Web development - Intermediate

Agenda

Day 1(How the web works):

- How the web works
- Client-server architecture
- Evolution of web(WWW vs Internet)
- Set up Developer Environment

Day 2(Bootstrap, JQuery and DOM manipulation):

- Bootstrap
- DOM
- DOM selectors and events
- jQuery

Day 3(HTTP/JSON/AJAX +Async JS):

- HTTP/HTTPS
- JSON
- AJAX
- Asynchronous JavaScript

Agenda

Day 4-5(Frameworks, React):

- Introduction about frameworks and How the frameworks work under the hood?
- Introduction to React(state, props, component)

Day 6(APIs and microservices):

- How APIs work?
- Evolution of APIs
- Micro services and Web services

Day 7(Backend):

- Basics
- Introduction to NodeJs and ExpressJs

Day 4

- Introduction about frameworks and How the frameworks work under the hood?
- Frameworks vs libraries
- Introduction to React(state, props, component)

What is a framework?

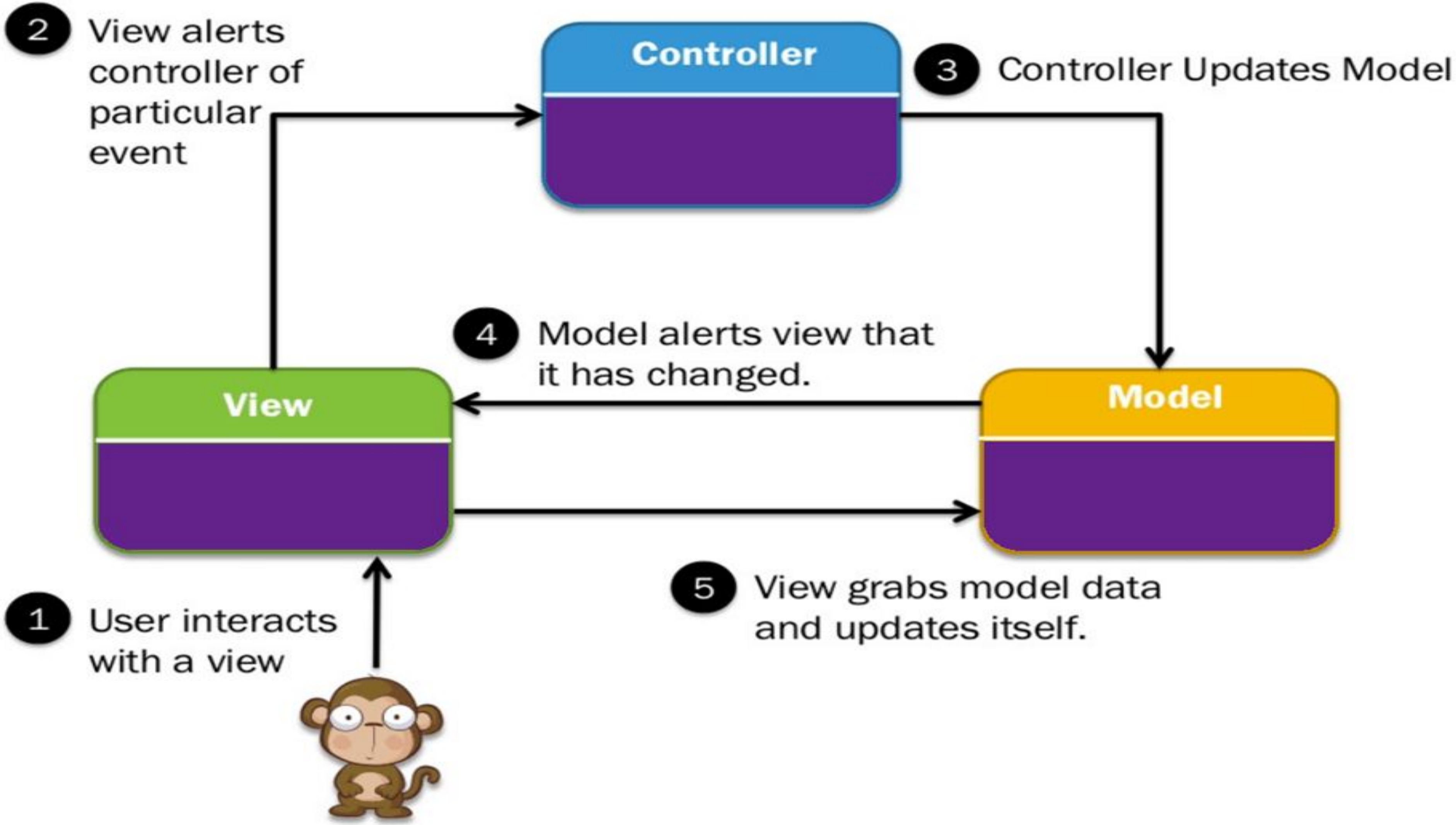
- abstraction in which software providing generic functionality
- providing application-specific software
- standard way to build and deploy applications and is a universal, reusable software environment
- support programs, compilers, code libraries, APIs that bring together all the different components to enable development of a project or system

