DZone
A DEVADA MEDIA PROPERTY

THE 2018 DZONE GUIDE TO

Dynamic Web & Mobile Application Development

VOLUME V

RESEARCH PARTNER SPOTLIGHT

GrapeCity®

Key Research Findings

BY JORDAN BAKER - CONTENT COORDINATOR, DEVADA

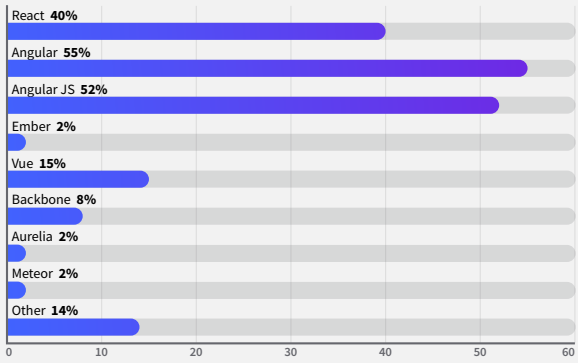
DEMOGRAPHICS

For this year's DZone Guide to Dynamic Web and Mobile Application Development, we surveyed our community of software professionals to get their thoughts on the state of the field. We received 1,202 responses, with a 64% completion rating. Based on the number of responses, we have calculated the margin of error for this survey at 4%.

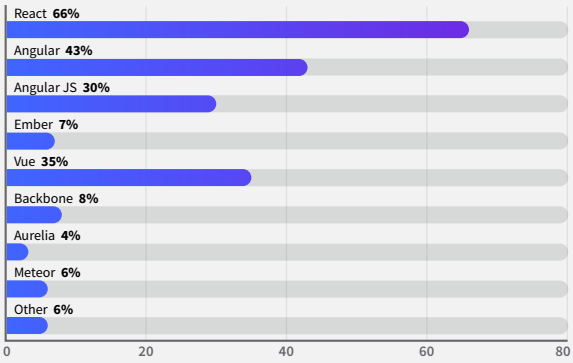
Below is a quick snapshot of the demographics of those surveyed.

- The average respondent has 14 years of experience in the industry.
- Respondents reported four main geographical areas where their companies are located:
  - 32%: USA
  - 25%: Europe
  - 12%: South America
  - 11%: South-Central Asia
- Respondents reported working in one of three main verticals:
  - 20% work for a software vendor
  - 19% are employed in the finance/banking industry
  - 10% work on e-commerce platforms
- Most survey-takers work in enterprise-sized organizations:
  - 24% for organizations sized 100-999
  - 21% for organizations sized 1,000-9,999
  - 21% for organizations sized 10,000+
- A majority of respondents work on immediate teams of ten people or less:
  - 33% work on a team of 6-10 people
  - 32% work on a team of 2-5 people
  - 14% work on a team of 11-15 people
- Most respondents work in one of three mains roles:
  - 41% are developers/engineers
  - 21% identify as developer team leads
  - 16% are employed as architects
- Respondents reported developing three main types of applications:
  - 85% develop web applications/services (SaaS)
  - 50% develop enterprise business apps
  - 27% develop native mobile applications
- Several important programming language ecosystems were reported:
  - 82% reported working in the Java ecosystem
  - 77% reported working in the client-side JavaScript ecosystem
  - 42% reported working in the Node.js ecosystem
  - 33% reported working in the Python ecosystem
  - 32% reported working in the C# ecosystem

1. WHICH OF THESE CLIENT-SIDE JAVASCRIPT FRAMEWORKS *HAVE* YOU USED?



2. WHICH OF THESE CLIENT-SIDE JAVASCRIPT FRAMEWORKS ARE YOU *INTERESTED* IN USING?



- Java, however, dominated the main programming languages used:
  - 60% use Java
  - 11% use C#
  - 11% use JavaScript (client- and/or server-side)
  - 7% use Python

FULL-STACK DEVELOPMENT

There exist three main forms of web application development: full-stack, front-end, and back-end. Among respondents, 48% work more with full-stack applications, 44% work with back-end apps, and only 7% concentrate on front-end applications. Additionally, there proved three main languages used among respondents: JavaScript (84%), HTML/CSS (75%), and Java (71%). The dominance of these three languages leads to the conclusion that respondents who develop full-stack applications mostly use Java as their backend language (though, as we'll see, the use of Node.js is increasing), JavaScript for DOM manipulation and other front-end logic, and HTML/CSS for styling.

