

UI Development with Vue.js and Vuetify



Mark Scott

PLURALSIGHT AUTHOR

@tripletdad99



Core UI Development



Vue.js

The “progressive framework” for building user interfaces



Vuetify

Material Design UI framework for Vue.js with a lot of ready to use components





Busy Module!

A lot of UI work ahead

Short amount of time

Buckle up!



Mockup and Business Requirements Review





Sign in

email

password

Sign in





- [Add Transaction](#)
- [Current Month](#)
- [Contact Us](#)
- [About](#)

Transaction for [Month - Year] [< Previous](#) [Next >](#)

Search

Date	Type	Description	Payment	Deposit	Balance

Rows per page: From 1 to X < >





- Add Transaction
- Current Month
- Contact Us
- About

New Transaction

-
-
-
-
-

Rows per page:

From 1 to X < >





- [Add Transaction](#)
- [Current Month](#)
- [Contact Us](#)
- [About](#)

Transaction for [Month - Year] [< Previous](#) [Next >](#)

Search

Date	Type	Description	Payment	Deposit	Balance

Rows per page:

From 1 to X

< >

+



Getting Our Development Environment Setup





EXTENSIONS



Welcome

package.json

transaction.js

user.js ...\\models

user.js ...\\routes

Welcome to GitLens

Extension: Vetur



Search Extensions in Marketplace

INSTALLED

**Vetur**

octref.vetur

Pine Wu

2,091,329



Repository

License

Vue tooling for VS Code

Disable

Uninstall

This extension is recommended based on the files you recently opened.

[Details](#) [Contributions](#) [Changelog](#) [Dependencies](#)

Vetur

VS Marketplace

v0.11.7

installs

2.09M

rating

5/5

(56)

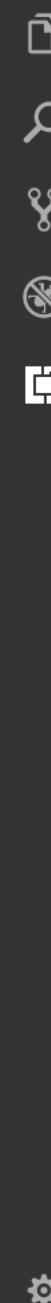
Vue tooling for VS Code, powered by [vue-language-server](#).Doc: <https://vuejs.github.io/vetur>Try it out with [Veturpack](#)! [VueConf 2017 Slide & Video](#)

Features

- Syntax-highlighting
- Snippet
- Emmet
- Linting / Error Checking
- Formatting
- Auto Completion
- Debugging

Quick Start

[Install Vetur](#)



EXTENSIONS



Extension: GitLens — Git supercharged



GitLens — Git supercharged

eamodio.gitlens

Eric Amodio | 3,774,508 | ★★★★★ | Repository | License

Supercharge the Git capabilities built into Visual Studio Code — Visualize code authorship at a glance via Git blame annotations and code lens, seamlessly n...