Software Framework

Sets of libraries or classes			
Built-in generic functionalities, Deals with standard low level details	Reusable software enviroment	Working template application	Can be modified by writing additional code

LIBRARY VERSUS FRAMEWORK

Library	Framework
Library is a set of reusable functions used by computer programs.	Framework is a piece of code that dictates the architecture of your project and aids in programs.
You are in full control when you call a method from a library and the control is then returned.	The code never calls into a framework, instead the framework calls you.
It's incorporated seamlessly into existing projects to add functionality that you can access using an API.	It cannot be seamlessly incorporated into an existing project. Instead it can be used when anew project is started.
They are important in program linking and binding process.	They provide a standard way to build and deploy applications
Example: jQuery is a JavaScript library that simplifies DOM manipulation.	Example: AngularJS is a JavaScript-based framework for dynamic web applications.


2 View alerts controller of particular event
Controller
3 Controller Updates Model
4 Model alerts view that it has changed.
View
Model
5 View grabs model data and updates itself.
1 User interacts with a view

Resources

MVC: https://bit.ly/3sWwywg

React - library or framework? https://bit.ly/3sXE7mD

React

- open-source, front end, JavaScript library
- building user interfaces or UI components
- concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality

Resources:

https://reactjs.org/

https://www.freecodecamp.org/news/how-to-build-a-react-project-with-create-react-app-in-10-steps/

https://www.freecodecamp.org/news/react-examples-reactjs/

https://paulallies.medium.com/react-create-app-without-react-create-app-7c8341282645

Class components vs functional components

Resources: https://bit.ly/3e4nOjl

To do list React tutorial with Hooks:
https://www.educative.io/blog/react-hooks-tutorial-todo-list

Hooks

- `useState`: returns a stateful value
- `useEffect`: perform side effects from function components
- `useContext`: accepts a context objects and returns current context value
- `useCallback`: pass an inline callback and an array of dependencies

Advantages:

- Isolating stateful logic, making it easier to test
- Sharing stateful logic without render props or higher-order components
- Separating your app’s concerns based on logic
- Avoiding ES6 classes

State and props

- state allows components to create and manage their own data. So unlike props, components cannot pass data with state, but they can create and manage it internally
- Props is short for properties and they are used to pass data between React components. React’s data flow between components is uni-directional (from parent to child only).