For the rest of this section, we'll divide full-stack development into its constituent parts: front-end and back-end. We'll examine how the developers in our response population create these two types of applications.

FRONT-END WEB DEVELOPMENT: JAVASCRIPT FLAVORS AND FRAMEWORKS

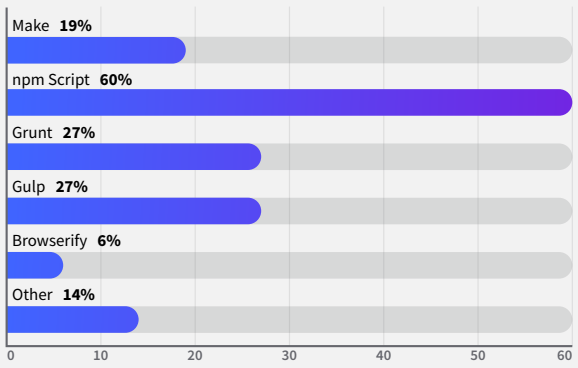
As noted above, JavaScript proved the most popular language for web application development among respondents. JavaScript, however, differs from many other popular development languages in that there is no one, true JavaScript. While the ECMAScript standard has been established, there exist several supersets under the main JavaScript umbrella other than ECMAScript, such as TypeScript, CoffeeScript, and Elm.

Of all these "flavors" of JavaScript, though, two established themselves as the most popular among our respondents: TypeScript (64%) and ES6 (58%). Interestingly, when we asked which flavor of JavaScript respondents are interested in (rather than which ones they've used), the percentages around TypeScript stayed rather static, but the percentages for ES6 fell by 10%, going from 58% who use it to 48% who are interested in it. And, though its adoption rate is low, CoffeeScript garnered a fair amount of interest as compared to its usage statistics. 12% of respondents reported using CoffeeScript, whereas 19% reported an interest in learning more about the language.

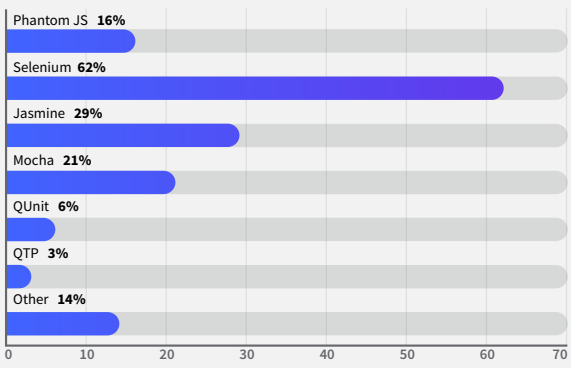
When setting up a front-end development environment, languages are only one part of the equation. Frameworks play an extremely important role in the creation of front-end apps, and three have come to dominate the landscape: React, Vue, and Angular. When we asked which of these frameworks respondents have used, 55% told us Angular, 52% reported AngularJS (i.e. the 1.x version of the Angular framework), 40% said React, and 15% said Vue. Much like we saw with the flavors of JavaScript, however, the frameworks developers are interested in differs from those they have used. When asked which client-side JavaScript framework they're interested in using, 66% reported React, 43% said Angular, 35% told us Vue, and 30% said AngularJS.

The steep drop-off in AngularJS is not surprising given that the framework is now on version seven and has since switched to using TypeScript as its primary language. The dramatic increase in React and Vue, though, is of interest. React saw a 26% increase (40% reporting to have used it and 66% reported to be interested) and Vue went up by 20% (15% used vs. 35% interested in). Interestingly, current versions of Angular dropped 12% between these two categories. While the unpopularity of Angular as compared to React at first seems surprising given the wide

3. WHAT BUILD SCRIPT TOOLS DO YOU USE?



4. WHAT DO YOU USE TO TEST YOUR WEB APPS?



spread adoption of Angular, this actually adheres to trends in the wider developer community. Earlier in 2018, Stack Overflow released a survey report that took into account the responses of over 100,000 developers and technologists. In this survey, React finished as the most loved web development framework and garnered the second most votes of any development framework (only behind TensorFlow). Additionally, React finished as the most wanted framework in this poll.