Disable | Uninstall

Details Contributions Changelog Dependencies

```

199 const [file, root] = Git.splitPath(fileName, repoPath, false);
200
201 try {
202   const data = await Git.diff(root, file, sha1, sha2);
203   return GitDiffParser.parse(data, this.config.debug);
204 }
205 catch (ex) {
206   // Trap and cache expected diff errors
207   if (entry) {
208     const msg = ex.message;
209     Logger.log(`Replace diff cache with empty promise for ${entry.key}`);
210     entry.set<ICachedDiff>(key, {
211       item: GitService.EmptyPromise,
212       errorMessage: msg
213     } as ICachedDiff);
214   }
215   return await GitService.EmptyPromise as IGitDiff;
216 }
217
218 return undefined;
219
220 }
221
222 async getDiffForLine(uri: GitUri, line: number, sha1: string, sha2: string)
223 try {
224   const diff = await this.getDiffForFile(uri, sha1, sha2);
225   if (diff === undefined) return undefined;
226
227   const chunk = diff.chunks.find(_ => Math.min(_.originalStart, _.curr
228   if (chunk === undefined) return undefined;
229
230   const [chunk.original[line - chunk.originalStart + 1],
231         chunk.changes[line - chunk.changesStart + 1]
232       ];
233   return [
234     chunk.previous[line - deleted - 1],
235     chunk.current[line + deleted]
236   ];
237 }
238
239 catch (ex) {
240   // search for the line skipping deleted lines -- since they don't
241   // keep track of the deleted lines for the original version
242   line = line - chunk.currentStart + 1;
243   let count = 0;
244   let deleted = 0;
245   for (const l of chunk.current) {
246     if (l === undefined) {
247       deleted++;
248       if (count === line) break;
249     }
250     continue;
251   }
252   if (count === line) break;
253   count++;
254 }
255
256 return [
257   chunk.previous[line - deleted - 1],
258   chunk.current[line + deleted]
259 ];
260 }
261
262 catch (ex) {
263   // search for the line skipping deleted lines -- since they don't
264   // keep track of the deleted lines for the original version
265   line = line - chunk.currentStart + 1;
266   let count = 0;
267   let deleted = 0;
268   for (const l of chunk.current) {
269     if (l === undefined) {
270       deleted++;
271       if (count === line) break;
272     }
273     continue;
274   }
275   if (count === line) break;
276   count++;
277 }
278
279 return [
280   chunk.previous[line - deleted - 1],
281   chunk.current[line + deleted]
282 ];
283 }
284
285 catch (ex) {
286   // search for the line skipping deleted lines -- since they don't
287   // keep track of the deleted lines for the original version
288   line = line - chunk.currentStart + 1;
289   let count = 0;
290   let deleted = 0;
291   for (const l of chunk.current) {
292     if (l === undefined) {
293       deleted++;
294       if (count === line) break;
295     }
296     continue;
297   }
298   if (count === line) break;
299   count++;
300 }
301
302 return [
303   chunk.previous[line - deleted - 1],
304   chunk.current[line + deleted]
305 ];
306 }
307
308 catch (ex) {
309   // search for the line skipping deleted lines -- since they don't
310   // keep track of the deleted lines for the original version
311   line = line - chunk.currentStart + 1;
312   let count = 0;
313   let deleted = 0;
314   for (const l of chunk.current) {
315     if (l === undefined) {
316       deleted++;
317       if (count === line) break;
318     }
319     continue;
320   }
321   if (count === line) break;
322   count++;
323 }
324
325 return [
326   chunk.previous[line - deleted - 1],
327   chunk.current[line + deleted]
328 ];
329 }
330
331 catch (ex) {
332   // search for the line skipping deleted lines -- since they don't
333   // keep track of the deleted lines for the original version
334   line = line - chunk.currentStart + 1;
335   let count = 0;
336   let deleted = 0;
337   for (const l of chunk.current) {
338     if (l === undefined) {
339       deleted++;
340       if (count === line) break;
341     }
342     continue;
343   }
344   if (count === line) break;
345   count++;
346 }
347
348 return [
349   chunk.previous[line - deleted - 1],
350   chunk.current[line + deleted]
351 ];
352 }
353
354 catch (ex) {
355   // search for the line skipping deleted lines -- since they don't
356   // keep track of the deleted lines for the original version
357   line = line - chunk.currentStart + 1;
358   let count = 0;
359   let deleted = 0;
360   for (const l of chunk.current) {
361     if (l === undefined) {
362       deleted++;
363       if (count === line) break;
364     }
365     continue;
366   }
367   if (count === line) break;
368   count++;
369 }
370
371 return [
372   chunk.previous[line - deleted - 1],
373   chunk.current[line + deleted]
374 ];
375 }
376
377 catch (ex) {
378   // search for the line skipping deleted lines -- since they don't
379   // keep track of the deleted lines for the original version
380   line = line - chunk.currentStart + 1;
381   let count = 0;
382   let deleted = 0;
383   for (const l of chunk.current) {
384     if (l === undefined) {
385       deleted++;
386       if (count === line) break;
387     }
388     continue;
389   }
390   if (count === line) break;
391   count++;
392 }
393
394 return [
395   chunk.previous[line - deleted - 1],
396   chunk.current[line + deleted]
397 ];
398 }
399
400 catch (ex) {
401   // search for the line skipping deleted lines -- since they don't
402   // keep track of the deleted lines for the original version
403   line = line - chunk.currentStart + 1;
404   let count = 0;
405   let deleted = 0;
406   for (const l of chunk.current) {
407     if (l === undefined) {
408       deleted++;
409       if (count === line) break;
410     }
411     continue;
412   }
413   if (count === line) break;
414   count++;
415 }
416
417 return [
418   chunk.previous[line - deleted - 1],
419   chunk.current[line + deleted]
420 ];
421 }
422
423 catch (ex) {
424   // search for the line skipping deleted lines -- since they don't
425   // keep track of the deleted lines for the original version
426   line = line - chunk.currentStart + 1;
427   let count = 0;
428   let deleted = 0;
429   for (const l of chunk.current) {
430     if (l === undefined) {
431       deleted++;
432       if (count === line) break;
433     }
434     continue;
435   }
436   if (count === line) break;
437   count++;
438 }
439
440 return [
441   chunk.previous[line - deleted - 1],
442   chunk.current[line + deleted]
443 ];
444 }
445
446 catch (ex) {
447   // search for the line skipping deleted lines -- since they don't
448   // keep track of the deleted lines for the original version
449   line = line - chunk.currentStart + 1;
450   let count = 0;
451   let deleted = 0;
452   for (const l of chunk.current) {
453     if (l === undefined) {
454       deleted++;
455       if (count === line) break;
456     }
457     continue;
458   }
459   if (count === line) break;
460   count++;
461 }
462
463 return [
464   chunk.previous[line - deleted - 1],
465   chunk.current[line + deleted]
466 ];
467 }
468
469 catch (ex) {
470   // search for the line skipping deleted lines -- since they don't
471   // keep track of the deleted lines for the original version
472   line = line - chunk.currentStart + 1;
473   let count = 0;
474   let deleted = 0;
475   for (const l of chunk.current) {
476     if (l === undefined) {
477       deleted++;
478       if (count === line) break;
479     }
480     continue;
481   }
482   if (count === line) break;
483   count++;
484 }
485
486 return [
487   chunk.previous[line - deleted - 1],
488   chunk.current[line + deleted]
489 ];
490 }
491
492 catch (ex) {
493   // search for the line skipping deleted lines -- since they don't
494   // keep track of the deleted lines for the original version
495   line = line - chunk.currentStart + 1;
496   let count = 0;
497   let deleted = 0;
498   for (const l of chunk.current) {
499     if (l === undefined) {
500       deleted++;
501       if (count === line) break;
502     }
503     continue;
504   }
505   if (count === line) break;
506   count++;
507 }
508
509 return [
510   chunk.previous[line - deleted - 1],
511   chunk.current[line + deleted]
512 ];
513 }
514
515 catch (ex) {
516   // search for the line skipping deleted lines -- since they don't
517   // keep track of the deleted lines for the original version
518   line = line - chunk.currentStart + 1;
519   let count = 0;
520   let deleted = 0;
521   for (const l of chunk.current) {
522     if (l === undefined) {
523       deleted++;
524       if (count === line) break;
525     }
526     continue;
527   }
528   if (count === line) break;
529   count++;
530 }
531
532 return [
533   chunk.previous[line - deleted - 1],
534   chunk.current[line + deleted]
535 ];
536 }
537
538 catch (ex) {
539   // search for the line skipping deleted lines -- since they don't
540   // keep track of the deleted lines for the original version
541   line = line - chunk.currentStart + 1;
542   let count = 0;
543   let deleted = 0;
544   for (const l of chunk.current) {
545     if (l === undefined) {
546       deleted++;
547       if (count === line) break;
548     }
549     continue;
550   }
551   if (count === line) break;
552   count++;
553 }
554
555 return [
556   chunk.previous[line - deleted - 1],
557   chunk.current[line + deleted]
558 ];
559 }
560
561 catch (ex) {
562   // search for the line skipping deleted lines -- since they don't
563   // keep track of the deleted lines for the original version
564   line = line - chunk.currentStart + 1;
565   let count = 0;
566   let deleted = 0;
567   for (const l of chunk.current) {
568     if (l === undefined) {
569       deleted++;
570       if (count === line) break;
571     }
572     continue;
573   }
574   if (count === line) break;
575   count++;
576 }
577
578 return [
579   chunk.previous[line - deleted - 1],
580   chunk.current[line + deleted]
581 ];
