-Secure Coding
-how to avoid Security related vulnerabilities
-Coding Guidelines
-Checklist for Secure Coding Practices

is to design and develop software by avoiding the weakness
=> How much Security is required
=> When can we say software is secured

Understand the threats involved: potential threats
Prioritize

health care app : risk - stealing of personal data

mitigate the risk

Strong access control

Moving Data : transmission of data

Data is at REST : strong encryption

- specific security concerns for a product
- Possible benefit
- Possible ways of security compromise
- Common practices followed in similar domain
- Identify the actual security issue

Primary pointers

Reasons for Inadequate Security Implementation

- # Priority is given to functional release
- # Ignorance about
- # no clarity
- # complexity
- # Not enough live data/info
- # Late consideration in SDLC
- # Language risks
- # Top level Security Guidelines missing

How to code a functionality

How to code securely

Most import design principal for Software security : Implement security by design / default

Secure Coding Guidelines

Security Coding standard identified at very beg.

Enforce to Adhere to these std.

Language Selection

I depth knowledge to implement security

Best Practices are language dependent, platform dependent, implementation dependent

Eg :

C/C++ strcpy()

Java : resources : file , db connection

Tools : optimize the security

IDE : alert, warn, debug tools

Static & Dynamic Analyzers : optimized to identify a specific type of error

: large number of false positive

Recommended to use multiple analyzer + manual testing

Secure Coding Guideline(source code) for tester to write test case for Conformance of Compliance

Code certified as Conforming or Non Confroming

Eg:

Files need to be closed if not in use

boundary checks

storage durations

BDD : Behavior Driven Development

Test : Common Programming mistakes

Checklist established for Secure Code Practices

- Input Validation
- Output Encoding
- Authentication & Password Management
- Session Management
-Access Control
-Cryptographic Practices
-Error-Handling
-Log
-Data Protection
-Communication Security
-System Configuration
-Database Security
-File Management
-Memory Management
- General Coding Practices

Input Validation :

classify them as trusted-untrusted

Centralized Input Validation Routine

Input Rejection

Redirects check

Character : <>" ' // \ & (....

../../ (Path)

Output Encoding

identify the destination

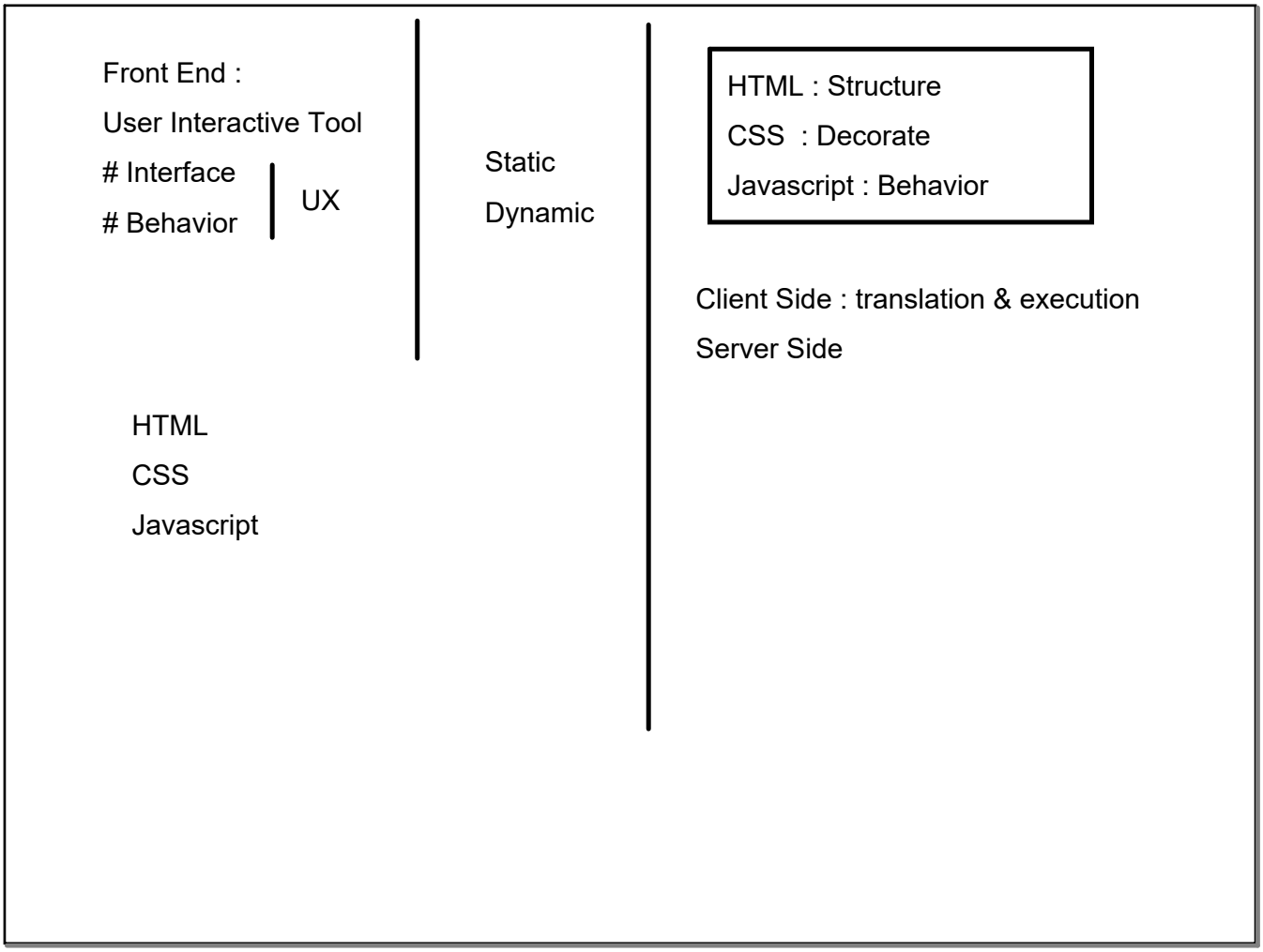
Authentication & PAssword Management:

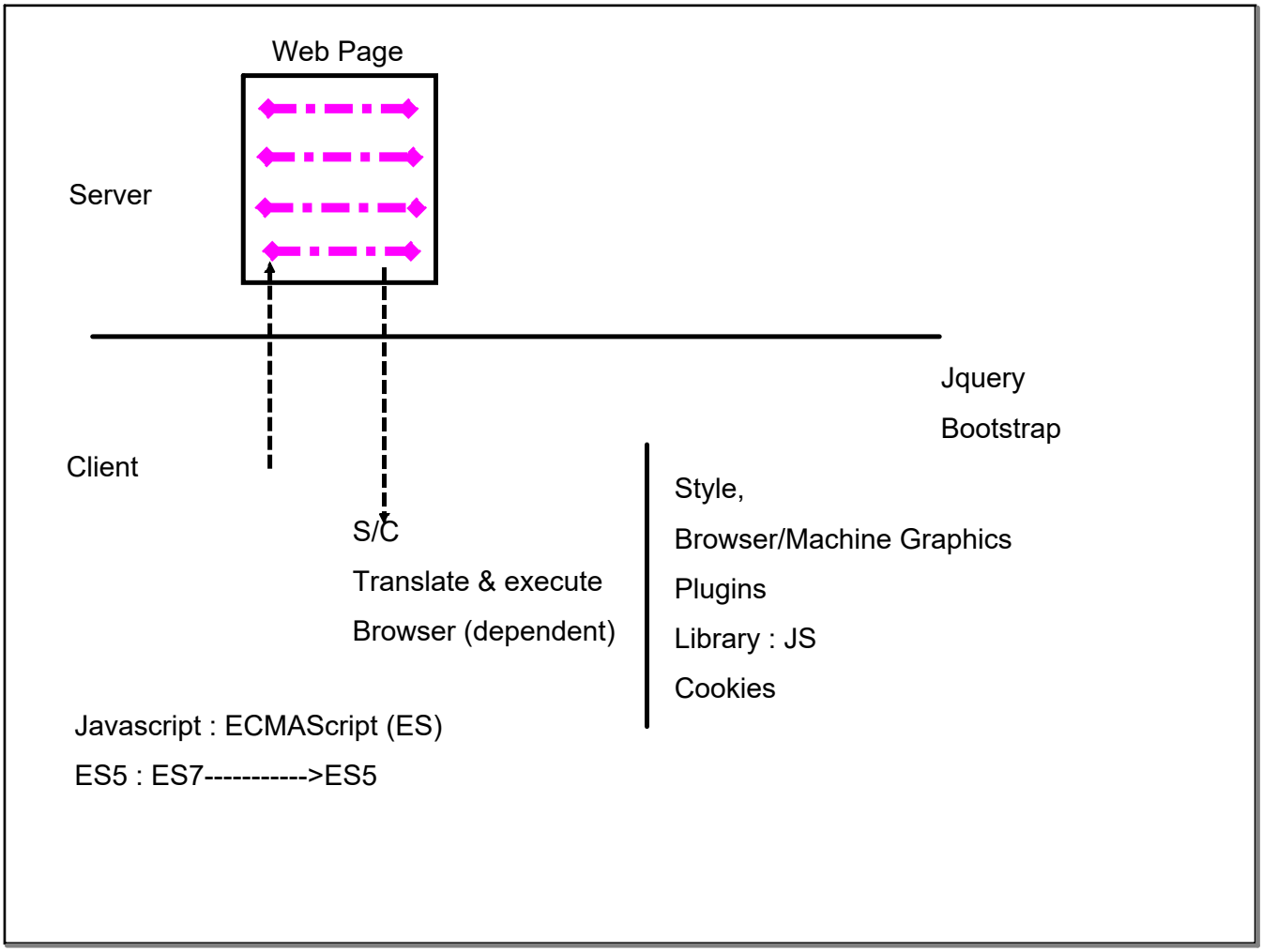
Session

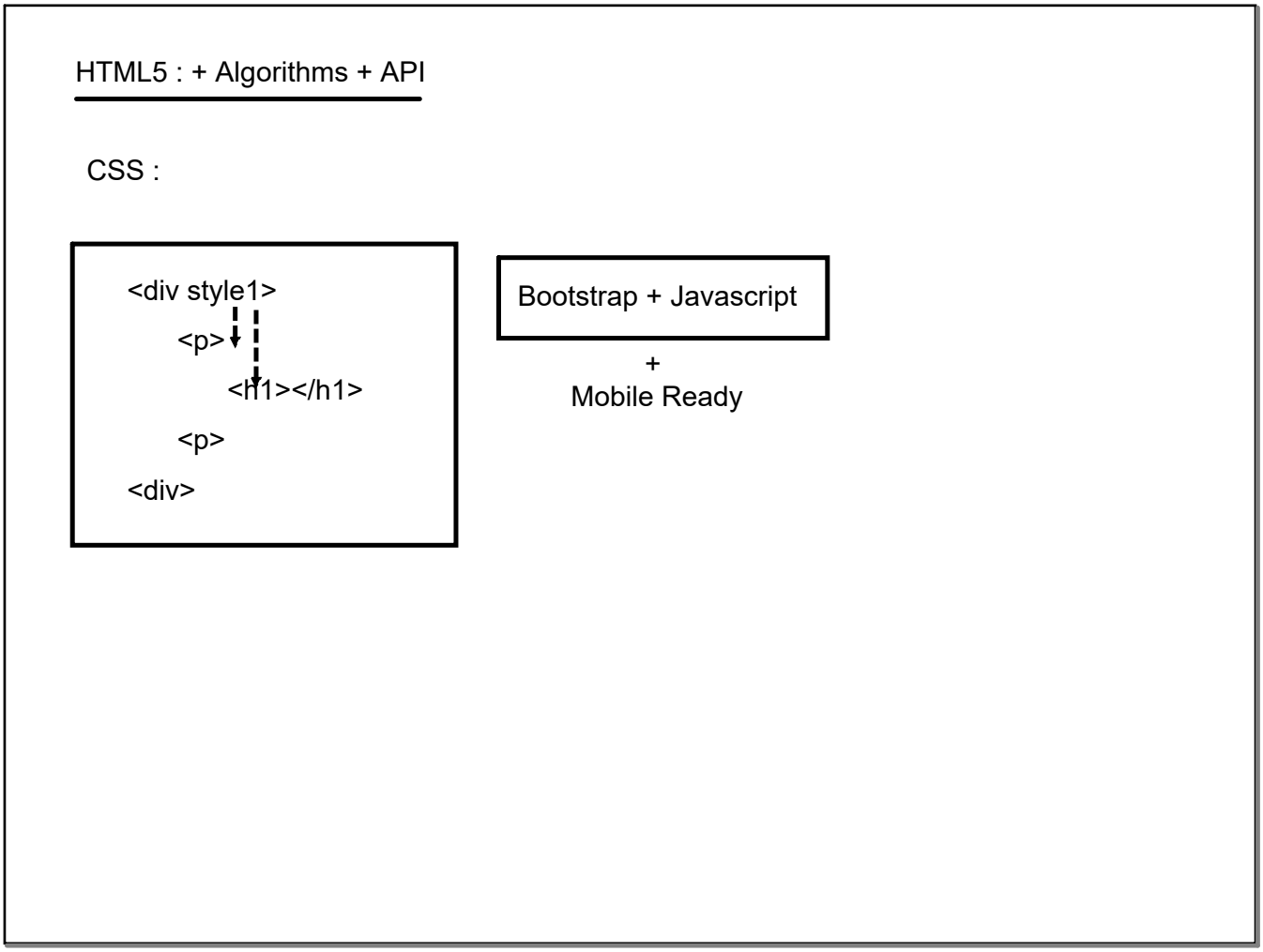
Session - Time

OWASP

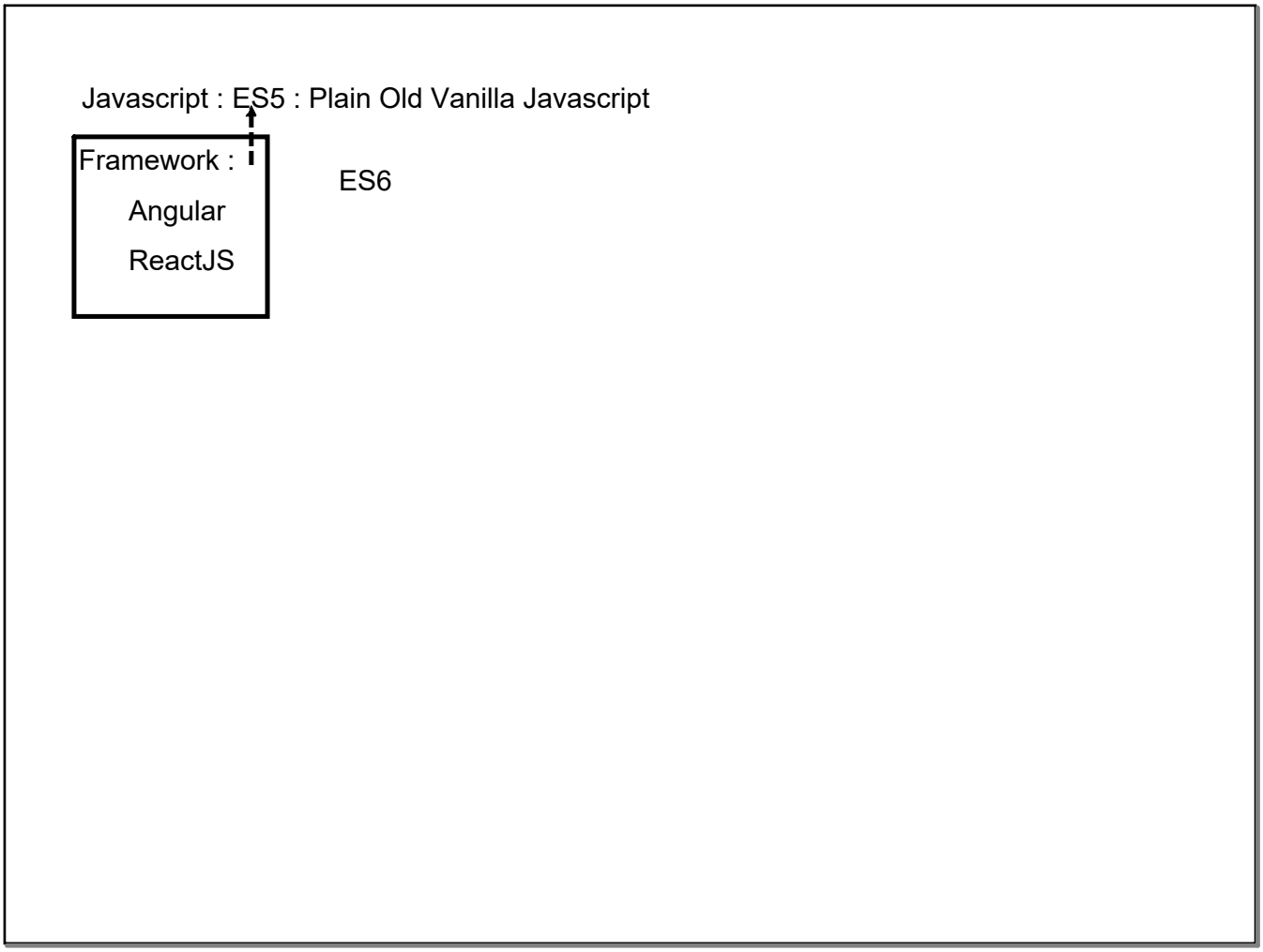
Open Web Application Security Project


Front End :
User Interactive Tool
Interface
Behavior
UX
Static
Dynamic
HTML : Structure
CSS : Decorate
Javascript : Behavior
Client Side : translation & execution
Server Side
HTML
CSS
Javascript


Web Page
Server
Client
S/C
Translate & execute
Browser (dependent)
Jquery
Bootstrap
Style,
Browser/Machine Graphics
Plugins
Library : JS
Cookies
Javascript : ECMAScript (ES)
ES5 : ES7---------->ES5


HTML5 : + Algorithms + API
CSS :
<div style1>
<p>
<h1></h1>
<p>

Bootstrap + Javascript
+
Mobile Ready


Javascript : ES5 : Plain Old Vanilla Javascript
Framework :
Angular
ReactJS
ES6