {
453               ...GatsbyImageSharpFluid
454             }
455           }
456         }
457       }
458     }
459   }
460 }
462
463 const query = graphql`query {
464   site {
465     siteMetadata {
466       title
467       description
468       author
469     }
470   }
471   cms {
472     blogPost {
473       id
474       title
475       description
476       image {
477         alt
478         file {
479           childImageSharp {
480             fluid {
481               ...GatsbyImageSharpFluid
482             }
483           }
484         }
485       }
486     }
487   }
488 }
490
491 const query = graphql`query {
492   site {
493     siteMetadata {
494       title
495       description
496       author
497     }
498   }
499   cms {
500     blogPost {
501       id
502       title
503       description
504       image {
505         alt
506         file {
507           childImageSharp {
508             fluid {
509               ...GatsbyImageSharpFluid
510             }
511           }
512         }
513       }
514     }
515   }
516 }
518
519 const query = graphql`query {
520   site {
521     siteMetadata {
522       title
523       description
524       author
525     }
526   }
527   cms {
528     blogPost {
529       id
530       title
531       description
532       image {
533         alt
534         file {
535           childImageSharp {
536             fluid {
537               ...GatsbyImageSharpFluid
538             }
539           }
540         }
541       }
542     }
543   }
544 }
547
548 const query = graphql`query {
549   site {
550     siteMetadata {
551       title
552       description
553       author
554     }
555   }
556   cms {
557     blogPost {
558       id
559       title
560       description
561       image {
562         alt
563         file {
564           childImageSharp {
565             fluid {
566               ...GatsbyImageSharpFluid
567             }
568           }
569         }
570       }
571     }
572   }
573 }
576
577 const query = graphql`query {
578   site {
579     siteMetadata {
580       title
581       description
582       author
583     }
584   }
585   cms {
586     blogPost {
587       id
588       title
589       description
590       image {
591         alt
592         file {
593           childImageSharp {
594             fluid {
595               ...GatsbyImageSharpFluid
596             }
597           }
598         }
599       }
600     }
601   }
602 }
605
606 const query = graphql`query {
607   site {
608     siteMetadata {
609       title
610       description
611       author
612     }
613   }
614   cms {
615     blogPost {
616       id
617       title
618       description
619       image {
620         alt
621         file {
622           childImageSharp {
623             fluid {
624               ...GatsbyImageSharpFluid
625             }
626           }
627         }
628       }
629     }
630   }
631 }
634
635 const query = graphql`query {
636   site {
637     siteMetadata {
638       title
639       description
640       author
641     }
642   }
643   cms {
644     blogPost {
645       id
646       title
647       description
648       image {
649         alt
650         file {
651           childImageSharp {
652             fluid {
653               ...GatsbyImageSharpFluid
654             }
655           }
656         }
657       }
658     }
659   }
660 }
663
664 const query = graphql`query {
665   site {
666     siteMetadata {
667       title
668       description
669       author
670     }
671   }
672   cms {
673     blogPost {
674       id
675       title
676       description
677       image {
678         alt
679         file {
680           childImageSharp {
681             fluid {
682               ...GatsbyImageSharpFluid
683             }
684           }
685         }
686       }
687     }
688   }
689 }
692
693 const query = graphql`query {
694   site {
695     siteMetadata {
696       title
697       description
698       author
699     }
700   }
701   cms {
702     blogPost {
703       id
704       title
705       description
706       image {
707         alt
708         file {
709           childImageSharp {
710             fluid {
711               ...GatsbyImageSharpFluid
712             }
713           }
714         }
715       }
716     }
717   }
718 }
721
722 const query = graphql`query {
723   site {
724     siteMetadata {
725       title
726       description
727       author
728     }
729   }
730   cms {
731     blogPost {
732       id
733       title
734       description
735       image {
736         alt
737         file {
738           childImageSharp {
739             fluid {
740               ...GatsbyImageSharpFluid
741             }
742           }
743         }
744       }
745     }
746   }
747 }
750
751 const query = graphql`query {
752   site {
753     siteMetadata {
754       title
755       description
756       author
757     }
758   }
759   cms {
760     blogPost {
761       id
762       title
763       description
764       image {
765         alt
766         file {
767           childImageSharp {
768             fluid {
769               ...GatsbyImageSharpFluid
770             }
771           }
772         }
773       }
774     }
775   }
776 }
779
780 const query = graphql`query {
781   site {
782     siteMetadata {
783       title
784       description
785       author
786     }
787   }
788   cms {
789     blogPost {
790       id
791       title
792       description
793       image {
794         alt
795         file {
796           childImageSharp {
797             fluid {
798               ...GatsbyImageSharpFluid
799             }
800           }
801         }
802       }
803     }
804   }
805 }
808
809 const query = graphql`query {
810   site {
811     siteMetadata {
812       title
813       description
814       author
815     }
816   }
817   cms {
818     blogPost {
819       id
820       title
821       description
822       image {
823         alt
824         file {
825           childImageSharp {
826             fluid {
827               ...GatsbyImageSharpFluid
828             }
829           }
830         }
831       }
832     }
833   }
834 }
837
838 const query = graphql`query {
839   site {
840     siteMetadata {
841       title
842       description
843       author
844     }
845   }
846   cms {
847     blogPost {
848       id
849       title
850       description
851       image {
852         alt
853         file {
854           childImageSharp {
855             fluid {
856               ...GatsbyImageSharpFluid
857             }
858           }
859         }
860       }
861     }
862   }
863 }
866
867 const query = graphql`query {
868   site {
869     siteMetadata {
870       title
871       description
872       author
873     }
874   }
875   cms {
876     blogPost {
877       id
878       title
879       description
880       image {
881         alt
882         file {
883           childImageSharp {
884             fluid {
885               ...GatsbyImageSharpFluid
886             }
887           }
888         }
889       }
890     }
891   }
892 }
895
896 const query = graphql`query {
897   site {
898     siteMetadata {
899       title
900       description
901       author
902     }
903   }
904   cms {
905     blogPost {
906       id
907       title
908       description
909       image {
910         alt
911         file {
912           childImageSharp {
913             fluid {
914               ...GatsbyImageSharpFluid
915             }
916           }
917         }
918       }
919     }
920   }
921 }
924
925 const query = graphql`query {
926   site {
927     siteMetadata {
928       title
929       description
930       author
931     }
932   }
933   cms {
934     blogPost {
935       id
936       title
937       description
938       image {
939         alt
940         file {
941           childImageSharp {
942             fluid {
943               ...GatsbyImageSharpFluid
944             }
945           }
946         }
947       }
948     }
949   }
950 }
953
954 const query = graphql`query {
955   site {
956     siteMetadata {
957       title
958       description
959       author
960     }
961   }
962   cms {
963     blogPost {
964       id
965       title
966       description
967       image {
968         alt
969         file {
970           childImageSharp {
971             fluid {
972               ...GatsbyImageSharpFluid
973             }
974           }
975         }
976       }
977     }
978   }
979 }
982
983 const query = graphql`query {
984   site {
985     siteMetadata {
986       title
987       description
988       author
989     }
990   }
991   cms {
992     blogPost {
993       id
994       title
995       description
996       image {
997         alt
998         file {
999           childImageSharp {
1000             fluid {
1001               ...GatsbyImageSharpFluid
1002             }
1003           }
1004         }
1005       }
1006     }
1007   }
1008 }
1011
1012 const query = graphql`query {
1013   site {
1014     siteMetadata {
1015       title
1016       description
1017       author
1018     }
1019   }
1020   cms {
1021     blogPost {
1022       id
1023       title
1024       description
1025       image {
1026         alt
1027         file {
1028           childImageSharp {
1029             fluid {
1030               ...GatsbyImageSharpFluid
1031             }
1032           }
1033         }
1034       }
1035     }
1036   }
1037 }
1040
1041 const query = graphql`query {
1042   site {
1043     siteMetadata {
1044       title
1045       description
1046       author
1047     }
1048   }
1049   cms {
1050     blogPost {
1051       id
1052       title
1053       description
1054       image {
1055         alt
1056         file {
1057           childImageSharp {
1058             fluid {
1059               ...GatsbyImageSharpFluid
1060             }
1061           }
1062         }
1063       }
1064     }
1065   }
1066 }
1069
1070 const query = graphql`query {
1071   site {
1072     siteMetadata {
1073       title
1074       description
1075       author
1076     }
1077   }
1078   cms {
1079     blogPost {
1080       id
1081       title
1082       description
1083       image {
1084         alt
1085         file {
1086           childImageSharp {
1087             fluid {
1088               ...GatsbyImageSharpFluid
1089             }
1090           }
1091         }
1092       }
1093     }
1094   }
1095 }
1098
1099 const query = graphql`query {
1100   site {
1101     siteMetadata {
1102       title
1103       description
1104       author
1105     }
1106   }
1107   cms {
1108     blogPost {
1109       id
1110       title
1111       description
1112       image {
1113         alt
1114         file {
1115           childImageSharp {
1116             fluid {
1117               ...GatsbyImageSharpFluid
1118             }
1119           }
1120         }
1121       }
1122     }
1123   }
1124 }
1127
1128 const query = graphql`query {
1129   site {
1130     siteMetadata {
1131       title
1132       description
1133       author
1134     }
1135   }
1136   cms {
1137     blogPost {
1138       id
1139       title
1140       description
1141       image {
1142         alt
1143         file {
1144           childImageSharp {
1145             fluid {
1146               ...GatsbyImageSharpFluid
1147             }
1148           }
1149         }
1150       }
1151     }
1152   }
1153 }
1156
1157 const query = graphql`query {
1158   site {
1159     siteMetadata {
1160       title
1161       description
1162       author
1163     }
1164   }
1165   cms {
1166     blogPost {
1167       id
1168       title
1169       description
1170       image {
1171         alt
1172         file {
1173           childImageSharp {
1174             fluid {
1175               ...GatsbyImageSharpFluid
1176             }
1177           }
1178         }
1179       }
1180     }
1181   }
1182 }
1185
1186 const query = graphql`query {
1187   site {
1188     siteMetadata {
1189       title
1190       description
1191       author
1192     }
1193   }
1194   cms {
1195     blogPost {
1196       id
1197       title
1198       description
1199       image {
1200         alt
1201         file {
1202           childImageSharp {
1203             fluid {
1204               ...GatsbyImageSharpFluid
1205             }
1206           }
1207         }
1208       }
1209     }
1210   }
1211 }
1214
1215 const query = graphql`query {
1216   site {
1217     siteMetadata {
1218       title
1219       description
1220       author
1221     }
1222   }
1223   cms {
1224     blogPost {
1225       id
1226       title
1227       description
1228       image {
1229         alt
1230         file {
1231           childImageSharp {
1232             fluid {
1233               ...GatsbyImageSharpFluid
1234             }
1235           }
1236         }
1237       }
1238     }
1239   }
1240 }
1243
1244 const query = graphql`query {
1245   site {
1246     siteMetadata {
1247       title
1248       description
1249       author
1250     }
1251   }
1252   cms {
1253     blogPost {
1254       id
1255       title
1256       description
1257       image {
1258         alt
1259         file {
1260           childImageSharp {
1261             fluid {
1262               ...GatsbyImageSharpFluid
1263             }
1264           }
1265         }
1266       }
1267     }
1268   }
1269 }
1272
1273 const query = graphql`query {
1274   site {
1275     siteMetadata {
1276       title
1277       description
1278       author
1279     }
1280   }
1281   cms {
1282     blogPost {
1283       id
1284       title
1285       description
1286       image {
1287         alt
1288         file {
1289           childImageSharp {
1290             fluid {
1291               ...GatsbyImageSharpFluid
1292             }
1293           }
1294         }
1295       }
1296     }
1297   }
1298 }
1299
1300 const query = graphql`query {
1301   site {
1302     siteMetadata {
1303       title
1304       description
1305       author
1306     }
1307   }
1308   cms {
1309     blogPost {
1310       id
1311       title
1312       description
1313       image {
1314         alt
1315         file {
1316           childImageSharp {
1317             fluid {
1318               ...GatsbyImageSharpFluid
1319             }
1320           }
1321         }
1322       }
1323     }
1324   }
1325 }
1328
1329 const query = graphql`query {
1330   site {
1331     siteMetadata {
1332       title
1333       description
1334       author
1335     }
1336   }
1337   cms {
1338     blogPost {
1339       id
1340       title
1341       description
1342       image {
1343         alt
1344         file {
1345           childImageSharp {
1346             fluid {
1347               ...GatsbyImageSharpFluid
1348             }
1349           }
1350         }
1351       }
1352     }
1353   }
1354 }
1357
1358 const query = graphql`query {
1359   site {
1360     siteMetadata {
1361       title
1362       description
1363       author
1364     }
1365   }
1366   cms {
1367     blogPost {
1368       id
1369       title
1370       description
1371       image {
1372         alt
1373         file {
1374           childImageSharp {
1375             fluid {
1376               ...GatsbyImageSharpFluid
1377             }
1378           }
1379         }
1380       }
1381     }
1382   }
1383 }
1386
1387 const query = graphql`query {
1388   site {
1389     siteMetadata {
1390       title
1391       description
1392       author
1393     }
1394   }
1395   cms {
1396     blogPost {
1397       id
1398       title
1399       description
1400       image {
1401         alt
1402         file {
1403           childImageSharp {
1404             fluid {
1405               ...GatsbyImageSharpFluid
1406             }
1407           }
1408         }
1409       }
1410     }
1411   }
1412 }
1415
1416 const query = graphql`query {
1417   site {
1418     siteMetadata {
1419       title
1420       description
1421       author
1422     }
1423   }
1424   cms {
1425     blogPost {
1426       id
1427       title
1428       description
1429       image {
1430         alt
1431         file {
1432           childImageSharp {
1433             fluid {
1434               ...GatsbyImageSharpFluid
1435             }
1436           }
1437         }
1438       }
1439     }
1440   }
1441 }
1444
1445 const query = graphql`query {
1446   site {
1447     siteMetadata {
1448       title
1449       description
1450       author
1451     }
1452   }
1453   cms {
1454     blogPost {
1455       id
1456       title
1457       description
1458       image {
1459         alt
1460         file {
1461           childImageSharp {
1462             fluid {
1463               ...GatsbyImageSharpFluid
1464             }
1465           }
1466         }
1467       }
1468     }
1469   }
1470 }
1473
1474 const query = graphql`query {
1475   site {
1476     siteMetadata {
1477       title
1478       description
1479       author
1480     }
1481   }
1482   cms {
1483     blogPost {
1484       id
1485       title
1486       description
1487       image {
1488         alt
1489         file {
1490           childImageSharp {
1491             fluid {
1492               ...GatsbyImageSharpFluid
1493             }
1494           }
1495         }
1496       }
1497     }
1498   }
1499 }
1499
1500 const query = graphql`query {
1501   site {
1502     siteMetadata {
1503       title
1504       description
1505       author
1506     }
1507   }
1508   cms {
1509     blogPost {
1510       id
1511       title
1512       description
1513       image {
1514         alt
1515         file {
1516           childImageSharp {
1517             fluid {
1518               ...GatsbyImageSharpFluid
1519             }
1520           }
1521         }
1522       }
1523     }
1524   }
1525 }
1528
1529 const query = graphql`query {
1530   site {
1531     siteMetadata {
1532       title
1533       description
1534       author
1535     }
1536   }
1537   cms {
1538     blogPost {
1539       id
1540       title
1541       description
1542       image {
1543         alt
1544         file {
1545           childImageSharp {
1546             fluid {
1547               ...GatsbyImageSharpFluid
1548             }
1549           }
1550         }
1551       }
1552     }
1553   }
1554 }
1557
1558 const query = graphql`query {
1559   site {
1560     siteMetadata {
1561       title
1562       description
1563       author
1564     }
1565   }
1566   cms {
1567     blogPost {
1568       id
1569       title
1570       description
1571       image {
1572         alt
1573         file {
1574           childImageSharp {
1575             fluid {
1576               ...GatsbyImageSharpFluid
1577             }
1578           }
1579         }
1580       }
1581     }
1582   }
1583 }
1586
1587 const query = graphql`query {
1588   site {
1589     siteMetadata {
1590       title
1591       description
1592       author
1593     }
1594   }
1595   cms {
1596     blogPost {
1597       id
1598       title
1599       description
1600       image {
1601         alt
1602         file {
1603           childImageSharp {
1604             fluid {
1605               ...GatsbyImageSharpFluid
1606             }
1607           }
1608         }
1609       }
1610     }
1611   }
1612 }
1615
1616 const query = graphql`query {
1617   site {
1618     siteMetadata {
1619       title
1620       description
1621       author
1622     }
1623   }
1624   cms {
1625     blogPost {
1626       id
1627       title
1628       description
1629       image {
1630         alt
1631         file {
1632           childImageSharp {
1633             fluid {
1634               ...GatsbyImageSharpFluid
1635             }
1636           }
1637         }
1638       }
1639     }
1640   }
1641 }
1644
1645 const query = graphql`query {
1646   site {
1647     siteMetadata {
1648       title
1649       description
1650       author
1651     }
1652   }
1653   cms {
1654     blogPost {
1655       id
1656       title
1657       description
1658       image {
1659         alt
1660         file {
1661           childImageSharp {
1662             fluid {
1663               ...GatsbyImageSharpFluid
1664             }
1665           }
1666         }
1667       }
1668     }
1669   }
1670 }
1673
1674 const query = graphql`query {
1675   site {
1676     siteMetadata {
1677       title
1678       description
1679       author
1680     }
1681   }
1682   cms {
1683     blogPost {
1684       id
1685       title
1686       description
1687       image {
1688         alt
1689         file {
1690           childImageSharp {
1691             fluid {
1692               ...GatsbyImageSharpFluid
1693             }
1694           }
1695         }
1696       }
1697     }
1698   }
1699 }
1699
1700 const query = graphql`query {
1701   site {
1702     siteMetadata {
1703       title
1704       description
1705       author
1706     }
1707   }
1708   cms {
1709     blogPost {
1710       id
1711       title
1712       description
1713       image {
1714         alt
1715         file {
1716           childImageSharp {
1717             fluid {
1718               ...GatsbyImageSharpFluid
1719             }
1720           }
1721         }
1722       }
1723     }
1724   }
1725 }
1728
1729 const query = graphql`query {
1730   site {
1731     siteMetadata {
1732       title
1733       description
1734       author
1735     }
1736   }
1737   cms {
1738     blogPost {
1739       id
1740       title
1741       description
1742       image {
1743         alt
1744         file {
1745           childImageSharp {
1746             fluid {
1747               ...GatsbyImageSharpFluid
1748             }
1749           }
1750         }
1751       }
1752     }
1753   }
1754 }
1757
1758 const query = graphql`query {
1759   site {
1760     siteMetadata {
1761       title
1762       description
1763       author
1764     }
1765   }
1766   cms {
1767     blogPost {
1768       id
1769       title
1770       description
1771       image {
1772         alt
1773         file {
1774           childImageSharp {
1775             fluid {
1776               ...GatsbyImageSharpFluid
1777             }
1778           }
1779         }
1780       }
1781     }
1782   }
1783 }
1786
1787 const query = graphql`query {
1788   site {
1789     siteMetadata {
1790       title
1791       description
1792       author
1793     }
1794   }
1795   cms {
1796     blogPost {
1797       id
1798       title
1799       description
1800       image {
1801         alt
1802         file {
1803           childImageSharp {
1804             fluid {
1805               ...GatsbyImageSharpFluid
1806             }
1807           }
1808         }
1809       }
1810     }
1811   }
1812 }
1815
1816 const query = graphql`query {
1817   site {
1818     siteMetadata {
1819       title
1820       description
1821       author
1822     }
1823   }
1824   cms {
1825     blogPost {
1826       id
1827       title
1828       description
1829       image {
1830         alt
1831         file {
1832           childImageSharp {
1833             fluid {
1834               ...GatsbyImageSharpFluid
1835             }
1836           }
1837         }
1838       }
1839     }
1840   }
1841 }
1844
1845 const query = graphql`query {
1846   site {
1847     siteMetadata {
1848       title
1849       description
1850       author
1851     }
1852   }
1853   cms {
1854     blogPost {
1855       id
1856       title
1857       description
1858       image {
1859         alt
1860         file {
1861           childImageSharp {
1862             fluid {
1863               ...GatsbyImageSharpFluid
1864             }
1865           }
1866         }
1867       }
1868     }
1869   }
1870 }
1873
1874 const query = graphql`query {
1875   site {
1876     siteMetadata {
1877       title
1878       description
1879       author
1880     }
1881   }
1882   cms {
1883     blogPost {
1884       id
1885       title
1886       description
1887       image {
1888         alt
1889         file {
1890           childImageSharp {
1891             fluid {
1892               ...GatsbyImageSharpFluid
1893             }
1894           }
1895         }
1896       }
1897     }
1898   }
1899 }
1899
1900 const query = graphql`query {
1901   site {
1902     siteMetadata {
1903       title
1904       description
1905       author
1906     }
1907   }
1908   cms {
1909     blogPost {
1910       id
1911       title
1912       description
1913       image {

```



1.4 Installation & Setup

1. Search VS Code

KNOWLEDGE GATE



1.5 VsCode Extensions

1. Live Server
2. Prettier



Level 0

Setup & Fundamentals

2. Website Components And Fundamentals



2.1 Client Side vs Server Side

	Client Side	Server Side
Execution Location	Executes on user's device.	Executes on a remote machine.
Languages	Primarily JavaScript, HTML, CSS.	PHP, Python, Java, Node.js, etc.
Main Job	Makes clicks and scrolls work	Manages saved information
Access Level	Can't access server data directly	Can read/write files, interact with databases.
Speed	Quicker for UI changes	Slower due to network latency.

2.2 FrontEnd / BackEnd / FullStack



Client Side / Front-End
Web Development

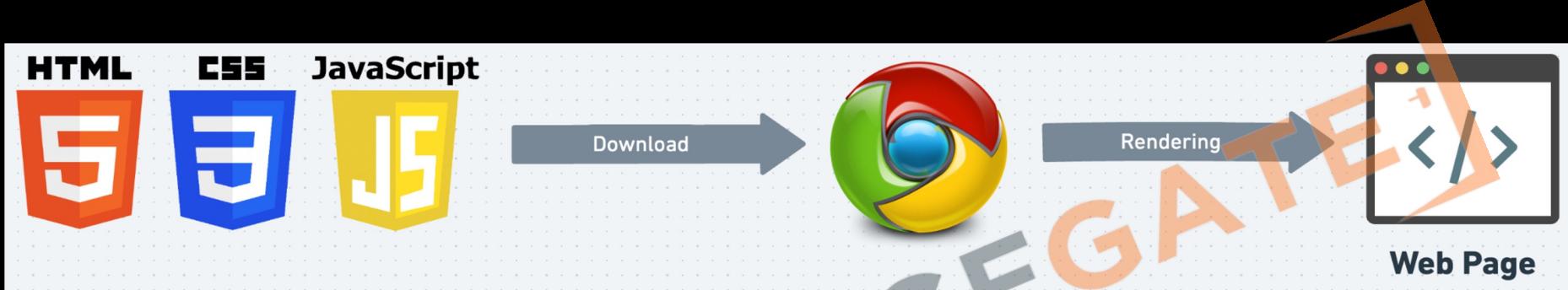


Server Side
Back-End



Full Stack

2.3 Role of Browser



1. **Displays Web Page:** Turns HTML code into what you see on screen.
2. **User Clicks:** Helps you interact with the web page.
3. **Updates Content:** Allows changes to the page using JavaScript.
4. **Loads Files:** Gets HTML, images, etc., from the server.



2.4 HTML

(Hypertext Markup Language)

1. **Structure:** Sets up the layout.
2. **Content:** Adds text, images, links.
3. **Tags:** Uses elements like `<p>`, `<a>`.
4. **Hierarchy:** Organizes elements in a tree.





2.5 CSS

(Cascading Style Sheets)

1. **Style:** Sets the look and feel.
2. **Colors & Fonts:** Customizes text and background.
3. **Layout:** Controls position and size.
4. **Selectors:** Targets specific **HTML** elements.





2.6 JS

(Java Script)

1. JavaScript has nothing to do with Java
2. Actions: Enables interactivity.
3. Updates: Alters page without reloading.
4. Events: Responds to user actions.
5. Data: Fetches and sends info to server.



Level 0 Revision

Setup & Fundamentals

1. IDE or Code Editor
 1. What is IDE
 2. Need of IDE
 3. IDE Selection
 4. Installation and Setup
 5. VsCode Extensions
2. Website Components and Fundamentals
 1. Client Side vs Server Side
 2. FrontEnd / BackEnd / FullStack
 3. Role of Browser
 4. HTML
 5. CSS
 6. JS



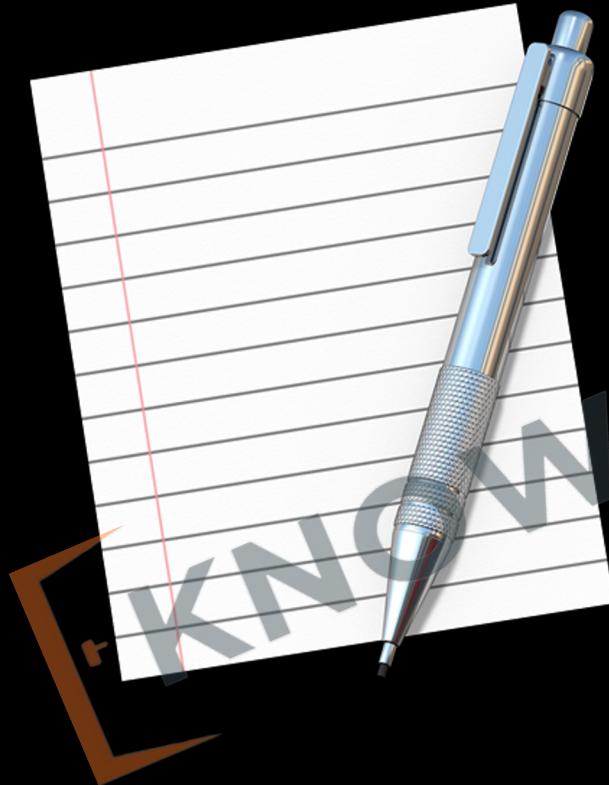
Level 1

HTML Basics

1. Starting up
 1. First File using Text Editor
 2. File Extension
 3. Opening the project in VsCode
 4. Index.html
2. Basics of HTML
 1. What are Tags
 2. Using Emmet ! to generate code
 3. Basic HTML Page
 4. MDN Documentation
 5. Comments
 6. Case Sensitivity

Level 1

HTML Basics



1. Starting Up

KNOWLEDGE STATE

1.1 First file using Text Editor

1. Create a folder with name **First Project** on your Desktop.
2. Open **Notepad**.
3. Create a file and save it as **index.html**
4. Copy Sample code
5. Open **Browser** and Check.



1.2 File Extension

HTML

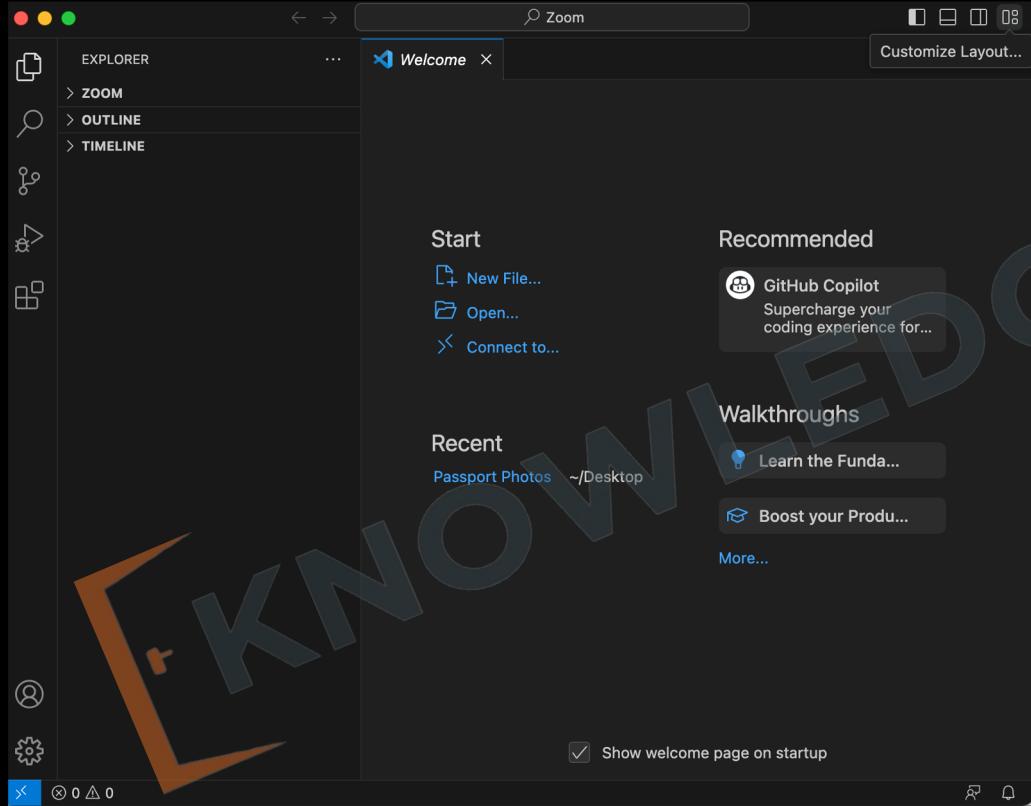
1. Most commonly used.
2. Works across all browsers.
3. Widely recognized and supported.
4. Typically saved as .html.



HTM

1. Less commonly used.
2. Originated for compatibility with older systems.
3. Works same as .html.
4. Typically saved as .htm.

1.3 Opening project in VsCode



E-KNOWLEDGE GATE

1.4 Importance of index.html

1. Default name of a website's homepage.
2. First page users see when visiting a website
3. Important for SEO (Search Engine Optimization)
4. Provides uniform starting point across servers
5. Serves as fallback when no file is specified in URL



Level 1

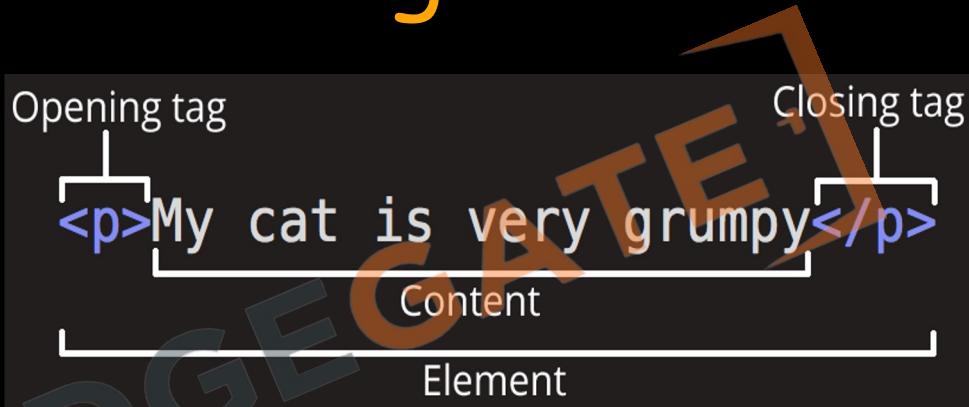
HTML Basics

2. Basics of HTML



2.1 What are Tags

1. Elements that are used to create a website are called HTML Tags.
2. Tags can contain content or other HTML tags.
3. Define elements like text, images, links



2.2 Using Emmet ! to generate code



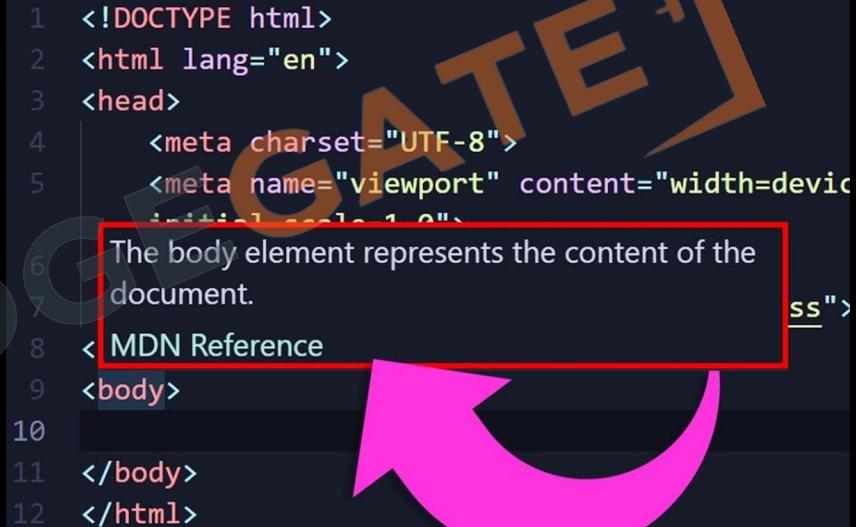
1. Type ! and wait for suggestions.

2.3 Basic HTML Page

```
<!DOCTYPE html>           Defines the HTML Version  
  
<html lang="en">          Parent of all HTML tags / Root element  
  
    <head>                Parent of meta data tags  
        <title>My First Webpage</title>  Title of the web page  
    </head>  
  
    <body>                Parent of content tags  
        <h1>Hello World!</h1>  Heading tag  
    </body>  
  
</html>
```

2.4 MDN Documentation

1. Visit <https://developer.mozilla.org/>
2. Official resource for HTML
3. Offers comprehensive guides and tutorials
4. Includes examples for real-world use
5. Updated with latest HTML features
6. Trusted by developers worldwide



The screenshot shows a portion of an MDN Reference page for the `<body>` element. The code is:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   </head>
7   <body>
8     <h1>MDN Reference</h1>
9     <p>The body element represents the content of the document.</p>
10    </body>
11  </html>
```

A red box highlights the explanatory text: "The body element represents the content of the document." A large pink arrow points from the bottom right towards this highlighted text.