582 }
583
584 catch (ex) {
585   // search for the line skipping deleted lines -- since they don't
586   // keep track of the deleted lines for the original version
587   line = line - chunk.currentStart + 1;
588   let count = 0;
589   let deleted = 0;
590   for (const l of chunk.current) {
591     if (l === undefined) {
592       deleted++;
593       if (count === line) break;
594     }
595     continue;
596   }
597   if (count === line) break;
598   count++;
599 }
600
601 return [
602   chunk.previous[line - deleted - 1],
603   chunk.current[line + deleted]
604 ];
605 }
606
607 catch (ex) {
608   // search for the line skipping deleted lines -- since they don't
609   // keep track of the deleted lines for the original version
610   line = line - chunk.currentStart + 1;
611   let count = 0;
612   let deleted = 0;
613   for (const l of chunk.current) {
614     if (l === undefined) {
615       deleted++;
616       if (count === line) break;
617     }
618     continue;
619   }
620   if (count === line) break;
621   count++;
622 }
623
624 return [
625   chunk.previous[line - deleted - 1],
626   chunk.current[line + deleted]
627 ];
628 }
629
630 catch (ex) {
631   // search for the line skipping deleted lines -- since they don't
632   // keep track of the deleted lines for the original version
633   line = line - chunk.currentStart + 1;
634   let count = 0;
635   let deleted = 0;
636   for (const l of chunk.current) {
637     if (l === undefined) {
638       deleted++;
639       if (count === line) break;
640     }
641     continue;
642   }
643   if (count === line) break;
644   count++;
645 }
646
647 return [
648   chunk.previous[line - deleted - 1],
649   chunk.current[line + deleted]
650 ];
651 }
652
653 catch (ex) {
654   // search for the line skipping deleted lines -- since they don't
655   // keep track of the deleted lines for the original version
656   line = line - chunk.currentStart + 1;
657   let count = 0;
658   let deleted = 0;
659   for (const l of chunk.current) {
660     if (l === undefined) {
661       deleted++;
662       if (count === line) break;
663     }
664     continue;
665   }
666   if (count === line) break;
667   count++;
668 }
669
670 return [
671   chunk.previous[line - deleted - 1],
672   chunk.current[line + deleted]
673 ];
674 }
675
676 catch (ex) {
677   // search for the line skipping deleted lines -- since they don't
678   // keep track of the deleted lines for the original version
679   line = line - chunk.currentStart + 1;
680   let count = 0;
681   let deleted = 0;
682   for (const l of chunk.current) {
683     if (l === undefined) {
684       deleted++;
685       if (count === line) break;
686     }
687     continue;
688   }
689   if (count === line) break;
690   count++;
691 }
692
693 return [
694   chunk.previous[line - deleted - 1],
695   chunk.current[line + deleted]
696 ];
697 }
698
699 catch (ex) {
700   // search for the line skipping deleted lines -- since they don't
701   // keep track of the deleted lines for the original version
702   line = line - chunk.currentStart + 1;
703   let count = 0;
704   let deleted = 0;
705   for (const l of chunk.current) {
706     if (l === undefined) {
707       deleted++;
708       if (count === line) break;
709     }
710     continue;
711   }
712   if (count === line) break;
713   count++;
714 }
715
716 return [
717   chunk.previous[line - deleted - 1],
718   chunk.current[line + deleted]
719 ];
720 }
721
722 catch (ex) {
723   // search for the line skipping deleted lines -- since they don't
724   // keep track of the deleted lines for the original version
725   line = line - chunk.currentStart + 1;
726   let count = 0;
727   let deleted = 0;
728   for (const l of chunk.current) {
729     if (l === undefined) {
730       deleted++;
731       if (count === line) break;
732     }
733     continue;
734   }
735   if (count === line) break;
736   count++;
737 }
738
739 return [
740   chunk.previous[line - deleted - 1],
741   chunk.current[line + deleted]
742 ];
743 }
744
745 catch (ex) {
746   // search for the line skipping deleted lines -- since they don't
747   // keep track of the deleted lines for the original version
748   line = line - chunk.currentStart + 1;
749   let count = 0;
750   let deleted = 0;
751   for (const l of chunk.current) {
752     if (l === undefined) {
753       deleted++;
754       if (count === line) break;
755     }
756     continue;
757   }
758   if (count === line) break;
759   count++;
760 }
761
762 return [
763   chunk.previous[line - deleted - 1],
764   chunk.current[line + deleted]
765 ];
766 }
767
768 catch (ex) {
769   // search for the line skipping deleted lines -- since they don't
770   // keep track of the deleted lines for the original version
771   line = line - chunk.currentStart + 1;
772   let count = 0;
773   let deleted = 0;
774   for (const l of chunk.current) {
775     if (l === undefined) {
776       deleted++;
777       if (count === line) break;
778     }
779     continue;
780   }
781   if (count === line) break;
782   count++;
783 }
784
785 return [
786   chunk.previous[line - deleted - 1],
787   chunk.current[line + deleted]
788 ];
789 }
790
791 catch (ex) {
792   // search for the line skipping deleted lines -- since they don't
793   // keep track of the deleted lines for the original version
794   line = line - chunk.currentStart + 1;
795   let count = 0;
796   let deleted = 0;
797   for (const l of chunk.current) {
798     if (l === undefined) {
799       deleted++;
800       if (count === line) break;
801     }
802     continue;
803   }
804   if (count === line) break;
805   count++;
806 }
807
808 return [
809   chunk.previous[line - deleted - 1],
810   chunk.current[line + deleted]
811 ];
812 }
813
814 catch (ex) {
815   // search for the line skipping deleted lines -- since they don't
816   // keep track of the deleted lines for the original version
817   line = line - chunk.currentStart + 1;
818   let count = 0;
819   let deleted = 0;
820   for (const l of chunk.current) {
821     if (l === undefined) {
822       deleted++;
823       if (count === line) break;
824     }
825     continue;
826   }
827   if (count === line) break;
828   count++;
829 }
830
831 return [
832   chunk.previous[line - deleted - 1],
833   chunk.current[line + deleted]
834 ];
835 }
836
837 catch (ex) {
838   // search for the line skipping deleted lines -- since they don't
839   // keep track of the deleted lines for the original version
840   line = line - chunk.currentStart + 1;
841   let count = 0;
842   let deleted = 0;
843   for (const l of chunk.current) {
844     if (l === undefined) {
845       deleted++;
846       if (count === line) break;
847     }
848     continue;
849   }
850   if (count === line) break;
851   count++;
852 }
853
854 return [
855   chunk.previous[line - deleted - 1],
856   chunk.current[line + deleted]
857 ];
858 }
859
860 catch (ex) {
861   // search for the line skipping deleted lines -- since they don't
862   // keep track of the deleted lines for the original version
863   line = line - chunk.currentStart + 1;
864   let count = 0;
865   let deleted = 0;
866   for (const l of chunk.current) {
867     if (l === undefined) {
868       deleted++;
869       if (count === line) break;
870     }
871     continue;
872   }
873   if (count === line) break;
874   count++;
875 }
876
877 return [
878   chunk.previous[line - deleted - 1],
879   chunk.current[line + deleted]
880 ];
881 }
882
883 catch (ex) {
884   // search for the line skipping deleted lines -- since they don't
885   // keep track of the deleted lines for the original version
886   line = line - chunk.currentStart + 1;
887   let count = 0;
888   let deleted = 0;
889   for (const l of chunk.current) {
890     if (l === undefined) {
891       deleted++;
892       if (count === line) break;
893     }
894     continue;
895   }
896   if (count === line) break;
897   count++;
898 }
899
900 return [
901   chunk.previous[line - deleted - 1],
902   chunk.current[line + deleted]
903 ];
904 }
905
906 catch (ex) {
907   // search for the line skipping deleted lines -- since they don't
908   // keep track of the deleted lines for the original version
909   line = line - chunk.currentStart + 1;
910   let count = 0;
911   let deleted = 0;
912   for (const l of chunk.current) {
913     if (l === undefined) {
914       deleted++;
915       if (count === line) break;
916     }
917     continue;
918   }
919   if (count === line) break;
920   count++;
921 }
922
923 return [
924   chunk.previous[line - deleted - 1],
925   chunk.current[line + deleted]
926 ];
927 }
928
929 catch (ex) {
930   // search for the line skipping deleted lines -- since they don't
931   // keep track of the deleted lines for the original version
932   line = line - chunk.currentStart + 1;
933   let count = 0;
934   let deleted = 0;
935   for (const l of chunk.current) {
936     if (l === undefined) {
937       deleted++;
938       if (count === line) break;
939     }
940     continue;
941   }
942   if (count === line) break;
943   count++;
944 }
945
946 return [
947   chunk.previous[line - deleted - 1],
948   chunk.current[line + deleted]
949 ];
950 }
951
952 catch (ex) {
953   // search for the line skipping deleted lines -- since they don't
954   // keep track of the deleted lines for the original version
955   line = line - chunk.currentStart + 1;
956   let count = 0;
957   let deleted = 0;
958   for (const l of chunk.current) {
959     if (l === undefined) {
960       deleted++;
961       if (count === line) break;
962     }
963     continue;
964   }
965   if (count === line) break;
966   count++;
967 }
968
969 return [
970   chunk.previous[line - deleted - 1],
971   chunk.current[line + deleted]
972 ];
973 }
974
975 catch (ex) {
976   // search for the line skipping deleted lines -- since they don't
977   // keep track of the deleted lines for the original version
