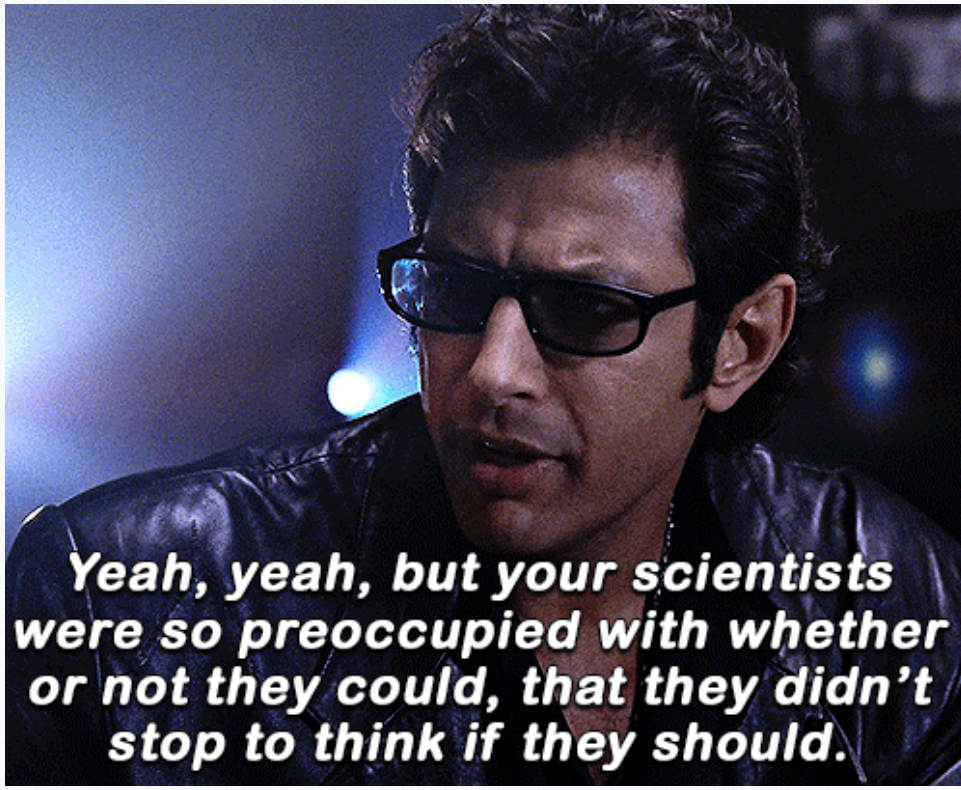
Custom Code

Just Because You Can,
Doesn't Mean You Should.



WHO AM I?

Software Engineer

Little Rock, AR

The Crossing, EPC

WHAT IS THIS ABOUT?

What is considered "custom" code?

What is the **true cost** of customizing?

When should you customize?

Agenda

1 **EXAMPLES OF CUSTOMIZATION**

2 **THE COST OF CUSTOMIZATION**

3 **A FOUNDATION FOR SUSTAINABLE CUSTOMIZATION**

- Key Roles
- Choosing a Development Team
- Common Misconceptions

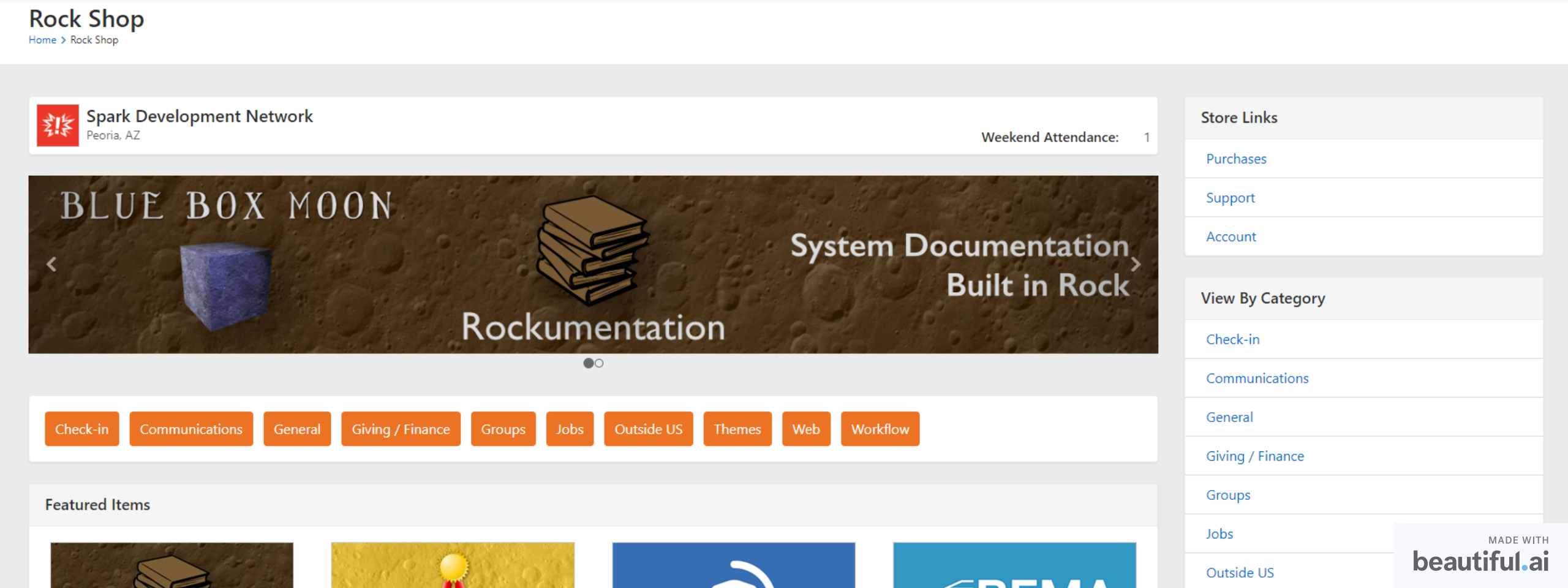
4 **QUESTIONS TO ASK BEFORE YOU CUSTOMIZE**

