

# Using AI for Research

Robert Hayden, MD

&

Bitá Behrouzi, MD

# **Part 1: Coding Concepts & JavaScript**

**Robert Hayden, MD**

# **Part 2: Python & More Data Demos**

**Bitah Behrouzi, MD**

# Why Talk about Coding for Non-Coders?

- **AI performs tasks by writing and executing code**
- **Simple scripts can help automate tedious tasks in your scholarly work, and AI can help you write that code**
- **Initial starting point to understand how AI can help you learn to code**

**Follow along / try these demos later:**

***Coding.NephroMaine.com***

# Outline

1. **Introducing JavaScript (and HTML)**
2. **Building blocks of code**
3. ***Demo 1: a simple browser-based web app***
4. **Privacy / policy considerations**
5. **JSON and data structures**
6. ***Demo 2: big data analysis for research***
7. **Debugging, Limitations, Big Takeaways**

# Outline

1. **Introducing JavaScript (and HTML)**
2. **Building blocks of code**
3. *Demo 1: a simple browser-based web app*
4. **Privacy / policy considerations**
5. **JSON and data structures**
6. *Demo 2: big data analysis for research*
7. **Debugging, Limitations, Big Takeaways**

# Why JavaScript and HTML?

- **Runs in web browsers (and on servers)**
- **Can build simple web apps and analyze data locally, without concern for privacy / policy violations!**
- **Easy to use: limited installation needed**

# JavaScript



Logic

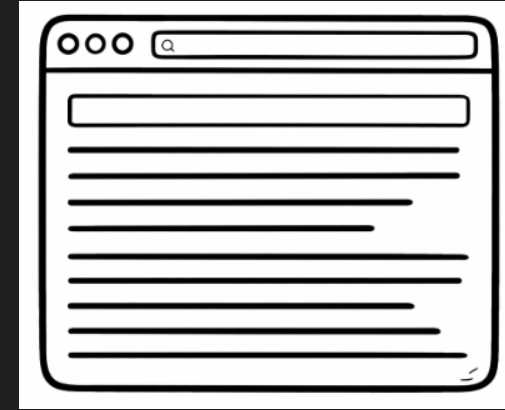
Calculations

Decision Trees

Functions

...

# HTML



Structure of the page

Inputs

Buttons

Text

...

# Outline

1. Introducing JavaScript (and HTML)
2. **Building blocks of code**
3. *Demo 1: a simple browser-based web app*
4. Privacy / policy considerations
5. JSON and data structures
6. *Demo 2: big data analysis for research*
7. Debugging, Limitations, Big Takeaways

# Building Blocks of Code

- **Variables:** store values. Could be text, numbers, lists...
- **Functions:** take inputs, transform, give outputs
- **If / Then:** branching logic
- **Loops:** for each row of data... for each patient... (scalable)

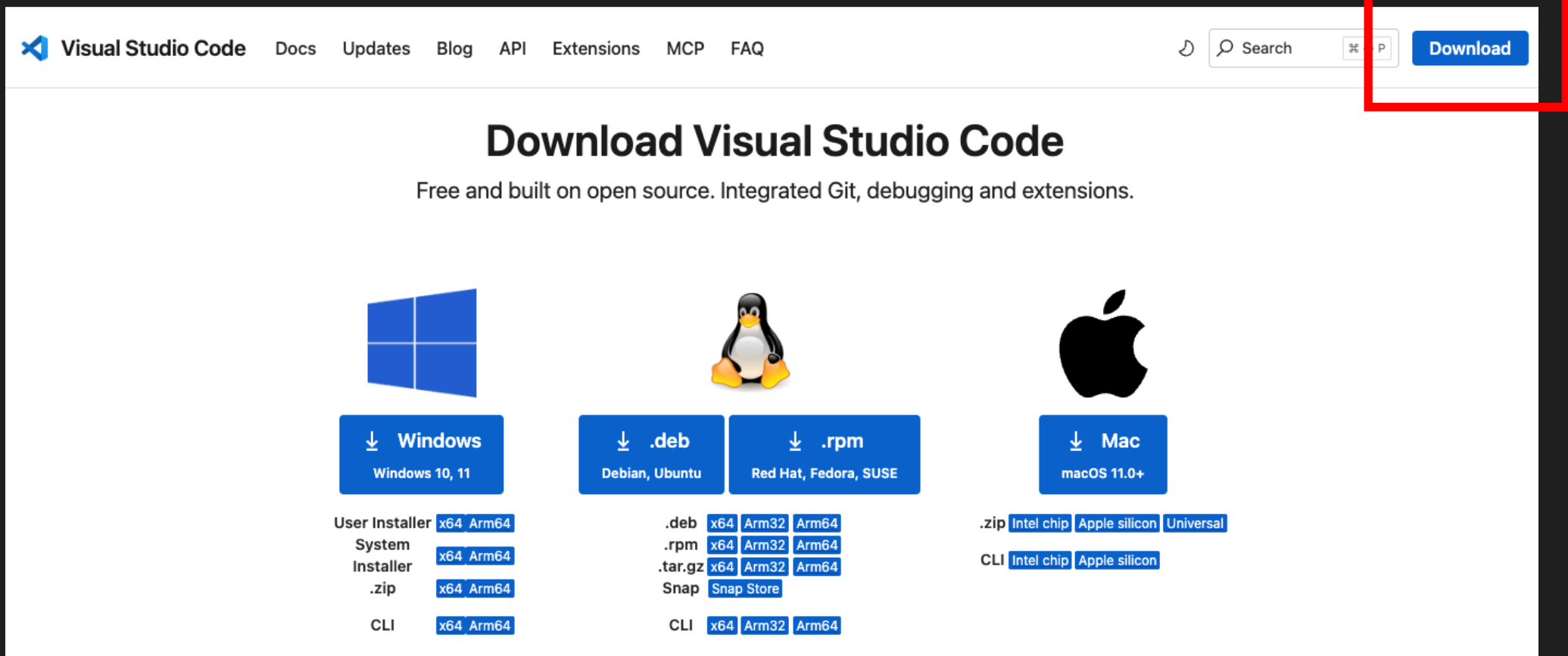
# Outline

1. Introducing JavaScript (and HTML)
2. Building blocks of code
3. **Demo 1: a simple browser-based web app**
4. Privacy / policy considerations
5. JSON and data structures
6. **Demo 2: big data analysis for research**
7. Debugging, Limitations, Big Takeaways

**DEMONSTRATION:**

**A Simple Web App**

# First: [code.visualstudio.com/Download](https://code.visualstudio.com/Download)




Visual Studio Code Docs Updates Blog API Extensions MCP FAQ

Search P **Download**


## Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



**↓ Windows**  
Windows 10, 11


User Installer	x64	Arm64
System Installer	x64	Arm64
.zip	x64	Arm64
CLI	x64	Arm64



**↓ .deb**  
Debian, Ubuntu

**↓ .rpm**  
Red Hat, Fedora, SUSE

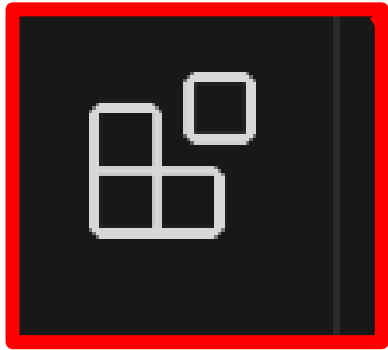
.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64	Arm32	Arm64



**↓ Mac**  
macOS 11.0+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip	Apple silicon	

# Second: extensions > install “Live Server” by Ritwick Dey



*extensions*

The screenshot shows the Visual Studio Code interface with the Extensions view open. The 'EXTENSIONS' sidebar on the left lists installed extensions, with 'Live Server' by Ritwick Dey highlighted. A red-bordered callout box provides details for the 'Live Server' extension, including its version (v5.7.9) and the 'Go Live' button. Another red-bordered callout box highlights the 'Go Live' button in the status bar at the bottom of the editor.

**EXTENSIONS**

Search Extensions in Marketpla...

**INSTALLED** 9

- Live Server** v5.7.9  
Launch a development local ...  
Ritwick Dey
- R**  
R Extension for Visual Studio...  
REditorSupport
- R Syntax**  
R Syntax Highlight for Visua...  
REditorSupport
- GitHub Copilot**  
Your AI pair programmer  
GitHub
- GitHub Copilot Chat**  
AI chat features powered by ...  
GitHub
- Pylance**  
A performant, feature-rich la...  
Microsoft
- Python**  
Python language support wit...

**RECOMMENDED** 8

**Live Server** v5.7.9  
Ritwick Dey  
Launch a development local Server with live reload feature for static & dyn...  
Disable Uninstall  
This extension is enabled globally.

**Go Live**

Ln 4, Col 1 Spaces: 4 UTF-8 LF HTML **Go Live**

# Web App Example

MD+ CALC

Search "QT interval" or "QT" or "EKG"

## Free Water Deficit in Hypernatremia

Calculates free water deficit by estimated total body water in a patient with hypernatremia or dehydration.

When to Use ▼ Pearls/Pitfalls ▼ Why Use ▼

Sex: Female Male

Age range: Child Adult Elderly

Weight: 64.4 kg

Sodium: Use only if sodium >140 155 mEq/L

Sodium desired: 140 mEq/L

**3.5 L**  
Free Water Deficit

Copy Results Next Steps


Next Steps Evidence Creator Insights

**FORMULA**

Free water deficit, L = (% total body water, fraction)\*(Weight, kg)\*[(Current sodium/Ideal sodium) - 1]

Favorite Share

**About the Creator**

 Dr. Nicolaos E. Madias  
[Are you Dr. Nicolaos E. Madias?](#)

**Also from MDCalc...**

**Related Calcs**

- [Body Fluid Balance Calculator](#)  
Calculates fluid balance from sodium concentrations which indicate net 0.9% saline, and free water losses (GI, urine, etc) and gains (IV fluids, PO, etc).
- [Sodium Correction for Hyperglycemia](#)  
Calculates the actual sodium level in patients with hyperglycemia.
- [Sodium Correction Rate in Hyponatremia/Hypernatremia](#)  
Calculates recommended IV fluid type, rate, and volume to correct hyponatremia or hypernatremia at desired rate depending on clinical scenario.

**Content Contributors**

- Lynda Szczech, MD, MSCE, FASN, FNKF

<https://www.mdcalc.com/calc/113/free-water-deficit-hypernatremia>

# Web App Example

» Next Steps Evidence Creator Insights

## FORMULA

Free water deficit, L = (% total body water, fraction)\*(Weight, kg)\*([Current sodium/Ideal sodium] – 1)

where % total body water (TBW) is:

- Adult male: 60% (i.e., use 0.6 in the equation)
- Adult female: 50% (0.5)
- Elderly male: 50% (0.5)
- Elderly female: 45% (0.45)
- Child: 60% (0.6)

This formula calculates the total solute in the patient's total body water and then determines how much water needs to be added to dilute it to the desired serum sodium.

Sufficient free water should be provided either orally or intravenously (e.g., 5% dextrose) to correct the serum sodium by up to 10 mEq/L in the first 24 hours.

<https://www.mdcalc.com/calc/113/free-water-deficit-hypernatremia>

## Prompt 1:

write me a simple html javascript code to make a web app that will take inputs and give an output of free water deficit using the following equation: Free water deficit,  $L = (\% \text{ total body water, fraction}) * (\text{Weight, kg}) * ([\text{Current sodium}/\text{Ideal sodium}] - 1)$

where % total body water (TBW) is:

Adult male: 60% (i.e., use 0.6 in the equation)

Adult female: 50% (0.5)

Elderly male: 50% (0.5)

Elderly female: 45% (0.45)

Child: 60% (0.6)... please write the html/javascript in one html so i can simply copy and paste






# ChatGPT 5.2

write me a simple html javascript code to make a web app that will take inputs and give an output of free water deficit using the following equation: Free water deficit,  $L = (\% \text{ total body water, fraction}) * (\text{Weight, kg}) * ([\text{Current sodium}/\text{Ideal sodium}] - 1)$  where % total body water (TBW) is: Adult male: 60% (i.e., use 0.6 in the equation) Adult female: 50% (0.5) Elderly male: 50% (0.5) Elderly female: 45% (0.45) Child: 60% (0.6)... please write the html/javascript in one html so i can simply copy and paste

Thought for 8s

Here is a single-file HTML app you can copy and paste:

```
<> HTML      
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8" />  
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  
  <title>Free Water Deficit Calculator</title>  
<style>  
  body {  
    font-family: Arial, sans-serif;  
    background: #f4f7fb;  
    margin: 0;  
    padding: 40px 20px;  
    color: #1f2937;  
  }  
  
  .container {  
    max-width: 520px;  
    margin: 0 auto;  
    background: #ffffff;  
    padding: 28px;  
    border-radius: 14px;  
    box-shadow: 0 6px 18px rgba(0,0,0,0.08);  
  }  
  
  h1 {  
    margin-top: 0;  
    font-size: 1.2em;  
  }  
</style>  
</head>  
<body>  
  <div class="container">  
    <h1>Free Water Deficit Calculator</h1>  
  </div>  
</body>  
</html>
```

## ChatGPT 5.2 ✓

```
        margin-top: 14px;
        font-weight: bold;
    }
</style>
</head>
<body>
  <div class="container">
    <h1>Free Water Deficit Calculator</h1>
    <p class="description">
      Calculates free water deficit using:<br>
      <strong>TBW × Weight × ((Current Na / Ideal Na) - 1)</strong>
    </p>

    <label for="category">Patient Category</label>
    <select id="category">
      <option value="0.6">Adult male (0.6)</option>
      <option value="0.5">Adult female (0.5)</option>
      <option value="0.5">Elderly male (0.5)</option>
      <option value="0.45">Elderly female (0.45)</option>
      <option value="0.6">Child (0.6)</option>
    </select>

    <label for="weight">Weight (kg)</label>
    <input type="number" id="weight" step="0.1" placeholder="Enter weight in kg" />

    <label for="currentNa">Current Sodium (mEq/L)</label>
    <input type="number" id="currentNa" step="0.1" placeholder="Enter current sodium" />

    <label for="idealNa">Ideal Sodium (mEq/L)</label>
    <input type="number" id="idealNa" step="0.1" value="140" placeholder="Enter ideal soc

    <button onclick="calculateDeficit()">Calculate</button>

    <div id="error" class="error"></div>
    <div id="result" class="result" style="display:none;"></div>

    <div class="note">
      This calculator is for educational purposes and should be interpreted in clinical c
    </div>
  </div>

  <script>
    function calculateDeficit() {
      const tbw = parseFloat(document.getElementById("category").value);
      const weight = parseFloat(document.getElementById("weight").value);
      const currentNa = parseFloat(document.getElementById("currentNa").value);
```

## ChatGPT 5.2 ▾

```
HTML

</div>
</div>

<script>
function calculateDeficit() {
  const tbw = parseFloat(document.getElementById("category").value);
  const weight = parseFloat(document.getElementById("weight").value);
  const currentNa = parseFloat(document.getElementById("currentNa").value);
  const idealNa = parseFloat(document.getElementById("idealNa").value);

  const errorDiv = document.getElementById("error");
  const resultDiv = document.getElementById("result");

  errorDiv.textContent = "";
  resultDiv.style.display = "none";

  if (isNaN(weight) || isNaN(currentNa) || isNaN(idealNa)) {
    errorDiv.textContent = "Please enter all values.";
    return;
  }

  if (weight <= 0 || currentNa <= 0 || idealNa <= 0) {
    errorDiv.textContent = "Values must be greater than zero.";
    return;
  }

  const deficit = tbw * weight * ((currentNa / idealNa) - 1);

  resultDiv.innerHTML = `
    <strong>Free Water Deficit:</strong> ${deficit.toFixed(2)} L
  `;
  resultDiv.style.display = "block";
}
</script>
</body>
</html>
```

A small clinical note: this equation gives a positive value when the current sodium is above the ideal sodium.

📄 👍 🗨️ ↗️ ↻️ ⋮

Ask anything

+ 🔄 Extended thinking ▾



## ChatGPT 5.2 ▾

```
HTML

</div>
</div>

<script>
function calculateDeficit() {
  const tbw = parseFloat(document.getElementById("category").value);
  const weight = parseFloat(document.getElementById("weight").value);
  const currentNa = parseFloat(document.getElementById("currentNa").value);
  const idealNa = parseFloat(document.getElementById("idealNa").value);

  const errorDiv = document.getElementById("error");
  const resultDiv = document.getElementById("result");

  errorDiv.textContent = "";
  resultDiv.style.display = "none";

  if (isNaN(weight) || isNaN(currentNa) || isNaN(idealNa)) {
    errorDiv.textContent = "Please enter all values.";
    return;
  }

  if (weight <= 0 || currentNa <= 0 || idealNa <= 0) {
    errorDiv.textContent = "Values must be greater than zero.";
    return;
  }

  const deficit = tbw * weight * ((currentNa / idealNa) - 1);

  resultDiv.innerHTML = `
    <strong>Free Water Deficit:</strong> ${deficit.toFixed(2)} L
  `;
  resultDiv.style.display = "block";
}
</script>
</body>
</html>
```

    **COPY ALL CODE**

A small clinical note: this equation gives a positive value when the current sodium is above the ideal sodium.

     ...

