To make it easy to update the website without causing problems, the responsibilities can be broken down as follows:

Responsibility 1: Set up automatic ways to test changes to the website.

Implement automated testing tools and frameworks such as Jest, Mocha, or Selenium to test website functionality and performance.
Write test scripts to cover critical functionalities, including user interactions, form submissions, and data retrieval.
Integrate continuous integration (CI) and continuous deployment (CD) pipelines using platforms like Jenkins, Travis CI, or GitHub Actions to automate the testing and deployment process.
Ensure that tests run automatically whenever changes are pushed to the code repository to catch any potential issues early in the development cycle.
Responsibility 2: Make sure new changes don't break what's already there.

Conduct thorough regression testing before deploying new changes to the production environment.
Create a staging environment that closely mirrors the production environment to test changes in a controlled environment.
Use version control systems like Git to track changes and rollback to previous versions if needed.
Implement feature flags or toggles to gradually roll out new features and monitor their impact on the website's performance and user experience.
Encourage developers to follow coding best practices and guidelines to minimize the risk of introducing bugs or breaking existing functionality.
Responsibility 3: Put new changes on the website for everyone to see.

Use a version control system (e.g., Git) to manage code changes and collaborate with team members.
Deploy changes to the production environment using a robust deployment pipeline with automated deployment scripts.
Monitor the deployment process and ensure that changes are successfully propagated to the live website without any disruptions.
Communicate updates and changes to stakeholders, users, or clients through release notes, email notifications, or announcements on the website.
Monitor website performance and user feedback after deployment to identify any issues or areas for improvement.
By implementing these strategies, you can ensure a smooth and efficient process for updating the website while minimizing the risk of causing problems or disruptions. Automated testing, continuous integration, and careful deployment practices help maintain the stability and reliability of the website while enabling rapid iteration and improvement.