- Assessing Your Risk With Customization
- Deciding to Customize

Examples of Customization

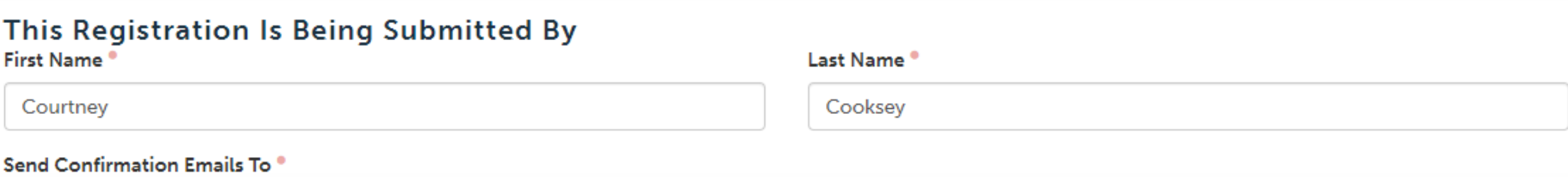
Plugins

Anything you can buy in the Rock Shop or steal from another church's GitHub. Hopefully supported by someone, but someone isn't Spark.



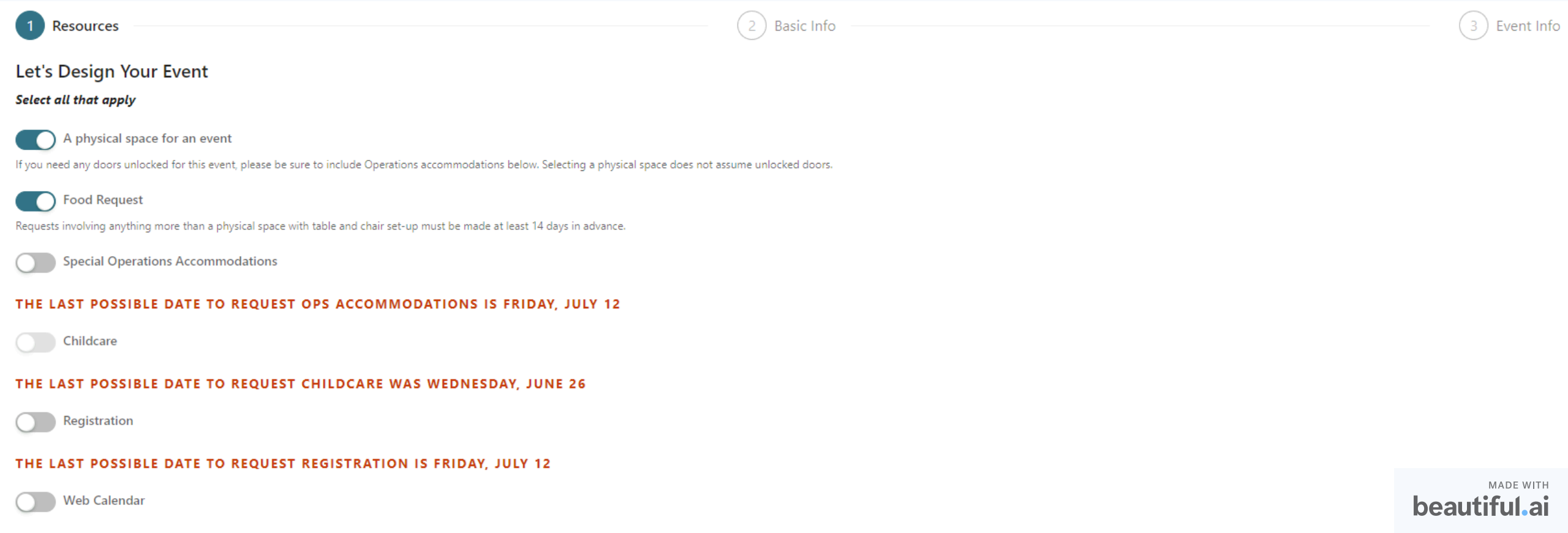
Changes to Core Blocks

After the 8th time a senior pastor failed to complete an event registration we changed the wording from "The Registration *Was Completed By*" to "*Is Being Submitted By*"



