LAVA 101

MADE WITH
beautiful.ai

Who Am I?

Software Engineer
Little Rock, AR
The Crossing, EPC

What is this about?

What is Lava?
How do you write Lava?
Where can you use Lava?

MADE WITH
beautiful.ai

AGENDA

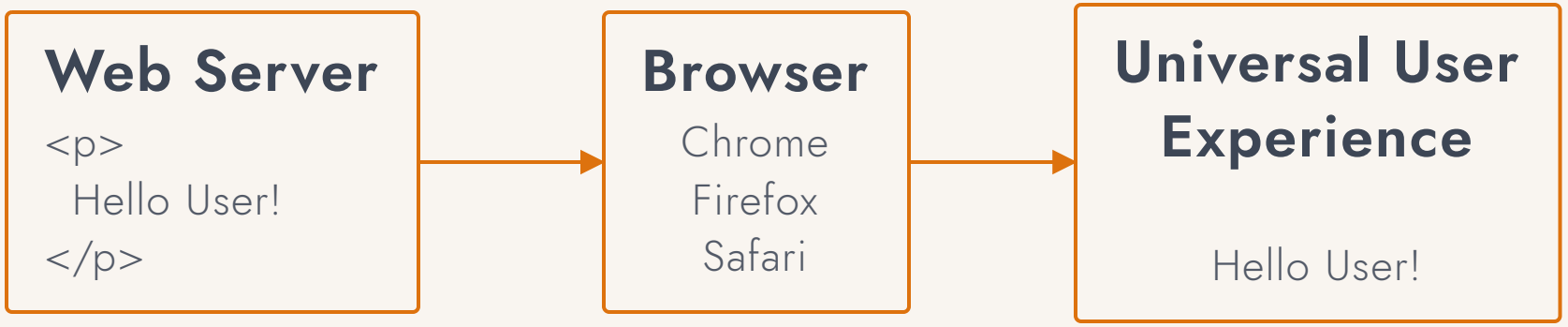
1. What is Lava?
2. Lava Tags
3. Lava Filters
4. Lava Commands
5. Resources

MADE WITH beautiful.ai

WHAT IS LAVA?