2.5 Comments

1. Used to add **notes** in HTML code
2. **Not displayed** on the web page
3. Syntax: `<!-- Comment here -->`
4. Helpful for **code organization**
5. Can be **multi-line** or **single-line**

Writing comments in HTML

Single-line Comment

```
1 <!--This is a single line  
comment in HTML. You cannot  
see it on a webpage. Click  
on view-source to see a  
message I left just for you.  
-->
```

Multi-line Comment

```
1 <!-- This is a multi-line  
comment in HTML.  
2 You cannot see it on a  
webpage.  
3 If you view-source on the  
browser you can see the  
comment there.-->
```

2.6 Case Sensitivity

1. HTML is case-insensitive for tag names
2. Attribute names are also be case-insensitive
3. Best practice: use lowercase for consistency

<html> = <HTML>

<p> = <P>

<head> = <HEAD>

<body> = <BODY>

Level 1 Revision

HTML Basics

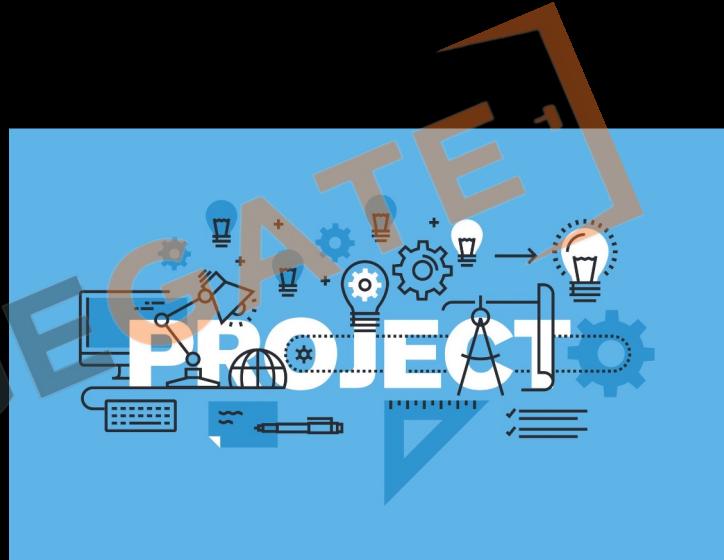
1. Starting up
 1. First File using Text Editor
 2. File Extension
 3. Opening the project in VsCode
 4. Index.html
2. Basics of HTML
 1. What are Tags
 2. Using Emmet ! to generate code
 3. Basic HTML Page
 4. MDN Documentation
 5. Comments
 6. Case Sensitivity



Project Level 1

HTML Basics

1. Create a new project with Index.html
2. Generate boilerplate code using Emmet
3. Write “I am learning with Prashant sir”
4. Use comments
5. Also use Case insensitive tags



Level 2

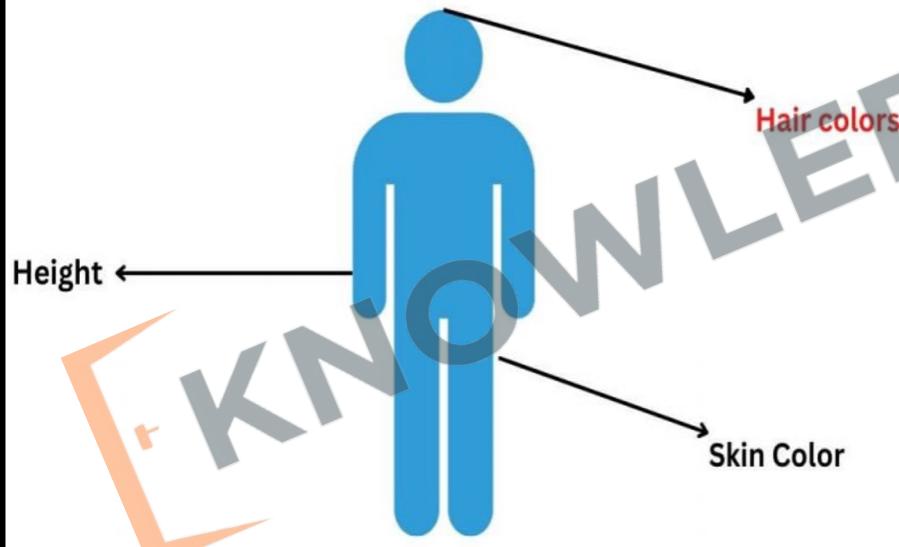
Must-Use HTML Tags

1. HTML Attributes
 1. What are HTML Attributes
 2. Id Property
2. HTML Tags
 1. Heading Tag
 2. Paragraph Tag
 3.
 <HR> tags
 4. Image Tag
 5. Video Tag
 6. Anchor Tag
 7. Bold / Italic / Underline / Strikethrough
 8. Pre Tag
 9. Big / Small Tag
 10. Superscript / Subscript
3. Character Entity Reference
 1. What are Character Entity References

Level 2

Must-Use HTML Tags

Attribute



1. HTML Attributes

1.1 What are HTML Attributes?

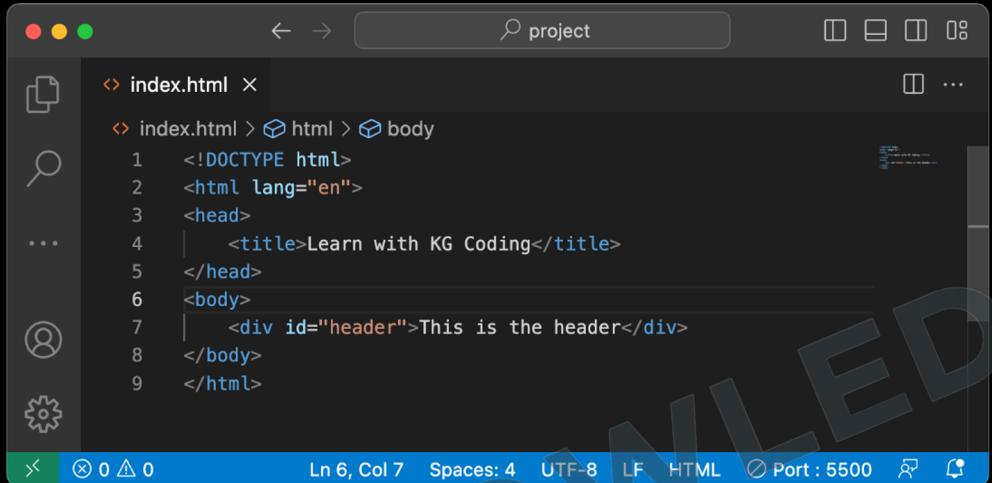
Html Attributes

Attribute

```
<tag attribute="value">Text Content </tag>
```

1. Provides additional information about elements
2. Placed within opening tags
3. Common examples: href, src, alt
4. Use name=value format
5. Can be single or multiple per element

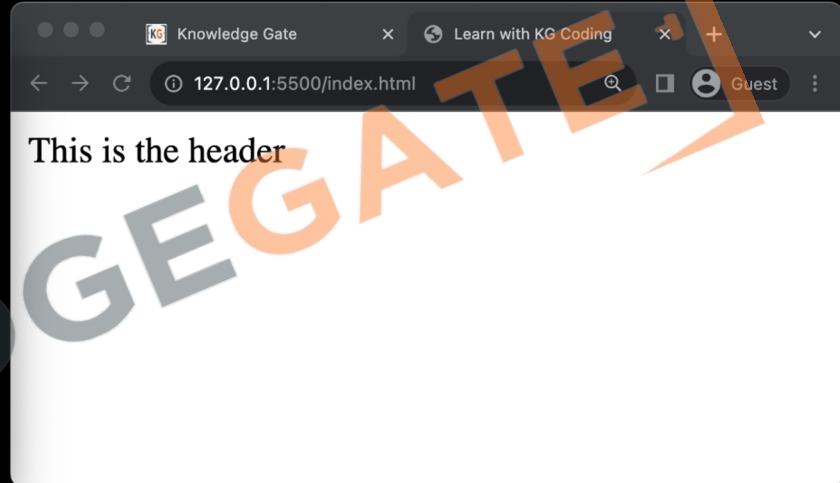
1.2 id property



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <div id="header">This is the header</div>
</body>
</html>
```

The code editor interface includes a sidebar with icons for file, search, and settings, and a status bar at the bottom.



- **Unique Identifier:** Each id should be unique within a page.
- **Anchoring:** Allows for direct links to sections using the `#id` syntax in URLs.
- **CSS & JavaScript:** Used for selecting elements for styling or scripting.

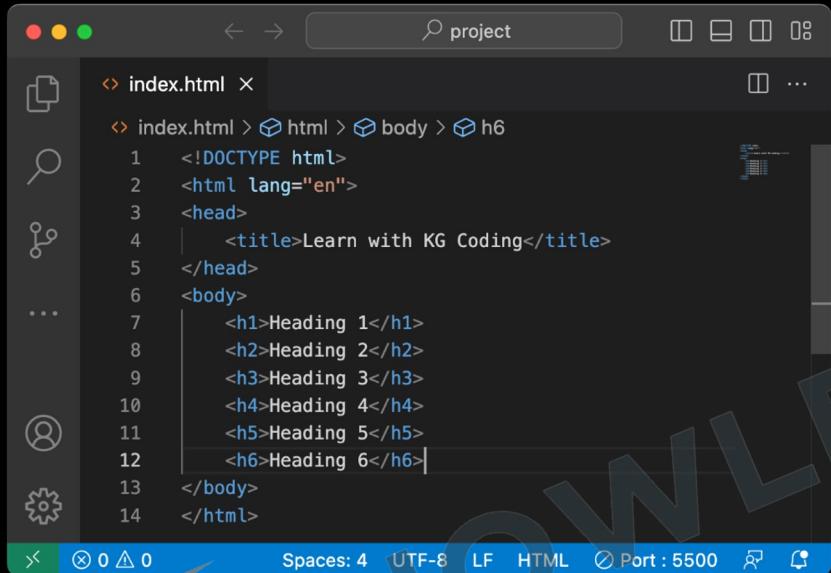
Level 2

Must-Use HTML Tags



2. Must-Use HTML Tags

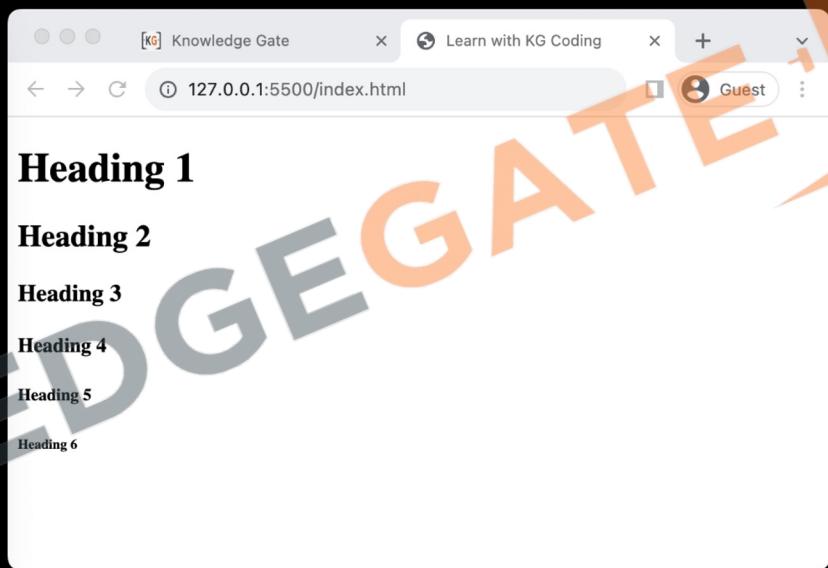
2.1 Heading Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, displaying the following HTML code:

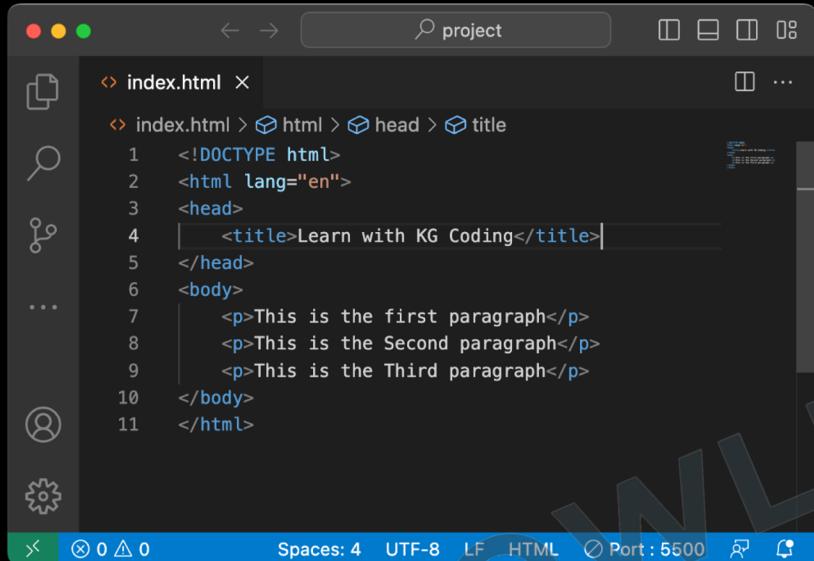
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <h3>Heading 3</h3>
    <h4>Heading 4</h4>
    <h5>Heading 5</h5>
    <h6>Heading 6</h6>
</body>
</html>
```

The code editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom indicating "Spaces: 4", "UTF-8", "LF", "HTML", and "Port : 5500".



1. Defines **headings** in a document
2. Ranges from **<h1>** to **<h6>**
3. **<h1>** is most important, **<h6>** is least
4. Important for **SEO**
5. Helps in structuring content

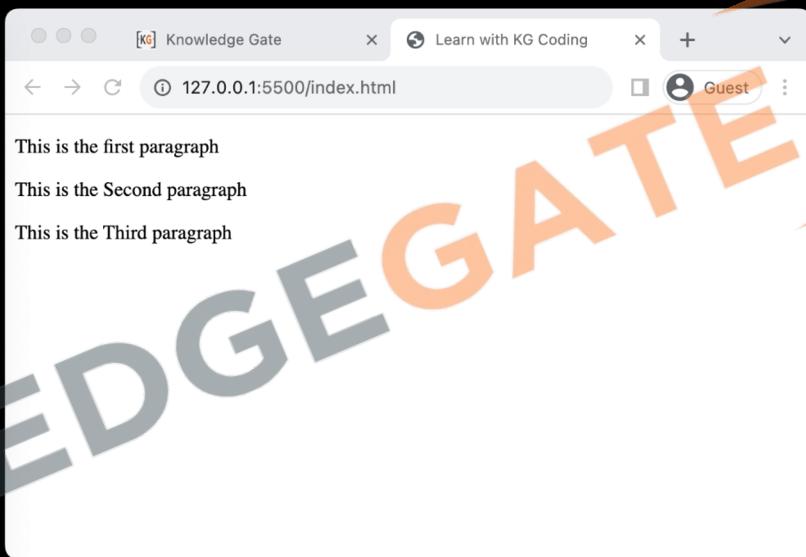
2.2 Paragraph Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

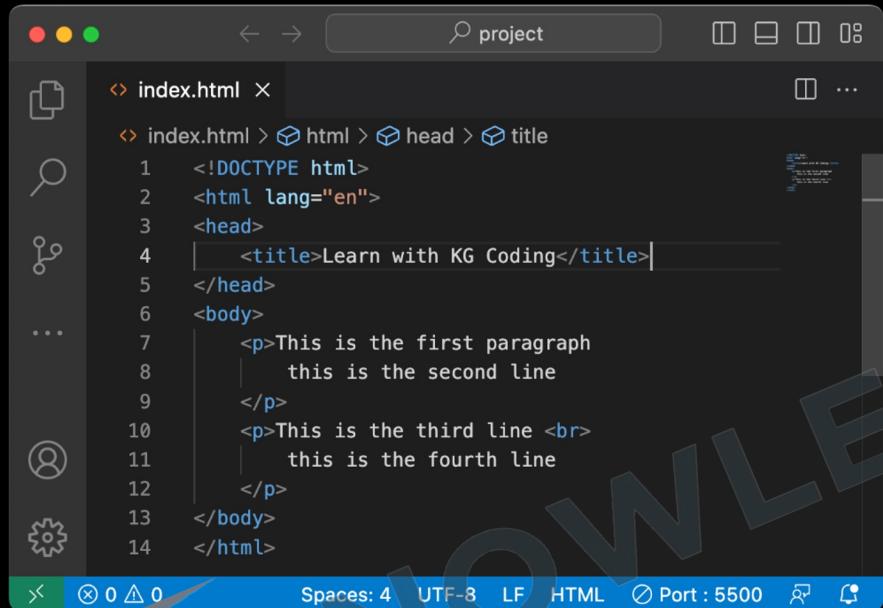
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <p>This is the first paragraph</p>
    <p>This is the Second paragraph</p>
    <p>This is the Third paragraph</p>
</body>
</html>
```

The code editor interface includes a sidebar with icons for file, search, and user, and a bottom status bar showing "Spaces: 4", "UTF-8", "LF", "HTML", "Port : 5500", and a refresh icon.



1. Used for defining **paragraphs**
2. Enclosed within **<p>** and **</p>** tags
3. Adds **automatic spacing** before and after
4. **Text wraps** to next line inside tag
5. Common in **text-heavy content**

2.3
 Tag



A screenshot of a code editor window titled "index.html". The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <p>This is the first paragraph<br/>
        this is the second line</p>
    <p>This is the third line<br/>
        this is the fourth line</p>
</body>
</html>
```

The code editor has a sidebar with various icons for file operations, search, and help. The bottom status bar shows "Spaces: 4", "UTF-8", "LF", "HTML", and "Port : 5500".



A screenshot of a web browser window titled "Knowledge Gate". The URL is "127.0.0.1:5500/index.html". The page content is:

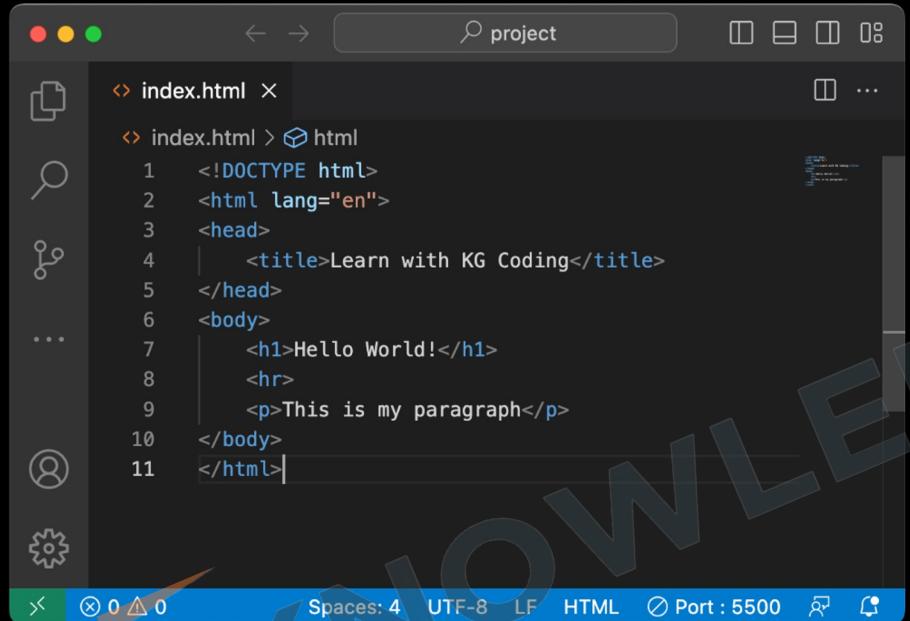
This is the first paragraph
this is the second line

This is the third line
this is the fourth line

A large, semi-transparent watermark reading "KNOWLEDGE GATE" is overlaid across the bottom of the browser window.

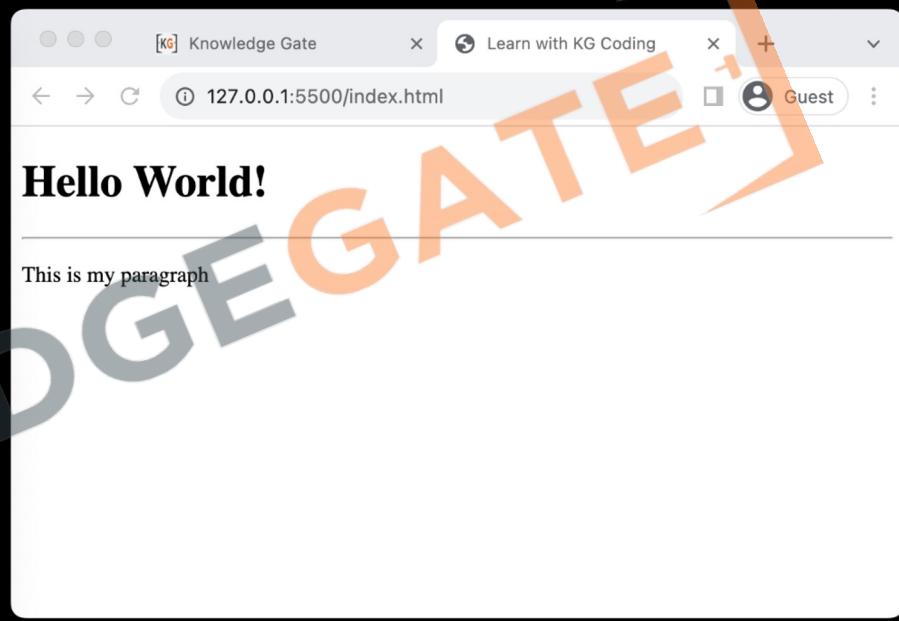
1.
 adds a line break within text
2.
 is empty, no closing tag needed
3.
 and
 are both valid

2.3 <HR> Tag



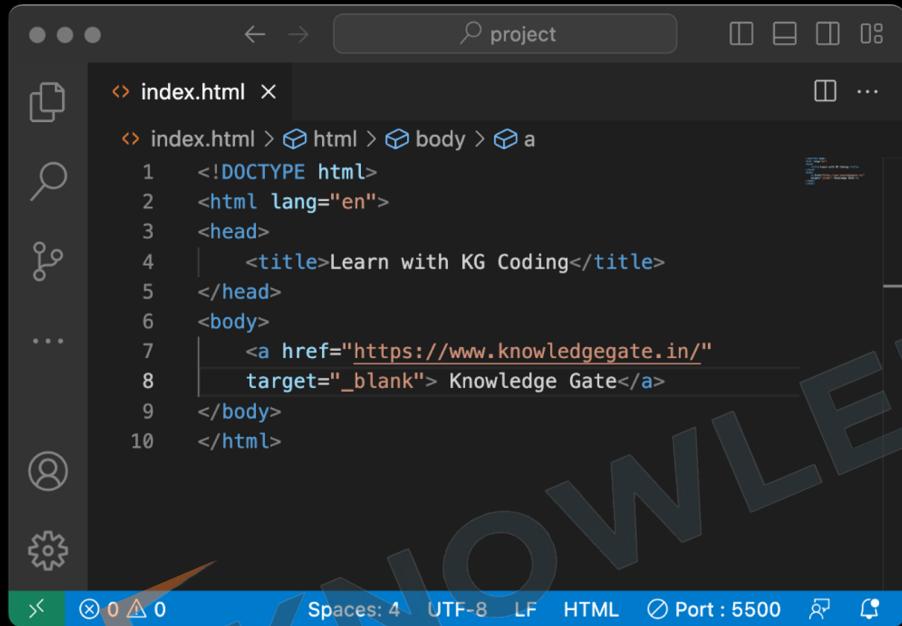
A screenshot of a code editor window titled "index.html". The code is as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <h1>Hello World!</h1>
    <hr>
    <p>This is my paragraph</p>
</body>
</html>
```



1. <hr> creates a horizontal rule or line
2. <hr> also empty, acts as a divider

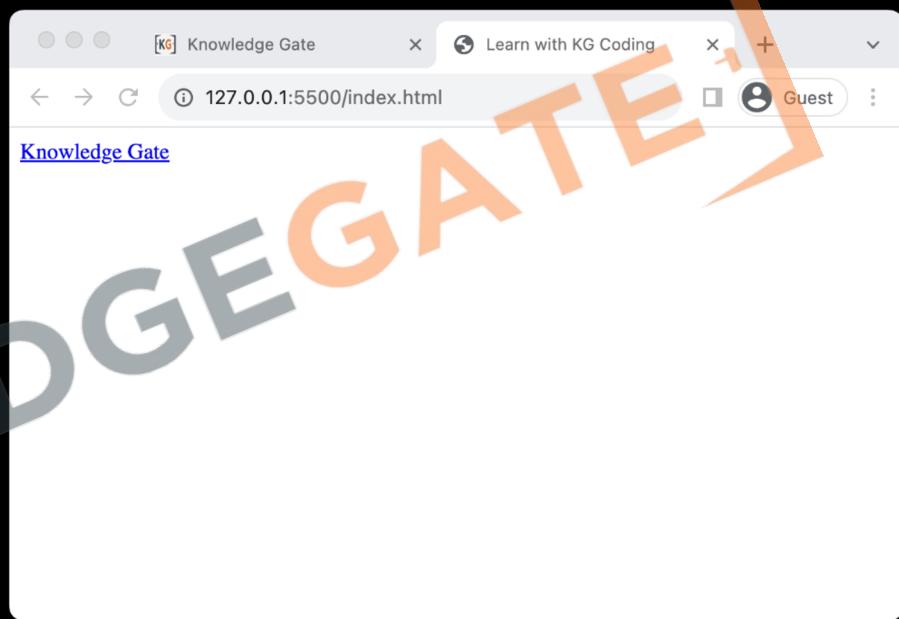
2.6 Anchor Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

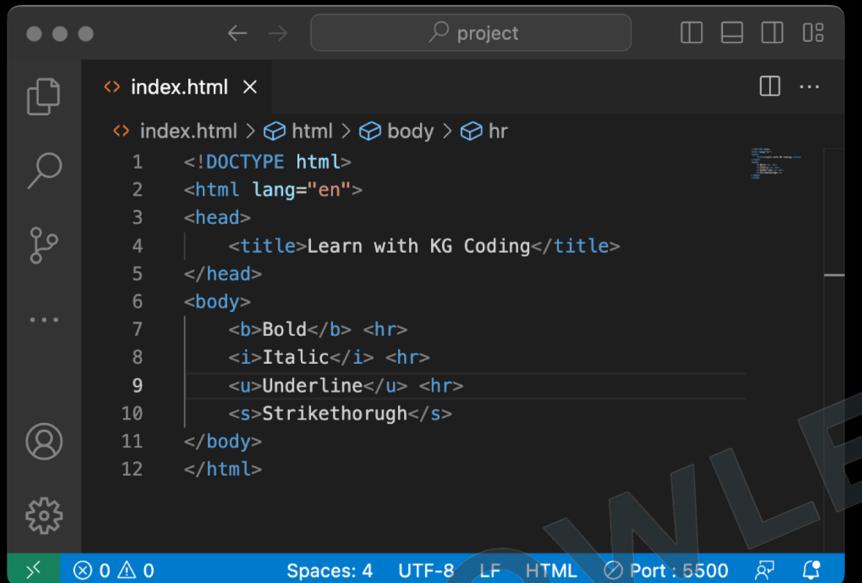
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <a href="https://www.knowledgegate.in/" target="_blank"> Knowledge Gate</a>
</body>
</html>
```

The code editor has a dark theme with orange icons. The status bar at the bottom shows "Spaces: 4" and "Port : 5500".