Ask anything

+  Extended thinking ▾



script\_1\_hyper\_na.html M X

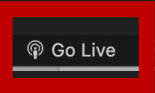
script\_1\_hyper\_na.html &gt; ...

```
1  
2  
3 <!DOCTYPE html>  
4 <html lang="en">  
5 <head>  
6   <meta charset="UTF-8">  
7   <title>Free Water Deficit Calculator</title>  
8  
9   <style>  
10    body {  
11      font-family: Arial, sans-serif;  
12      max-width: 500px;  
13      margin: 40px auto;  
14      line-height: 1.5;  
15    }  
16    label {  
17      display: block;  
18      margin-top: 12px;  
19    }  
20    input, select, button {  
21      width: 100%;  
22      padding: 8px;  
23      margin-top: 4px;  
24      font-size: 14px;  
25    }  
26    button {  
27      margin-top: 20px;  
28      cursor: pointer;  
29    }  
30    .result {  
31      margin-top: 20px;  
32      font-weight: bold;  
33    }  
34  </style>  
35  
36 </head>  
37  
38 <body>  
39  
40   <h2>Free Water Deficit Calculator</h2>  
41  
42   <label>  
43     Patient Category  
44     <select id="category">  
45       <option value="0.6">Adult Male (TBW 0.6)</option>
```

PASTE INTO VISUAL STUDIO CODE HTML FILE

```
1 |
2 |
3 <!DOCTYPE html>
4 <html lang="en">
5 <head>
6 <meta charset="UTF-8">
7 <title>Free Water Deficit Calculator</title>
8
9 <style>
10 body {
11   font-family: Arial, sans-serif;
12   max-width: 500px;
13   margin: 40px auto;
14   line-height: 1.5;
15 }
16 label {
17   display: block;
18   margin-top: 12px;
19 }
20 input, select, button {
21   width: 100%;
22   padding: 8px;
23   margin-top: 4px;
24   font-size: 14px;
25 }
26 button {
27   margin-top: 20px;
28   cursor: pointer;
29 }
30 .result {
31   margin-top: 20px;
32   font-weight: bold;
33 }
34 </style>
35
36 </head>
37
38 <body>
39
40 <h2>Free Water Deficit Calculator</h2>
41
42 <label>
43   Patient Category
44 <select id="category">
45   <option value="0.6">Adult Male (TPW 0.6)</option>
```

CLICK "GO LIVE"



## Free Water Deficit Calculator

Patient Category

Adult Male (TBW 0.6) ▾

Weight (kg)

Current Sodium (mEq/L)

Ideal Sodium (mEq/L)

140

Calculate

# HTML

## Free Water Deficit Calculator

Patient Category

Adult Male (TBW 0.6) ▼

Weight (kg)

Current Sodium (mEq/L)

Ideal Sodium (mEq/L)

140

Calculate

```
<h2>Free Water Deficit Calculator</h2>

<label>
  Patient Category
  <select id="category">
    <option value="0.6">Adult Male (TBW 0.6)</option>
    <option value="0.5">Adult Female (TBW 0.5)</option>
    <option value="0.5">Elderly Male (TBW 0.5)</option>
    <option value="0.45">Elderly Female (TBW 0.45)</option>
    <option value="0.6">Child (TBW 0.6)</option>
  </select>
</label>

<label>
  Weight (kg)
  <input type="number" id="weight" step="0.1">
</label>

<label>
  Current Sodium (mEq/L)
  <input type="number" id="currentNa" step="0.1">
</label>

<label>
  Ideal Sodium (mEq/L)
  <input type="number" id="idealNa" step="0.1" value="140">
</label>

<button onclick="calculateFWD()">Calculate</button>

<div class="result" id="result"></div>
```

# JavaScript

## Free Water Deficit Calculator

Patient Category

Adult Male (TBW 0.6) ✓

Weight (kg)

Current Sodium (mEq/L)

Ideal Sodium (mEq/L)

140

Calculate

```
<script>
function calculateFWD() {
  const tbwFraction = parseFloat(document.getElementById("category").value);
  const weight = parseFloat(document.getElementById("weight").value);
  const currentNa = parseFloat(document.getElementById("currentNa").value);
  const idealNa = parseFloat(document.getElementById("idealNa").value);

  if (
    isNaN(tbwFraction) ||
    isNaN(weight) ||
    isNaN(currentNa) ||
    isNaN(idealNa) ||
    weight <= 0 ||
    idealNa <= 0
  ) {
    document.getElementById("result").innerText =
      "Please enter valid inputs.";
    return;
  }

  const freeWaterDeficit =
    tbwFraction * weight * ((currentNa / idealNa) - 1);

  document.getElementById("result").innerText =
    "Free Water Deficit: " +
    freeWaterDeficit.toFixed(2) +
    " L";
}
</script>
```

# JavaScript

## Free Water Deficit Calculator

Patient Category

Adult Male (TBW 0.6) ✓

Weight (kg)

Current Sodium (mEq/L)

Ideal Sodium (mEq/L)

140

Calculate

```
<script>
function calculateFWD() {
  const tbwFraction = parseFloat(document.getElementById("category").value);
  const weight = parseFloat(document.getElementById("weight").value);
  const currentNa = parseFloat(document.getElementById("currentNa").value);
  const idealNa = parseFloat(document.getElementById("idealNa").value);

  if (
    isNaN(tbwFraction) ||
    isNaN(weight) ||
    isNaN(currentNa) ||
    isNaN(idealNa) ||
    weight <= 0 ||
    idealNa <= 0
  ) {
    document.getElementById("result").innerText =
      "Please enter valid inputs.";
    return;
  }

  const freeWaterDeficit =
    tbwFraction * weight * ((currentNa / idealNa) - 1);

  document.getElementById("result").innerText =
    "Free Water Deficit: " +
    freeWaterDeficit.toFixed(2) +
    " L";
}
</script>
```

**If / Then**  
(branching logic)

# JavaScript

## Free Water Deficit Calculator

Patient Category

Adult Male (TBW 0.6) ✓

Weight (kg)

Current Sodium (mEq/L)

Ideal Sodium (mEq/L)

140

Calculate

```
<script>
function calculateFWD() {
  const tbwFraction = parseFloat(document.getElementById("category").value);
  const weight = parseFloat(document.getElementById("weight").value);
  const currentNa = parseFloat(document.getElementById("currentNa").value);
  const idealNa = parseFloat(document.getElementById("idealNa").value);

  if (
    isNaN(tbwFraction) ||
    isNaN(weight) ||
    isNaN(currentNa) ||
    isNaN(idealNa) ||
    weight <= 0 ||
    idealNa <= 0
  ) {
    document.getElementById("result").innerText =
      "Please enter valid inputs.";
    return;
  }

  const freeWaterDeficit =
    tbwFraction * weight * ((currentNa / idealNa) - 1);

  document.getElementById("result").innerText =
    "Free Water Deficit: " +
    freeWaterDeficit.toFixed(2) +
    " L";
}
</script>
```

**Function**  
(equation within)

# Outline

1. Introducing JavaScript (and HTML)
2. Building blocks of code
3. *Demo 1: a simple browser-based web app*
4. **Privacy / policy considerations**
5. JSON and data structures
6. *Demo 2: big data analysis for research*
7. **Debugging, Limitations, Big Takeaways**

# Avoid Privacy Concerns with HTML/JavaScript

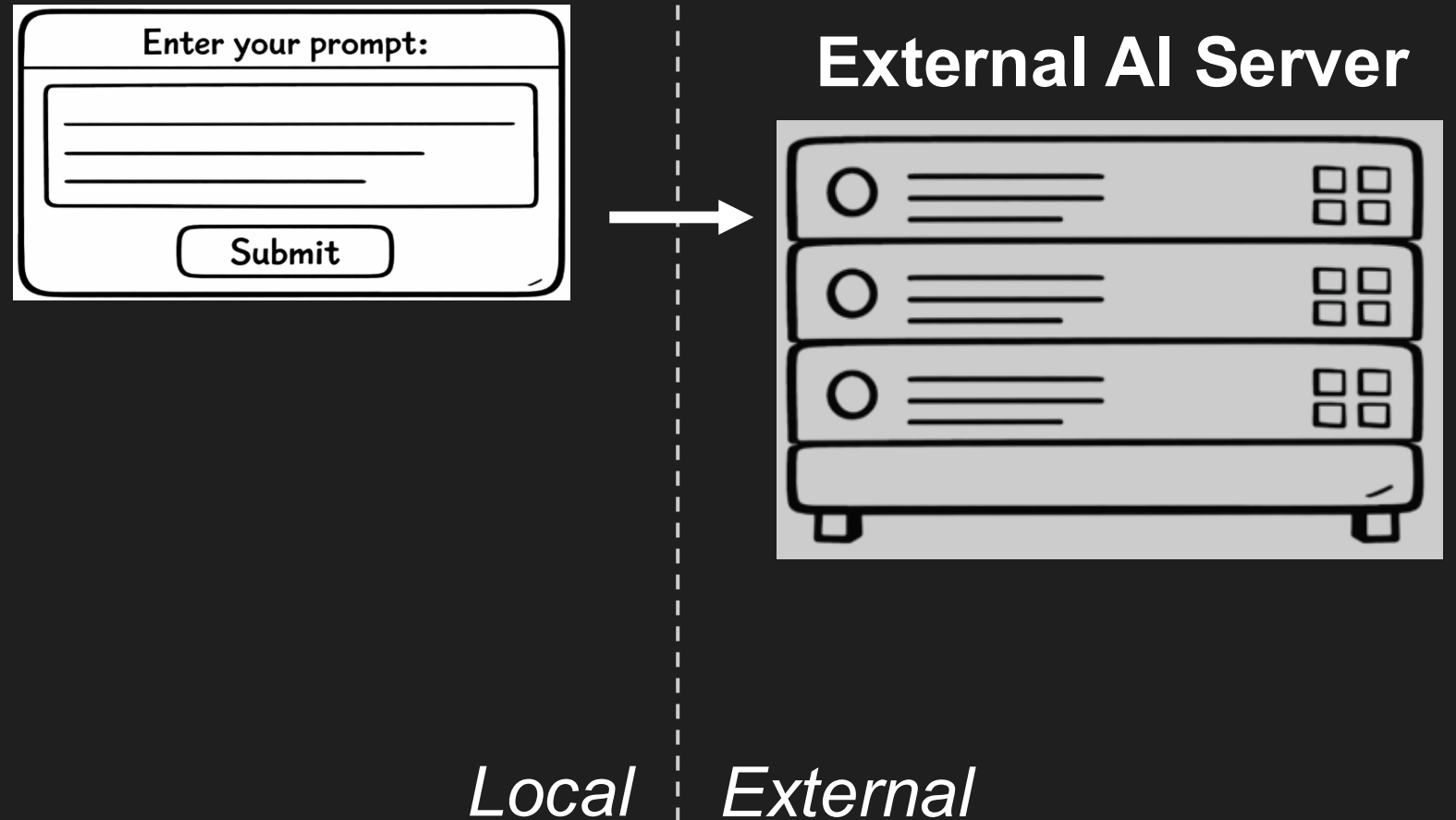


Enter your prompt:

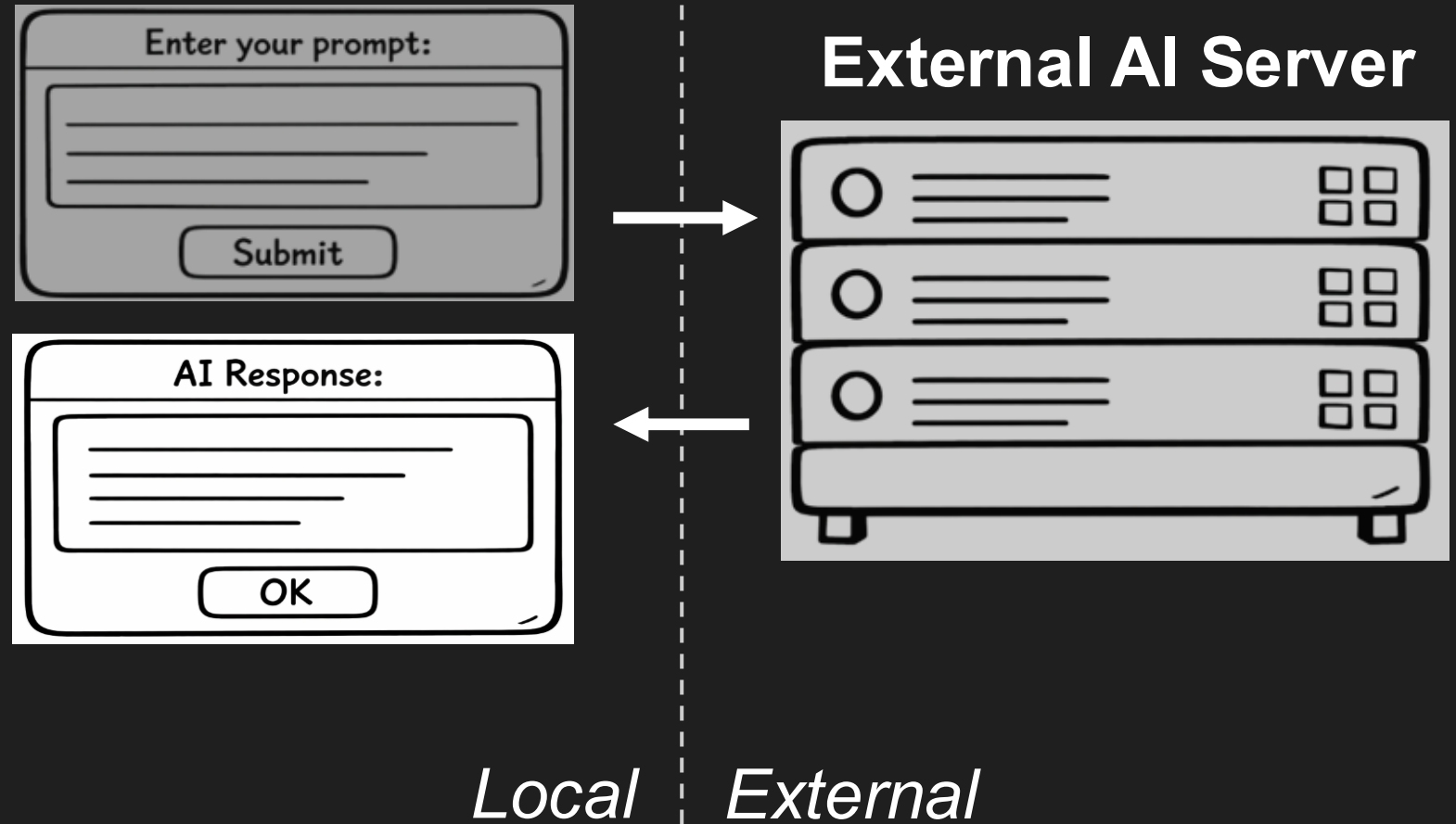
Submit

A simple, cartoon-style form with a rounded rectangular border. At the top, it says "Enter your prompt:". Below this is a text input field with three horizontal lines. At the bottom, there is a rounded rectangular button labeled "Submit".

# Avoid Privacy Concerns with HTML/JavaScript



# Avoid Privacy Concerns with HTML/JavaScript



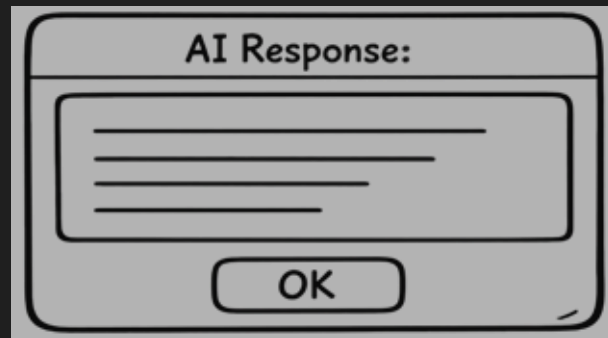
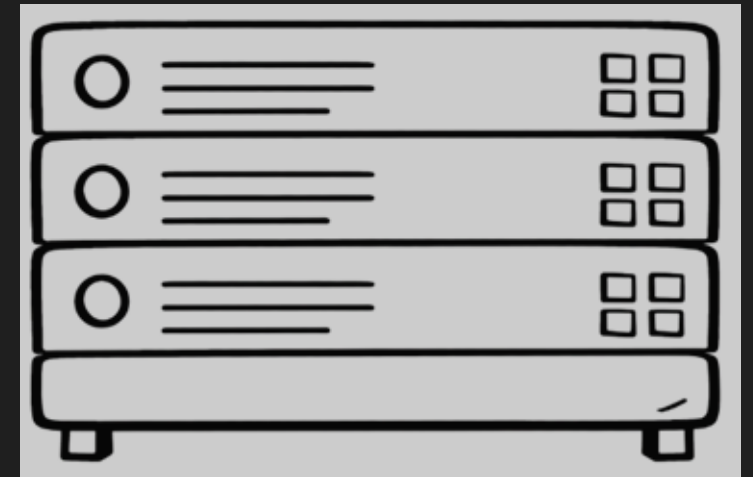
# Avoid Privacy Concerns with HTML/JavaScript

*Cannot enter sensitive data into any non-internal AI model*

*[!]*



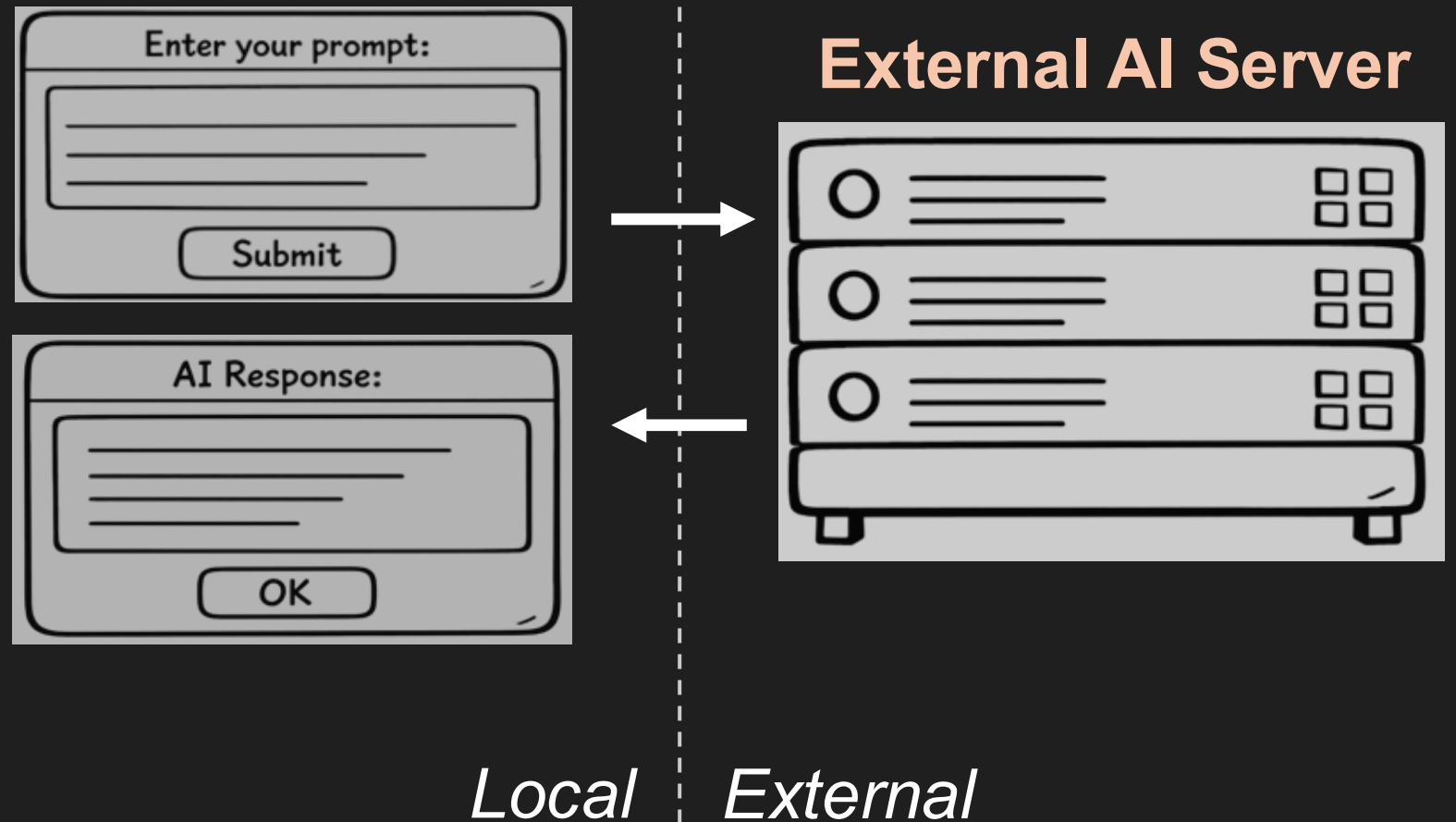
**External AI Server**



*Local* | *External*

# Avoid Privacy Concerns with HTML/JavaScript

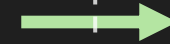
*Use AI to build generic tool to help analyze data locally*