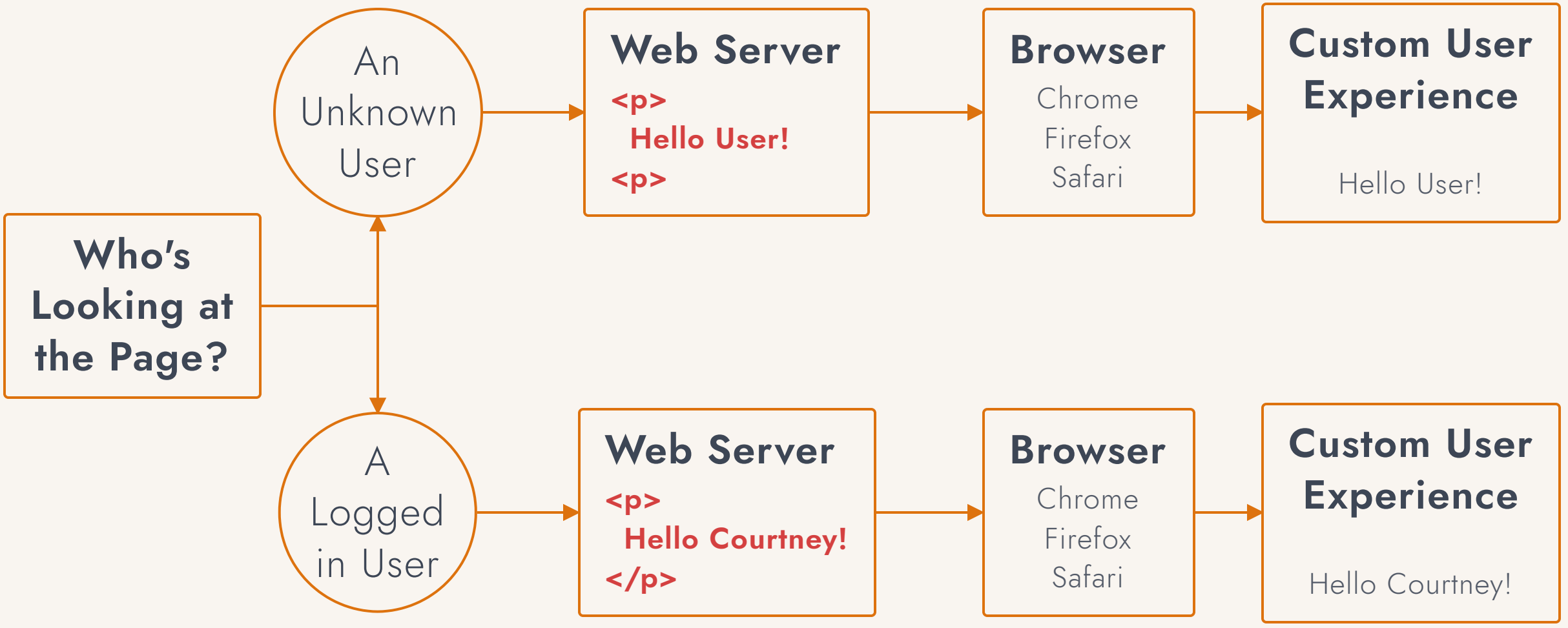
MADE WITH
beautiful.ai

STATIC WEBSITES



MADE WITH
beautiful.ai

DYNAMIC WEBSITES



MADE WITH beautiful.ai

TYPES OF WEB LANGUAGES

Markup Languages

HTML

Define the layout of elements on a page.

Style Sheet Languages

CSS

Define how elements on the page should be displayed.

Templating Languages

LAVA

Allow you to define placeholders in HTML that your server injects into the code to customize the output.

MADE WITH
beautiful.ai

Edit HTML

Display from

to

Segoe UI 16

```
<p>
    Maecenas euismod lectus est, sed iaculis velit ornare sed. Fusce vel purus facilisis leo ultrices dignissim varius congue nisl. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Mauris in tempor sapien. Nullam volutpat dui nec rutrum accumsan. Donec consequat accumsan eleifend. Aenean molestie, nulla nec sodales malesuada, nulla nunc elementum dolor, ultrices elementum justo lacus eu metus. Duis eget libero sapien.</p>
<p>
    Nunc laoreet nisi ut mauris dictum vulputate. Donec suscipit congue rhoncus. Aenean cursus nec nibh a gravida. Etiam porta rhoncus dictum. Praesent cursus, ante in feugiat accumsan, justo purus cursus arcu, nec faucibus eros neque quis dolor. Nullam enim mi, fringilla non enim vel, lacinia viverra turpis. Donec nisi velit, sagittis at orci euismod, pretium mollis nibh. Proin ut scelerisque neque, sed tristique arcu. Nulla ullamcorper mauris dui, sed cursus ipsum pulvinar non. Suspendisse fermentum quis nulla rhoncus bibendum. Nam ipsum dolor, malesuada sit amet ornare congue, elementum quis dolor. Interdum et malesuada fames ac ante ipsum primis in faucibus. Integer non felis et mi venenatis accumsan vitae in orci. Etiam in nibh sed lorem aliquet pharetra eget ut dolor. Nunc aliquet dolor dapibus ante porttitor tempor.</p>
<p>
    Proin molestie ante nisi, sit amet pharetra turpis porttitor at. Aliquam semper tempus posuere. Vestibulum sem ipsum, suscipit vitae sodales at, sagittis eget leo. Cras aliquet mollis mi eget mollis. Donec sapien erat, molestie a varius a, laoreet a est. Mauris sagittis malesuada pharetra. Vivamus id diam eget sem consequat eleifend a vitae justo. Maecenas porta convallis arcu ac hendrerit. Suspendisse sem orci, adipiscing nec dignissim in, gravida sit amet nisl . Mauris laoreet pulvinar enim, quis fringilla dui feugiat ut. Nulla facilisi.</p>
```

Save

MADE WITH beautiful.ai

LAVA SYNTAX

Use double curly braces to define a placeholder in your HTML.

```
<p>
    Hello User!
</p>
```

```
<p>
    Hello {{ User }}!
</p>
```

MADE WITH
beautiful.ai

LAVA SYNTAX

Cool...but that didn't work.

MADE WITH
beautiful.ai

LAVA SYNTAX

- **Variable**

  An object that lives on the server. The contents of this object can change based on work the server does or on the context the server is provided.

- **Entity**

  An object with properties and attributes.

  A Person is an entity in Rock.

  First Name is a *property* of a Person entity.

  Allergy is an *attribute* of a Person entity.