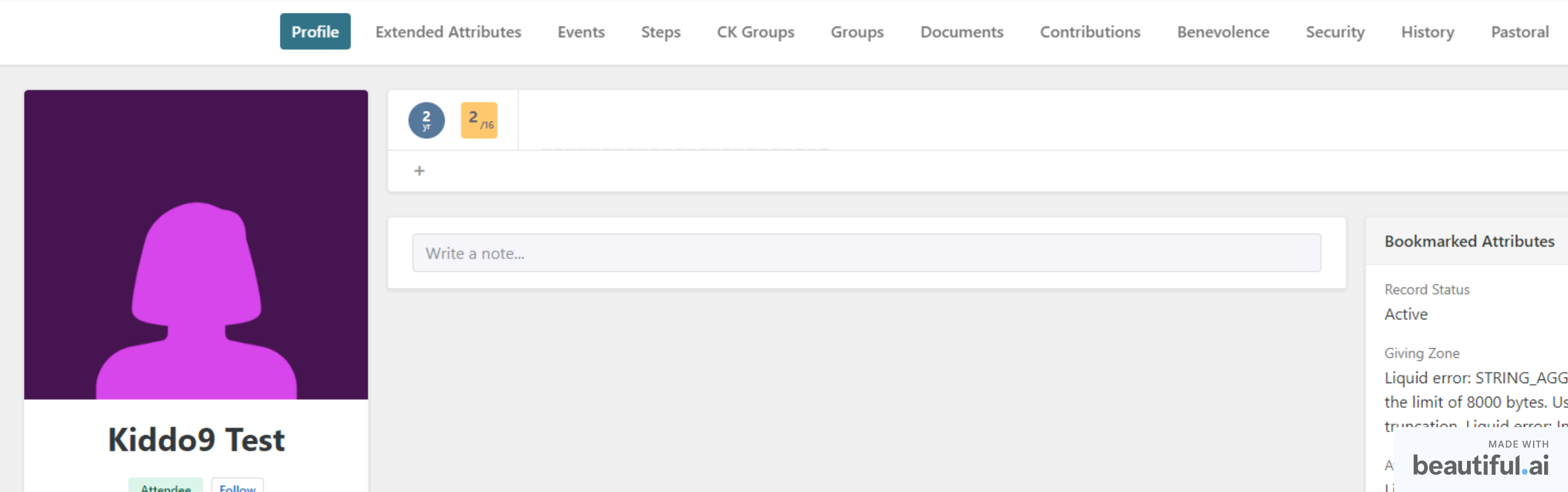
Custom Blocks

We realized the hours we spent telling staff, "no, you can't add childcare 24 hours before a 300 person event", was enough that it was worth creating a custom form that wouldn't let them ask in the first place.



The Core Utilities You've Used to Create Your Own Product

How much have you changed the default Person Profile page? Have you added additional tabs to the page?



Why Shouldn't We Customize Everything, All the Time?

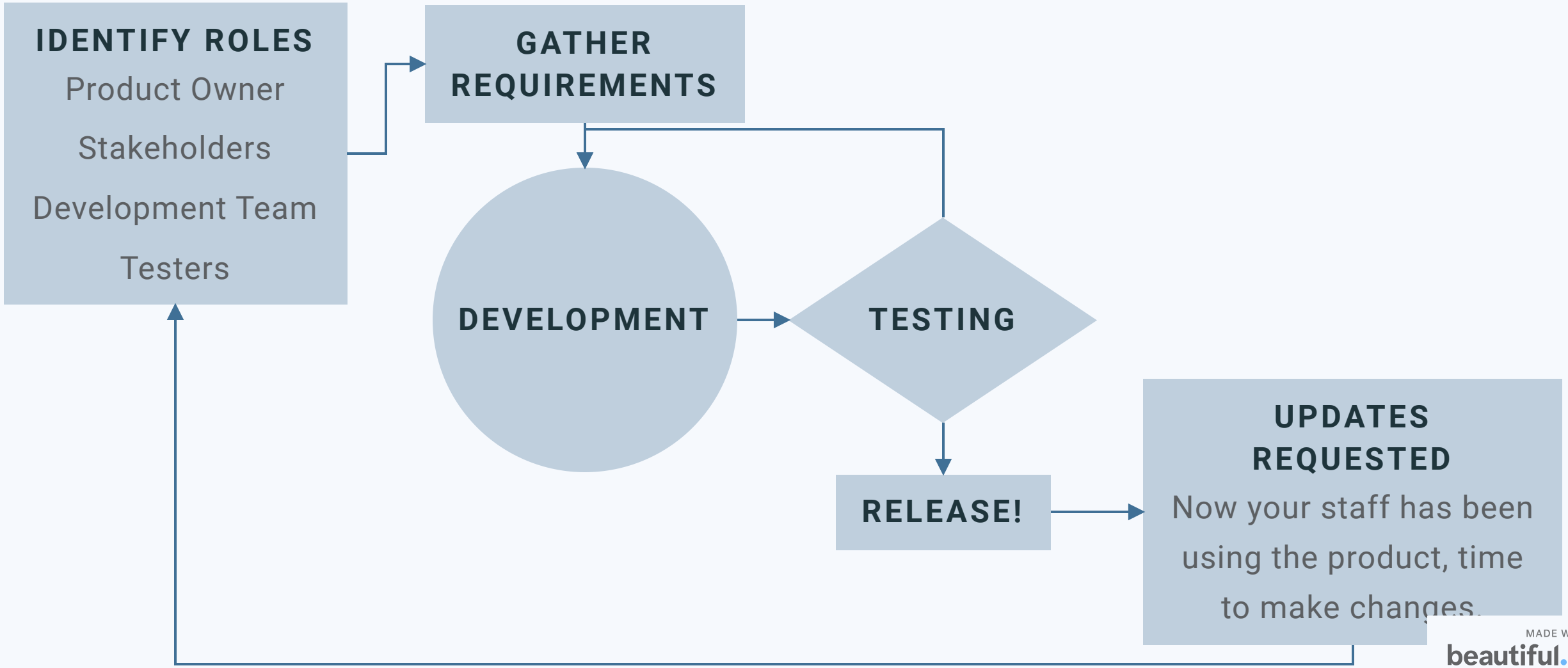
UNEXPECTED TECHNICAL DEBT

WELL, WELL, WELL...IF IT ISN'T THE CONSEQUENCES OF MY OWN ACTIONS.