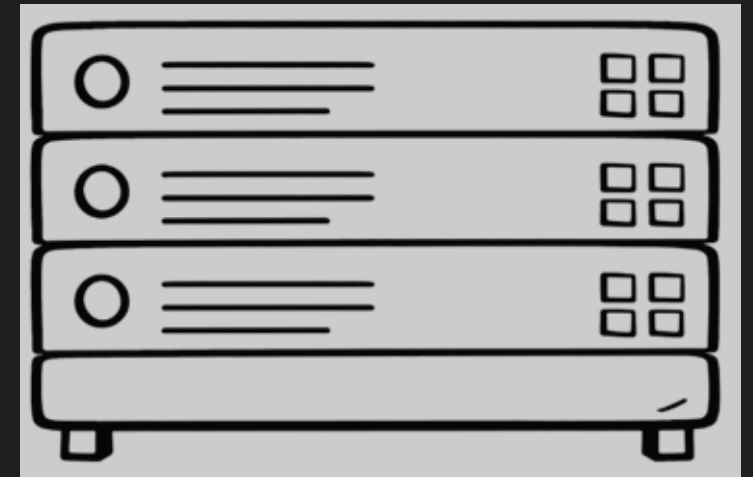
# Avoid Privacy Concerns with HTML/JavaScript

*Use AI to build generic tool to help analyze data locally*

- 1) *Prompt without sensitive data to build local app tool*



**External AI Server**

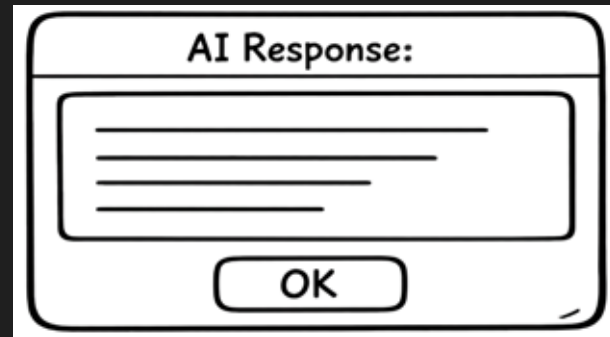


*Local* | *External*

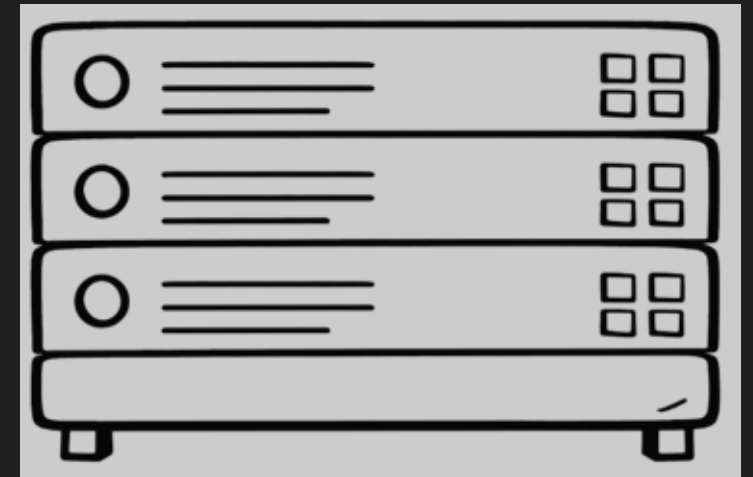
# Avoid Privacy Concerns with HTML/JavaScript

*Use AI to build generic tool to help analyze data locally*

*2) AI returns code for local app*



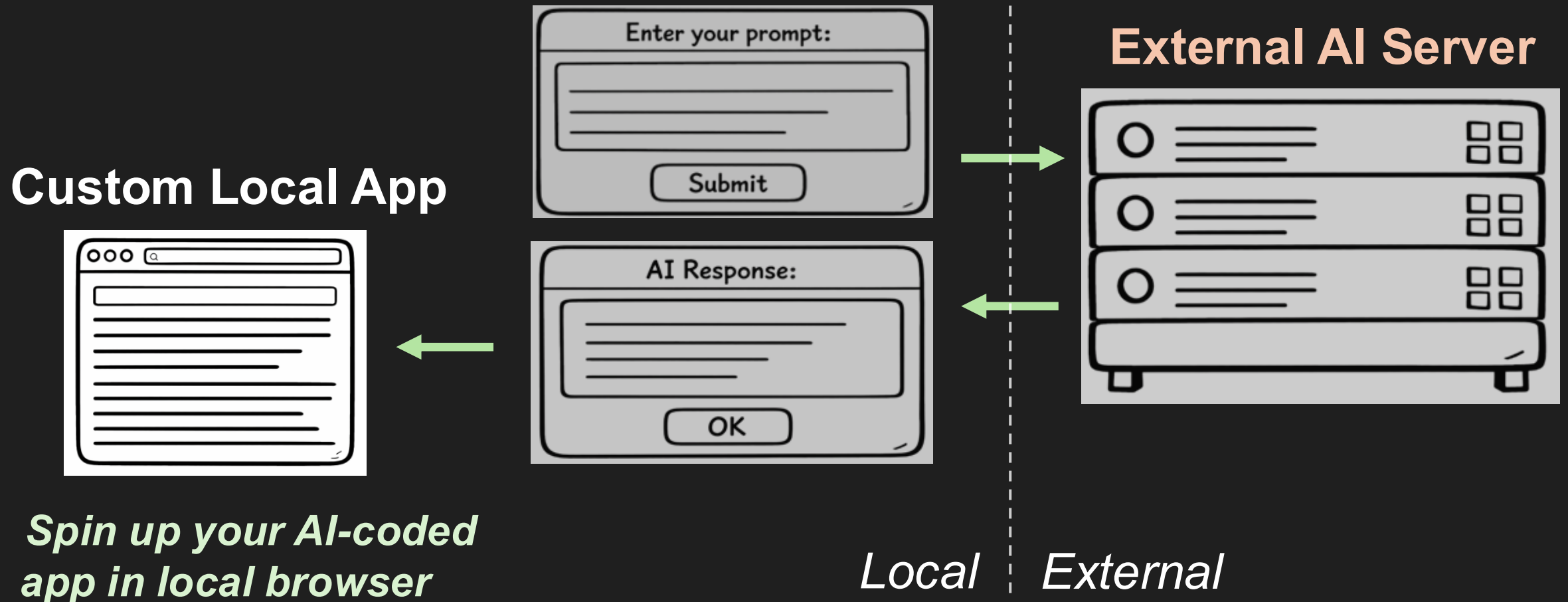
**External AI Server**



*Local* | *External*

# Avoid Privacy Concerns with HTML/JavaScript

*Use AI to build generic tool to help analyze data locally*

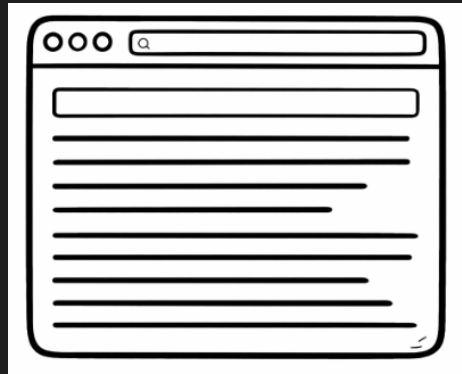


# Avoid Privacy Concerns with HTML/JavaScript

*Use AI to build generic tool to help analyze data locally*

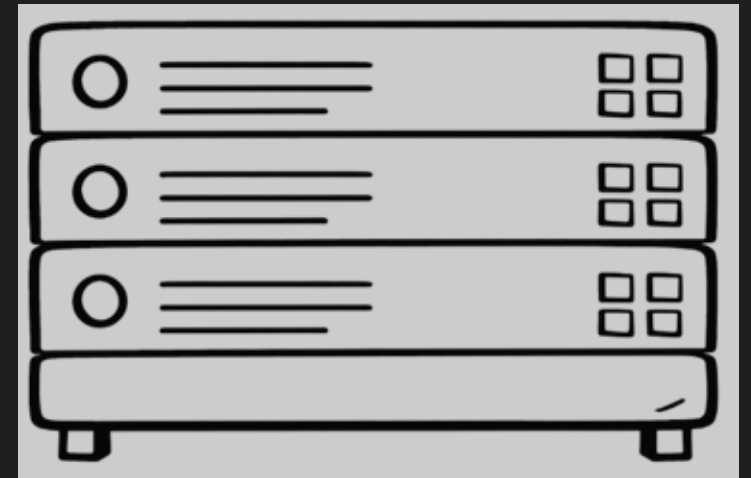
*Now all analysis is local, without any more AI*

## Custom Local App



*Local* | *External*

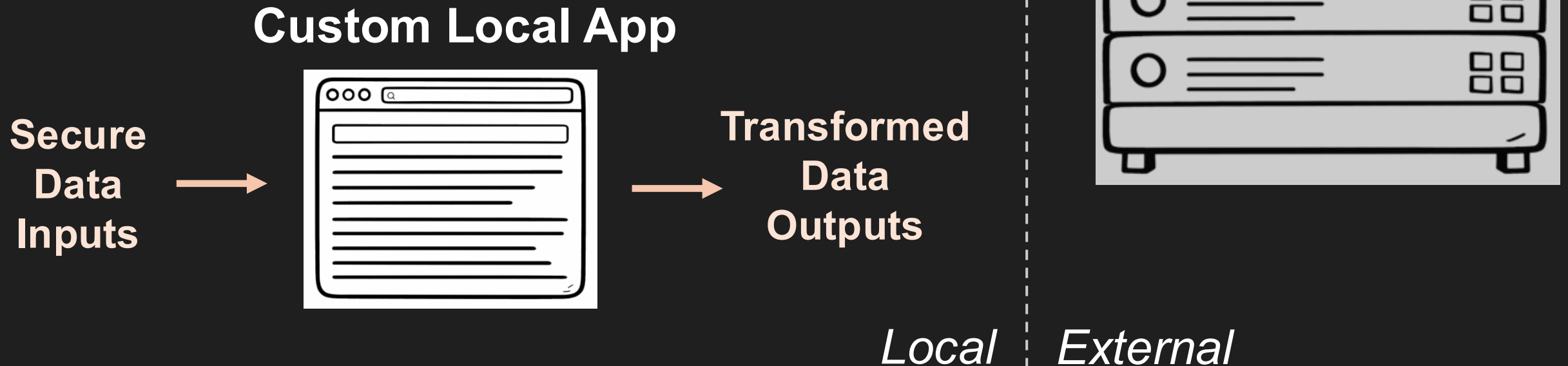
## External AI Server



# Avoid Privacy Concerns with HTML/JavaScript

*Use AI to build generic tool to help analyze data locally*

4) *Transform secure data locally for analysis  
(no AI is used at this critical stage)*



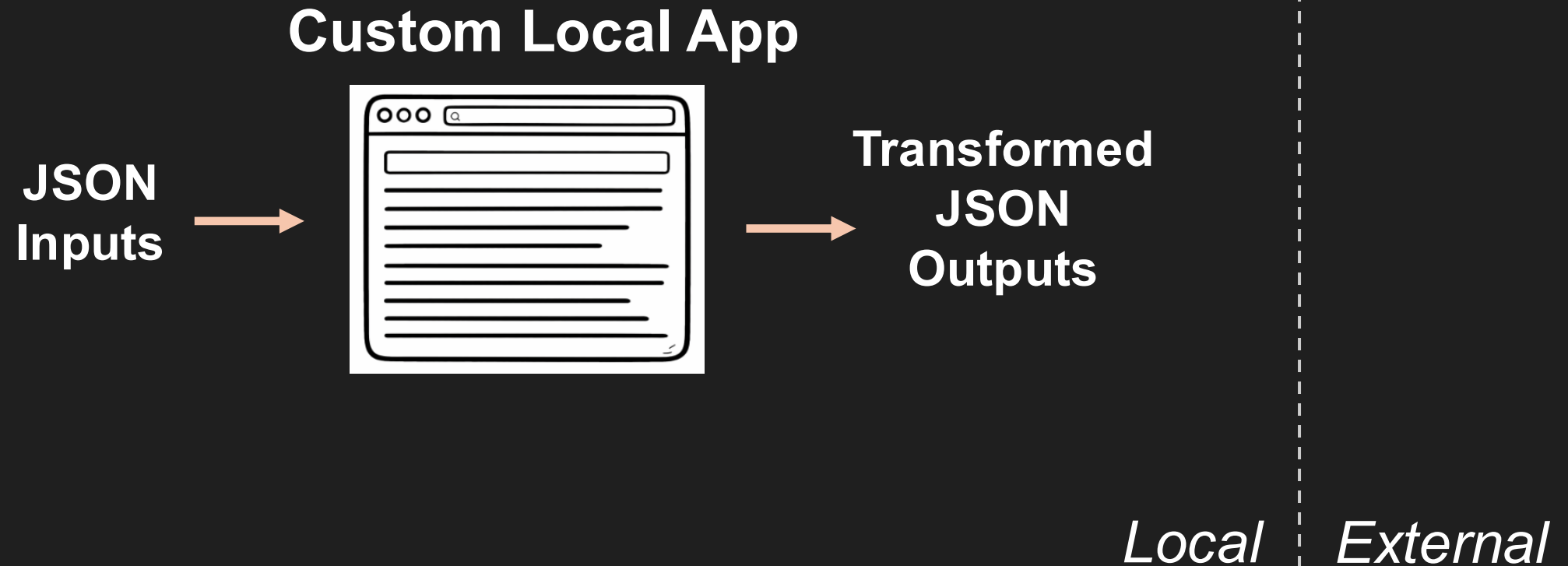
# Outline

1. Introducing JavaScript (and HTML)
2. Building blocks of code
3. *Demo 1: a simple browser-based web app*
4. Privacy / policy considerations
5. **JSON and data structures**
6. *Demo 2: big data analysis for research*
7. **Debugging, Limitations, Big Takeaways**

# JSON

- **Standardized format to store data**
- **Integrates with JavaScript and most other programming languages**
- **Used for APIs (Application Programming Interfaces)**
- **Standardized format for AI to read and write data**

# JSON



# JSON

## Key-value pairs

```
{  
  "name": "rob",  
  "age": 35,  
  "male": true,  
  "weightKg": 70,  
  "sodiumValues": [150, 155, 152]  
}
```

# JSON

## Key-value pairs

```
{  
  "name": "rob",  
  "age": 35,  
  "male": true,  
  "weightKg": 70,  
  "sodiumValues": [150, 155, 152]  
}
```

## Types of data

```
{  
  "name": String,  
  "age": Number,  
  "male": Boolean,  
  "weightKg": Number,  
  "sodiumValues": Array  
}
```

# JSON

Object

```
[{  
  "name": "rob",  
  "age": 35,  
  "male": true,  
  "weightKg": 70,  
  "sodiumValues": [  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:01:00"  
    },  
    {  
      "sodiumValue": 155,  
      "dateTimeStamp": "01/01/2026 00:03:00"  
    },  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:05:00"  
    }  
  ]  
},
```

# JSON

Object

```
[{  
  "name": "rob",  
  "age": 35,  
  "male": true,  
  "weightKg": 70,  
  "sodiumValues": [  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:01:00"  
    },  
    {  
      "sodiumValue": 155,  
      "dateTimeStamp": "01/01/2026 00:03:00"  
    },  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:05:00"  
    }  
  ]  
},
```

String

Boolean



# JSON

Object

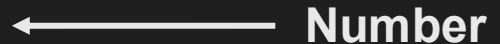
```
[{  
  "name": "rob",  
  "age": 35,  
  "male": true,  
  "weightKg": 70,  
  "sodiumValues": [  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:01:00"  
    },  
    {  
      "sodiumValue": 155,  
      "dateTimeStamp": "01/01/2026 00:03:00"  
    },  
    {  
      "sodiumValue": 150,  
      "dateTimeStamp": "01/01/2026 00:05:00"  
    }  
  ]  
},
```