978   line = line - chunk.currentStart + 1;
979   let count = 0;
980   let deleted = 0;
981   for (const l of chunk.current) {
982     if (l === undefined) {
983       deleted++;
984       if (count === line) break;
985     }
986     continue;
987   }
988   if (count === line) break;
989   count++;
990 }
991
992 return [
993   chunk.previous[line - deleted - 1],
994   chunk.current[line + deleted]
995 ];
996 }
997
998 catch (ex) {
999   // search for the line skipping deleted lines -- since they don't
1000   // keep track of the deleted lines for the original version
1001   line = line - chunk.currentStart + 1;
1002   let count = 0;
1003   let deleted = 0;
1004   for (const l of chunk.current) {
1005     if (l === undefined) {
1006       deleted++;
1007       if (count === line) break;
1008     }
1009     continue;
1010   }
1011   if (count === line) break;
1012   count++;
1013 }
1014
1015 return [
1016   chunk.previous[line - deleted - 1],
1017   chunk.current[line + deleted]
1018 ];
1019 }
1020
1021 catch (ex) {
1022   // search for the line skipping deleted lines -- since they don't
1023   // keep track of the deleted lines for the original version
1024   line = line - chunk.currentStart + 1;
1025   let count = 0;
1026   let deleted = 0;
1027   for (const l of chunk.current) {
1028     if (l === undefined) {
1029       deleted++;
1030       if (count === line) break;
1031     }
1032     continue;
1033   }
1034   if (count === line) break;
1035   count++;
1036 }
1037
1038 return [
1039   chunk.previous[line - deleted - 1],
1040   chunk.current[line + deleted]
1041 ];
1042 }
1043
1044 catch (ex) {
1045   // search for the line skipping deleted lines -- since they don't
1046   // keep track of the deleted lines for the original version
1047   line = line - chunk.currentStart + 1;
1048   let count = 0;
1049   let deleted = 0;
1050   for (const l of chunk.current) {
1051     if (l === undefined) {
1052       deleted++;
1053       if (count === line) break;
1054     }
1055     continue;
1056   }
1057   if (count === line) break;
1058   count++;
1059 }
1060
1061 return [
1062   chunk.previous[line - deleted - 1],
1063   chunk.current[line + deleted]
1064 ];
1065 }
1066
1067 catch (ex) {
1068   // search for the line skipping deleted lines -- since they don't
1069   // keep track of the deleted lines for the original version
1070   line = line - chunk.currentStart + 1;
1071   let count = 0;
1072   let deleted = 0;
1073   for (const l of chunk.current) {
1074     if (l === undefined) {
1075       deleted++;
1076       if (count === line) break;
1077     }
1078     continue;
1079   }
1080   if (count === line) break;
1081   count++;
1082 }
1083
1084 return [
1085   chunk.previous[line - deleted - 1],
1086   chunk.current[line + deleted]
1087 ];
1088 }
1089
1090 catch (ex) {
1091   // search for the line skipping deleted lines -- since they don't
1092   // keep track of the deleted lines for the original version
1093   line = line - chunk.currentStart + 1;
1094   let count = 0;
1095   let deleted = 0;
1096   for (const l of chunk.current) {
1097     if (l === undefined) {
1098       deleted++;
1099       if (count === line) break;
1100     }
1101     continue;
1102   }
1103   if (count === line) break;
1104   count++;
1105 }
1106
1107 return [
1108   chunk.previous[line - deleted - 1],
1109   chunk.current[line + deleted]
1110 ];
1111 }
1112
1113 catch (ex) {
1114   // search for the line skipping deleted lines -- since they don't
1115   // keep track of the deleted lines for the original version
1116   line = line - chunk.currentStart + 1;
1117   let count = 0;
1118   let deleted = 0;
1119   for (const l of chunk.current) {
1120     if (l === undefined) {
1121       deleted++;
1122       if (count === line) break;
1123     }
1124     continue;
1125   }
1126   if (count === line) break;
1127   count++;
1128 }
1129
1130 return [
1131   chunk.previous[line - deleted - 1],
1132   chunk.current[line + deleted]
1133 ];
1134 }
1135
1136 catch (ex) {
1137   // search for the line skipping deleted lines -- since they don't
1138   // keep track of the deleted lines for the original version
1139   line = line - chunk.currentStart + 1;
1140   let count = 0;
1141   let deleted = 0;
1142   for (const l of chunk.current) {
1143     if (l === undefined) {
1144       deleted++;
1145       if (count === line) break;
1146     }
1147     continue;
1148   }
1149   if (count === line) break;
1150   count++;
1151 }
1152
1153 return [
1154   chunk.previous[line - deleted - 1],
1155   chunk.current[line + deleted]
1156 ];
1157 }
1158
1159 catch (ex) {
1160   // search for the line skipping deleted lines -- since they don't
1161   // keep track of the deleted lines for the original version
1162   line = line - chunk.currentStart + 1;
1163   let count = 0;
1164   let deleted = 0;
1165   for (const l of chunk.current) {
1166     if (l === undefined) {
1167       deleted++;
1168       if (count === line) break;
1169     }
1170     continue;
1171   }
1172   if (count === line) break;
1173   count++;
1174 }
1175
1176 return [
1177   chunk.previous[line - deleted - 1],
1178   chunk.current[line + deleted]
1179 ];
1180 }
1181
1182 catch (ex) {
1183   // search for the line skipping deleted lines -- since they don't
1184   // keep track of the deleted lines for the original version
1185   line = line - chunk.currentStart + 1;
1186   let count = 0;
1187   let deleted = 0;
1188   for (const l of chunk.current) {
1189     if (l === undefined) {
1190       deleted++;
1191       if (count === line) break;
1192     }
1193     continue;
1194   }
1195   if (count === line) break;
1196   count++;
1197 }
1198
1199 return [
1200   chunk.previous[line - deleted - 1],
1201   chunk.current[line + deleted]
1202 ];
1203 }
1204
1205 catch (ex) {
1206   // search for the line skipping deleted lines -- since they don't
1207   // keep track of the deleted lines for the original version
1208   line = line - chunk.currentStart + 1;
1209   let count = 0;
1210   let deleted = 0;
1211   for (const l of chunk.current) {
1212     if (l === undefined) {
1213       deleted++;
1214       if (count === line) break;
1215     }
1216     continue;
1217   }
1218   if (count === line) break;
1219   count++;
1220 }
1221
1222 return [
1223   chunk.previous[line - deleted - 1],
1224   chunk.current[line + deleted]
1225 ];
1226 }
1227
1228 catch (ex) {
1229   // search for the line skipping deleted lines -- since they don't
1230   // keep track of the deleted lines for the original version
1231   line = line - chunk.currentStart + 1;
1232   let count = 0;
1233   let deleted = 0;
1234   for (const l of chunk.current) {
1235     if (l === undefined) {
1236       deleted++;
1237       if (count === line) break;
1238     }
1239     continue;
1240   }
1241   if (count === line) break;
1242   count++;
1243 }
1244
1245 return [
1246   chunk.previous[line - deleted - 1],
1247   chunk.current[line + deleted]
1248 ];
1249 }
1250
1251 catch (ex) {
1252   // search for the line skipping deleted lines -- since they don't
1253   // keep track of the deleted lines for the original version
1254   line = line - chunk.currentStart + 1;
1255   let count = 0;
1256   let deleted = 0;
1257   for (const l of chunk.current) {
1258     if (l === undefined) {
1259       deleted++;
1260       if (count === line) break;
1261     }
1262     continue;
1263   }
1264   if (count === line) break;
1265   count++;
1266 }
1267
1268 return [
1269   chunk.previous[line - deleted - 1],
1270   chunk.current[line + deleted]
1271 ];
1272 }
1273
1274 catch (ex) {
1275   // search for the line skipping deleted lines -- since they don't
1276   // keep track of the deleted lines for the original version
1277   line = line - chunk.currentStart + 1;
1278   let count = 0;
1279   let deleted = 0;
1280   for (const l of chunk.current) {
1281     if (l === undefined) {
1282       deleted++;
1283       if (count === line) break;
1284     }
1285     continue;
1286   }
1287   if (count === line) break;
1288   count++;
1289 }
1290
1291 return [
1292   chunk.previous[line - deleted - 1],
1293   chunk.current[line + deleted]
1294 ];
1295 }
1296
1297 catch (ex) {
1298   // search for the line skipping deleted lines -- since they don't
1299   // keep track of the deleted lines for the original version
1300   line = line - chunk.currentStart + 1;
1301   let count = 0;
1302   let deleted = 0;
1303   for (const l of chunk.current) {
1304     if (l === undefined) {
1305       deleted++;
1306       if (count === line) break;
1307     }
1308     continue;
1309   }
1310   if (count === line) break;
1311   count++;
1312 }
1313
1314 return [
1315   chunk.previous[line - deleted - 1],
1316   chunk.current[line + deleted]
1317 ];
1318 }
1319
1320 catch (ex) {
1321   // search for the line skipping deleted lines -- since they don't
1322   // keep track of the deleted lines for the original version
1323   line = line - chunk.currentStart + 1;
1324   let count = 0;
1325   let deleted = 0;
1326   for (const l of chunk.current) {
1327     if (l === undefined) {
1328       deleted++;
1329       if (count === line) break;
1330     }
1331     continue;
1332   }
1333   if (count === line) break;
1334   count++;
1335 }
1336
1337 return [
1338   chunk.previous[line - deleted - 1],
1339   chunk.current[line + deleted]
1340 ];
1341 }
1342
1343 catch (ex) {
1344   // search for the line skipping deleted lines -- since they don't
1345   // keep track of the deleted lines for the original version
1346   line = line - chunk.currentStart + 1;
1347   let count = 0;
1348   let deleted = 0;
1349   for (const l of chunk.current) {
1350     if (l === undefined) {
1351       deleted++;
1352       if (count === line) break;
1353     }
1354     continue;
1355   }
1356   if (count === line) break;
1357   count++;
1358 }
1359
```