1. Used for creating **hyperlinks**
2. Requires **href** attribute for URL
3. Can link to external sites or internal pages
4. Supports **target** attribute to control link behavior

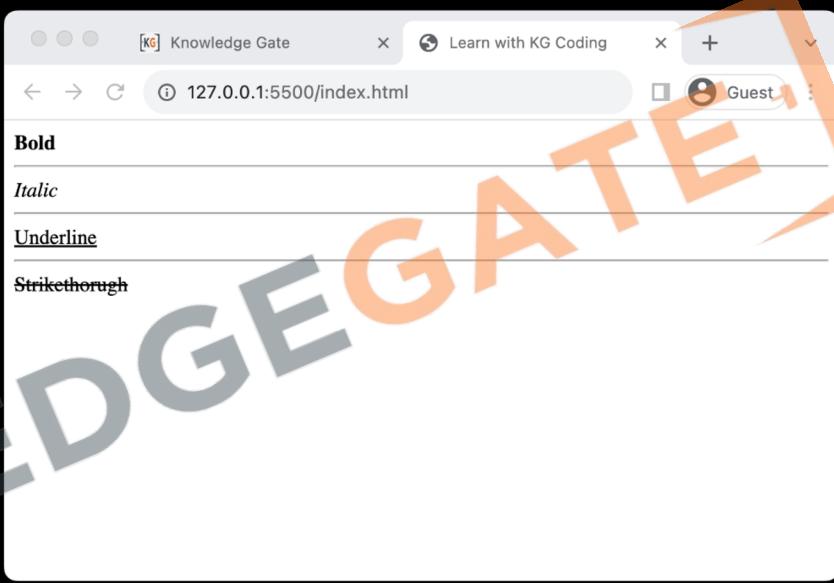
2.7 Bold/Italic/Underline/Strikethrough Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, displaying the following code:

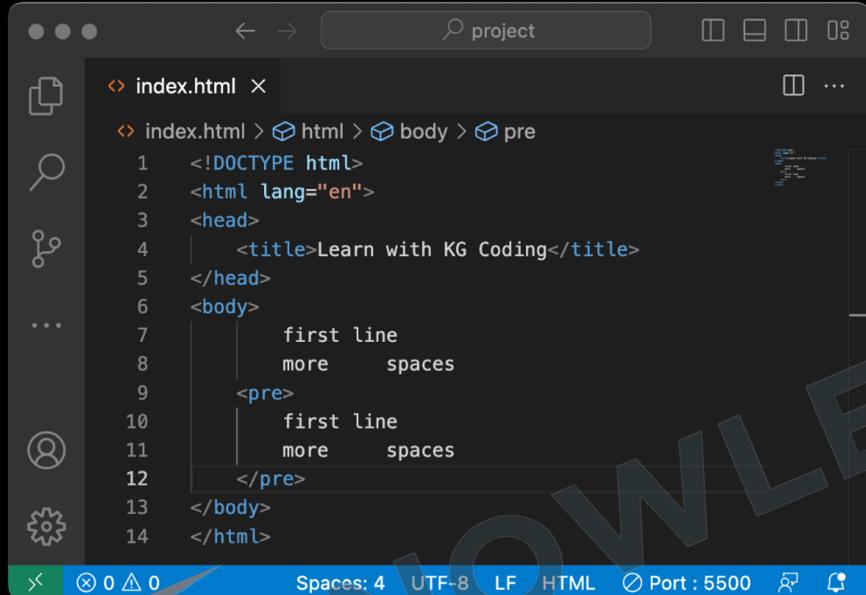
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <b>Bold</b> <hr>
    <i>Italic</i> <hr>
    <u>Underline</u> <hr>
    <s>Strikethrough</s>
</body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. It includes standard icons for file operations like new, save, and search, and a sidebar with various project-related icons.

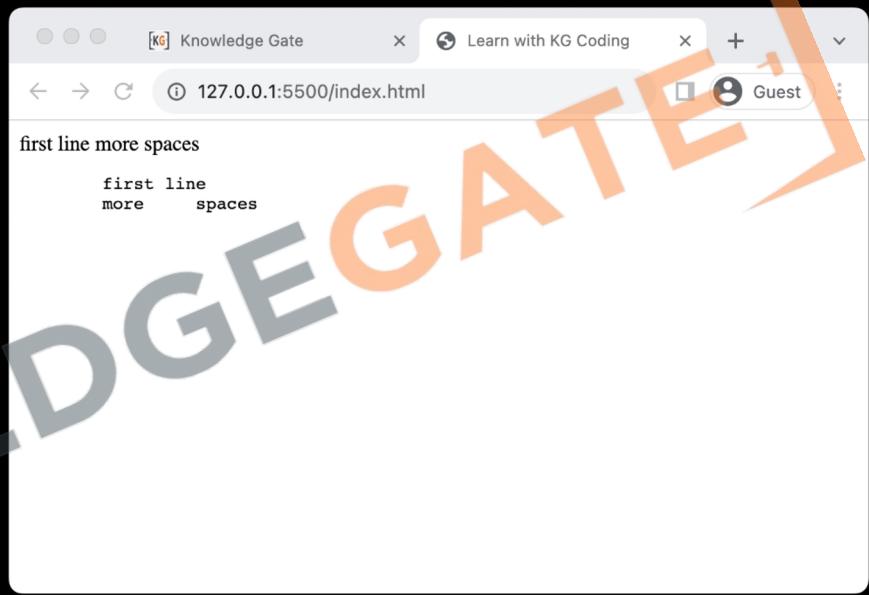


1. **** makes text bold
2. **<i>** makes text italic
3. **<u>** underlines text
4. **<s>** or **<strike>** applies strikethrough
5. Primarily used for text styling and emphasis

2.8 Pre Tag



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    first line
    more spaces
<pre>
    first line
    more spaces
</pre>
</body>
</html>
```

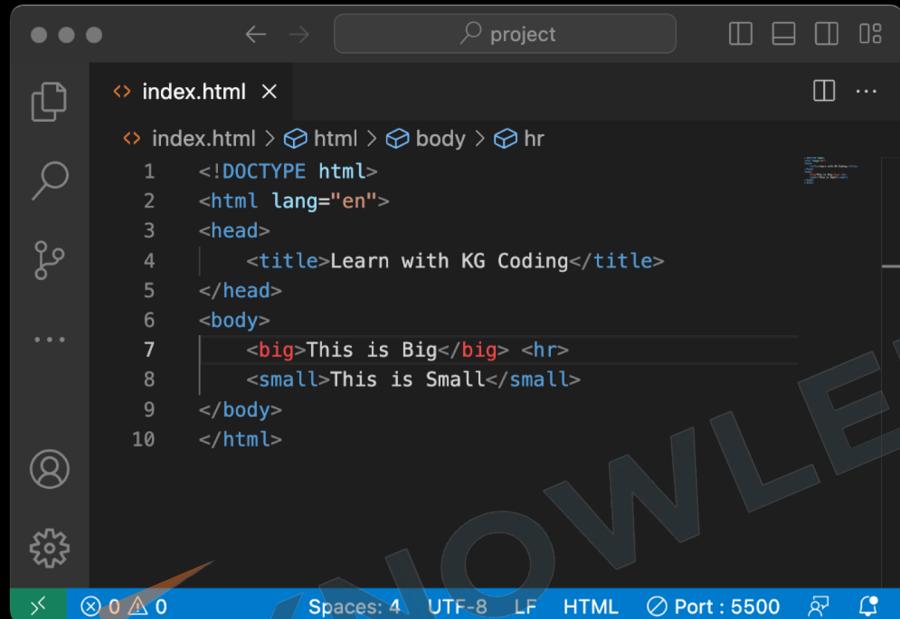


first line more spaces

first line
more spaces

1. Preserves text formatting
2. Maintains whitespace and line breaks
3. Useful for displaying code
4. Enclosed within <pre> and </pre> tags

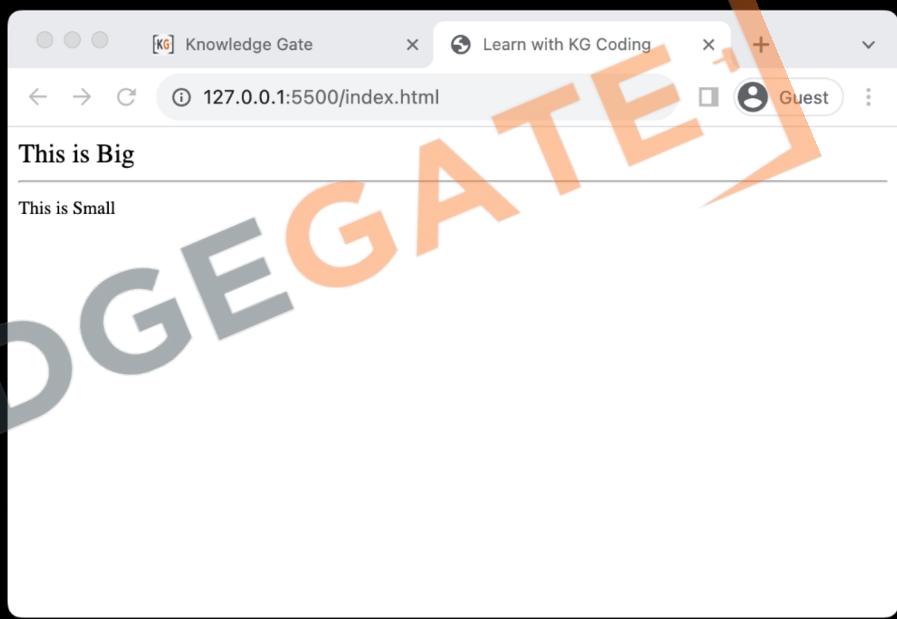
2.9 Big/Small Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

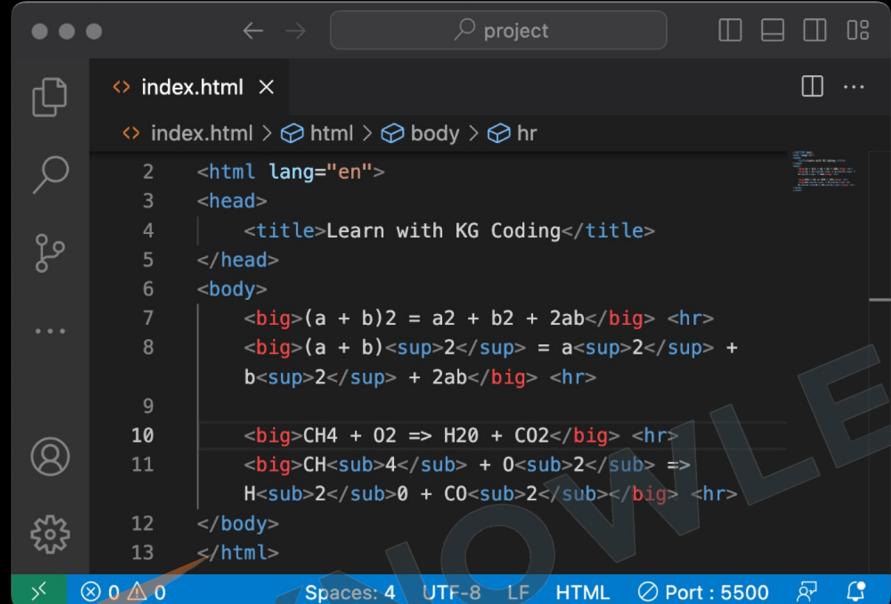
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <big>This is Big</big> <hr>
    <small>This is Small</small>
</body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. A sidebar on the left contains various icons for file operations like copy, paste, search, and user management.



1. **<big>** increases text size
2. **<small>** decreases text size
3. Less common due to CSS alternatives

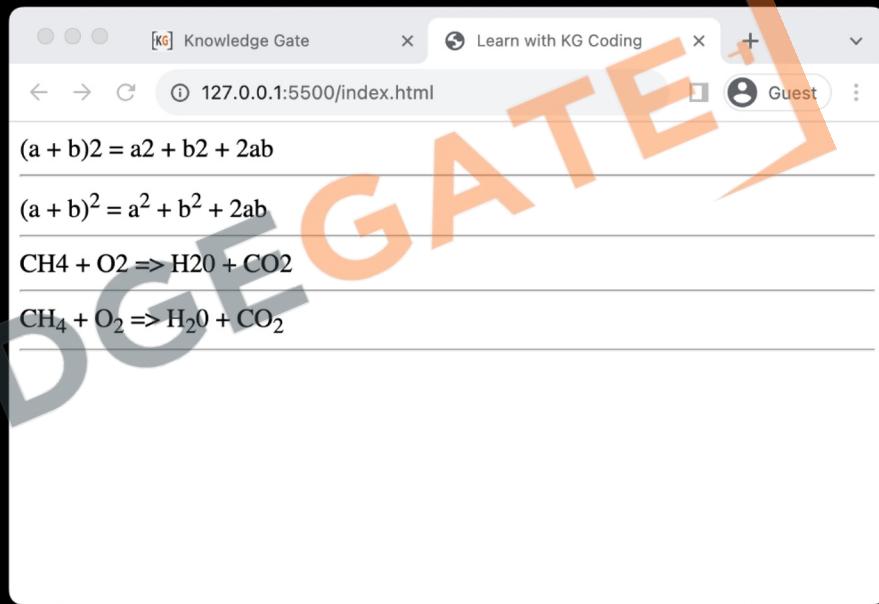
2.10 Superscript/Subscript Tag



A screenshot of a code editor showing the file index.html. The code contains mathematical equations using the ^{and _{tags. The first equation is $(a + b)^2 = a^2 + b^2 + 2ab$. The second equation is $(a + b)^2 = a^2 + b^2 + 2ab$. Below these are two more equations: $CH_4 + O_2 \Rightarrow H_2O + CO_2$ and $CH_4 + O_2 \Rightarrow H_2O + CO_2$. The code editor has a dark theme with icons on the left.}}

```
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <big>(a + b)2 = a2 + b2 + 2ab</big> <hr>
    <big>(a + b)<sup>2</sup> = a<sup>2</sup> + b<sup>2</sup> + 2ab</big> <hr>

    <big>CH4 + O2 => H2O + CO2</big> <hr>
    <big>CH<sub>4</sub> + O<sub>2</sub> => H<sub>2</sub>O + CO<sub>2</sub></big> <hr>
</body>
</html>
```



A screenshot of a web browser displaying the rendered content of index.html. The equations are displayed correctly with superscripts and subscripts. A large orange watermark reading "KNOWLEDGE GATE" is overlaid on the page.

$(a + b)^2 = a^2 + b^2 + 2ab$
 $(a + b)^2 = a^2 + b^2 + 2ab$
 $CH_4 + O_2 \Rightarrow H_2O + CO_2$
 $CH_4 + O_2 \Rightarrow H_2O + CO_2$

1. `<sup>` makes text superscript
2. `<sub>` makes text subscript
3. Used for mathematical equations, footnotes
4. Does not change font size, just position

Level 2

Must-Use HTML Tags

3. Character Entity Reference

Reference

3.1 Character Entity Reference

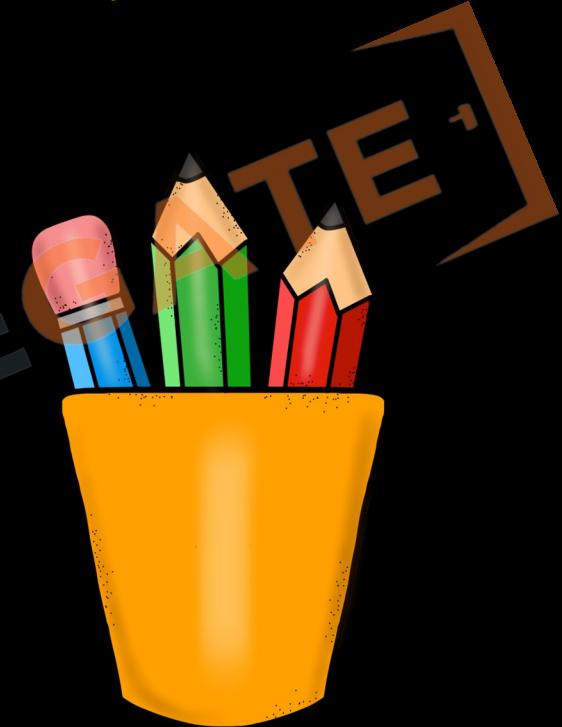
1. Used to display reserved or special characters
2. Syntax often starts with & and ends with ; (e.g., & for &)

 	-	–	-	−	°	°	Δ	Δ	α	α	
€	€	—	—	±	±	°	º	Λ	Λ	β	β
ƒ	¢	…	…	√	√	ª	ª	Θ	Θ	γ	γ
£	£	§	§	∞	∞	¹	¹	Ξ	Ξ	δ	δ
¥	¥	¶	¶	∞	∝	²	²	Π	Π	ε	ε
¤	¤	†	†	×	×	³	³	Σ	Σ	ζ	ζ
f	ƒ	‡	‡	÷	÷	¼	¼	Φ	Φ	η	η
©	©		¡	~	∼	½	½	Ψ	Ψ	θ	θ
®	®	¿	¿	≈	≈	¾	¾	Ω	Ω	ι	ι
™	™	%	‰	≡	≅	∴	∴	∇	∇	κ	κ

Level 2 Revision

Must-Use HTML Tags

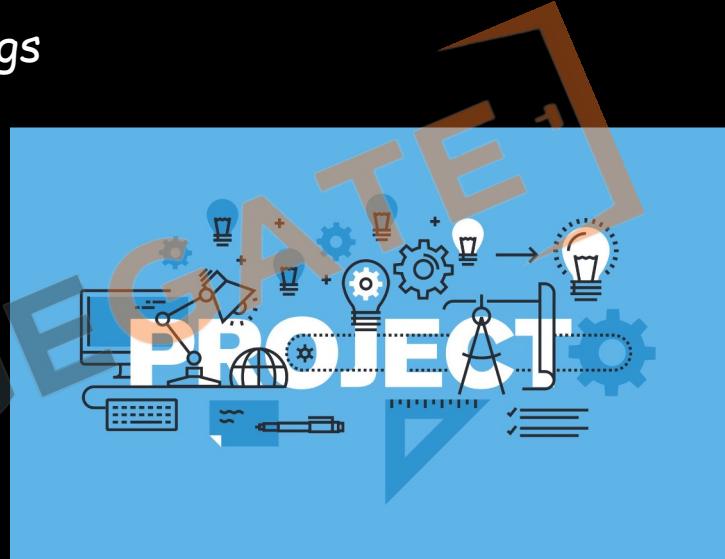
1. HTML Attributes
 1. What are HTML Attributes
 2. Id Property
2. HTML Tags
 1. Heading Tag
 2. Paragraph Tag
 3.
 <HR> tags
 4. Image Tag
 5. Video Tag
 6. Anchor Tag
 7. Bold / Italic / Underline / Strikethrough
 8. Pre Tag
 9. Big / Small Tag
 10. Superscript / Subscript
3. Character Entity Reference
 1. What are Character Entity References



Project Level 2

Must-Use HTML Tags

1. Create a page with **heading**, **paragraph**, **line breaks** and **separators**.
2. Use an **image** with height 300, which is a **link** to another page.
3. Use **bold**, **italic**, **underline** and **strike through** in one line.
4. Write third equation of motion using **superscript** and **subscript**.



Level 3

Browser Tools

1. Browser Tools
 1. View Page Source
 2. Inspect Element
 3. HTML without CSS
2. Responsive Design
 1. Different screen size
3. Live Edit Code
 1. Live edit HTML
 2. Live edit CSS
 3. Live edit JS
 4. Changes only happening at client
4. Validating Web pages
 1. Using validator.w3.org

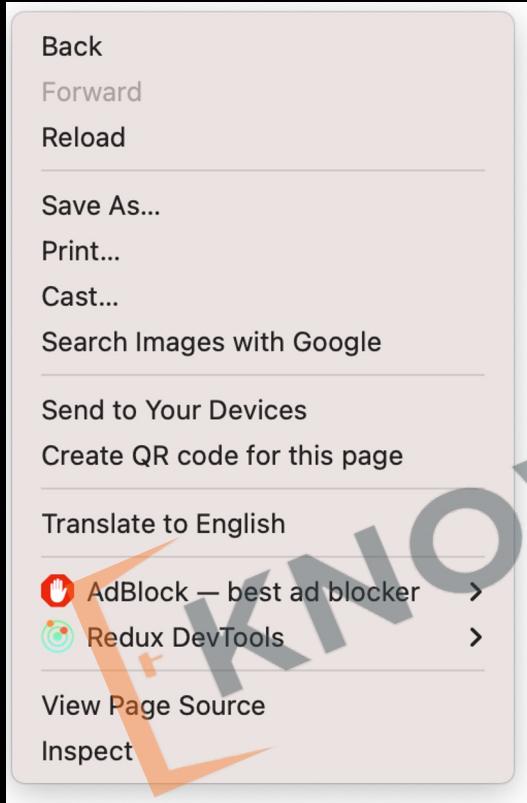
Level 3

Browser Tools



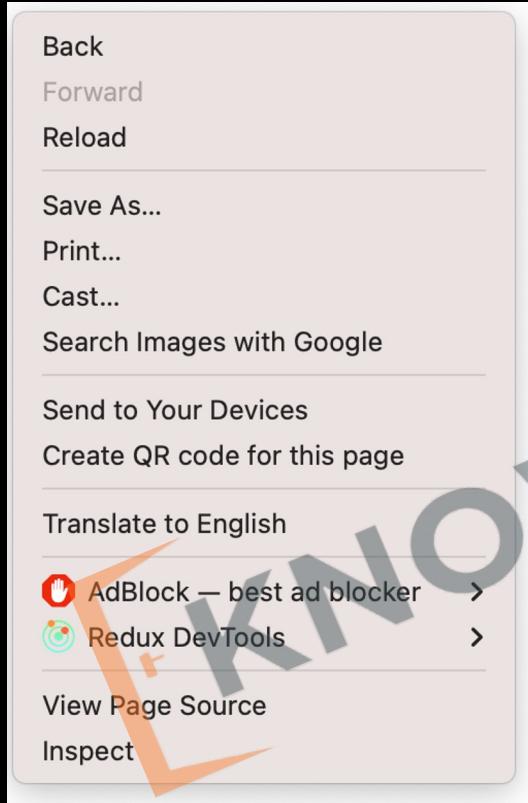
1. Browser Tools

1.1 View Page Source



1. Displays raw HTML and CSS
2. Useful for debugging and learning
3. Shows external files like JavaScript links

1.2 Inspect Element



1. Allows real-time editing of HTML/CSS
2. Useful for debugging and testing
3. Shows element hierarchy and layout
4. Includes console for JavaScript
5. Highlights selected elements on page

Level 3

Browser Tools



KNOWLEDGE AGATE¹
**Responsive
Design**²

2.1 Different Screen Sizes



1. Adapts layout for different screen sizes
2. Flexible layouts
3. Optimizes images and assets
4. Enhances user experience on mobile and desktop

Level 3

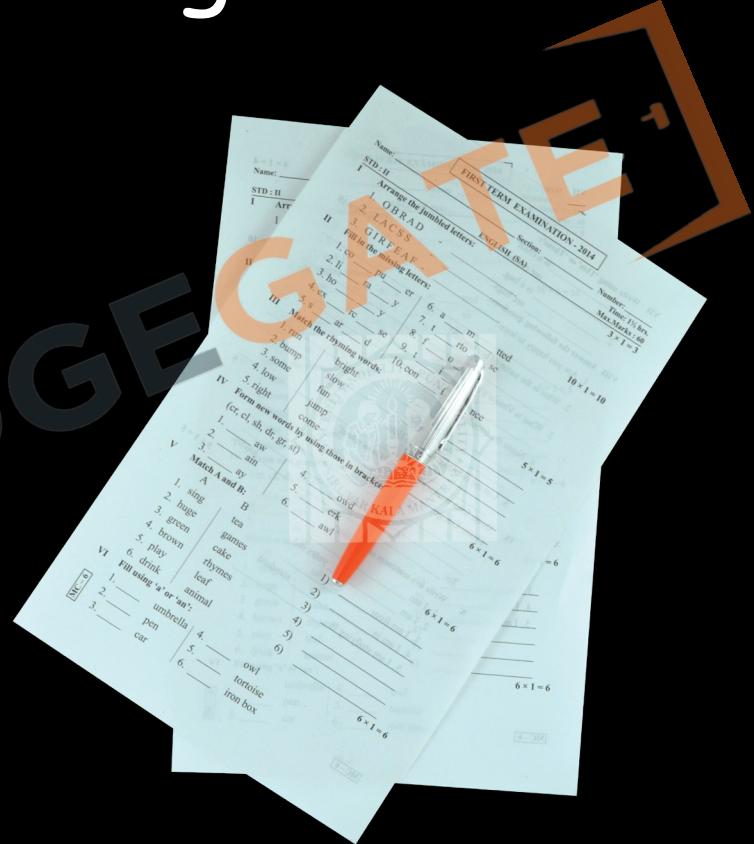
Browser Tools



3.GATE¹
Live Edit
Code

3.4 Changes happening at Client

1. Changes made are temporary
2. Affect only the current session
3. Not saved to the server
4. Reset upon page reload
5. Useful for testing, not permanent fixes



Like: If you change the question in your question paper that has no effect on actual exam.

Level 3

Browser Tools



4. **Validating**
WebPages

KNOWLEDGE CATE 1

4.1 Using validator.w3.org

Nu Html Checker

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

Showing results for contents of text-input area

Checker Input

Show source outline image report Options...

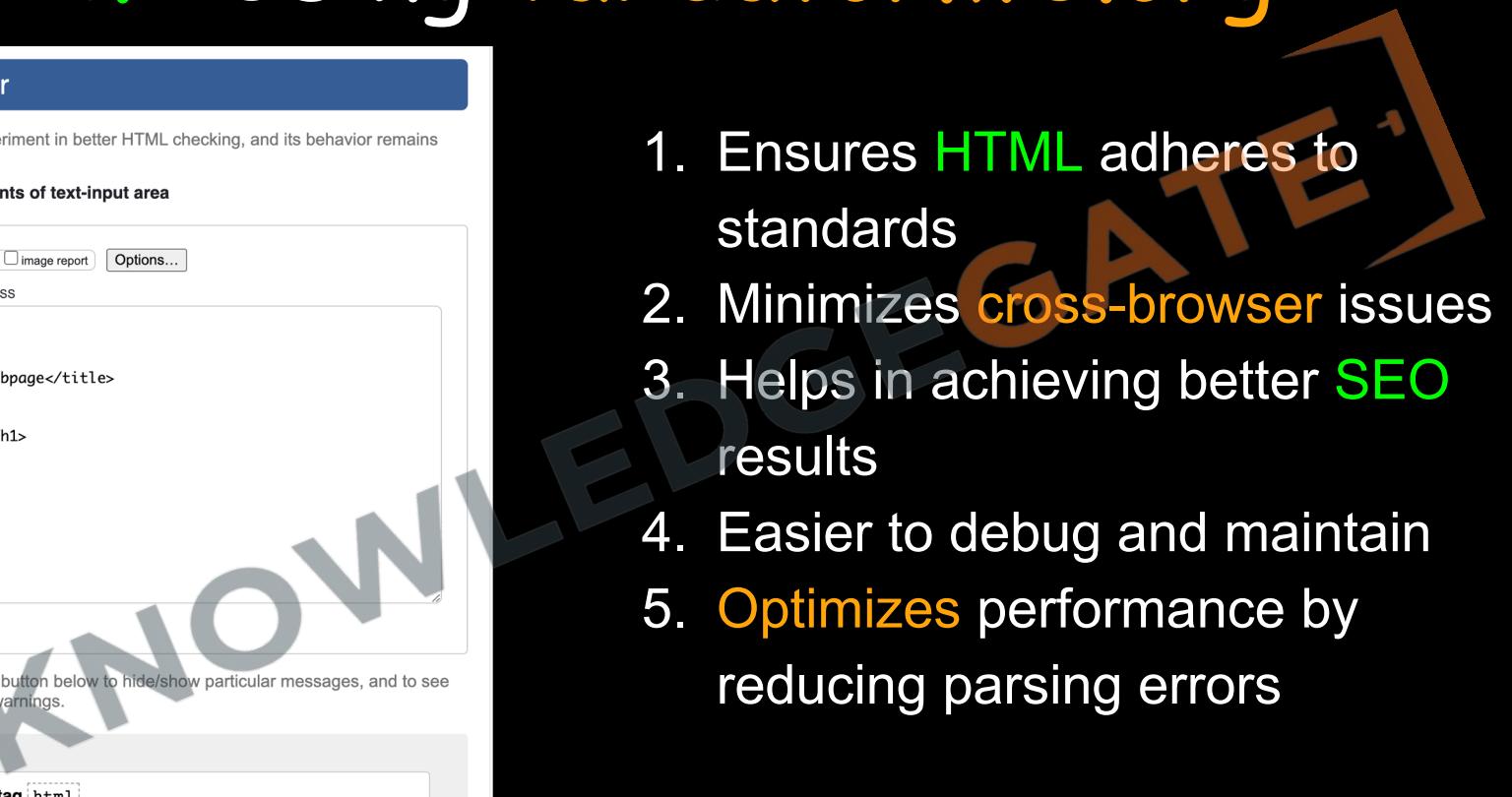
Check by css

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>My First Webpage</title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
<html>
```

Use the Message Filtering button below to hide/show particular messages, and to see total counts of errors and warnings.