MADE WITH
beautiful.ai

DATABASE AN AN EXCEL WORKBOOK

The Excel Workbook is our Database.
Each Excel Sheet is a Table in the Database.
Each Row in a Sheet is a Record/Entity we can retrieve from the Table in the Database.
Each Column is a Property of the Entity.

Person Table

ID	FirstName	LastName	Email	BirthDate
1	Daphne	Blake	dangerprone@scooby.doo	8/6/1951
2	Elizabeth	Bennet	lizzie@darcy.com	12/16/1774
3	Luke	Skywalker	luke@jedisrule.org	19 BBY

MADE WITH beautiful.ai

LAVA SYNTAX

The variable that stores the current user on the server is called **CurrentPerson**.

```
<p>
    Hello {{ CurrentPerson.NickName }}!
</p>
```

Use a **.** to access a property of your variable.

MADE WITH beautiful.ai

ALL LAVA IS DEFINED BY A SET OF MATCHING SYMBOLS

When we want to print something to the page, we use double curly braces.

```
<p>
    Hello {{ CurrentPerson.NickName }}!
</p>
```

When we want to do work in Lava, we use a curly brace and percent sign.

```
{% assign x = 7 %}
<p>
    X is {{ x }}.
</p>
```

MADE WITH beautiful.ai

LAVA TAGS

MADE WITH
beautiful.ai

A LAVA TAG IS A KEYWORD THAT ALLOWS US TO ADD ADDITIONAL LOGIC

Tags allow us to do things like:

- Create custom variables and set their value
- Handle scenarios with conditional logic (if a then b else c)
- Repeat a section of logic without writing the same code over and over
- Define a section of code that should not process the Lava within it

MADE WITH beautiful.ai

TAGS CAN BE INLINE OR BLOCK

Sometimes work we do in Lava is simple, we can write what needs to be done in one line. This can be done with inline tags.

```
{% assign myVariable = 'This is the value of my variable.'
%}
```

MADE WITH
beautiful.ai

TAGS CAN BE INLINE OR BLOCK

Other times it won't be so simple. We may need to process other Lava code inside of the tag we are using. In this case we need to use a block tag.

```
{% capture myVariable %}
    ...Some code...
{% endcapture %}
```

Here, **capture** is the block tag we are using. To define the end of a block tag, you add another line with the tag name preceded by "**end**" wrapped in the curly brace/percent symbols.

MADE WITH
beautiful.ai

THE ASSIGN TAG IS USED TO CREATE AND SET THE VALUE OF CUSTOM VARIABLES

Assign creates the variable the first time it is used and the second time it just updates the value of the existing variable.

```
{% assign myVariable = 'The first value' %}
myVariable is {{ myVariable }}
<br/>
{% assign myVariable = 'The second value' %}
myVariable has been changed to {{ myVariable }}
```

MADE WITH beautiful.ai

TYPES OF VALUES YOU CAN ASSIGN TO YOUR CUSTOM VARIABLES

- **String**
  Typically letters, words, or phrases. Strings are text values.

  {% **assign** myString = 'This is a string.' %}

- **Numeric**
  Numeric values like integers or decimals.

  {% **assign** myNumber = 7 %}

- **Boolean**
  True or False. There are keywords for **true** and **false**

  {% **assign** myBool = **true** %}

- **Date**
  Dates are allowed in many formats, typically Rock stores date values as yyyy-MM-ddTHH:mm:ss

  {% **assign** myDate = '2024-01-01 06:00:00' | Date:'yyyy-MM-dd HH:mm:ss' %}

MADE WITH beautiful.ai

THE CAPTURE TAG IS USED TO CREATE AND SET THE VALUE OF CUSTOM VARIABLES

Capture is very similar to **assign**. We use **capture** when the value we want to save to the variable is complex or involves conditional logic.

```
{% capture greeting %}Hello, {{CurrentPerson.NickName}}{% endcapture %}
{{ greeting }}
```

Everything between the capture tags will be stored in the variable.

Unlike **assign** which is an *inline* tag, **capture** is a *block* tag.

MADE WITH beautiful.ai

IF IS A TAG THAT ALLOWS FOR CONDITIONAL LOGIC

If is a block tag. The code between the start and end tags will only run *if* the condition in the opening tag is met.

```
{% if CurrentPerson.LastName == 'Decker' %}
    ...This is only run when last name is Decker...
{% endif %}
```

MADE WITH beautiful.ai

THE COMPARISON OPERATIONS FOR IF STATEMENTS

- **Equality ==**

  Is the first term equal to the second term?