When we compare the popular flavors of JavaScript to the main JavaScript-based frameworks used in front-end development, we see some interesting trends. Among respondents who code in React, 74% use ES6 and 74% use TypeScript. Among Vue developers, 83% use ES6 and 64% use TypeScript. While both of these frameworks were originally designed to be used with ES6-style JavaScript, the popularity of TypeScript has caused support for the language to appear in both of these frameworks. Docs on TypeScript support for Vue.js can be found here and for React.js here. It is interesting to note how significantly more popular TypeScript has proven thus far among React developers than Vue developers. This difference could simply be due to the fact that React is backed by Facebook and has had more resources to put into their support of TypeScript, from both a coding and community marketing perspective.

BACKEND DEVELOPMENT

THE RISE OF NODE.JS

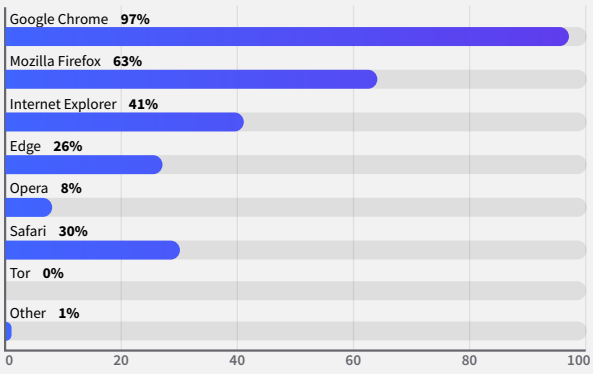
90% of respondents reported using JavaScript on the client-side; as discussed above, this is to be expected. Interestingly, 41% of respondents use JavaScript on the server-side, up from the 36% of respondents who targeted the server-side with their JavaScript in our *2017 DZone Guide to Web Development*. This high adoption rate of JavaScript on the back-end correlates with the increase in the usage rates for the Node.js runtime. In this year's survey, 42% of survey takers reported that their organization uses the Node.js ecosystem. This is up from 35% in 2017, nearly mirroring the growth rate of server-side JavaScript over the past year.

Of those respondents who work with the Node.js ecosystem, 54% do so on full-stack development projects and 40% on backend development projects. When we correlate our data on respondents who work in the Node.js ecosystem with our data on databases used for web applications, non-relational (or NoSQL) databases have higher rates of adoption among Node.js developers. For respondents whose organizations use Node.js, 56% use MongoDB. For those who report using the Java ecosystem, 43% use MongoDB; among Python ecosystem users, 49% use MongoDB; and with C# ecosystem adopters, 37% choose MongoDB as their database. This higher rate of adoption for MongoDB among Node developers versus those who work in other backend environments such as Java, Python, and C# could well be due to the asynchronicity of both technologies. Additionally, many developers use Mongoose, an open-source data schema solution, with MongoDB (source). Mongoose utilizes the JavaScript language, thus allowing Node.js developers to write their database logic in the same language as their application logic. For a more detailed discussion on databases, see the *2018 DZone Guide to Databsaes: Relational and Beyond*.

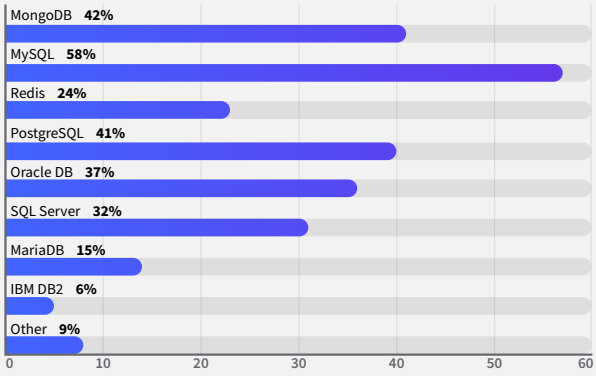
SERVER-SIDE OPERATIONS

The back-end of an application is a complicated place, full of API and database calls, logic, and more. When we asked how respondents typically divide their work between client and server, 76% told us the server-side exposes APIs, 70% have the server-side perform business logic, and 66% use the server-side to integrate systems such as databases, message queues, and EIS. Despite the growth of Node.js noted above, Java remains the dominant language for performing such operations on the backend. Thus, for the rest of this section, we'll use the

5. WHICH BROWSERS DO YOU ACTIVELY DEVELOP FOR?



6. WHAT DATABASES ARE YOU USING WITH YOUR WEB APPS?



statistics gathered from respondents who use Java to build web applications as our means of comparative analysis.