Message Filtering

1. Error Stray start tag `html`.
From line 9, column 1; to line 9, column 6
`></body><html>`

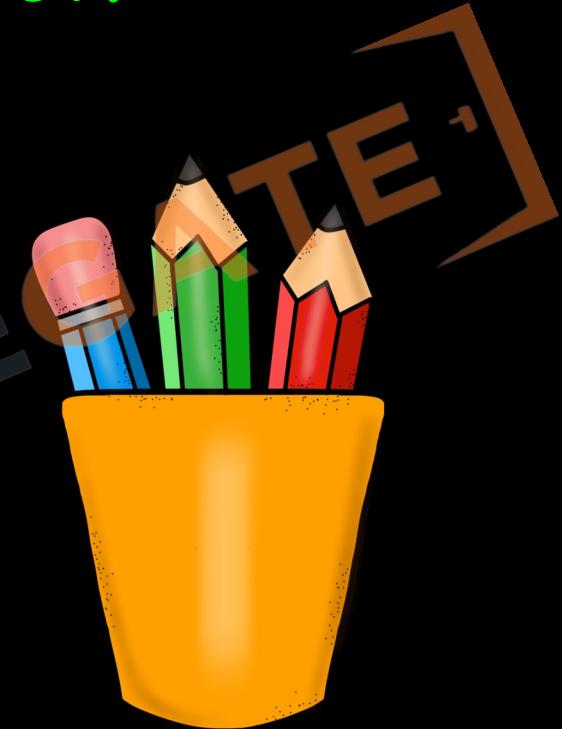


1. Ensures **HTML** adheres to standards
2. Minimizes **cross-browser** issues
3. Helps in achieving better **SEO** results
4. Easier to debug and maintain
5. **Optimizes** performance by reducing parsing errors

Level 3 Revision

Browser Tools

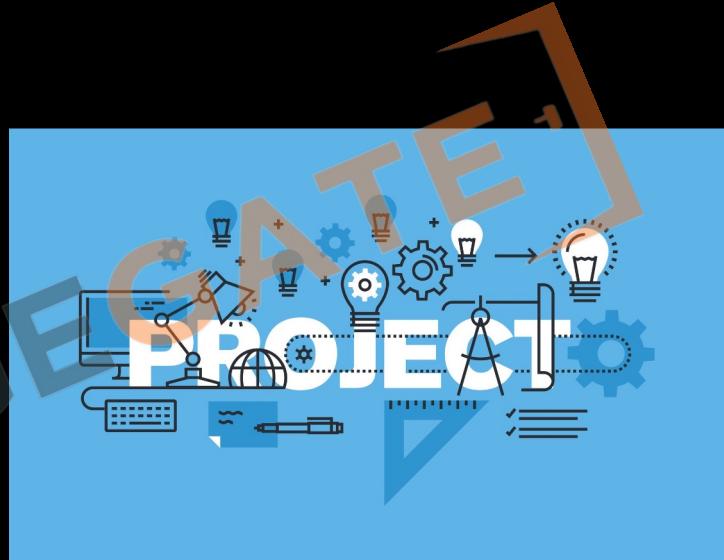
1. Browser Tools
 1. View Page Source
 2. Inspect Element
 3. HTML without CSS
2. Responsive Design
 1. Different screen size
3. Live Edit Code
 1. Live edit HTML
 2. Live edit CSS
 3. Live edit JS
 4. Changes only happening at client
4. Validating Web pages
 1. Using validator.w3.org



Project Level 3

Browser Tools

1. Save Source of **Instagram** in a file and check the render.
2. **Inspect** the likes element on the page and read the code to understand.
3. Change number of likes on Your **Instagram** post
4. **Validate** the page we created in last project.



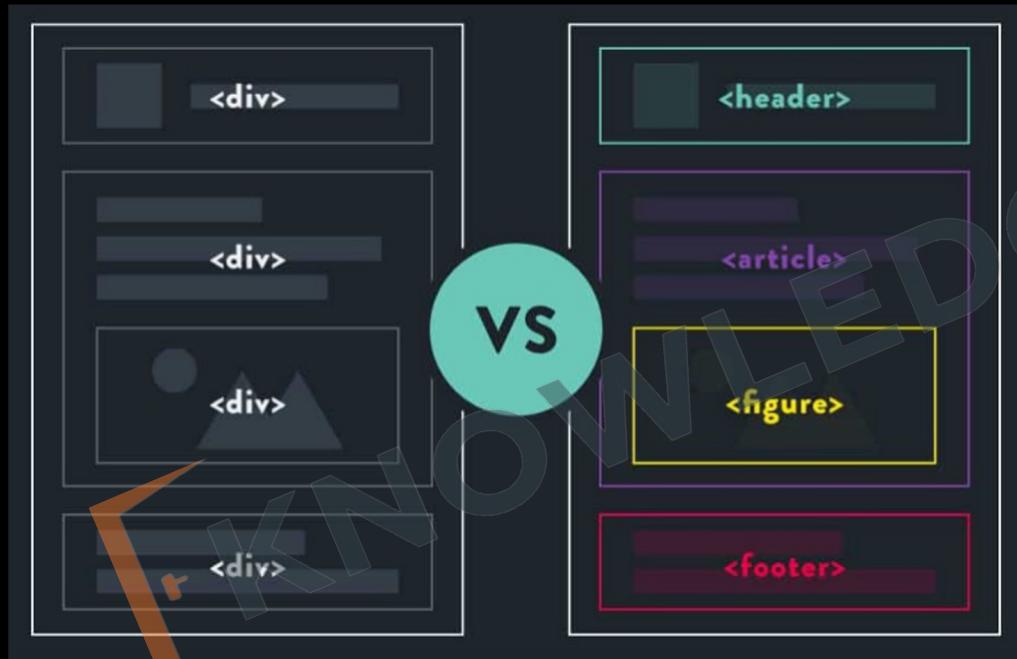
Level 4

HTML and Project Structure

1. Semantic Tags
 1. Semantic / Non-Semantic Tags
2. Body Tags
 1. Header Tag
 2. Main Tag
 1. Section Tag
 2. Article Tag
 3. Aside Tag
 3. Footer Tag
3. Folder Structure
 1. Recommended Folder structure
4. More Tags
 1. Navigation tags
 2. Block / Inline Elements
 3. Div tags
 4. Span Tags

Level 4

HTML and Project Structure



1. Semantic Tags

1.1 Semantic/Non-Semantic Tags

Semantic Tags

- Meaningful: Describe content.
- SEO: Good for search engines.
- Accessibility: Useful for screen readers.
- Examples: `<header>`, `<footer>`, `<article>`, `<section>`, `<nav>`.

Non-Semantic Tags

- Generic: No specific meaning.
- For Styling: Used for layout.
- No SEO: Not SEO-friendly.
- Examples: `<div>`, ``, `<i>`, ``.

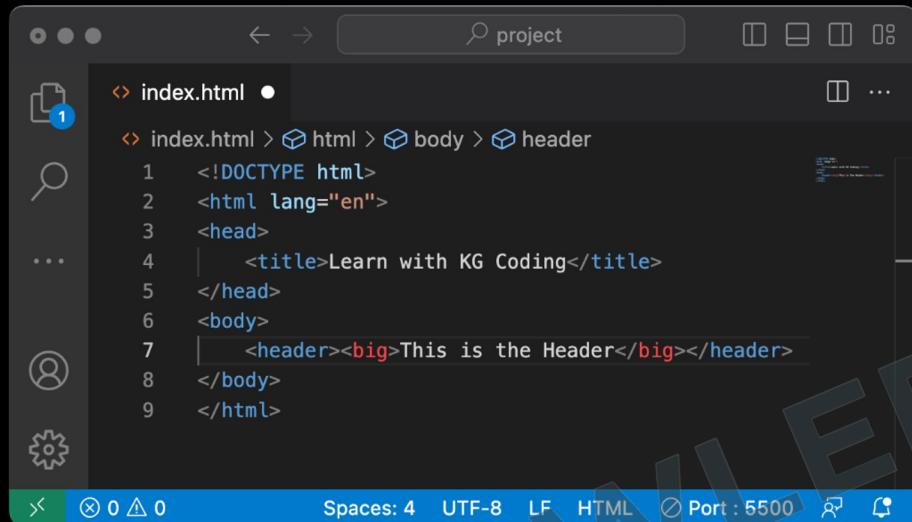
Level 4

HTML and Project Structure



2. Body Tags

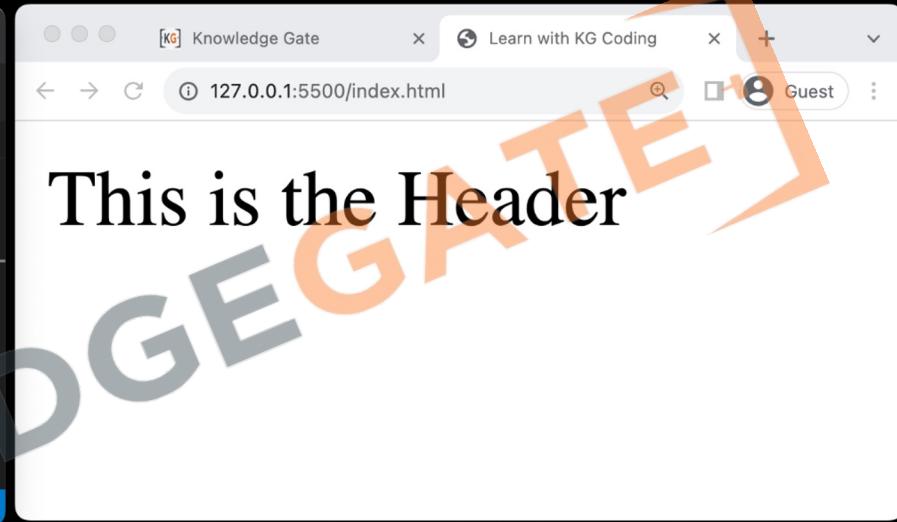
2.1 Header Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <header><big>This is the Header</big></header>
</body>
</html>
```

The code editor has a dark theme with light-colored text. It includes icons for file operations, search, and help. The status bar at the bottom shows "Spaces: 4", "UTF-8", "LF", "HTML", "Port : 5500", and other small icons.



1. **Purpose:** Used to contain introductory content or navigation links.
2. **Semantic:** It's a semantic tag, providing meaning to the enclosed content.
3. **Location:** Commonly found at the top of web pages, but can also appear within `<article>` or `<section>` tags.
4. **Multiple Instances:** Can be used more than once on a page within different sections.

2.2 Main Tag

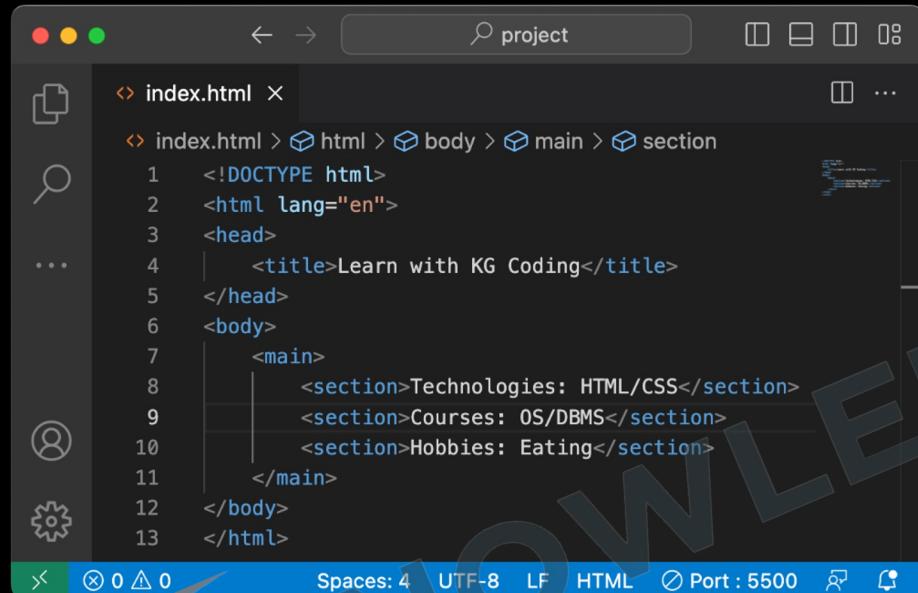
The image shows a code editor on the left and a browser window on the right. The code editor displays the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <header><big>This is the Header</big></header>
    <hr><main>This is main space</main>
</body>
</html>
```

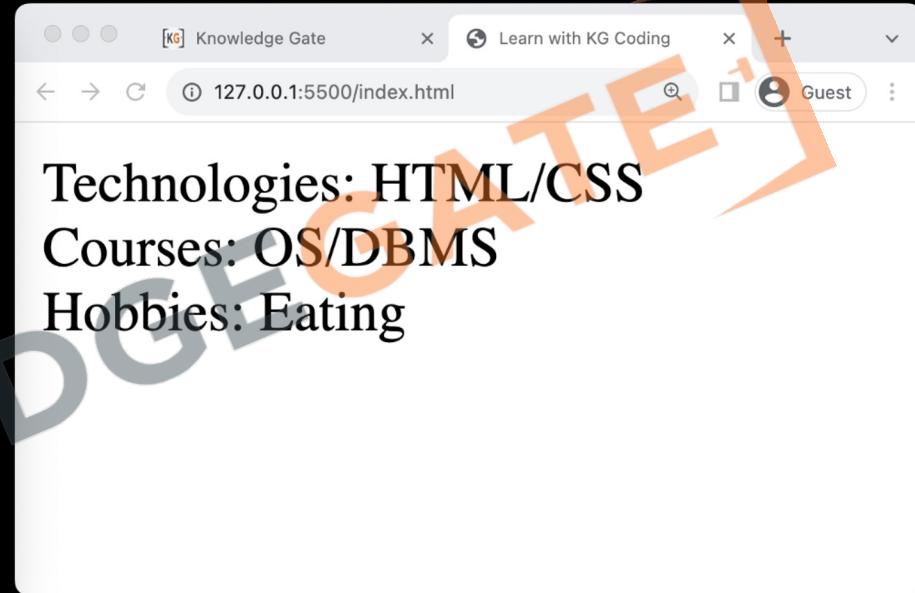
The browser window shows the rendered HTML. It has a header with the text "This is the Header" and a main content area below it with the text "This is main space". A large watermark reading "KNOWLEDGE GATE" is diagonally across the center.

1. **Purpose:** Encloses the primary content of a webpage.
2. **Semantic:** Adds meaning, indicating the main content area.
3. **Unique:** Should appear only once per page.
4. **Accessibility:** Helps screen readers identify key content.
5. **Not for Sidebars:** Excludes content repeated across multiple pages like site navigation or footer.

2.2.1 Section Tag



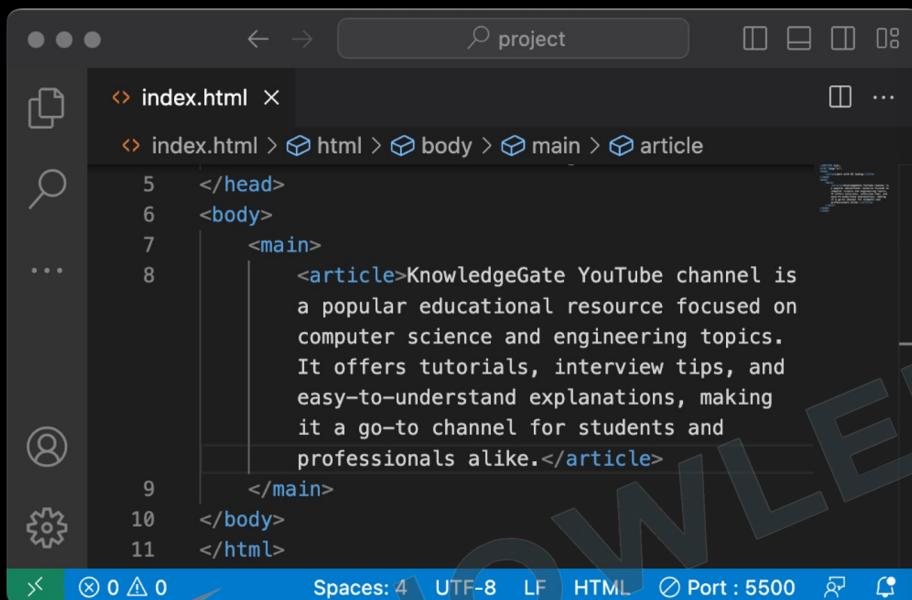
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <main>
        <section>Technologies: HTML/CSS</section>
        <section>Courses: OS/DBMS</section>
        <section>Hobbies: Eating</section>
    </main>
</body>
</html>
```



Technologies: HTML/CSS
Courses: OS/DBMS
Hobbies: Eating

1. **Purpose:** Groups related content in a distinct section.
2. **Semantic:** Adds structure and meaning.
3. **Headers:** Often used with a heading `<h1>` to `<h6>` to indicate section topic.
4. **Nested:** Can be nested within other `<section>` or `<article>` tags.

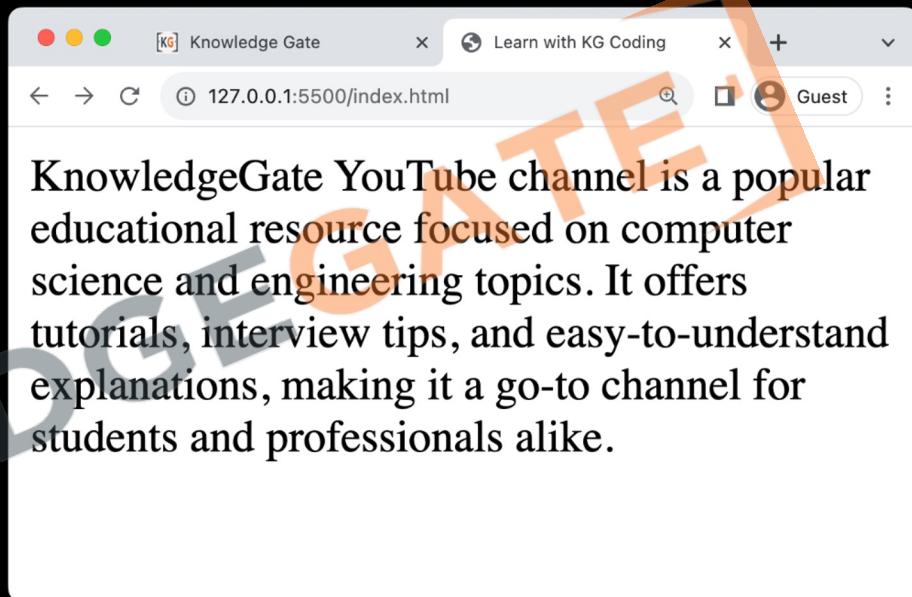
2.2.2 Article Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

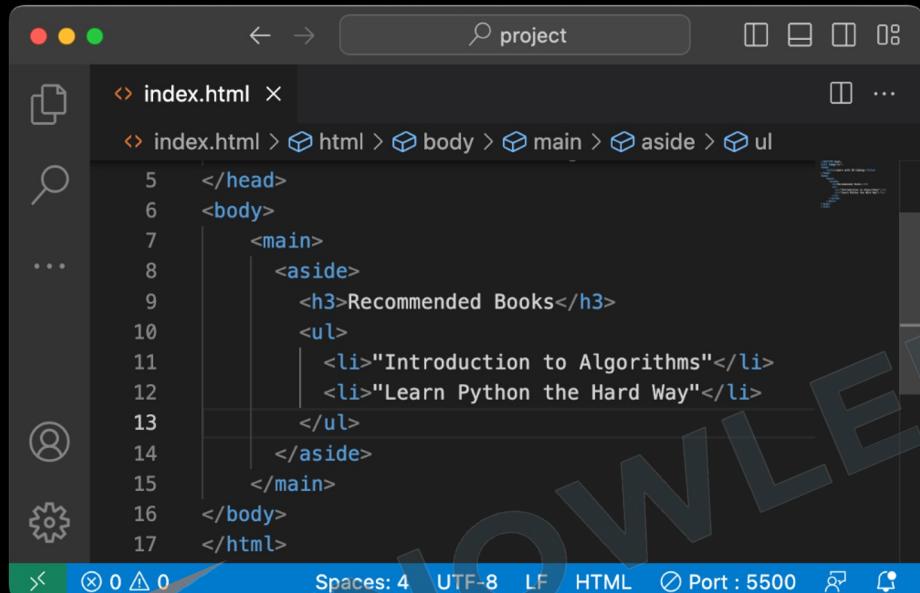
```
5  </head>
6  <body>
7      <main>
8          <article>KnowledgeGate YouTube channel is
9              a popular educational resource focused on
10             computer science and engineering topics.
11             It offers tutorials, interview tips, and
12             easy-to-understand explanations, making
13             it a go-to channel for students and
14             professionals alike.</article>
15     </main>
16  </body>
17 </html>
```

The code editor has a dark theme with light-colored syntax highlighting. The status bar at the bottom shows "Spaces: 4", "UTF-8", "LF", "HTML", "Port : 5500", and a refresh icon.



1. Purpose: Encloses content that stands alone, like a **blog post or news story**.
2. Semantic: Provides contextual meaning.
3. Independence: Content should make sense even if taken out of the page context.
4. Multiple Instances: Can be used multiple times on the same page

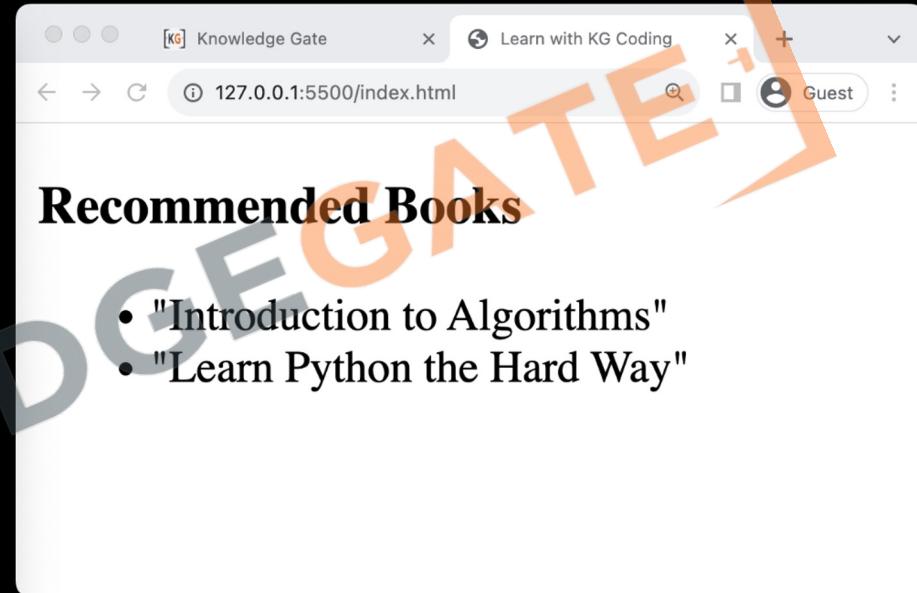
2.2.3 Aside Tag



A screenshot of a code editor showing the file `index.html`. The code is as follows:

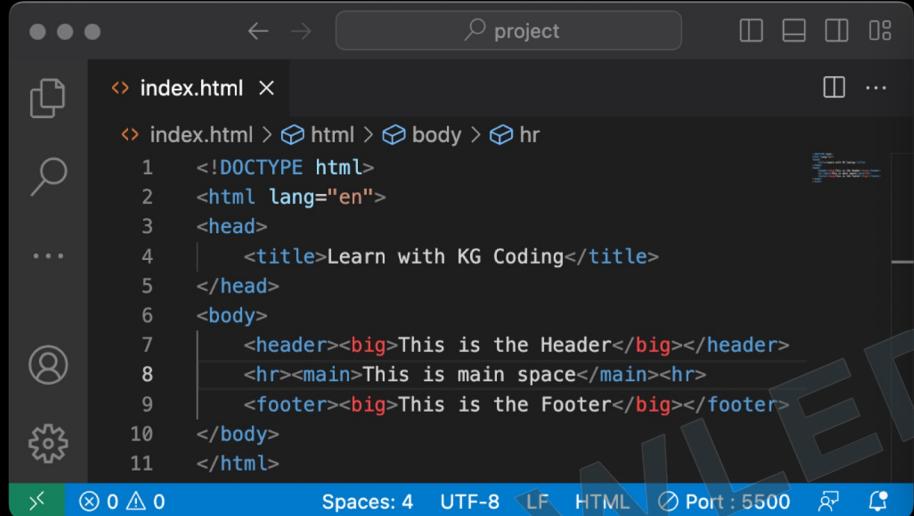
```
</head>
<body>
  <main>
    <aside>
      <h3>Recommended Books</h3>
      <ul>
        <li>"Introduction to Algorithms"</li>
        <li>"Learn Python the Hard Way"</li>
      </ul>
    </aside>
  </main>
</body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. The sidebar shows the file structure: `index.html` > `html` > `body` > `main` > `aside` > `ul`.



1. **Purpose:** Contains sidebar or supplementary content.
2. **Semantic:** Indicates content tangentially related to the main content.
3. **Not Crucial:** Content is not essential to understanding the main content.
4. **Examples:** Could hold **widgets**, quotes, or ads.

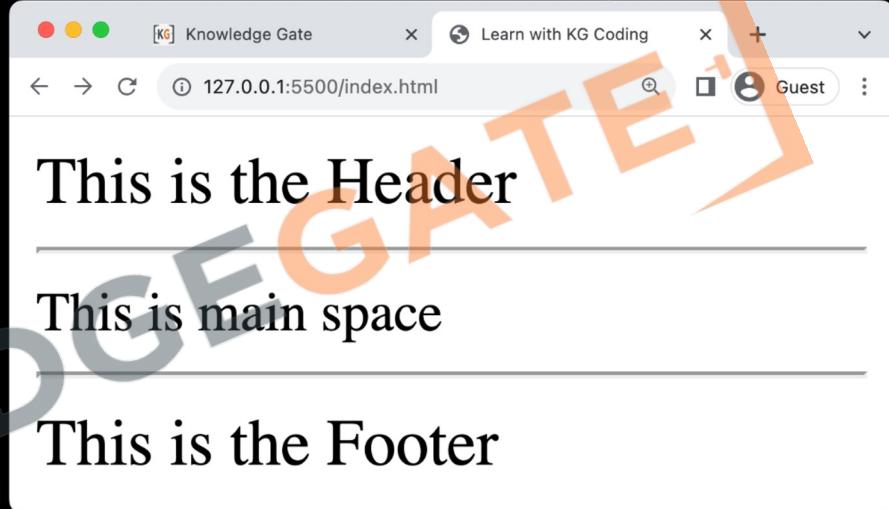
2.3 Footer Tag



A screenshot of a code editor showing the file `index.html`. The code is structured as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Learn with KG Coding</title>
  </head>
  <body>
    <header><big>This is the Header</big></header>
    <hr><main>This is main space</main><hr>
    <footer><big>This is the Footer</big></footer>
  </body>
</html>
```

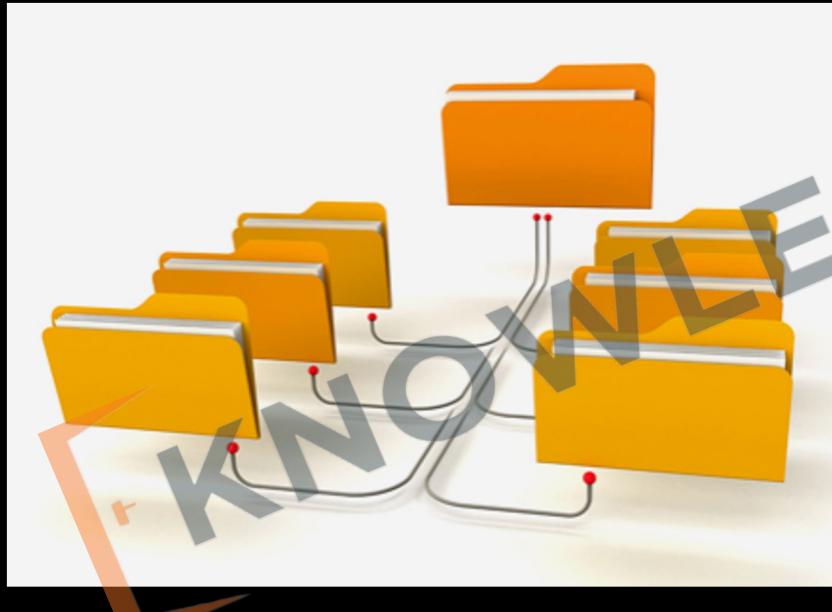
The code editor interface includes a sidebar with icons for file, search, and user, and a bottom bar with tabs for Spaces: 4, UTF-8, LF, HTML, Port: 5500, and a refresh icon.



1. **Purpose:** For footer content like extra info or links.
2. **Semantic:** Provides meaning to enclosed content.
3. **Location:** Typically at the **bottom** of pages or sections.
4. **Content:** Includes copyrights, contact info, and social links.
5. **Multiple Instances:** Can be used more than once on a page.

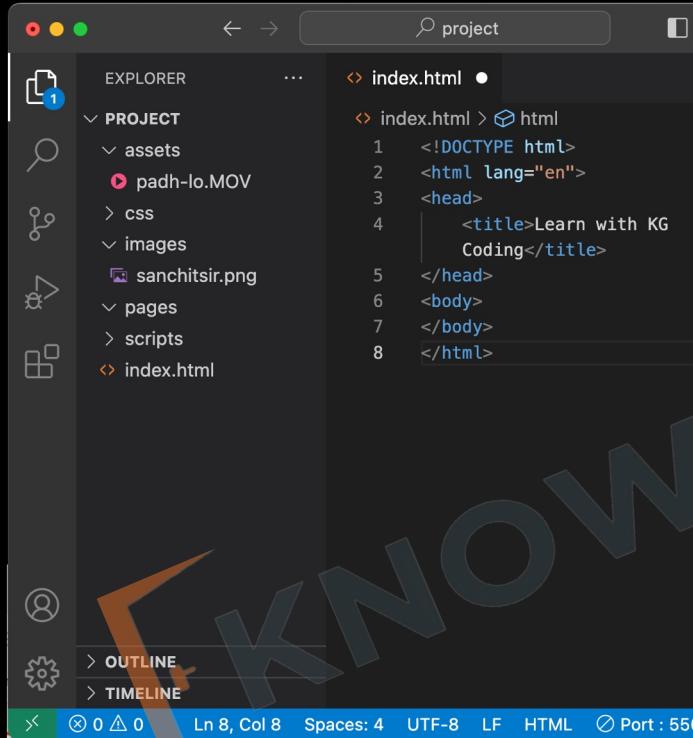
Level 4

HTML and Project Structure



3. Folder
Structure

3.1 Recommended Folder Structure



1. **Root Directory:** Main folder containing all website files.
2. **HTML Files:** Store main .html files at the root level for easy access.
3. **CSS Folder:** Create a css/ folder for all Cascading Style Sheets.
4. **JS Folder:** Use a scripts/ folder for JavaScript files.
5. **Images Folder:** Store images in an images/ or images/ folder.
6. **Assets:** Other assets like fonts can go in an assets/ folder.
7. **Sub-directories:** For multi-page websites, use sub-folders to categorize content.

Level 4

HTML and Project Structure



KNOWLEDGE GATE¹
4. More
Tags

4.1 Navigation Tags

The image shows a code editor on the left and a web browser on the right. The code editor displays the file 'index.html' with the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <nav>
        <a href="#home">Home</a><br>
        <a href="#about">About</a><br>
        <a href="#services">Services</a><br>
        <a href="#contact">Contact</a><br>
    </nav>
</body>
</html>
```

The browser window on the right shows the rendered HTML with four underlined links: Home, About, Services, and Contact. A large, diagonal watermark reading "KNOWLEDGE GATE" is overlaid across both screens.