The Cost of Customization

Project Lifecycle

A ministry has a problem, how do we create a product that solves their issue?

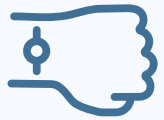


Failure in Requirements Gathering Can Doom a Project Before it Starts

- **ACCURATE DISCOVERY TAKES EXPERIENCE AND EXPERTISE**
- **A CLEAR ISSUE THAT NEEDS TO BE ADDRESSED**
  Projects without a clear goal often fall apart and waste resources.
- **SOMEONE TO ARTICULATE THAT PROBLEM TO AN EXPERT**
  Beware of discovery telephone.
- **AN EXPERT TO PROVIDE THE SOLUTION**

Those Requirements Add Up

- **TIMEFRAME**
  A solution needed in two weeks is a different product than a solution needed in two months.



- **OPTIMIZATION**
  Is there complex logic? Will it be used by one staff member a few times a year? Or hundreds of public users every day?

- **EXTERNAL DATA**
  Is there another application you need to pull data from? Send data to?

Creating the Product

- **DO YOU HAVE A TEAM THAT CAN CREATE THE *RIGHT* SOLUTION FOR YOU?**
  A team without a lot of Rock knowledge might be able to build something that works, but it won't be something that lasts.

- **DID YOUR DEVELOPMENT TEAM HAVE THE TOOLS THEY NEEDED FOR DISCOVERY?**
  Don't let your development team find out about requirements half way through building the product.

Development is a Team Sport

- **YOUR DEVELOPMENT TEAM ARE NOT THE END USERS**

  No matter how well you explain the problem to your team, they aren't the ones who will use the tool. *You have to test the product* and make sure it is what you need.

- **YOU WILL ALWAYS BE SURPRISED BY USER BEHAVIOR**

  "I can't submit an event. The form is broken!"

  "...You didn't pick a date."

Custom Code Requires Custom Upgrading

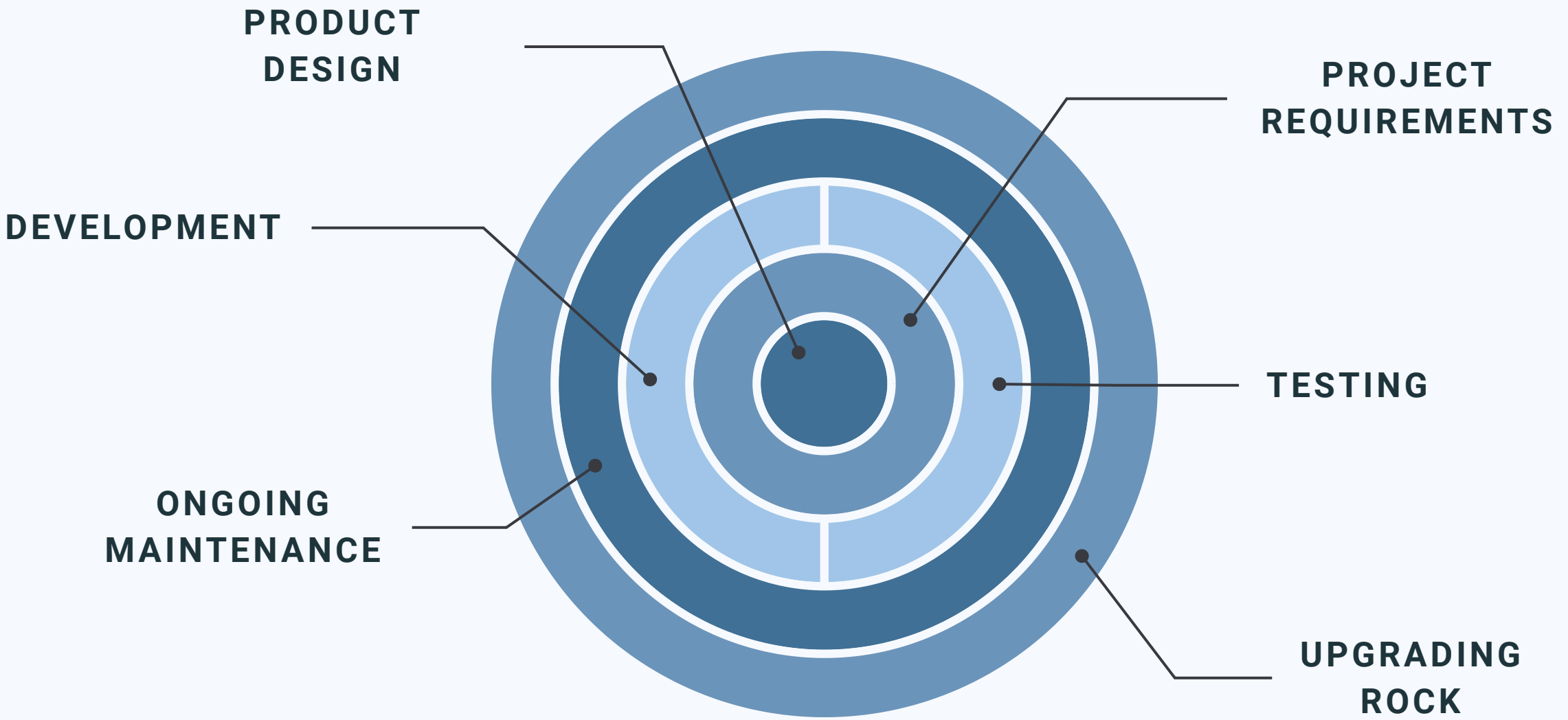
- **UPGRADING WITHOUT TESTING IS ASKING FOR FAILURE**

  Technology is constantly evolving. When Microsoft stops supporting libraries, Rock can't keep safely using them. Rock has to update its framework which leads to breaking changes.