String

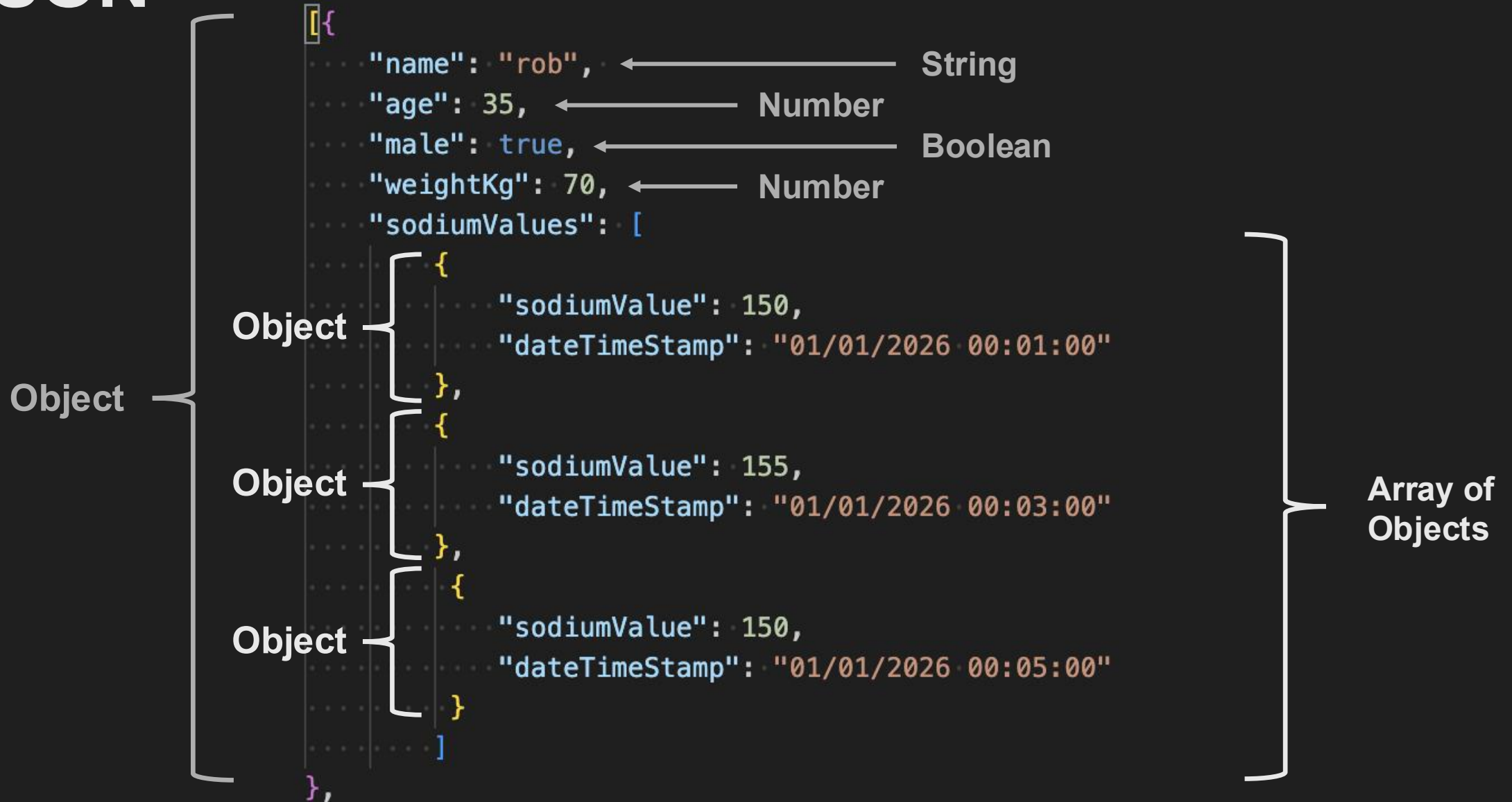
Number

Boolean

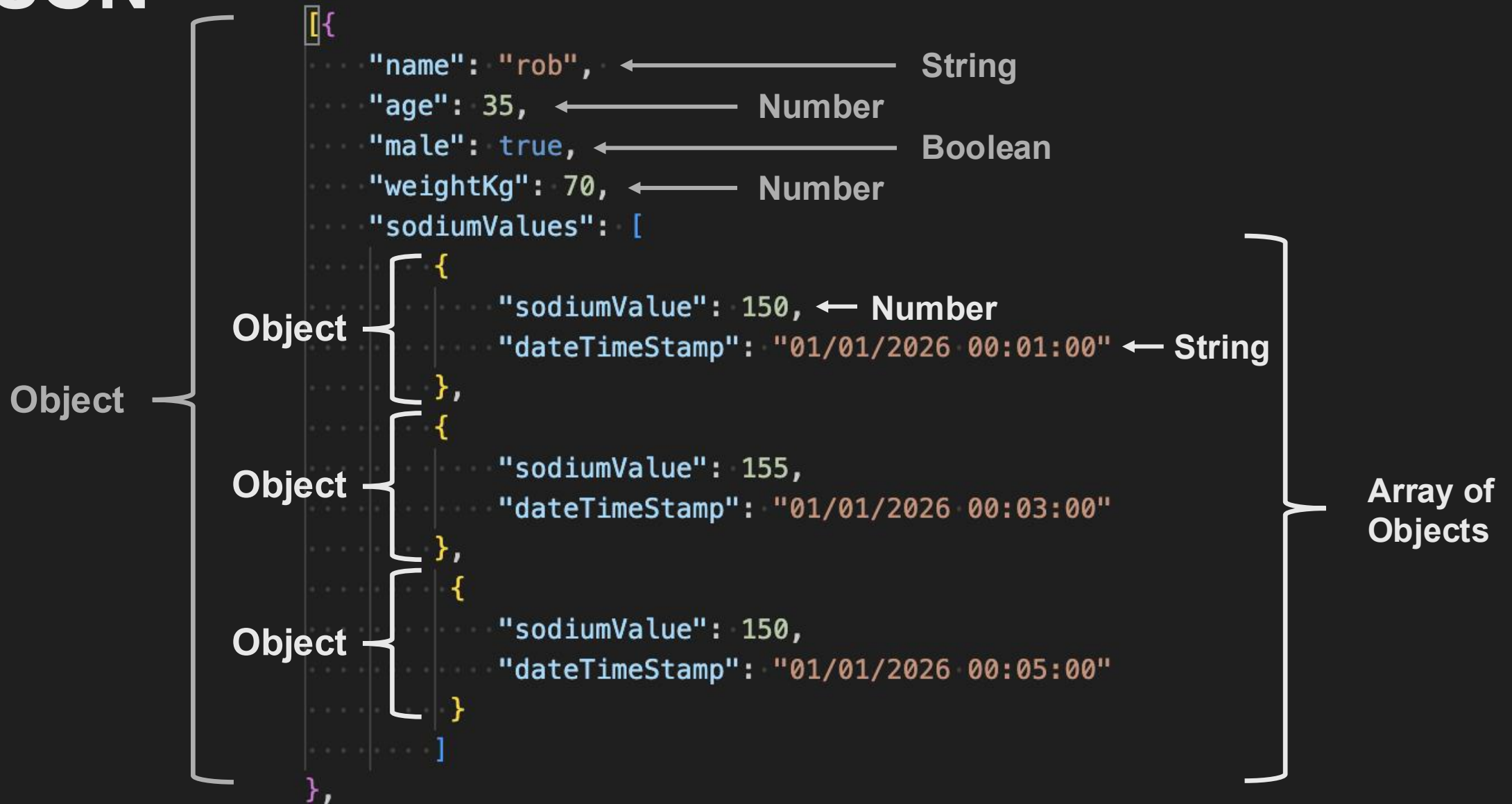
Number



# JSON



# JSON



# JSON

```
[{
  "name": "rob",
  "age": 35,
  "male": true,
  "weightKg": 70,
  "sodiumValues": [
    {
      "sodiumValue": 150,
      "dateTimeStamp": "01/01/2026 00:01:00"
    },
    {
      "sodiumValue": 155,
      "dateTimeStamp": "01/01/2026 00:03:00"
    },
    {
      "sodiumValue": 150,
      "dateTimeStamp": "01/01/2026 00:05:00"
    }
  ]
},
{
  "name": "hayden",
  "age": 55,
  "male": true,
  "weightKg": 80,
  "sodiumValues": [
    {
      "sodiumValue": 165,
      "dateTimeStamp": "01/05/2026 00:01:00"
    }
  ]
}
]
```

Object

# JSON

Array of Objects

```
[{
  "name": "rob",
  "age": 35,
  "male": true,
  "weightKg": 70,
  "sodiumValues": [
    {
      "sodiumValue": 150,
      "dateTimeStamp": "01/01/2026 00:01:00"
    },
    {
      "sodiumValue": 155,
      "dateTimeStamp": "01/01/2026 00:03:00"
    },
    {
      "sodiumValue": 150,
      "dateTimeStamp": "01/01/2026 00:05:00"
    }
  ]
},
{
  "name": "hayden",
  "age": 55,
  "male": true,
  "weightKg": 80,
  "sodiumValues": [
    {
      "sodiumValue": 165,
      "dateTimeStamp": "01/05/2026 00:01:00"
    }
  ]
}
]
```

Object

Object

## ***Prompt 2:***

***synthetic\_data***

I want a simple JavaScript and HTML code (all in one HTML script so I can simply copy/paste the whole script for one file) that will show on the web browser page the value for each object in an array, where the object has the following structure:

```
const patients = [  
{  
  "name": String,  
  "age": Number,  
  "male": Boolean,  
  "weightKg": Number,  
  "sodiumValues": [  
    {  
      "sodiumValue": Number,  
      "dateTimeStamp": String  
    }  
  ],  
  "waterGiven": [  
    {  
      "waterGivenCC": Number,  
      "dateTimeStamp": String  
    }  
  ]  
}
```

The input JSON should go into an input field, such that the user can copy/paste the input JSON before clicking a button that renders it within the application.

```
// -----  
// Render patients to the page  
// -----  
const output = document.getElementById("output");  
  
patients.forEach((patient, index) => {  
  const div = document.createElement("div");  
  div.style.border = "1px solid #ccc";  
  div.style.padding = "10px";  
  div.style.marginBottom = "15px";  
  
  div.innerHTML = `  
    <h3>Patient ${index + 1}</h3>  
    <p><strong>Name:</strong> ${patient.name}</p>  
    <p><strong>Age:</strong> ${patient.age}</p>  
    <p><strong>Male:</strong> ${patient.male}</p>  
    <p><strong>Sodium Values:</strong>  
    <ul>  
      <li>${patient.sodiumValues.map(s =>  
        <li>${s.sodiumValue} mmol/L @ ${s.dateTimeStamp}</li>  
      )}.join("")`  
    </ul>  
    <p><strong>Water Given:</strong></p>  
    <ul>  
      <li>${  
        patient.waterGiven.length === 0  
          ? "<li>None</li>"  
          : patient.waterGiven.map(w =>  
            <li>${w.waterGivenCC} cc @ ${w.dateTimeStamp}</li>  
          ).join("")  
        }  
    </ul>  
  `;  
  ;  
  
  output.appendChild(div);  
});  
</script>
```

Loop

patients.forEach((patient,

127.0.0.1:5500/example\_hyperNa\_code\_1.html

## Patient Data

### Patient 1

**Name:** Alice Johnson

**Age:** 45

**Male:** false

**Weight (kg):** 65

**Sodium Values:**

- 128 mmol/L @ 2026-02-01T08:00
- 130 mmol/L @ 2026-02-01T16:00

**Water Given:**

- 500 cc @ 2026-02-01T09:00

### Patient 2

**Name:** Bob Smith

**Age:** 60

**Male:** true

**Weight (kg):** 82

**Sodium Values:**

- 134 mmol/L @ 2026-02-02T07:30

**Water Given:**

- 250 cc @ 2026-02-02T10:00

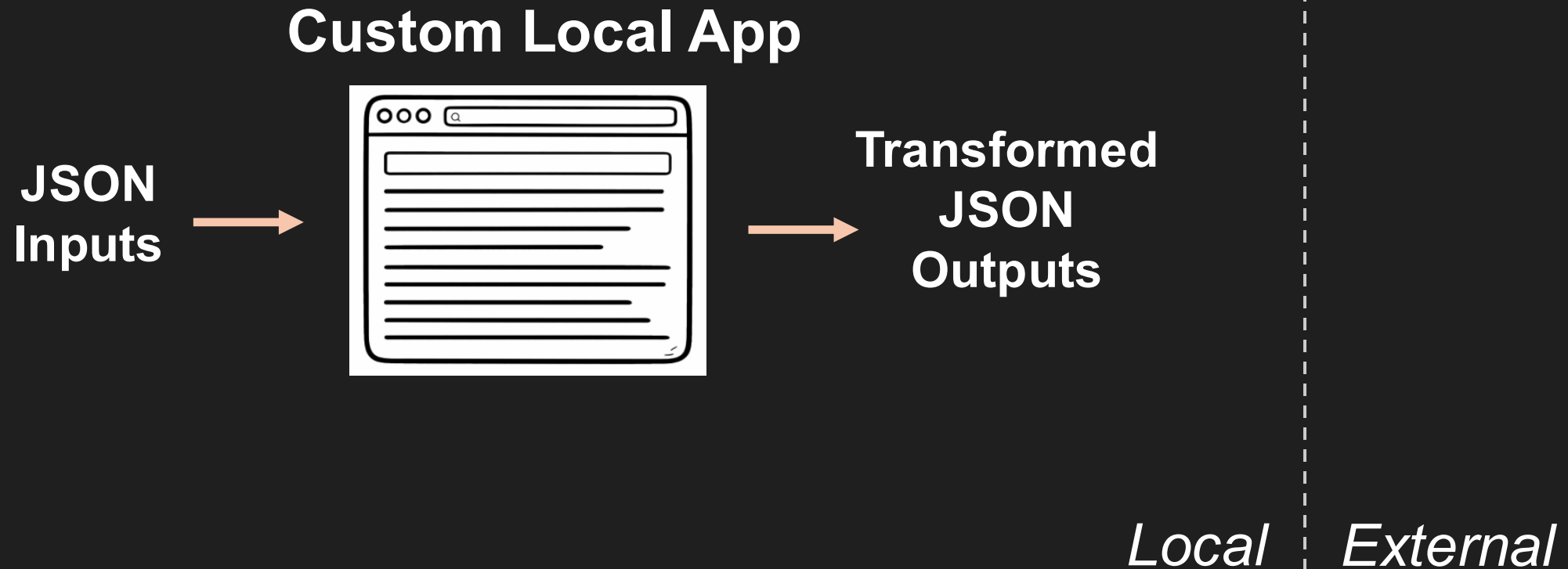
### Patient 3

# Outline

1. Introducing JavaScript (and HTML)
2. Building blocks of code
3. *Demo 1: a simple browser-based web app*
4. Privacy / policy considerations
5. JSON and data structures
6. ***Demo 2: big data analysis for research***
7. Debugging, Limitations, Big Takeaways

# JSON

## Transform Data *Locally*



# Hypothetical Research Example:

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[ \text{Na} ] = \frac{\text{Na}_E + \text{K}_E}{\text{TBW}}$$

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

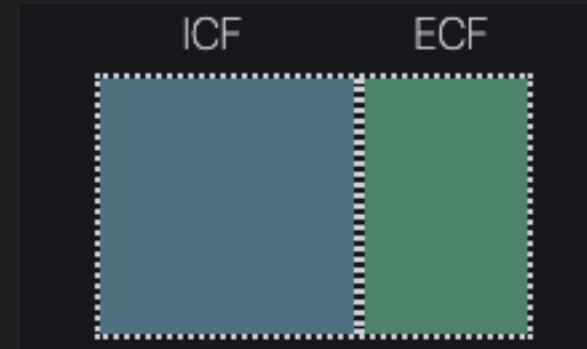
$$[ \text{Na} ] = \frac{\text{Na}_E + \text{K}_E}{\text{TBW}^*}$$

\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[Na]_a = \frac{Na_E + K_E}{TBW_a}$$



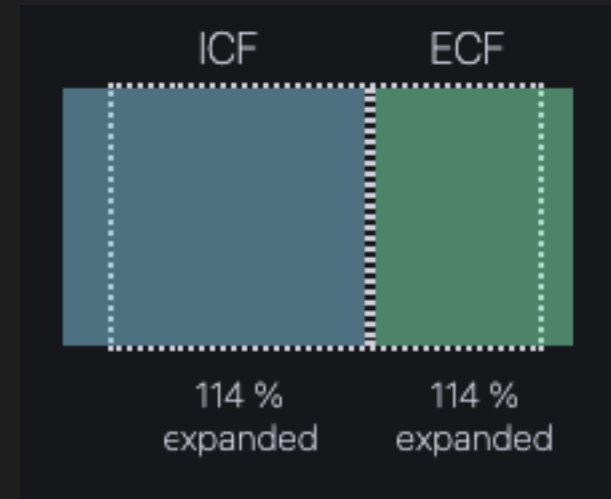
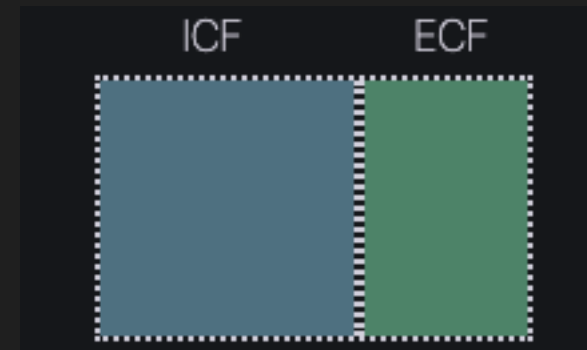
\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[Na]_a = \frac{Na_E + K_E}{TBW_a}$$

$$[Na]_b = \frac{Na_E + K_E}{TBW_a + Water_{interval}}$$

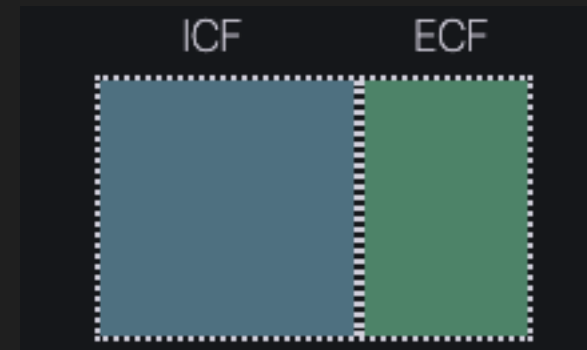


\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{Na_E + K_E}{TBW_a : 42 L}$$

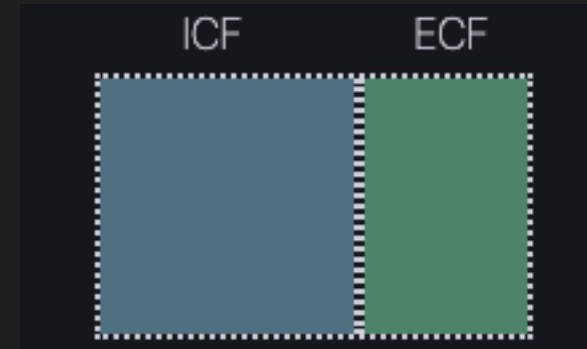


\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{\text{Na}_E + \text{K}_E : 5,880}{\text{TBW}_a : 42 \text{ L}}$$