Stubbing in the Vue Files



Vue Files We'll Need



Login – no registration page needed



Transactions – list of transactions for the current month



Edit Transaction – add new and edit existing transactions



About – as time allows – not critical for “MVP”



Reminder: We're using Webpack



Add Vue.js directly to .html file(s)

Single File Components

Can include other Components

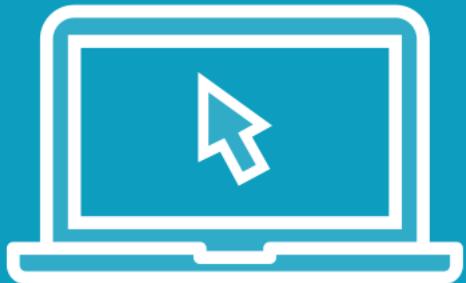
Structure of a Component

- Template
- Script
- Style

Webpack bundles this together for us!



Demo



main.js

App.vue

Components

- Home.vue
- Transactions.vue
- EditTransaction.vue
- Login.vue
- About.vue
- Etc.

Build with Webpack

Review output

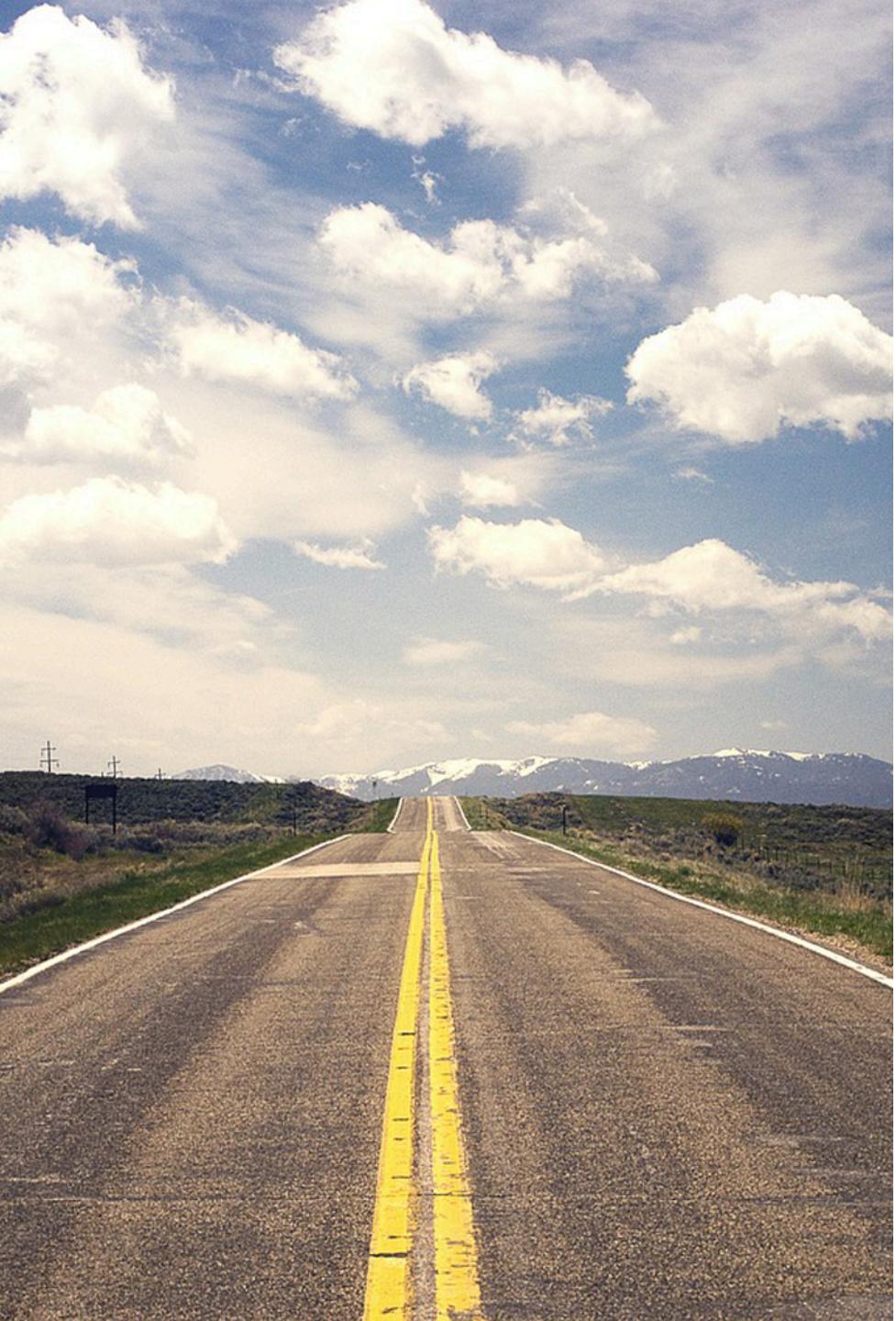


Blank Slide



Setting Up Routing





Setup Routes

- Home (default route)
- Login
- About

Transactions

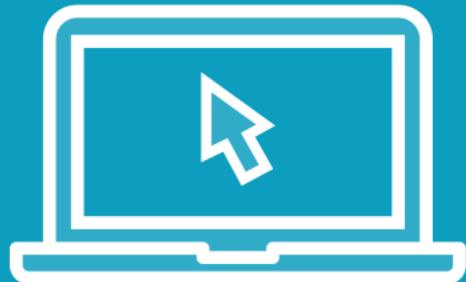
- Child component in Home.vue
- Route not needed

Edit Transaction

- Route not needed
- We'll use modal for this



Demo



“router” folder already added by vue-cli

One route file

- index.js

Official router package: vue-router

Setup routes

Default “fallback” route

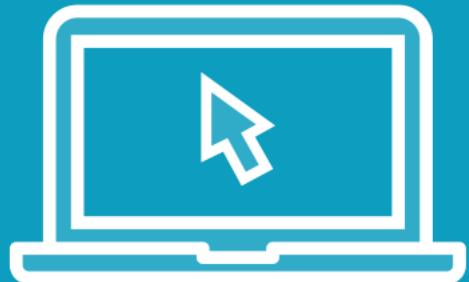
Time to develop some routes!