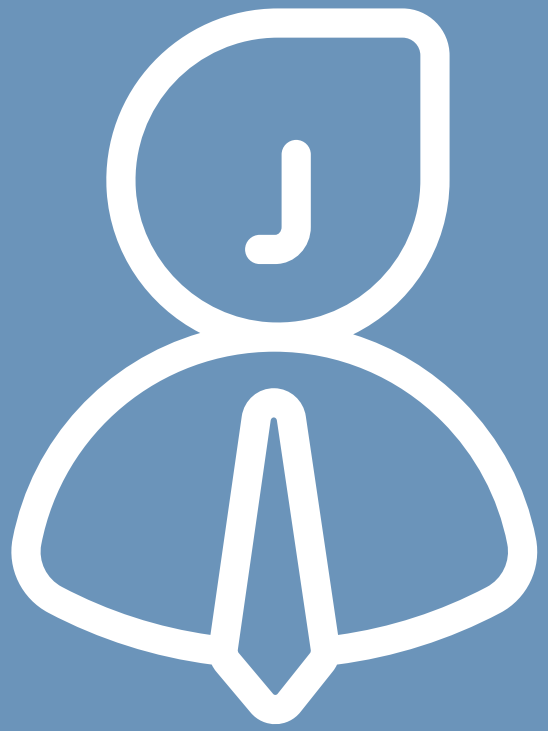
- **SPARK IS NOT RESPONSIBLE FOR CUSTOM CODE**

  The core Rock code base is 11.5 GB of over 100K files. Spark doesn't have the access or responsibility to maintain anything outside of that.

Your Initial Idea Quickly Multiplies



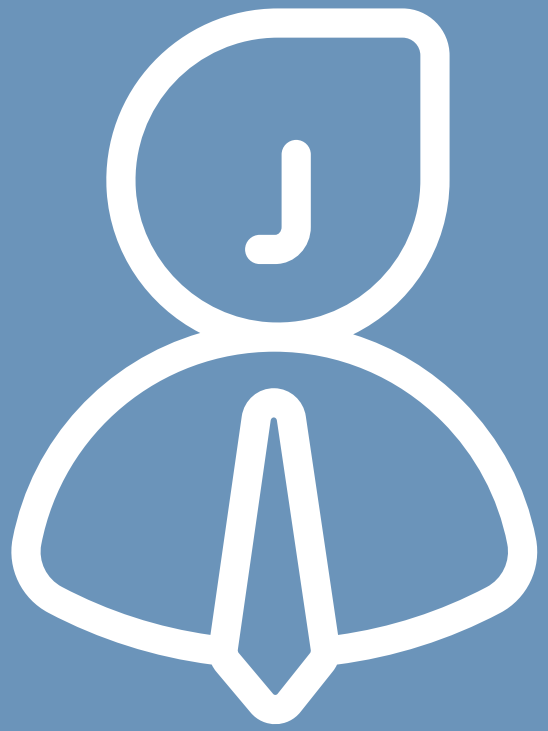
A Foundation for Sustainable Customization



Product Owner Role

- **WHO OWNS ROCK?**
  Someone needs to be in charge of the overall vision and management of your Rock instance. They need authority and expertise.

- **WHO OWNS THE PROJECT?**
  Every time you start a new project for a ministry, you must identify a Product Owner for the project. They need to be ready to be actively involved.



Product Owner Responsibilities

- **PROJECT SCOPE**
  The Product Owner decides what tasks are involved in a project and what the desired end product is.

- **TASK SIGN OFF**
  No task is "completed" until the Product Owner signs off, they are the only one who can verify when a project or task is truly complete and fully tested.

- **EXTREME OWNERSHIP**
  Most project failures are not one person's fault, but your Product Owner needs to be ready to accept total responsibility for the result of a project.

Stakeholder Role

- **WHO CAN INFLUENCE THE PRODUCT?**

  Senior Pastors

  Ministry Directors

  Staff Members

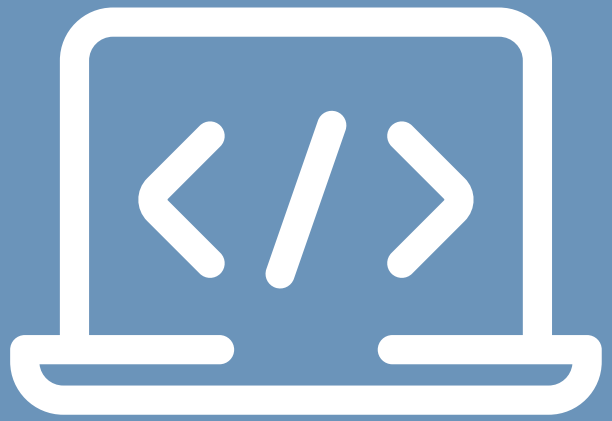
  Church Members

Stakeholder Responsibilities

- **COLLABORATION WITH PRODUCT OWNER**
  Stakeholders should not make demands of your development team.