Resources: https://bit.ly/3nylmVF

Build tools

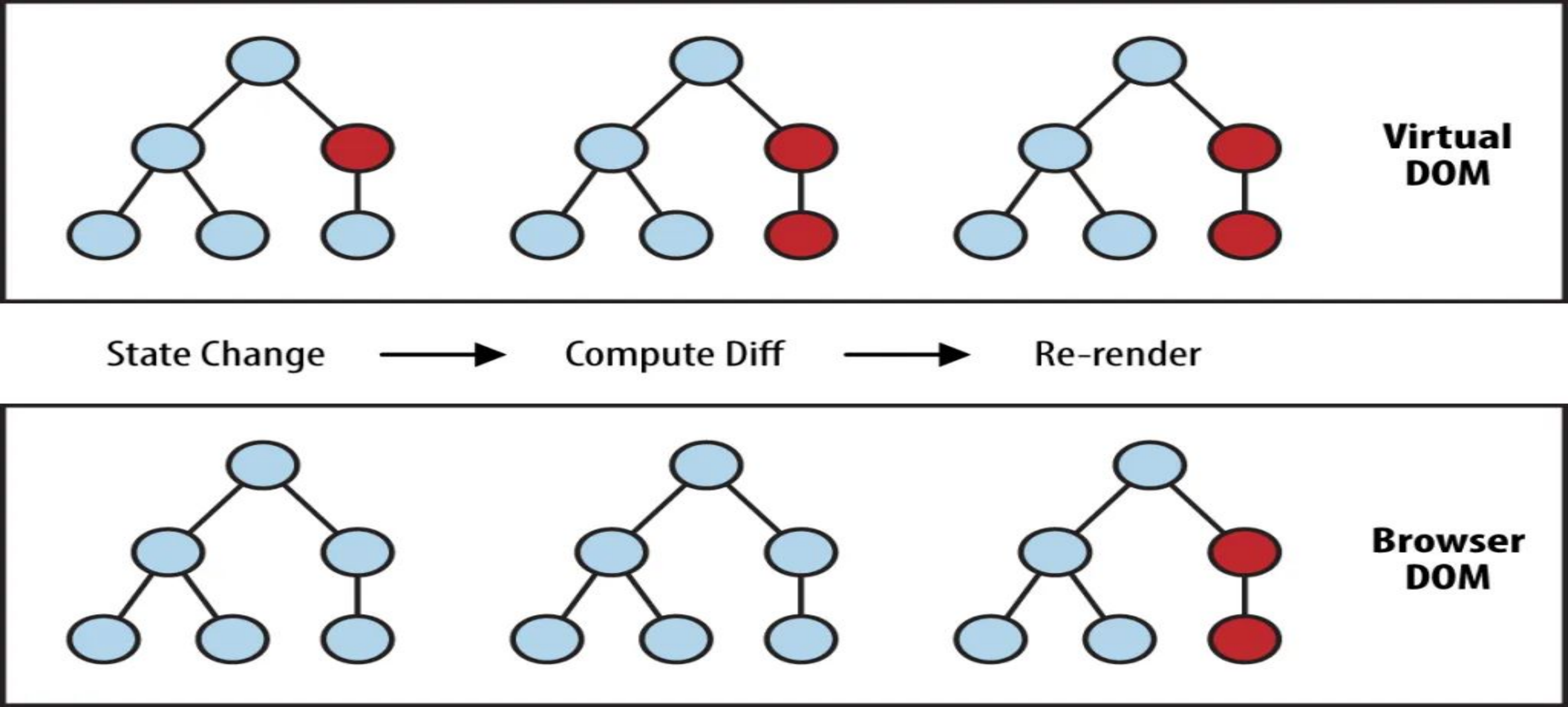
- Used to transpile or convert to vanilla HTML, CSS, Js that can be understood by the browser
- Takes care of compiling, linking, type checks(TypeScript), minification, bundling, hot reload.
- Packages the code into reusable, executable form.
- Bundler: tool to put your code and all your code together in one JS file. E.g. Webpack
- Task runner: compiles SCSS -> CSS, TypeScript -> Javascript

Webpack

- Module bundler: Modules with dependencies gets converted to static assets representing those modules,
- Webpack-cli provides commands
- Configurations in webpack.config.js

Official documentation: https://webpack.js.org/

Virtual DOM


Virtual DOM
State Change
Compute Diff
Re-render
Browser DOM

Resources for understanding Virtual DOM

https://programmingwithmosh.com/react/react-virtual-dom-explained/

Thank you!