Blank Slide



Demo



Baseline setup done

Basic architectural structure in place

- Single-file Vue components
- Routes

Time to build and test run the app

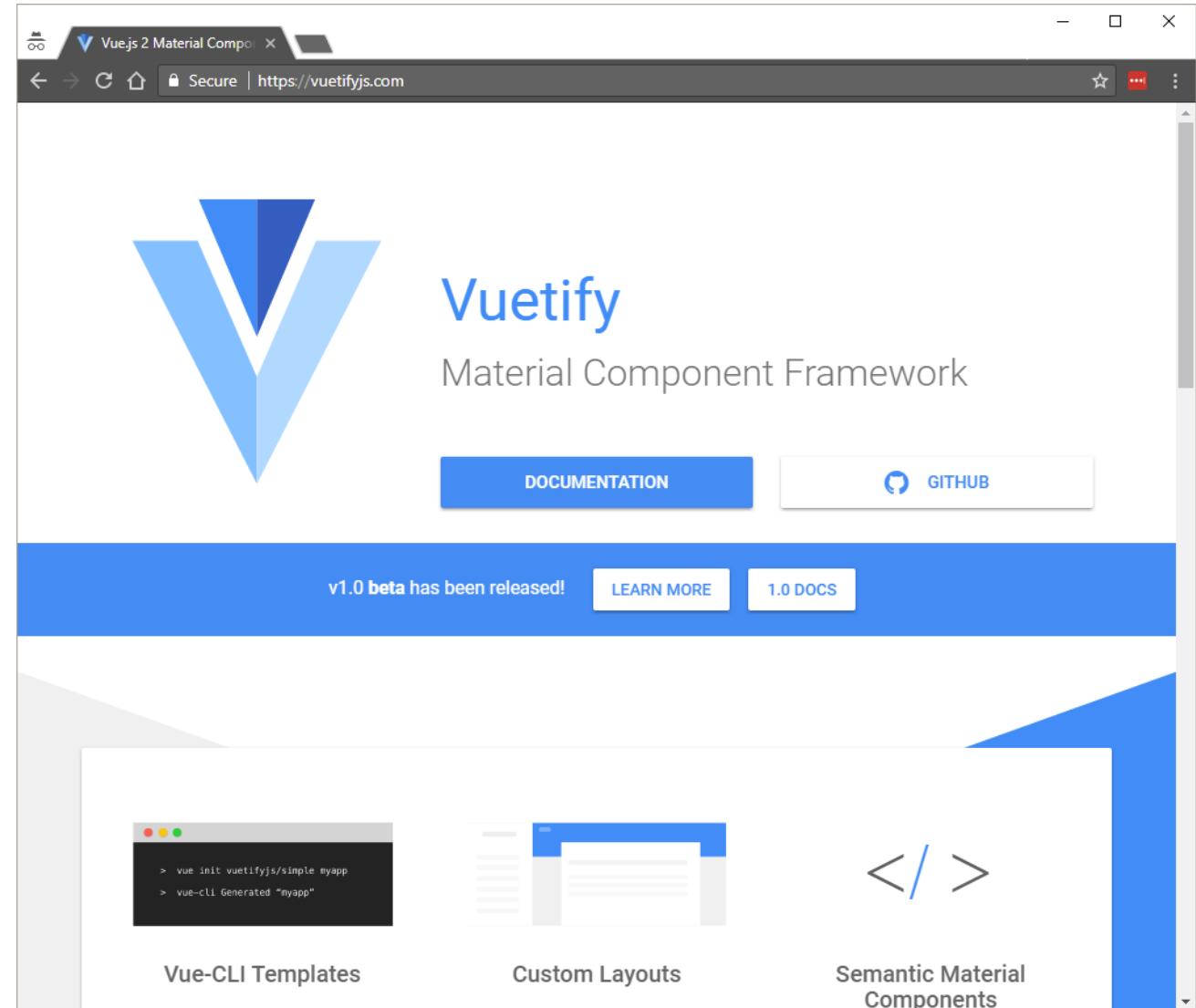
- Verify routes
- Test nested components in Home.vue



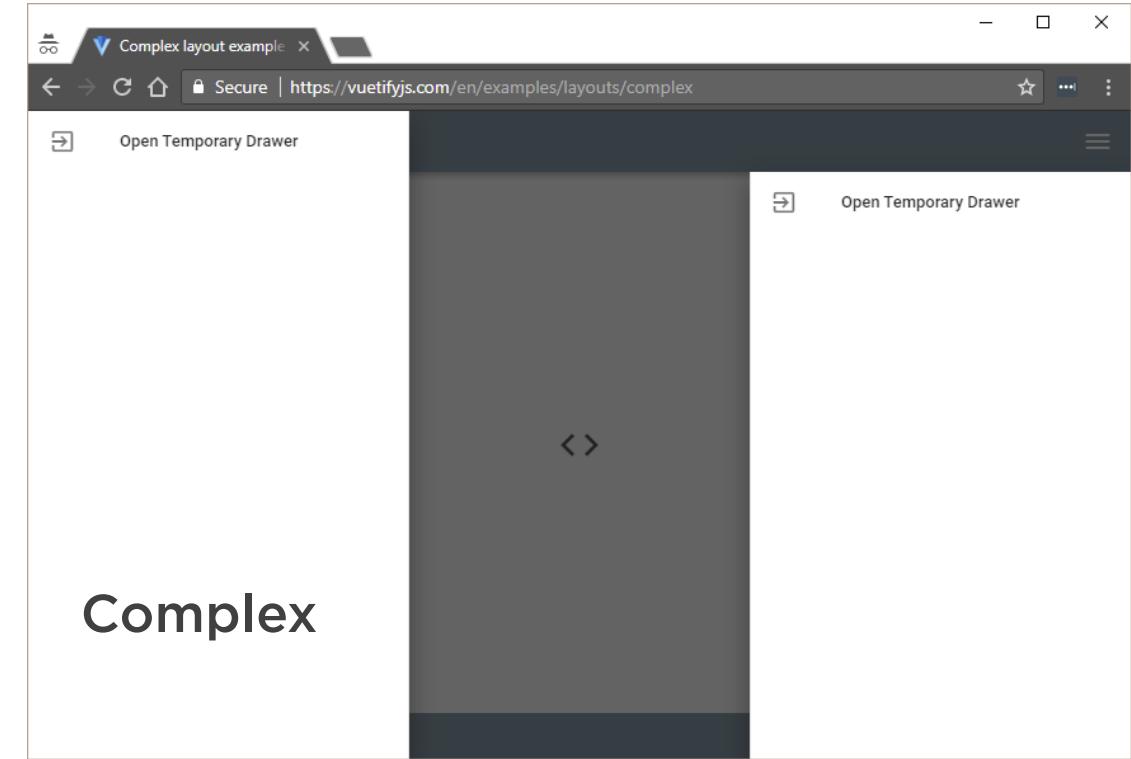
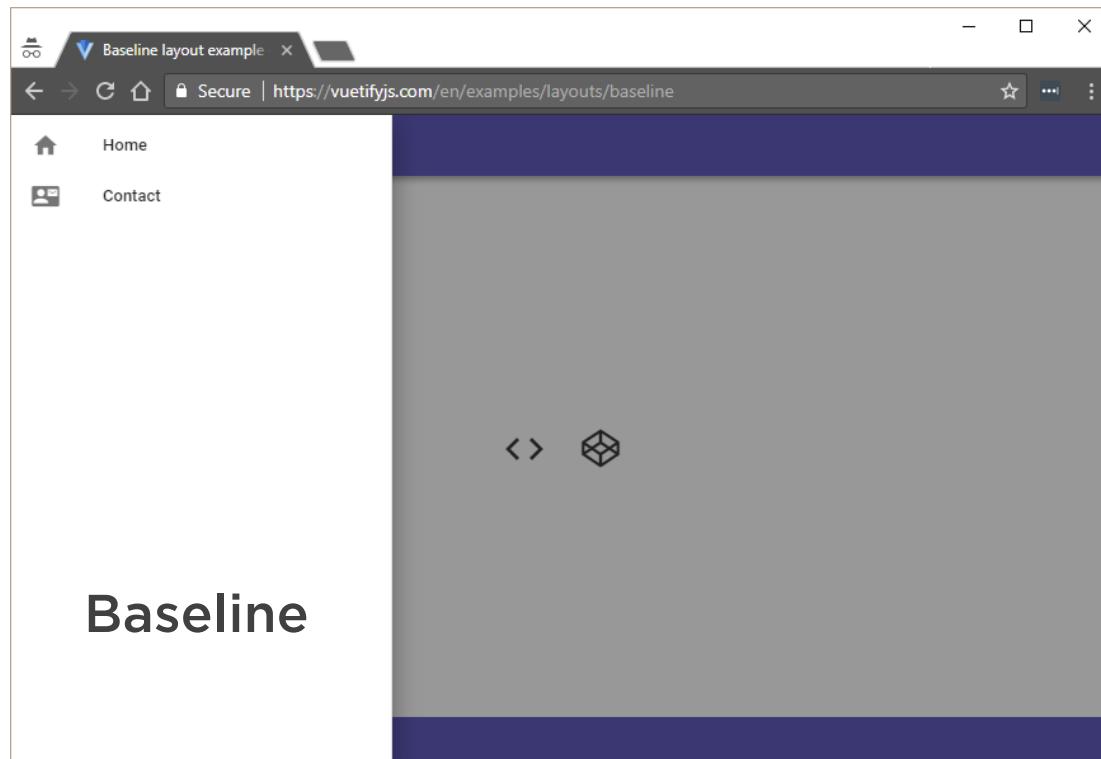
Selecting a Vuetify Layout



