**HTTP Headers:**

1. GET: returns data (should not mutate)
2. POST: server will decide the URL
3. PUT: requests that the enclosed entity be stored under the supplied Request-URI. If the Request-URI refers to an already existing resource, the enclosed entity SHOULD be considered as a modified version of the one residing on the origin server.
4. PATCH: partial modification on an entity
5. DELETE: deletes resource
6. HEAD: similar to GET, but no body
7. OPTIONS: describe communication options for the target resource

**DOCTYPE:** Requires for webpages, used to tell the user agents what version of the HTML specifications the document respects

* if the user agent recognizes it as a valid doctype, then **no quirks mode**
* if not, then **quirks mode** (layout emulates nonstandard behavior)  
  (+there’s a limited quirks mode, which is like almost standard)  
  (these things were made because of early days before W3C wen Netscape Navigator and IE used different syntaxes)

HTML5 standard: <!DOCTYPE html>

**Multilanguage support:**

* request is made from the client, the user agets usually adds Accept-Language header
* returned HTML should have: <html lang="en">  
  (the backend puts together the HTML usually with stuff like **i18n** placeholders)

What to watch out for:

* HTML contains lang attribute
* Reading direction
* Varying lengths of text
* Images with text on them
* Colors are preceived differently across cultures
* Date formats

**What are data- attributes?**  
Before JS frameworks, it was popular to store extra data in the DOM itself  
These days, these are mostly only encouraged for E2E testing, to mark elements as data-test attribute, to help stuff like Selenium and Cypress.IO find them

**CSS specificity**: browsers decide which CSS property values are the most relevant to an element and apply that (overriding lower level values if necessary)

1. Inline styles
2. ID selectors
3. Class selectors
4. Tags, pseudo element selectors

**Reset CSS:** strip all default browser styling (redeclare everything yourself)  
**Normalize CSS:** preserve useful default styles (+ correct bugs for common browser dependencies)

**Float:**

* Absolute: gets taken out of the flow of page (until first relative position)
  + Clearfix solution (usually clear can be used below absoluteltly positioned elements, because by default their parents has to clear, but with this, :after is used)

**Cookies:**

* Can be initiated by server or client
* Manually set expiration
* Can be accessed from any window

**Localstorage:**

* Client initiates only
* Only can be deleted
* Can be accessed from any window

**Sessionstorage:**

* Client initiates only
* Gets deleted when tab closes
* Can only be accessed from the tab