Make.com Notes

Day 0

- Common tools
  - Data Bases
  - CRM
  - Email / marketing automation
  - E-commerce
  - CMS
  - project management
  - office
  - Telephony

Day 1

- JSON Essentials
  - Array eg: ["sheet1", "words", 234]
  - Collection eg: {"number" 201, "nameOwner": "Alex", "PaymentDate": "05/01/2022"}
  - object: a collection of name/value pairs

{"key": "value"}

* formatting:
- string in double quote
- rest NOT in quote.
- backslash for escape special characters.

Day 2

- Variables
- Functions
  - * Several function.
    - ()
    - set
    - if
    - ifempty
    - switch
    - omit
    - pick.
  - * Date/Time Functions
  - * Math Functions
  - * Text Functions
  - * Functions working with Arrays

* tips: can check JSON bubble to get the target value when creating variables.

Day 3

- Connection.
- webhook ☆ one webhook per Scenario

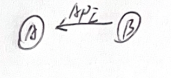
url + ? + parameters
eg: http://hook.us1.make.com/xxx?name=yyy
url → webhook → make (client)
(parameter) parameter

Webhook response
{ Content-type: text/html → HTML page
  location: https://google.com → redirection.

HTTP { Get - url? parameter.
       post - JSON

API vs. webhook.

- API.
- Webhook: (event driven)



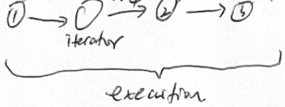
Day 3 Execution & cycles

Execution { "Run" button / scheduled trigger

iterator → cycle

* May cycles in one execution

variable lifetime. (one cycle / one execution / Never in)



x1 after iterator - one cycle.

x4 means 4 bundles, 4 cycles

Day 4 iterator and aggregator

iterator - input : Array

- output : Bundles of items in Array

or choose variable

* Create Array directly or use Map function

eg: split("test1, test2, 123;", ")

=> ["test1", "test2", 123]

Aggregation * incorrect source can lead to inaccessible variables outside aggregation.

eg: Aggregating email attachments using correct sources to ensure data accessible

* variable inside aggregation sequence are NOT available outside of it

Tips: Group array by parameters such as file name, using function like split and Get.

Aggregator: { Array Aggregator / Text Aggregator / Table Aggregator / Numeric Aggregator

* Rmb for preventing errors by stopping process if aggregation result is empty

Day 5 complex Arrays

API → get data → Arrays (names, email, phone number) → iterator (one record at a time) → Array

- Map

Array1 phone number → Map → Array2 phonenumber → New JSON merged filled

* Efficiency - Avoid multiple API calls by merging arrays in a single operation. (API limits) ↓ reduce potential errors / optimize operation cost.

Day 7 HTTP Request, HTTP Module.

⚠ Most module in Make use APIs for external communication
custom API calls can be made using HTTP modules

Basic API Request structure
- URL
- Header : for authorization / content-type ← eg: bearer token
- Body : contain data to be sent in POST request

★ URL require parameters encoded
eg: space → %20
@ → %40

Request methods:
- GET (Retrieve data)
- POST (send data / create or update resources)

Get
URL + ? + parameters
↑ Query strings.

POST
method + headers + Body.

Request Body Types
- URL encoded forms
- multi-part forms
- JSON

Always → set content-type header

★ API Body
JSON format
Make ★ structure data → Request Body → JSON
JSON → Parsing → structure data

Responses
- successful – 200
- redirection – 300 / 3XX
- client error – 400 / 4XX
- server error – 500 / 5XX

Custom Request Type:
- multi-part requests (text in request body)
- Custom text → set request body to custom; set header (seperator / put text in request content)

Authorization setup:
username + password – basic authorization (base64 - encoded)

★ website → Cookies → authorization header

Request Timeout
Default – 40s / Max – 5 mins per call

Certificate-based
Certificate → Services call pages

★ Reject connection if certificate is incorrect → Certificate validate / fail

API call → applications (Jira) spreadsheet

1. HTTP Modules
- GET Files
- Resolve Target URL

Custom API calls
specifying the type of Data (Text, JSON, Binary)