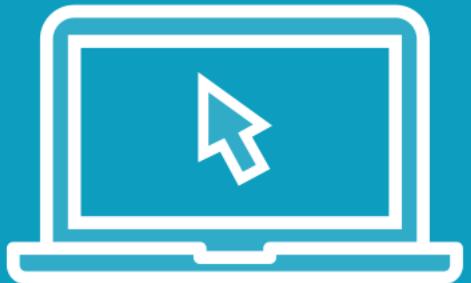
Vuetify
vuetifyjs.com



Vuetify Pre-Defined Layouts



Demo



Preview pre-defined layouts

Select a layout for our project

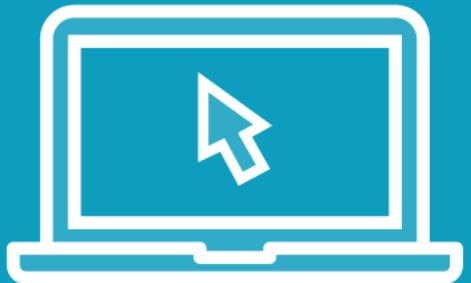


Blank Slide

Bump into Chrome here to show vuetify website



Demo



Add Vuetify

Add selected layout

- Do a test run
- Fix any issues
- Import Vuetify styles
- Import material design icons



Blank Slide

Bump into Visual Studio Code here...



Vuetify Components Review





We've already built components

Component are like building blocks

Examples

- Buttons
- Dialogs
- Data tables
- Forms

Vuetify includes a large number of UI components



Blank Slide

Bump into Chrome here to show vuetify website



Using Vuetify Components



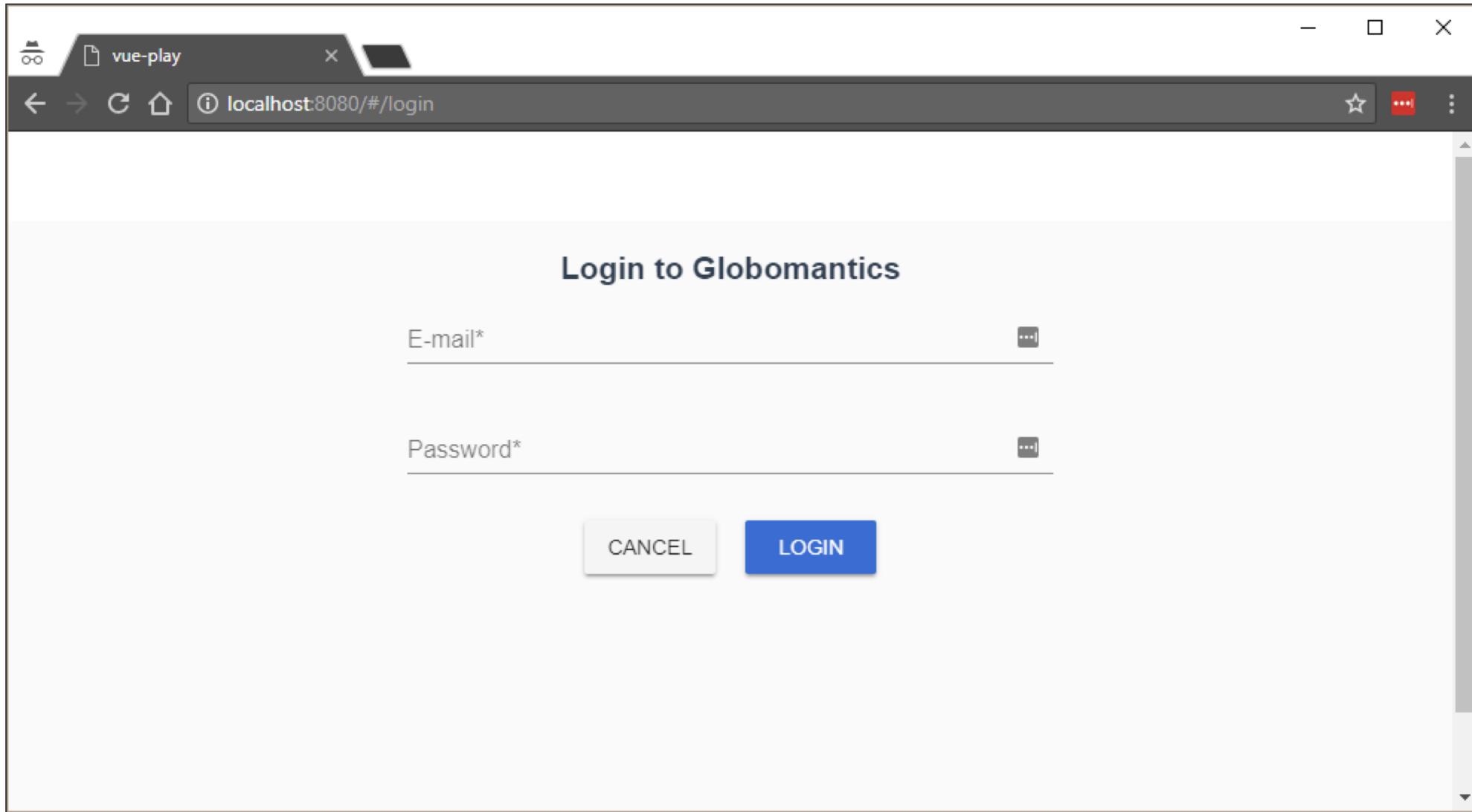
Components We Need



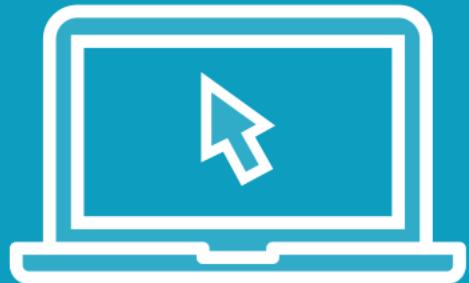
Button
Data Tables
Dialogs
Forms
Icons
Pickers
Snackbars