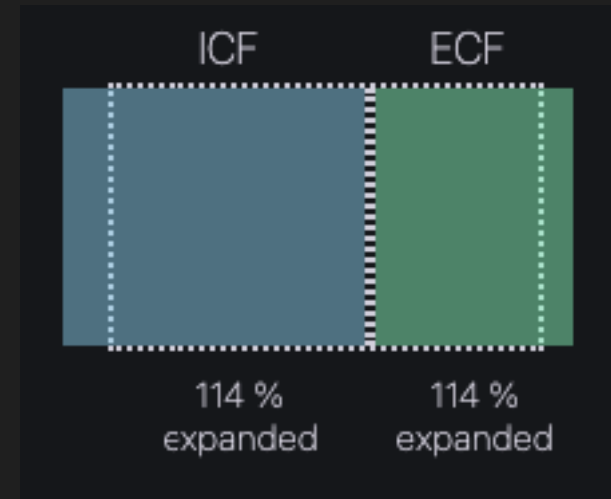
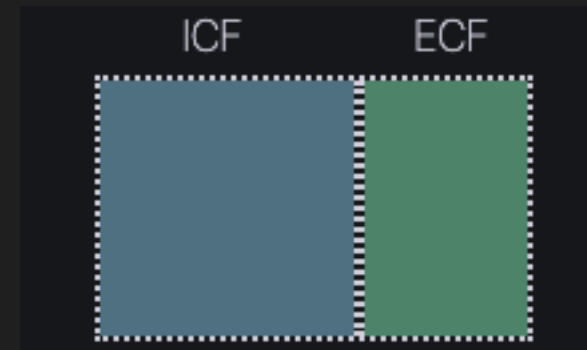
\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{Na_E + K_E : 5,880}{TBW_a : 42 L}$$

$$[Na]_b = \frac{Na_E + K_E}{TBW_a + Water_{interval}}$$



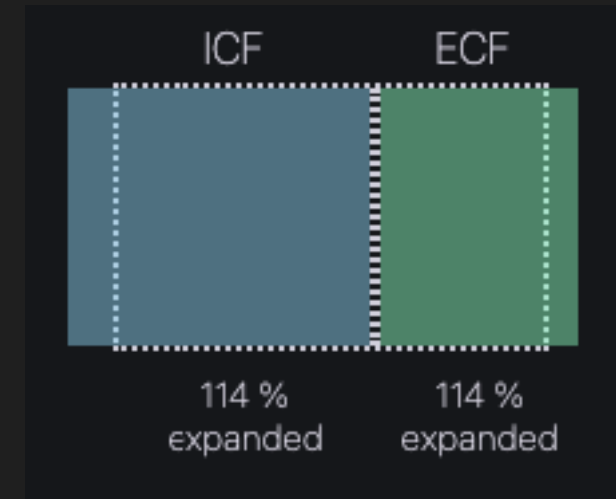
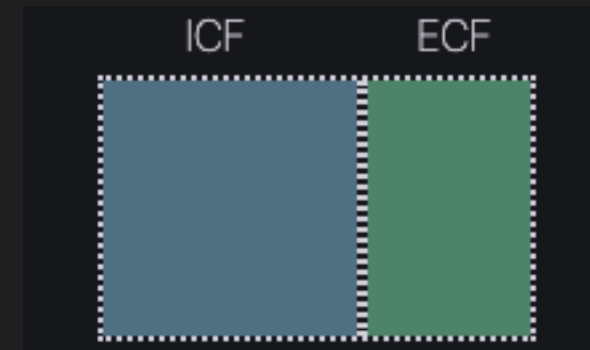
\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{Na_E + K_E : 5,880}{TBW_a : 42 L}$$

$$[Na]_b = \frac{Na_E + K_E : 5,880}{42 L + Water_{interval}}$$



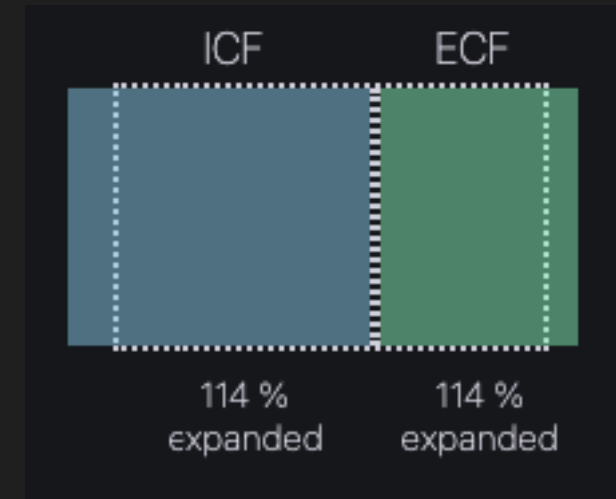
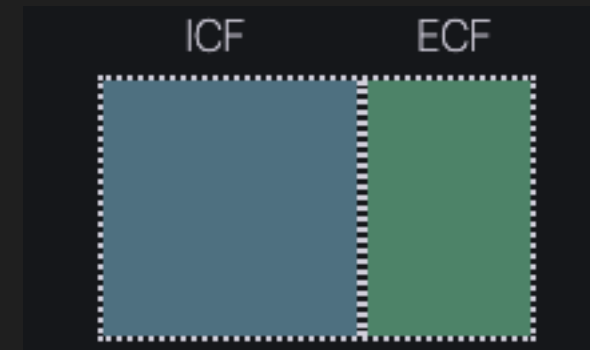
\* Assume TBW (L) = 0.5 x weight (kg)

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{Na_E + K_E : 5,880}{TBW_a : 42 L}$$

$$[Na]_b = \frac{Na_E + K_E : 5,880}{42 L + 6 L_{interval}}$$



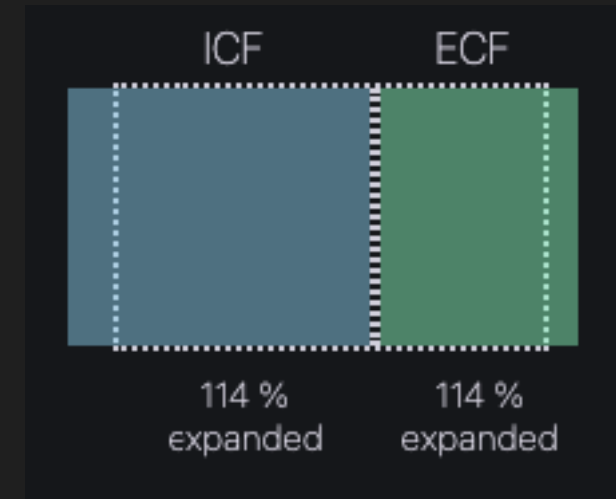
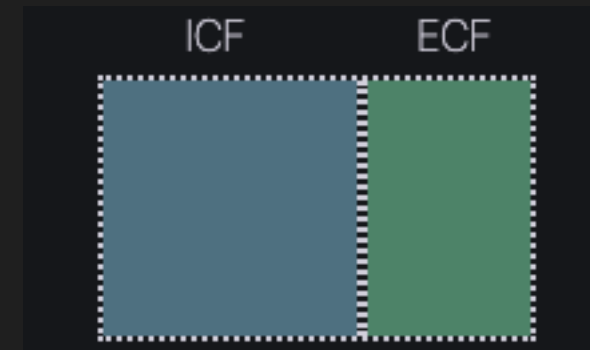
\* Assume  $TBW (L) = 0.5 \times \text{weight (kg)}$

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

$$[140]_a = \frac{\text{Na}_E + \text{K}_E : 5,880}{\text{TBW}_a : 42 \text{ L}}$$

$$[123]_b = \frac{\text{Na}_E + \text{K}_E : 5,880}{42 \text{ L} + 6 \text{ L}_{\text{interval}}}$$



\* Assume TBW (L) = 0.5 x weight (kg)

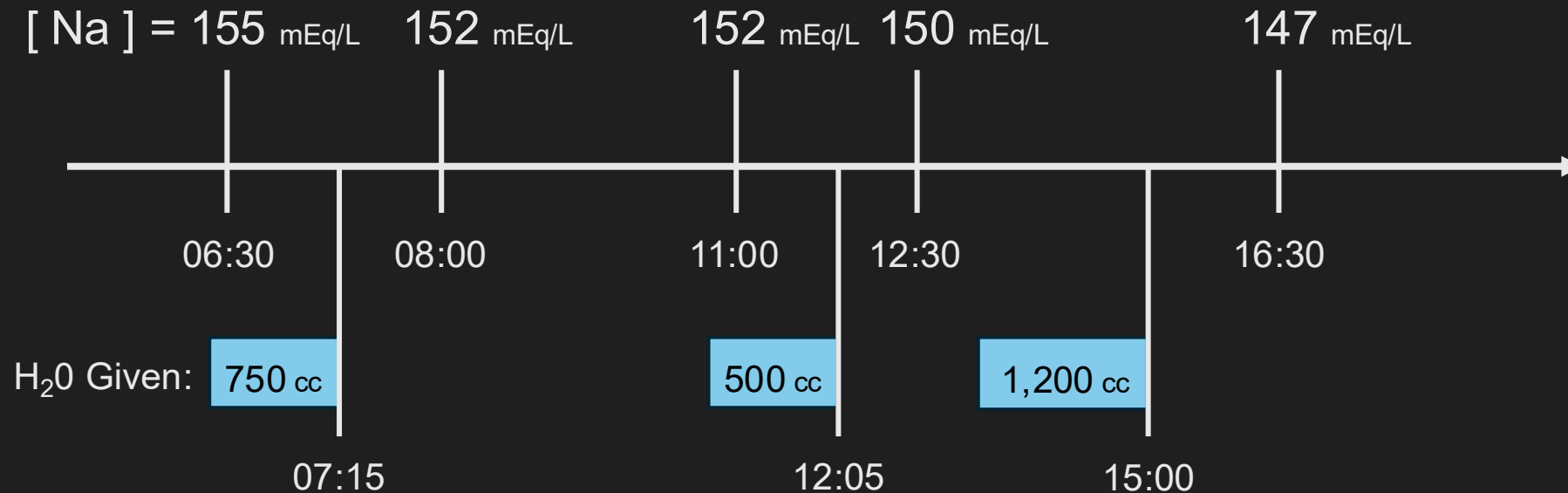
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



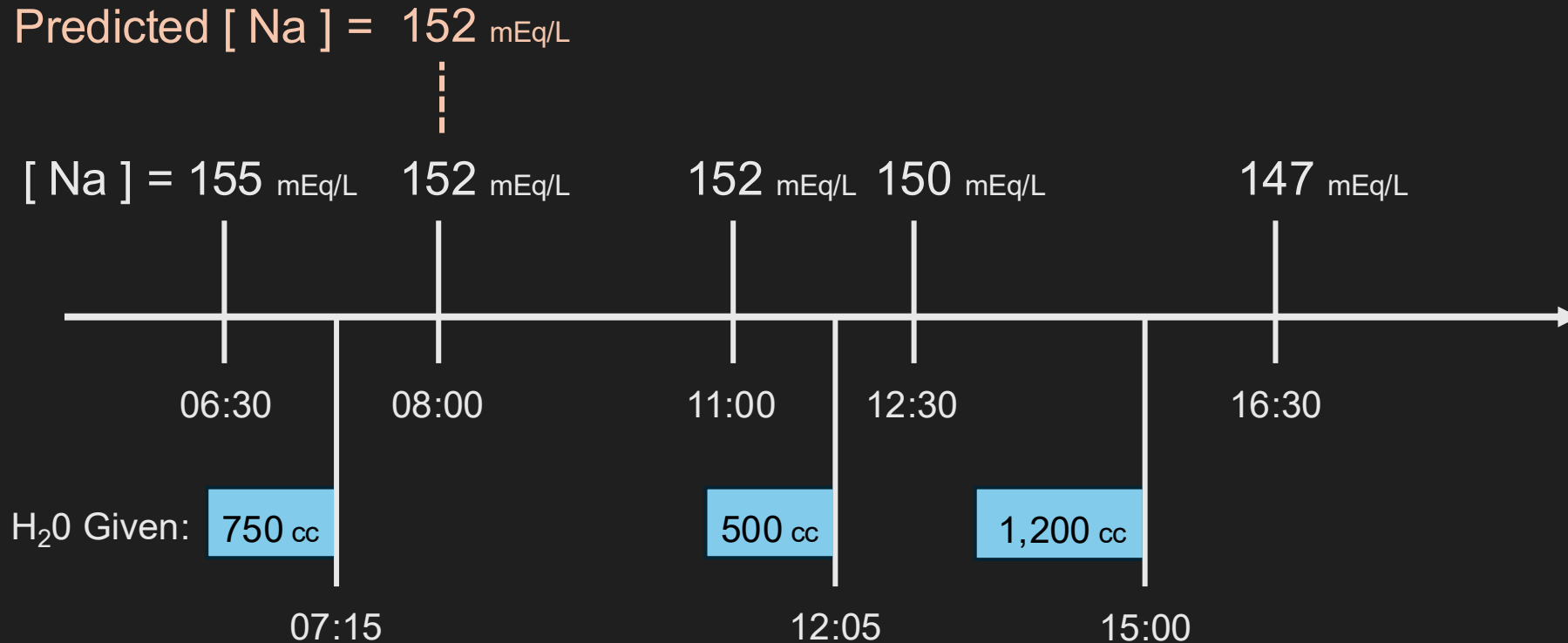
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



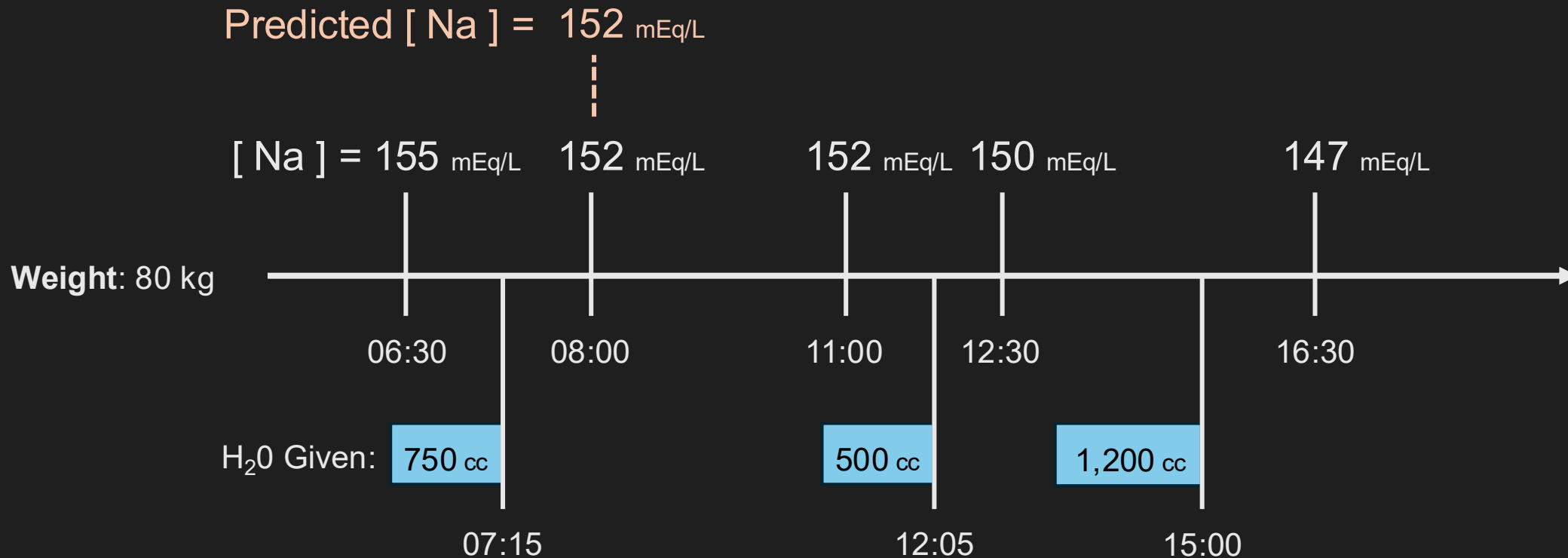
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



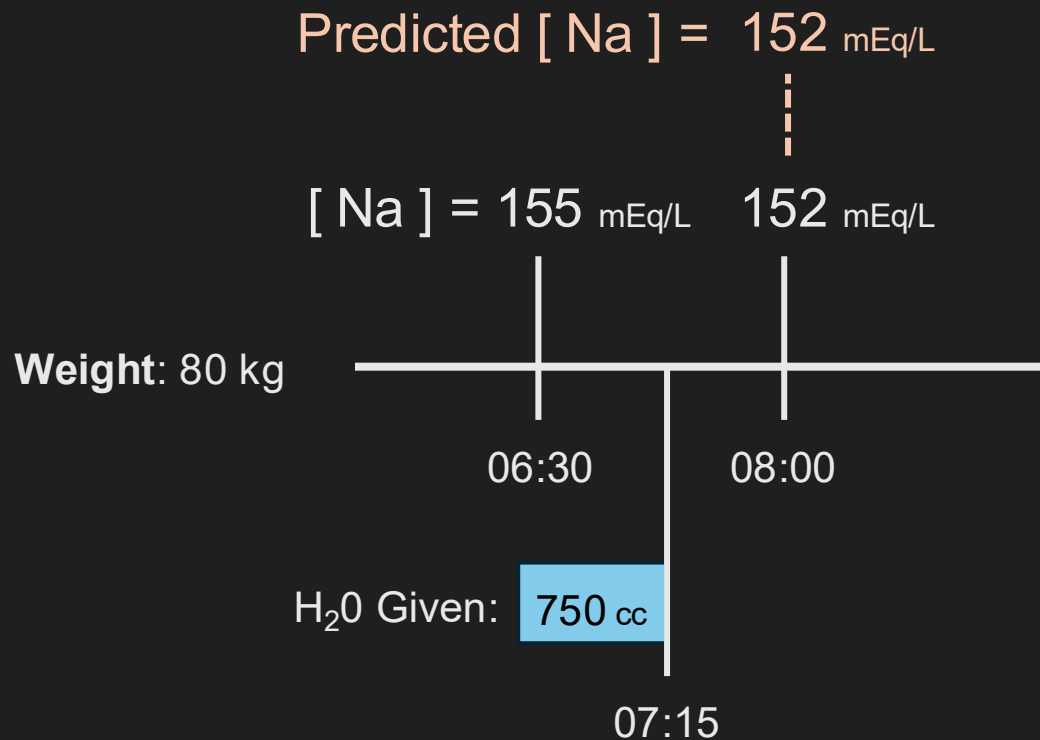
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

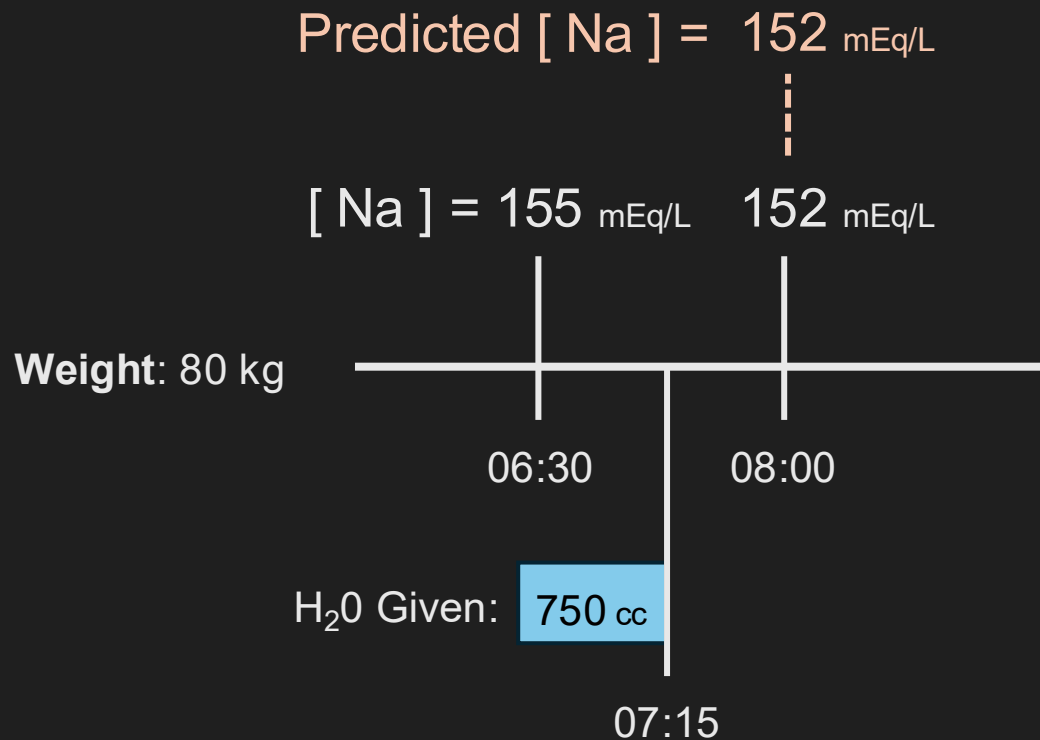
# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



$$[ 155 ]_a = \frac{\text{Na}_E + \text{K}_E : 6,200}{\text{TBW}_a : 40 \text{ L}}$$