  In programming a single equal sign **=** is used for assigning the value of a variable so to check equality we use a double equal sign **==**

- **Mathematical Comparisons**

  <, <=, >, >=

  Is the first term less than **<**, less than or equal to **<=**, greater than **>**, or greater than or equal to **>=** the second term?

  You can use these comparisons on numbers, dates, and even text

- **Inequality !=**

  Is the first term not equal to the second term?

  In programming an exclamation point **!** means **not**

- **Contains contains**

  Does the first term have a part of it equal to the second term?

  e.g. we may want to find all people who have the word 'Peanut' in their list of allergies

MADE WITH beautiful.ai

USING COMPARISON OPERATORS

```
{% if CurrentPerson.LastName == 'Decker' %}
    ...Prints when the logged in user's last name is Decker...
{% endif %}
{% if CurrentPerson.LastName != 'Decker' %}
    ...Prints when the logged in user's last name is anything except Decker...
{% endif %}

{% if CurrentPerson.LastName contains 'D' %}
    ...Prints when the logged in user's last name has a "D"...
{% endif %}

{% if CurrentPerson.Age >= 50 %}
    ...Prints when the logged in user's age is 50 or higher...
{% endif %}
```

MADE WITH beautiful.ai

YOU CAN HAVE MULTIPLE CONDITIONS

- **All conditions must be met and**

  Used when you have multiple criteria that must be met in order for the section of code to run.

  {% **if** Person.LastName == 'Decker' **and** Person.FirstName == 'Ted' %}

  Will only run for someone named Ted Decker

- **Any condition must be met or**

  Used when any of the conditions should trigger the section of code to run.

  {% **if** Person.LastName == 'Decker' **or** Person.LastName == 'Marbles' %}

  Will run for anyone who's last name is Decker or Marbles

MADE WITH beautiful.ai

STACKING CONDITIONS

```
{% if CurrentPerson.LastName == 'Decker' and CurrentPerson.FirstName == 'Ted' %}
    ...Prints when the logged in user's name is Ted Decker...
{% endif %}

{% if CurrentPerson.LastName == 'Decker' or CurrentPerson.LastName == 'Marbles' %}
    ...Prints when the logged in user's last name is Decker OR Marbles...
{% endif %}
```

MADE WITH
beautiful.ai

THE ELSE TAG CAN BE USED TO EXTEND THE IF TAG

The **else** tag doesn't need a closing tag. It is a companion of the **if** tag and can only be used inside of an **if** tag.

```
{% if CurrentPerson.LastName == 'Decker' %}
    Hello Decker Family!
{% else %}
    Hello!
{% endif %}
```

MADE WITH beautiful.ai

THE ELSEIF TAG CAN BE USED WHEN YOU HAVE MORE THAN ONE CONDITION THAT NEEDS CUSTOM LOGIC

```
{% if CurrentPerson.LastName == 'Decker' %}
    Hi Decker Family!
{% elseif CurrentPerson.LastName == 'Marbles' %}
    Goodday Marbles Family!
{% else %}
    Hello!
{% endif %}
```

MADE WITH beautiful.ai

IF BLOCKS WILL STOP PROCESSING AS SOON AS A CONDITION IS MET

```
{% if CurrentPerson.LastName == 'Decker' %}
    Hi Decker Family!
{% elseif CurrentPerson.NickName == 'Ted' %}
    How are you, Ted?
{% else %}
    Hello!
{% endif %}
```

MADE WITH beautiful.ai

A FOR LOOP IS USED TO REPEAT A SECTION OF LOGIC FOR MANY ITEMS

A for loop is going to create a new variable for us to use inside the for loop code. It will contain the current item from the list we provide.

```
{% for currentGroupMembership in CurrentPerson.Members %}
    {{ currentGroupMembership.Group.Name }} <br/>
{% endfor %}
```

This loop will print the name of every group that the current person is a member of.

currentGroupMembership will only exist inside of the for tag.

