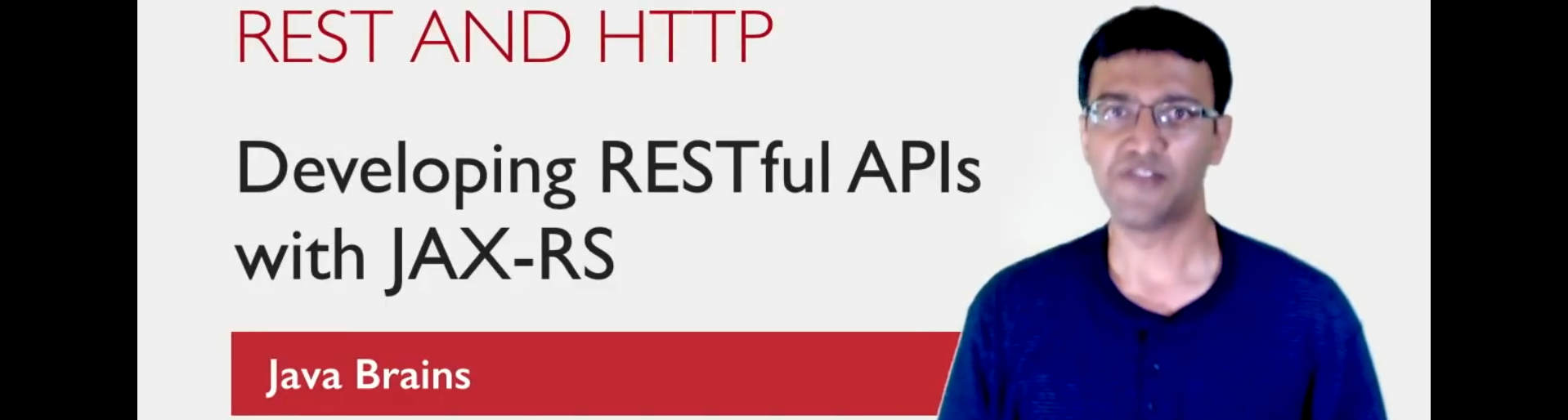
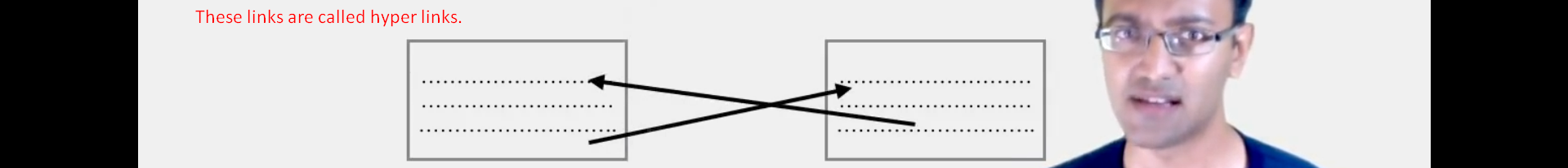
1. Remember:
   1. Resource = Entity
2. The concept of REST is very related to HTTP.  
   REST is inspired a lot by the concept of HTTP.  
   All the time you’re making request over web, you’re sending HTTP request and getting response.
3. No surprised Roy fielding as one of the principal authors of the HTTP specification used the concepts of HTTP behind REST.
4. To understand REST, have basic understanding of HTTP.
5. Let’s have a basic understanding of HTTP.
   1. 
   2. **Protocol**: A language or mechanism for communication.
   3. The stuff that you transfer in HTTP response contains Hypertext.
   4. **Hypertext**: **A structured form of text** having a property (it contains logical links to other pages called hyperlinks)  
      **A structure form of text:** Complete web page.  
      
   5. **HTML**: Common and popular way to write Hypertext.
6. Let’s focus on those HTTP concepts which are needed in REST and REST is inspired by.
7. **Resource Locators**: Resource-Based Addresses vs Action-Based Address.
   1. Since APIs are addresses and API designers or web service developers need to decide what those addresses should be.
   2. The practice in REST is to have **Resource-Based Addresses**.
   3. Let’s take an example.
   4. In weather web service, to look up a zip code, the address could be 
   5. This address is not Resourced-Based but more **action-based**.  
      It tells you “there is something like weatherLookup.do which takes zipcode as parameter and does something”.
   6. Resource-Based addresses on the other hand indicate just resources and independent of server side implementation.
   7.   
       This address is not making server do anything rather just looking up something that already exists.  
      
8. **HTTP Methods**: Now as we have decided what addresses, now how to interact with it?
   1. HTTP Methods = Verbs.
   2. Used to interact with URLs.
   3. GET, POST, POST, DELETE etc.
   4. They work well with Resource-Based addresses.
   5. In SOAP, all requests are made using POST which is not the case with REST.
9. **Metadata**:
   1. We decided address and HTTP method. Now when they make the call, what response they get?
   2. HTTP also defines something called response headers such as status code which let the server send back some extra data which can be useful for the clients.
   3. **HTTP status codes**:  
      200: success  
      500: Service Error  
      404: Not Found.
   4. Why to send error codes?
      1. If client is human, in case of error/exception or some other problems on server, we send readable text in a nice format to the client along with links maybe to go to home page but in REST, client is a piece of code. Which is why sending a right status code is very important. We send one of these codes which are designed for specific purpose.  
         The client (piece of code) can read those status codes to figure out whether request is successful, failure. If failure why.
10. Format of request/response?
    1. No specification in REST what should be the format of data sent in request.
    2. It could be XML, JSON or some other formats.
    3. How does server and client figure out the format?
    4. **Answer**: Header 🡺 content-type
    5. There are some pre-defined standard values for content types.
    6. text/html, application/json
    7. Same API could return multiple content types depending on the type requested by a client.
    8. This process is known as **content negotiation**.
11. **Summary**:
    1. **Resource-Based URIs**:
       1. Every resource/entity on server should be identifiable by a single API.
       2. It should not indicate any action.
    2. **HTTP methods**:
       1. Choose right one depending on what API does.
    3. **HTTP Status Code**:
       1. Right status code.
    4. **Message Headers**:
       1. All requests and responses need to have right headers such as content-type
12. In next tutorial we will put these concepts into practice by implementing a RESTful API