Despite the popularity of MongoDB among Node.js developers noted above, web developers working in Java seem to prefer traditional SQL databases. Among those respondents who told us they build web apps with Java, 61% use MySQL databases, 46% use Oracle DB, 45% use MongoDB, and 43% use PostgreSQL. Comparing these numbers to the adoption rates of these databases among the general survey population, Oracle DB proved more popular among Java-based web developers. Among the general survey population, 58% use MySQL, 42% use MongoDB, 41% use PostreSQL, and 37% use Oracle DB for their database needs. One potential explanation for Oracle DB's higher than average popularity among Java-based web developers is that both the Java language and Oracle DB are developed by the same organization and would thus be made to work well together.

When it comes to pushing data to the server, 67% of the general survey population use the WebSocket API, 34% use HTTP streaming, 25% use webhooks, another 25% use polling, and 19% reported using server-sent events. When we compare these numbers to our Java-based web developers, these percentages all dramatically fall. Among Java web developers, 37% use the WebSocket API, 19% use HTTP streaming, 14% use polling, 12% use webhooks, and 11% use server-sent events.

The adoption rate of the three most popular web servers among respondents (Apache Tomcat, Apache Web Server, and NGINX), also differed between the general survey population and Java web developers, though not as dramatically as in the case of the means of pushing data to the server. Among the general population, 62% reported using Apache Tomcat as their web server, 50% said they use the Apache Web Server, and 55% reported using NGINX. Among those respondents who use Java to build web apps, 75% use Apache Tomcat, 52% Apache Web Server, and 44% NGINX.

ENVIRONMENTS TARGETED: JAVASCRIPT AS A LANGUAGE FOR MOBILE DEVELOPMENT

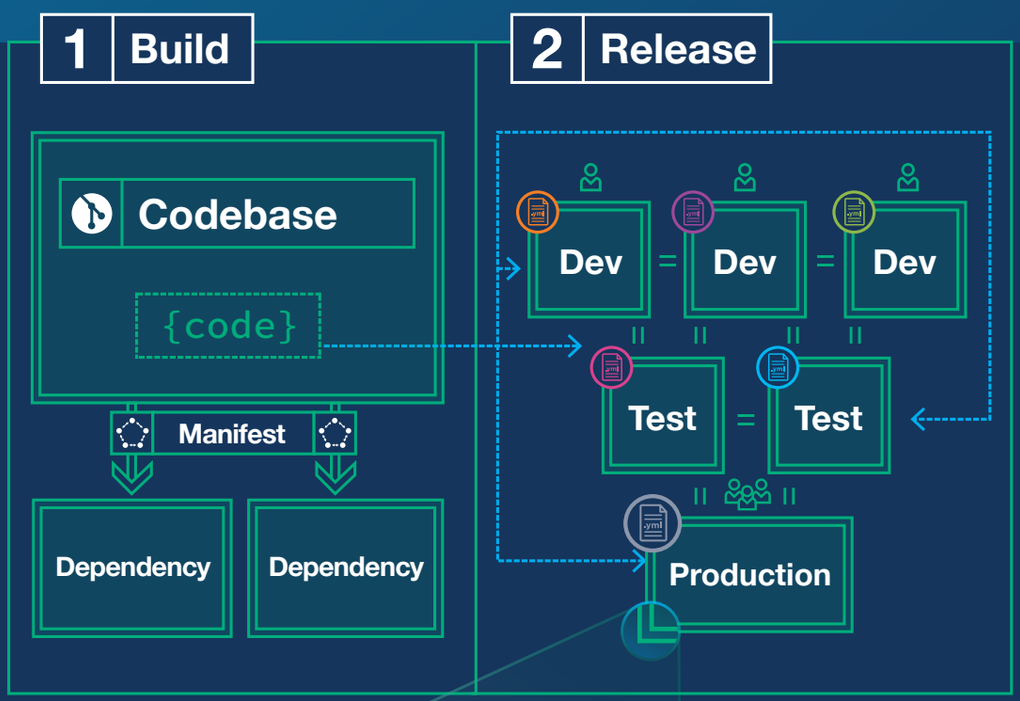
When it comes to the environments targeted by web developers, one browser proved almost unanimous among respondents. 97% of survey takers reported actively developing for Google's Chrome browser. The runner up, Mozilla Firefox, is targeted by 64% of developers, followed by Internet Explorer at 41%, Safari at 30%, and Edge at 26%. Clearly, Chrome dominates the landscape of desktop browser-based development efforts. Interestingly, the results prove much the same when we look at the browsers targeted by hybrid application developers. 99% of respondents who develop hybrid applications develop for Chrome, 68% for Mozilla Firefox, 40% for Internet Explorer, 37% for Safari, and 28% for Edge.