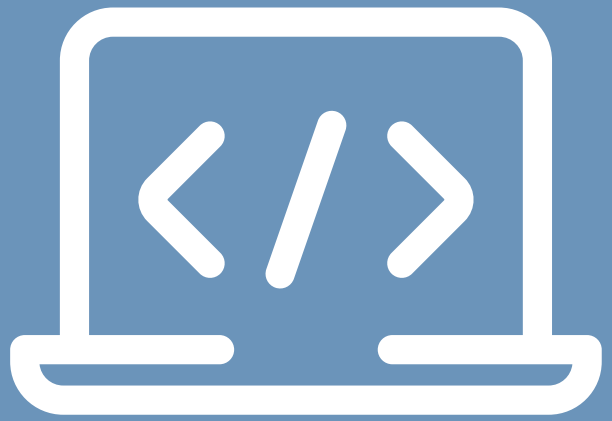
  They can advise and work with the Product Owner but all decisions about a project belong to the Product Owner.

- **TESTING**
  If you are going to let someone have influence on a product's design they should be a part of making sure it works.



Developer Role

- **WHO IS BUILDING THE PRODUCT?**
  This isn't necessarily a software developer. Depending on the scope and complexity it could be someone with a ton of Rock expertise.



Developer Responsibilities

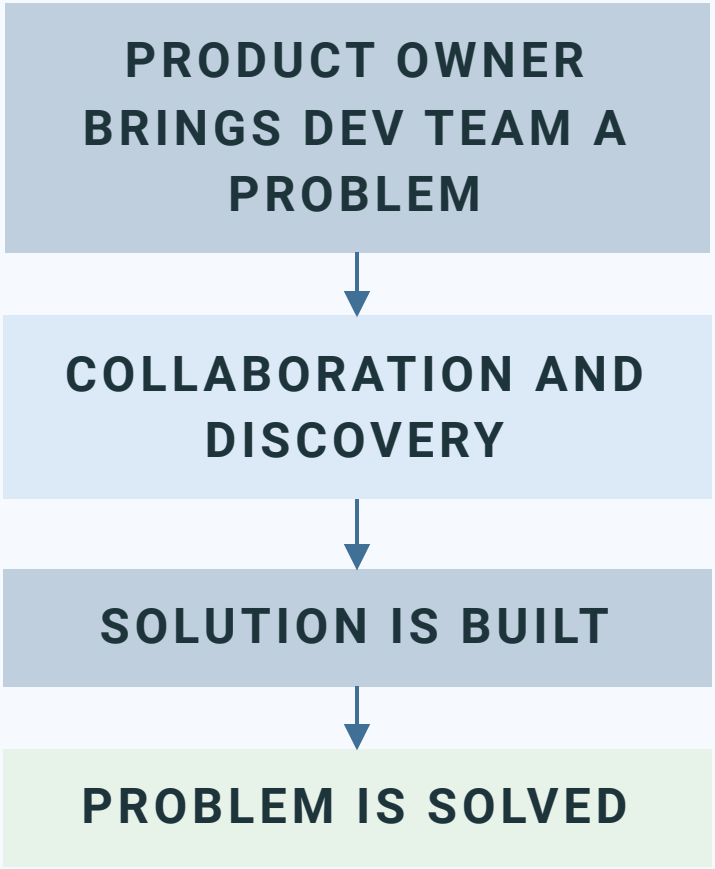
- **REQUIREMENTS GATHERING**
  The developer needs to fully comprehend what issue the ministry is facing.
- **RESEARCH**
  The developer needs to spend some time analyzing and evaluating possible solutions.
- **CREATING A PRODUCT**
  They need to build a solution for the problem.

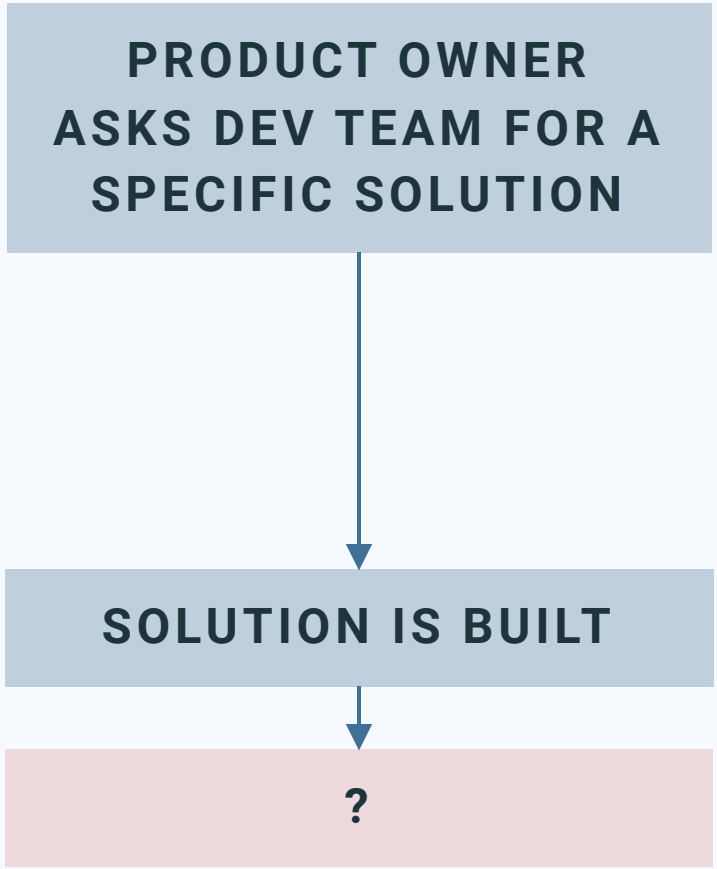
How to Have a Great Partnership

You will always have a client/contractor relationship with your development team. In that relationship your dev team can play two roles.

