XPath Rules and Best Practices

# 📜 Basic XPath Rules

* ✔️ Use // to search anywhere  
   ➤ Selects nodes anywhere in the document  
   🧪 Example: //input
* ✔️ Use / for direct child  
   ➤ Selects the immediate child  
   🧪 Example: /html/body/div
* ✔️ Use @ for attributes  
   ➤ Targets an attribute of an element  
   🧪 Example: //input[@id='username']
* ✔️ Use [] for conditions  
   ➤ Filters elements based on attributes or index  
   🧪 Example: //input[@type='text']
* ✔️ Use text() to match visible text  
   ➤ Matches elements by the inner text  
   🧪 Example: //button[text()='Login']
* ✔️ Use contains() for partial matches  
   ➤ Useful when values change dynamically  
   🧪 Example: //input[contains(@class, 'form')]
* ✔️ Use and / or for multiple conditions  
   ➤ Combine multiple attribute checks  
   🧪 Example: //input[@type='text' and @name='email']
* ✔️ Use \* to match any tag  
   ➤ Wildcard to select any element  
   🧪 Example: //\*[@id='main']

# 🧭 Advanced XPath Rules

* ✔️ Use position() for indexing  
   ➤ Gets element by its position  
   🧪 Example: (//input)[2]
* ✔️ Use axes like parent::, following-sibling::  
   ➤ Navigate complex DOMs  
   🧪 Example: //label[text()='Email']/following-sibling::input
* ✔️ Avoid absolute XPath  
   ➤ Full path is brittle and breaks easily  
   🧪 Example: /html/body/div[2]/form/input[1]
* ✔️ Prefer relative XPath  
   ➤ Starts from relevant nodes, more stable  
   🧪 Example: //form[@id='loginForm']/input[1]

# 🔥 Common XPath Functions

* ✔️ text() – Selects the element's text
* ✔️ contains() – Partial match for text or attribute
* ✔️ starts-with() – Matches the beginning of a value
* ✔️ last() – Selects the last node in a set
* ✔️ normalize-space() – Trims extra white spaces

# ✅ Best Practices

* 🔹 Prefer unique attributes (id, name) when possible.
* 🔹 Use contains() for dynamic classes or IDs.
* 🔹 Avoid relying on index-based XPath unless structure is stable.
* 🔹 Always verify XPath manually using browser DevTools or tools like SelectorsHub.