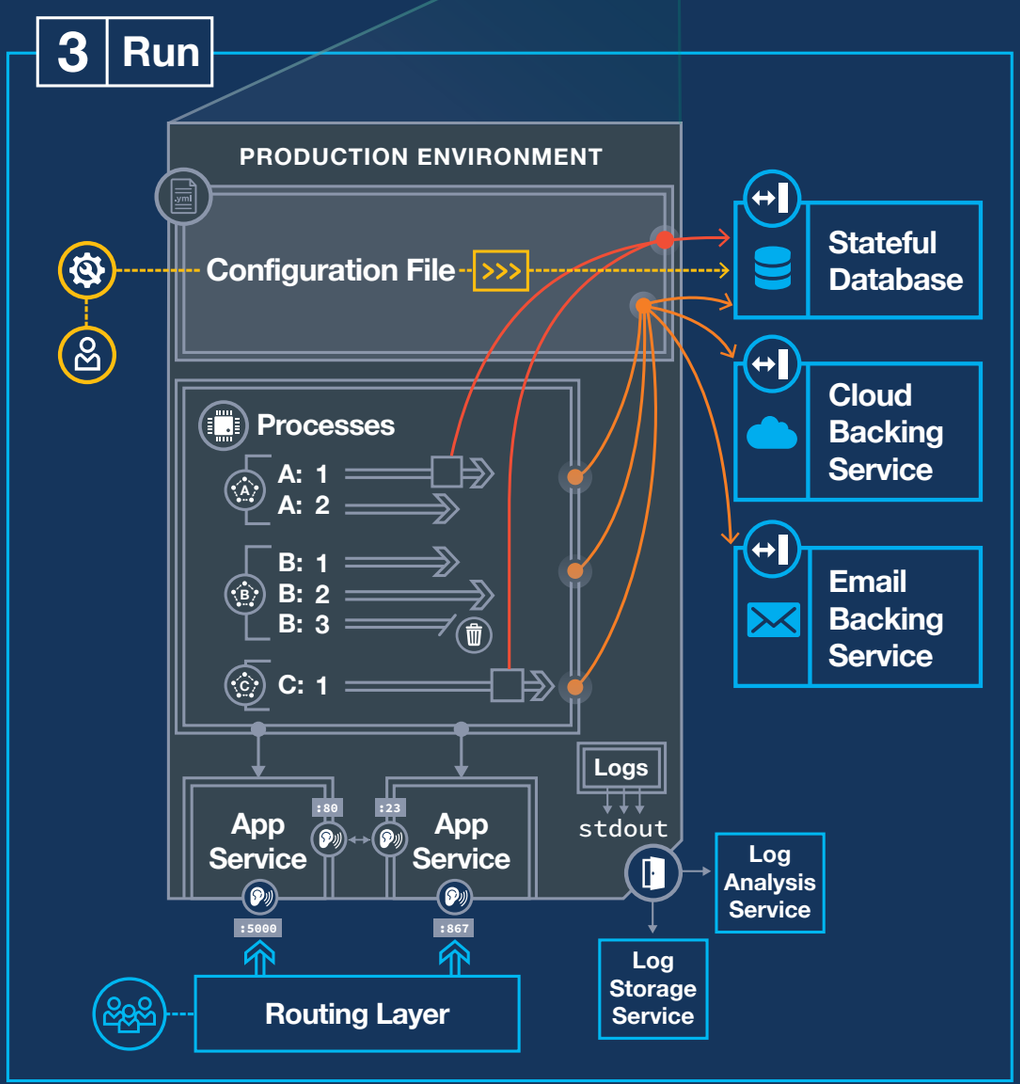
Hybrid and native application developers target more than just the browser; they also have to worry about how their applications perform on various operating systems. 82% of respondents who develop native or hybrid applications do so for Android and 59% for iOS. Another 25% of native and hybrid app developers focus their efforts on React Native. While the prominence of Android and iOS is unsurprising, the proliferation of React Native as a popular mobile application development framework has led to an increase in the use of JavaScript as a language for mobile development.