- **CONSULTANT**
  A partner that can collaborate with you, understand your ministry needs, and find the right solution for your church.
- **EXECUTOR**
  A partner that will just do exactly what you ask, no matter the consequences.

The Difference Between Consulting and Executing





Choosing a Development Team: Budget

- **DEVELOPER ON STAFF**
  Can you afford a senior developer? A Team?

- **RETAINER WITH A PARTNER**
  You may not have enough money for a new hire, but having a regular number of hours with a contractor is also sustainable for custom development.

- **SINGLE PROJECT WITH A PARTNER**
  Who will maintain the product? What will you do when you upgrade?

Choosing a Development Team: Rock Expertise

- **YOUR LEVEL OF ROCK KNOWLEDGE**
  How well does your Rock Product Owner know the ins and out of what is available?
  The rest of your staff?

- **PARTNER'S LEVEL OF ROCK KNOWLEDGE**
  Does this contractor regularly work with Rock?
  Do they only work with certain parts of Rock? (i.e. payments)

Choosing a Development Team: Provided Services

- **CUSTOM SOFTWARE**
  Does the contractor actually offer custom software?

- **MAINTENANCE**
  Does the contract state the dev team's responsibility to you ends after initial development and release?

  Will they fix bugs found after release?

  Will they maintain the software through Rock upgrades?

Clarifying Common Misconceptions

YOU SHOULD ASK THE DEV TEAM FOR A SPECIFIC SOLUTION

THE DEVELOPMENT TEAM IS IN CHARGE OF PROVIDING A SOLUTION

The dev team are your Rock experts. Don't limit them by giving them the solution you think is right.

THE DEVELOPMENT TEAM ALREADY TESTED EVERYTHING

THE DEVELOPMENT TEAM IS NOT RESPONSIBLE FOR TESTING

Your dev team is going to test, but they are not the end users.

YOU SHOULDN'T EXPOSE MINISTRIES TO ALPHA VERSIONS

BRING IN A DIVERSE GROUP OF TESTERS

You've got that one person on staff that always seems to do something unexpected that breaks your perfectly functioning product at release. Bring them in early.

Migration Checklist (Upgrading Rock)

- [ ] Read the Release Notes for the new version you are upgrading to
- [ ] Make a note of any major breaking changes or deprecated features you need to test or make changes to your environment for (i.e. Lava Engine Changes)
- [ ] Export Copy of Production Database
- [ ] Restore Copy of Production Data on Dev Database Server
- [ ] Sanitize Database (Very important, don't want to send duplicate emails out, or let people make contributions)
- [ ] Connect Dev Environment to the Sanitized Database
- [ ] Upgrade Dev Environment to Desired Version
- [ ] Upgrade Plugins
- [ ] Testing Round One (Rock Team, Advanced Users)
    - [ ] Critical functions (Whatever they are for you, make sure the way you most often use them is working, i.e. Send an Email, Make a Donation, Watch the Livestream, etc.)
    - [ ] Custom Blocks
    - [ ] Modified Core Blocks (depending on how you upgrade you might need to reapply changes)
    - [ ] Find things that aren't custom code but are custom to your instance and make sure they don't need additional configuration after the migration (e.g. custom sub pages in person profile may need to be created again and updated after moving to v15+)
    - [ ] Create a list of all these things you needed to test (Page, Block, Description of Test, Notes)
- [ ] Bug Fixes and Upgrade Development
- [ ] Testing Round Two (Invite more or all of your staff)
    - [ ] Critical functions (What does each person need to do their jobs? This could be complex workflows you've built for them, or custom dashboards. Make them verify everything works the way they expect it to!)
- [ ] Create a list of post migration configuration you will need to do (Page, Block, Description)
- [ ] Create a list of post migration custom plugin updates you will need to do
- [ ] After verifying everything in the Dev Environment works as expected select the date you will upgrade production
- [ ] Upgrade Production Environment
- [ ] Run through post-migration check lists of additional configuration and custom plugin upgrades or core block modifications you need to apply
- [ ] Keep the lists you made this migration and add to them as new projects come up, now you have a comprehensive list of custom things to test for your next m

Questions to Ask Before You Customize

Is Our Church Ready for Customization?

- DEVOTE TIME AND EXPERTISE TO DISCOVERY
- CONSISTENT DEVELOPER ACCESS
- AMPLE TIME
- EXPERTISE IN ROCK
- DEDICATED PRODUCT OWNER
- COMMITMENT TO TESTING

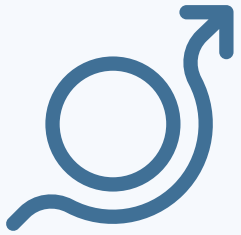
- INSUFFICENT DISCOVERY, NO CLEARLY DEFINED GOAL
- LIMITED DEVELOPER ACCESS
- LIMITED TIME
- LIMITED ROCK KNOWLEDGE
- NO ONE TO OWN PRODUCT
- LIMITED TO NO TESTING

Can people do their jobs?

IS THE PROBLEM YOU ARE TRYING TO SOLVE STOPPING SOMEONE FROM DOING A PART OF THEIR JOB?