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

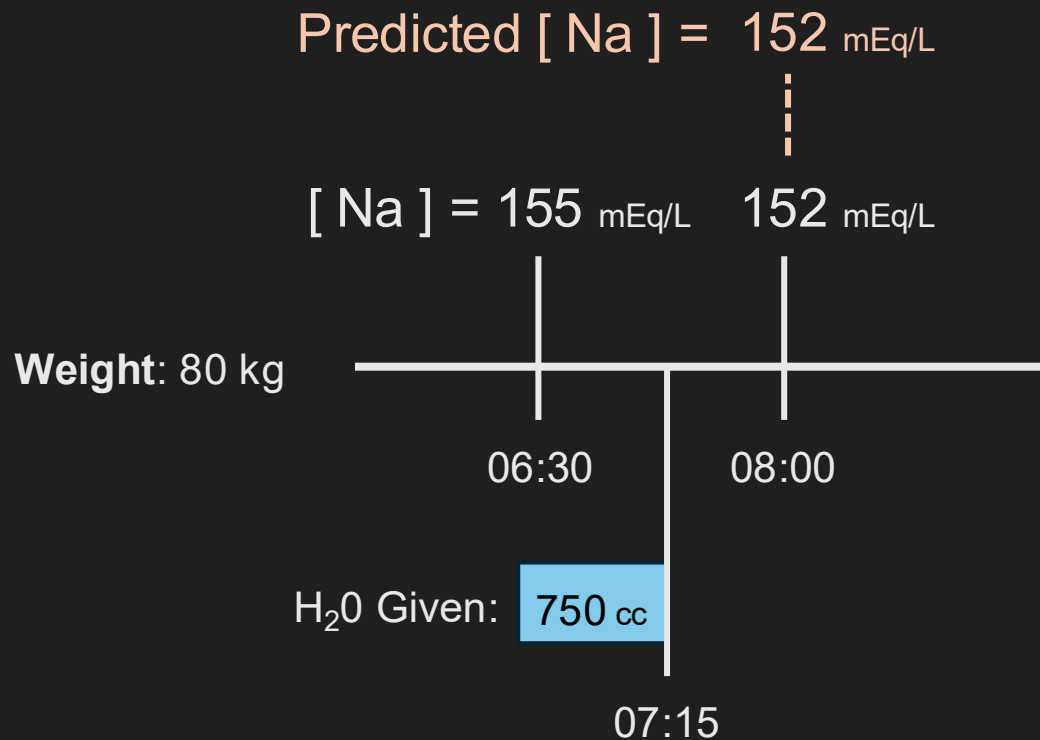


$$[ 155 ]_a = \frac{\text{Na}_E + \text{K}_E : 6,200}{\text{TBW}_a : 40 \text{ L}}$$

$$[ \text{Na} ]_b = \frac{\text{Na}_E + \text{K}_E : 6,200}{40 \text{ L} + 0.75 \text{ L interval}}$$

<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation

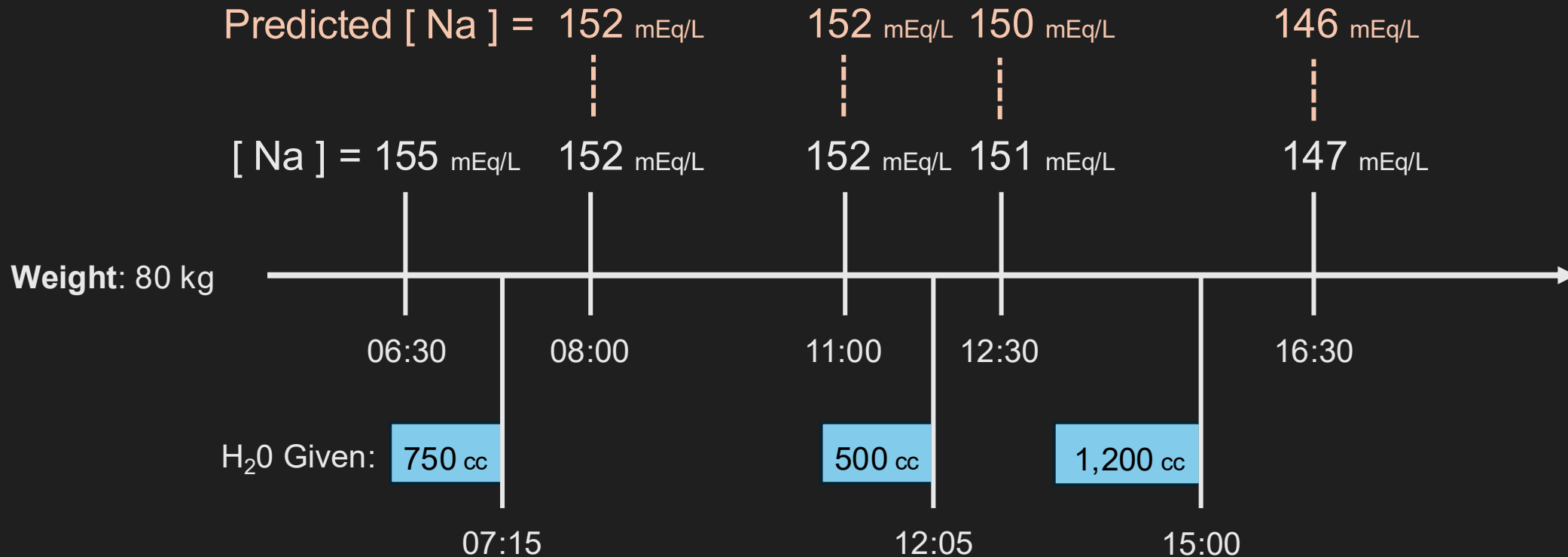


$$[ 155 ]_a = \frac{\text{Na}_E + \text{K}_E : 6,200}{\text{TBW}_a : 40 \text{ L}}$$

$$[ 152 ]_b = \frac{\text{Na}_E + \text{K}_E : 6,200}{40 \text{ L} + 0.75 \text{ L interval}}$$

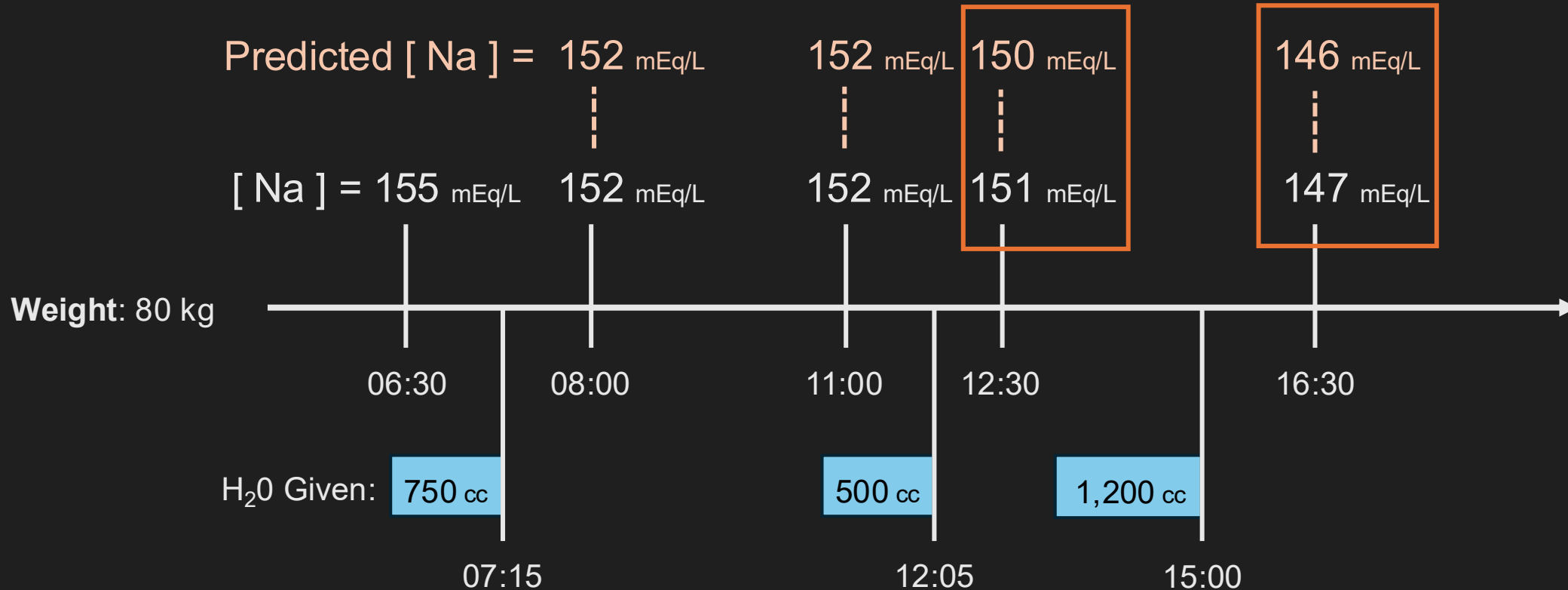
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

# Hypothetical Research Example: Evaluating the Rose<sup>1</sup> equation



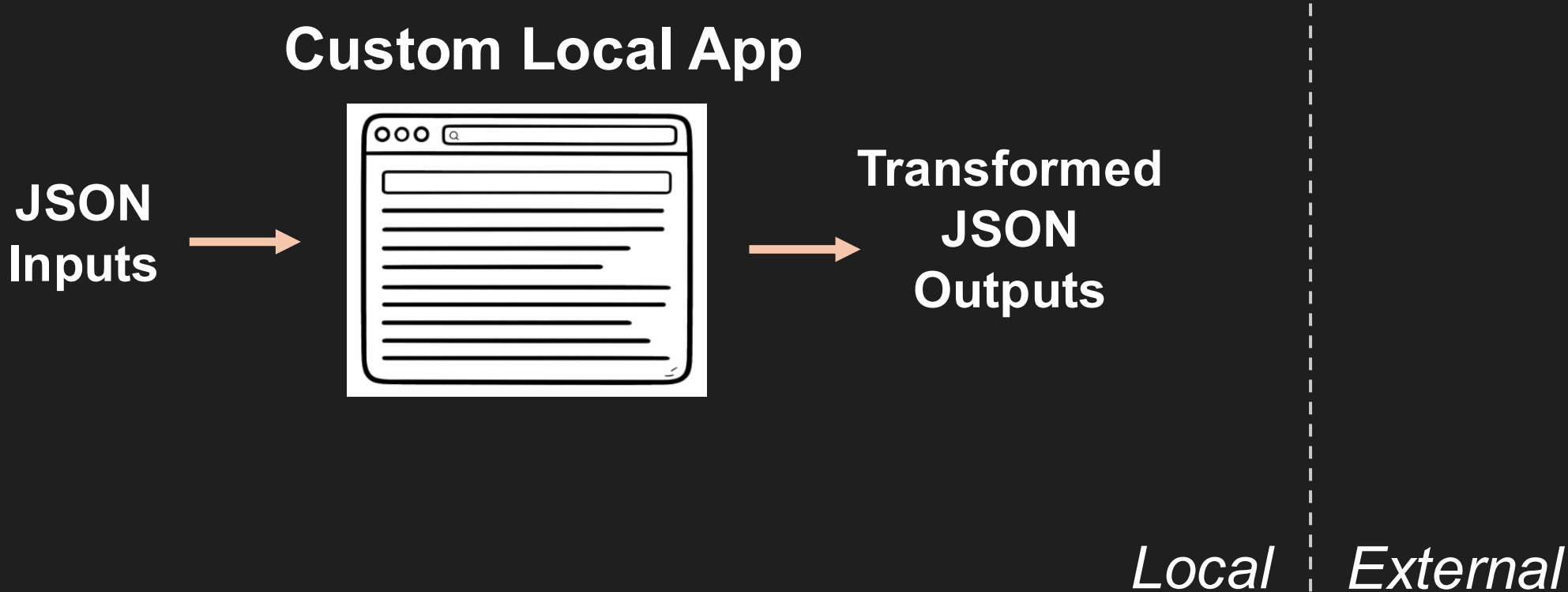
<sup>1</sup> Rose BD, Post TW. Clinical Physiology of Acid-Base and Electrolyte Disorders, 5th ed, McGraw-Hill, New York 2001

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
},
```

*synthetic\_data*

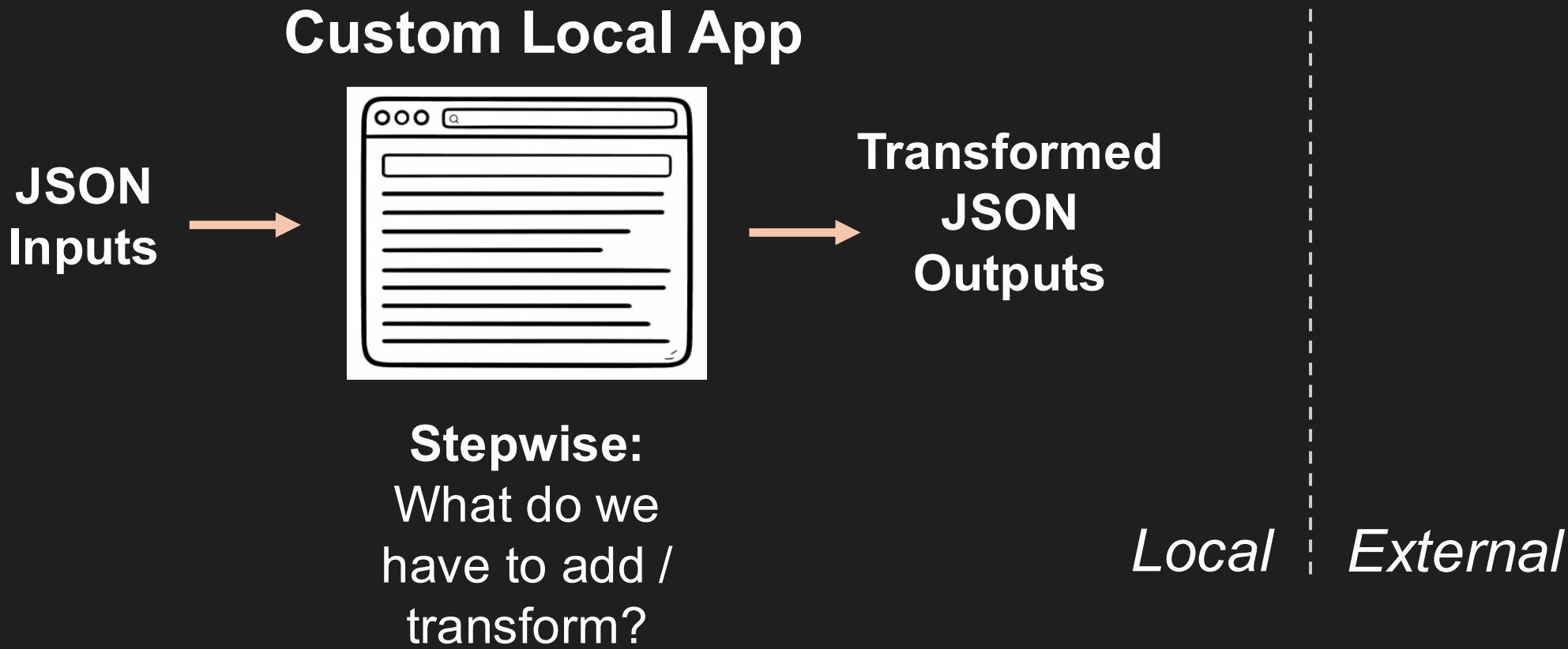
# JSON

## *Pseudo-Code Your App's Function*



# JSON

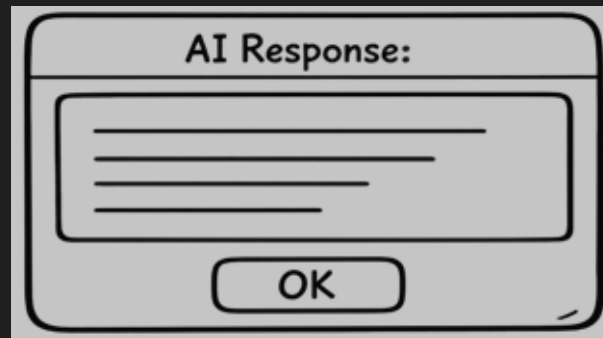
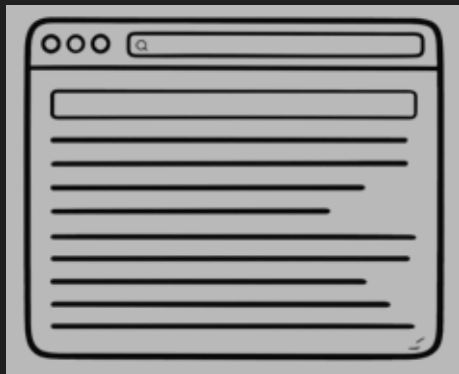
## Pseudo-Code Your App's Function



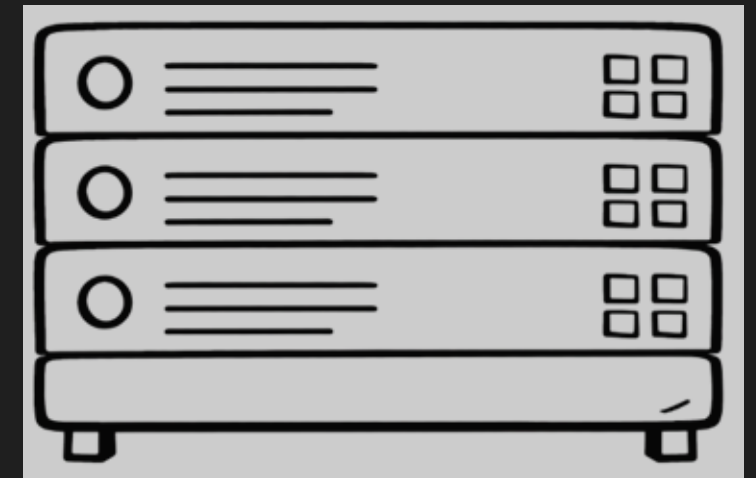
# JSON

## Pseudo-Code:

Custom Local App



External AI Server



Local | External