MADE WITH beautiful.ai

THE FOR TAG CAN ACCEPT ADDITIONAL PARAMETERS TO EFFECT HOW IT RUNS

- **limit**
  Used to limit the total number of times a loop can run
- **offset**
  Used to skip over a number of items in the list before starting the loop
- **reversed**
  Used to process the list of items in the reverse order they appear in the list

MADE WITH beautiful.ai

ADDITIONAL PARAMETERS WILL GO IN THE OPENING TAG OF THE FOR LOOP

```
{% for currentGroupMembership in CurrentPerson.Members
limit:3 %}
    {{ currentGroupMembership.Group.Name }} <br/>
{% endfor %}
```

This loop will print the name of the first three groups that the current person is a member of.

MADE WITH beautiful.ai

AN EXAMPLE

Variables For Loop

Properties If Statement

```
{% assign onServingTeam = false %}
{% for membership in CurrentPerson.Members %}
    {% if membership.IsActive == true and membership.Group.IsActive == true %}
        {% if membership.Group.GroupType.GroupTypePurposeValue != null %}
            {% if membership.Group.GroupType.GroupTypePurposeValue.Value == 'Serving Area' %}
                {% assign onServingTeam = true %}
            {% endif %}
        {% endif %}
    {% endif %}
{% endfor %}
{% if onServingTeam == true %}
    Thank you for serving in our church!
{% else %}
    Looking for a place to volunteer?
{% endif %}
```

Initialize our check variable to false.

Loop through each group membership for the current person.

Check if the group and membership are Active.

Verify the group's group type has a purpose.

And that purpose is Serving Area.

Set our check variable to true because they are serving.

Display thank you message if they are serving.

Call to serve if they are not.

MADE WITH beautiful.ai

LAVA FILTERS

MADE WITH
beautiful.ai

A LAVA FILTER IS A KEYWORD THAT ALLOWS US TO MUTATE A VARIABLE'S VALUE

Filters allow us to do things like:

- Numeric operations (addition, subtraction, multiplication, division, etc)
- Format date values
- Modify date values (add or subtract a length of time, get an upcoming Sunday)
- Change the case of some text

MADE WITH beautiful.ai

LAVA FILTERS ALSO ALLOW US TO PERFORM SIMPLE OPERATIONS

Filters allow us to do things like:

- Get the first or last item from a list
- Sort a list of items or group a list of items by a common property
- Get information about a person like group membership
- Get the value of an attribute for an entity
- Get the JSON value of a variable

MADE WITH beautiful.ai

FILTER SYNTAX

Filters will always be written in the pattern

Input | FilterName: 'Parameter','SecondParameter'

```
{{ CurrentPerson | Attribute:'Allergy' }}
```

MADE WITH
beautiful.ai

GET A SUBSET OF A LIST WITH THE WHERE FILTER

Instead of checking each group membership for validity, we can filter out the memberships that don't meet our criteria with the filters.

```
{% assign activeMemberships = CurrentPerson.Members | Where:'GroupMemberStatus','Active'
%}
{{ activeMemberships | Size }}
```

We'll start by filtering to active group memberships with **Where**.

MADE WITH beautiful.ai

USE THE SELECT FILTER TO TRANSFORM THE ENTITY TYPE

Filter the list of memberships to ones in active groups by transforming the list of group memberships to a list of groups.

```
{% assign activeMemberships = CurrentPerson.Members | Where:'GroupMemberStatus','Active'
%}
{% assign activeGroups = activeMemberships | Select:'Group' | Where:'IsActive',true %}
{{ activeGroups | Size }}
```

Then filter the array of groups to the ones that are active.

MADE WITH beautiful.ai

TRANSFORM THE LIST OF GROUPS TO THEIR GROUP TYPES AND THEN PURPOSES

We will filter down to the items where the purpose of the group type is "Serving Area".

```
{% assign activeMemberships = CurrentPerson.Members | Where:'GroupMemberStatus','Active'
%}
{% assign activeGroups = activeMemberships | Select:'Group' | Where:'IsActive',true %}
{% assign groupPurposes = activeGroups | Select:'GroupType' |
Select:'GroupTypePurposeValue' %}
{% assign servingTeams = groupPurposes | Where:'Value','Serving Area' %}
{{ servingTeams | Size }}
```

MADE WITH beautiful.ai

FILTERS CAN BE CHAINED TOGETHER

We will include all of our transformations and filtering in one line and save the number of group memberships a person belongs to meeting all the requirements.

```
{% assign numberOfServingTeams = CurrentPerson.Members |
Where:'GroupMemberStatus','Active' | Select:'Group' | Where:'IsActive', true |
Select:'GroupType' | Select:'GroupTypePurposeValue' | Where:'Value','Serving Area' | Size
%}
{% if numberOfServingTeams > 0 %}
    Thank you for serving in our church!
{% else %}
    Looking for a place to volunteer?
{% endif %}
```