When we asked which languages respondents are currently using for building native or hybrid mobile apps, 72% reported using JavaScript and 68% said Java, while there were only 24 write-in responses for Swift (the official language for developing iOS software) and 5 or Objective-C (a popular language for iOS development). When we compare the two main languages, JavaScript and Java, to our data on mobile platforms, an interesting pattern emerges. Among those respondents who develop for Android, 74% use Java and 65% use JavaScript. Among those respondents who develop for iOS, 77% use JavaScript and 65% use Java. This high percentage of JavaScript in both Android and iOS development seems due largely to frameworks such as React Native. According to React Native's documentation, it allows developers to "build a real mobile app that's indistinguishable from an app built using Objective-C or Java." This most assuredly accounts for the remarkably low use of Swift and Objective-C among this year's respondents. Diving into this data a little further, though, reveals that JavaScript may be more popular as a language for hybrid application development. Among respondents who develop hybrid applications, 87% use JavaScript and 63% use Java. For survey takers who create native mobile apps, 75% use Java and 61% use JavaScript.

The 12-Factor App

Modern web applications run in heterogeneous environments, scale elastically, update frequently, and depend on independently deployed backing services. Modern application architectures and development practices must be designed accordingly. The PaaS-masters at Heroku summarized lessons learned from building hundreds of cloud-native applications into the twelve factors visualized below.





the 12 factors

Codebase
One codebase, many deploys, strict version control

Dependencies
Explicitly declare and isolate dependencies

Configuration
Store config in each deploy environment, preferably using environmental variables

Backing Services
Treat backing services as resources (neutral as to local vs. third-party) located via config

Build, Release, Run
Strictly separate build, release, and run; never change code at runtime

Processes
Keep all processes stateless and share-nothing; store state (with other persistent data) in a stateful backing service

Port Binding
Bind every service to a port and listen on that port; don't rely on runtime server injection

Concurrency
Distinguish process types (e.g. web, background worker, utility) and scale each type independently

Disposability
Make processes start up quickly (<4s from launch to ready) and shut down safely (for web process: stop listening, finish current requests, exit; for worker process: return current job to work queue)

Dev/Prod Parity
Maximize dev/prod parity by minimizing gaps in time (between deploys: hours), personnel (authors=deployers), and environment (use adapters for backing services)

Logs
Log by writing all output streams to stdout; rout streams using non-app services

Admin Processes
Run one-off/admin processes (db migration, REPL , one-time scripts) in same environment as normal processes

diving deeper

INTO **DYNAMIC WEB & MOBILE APPLICATION DEVELOPMENT**

zones

Web Dev dzone.com/webdev
Web professionals make up one of the largest sections of IT audiences; we are collecting content that helps Web professionals navigate in a world of quickly changing language protocols, trending frameworks, and new standards for UX.

Performance dzone.com/performance
Scalability and optimization are constant concerns for the Developer and Operations manager. The Performance Zone focuses on all things performance, covering everything from database optimization to garbage collection to tweaks to keep your code as efficient as possible.

Security dzone.com/security
The Security Zone covers topics related to securing applications and the machines that run them. The goal of the Zone is to help prepare the people who make and support those applications stay up to date on industry best practices, remain aware of new and omnipresent threats, and help them to think "security-first."

twitter


@umaar


@AddyOsmani


@SaraSoueidan


@Swizec


@sophiebits


@plavookac


@sarah_edo


@leisa


@rauschma


@rachelandrew

refcardz

Essential Liferay
Includes web content management, workflow, administration, hot tips, and more.

Node.js
This updated Refcard introduces Node, explains how it works, and dives into its architecture. Learn how to install and use Node to its full potential and access a list of the most commonly used APIs.

Introduction to Web Components
Download this Refcard to learn more about web components. Java Champion Kito Mann walks you through the process from set up to using helpful libraries.

podcasts

JS Party
Recording live every Thursday, JS Party explores JavaScript and the web in general.