1. **Purpose:** Encloses navigation links or menus.
2. **Semantic:** Signals that the content is meant for navigating the site.
3. **Common Content:** Usually contains lists ``, `` of links `<a>`.
4. **Accessibility:** Aids screen readers in identifying site navigation.

4.2 Block / Inline Elements

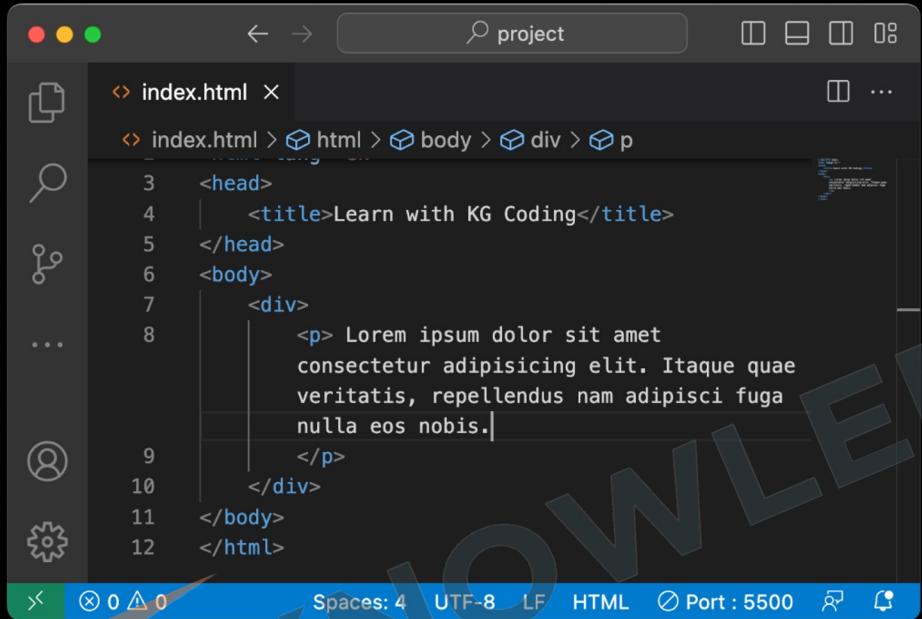
Block Elements

- **New Line:** Start on a new line.
- **Full Width:** Take up all horizontal space.
- **Styling:** Can have margins and padding.
- **Size:** Width and height can be set.
- **Examples:** <div>, <p>, <h1>, , .

Inline Elements

- **Flow:** Stay in line with text.
- **Width:** Just as wide as the content.
- **No Break:** No new line between elements.
- **Limited Styling:** Can't set size easily.
- **Examples:** , <a>, , , .

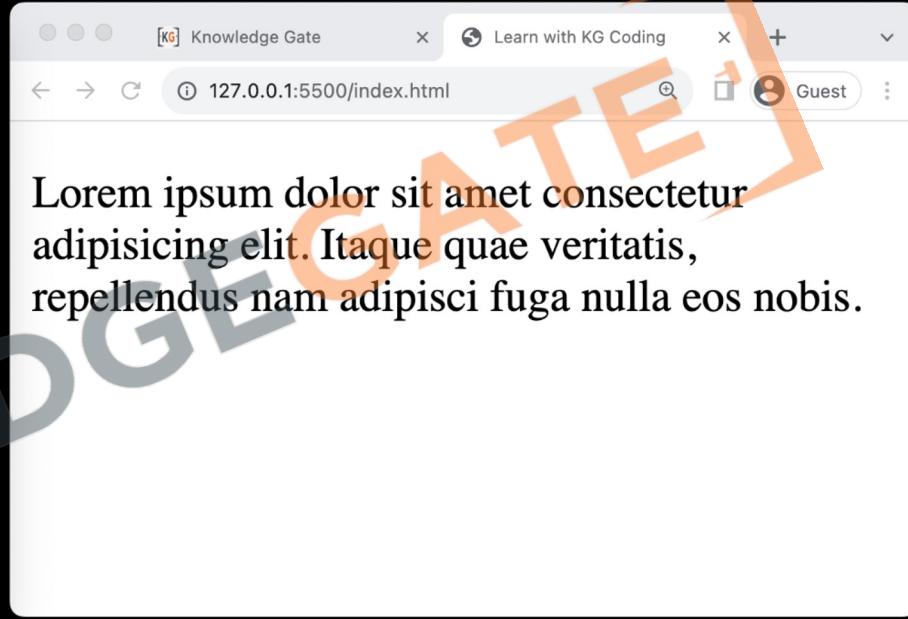
4.3 Div Tags



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

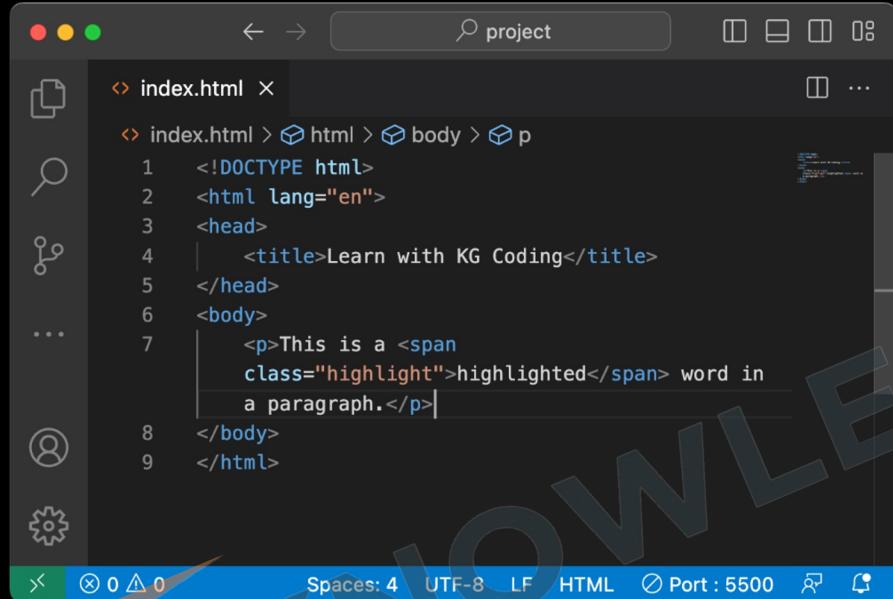
```
<head>
  <title>Learn with KG Coding</title>
</head>
<body>
  <div>
    <p> Lorem ipsum dolor sit amet
    consectetur adipisicing elit. Itaque quae veritatis,
    repellendus nam adipisci fuga
    nulla eos nobis.</p>
  </div>
</body>
</html>
```

The code editor has a dark theme with orange icons. A sidebar on the left contains icons for file, search, and settings. The bottom status bar shows "Spaces: 4", "UTF-8", "LF", "HTML", "Port : 5500", and a refresh icon.



1. **Purpose:** Acts as a container for other **HTML** elements.
2. **Non-Semantic:** Doesn't provide inherent meaning to enclosed content.
3. **Styling:** Commonly used for layout and styling via **CSS**.
4. **Flexibility:** Highly versatile and can be customized using classes or IDs.

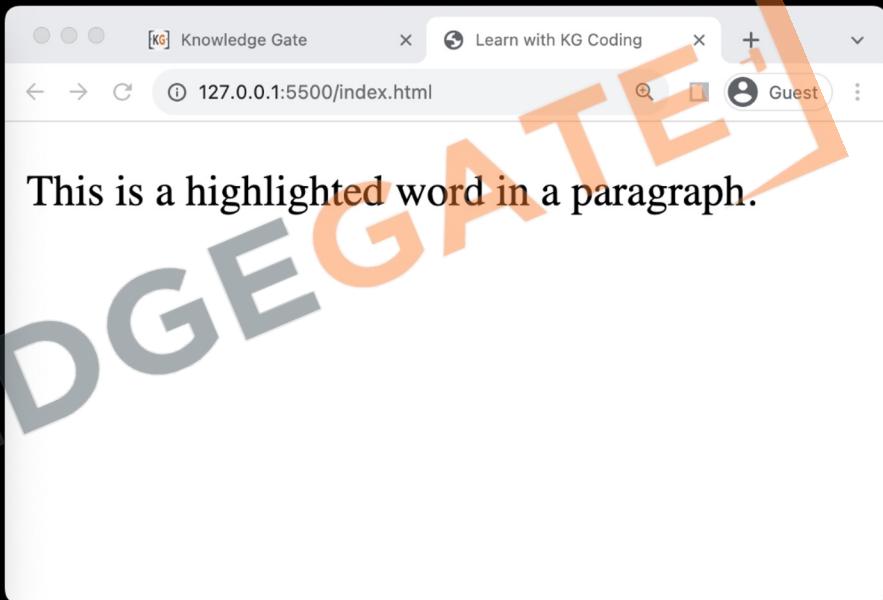
4.4 Span Tags



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <p>This is a <span class="highlight">highlighted</span> word in a paragraph.</p>
</body>
</html>
```

The word "highlighted" is highlighted with a blue background and white text. The status bar at the bottom shows "Spaces: 4", "UTF-8", "LF", "HTML", and "Port : 5500".



1. **Purpose:** Used for inline elements to style or manipulate a portion of text.
2. **Non-Semantic:** Doesn't add specific meaning to the enclosed text.
3. **Styling:** Commonly used for changing color, font, or adding effects via CSS.
4. **Inline Nature:** Doesn't break text flow or create a new block-level element.

Level 4 Revision

HTML and Project Structure

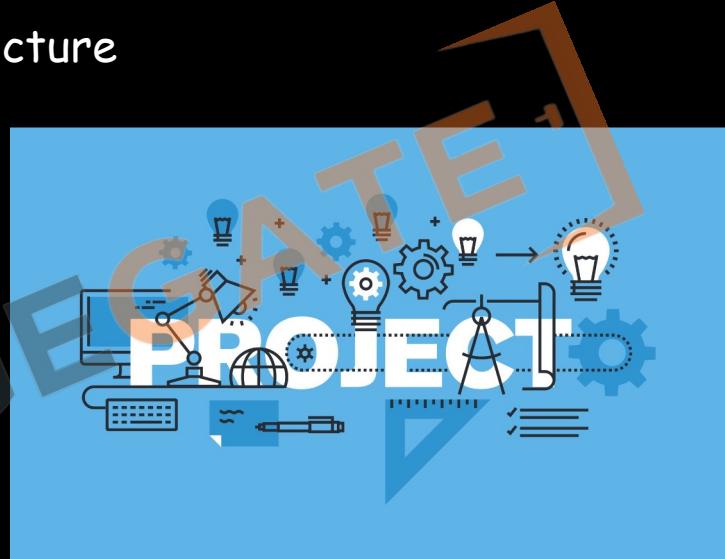
1. Semantic Tags
 1. Semantic / Non-Semantic Tags
2. Body Tags
 1. Header Tag
 2. Main Tag
 1. Section Tag
 2. Article Tag
 3. Aside Tag
 3. Footer Tag
3. Folder Structure
 1. Recommended Folder structure
4. More Tags
 1. Navigation tags
 2. Block / Inline Elements
 3. Div tags
 4. Span Tags



Project Level 4

HTML and Project Structure

1. Create a **page** with header, footer, main(section, article, aside tag).
2. Make sure the project from level 3 has correct **folder** structure.
3. Create **groupings** of multiple tags using div.
4. Create **navigation** to important sections of your page.



Level 5

List, Tables & Forms

1. List Tag
 1. Ordered Lists
 2. Types of Ordered Lists
 3. Unordered Lists
2. Table Tag
 1. `<tr>`, `<td>`, `<th>` tags
 2. Captions
 3. Col spans
3. Forms
 1. Input tag
 2. Action Attributes
 3. Name and Value Property
 4. Label Tag
 5. Exploring Types
4. iFrame Tag
 1. Using iFrames

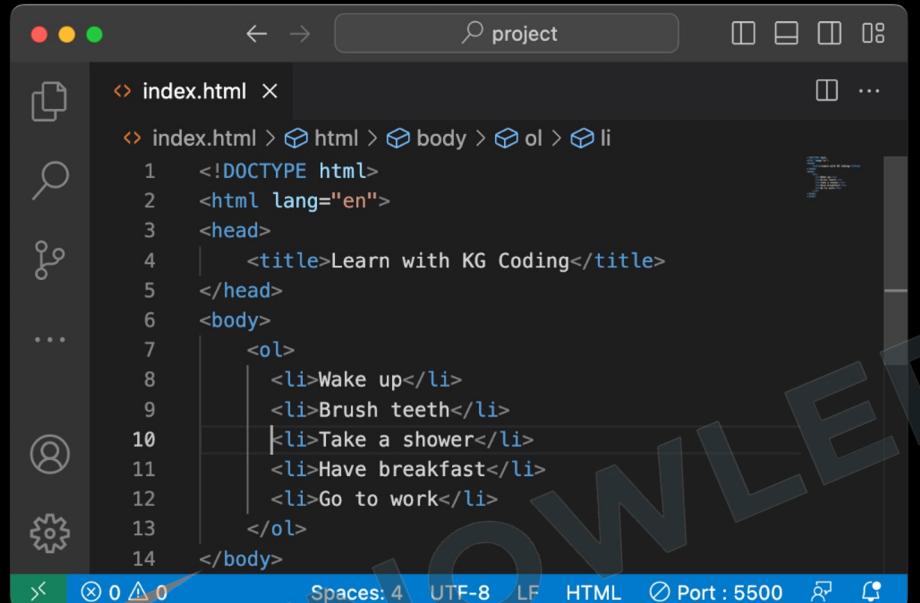
Level 5

List, Tables & Forms



EDGE GATE 1. List Tag

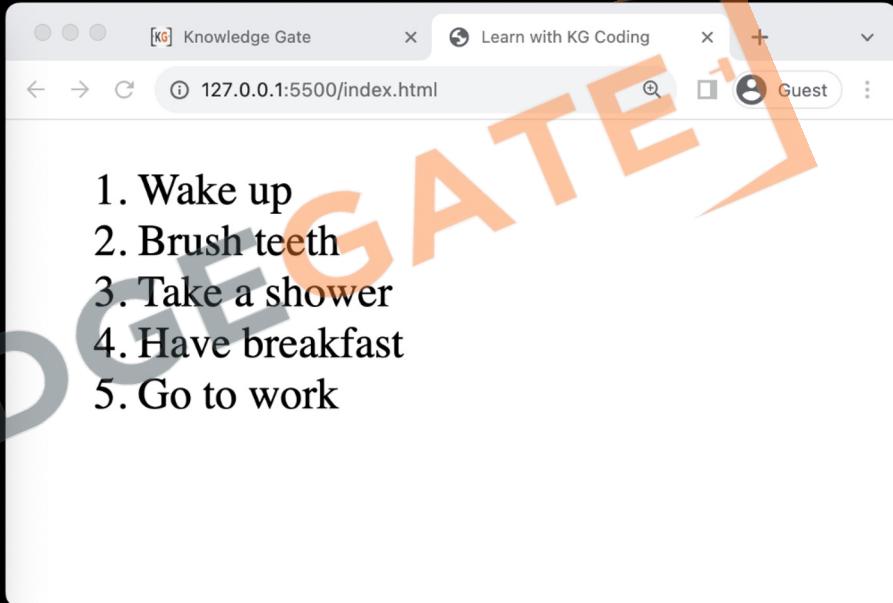
1.1 Ordered Lists



A screenshot of a code editor showing the file `index.html`. The code contains an ordered list:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <ol>
        <li>Wake up</li>
        <li>Brush teeth</li>
        <li>Take a shower</li>
        <li>Have breakfast</li>
        <li>Go to work</li>
    </ol>
</body>
```

The code editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom indicating `Spaces: 4`, `UTF-8`, `HTML`, and `Port : 5500`.



1. Purpose: Used for creating lists with items that have a specific order.
2. Default: Items are automatically numbered.
3. Nesting: Can be nested within other lists.

1.2 Types of Ordered Lists

Ordered Lists

- **Numeric:** Default type, (1, 2, 3, ...)
Attribute: `type="1"`
- **Uppercase Letters:** (A, B, C, ...)
Attribute: `type="A"`
- **Lowercase Letters:** (a, b, c, ...)
Attribute: `type="a"`
- **Uppercase Roman:** (I, II, III, ...)
Attribute: `type="I"`
- **Lowercase Roman:** (i, ii, iii, ...)
Attribute: `type="i"`

A. Apple
B. Banana
C. Cherry
D. Dragonfruit

a. Apple
b. Banana
c. Cherry
d. Dragonfruit

I. Apple
II. Banana
III. Cherry
IV. Dragonfruit

i. Apple
ii. Banana
iii. Cherry
iv. Dragonfruit

1.3 Unordered Lists

The image shows a code editor on the left and a browser window on the right. The code editor displays the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <ul>
        <li>Apple</li>
        <li>Banana</li>
        <li>Cherry</li>
        <li>Dragonfruit</li>
    </ul>
</body>
</html>
```

The browser window shows the rendered output of the code:

- Apple
- Banana
- Cherry
- Dragonfruit

A large, semi-transparent watermark reading "KNOWLEDGE GATE" is overlaid across both screens.

1. **Purpose:** Used for lists where the order of items doesn't matter.
2. **Default:** Items are usually bulleted.
3. **Nesting:** Can be nested within other lists.

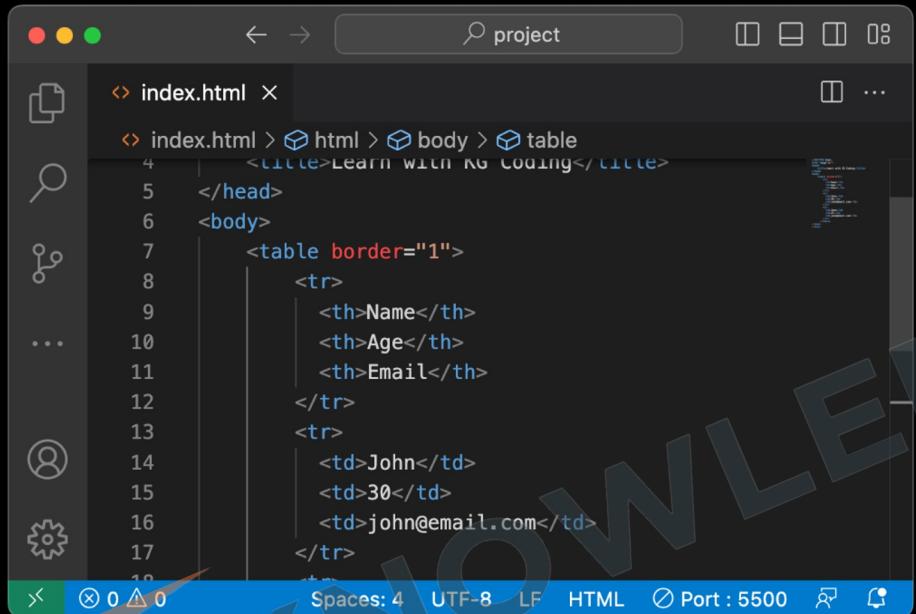
Level 5

List, Tables & Forms



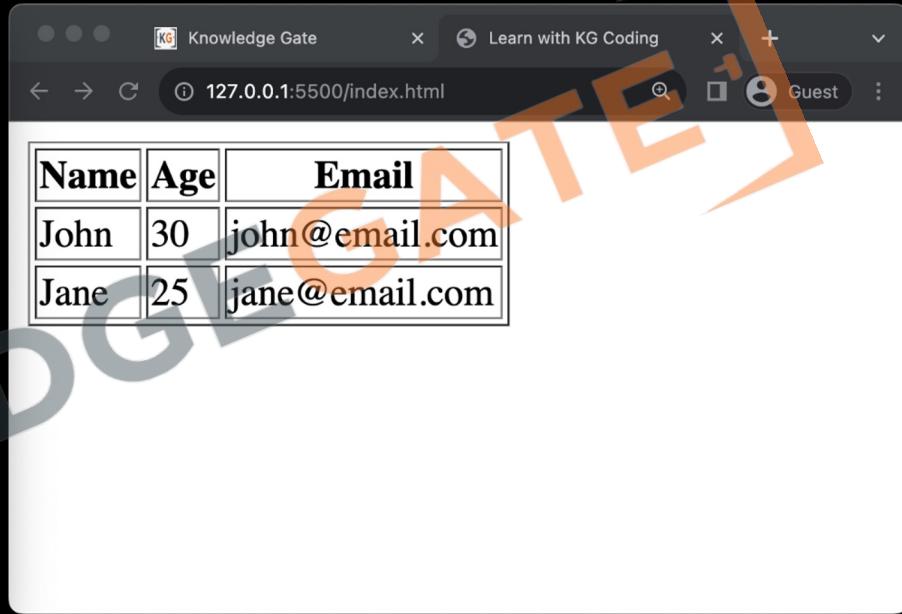
NON-EDGE GATE 1
Table Tag

2.1 <tr>, <td>, <th> Tags



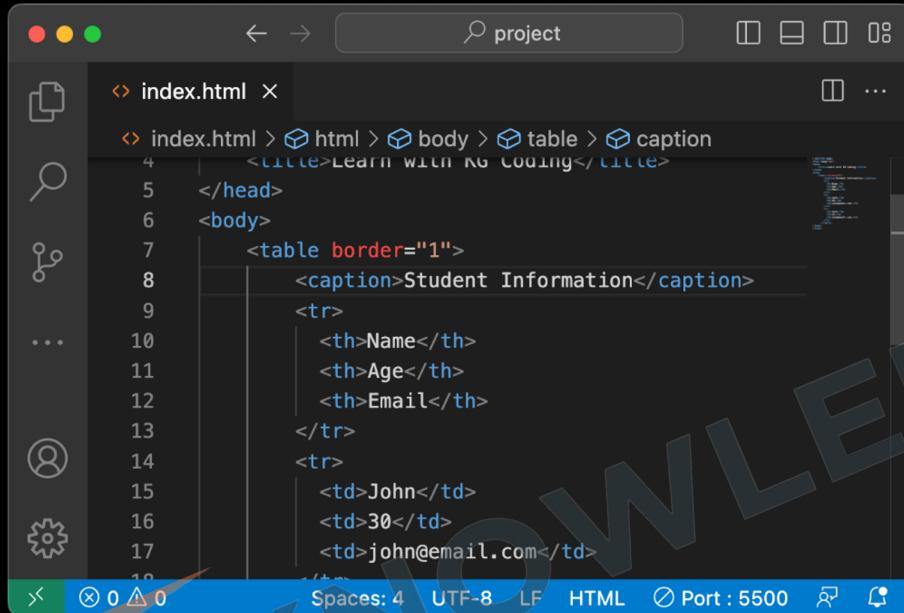
The screenshot shows a code editor window with the file "index.html" open. The code displays a simple HTML table structure with three rows. The first row contains three header cells ("Name", "Age", "Email") defined by tags. The second row contains three data cells ("John", "30", "john@email.com") defined by tags. The third row contains three data cells ("Jane", "25", "jane@email.com") defined by tags. The code is written in a dark-themed code editor. | | |

```
<html>
  <head>
    <title>Learn with KG Coding</title>
  </head>
  <body>
    <table border="1">
      <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Email</th>
      </tr>
      <tr>
        <td>John</td>
        <td>30</td>
        <td>john@email.com</td>
      </tr>
      <tr>
        <td>Jane</td>
        <td>25</td>
        <td>jane@email.com</td>
      </tr>
    </table>
  </body>
</html>
```



1. **<tr> Table Row** : Used to define a row in an HTML table.
2. **<th> Table Header** : Used for header cells within a row.
Text is bold and centered by default.
3. **<td> Table Data** : This Holds the actual data.

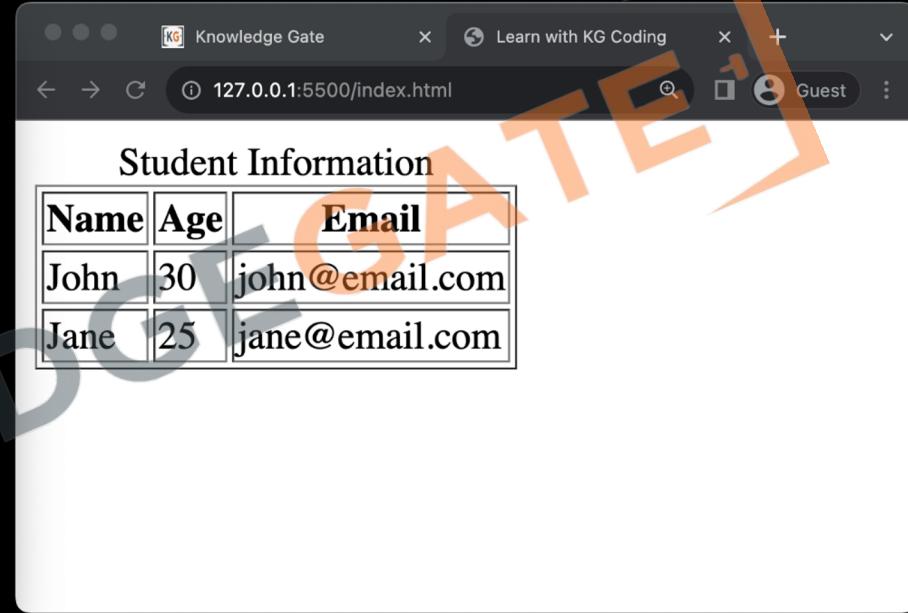
2.2 Captions



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following HTML code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Learn HTML Using Coding</title>
  </head>
  <body>
    <table border="1">
      <caption>Student Information</caption>
      <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Email</th>
      </tr>
      <tr>
        <td>John</td>
        <td>30</td>
        <td>john@email.com</td>
      </tr>
    </table>
  </body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. The status bar at the bottom shows "Spaces: 4", "UTF-8", "LF", "HTML", "Port: 5500", and other icons.



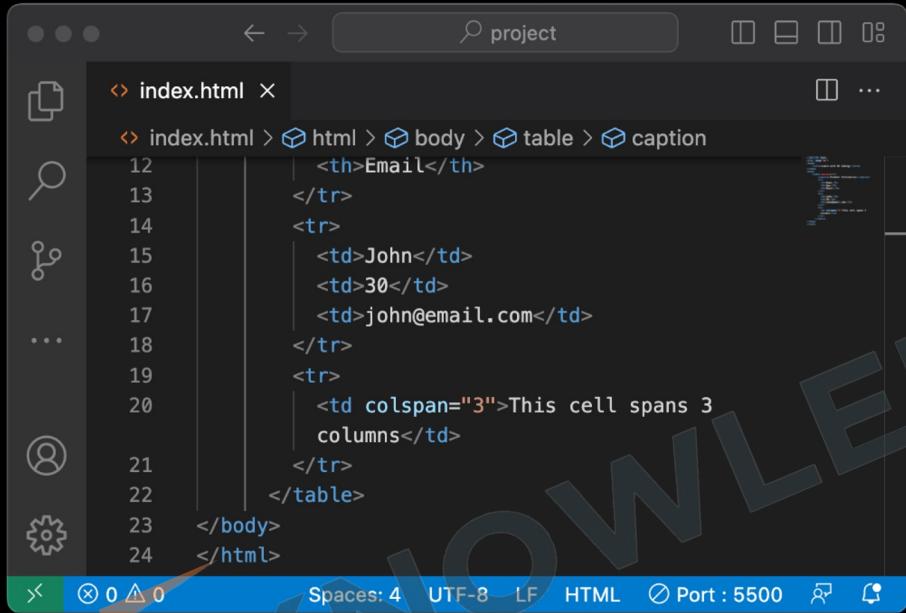
A screenshot of a web browser window titled "Knowledge Gate" with the URL "127.0.0.1:5500/index.html". The page displays a table with the following data:

Name	Age	Email
John	30	john@email.com
Jane	25	jane@email.com

The browser interface includes a search bar, a refresh button, and a tab labeled "Learn with KG Coding". A watermark reading "KNOWLEDGE GATE" is diagonally across the page.

1. **Purpose:** Provides a title or description for a table.
2. **Placement:** Must be inserted **immediately after** the `<table>` opening tag.
3. **Alignment:** **Centered** above the table by default.
4. **Accessibility:** Helps screen readers understand the table's purpose.

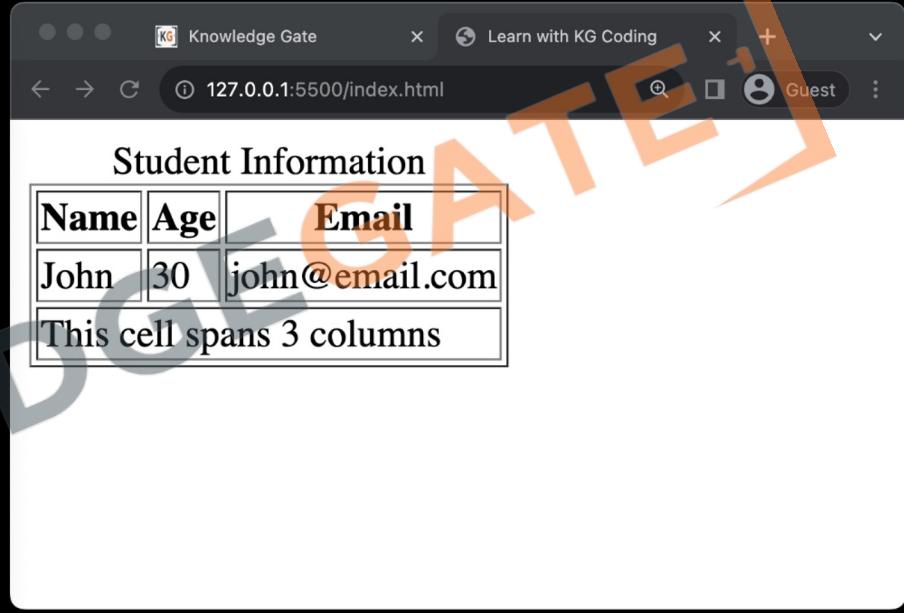
2.3 Col Spans



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following HTML code:

```
<html>
  <body>
    <table>
      <caption>Student Information</caption>
      <thead>
        <tr>
          <th>Name</th>
          <th>Age</th>
          <th>Email</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>John</td>
          <td>30</td>
          <td>john@email.com</td>
        </tr>
        <tr>
          <td colspan="3">This cell spans 3 columns</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. The bottom status bar shows "Spaces: 4", "UTF-8", "LF", "HTML", "Port : 5500", and a refresh icon.