MADE WITH beautiful.ai

THE TOJSON FILTER TURNS ENTITIES INTO A READABLE FORMAT OF KEY VALUE PAIRS

Not sure what properties are available on the entity you have?

```
{{ CurrentPerson.Members | First | ToJSON }}
```


```
{ "ArchivedByPersonAlias": null, "GroupMemberAssignments": [], "GroupMemberRequirements":
[], "GroupRole": { "IsSystem": true, "GroupTypeId": 1, "Name": "Member", "Description":
"Member of a group", "Order": 0, "MaxCount": null, "MinCount": null, "IsLeader": false...
```


MADE WITH beautiful.ai

USE A JSON FORMATTER TO DISPLAY THE DATA

Don't put personal or sensitive information into an online service


```
{
  "Id": 12345,
  "GroupId": 3,
  "PersonId": 456,
  "GroupRoleId": 1,
  "GroupRole": {
    "IsSystem": true,
    "GroupTypeId": 1,
    "Name": "Member",
    "IsLeader": false,
    "CreatedDateTime": null,
    "ModifiedDateTime": "2019-10-22T09:47:18.62",
    "CreatedByPersonAliasId": null,
    "ModifiedByPersonAliasId": 37830,
    "Id": 1
  }
}
```


MADE WITH beautiful.ai

READING JSON

- **Objects { }**
  Objects in JSON will start and end with a single curly brace

- **Properties "Key": Value**
  Properties on an object will be written with their Key in quotes, a colon, and then their value.

- **Value "Key": Value**
  Values can be text, numeric, boolean, objects, or arrays.

- **Array [ ]**
  Arrays are a collection of values similar to a list. Arrays will start and end with a single square bracket.

  Arrays do not have properties. The objects in them do.

MADE WITH
beautiful.ai

JSON FOR A PERSON RECORD

```
{
    "Id": 12345,
    "FirstName": "Ted",
    "LastName": "Decker",
    "ConnectionStatusValue": {
        "DefinedTypeId": 4,
        "Value": "Attendee"
    },
    "PhoneNumbers": [
        {
            "Id": 212,
            "Number": 0123456789,
            "NumberTypeValue": {
                "Value": "Mobile"
            },
        },
        { ...Work Phone... }
    ]
}
```

CONNECTION STATUS VALUE IS NOT A COLUMN IN THE PERSON TABLE

Database tables do not store entire objects or arrays (lists) as columns values.
ConnectionStatusValueId is a Property of the Person entity.
Rock understands this connection and adds the object for the Connection Status Defined Value as a property on the record you have retrieved.

Defined Value Table

Id	Value	DefinedTypeId
65	Member	4
66	Visitor	4
67	Prospect	4

Person Table

Id	FirstName	LastName	ConnectionStatusValueId
123	Fred	Jones	65
124	Bugs	Bunny	66
125	Yogi	Bear	65

MADE WITH beautiful.ai

ACCESS TO THE OBJECT AS A PROPERTY ALLOWS US TO EASILY PRINT THE VALUE

```
{
    "Id": 1234,
    "ConnectionStatusValueId": 64,
    "ConnectionStatusValue": {
        "DefinedTypeId": 4,
        "Value": "Attendee"
    }
}
```

```
{{ CurrentPerson.ConnectionStatusValue.Value }}
```

MADE WITH beautiful.ai

ATTRIBUTE VALUES ARE NOT STORED IN THE PERSON TABLE

Similar to Defined Values, Attribute Values are stored in their own table. They link to the Attribute and Entity they apply to.