```
{
  synthetic_data
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
},
```

- 1) For each patient, add a key-value pair for "totalBodyWater" (weight x 0.5)

*synthetic\_data*

*Coding.NephroMaine.com*

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
+ "totalBodyWater": 36,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
},
```

- 1) For each patient, add a key-value pair for "totalBodyWater" (weight x 0.5)

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
},
```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  + "totalCations": 5616,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
}
```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  "totalCations": 5616,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30" },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15" },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00" },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10" },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05" },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20" }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
}
```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues
- 3) For each patient’s sodiumValues objects (except for the first in array, which will have a value of “null” for each new variable), do the following:
  - Add a key-value pair to the sodiumValues object for “newTotalBodyWater”, which is equal to the original value of patient’s total body water plus any additional “waterGivenCC” up until the point in time for the dateTimeStamp value of that sodiumValue

```

{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  "totalCations": 5616,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30",
      "newTotalBodyWater": 36 },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15",
      "newTotalBodyWater": 36.25 },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00",
      "newTotalBodyWater": 36.25 },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10",
      "newTotalBodyWater": 36.75 },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05",
      "newTotalBodyWater": 36.75 },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20",
      "newTotalBodyWater": 37 }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
}

```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues
- 3) For each patient’s sodiumValues objects (except for the first in array, which will have a value of “null” for each new variable), do the following:
  - Add a key-value pair to the sodiumValues object for “newTotalBodyWater”, which is equal to the original value of patient’s total body water plus any additional “waterGivenCC” up until the point in time for the dateTimeStamp value of that sodiumValue

```

{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  "totalCations": 5616,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30",
      "newTotalBodyWater": 36 },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15",
      "newTotalBodyWater": 36.25 },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00",
      "newTotalBodyWater": 36.25 },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10",
      "newTotalBodyWater": 36.75 },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05",
      "newTotalBodyWater": 36.75 },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20",
      "newTotalBodyWater": 37 }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" },
    { "waterGivenCC": 500,
      "dateTimeStamp": "2026-02-01T13:40" },
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T19:30" }
  ]
}

```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues
- 3) For each patient’s sodiumValues objects (except for the first in array, which will have a value of “null” for each new variable), do the following:
  - Add a key-value pair to the sodiumValues object for “newTotalBodyWater”, which is equal to the original value of patient’s total body water plus any additional “waterGivenCC” up until the point in time for the dateTimeStamp value of that sodiumValue
  - Add a key-value pair to the sodiumValues object for “predictedSodium”, which is equal to “totalCations” for the patient divided by “newTotalBodyWater” for that given sodiumValues object, as already calculated in prior step

```
{
  "name": "Patient 1",
  "age": 34,
  "male": true,
  "weightKg": 72,
  "totalBodyWater": 36,
  "totalCations": 5616,
  "sodiumValues": [
    { "sodiumValue": 156,
      "dateTimeStamp": "2026-02-01T06:30",
      "newTotalBodyWater": 36,
      "predictedSodium": null },
    { "sodiumValue": 155,
      "dateTimeStamp": "2026-02-01T09:15",
      "newTotalBodyWater": 36.25,
      "predictedSodium": 155 },
    { "sodiumValue": 153,
      "dateTimeStamp": "2026-02-01T12:00",
      "newTotalBodyWater": 36.25,
      "predictedSodium": 155 },
    { "sodiumValue": 154,
      "dateTimeStamp": "2026-02-01T15:10",
      "newTotalBodyWater": 36.75,
      "predictedSodium": 153 },
    { "sodiumValue": 152,
      "dateTimeStamp": "2026-02-01T18:05",
      "newTotalBodyWater": 36.75,
      "predictedSodium": 153 },
    { "sodiumValue": 150,
      "dateTimeStamp": "2026-02-01T21:20",
      "newTotalBodyWater": 37,
      "predictedSodium": 152 }
  ],
  "waterGiven": [
    { "waterGivenCC": 250,
      "dateTimeStamp": "2026-02-01T07:05" }
  ]
}
```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)
- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues
- 3) For each patient’s sodiumValues objects (except for the first in array, which will have a value of “null” for each new variable), do the following:
  - Add a key-value pair to the sodiumValues object for “newTotalBodyWater”, which is equal to the original value of patient’s total body water plus any additional “waterGivenCC” up until the point in time for the dateTimeStamp value of that sodiumValue
  - Add a key-value pair to the sodiumValues object for “predictedSodium”, which is equal to “totalCations” for the patient divided by “newTotalBodyWater” for that given sodiumValues object, as already calculated in prior step

# Prompt 3:

I need a script (one single file with JavaScript and HTML that will display in a web browser) that will take a JSON (that can be copied and pasted into an input on the web browser) and perform a series of functions outlined in the below instructions to create a new modified JSON, which will show up in the web browser as well. Below is the structure of the original JSON:

```
patient = {
  "name": String,
  "age": Number,
  "male": Boolean,
  "weightKg": Number,
  "sodiumValues": [
    {
      "sodiumValue": Number,
      "dateTimeStamp": String
    }
  ],
  "waterGiven": [
    {
      "waterGivenCC": Number,
      "dateTimeStamp": String
    }
  ]
}
```

The format of the dateTimeStamp string, which provides the date/time stamp, is like this example: "2026-02-02T06:00"

There should be a download button that allows the user to download the new modified JSON file.

Here are the instructions to take the initial JSON and end up with the new modified JSON:

1) For each patient, add a key-value pair for "totalBodyWater" (weight x 0.5)

2) For each patient, add a key-value pair for "totalCations", which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues

3) For each patient's sodiumValues objects (except for the first in array, which will have a value of "null" for each new variable), do the following:

3a) Add a key-value pair to the sodiumValues object for "newTotalBodyWater", which is equal to the original value of patient's total body water plus any additional "waterGivenCC" up until the point in time for the dateTimeStamp value of that sodiumValue

3b) Add a key-value pair to the sodiumValues object for "predictedSodium", which is equal to "totalCations" for the patient divided by "newTotalBodyWater" for that given sodiumValues object, as already calculated in prior step

Provide the entire HTML/JavaScript code so I can simply copy and paste

# Patient JSON Processor

TBW • Total Cations • Predicted Sodium

Paste a **single patient object** or an **array of patient objects** into the input. Click **Process** to generate a modified JSON with:  $\text{patient.totalBodyWater} = \text{weightKg} \times 0.5$ ,  $\text{patient.totalCations} = \text{totalBodyWater} \times \text{first sodiumValue}$ , and for each sodiumValues entry after the first:  $\text{newTotalBodyWater}$  and  $\text{predictedSodium}$ .

## Input JSON

```
[
  {
    "name": "Patient 1",
    "age": 34,
    "male": true,
    "weightKg": 72,
    "sodiumValues": [
      { "sodiumValue": 156, "dateTimeStamp": "2026-02-01T06:30" },
      { "sodiumValue": 155, "dateTimeStamp": "2026-02-01T09:15" },
      { "sodiumValue": 153, "dateTimeStamp": "2026-02-01T12:00" },
      { "sodiumValue": 154, "dateTimeStamp": "2026-02-01T15:10" },
      { "sodiumValue": 152, "dateTimeStamp": "2026-02-01T18:05" },
      { "sodiumValue": 150, "dateTimeStamp": "2026-02-01T21:20" }
    ]
  }
]
```

Process

Load Example

Clear

Date/time stamps should look like 2026-02-02T06:00. Water is summed if its timestamp is  $\leq$  the sodium timestamp.

Ready.

## Modified Output JSON

Output will appear here...

Copy Output

Download JSON

Download uses your browser's standard file download (a .json file).

# Patient JSON Processor

TBW • Total Cations • Predicted Sodium

Paste a **single patient object** or an **array of patient objects** into the input. Click **Process** to generate a modified JSON with:  $\text{patient.totalBodyWater} = \text{weightKg} \times 0.5$ ,  $\text{patient.totalCations} = \text{totalBodyWater} \times \text{first sodiumValue}$ , and for each sodiumValues entry after the first: `newTotalBodyWater` and `predictedSodium`.

## Input JSON

```
[
  {
    "name": "Patient 1",
    "age": 34,
    "male": true,
    "weightKg": 72,
    "sodiumValues": [
      { "sodiumValue": 156, "dateTimeStamp": "2026-02-01T06:30" },
      { "sodiumValue": 155, "dateTimeStamp": "2026-02-01T09:15" },
      { "sodiumValue": 153, "dateTimeStamp": "2026-02-01T12:00" },
      { "sodiumValue": 154, "dateTimeStamp": "2026-02-01T15:10" },
      { "sodiumValue": 152, "dateTimeStamp": "2026-02-01T18:05" },
      { "sodiumValue": 150, "dateTimeStamp": "2026-02-01T21:20" }
    ]
  }
]
```

Process

Load Example

Clear

Date/time stamps should look like `2026-02-02T06:00`. Water is summed if its timestamp is  $\leq$  the sodium timestamp.

Ready.

## Modified Output JSON

```
    "waterGivenCC": 250,
    "dateTimeStamp": "2026-02-01T07:05"
  },
  {
    "waterGivenCC": 500,
    "dateTimeStamp": "2026-02-01T13:40"
  },
  {
    "waterGivenCC": 250,
    "dateTimeStamp": "2026-02-01T19:30"
  }
],
  {
    "totalBodyWater": 36,
    "totalCations": 5616
  },
  {
    "name": "Patient 2",
    "age": 58,
    "male": false,
```

Copy Output

Download JSON

Download uses your browser's standard file download (a .json file).

# Patient JSON Processor

TBW • Total Cations • Predicted Sodium

Paste a **single patient object** or an **array of patient objects** into the input. Click **Process** to generate a modified JSON with:  $\text{patient.totalBodyWater} = \text{weightKg} \times 0.5$ ,  $\text{patient.totalCations} = \text{totalBodyWater} \times \text{first sodiumValue}$ , and for each sodiumValues entry after the first: `newTotalBodyWater` and `predictedSodium`.

## Input JSON

```
[
  {
    "name": "Patient 1",
    "age": 34,
    "male": true,
    "weightKg": 72,
    "sodiumValues": [
      { "sodiumValue": 156, "dateTimeStamp": "2026-02-01T06:30" },
      { "sodiumValue": 155, "dateTimeStamp": "2026-02-01T09:15" },
      { "sodiumValue": 153, "dateTimeStamp": "2026-02-01T12:00" },
      { "sodiumValue": 154, "dateTimeStamp": "2026-02-01T15:10" },
      { "sodiumValue": 152, "dateTimeStamp": "2026-02-01T18:05" },
      { "sodiumValue": 150, "dateTimeStamp": "2026-02-01T21:20" }
    ]
  }
]
```

Process

Load Example

Clear

Date/time stamps should look like `2026-02-02T06:00`. Water is summed if its timestamp is  $\leq$  the sodium timestamp.

Ready.

## Modified Output JSON

```
[
  {
    "name": "Patient 1",
    "age": 34,
    "male": true,
    "weightKg": 72,
    "sodiumValues": [
      {
        "sodiumValue": 156,
        "dateTimeStamp": "2026-02-01T06:30",
        "newTotalBodyWater": null,
        "predictedSodium": null
      },
      {
        "sodiumValue": 155,
        "dateTimeStamp": "2026-02-01T09:15",
        "newTotalBodyWater": 36.25,
        "predictedSodium": 154.9241379310345
      },
    ]
  }
]
```

Copy Output

Download JSON

Download uses your browser's standard file download (a .json file).

```
... // Main transform for one patient
... function transformPatient(patient) {
...   const p = deepClone(patient);
...
...   // 1) totalBodyWater = weightKg * 0.5
...   const weightKg = safeNumber(p.weightKg);
...   const totalBodyWater = (weightKg === null) ? null : weightKg * 0.5;
...   p.totalBodyWater = totalBodyWater;
...
...   // Guard: sodiumValues array
...   const sv = Array.isArray(p.sodiumValues) ? p.sodiumValues : [];
...   if (sv.length === 0) {
...     // Still add totalCations as null if we can't compute
...     p.totalCations = null;
...     p.sodiumValues = sv;
...   }
...   return p;
... }
...
... // 2) totalCations = totalBodyWater * first sodiumValue
... const firstNa = safeNumber(sv[0]? sodiumValue);
... p.totalCations = (totalBodyWater === null || firstNa === null) ? null : totalBodyWater * firstNa;
```

- 1) For each patient, add a key-value pair for “totalBodyWater” (weight x 0.5)

```
... // Main transform for one patient
... function transformPatient(patient) {
...   const p = deepClone(patient);

...   // 1) totalBodyWater = weightKg * 0.5
...   const weightKg = safeNumber(p.weightKg);
...   const totalBodyWater = (weightKg === null) ? null : weightKg * 0.5;
...   p.totalBodyWater = totalBodyWater;

...   // Guard: sodiumValues array
...   const sv = Array.isArray(p.sodiumValues) ? p.sodiumValues : [];
...   if (sv.length === 0) {
...     // Still add totalCations as null if we can't compute
...     p.totalCations = null;
...     p.sodiumValues = sv;
...     return p;
...   }

...   // 2) totalCations = totalBodyWater * first sodiumValue
...   const firstNa = safeNumber(sv[0]?.sodiumValue);
...   p.totalCations = (totalBodyWater === null || firstNa === null) ? null : totalBodyWater * firstNa;
... }
```

- 2) For each patient, add a key-value pair for “totalCations”, which is equal to totalBodyWater x the value of the first sodiumValue within sodiumValues

```

..... // 3) For each sodiumValues object after the first:
..... for (let i = 0; i < sv.length; i++) {
.....   const entry = sv[i] || {};
.....   const entryMs = toMillis(entry.dateTimeStamp);

.....   if (i === 0) {
.....     entry.newTotalBodyWater = null;
.....     entry.predictedSodium = null;
.....     sv[i] = entry;
.....     continue;
.....   }

```

```

..... // 3a) newTotalBodyWater = totalBodyWater + any waterGivenCC up to this sodium timestamp
..... const waterSumCC = Number.isFinite(entryMs)
.....   ? sumWaterUpTo(p.waterGiven, entryMs)
.....   : 0;

..... // Convert CC to L (1000 cc = 1 L)
..... const waterSumL = waterSumCC / 1000;

..... const newTBW = (totalBodyWater === null) ? null : totalBodyWater + waterSumL;
..... entry.newTotalBodyWater = newTBW;

```

```

..... // 3b) predictedSodium = totalCations / newTotalBodyWater
..... const totalCations = safeNumber(p.totalCations);
..... entry.predictedSodium =
.....   (totalCations === null || newTBW === null || newTBW === 0)
.....   ? null
.....   : totalCations / newTBW;

..... sv[i] = entry;
..... }

```

3) For each patient's sodiumValues objects (except for the first in array, which will have a value of "null" for each new variable), do the following:

- Add a key-value pair to the sodiumValues object for "newTotalBodyWater", which is equal to the original value of patient's total body water plus any additional "waterGivenCC" up until the point in time for the dateTimeStamp value of that sodiumValue

...

```

..... // 3) For each sodiumValues object after the first:
..... for (let i = 0; i < sv.length; i++) {
.....   const entry = sv[i] || {};
.....   const entryMs = toMillis(entry.dateTimeStamp);

.....   if (i === 0) {
.....     entry.newTotalBodyWater = null;
.....     entry.predictedSodium = null;
.....     sv[i] = entry;
.....     continue;
.....   }

```

```

..... // 3a) newTotalBodyWater = totalBodyWater + any waterGivenCC up to this sodium timestamp
..... const waterSumCC = Number.isFinite(entryMs)
.....   ? sumWaterUpTo(p.waterGiven, entryMs)
.....   : 0;

..... // Convert CC to L (1000 cc = 1 L)
..... const waterSumL = waterSumCC / 1000;

..... const newTBW = (totalBodyWater === null) ? null : totalBodyWater + waterSumL;
..... entry.newTotalBodyWater = newTBW;

```

```

..... // 3b) predictedSodium = totalCations / newTotalBodyWater
..... const totalCations = safeNumber(p.totalCations);
..... entry.predictedSodium =
.....   (totalCations === null || newTBW === null || newTBW === 0)
.....   ? null
.....   : totalCations / newTBW;

..... sv[i] = entry;
..... }

```

3) For each patient's sodiumValues objects (except for the first in array, which will have a value of "null" for each new variable), do the following:

- Add a key-value pair to the sodiumValues object for "newTotalBodyWater", which is equal to the original value of patient's total body water plus any additional "waterGivenCC" up until the point in time for the dateTimeStamp value of that sodiumValue

...

```

function sumWaterUpTo(waterGivenArray, sodiumTimestampMs) {
  if (!Array.isArray(waterGivenArray) || waterGivenArray.length === 0) return 0;

  let sum = 0;
  for (const w of waterGivenArray) {
    if (!w) continue;
    const wMs = toMillis(w.dateTimeStamp);
    const cc = safeNumber(w.waterGivenCC);
    if (!Number.isFinite(wMs) || cc === null) continue;
    if (wMs <= sodiumTimestampMs) sum += cc;
  }
  return sum;
}

```

```

// 3a) newTotalBodyWater = totalBodyWater + any waterGivenCC up to this sodium timestamp
const waterSumCC = Number.isFinite(entryMs)
  ? sumWaterUpTo(p.waterGiven, entryMs)
  : 0;

// Convert CC to L (1000 cc = 1 L)
const waterSumL = waterSumCC / 1000;

const newTBW = (totalBodyWater === null) ? null : totalBodyWater + waterSumL;
entry.newTotalBodyWater = newTBW;

```

```

// 3b) predictedSodium = totalCations / newTotalBodyWater
const totalCations = safeNumber(p.totalCations);
entry.predictedSodium =
  (totalCations === null || newTBW === null || newTBW === 0)
  ? null
  : totalCations / newTBW;

sv[i] = entry;
}

```

3) For each patient's sodiumValues objects (except for the first in array, which will have a value of "null" for each new variable), do the following:

- Add a key-value pair to the sodiumValues object for "newTotalBodyWater", which is equal to the original value of patient's total body water plus any additional "waterGivenCC" up until the point in time for the dateTimeStamp value of that sodiumValue

...