A screenshot of a web browser window titled "Knowledge Gate" showing the URL "127.0.0.1:5500/index.html". The page title is "Student Information". The table has three columns: Name, Age, and Email. The first row contains the data "John", "30", and "john@email.com". The second row contains a single cell with the text "This cell spans 3 columns". A large orange watermark reading "KNOWLEDGE GATE" is overlaid across the entire browser window.

Name	Age	Email
John	30	john@email.com
This cell spans 3 columns		

1. **Attribute:** Uses the `colspan` attribute in `<td>` or `<th>` tags.
2. **Purpose:** Allows a cell to **span multiple columns** horizontally.
3. **Alignment:** Takes the space of the **specified number of columns**.
4. **Layout:** Useful for combining cells to create complex table layouts.

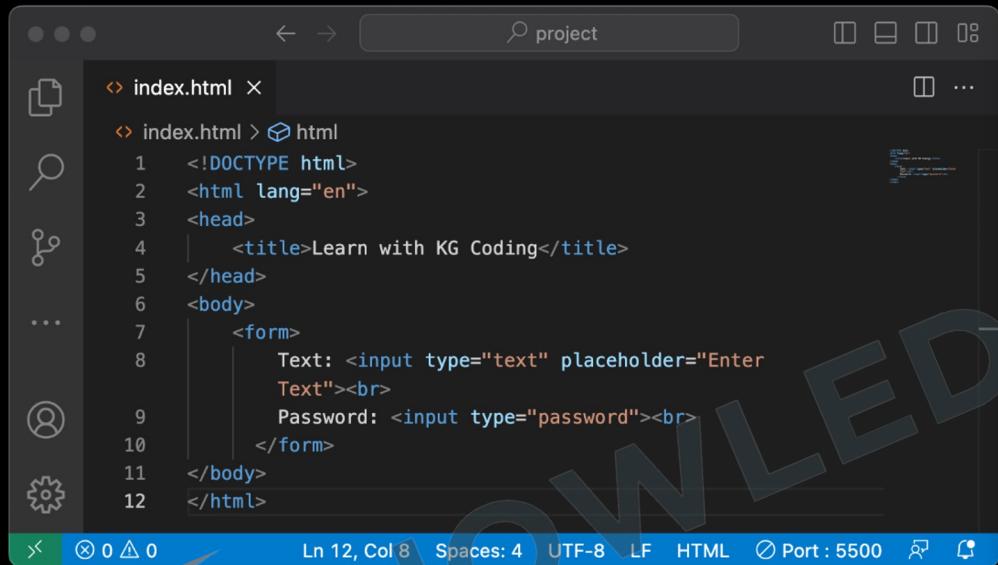
Level 5

List, Tables & Forms



KNOWLEDGE GATE 3 FORMS

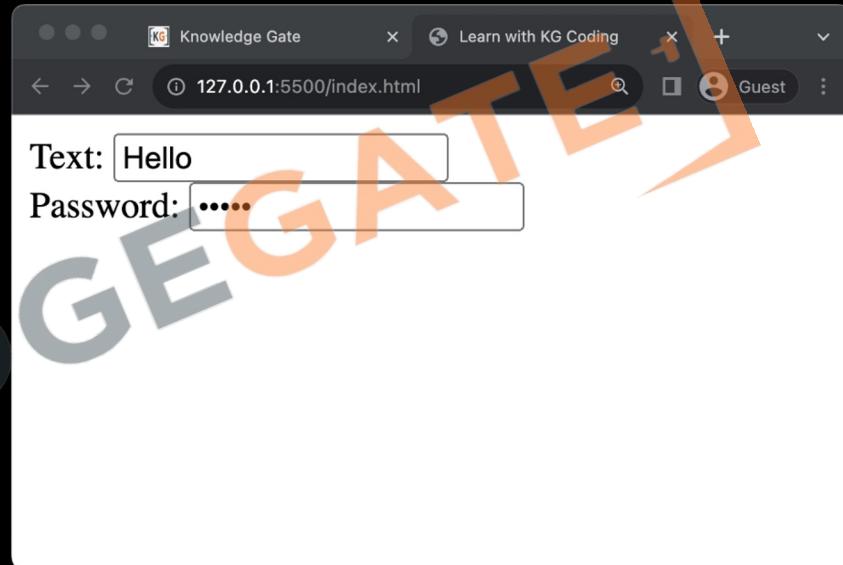
3.1 Input Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, displaying the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
    </form>
</body>
</html>
```

The code editor interface includes a sidebar with icons for file, search, and settings, and a status bar at the bottom showing line 12, column 8, and other file details.



1. **Purpose:** Used within a `<form>` element to collect user input.
2. **Self-Closing:** The `<input>` tag is self-closing; doesn't require a closing tag.
3. **Attributes:** Common attributes are `name`, `value`, `placeholder`, and `required`.

3.2 Action attribute

The image shows a code editor on the left and a browser window on the right. The code editor displays the file 'index.html' with the following content:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form action="/submit.php" method="post">
        <input type="text" name="name">
        <input type="submit" value="Submit">
    </form>
</body>
</html>
```

The browser window shows the rendered HTML with a text input field and a submit button labeled 'Submit'. A large, semi-transparent watermark reading 'KNOWLEDGE GATE' diagonally across the center of the screen.

1. **Purpose:** Specifies the URL to which the form data should be sent when submitted.
2. **Default:** If not specified, the form will be submitted to the current page's URL.
3. **Server-Side:** Usually points to a server-side script (like PHP, Python, etc.) that processes the form data.

3.3 Name and Value property

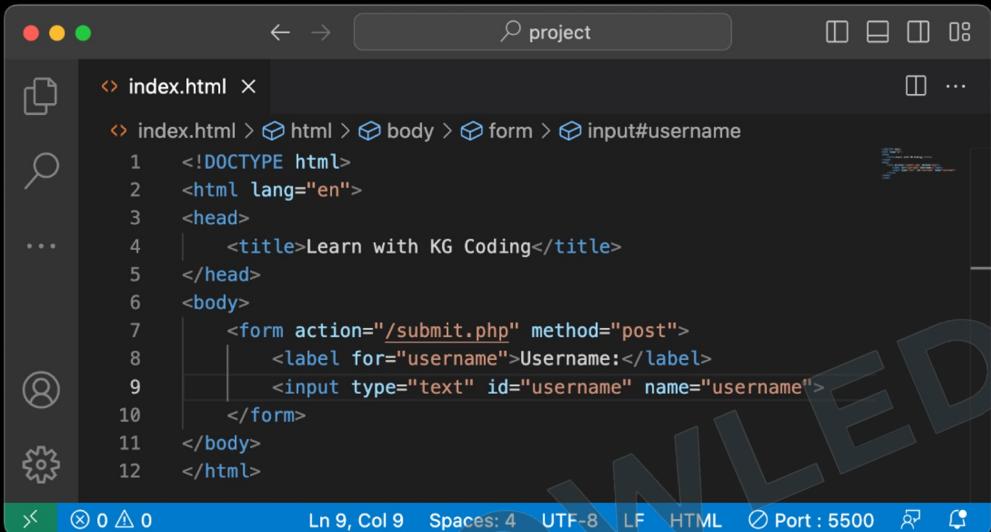
The image shows a code editor on the left and a browser window on the right. The code editor displays the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form action="/submit.php" method="post">
        <input type="text" name="username" value="John">
    </form>
</body>
</html>
```

The browser window shows a single input field containing the text "John". A large watermark reading "KNOWLEDGE GATE" diagonally across the center of the image.

- `name` Property:
 - **ID for Data:** Identifies form elements when submitting.
 - **Unique:** Should be unique to each element for clarity.
- `value` Property:
 - **Default Data:** Sets initial value for input elements.
 - **Sent to Server:** This is the data sent when form is submitted.

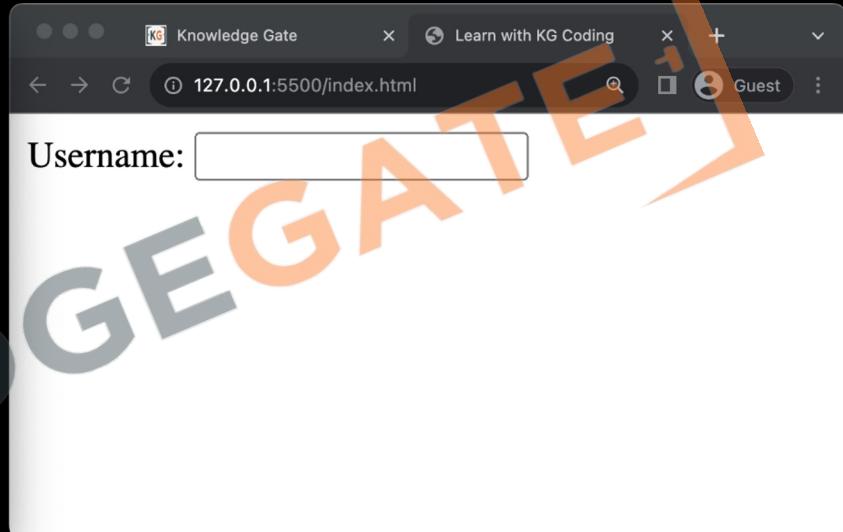
3.4 Label Tag



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following code:

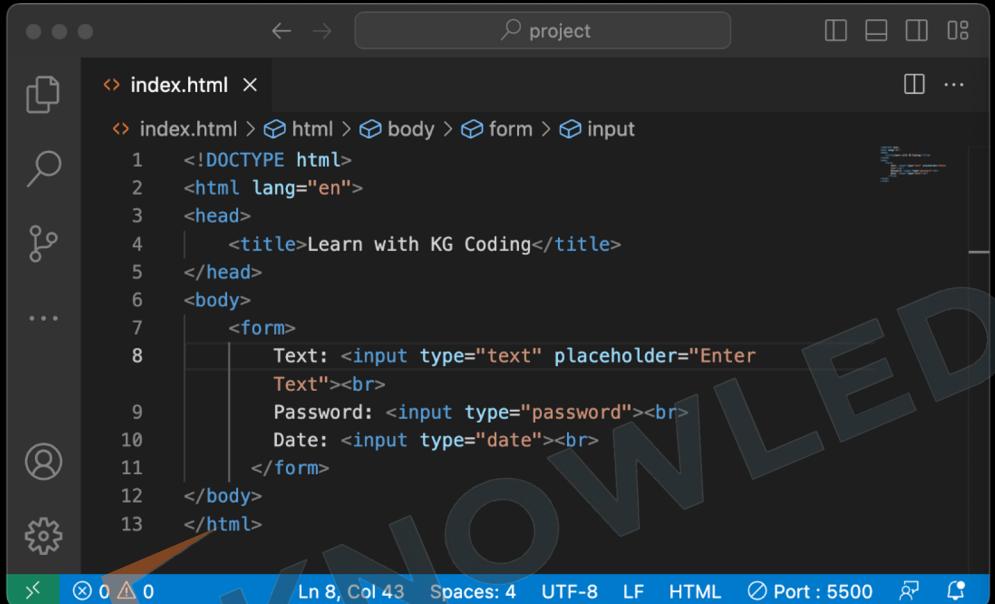
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form action="/submit.php" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
    </form>
</body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. A sidebar on the left contains icons for file operations like new, open, save, and search.



- **Purpose:** Adds a text description to form elements.
- **for Attribute:** Connects the label to a specific **form element** using the element's id.
- **Accessibility:** Makes the form more accessible.
- **Readability:** Enhances form readability and usability.

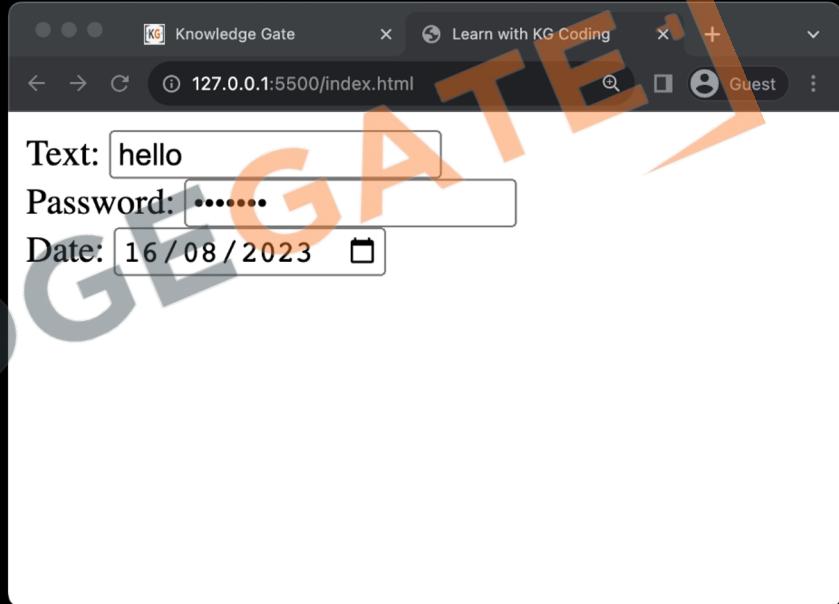
3.5 Input type: Date



The screenshot shows a code editor interface with a dark theme. On the left, there are various icons for file operations like new, open, save, and search. The main area displays the content of 'index.html'.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
    </form>
</body>
</html>
```

The status bar at the bottom indicates the file is saved (0 changes), the current line is Ln 8, column is Col 43, there are 4 spaces, the encoding is UTF-8, and the line separator is LF. It also shows the HTML tab is selected and the port is set to 5500.

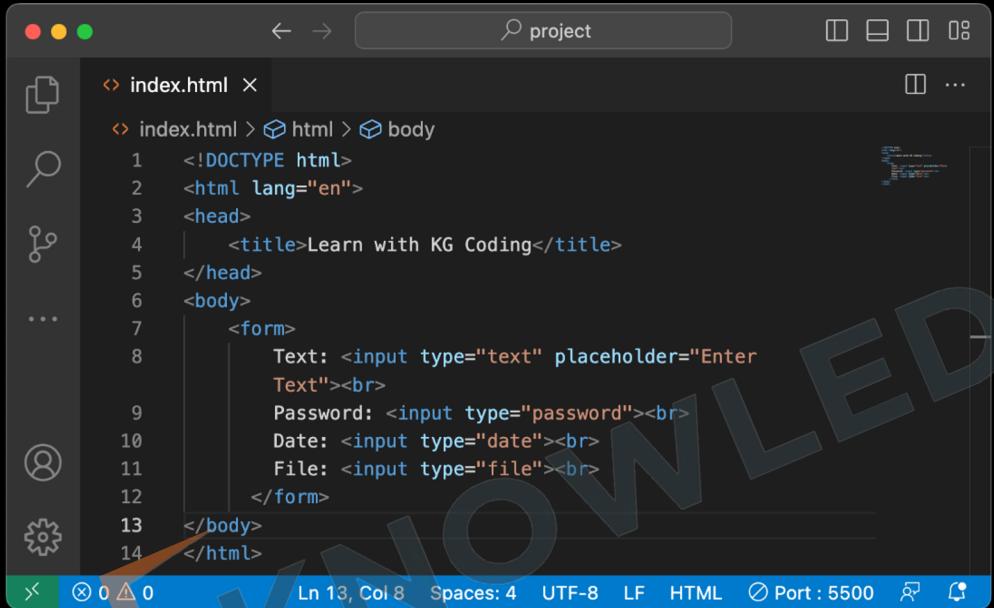


The screenshot shows a web browser window titled 'Knowledge Gate' with the URL '127.0.0.1:5500/index.html'. The page contains a form with three fields:

- Text:
- Password:
- Date: (with a small calendar icon to its right)

A large, semi-transparent watermark reading 'KNOWLEDGE GATE' is overlaid across the entire browser window.

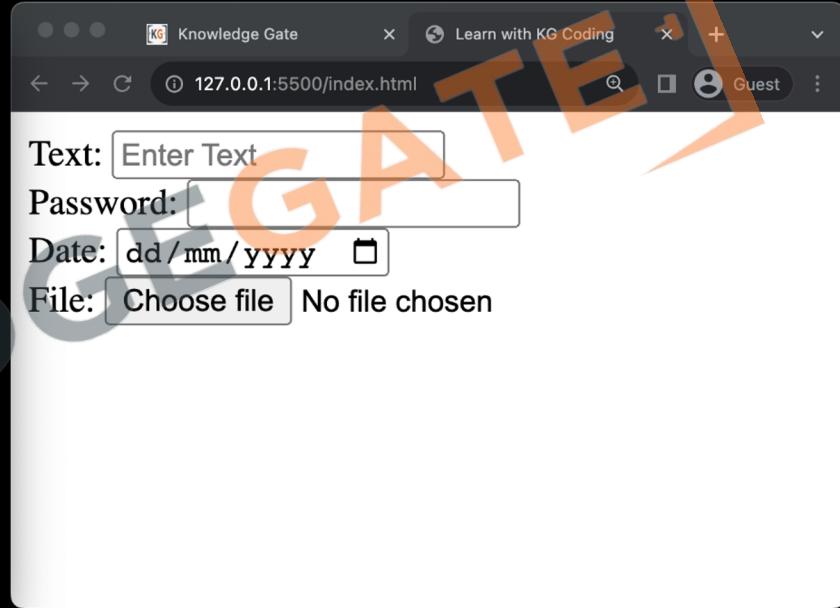
3.5 Input type: File



A screenshot of a code editor window titled "project". The file "index.html" is open, showing the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
        File: <input type="file"><br>
    </form>
</body>
</html>
```

The code editor has a dark theme with light-colored syntax highlighting. A sidebar on the left contains icons for file operations like new, open, save, and search.



A screenshot of a web browser window titled "Knowledge Gate" showing the URL "127.0.0.1:5500/index.html". The page displays the following form fields:

Text:

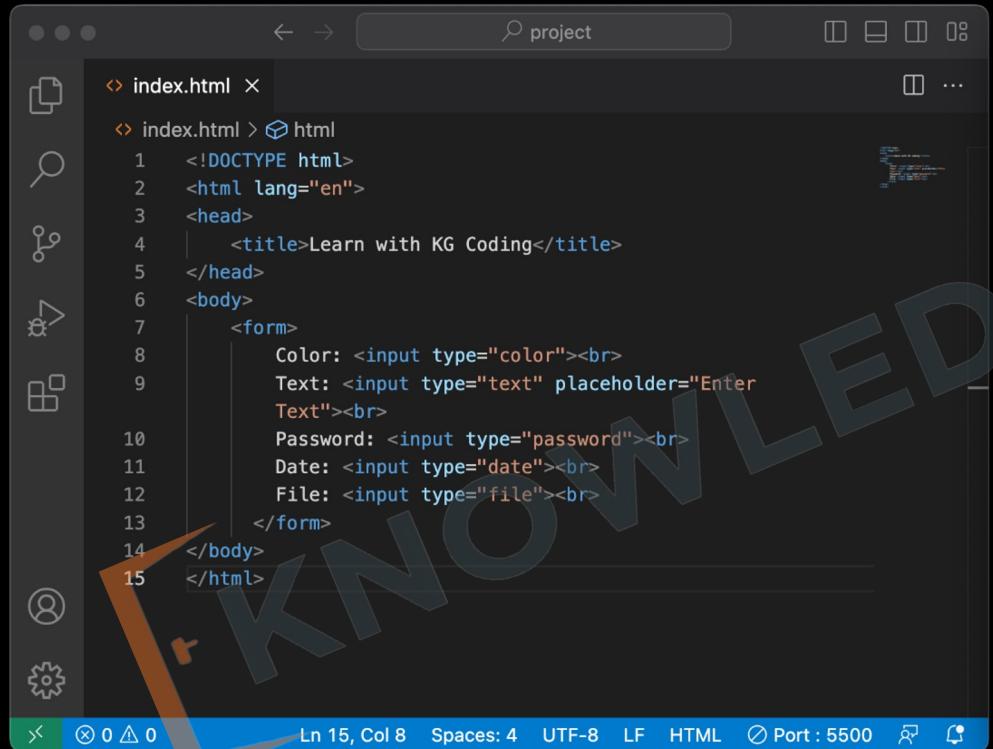
Password:

Date:

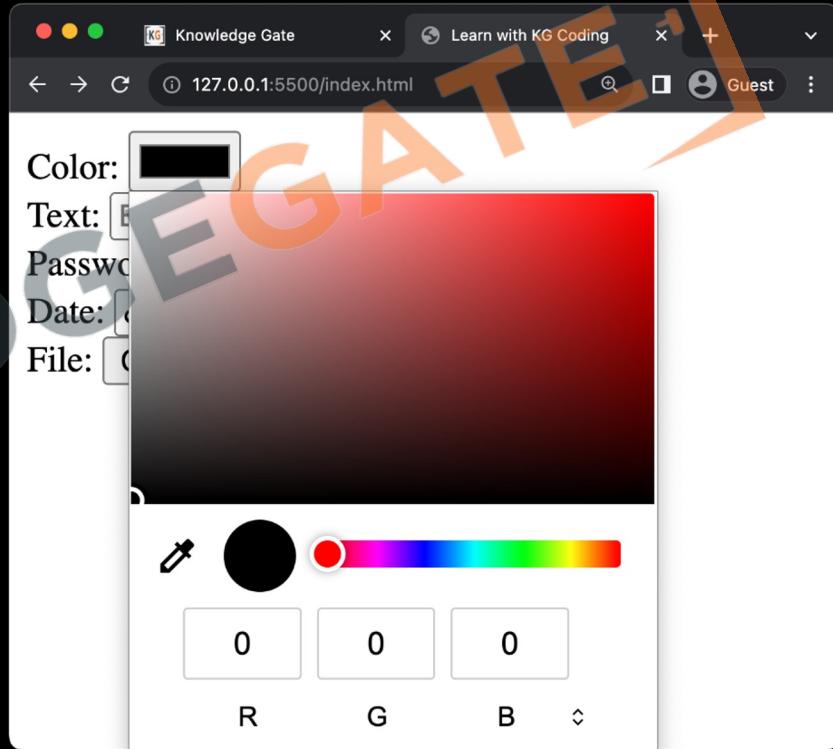
File: Choose file No file chosen

The browser has a light theme. A large watermark reading "KNOWLEDGE GATE" is overlaid across both the code editor and the browser window.

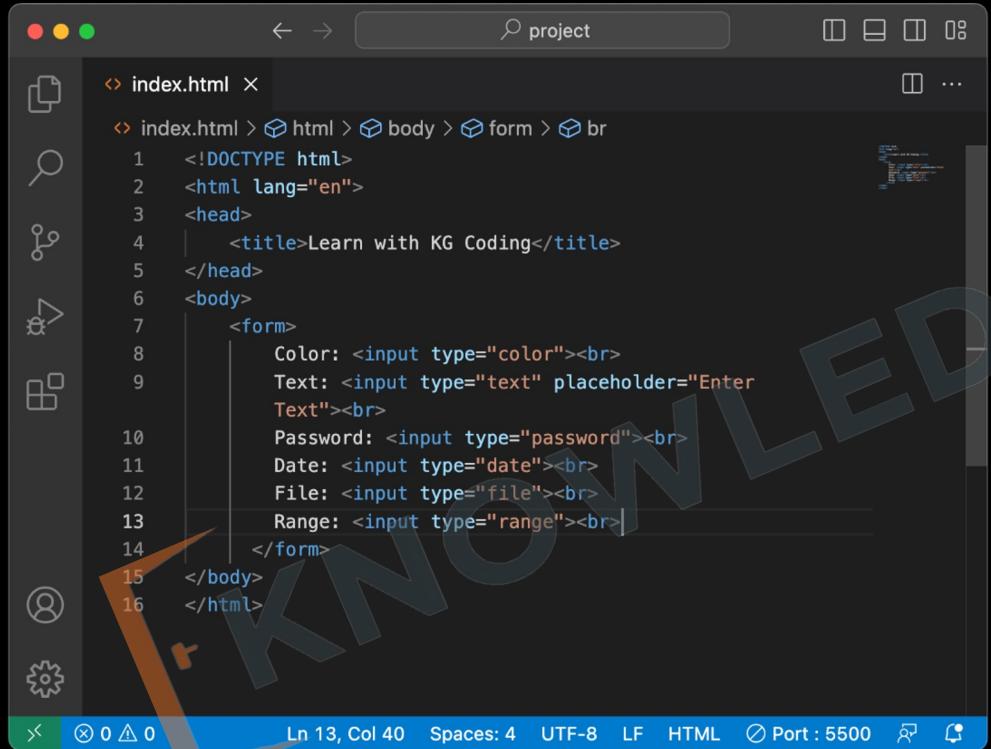
3.5 Input type: Color



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Color: <input type="color"><br>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
        File: <input type="file"><br>
    </form>
</body>
</html>
```

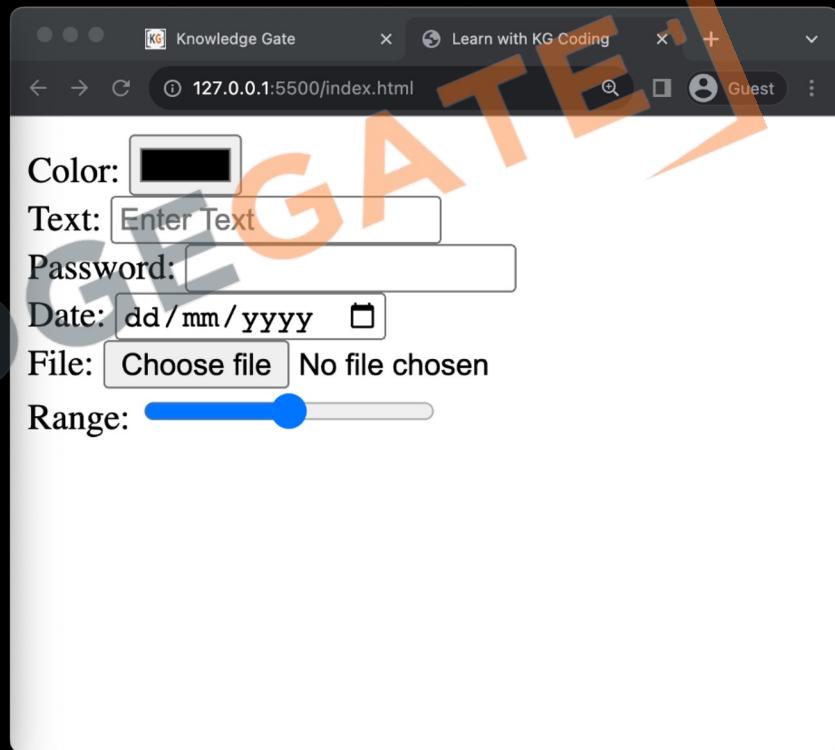


3.5 Input type: Range



```
index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Color: <input type="color"><br>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
        File: <input type="file"><br>
        Range: <input type="range"><br>
    </form>
</body>
</html>
```