Login



Demo



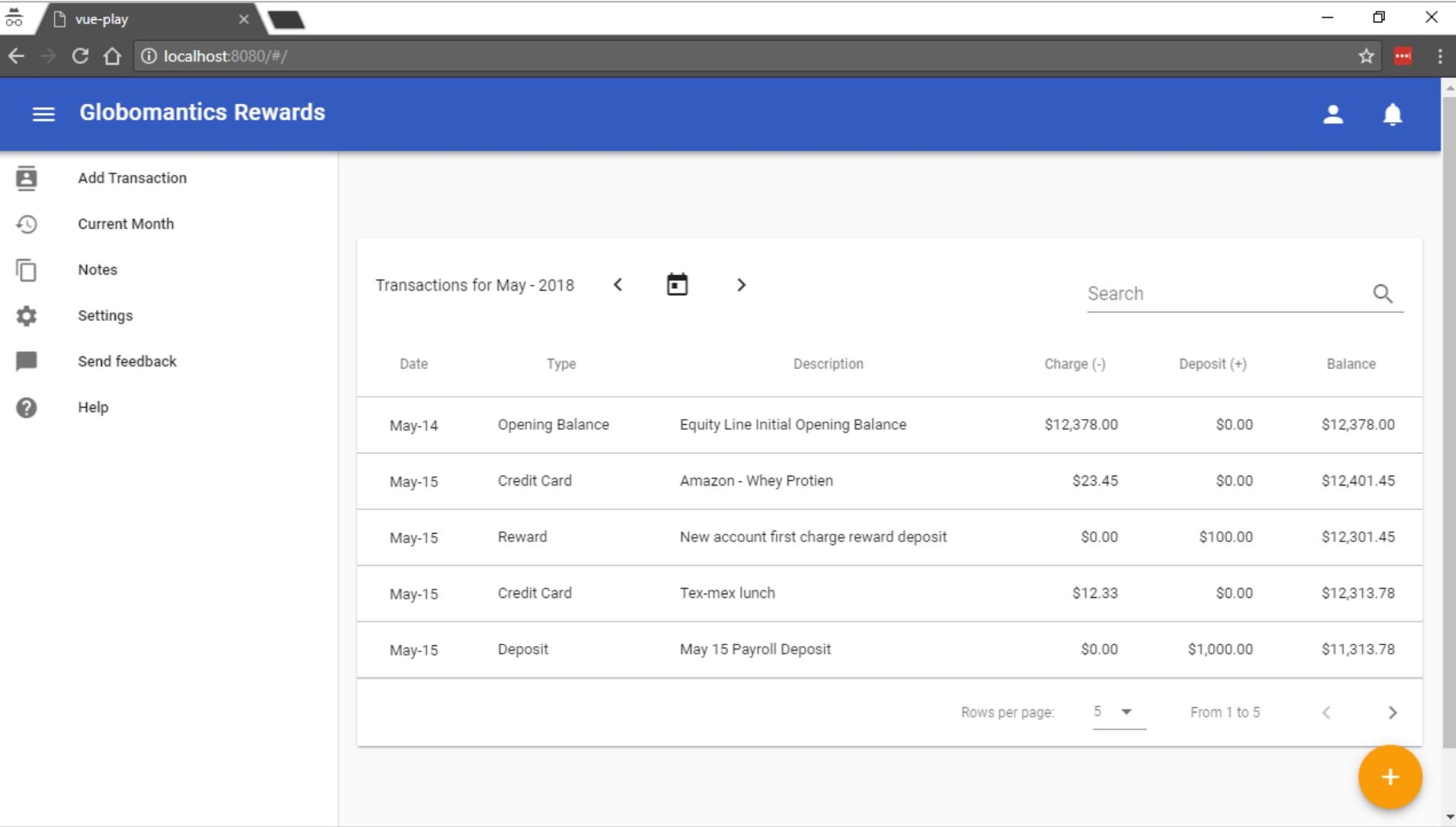
Using Vuetify Components
“Vuetiful”



Review Vuetified Components



Home and Transactions



The screenshot shows a web application interface for 'Globomantics Rewards' running on a local server at `localhost:8080/`. The left sidebar contains navigation links: 'Add Transaction', 'Current Month', 'Notes', 'Settings', 'Send feedback', and 'Help'. The main content area displays a table of transactions for May 2018. The table has columns for Date, Type, Description, Charge (-), Deposit (+), and Balance. The transactions listed are:

Date	Type	Description	Charge (-)	Deposit (+)	Balance
May-14	Opening Balance	Equity Line Initial Opening Balance	\$12,378.00	\$0.00	\$12,378.00
May-15	Credit Card	Amazon - Whey Protein	\$23.45	\$0.00	\$12,401.45
May-15	Reward	New account first charge reward deposit	\$0.00	\$100.00	\$12,301.45
May-15	Credit Card	Tex-mex lunch	\$12.33	\$0.00	\$12,313.78
May-15	Deposit	May 15 Payroll Deposit	\$0.00	\$1,000.00	\$11,313.78

At the bottom right of the main content area is a large orange button with a white plus sign (+). The bottom right corner of the image features a circular icon with a play button symbol.

EditTransaction

The screenshot shows a web application for managing rewards transactions. The main interface is titled "Globomantics Rewards". A sidebar on the left contains links for "Add Transaction", "Current Month", "Notes", "Settings", "Send feedback", and "Help". The main content area is a "New Transaction" dialog with the following fields:

- Transaction Date: 2018-04-28
- Transaction Type: (dropdown menu)
- Description: (text input)
- Amount: 0 (with a minus sign and a plus sign)
- Notes: (text input)

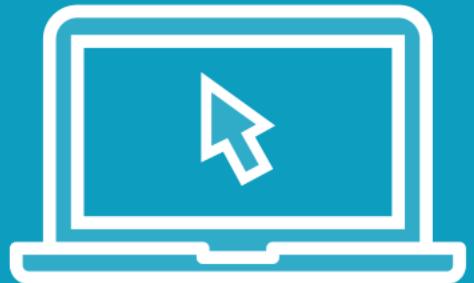
At the bottom of the dialog are "CANCEL" and "SAVE" buttons. To the right of the dialog is a table showing a transaction history:

Deposit (+)	Balance
\$0.00	\$12,378.00
\$0.00	\$12,401.45
\$100.00	\$12,301.45
\$0.00	\$12,313.78
\$1,000.00	\$11,313.78

At the bottom of the table, it says "From 1 to 5" with navigation arrows. A large orange button with a plus sign is located at the bottom right of the table area. The browser address bar shows "localhost:8080/#/" and the tab title is "vue-play".



Demo



Review Vuetified Components

- Home
- Transactions
- EditTransaction



Next Steps



Vuetify has been setup

Built the login component

Reviewed other key components

However...

- Component state is local currently
- Global application state is needed