Attribute Values Table

ID	AttributeId	EntityId	Value
4	Allergy	Lizzie Bennet	Tree Nuts
5	School	Lizzie Bennet	Ridell High
6	Allergy	Luke Skywalker	Strawberries

MADE WITH beautiful.ai

THE ATTRIBUTE VALUES ARRAY IS TERRIBLE TO WORK WITH

```
{
    "Id": 1234,
    "AttributeValues": [
        {
            "AttributeId": 740,
            "EntityId": 30820,
            "Value": "The Crossing"
        },
        { ...Another Attribute Value... }
    ]
}
```

MADE WITH beautiful.ai

ENTITY ATTRIBUTES CAN BE ACCESSED WITH THE ATTRIBUTE FILTER

Instead of wading through that mess...

```
{{ Person.AttributeValues | Where:'AttributeId',740 | First | Property:'Value' }}
```

Attribute Values can be accessed by the Attribute **Key** with the Attribute filter.

```
{{ CurrentPerson | Attribute:'Allergy' }}
```

MADE WITH beautiful.ai

ected to the Rock Test database

Edit Employer Id: 740

Name •
Employer

Abbreviated Name
Employer

Description
The company that person is employed at

Categories
Employment

Key •
Employer

Required
Require a value

Show on Bulk
Yes

Active

Public

Field Type
Text

Password Field
Yes

Max Characters

Show Character Limit Countdown
Yes

MADE WITH beautiful.ai

SOME OTHER FILTERS

Add a timespan to a date (default will be days) and format the result.

```
{{ 'Now' | Date | DateAdd:7 | Date:'MM/dd/yy' }}
```

Check if a person belongs to a group (possibly a security role).

```
{{ CurrentPerson | Group:'2','Active' }}
```

List the nearest campus to a person.

```
{{ CurrentPerson | NearestCampus | Property:'Name' }}
```

MADE WITH beautiful.ai

LAVA COMMANDS

MADE WITH
beautiful.ai

WHAT IS A LAVA COMMAND?

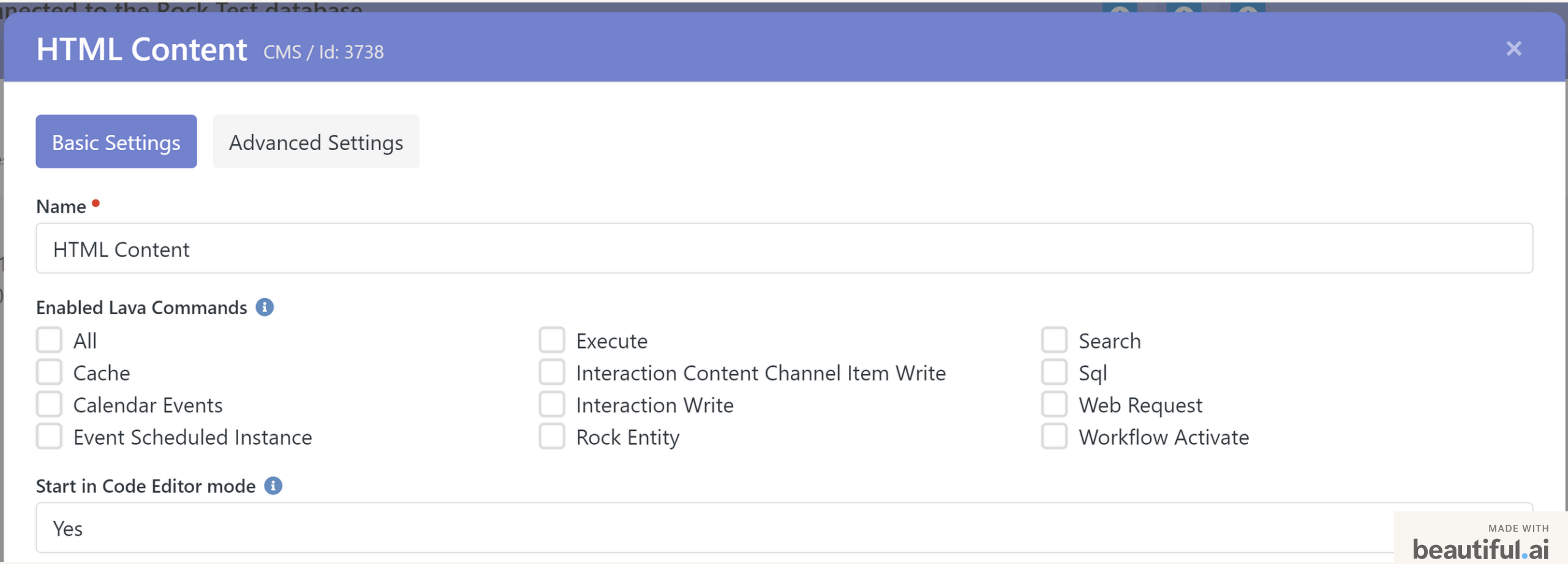
A Lava Command is a keyword in Lava that the server knows to process in a certain way. Lava Commands allow you to:

- Get data like Group Memberships, Financial Transactions, or Registrations
- Run custom SQL queries
- Create custom Interactions
- Launch Workflows
- And more!