Ln 13, Col 40 Spaces: 4 UTF-8 LF HTML ⚙️ Port : 5500 ⚙️ 🔍



Color:

Text:

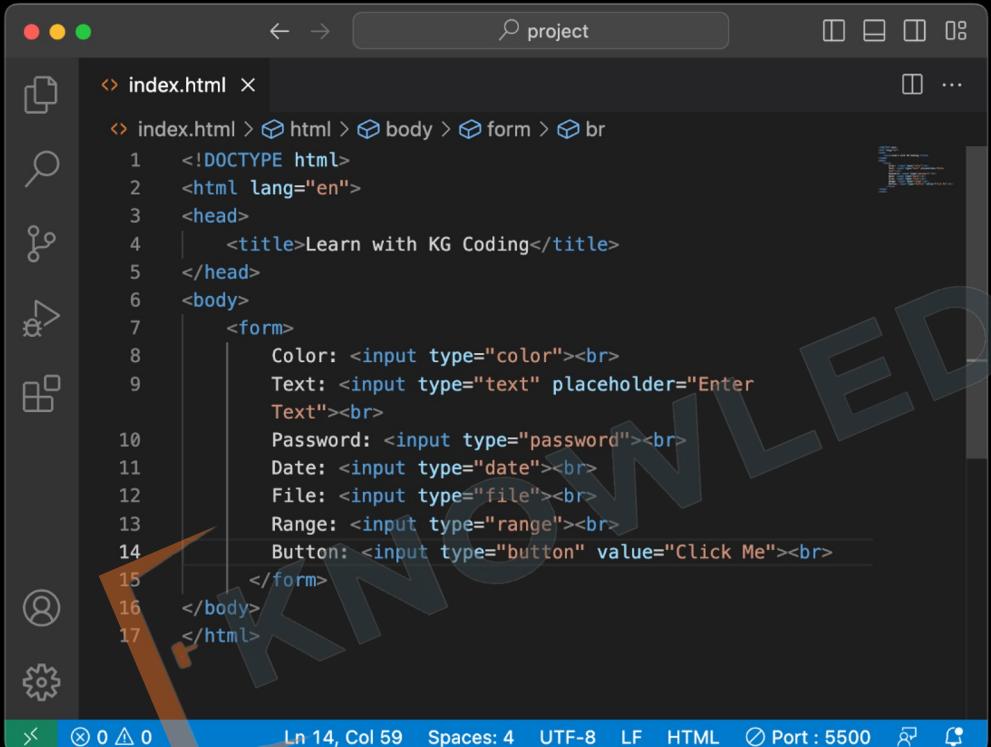
Password:

Date:

File: Choose file No file chosen

Range:

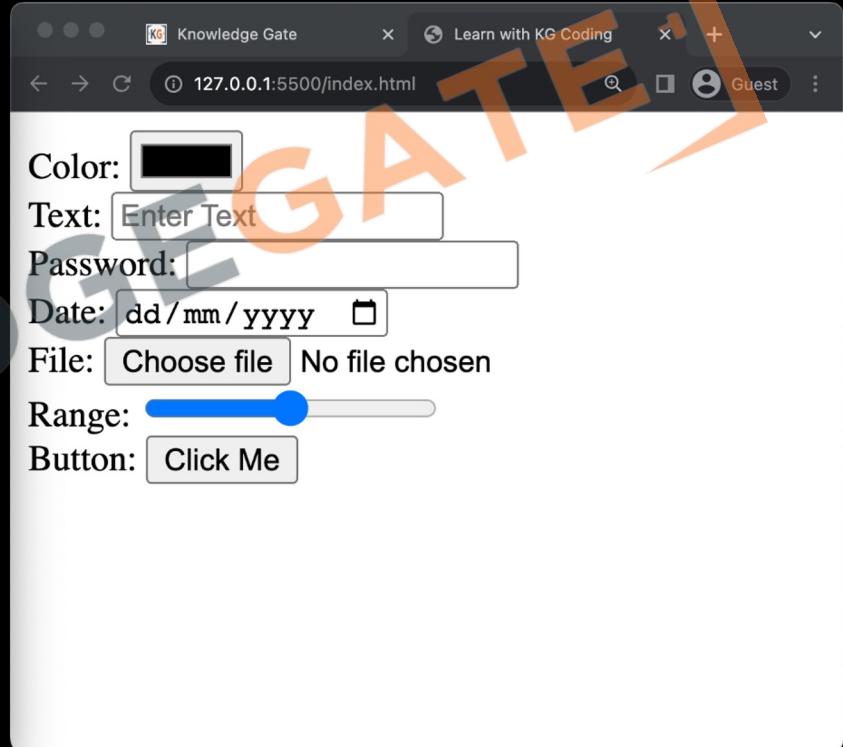
3.5 Input type: Button



A screenshot of a code editor showing the file `index.html`. The code defines a form with several input fields:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Color: <input type="color"><br>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
        File: <input type="file"><br>
        Range: <input type="range"><br>
        Button: <input type="button" value="Click Me"><br>
    </form>
</body>
</html>
```

The code editor interface includes a sidebar with icons for file operations, a search bar, and a status bar at the bottom indicating line 14, column 59, and other file details.

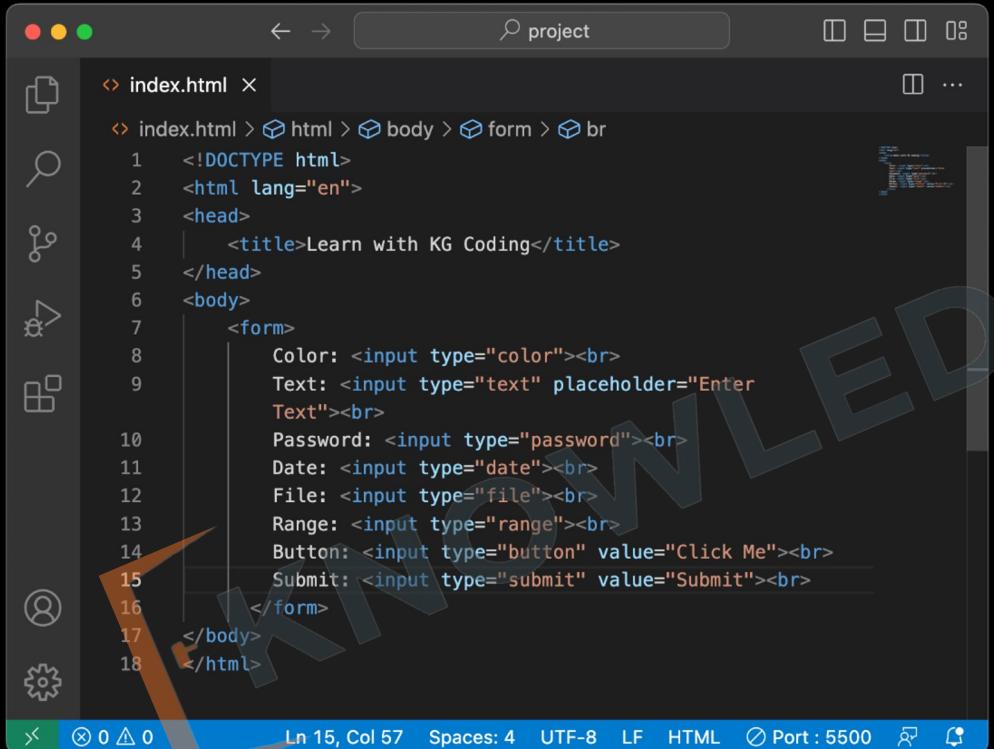


A screenshot of a web browser showing the rendered HTML from the code editor. The page displays the following input fields:

- Color: A color picker button.
- Text: An input field with placeholder text "Enter Text".
- Password: An input field for entering password information.
- Date: An input field for selecting a date in the format dd/mm/yyyy.
- File: An input field for selecting a file, currently showing "Choose file" and "No file chosen".
- Range: A horizontal slider for numerical values.
- Button: A button labeled "Click Me".

The browser window has a watermark "KNOWLEDGE GATE" across it. The address bar shows the URL `127.0.0.1:5500/index.html`.

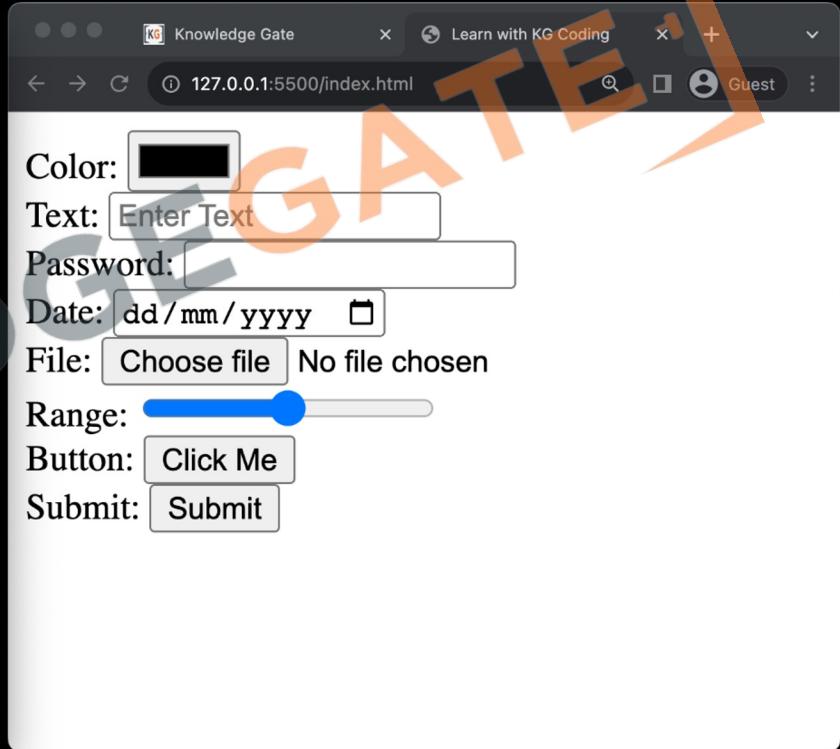
3.5 Input type: Submit



A screenshot of a code editor window titled "index.html". The code displays various input types:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        Color: <input type="color"><br>
        Text: <input type="text" placeholder="Enter Text"><br>
        Password: <input type="password"><br>
        Date: <input type="date"><br>
        File: <input type="file"><br>
        Range: <input type="range"><br>
        Button: <input type="button" value="Click Me"><br>
        Submit: <input type="submit" value="Submit"><br>
    </form>
</body>
</html>
```

The status bar at the bottom shows: Ln 15, Col 57 Spaces: 4 UTF-8 LF HTML ⚙️ Port : 5500 🔍 ⚡



A screenshot of a web browser window titled "Knowledge Gate". The URL is "127.0.0.1:5500/index.html". The page contains a form with the following fields:

Color:

Text:

Password:

Date:

File: No file chosen

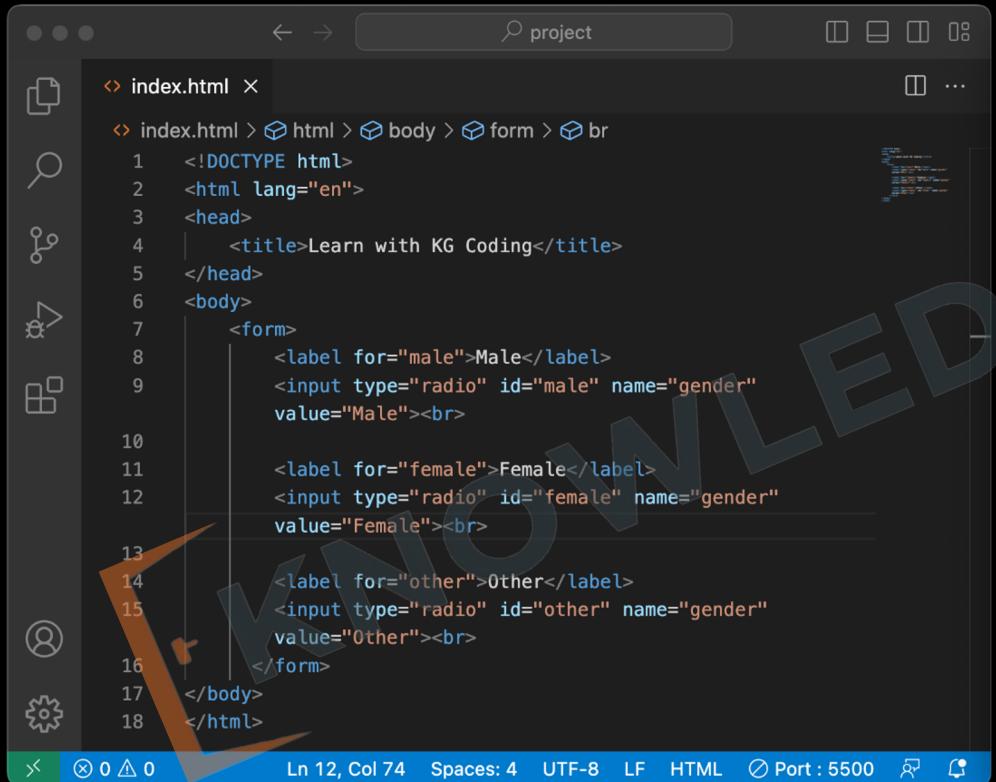
Range:

Button:

Submit:

The browser interface includes a search bar, tabs, and a guest sign-in button.

3.5 Input type: Radio



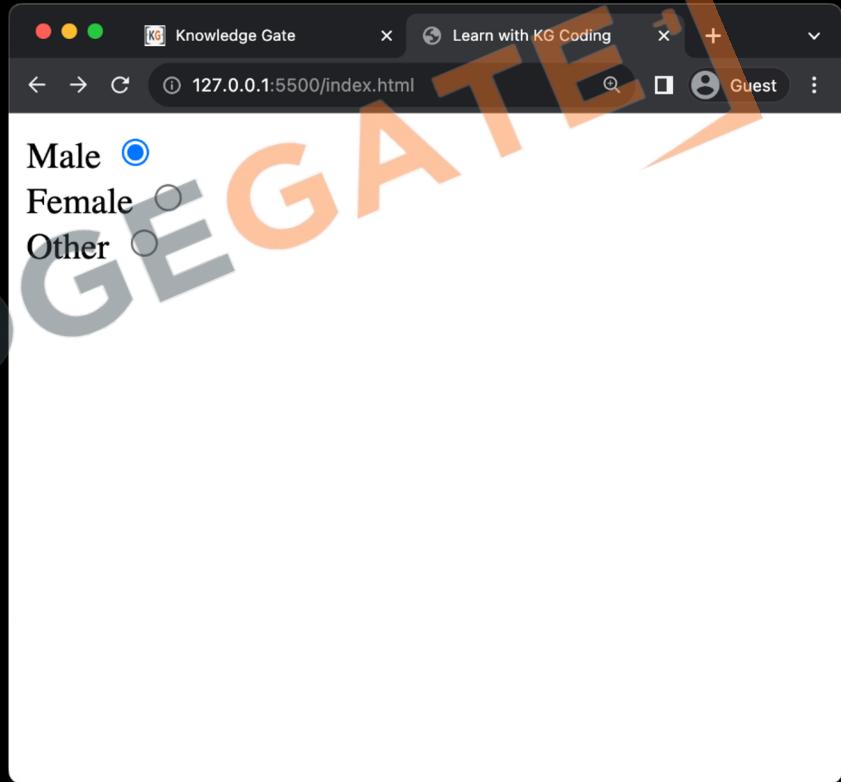
A screenshot of a code editor showing the file `index.html`. The code defines a form with three radio buttons for gender selection:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        <label for="male">Male</label>
        <input type="radio" id="male" name="gender" value="Male"><br>

        <label for="female">Female</label>
        <input type="radio" id="female" name="gender" value="Female"><br>

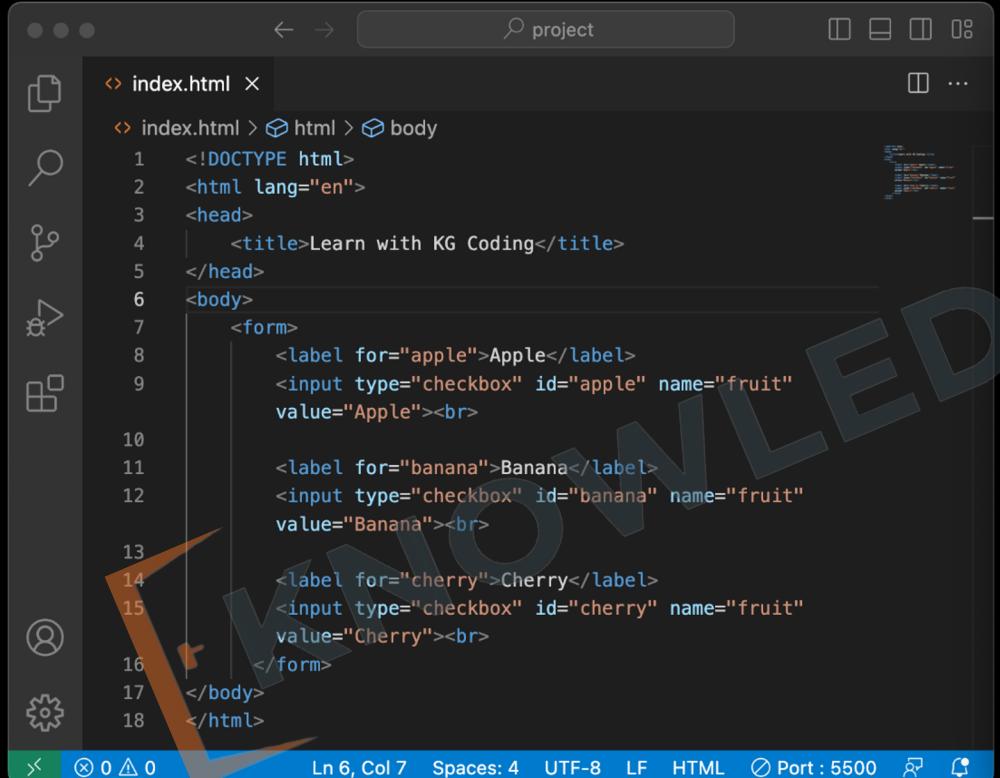
        <label for="other">Other</label>
        <input type="radio" id="other" name="gender" value="Other"><br>
    </form>
</body>
</html>
```

The code editor interface includes a sidebar with various icons and a status bar at the bottom.



A screenshot of a web browser window titled "Knowledge Gate" showing the URL `127.0.0.1:5500/index.html`. The page displays a form with three radio buttons labeled "Male", "Female", and "Other". The "Male" option is selected, indicated by a blue outline around the radio button and the word "Male" in bold. The browser interface includes a title bar, tabs, and a status bar.

3.5 Input type: Checkbox



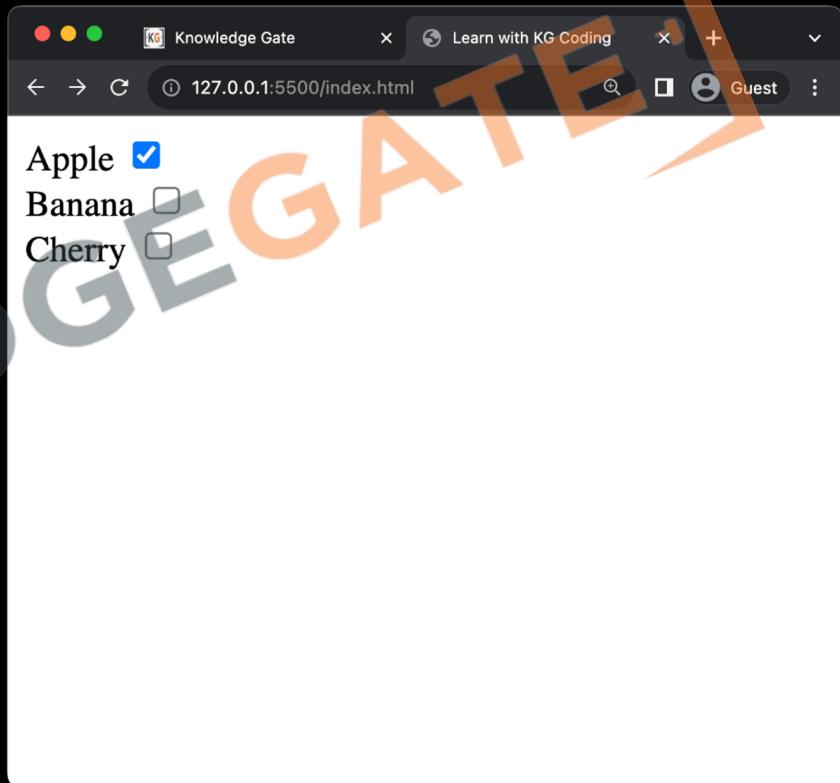
A screenshot of a code editor showing the file `index.html`. The code defines a form with three checkbox inputs, each associated with a fruit label:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        <label for="apple">Apple</label>
        <input type="checkbox" id="apple" name="fruit" value="Apple"><br>

        <label for="banana">Banana</label>
        <input type="checkbox" id="banana" name="fruit" value="Banana"><br>

        <label for="cherry">Cherry</label>
        <input type="checkbox" id="cherry" name="fruit" value="Cherry"><br>
    </form>
</body>
</html>
```

The code editor interface includes a sidebar with various icons and a status bar at the bottom.

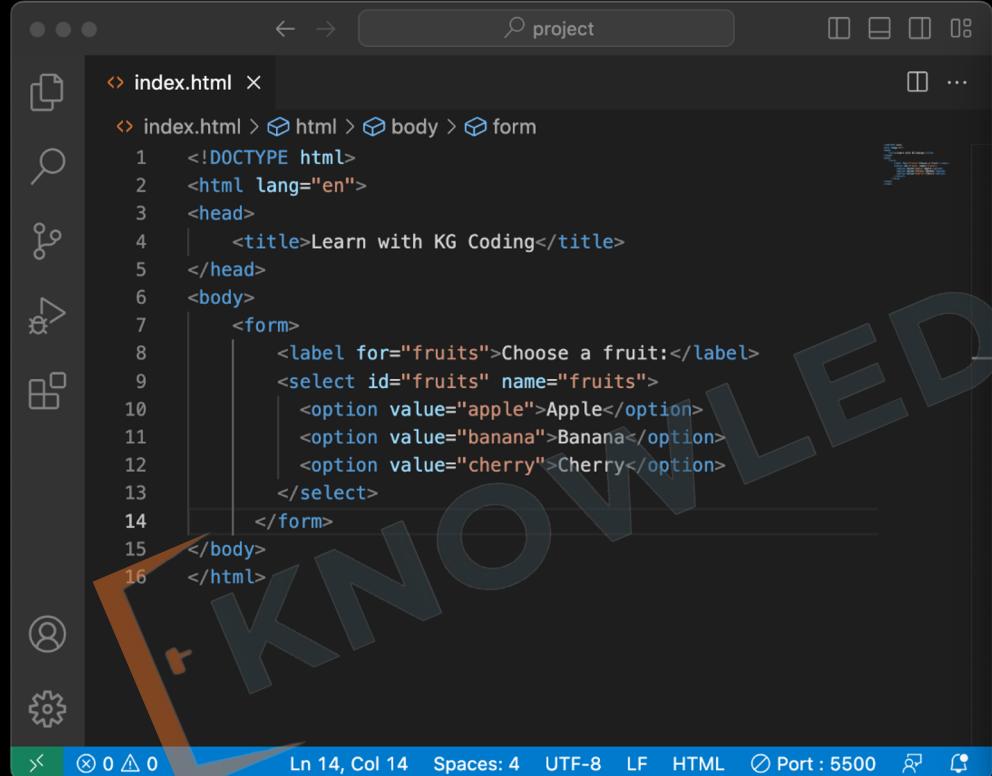


A screenshot of a web browser window titled "Knowledge Gate". The URL is `127.0.0.1:5500/index.html`. The page displays three items with checkboxes:

- Apple
- Banana
- Cherry

The browser interface includes a title bar, tabs, and a status bar at the bottom.

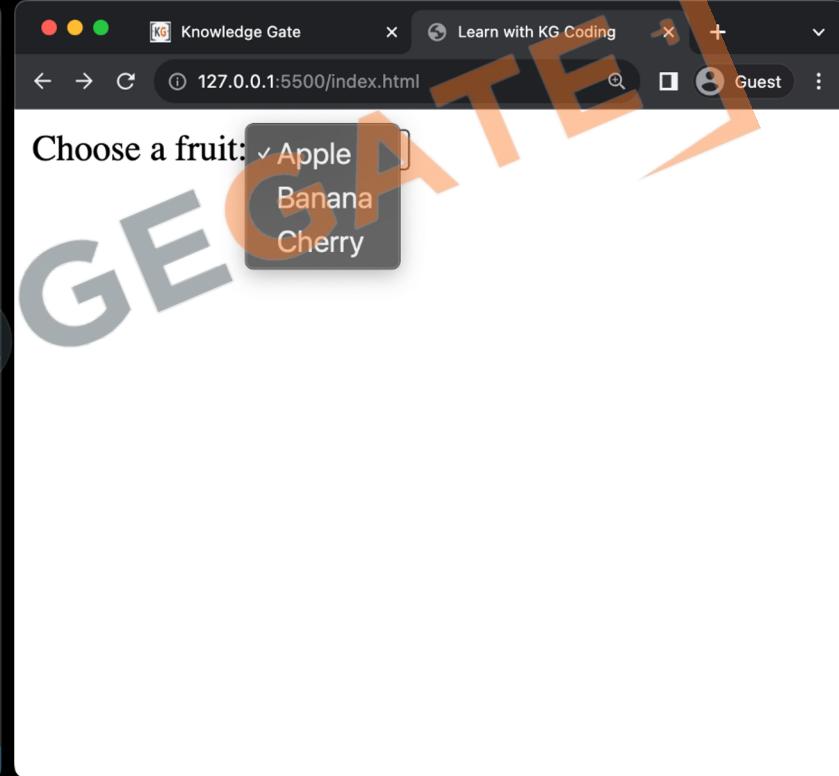
3.5 Input type: Select



A screenshot of a code editor displaying the file `index.html`. The code defines a simple HTML form with a `select` element for selecting a fruit. The code is as follows:

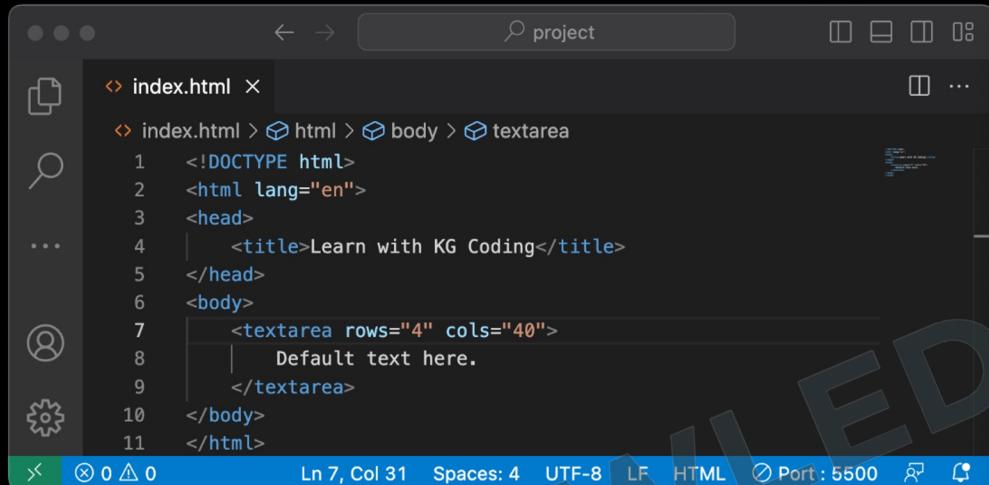
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <form>
        <label for="fruits">Choose a fruit:</label>
        <select id="fruits" name="fruits">
            <option value="apple">Apple</option>
            <option value="banana">Banana</option>
            <option value="cherry">Cherry</option>
        </select>
    </form>
</body>
</html>
```

The code editor interface includes a sidebar with various icons for file operations, a search bar, and a status bar at the bottom indicating line 14, column 14, and other file details.

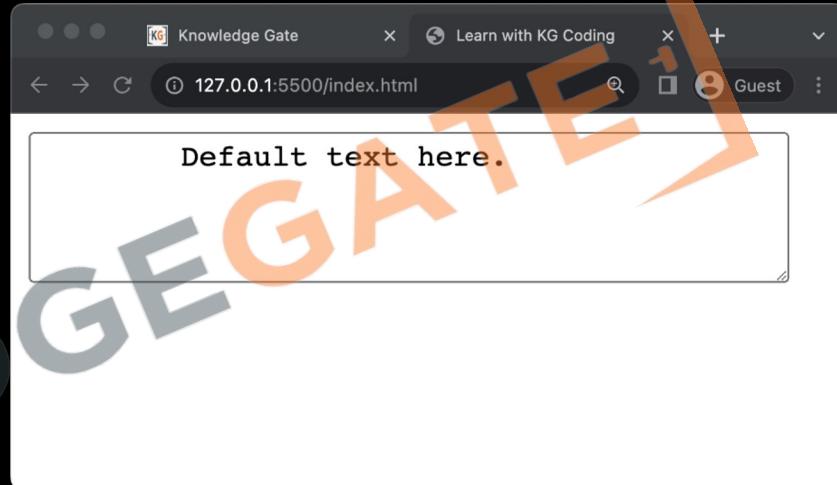


A screenshot of a web browser window titled "Knowledge Gate" showing the URL `127.0.0.1:5500/index.html`. The page displays the text "Choose a fruit:" followed by a dropdown menu with three options: "Apple", "Banana", and "Cherry". The word "KNOWLEDGE GATE" is overlaid diagonally across the browser window.

3.5 Input type: TextArea



```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <textarea rows="4" cols="40">
        Default text here.
    </textarea>
</body>
</html>
```



1. Purpose: `<textarea>` is used for multi-line text input in forms.
 1. **rows Property:** Specifies the visible number of lines in the textarea.
 2. **cols Property:** Sets the visible width measured in average character widths.
2. Resizable: Some browsers allow users to manually resize the textarea.

Level 5

List, Tables & Forms



4GiFATE¹
Frame
Tag

4.1 Using iFrames

The image shows a code editor on the left and a web browser on the right. The code editor displays the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Learn with KG Coding</title>
</head>
<body>
    <iframe width="300" height="200" src="https://en.wikipedia.org/wiki/Main_Page"></iframe>
</body>
</html>
```

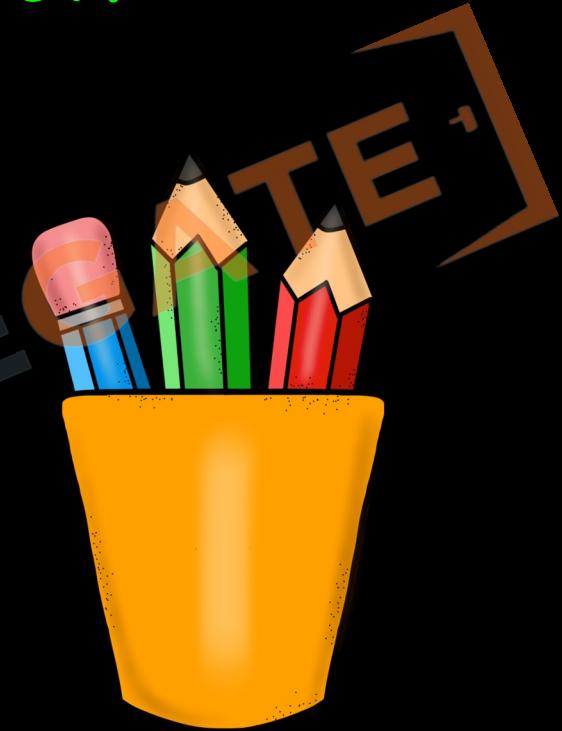
The browser window shows the rendered output of the code, which is a Wikipedia page. A large watermark reading "ACKNOWLEDGED" is diagonally across the image.