Shop Talk Show
This self-proclaimed "internet show about the internet" will teach you about various web development topics through interviews with people working in web dev.

CodePen Radio
Learn about a wide range of topics related to web dev with episodes about debugging, feature testing, serverless, and more.

books

PHP and MySQL Web Development
Learn how to use PHP and MySQL to produce web applications, how to work with a MySQL database, and how to use PHP to interact with your database and server.

JavaScript and JQuery: Interactive Front-End Web Development
Learn about basic programming concepts, core elements of the JavaScript language, JQuery, and more.

Android Programming: The Big Nerd Ranch Guide
Learn how to use hands-on example apps in Android studio to create apps that can integrate with other apps, download and display pictures from the web play sounds, and more.

Expanding Developers' Capabilities With Complete JavaScript Components

SpreadJS 12 brings a major feature enhancement to your favorite JavaScript spreadsheet component: shapes! What will you do with data-driven shapes?

Comprised of both built-in and custom shapes, this new feature gives you the power to enhance your spreadsheets and applications with data-driven graphics and flowcharts. With 60+ built-in Excel-like shapes, you can:

- Import and export shapes-based Excel documents seamlessly
- Draw flowcharts with connectors
- Create annotations with arrows
- Add action buttons like play or skip to your apps
- Generate Gantt charts from a spreadsheet schedule

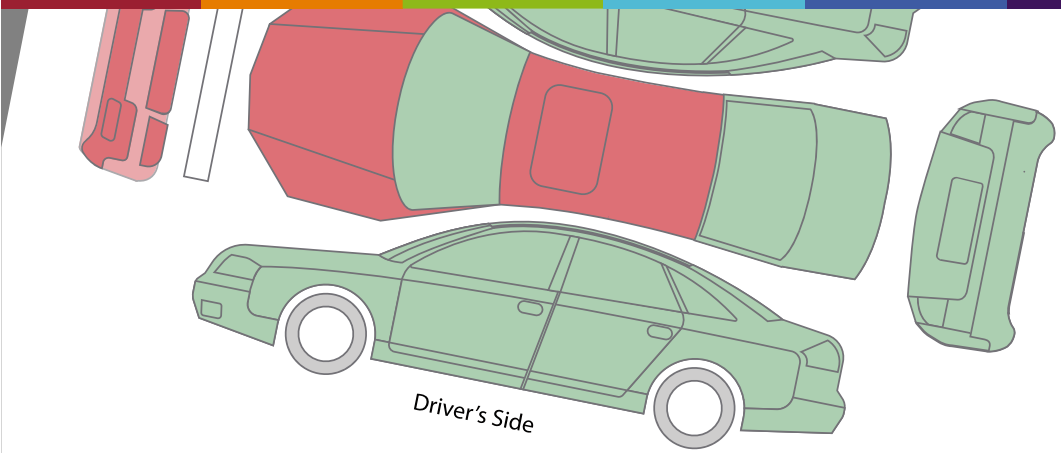
SpreadJS custom shapes can be implemented in many different ways. You can draw shapes to your specifications and add interactions like highlights, information callouts, or take some database action.

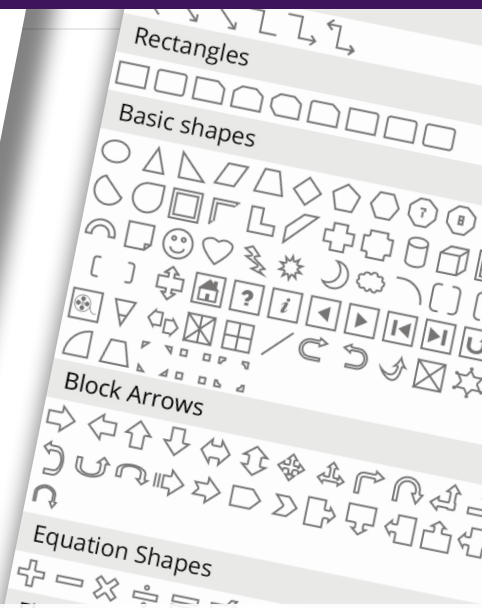
The shapes feature can be used to make different kinds of interactive diagrams. You can create a visual, interactive floor plan that allows users to see who sits at a desk or add information about amenities. Construct a production floor plan for a manufacturing facility, and highlight areas experiencing slowdowns or problems. You can also design an interactive, touch-based map of a car so users can highlight damage for insurance claims.