MADE WITH beautiful.ai

ENABLING LAVA COMMANDS

Before you can use a Lava Command you must enable it for the context of your current block.



MADE WITH beautiful.ai

RETRIEVE DATA WITH THE ENTITY COMMAND

To get information about a specific type of data we will use the name of that data type (lowercase and no spaces) to let the server know we want to retrieve data.

```
{% groupmember %}
```

When we are done with the data we will tell the server we no longer need it by adding "end" to the name of the data type.

```
{% endgroupmember %}
```

MADE WITH
beautiful.ai

PROVIDE INFORMATION ABOUT THE DESIRED DATA WITH PARAMETERS

To get data using this entity command, we need to provide more information to the server about the specific data we are trying to access.

```
{% groupmember id:'1' %}
{% endgroupmember %}

{% groupmember ids:'1,2,3' %}
{% endgroupmember %}

{% groupmember where:'GroupId == 4' %}
{% endgroupmember %}
```

MADE WITH beautiful.ai

USING THE ENTITY DATA

When accessing entity data by id, the variable available to you will be the same name as the entity type.

```
{% groupmember id:'1' %}
    {{ groupmember.Person.FirstName }}
{% endgroupmember %}
```

MADE WITH
beautiful.ai

USING THE ENTITY DATA

When accessing entity data by ids or a where clause, the variable available to you will be the same name as the entity type + "**Items**".

```
{% groupmember ids:'1,2,3' %}
    {{ groupmemberItems[0].Person.FirstName }}
{% endgroupmember %}
{% groupmember where:'GroupId == 4' %}
    {% assign firstMember = groupmemberItems | First %}
    {{ firstMember.Person.FirstName }}
{% endgroupmember %}
```

MADE WITH beautiful.ai

DON'T PLAY AROUND WITH THE SQL COMMAND

- There is a Lava command that let's you run SQL directly on your server.
- Running SQL can open you up to SQL Injection Attacks if you do not handle user input properly.
- Don't use the SQL Command if you are unfamiliar with SQL and the dangers involved

MADE WITH beautiful.ai

THE PERSONALIZE COMMAND ALLOWS FOR DYNAMIC CONTENT

Bring your web page to life with dynamic content based on personalization segments.

```
{% personalize segment:'SegmentKey' requestfilter:'FilterKey' matchtype:'all' %}
  Content for people in this segment!
{% otherwise %}
  Content for everyone else!
{% endpersonalize %}
```

- **segment:** a comma separated list of personalization segement keys
- **requestfilter:** a comma separated list of request filter keys
- **matchtype:** does someone need to be in any of the request filters or all?

MADE WITH beautiful.ai

OTHER NOTABLE COMMANDS

- **Calendar Events**
  Using a Calendar id and Audience ids get an array of **EventScheduledInstances** to print event details

- **Event Scheduled Instance**
  Given an Event Item id get the upcoming occurrences

- **Interaction Content Channel Item Write**
  Write custom interactions based on user activity

- **Workflow Activate**
  Launch a new workflow or create a new activity for a workflow with Lava

MADE WITH beautiful.ai

RESOURCES

MADE WITH
beautiful.ai

HTTPS://COMMUNITY.ROCKRMS.COM/LAVA
HTTPS://ROCKRMS.COM/ROCKSHOP/PLUGIN/22/LAVA-TESTER

Rock Shop Preview

This plugin is only available on the Rock Shop. To install this plugin, select **Admin Tools > Rock Shop** from your own instance of Rock.

Lava Tester by Central Christian Church (AZ)

Free

Note: plugin details are provided here only as a reference for what is available in the Rock Shop. To install this plugin, select **Admin Tools > Rock Shop** from your own instance of Rock.

Package Description

The Rock Lava Tester block will save you hours of time by letting you *quickly test* your *Lava* against your real-world entities including Person, Group, Workflow, etc. The tool will pass each of the entities you select to your Lava and then render it using the standard Rock Lava engine. It also lets you save Lava snippets in up to 25 save slots for recalling later.

This block has literally saved our team countless hours testing the Lava that we're working out in our new email templates and workflow actions.

Support Website

MADE WITH beautiful.ai