1. **Embedded Content:** Allows you to embed another webpage or multimedia content within a webpage.
2. **src Attribute:** Specifies the URL of the content to be embedded.
3. **Dimensions:** Width and height can be set using width and height attributes.

Level 5 Revision

List, Tables & Forms

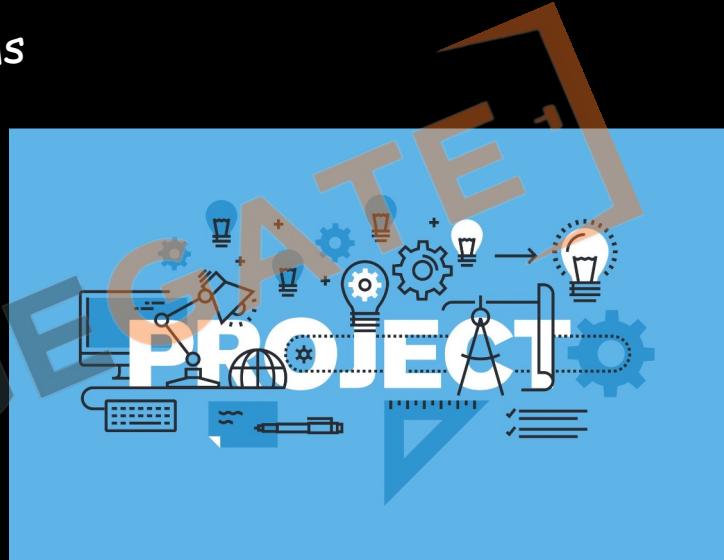
1. List Tag
 1. Ordered Lists
 2. Types of Ordered Lists
 3. Unordered Lists
2. Table Tag
 1. `<tr>`, `<td>`, `<th>` tags
 2. Captions
 3. Col spans
3. Forms
 1. Input tag
 2. Action Attributes
 3. Name and Value Property
 4. Label Tag
 5. Exploring Types
4. iFrame Tag
 1. Using iFrames



Project Level 5

List, Tables & Forms

1. Create a **page** with all type of ordered list and one unordered list.
2. Create a **table** with headings, captions and a few rows. One of heading should take at least 3 columns.
3. Create a **contact me** form with relevant details for your resume website.
4. Use iFrame to add this video to your page.



Level Bonus

Github Pages & CodeSpace

1. Github

1. What is Version Control
2. What is Git and GitHub
3. Account Creation
4. Creating a Repo
5. Creating a Codespace
6. Creating a Github page
7. Publishing our project

2. FrameWorks

1. React
2. Angular
3. Vue



Level Bonus

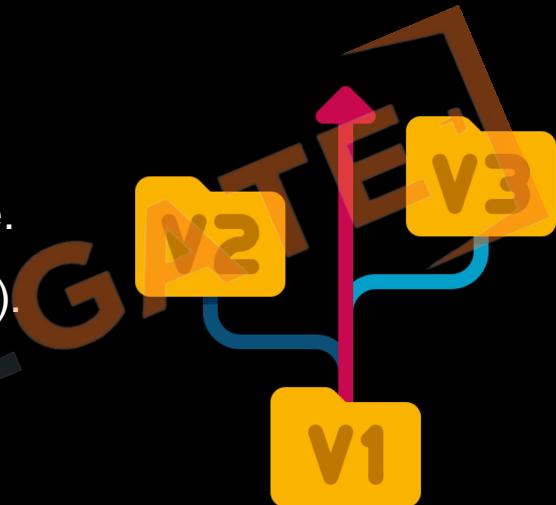
Github Pages & CodeSpace



MIN KNOWLEDGE 1.GATE

1.1 What is Version Control

1. **Definition:** A system to track changes in files over time.
2. **Types:** Centralized (like **SVN**) and Distributed (like **Git**).
3. **Purpose:** Helps in teamwork and fixes mistakes.
4. **Snapshots:** Each 'commit' saves a file version.
5. **Branching:** Lets you work on different tasks separately.
6. **Merge:** Combines changes from different people.
7. **Undo:** Easy to revert to older file versions.



1.2 What is Git and GitHub

What is Git?

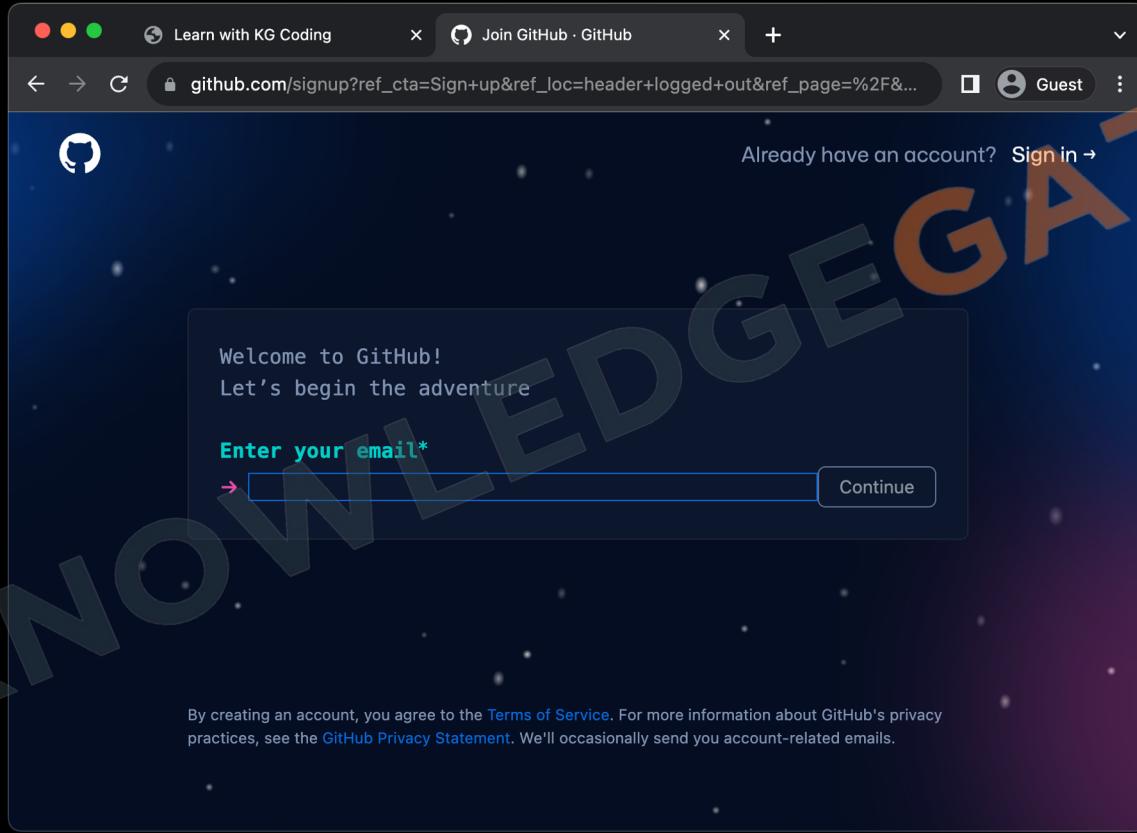
- **Definition:** A software tool that tracks changes in code, enabling collaboration and version control.
- **Commit:** Records a snapshot of file changes.
- **Branch:** Allows separate paths of development.
- **Merge:** Combines changes from different branches.

What is GitHub?

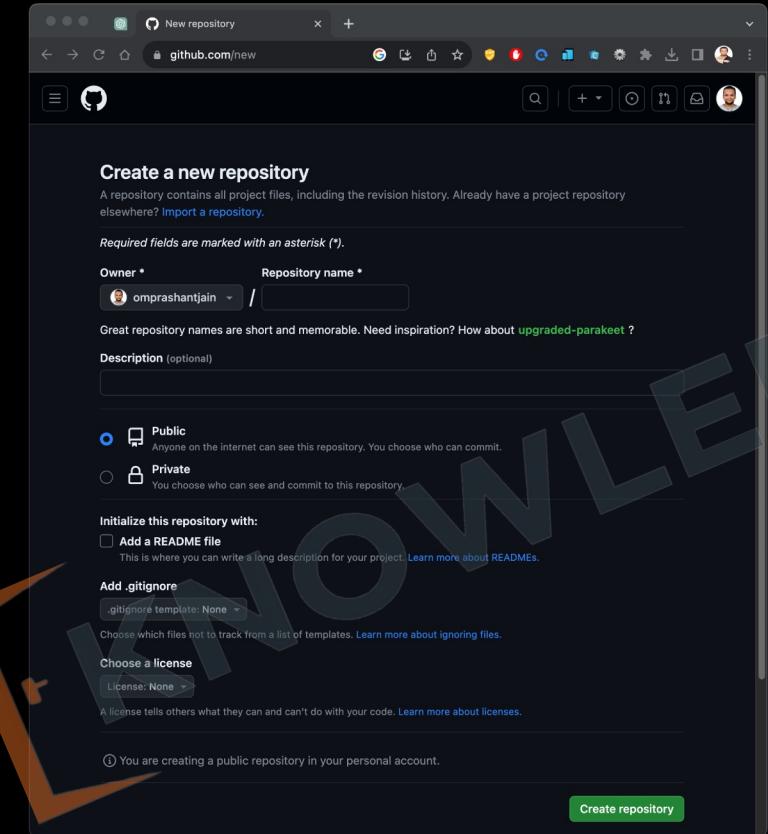
- **Definition:** A web service for hosting and collaborating on Git repositories.
- **Fork:** Creates a personal copy of another user's repository.
- **Pull Request:** A way to propose changes to existing code.
- **Issues:** Used for tracking bugs and feature ideas.



1.3 Account Creation



1.4 Creating a Repo



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Owner *



omprashantjain

Repository name *



/

Great repository names are short and memorable. Need inspiration? How about [upgraded-parakeet](#) ?

Description (optional)

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

Initialize this repository with:

Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

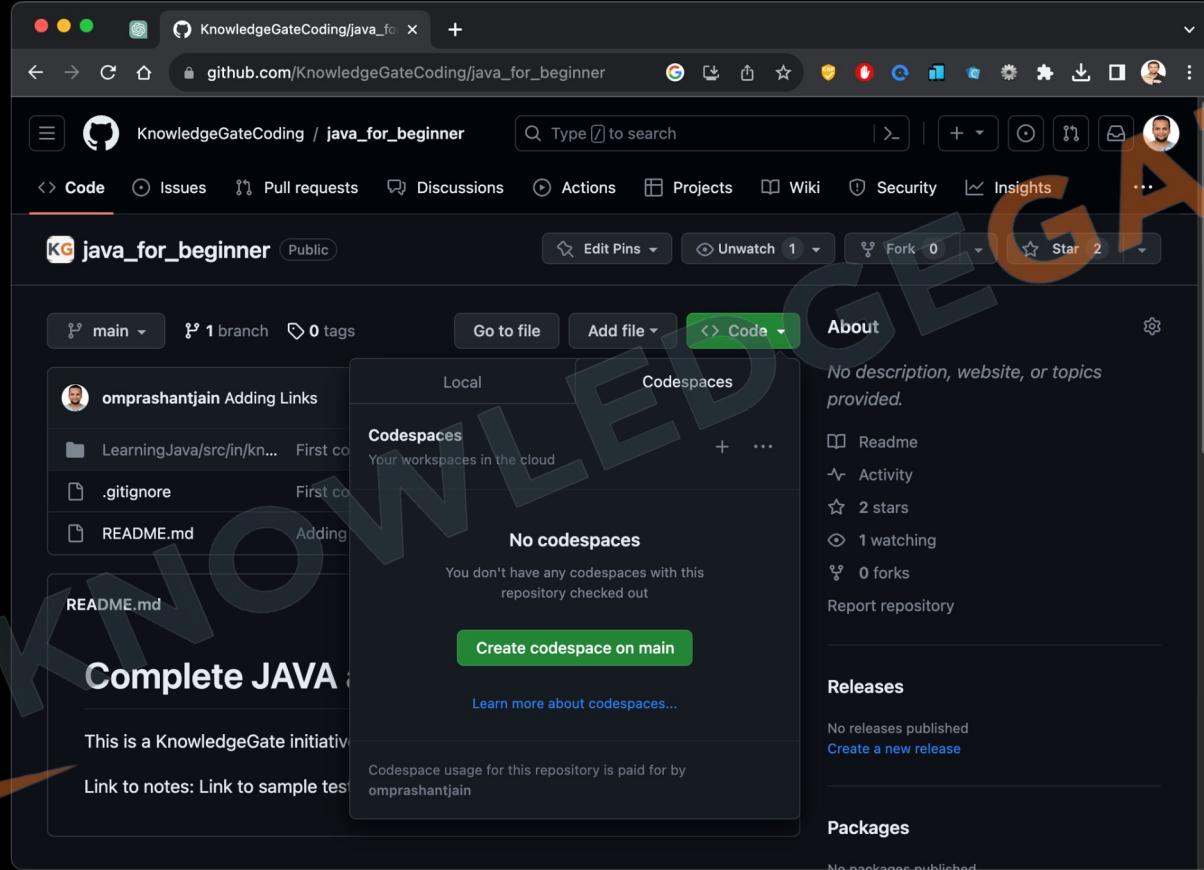
License: None

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

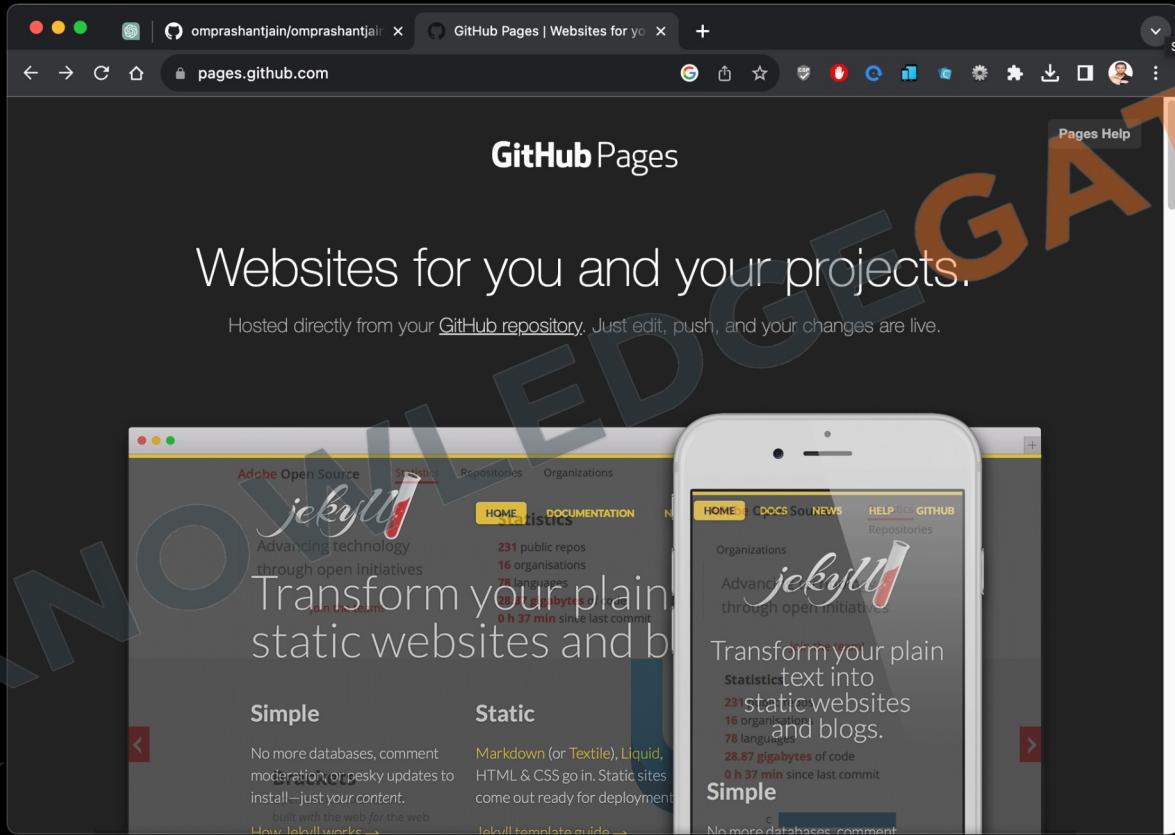
You are creating a public repository in your personal account.

Create repository

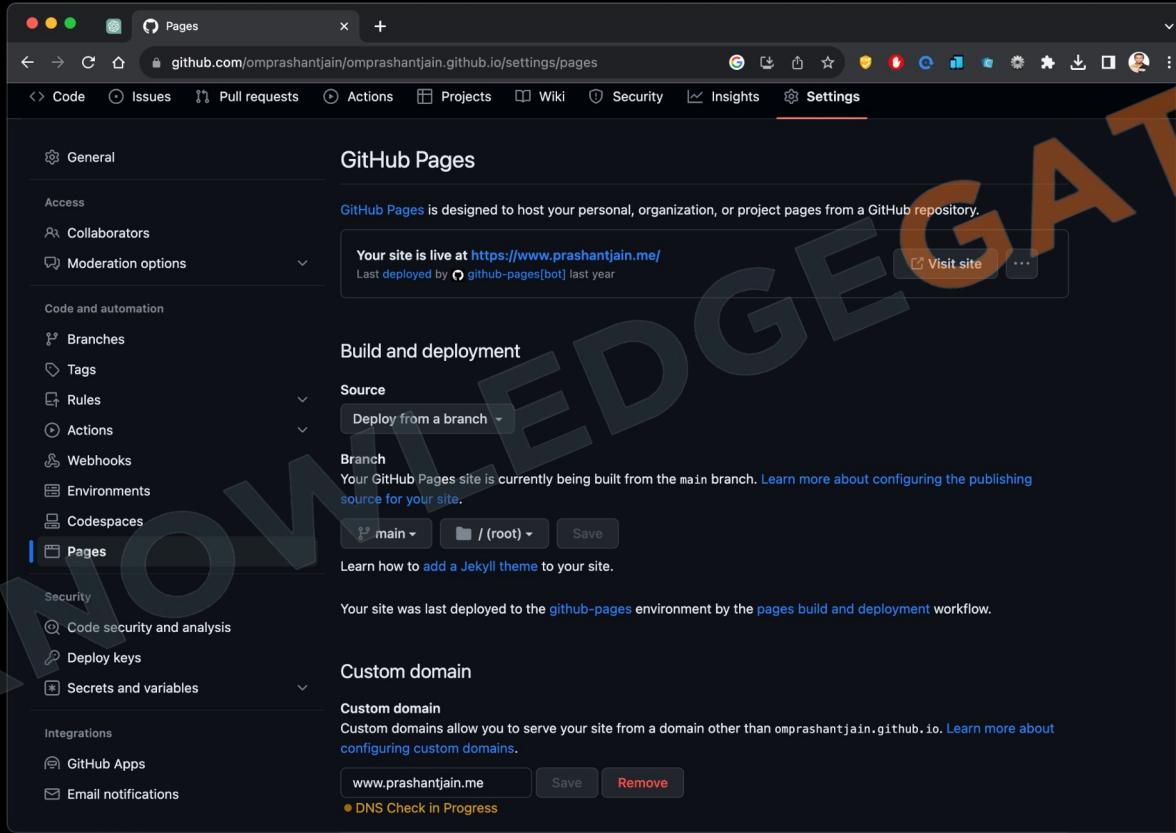
1.5 Creating a CodeSpace



1.6 Creating a Github Page



1.7 Publishing our Project



The screenshot shows the GitHub Pages settings page for the repository `omprashantjain/omprashantjain.github.io`. The main content area displays the message "GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository." Below this, it states "Your site is live at <https://www.prashantjain.me/>" and "Last deployed by [github-pages\[bot\]](#) last year". A large orange watermark reading "KNOWLEDGEABLE" is overlaid across the center of the page.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is live at <https://www.prashantjain.me/>
Last deployed by [github-pages\[bot\]](#) last year

Build and deployment

Source: Deploy from a branch

Branch: Your GitHub Pages site is currently being built from the `main` branch. Learn more about configuring the publishing source for your site.

main / (root) Save

Learn how to add a Jekyll theme to your site.

Your site was last deployed to the `github-pages` environment by the `pages` build and deployment workflow.

Custom domain

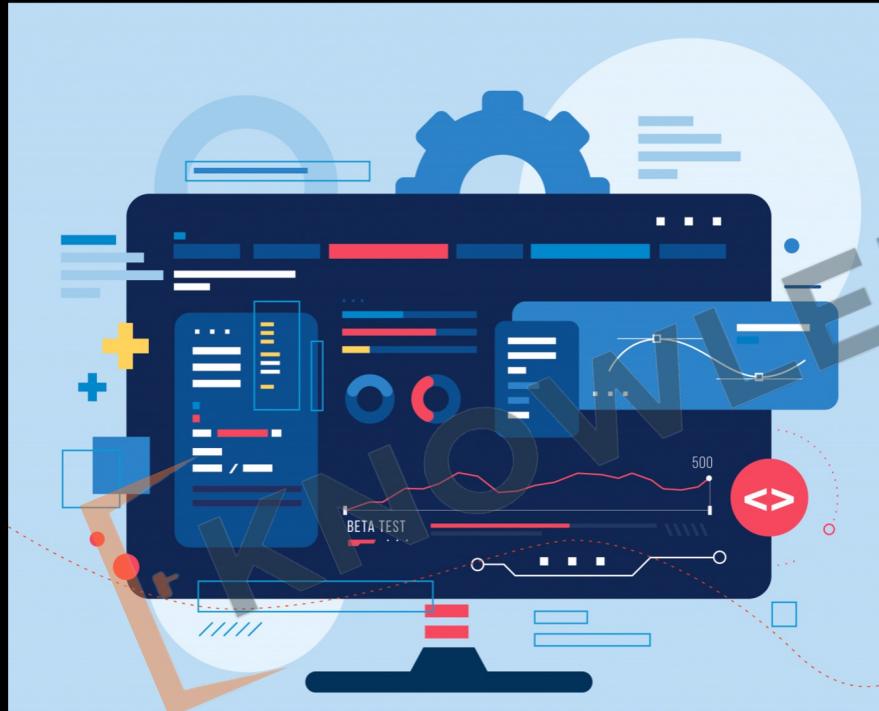
Custom domain: Custom domains allow you to serve your site from a domain other than `omprashantjain.github.io`. Learn more about configuring custom domains.

www.prashantjain.me Save Remove
● DNS Check in Progress

The left sidebar lists various GitHub settings categories: General, Access, Collaborators, Moderation options, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), Integrations (GitHub Apps, Email notifications).

Level Bonus

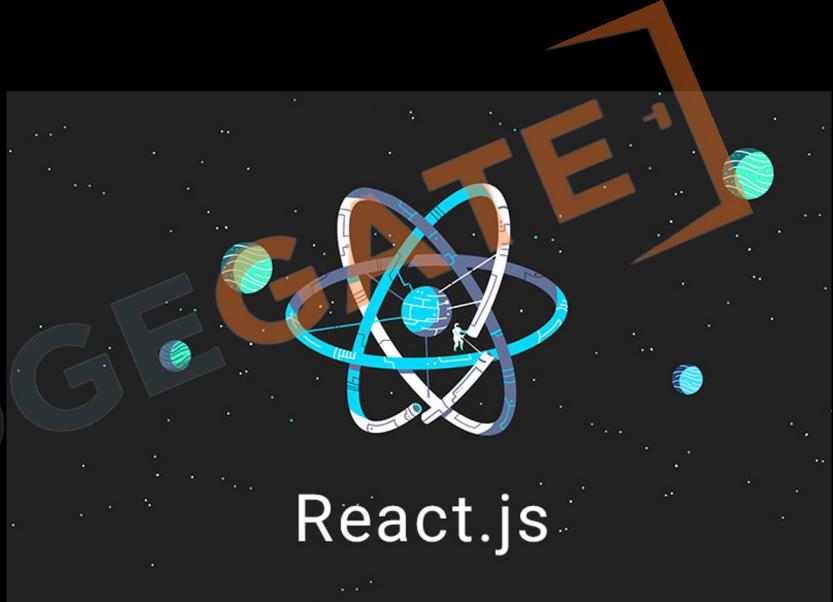
Github Pages & CodeSpace



EDGEGATE¹
2. Frameworks

3.1 ReactJS

1. **Definition:** A tool for making websites interactive.
2. **Components:** Reusable pieces for building a webpage.
3. **Virtual DOM:** Makes websites faster by updating only what's needed.
4. **JSX:** A special way to write code that looks like **HTML**.
5. **State:** Keeps track of changes on the webpage.
6. **Props:** Shares information between different parts of a webpage.



3.2 AngularJS

1. **Definition:** A framework for building web applications, developed by **Google**.
2. **Two-Way Data Binding:** Updates both the view and the model simultaneously.
3. **Directives:** Custom **HTML** tags for added functionality.
4. **Dependency Injection:** **Automatically manages** how parts of the app work together.
5. **Controllers:** Manages the data for a specific part of the webpage.
6. **SPA Support:** Good for **Single Page Applications** where the page doesn't reload.



3.3 VueJS

1. **Definition:** A JavaScript framework for creating web interfaces.
2. **Components:** Small, reusable parts for building a website.
3. **Reactivity:** Automatically updates the webpage when data changes.
4. **Directives:** Special tokens in HTML for added functionality.
5. **Vuex:** Helps manage shared data across the site.
6. **Single-File Components:** Keeps template, script, and style in one file.



Level Bonus Revision

Github Pages & CodeSpace

1. Github

1. What is Version Control
2. What is Git and GitHub
3. Account Creation
4. Creating a Repo
5. Creating a Codespace
6. Creating a Github page
7. Publishing our project

2. FrameWorks

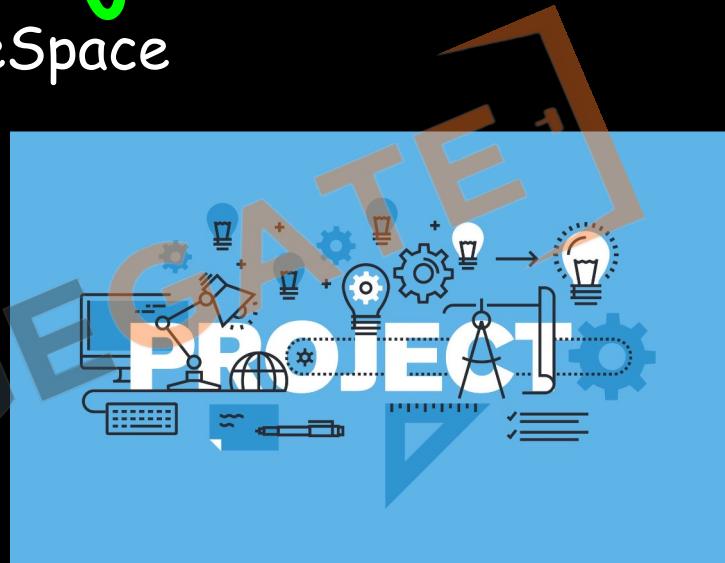
1. React
2. Angular
3. Vue



Level Bonus Project

Github Pages & CodeSpace

1. Create a Github account if you don't have one.
2. Create a repo for the project you have done till now.
3. Create a Codespace and make changes.
4. Publish it using Github pages.



Major Project

Idea

1. Create your **Resume** or Portfolio website.
2. Create a repo in **Github**.
3. Publish it on **Github pages**.
4. Add the link to your **resume**.
5. Add the link in the **comment** section.