```

..... // 3) For each sodiumValues object after the first:
..... for (let i = 0; i < sv.length; i++) {
.....     const entry = sv[i] || {};
.....     const entryMs = toMillis(entry.dateTimeStamp);

.....     if (i === 0) {
.....         entry.newTotalBodyWater = null;
.....         entry.predictedSodium = null;
.....         sv[i] = entry;
.....         continue;
.....     }

.....     // 3a) newTotalBodyWater = totalBodyWater + any waterGivenCC up to this sodium timestamp
.....     const waterSumCC = Number.isFinite(entryMs)
.....         ? sumWaterUpTo(p.waterGiven, entryMs)
.....         : 0;

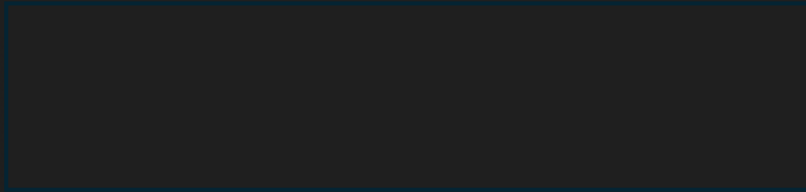
.....     // Convert CC to L (1000 cc = 1 L)
.....     const waterSumL = waterSumCC / 1000;

.....     const newTBW = (totalBodyWater === null) ? null : totalBodyWater + waterSumL;
.....     entry.newTotalBodyWater = newTBW;

.....     // 3b) predictedSodium = totalCations / newTotalBodyWater
.....     const totalCations = safeNumber(p.totalCations);
.....     entry.predictedSodium =
.....         (totalCations === null || newTBW === null || newTBW === 0)
.....         ? null
.....         : totalCations / newTBW;

.....     sv[i] = entry;
..... }

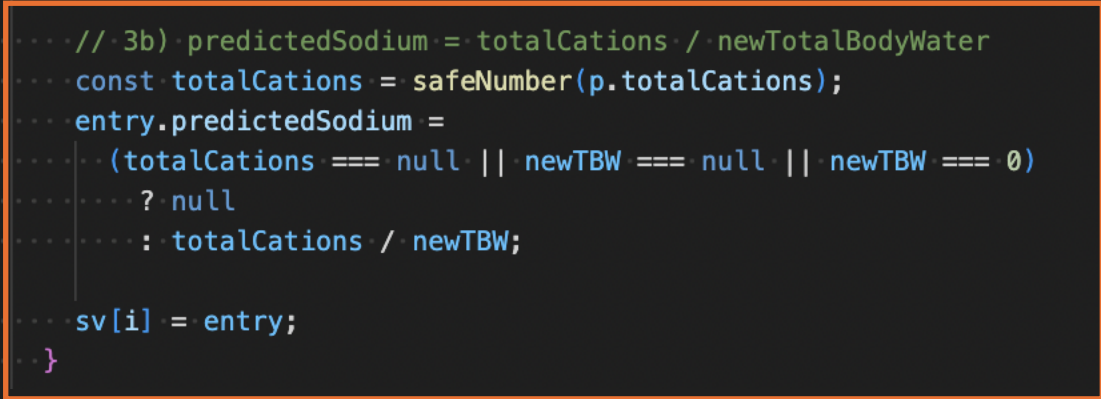
```



3) For each patient's sodiumValues objects (except for the first in array, which will have a value of "null" for each new variable), do the following:

...

- Add a key-value pair to the sodiumValues object for "predictedSodium", which is equal to "totalCations" for the patient divided by "newTotalBodyWater" for that given sodiumValues object, as already calculated in prior step



```
·//·Main·transform·for·one·patient  
·function·transformPatient(patient)·{
```

```
····
```

```
·····p.sodiumValues = sv;  
·····return p;  
···}
```

```
·function·transformInput(data)·{  
·····//·Accept:·single·patient·object·OR·array·of·patient·objects  
·····if·(Array.isArray(data))·{  
······return·data.map(transformPatient);  
·····}  
·····if·(data·&&·typeof·data·===·"object")·{  
······return·transformPatient(data);  
·····}  
·····throw·new·Error("Input·must·be·a·patient·object·or·an·array·of·patient·objects.");  
···}
```

**Final Step:** Loop through array of all patients and run the function *transformPatient* ()

## Prompt 4:

I need an HTML/JavaScript code that will take a JSON, which the user should be able to copy and paste into an input in the web app, and then click a button such that:

- 1) the JSON is converted into a CSV file
- 2) the CSV file gets downloaded. The JSON is an array of objects, such that each object is a patient.

All data should be included in the CSV output. The structure of the JSON for one patient object is shown below for reference. Please provide entire HTML/JavaScript code so that it can simply be copied and pasted.

```
patient = {
  "name": String,
  "age": Number,
  "male": Boolean,
  "weightKg": Number,
  "sodiumValues": [
    {
      "sodiumValue": Number,
      "dateTimeStamp": String,
      "newTotalBodyWater": Number,
      "predictedSodium": Number
    }
  ],
  "waterGiven": [
    {
      "waterGivenCC": Number,
      "dateTimeStamp": String
    }
  ],
  "totalBodyWater": Number,
  "totalCations": Number
}
```

## Patients JSON → CSV (Download)

Paste an array of patient objects into the box, then click **Convert & Download CSV**.

Output is a **"long format" CSV**: each patient-level field repeats, and each row represents either a `sodiumValues` entry or a `waterGiven` entry (so *all data* is preserved).

Paste JSON array here, e.g.:

```
[
  {
    "name": "Patient A",
    "age": 55,
    "male": true,
    "weightKg": 72,
    "sodiumValues": [
      {"sodiumValue": 150, "dateTimeStamp": "2026-02-02T06:00", "newTotalBodyWater": 42.1, "predictedSodium": 148}
    ],
    "waterGiven": [
      {"waterGivenCC": 250, "dateTimeStamp": "2026-02-02T07:30"}
    ],
    "totalBodyWater": 43,
    "totalCations": 6500
  }
]
```

**Convert & Download CSV**

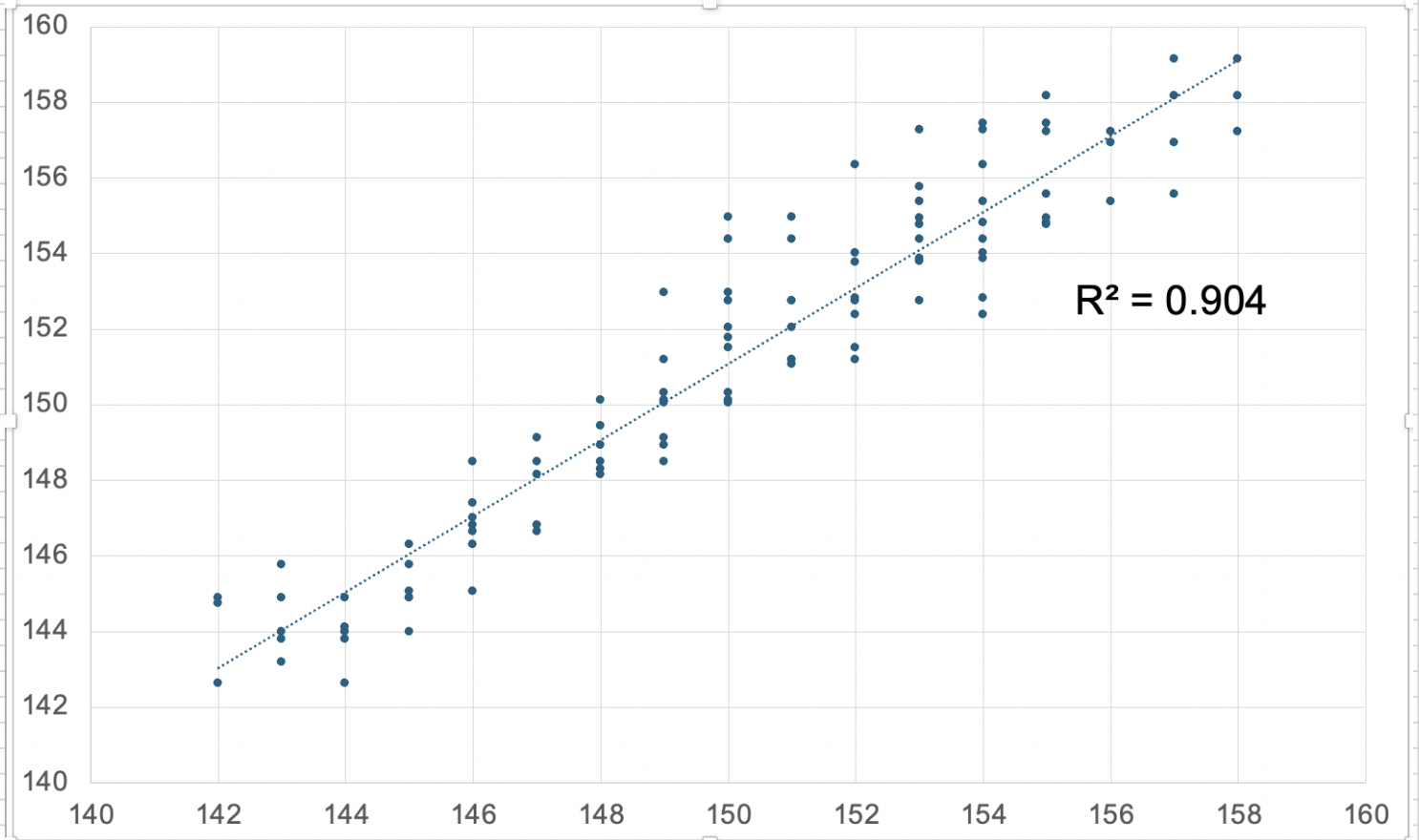
**Load Example JSON**

Tip: If you want "wide format" (one row per patient), tell me and I'll adjust.

Status: waiting for input...

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	name	age	male	weightKg	totalBodyWater	totalCations	recordType	recordDateTimeStamp	sodiumValue	sodiumDateTimeStamp	newTotalBodyWater	predictedSodium	waterGivenCC	waterDateTimeStamp
2	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T06:30	156	2026-02-01T06:30				
3	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T09:15	155	2026-02-01T09:15	36.25	154.9241379		
4	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T12:00	153	2026-02-01T12:00	36.25	154.9241379		
5	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T15:10	154	2026-02-01T15:10	36.75	152.8163265		
6	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T18:05	152	2026-02-01T18:05	36.75	152.8163265		
7	Patient 1	34	TRUE	72	36	5616	sodium	2026-02-01T21:20	150	2026-02-01T21:20	37	151.7837838		
8	Patient 1	34	TRUE	72	36	5616	water	2026-02-01T07:05					250	2026-02-01T07:05
9	Patient 1	34	TRUE	72	36	5616	water	2026-02-01T13:40					500	2026-02-01T13:40
10	Patient 1	34	TRUE	72	36	5616	water	2026-02-01T19:30					250	2026-02-01T19:30
11	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T05:50	158	2026-02-02T05:50				
12	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T08:30	157	2026-02-02T08:30	32.5	155.5692308		
13	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T11:10	155	2026-02-02T11:10	32.5	155.5692308		
14	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T14:00	154	2026-02-02T14:00	32.75	154.3816794		
15	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T17:25	153	2026-02-02T17:25	32.75	154.3816794		
16	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T20:10	151	2026-02-02T20:10	32.75	154.3816794		
17	Patient 2	58	FALSE	64	32	5056	sodium	2026-02-02T22:40	150	2026-02-02T22:40	32.75	154.3816794		
18	Patient 2	58	FALSE	64	32	5056	water	2026-02-02T06:20					500	2026-02-02T06:20
19	Patient 2	58	FALSE	64	32	5056	water	2026-02-02T12:45					250	2026-02-02T12:45
20	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T06:10	152	2026-02-03T06:10				
21	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T08:55	151	2026-02-03T08:55	40.25	151.0559006		
22	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T11:30	149	2026-02-03T11:30	40.5	150.1234568		
23	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T14:05	150	2026-02-03T14:05	40.5	150.1234568		
24	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T16:50	148	2026-02-03T16:50	41	148.2926829		
25	Patient 3	71	TRUE	80	40	6080	sodium	2026-02-03T19:25	146	2026-02-03T19:25	41.25	147.3939394		
26	Patient 3	71	TRUE	80	40	6080	water	2026-02-03T07:00					250	2026-02-03T07:00
27	Patient 3	71	TRUE	80	40	6080	water	2026-02-03T10:15					250	2026-02-03T10:15
28	Patient 3	71	TRUE	80	40	6080	water	2026-02-03T15:40					500	2026-02-03T15:40
29	Patient 3	71	TRUE	80	40	6080	water	2026-02-03T18:10					250	2026-02-03T18:10
30	Patient 4	46	FALSE	59	29.5	4425	sodium	2026-02-04T07:00	150	2026-02-04T07:00				
31	Patient 4	46	FALSE	59	29.5	4425	sodium	2026-02-04T09:20	149	2026-02-04T09:20	29.8	148.4899329		
32	Patient 4	46	FALSE	59	29.5	4425	sodium	2026-02-04T12:10	148	2026-02-04T12:10	29.8	148.4899329		
33	Patient 4	46	FALSE	59	29.5	4425	sodium	2026-02-04T14:55	147	2026-02-04T14:55	29.8	148.4899329		
34	Patient 4	46	FALSE	59	29.5	4425	sodium	2026-02-04T18:00	146	2026-02-04T18:00	29.8	148.4899329		
35	Patient 4	46	FALSE	59	29.5	4425	water	2026-02-04T08:05					300	2026-02-04T08:05
36	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T05:40	160	2026-02-05T05:40				
37	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T08:10	158	2026-02-05T08:10	46.25	159.1351351		
38	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T10:45	157	2026-02-05T10:45	46.25	159.1351351		
39	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T13:30	155	2026-02-05T13:30	46.75	157.4331551		
40	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T16:15	154	2026-02-05T16:15	46.75	157.4331551		
41	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T19:05	153	2026-02-05T19:05	47.25	155.7671958		
42	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T22:10	151	2026-02-05T22:10	47.5	154.9473684		
43	Patient 5	63	TRUE	92	46	7360	sodium	2026-02-05T23:55	150	2026-02-05T23:55	47.5	154.9473684		
44	Patient 5	63	TRUE	92	46	7360	water	2026-02-05T06:25					250	2026-02-05T06:25
45	Patient 5	63	TRUE	92	46	7360	water	2026-02-05T11:20					500	2026-02-05T11:20

H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
recordDateTimeStamp	sodiumValue	sodiumDateTimeStamp	newTotalBodyWater	predictedSodium	waterGivenCC	waterDateTimeStamp											
2026-02-16T13:55	144	2026-02-16T13:55	30.5	142.6229508													
2026-02-16T16:25	142	2026-02-16T16:25	30.5	142.6229508													
2026-02-07T18:40	143	2026-02-07T18:40	38.5	143.1818182													
2026-02-16T08:40	144	2026-02-16T08:40	30.25	143.8016529													
2026-02-16T11:10	143	2026-02-16T11:10	30.25	143.8016529													
2026-02-10T13:15	144	2026-02-10T13:15	27.75	144													
2026-02-10T16:00	145	2026-02-10T16:00	27.75	144													
2026-02-10T18:30	143	2026-02-10T18:30	27.75	144													
2026-02-07T16:00	144	2026-02-07T16:00	38.25	144.1176471													
2026-02-18T23:10	142	2026-02-18T23:10	35.75	144.7552448													
2026-02-12T08:55	145	2026-02-12T08:55	33.25	144.9022556													
2026-02-12T11:15	144	2026-02-12T11:15	33.25	144.9022556													
2026-02-12T13:50	145	2026-02-12T13:50	33.25	144.9022556													
2026-02-12T16:30	143	2026-02-12T16:30	33.25	144.9022556													
2026-02-12T19:05	142	2026-02-12T19:05	33.25	144.9022556													
2026-02-07T11:00	145	2026-02-07T11:00	38	145.0657895													
2026-02-07T13:30	146	2026-02-07T13:30	38	145.0657895													
2026-02-18T18:50	145	2026-02-18T18:50	35.5	145.7746479													
2026-02-18T21:30	143	2026-02-18T21:30	35.5	145.7746479													
2026-02-15T18:20	146	2026-02-15T18:20	40.25	146.310559													
2026-02-15T21:00	145	2026-02-15T21:00	40.25	146.310559													
2026-02-10T08:00	147	2026-02-10T08:00	27.25	146.6422018													
2026-02-10T10:30	146	2026-02-10T10:30	27.25	146.6422018													
2026-02-18T13:55	147	2026-02-18T13:55	35.25	146.8085106													
2026-02-18T16:20	146	2026-02-18T16:20	35.25	146.8085106													
2026-02-07T08:30	146	2026-02-07T08:30	37.5	147													
2026-02-03T19:25	146	2026-02-03T19:25	41.25	147.3939394													
2026-02-15T13:05	147	2026-02-15T13:05	39.75	148.1509434													
2026-02-15T15:40	148	2026-02-15T15:40	39.75	148.1509434													
2026-02-03T16:50	148	2026-02-03T16:50	41	148.2926829													
2026-02-04T09:20	149	2026-02-04T09:20	29.8	148.4899329													
2026-02-04T12:10	148	2026-02-04T12:10	29.8	148.4899329													
2026-02-04T14:55	147	2026-02-04T14:55	29.8	148.4899329													
2026-02-04T18:00	146	2026-02-04T18:00	29.8	148.4899329													
2026-02-18T09:00	149	2026-02-18T09:00	34.75	148.9208633													
2026-02-18T11:25	148	2026-02-18T11:25	34.75	148.9208633													
2026-02-08T18:15	149	2026-02-08T18:15	31.5	149.1111111													
2026-02-08T20:35	147	2026-02-08T20:35	31.5	149.1111111													
2026-02-13T18:45	148	2026-02-13T18:45	43	149.4418605													
2026-02-15T08:00	150	2026-02-15T08:00	39.25	150.0382166													
2026-02-15T10:35	149	2026-02-15T10:35	39.25	150.0382166													
2026-02-03T11:30	149	2026-02-03T11:30	40.5	150.1234568													
2026-02-03T14:05	150	2026-02-03T14:05	40.5	150.1234568													
2026-02-19T17:45	148	2026-02-19T17:45	39.75	150.1257862													
2026-02-13T13:20	149	2026-02-13T13:20	42.75	150.3157895													
2026-02-13T16:00	150	2026-02-13T16:00	42.75	150.3157895													
2026-02-03T08:55	151	2026-02-03T08:55	40.25	151.0559006													
2026-02-13T08:15	152	2026-02-13T08:15	42.5	151.2													



# Outline

1. Introducing JavaScript (and HTML)
2. Building blocks of code
3. *Demo 1: a simple browser-based web app*
4. Privacy / policy considerations
5. JSON and data structures
6. *Demo 2: big data analysis for research*
7. **Debugging, Limitations, Big Takeaways**

# ADDITIONAL POINTS

- Ask your AI tool about the code it wrote for you
- Use AI to help debug code

# LIMITATIONS

- **Privacy concerns – follow institutional guidelines**
- **Be mindful of risk of hallucinations and other errors**
- **Ensure all code is understood**
- **Test rigorously with known inputs and outputs**
- **Avoid use for statistics and high-risk cases**

# CONCLUSIONS

- **A starting point: AI can make coding accessible**
- **Understanding data structures is a key component of AI literacy**
- **Use AI responsibly: test everything, ask questions, use caution**

# **Part 1: Coding Concepts & JavaScript**

**Robert Hayden, MD**

# **Part 2: Python & More Data Demos**

**Bitá Behrouzi, MD**