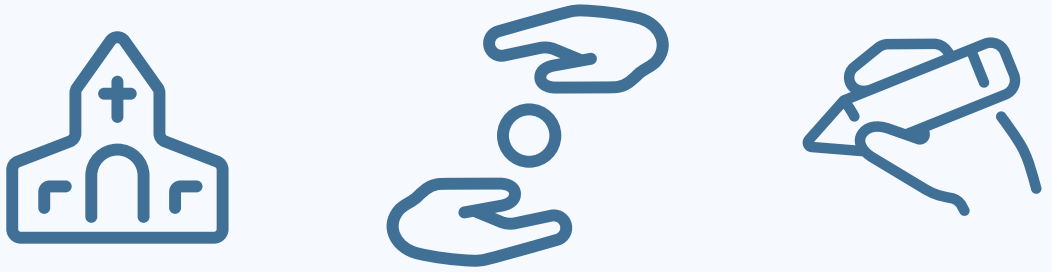
Is there a workaround?

IT MIGHT TAKE MORE TIME OR NOT BE AS EASY TO USE BUT IS THERE ALREADY SOME KIND OF SOLUTION IN PLACE?



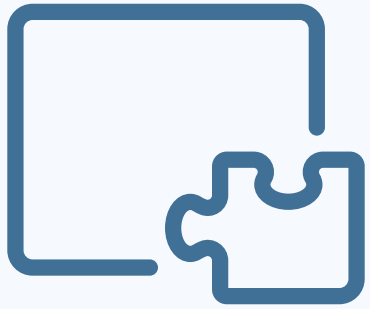
Does the Issue Affect Primary Operations?

SUNDAY MORNING SERMONS, GIVING TO THE CHURCH, SIGNING UP FOR EVENTS



Is there a solution in Core?

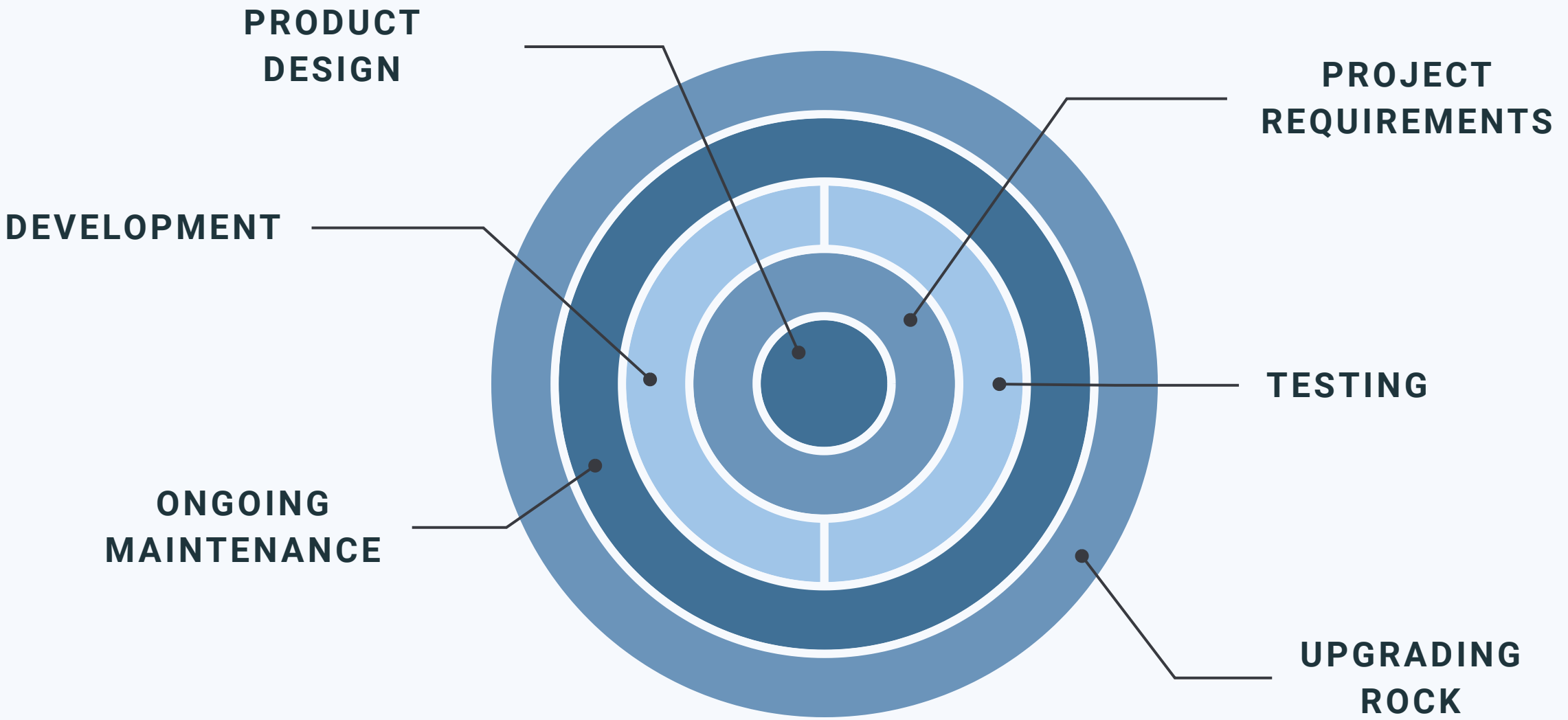
LOOK AT NEWER VERSIONS OF ROCK, EXPLORE OPTIONS IF YOU USE CORE FEATURES LIKE WORKFLOWS AND CONTENT CHANNELS



How do you upgrade Rock?

IF YOU DON'T TEST, YOU PROBABLY SHOULDN'T ADD CUSTOM CODE

Your Initial Idea Quickly Multiplies



In Summary

OWN YOUR ROCK INSTANCE

TEST

BE HONEST IN YOUR SELF EVALUATION BEFORE YOU CUSTOMIZE