Power up your application with both built-in and custom shapes that enhance your spreadsheets with data-driven graphics, flowcharts, Gantt charts, and annotations.

GrapeCity's family of products provides developers, designers, and architects with the ultimate collection of easy-to-use tools for building sleek, high-performing, feature-complete applications. In addition to SpreadJS, GrapeCity's JavaScript offerings include Wijmo, high-performance HTML5/JavaScript UI controls for building enterprise-grade applications. Wijmo provides a complete collection of time-saving HTML5/JavaScript UI controls for building touch-first and lightweight applications. Wijmo's high-speed UI controls include FlexGrid, the industry's best Angular data grid.

WRITTEN BY JODY HANDLEY
PRODUCT MARKETING MANAGER AT GRAPECITY

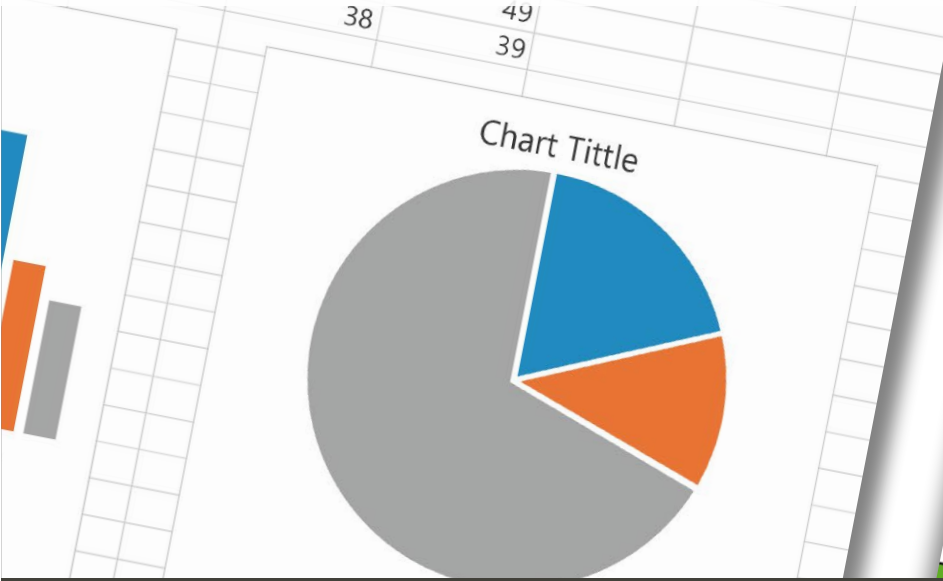


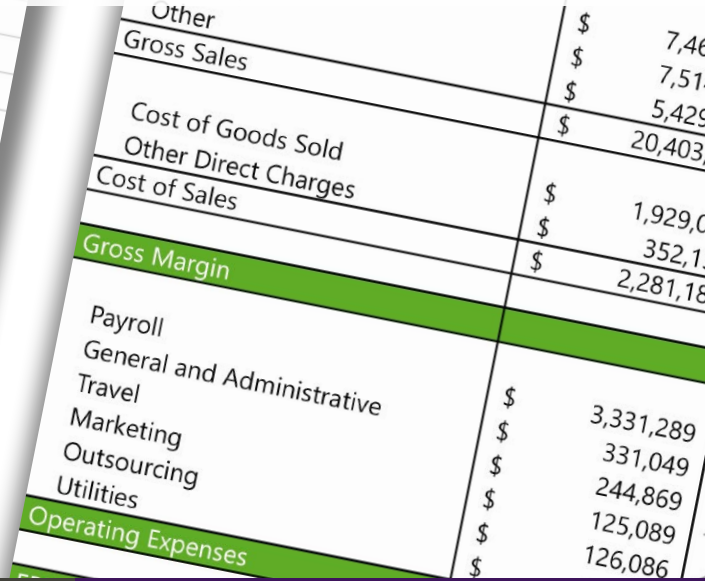


SpreadJS

Wijmo

JavaScript Components for Enterprise Web Apps





Download your free trial at **GrapeCity.com**

GrapeCity®