1. What is Emmet?

Set of plugins for text editors that allow for high speed coding and editing in HTML,XML etc.

Syntax:

Elements – you can use elements names like div or p to generate html tags. Emmet doesn’t have a predefined set of available tag names, you can write any word and transform it into a tag div , foo etc.

Child: >

div>ul>li –

<div>

    <ul>

        <li></li>

    </ul>

   </div>

Sibling – + operator

Div+p+bq

<div></div>

   <p></p>

   <blockquote></blockquote>

Climb up- ^

With > operator you’re descending down the generated tree and positions of all sibling elements will be resolved against the most deepest element:

With ^ operator, you can climb one level up the tree and change context where following elements should appear:

div+div>p>span+em^bq

results into->

<div></div>

<div>

<p><span></span><em></em></p>

<blockquote></blockquote>

</div>

Multiplication

With \* operator you can define how many times element should be outputted.

ul>li\*5

...outputs to

<ul>

<li></li>

<li></li>

<li></li>

<li></li>

<li></li>

</ul>

Grouping – ()

Parenthesises are used by Emmets’ power users for grouping subtrees in complex abbreviations:

div>(header>ul>li\*2>a)+footer>p

...expands to

<div>

<header>

<ul>

<li><a href=""></a></li>

<li><a href=""></a></li>

</ul>

</header>

<footer>

<p></p>

</footer>

</div>

Attribute operator

div#header+div.page+div#footer.class1.class2.class3

...will output

<div id="header"></div>

<div class="page"></div>

<div id="footer" class="class1 class2 class3"></div>

**Custom attributes**

You can use [attr] notation (as in CSS) to add custom attributes to your element:

td[title="Hello world!" colspan=3]

...outputs

<td title="Hello world!" colspan="3"></td>

### Item numbering: $

With multiplication \* operator you can repeat elements, but with $ you can number them. Place $ operator inside element’s name, attribute’s name or attribute’s value to output current number of repeated element:

ul>li.item$\*5

...outputs to

<ul>

<li class="item1"></li>

<li class="item2"></li>

<li class="item3"></li>

<li class="item4"></li>

<li class="item5"></li>

</ul>

## Text: {}

You can use curly braces to add text to element:

a{Click me}

...will produce

<a href="">Click me</a>

Don’t use space in b/w because space is stop symbol where emmet stops abbreviation parsing.

1. Diff b/w library and framework?

The technical difference b/w framework and library lies in a term inversion of control.

When you use a library, you oversee the flow of the application. You care choosing when and where to call the library. When you use a framework, the framework oversees the flow. It provides some places for you to plug in your code, but it calls the code you plugged in as needed.

1. What is CDN? Why do we use it?

A content delivery network is a geographically distributed group of servers that caches content close to end users. A CDN allows for the quick transfer of assets needed for loading internet content, including HTML pages, including HTML pages etc.

Denial of service attack.

1. Why is react known as react?

It was developed by Facebook, and the name “React” was chosen because it is meant to help developers build user interfaces that are fast and responsive, or “reactive.” The library was designed to “react” to changes in data. When data in a React application changes, the components that depend on that data are automatically updated, which allows for efficient and seamless updates to the user interface. The name “React” reflects this reactive nature of the library. The idea behind React is to build reusable components that can be rendered on the front-end, rather than writing a new piece of code every time you need to display something on the screen. This makes it easier and more efficient to build complex user interfaces.

1. What is cross-origin in the script tag?

Origin – web content’s origin is defined by the scheme (protocol), hostname (domain), and port of the URL used to access it. Two objects have the same origin only when the scheme, hostname, and port all match.

Some operations are restricted to same origin content, and this restriction can be lifted using CORS.

CORS – cross origin resource sharing is an HTTP-header based mechanism that allows a server to indicate any origins other than its own form which a browser should permit loading resources. CORS also relies on a mechanism by which browsers make a “preflight” request to the server hosting the across origin resources, in order to check that the server hosting the cross-origin resource, in order to check that the server will permit the actual request. In that preflight, the browser sends headers that indicate the HTTP method and headers that will be used in the actual request.

The cross-origin attribute, valid on the audio, image, link and script elements, provides support for CORS, defining how the element handles cross-origin requests thereby enabling the configuration of the CORS requests for the element’s fetched data. Depending on the element, the attribute can be a CORS settings attribute.

The integrity attribute allows a browser to check the fetched script to ensure that the code is never loaded if the source has been manipulated.

1. Diff b/w react and React DOM?

React DOM is the glue b/w React and the DOM. Often, you will only use it one it for one single thing mounting with ReactDOM.render(). Another useful feature of ReactDOM is ReactDOM.findDOMNode() which you can use to gain direct access to a DOM element. If your app is isomorphic, you would also use ReactDOM.readerToString() in your back-and code.

For everything else, there’s React. You use React to define and create your elements, for lifecycle hooks, etc. i.e. the guts of a React application.

The reason React and React DOM were split into two libraries was due to the arrival of React native. React contains functionality utilized only in web apps.

This paves the way to writing components that can be shared between the web version of React and React Native. We don’t expect all the code in an app to be shared, but we want to able to be able to share the components that do behave the same across platforms.

1. Diff b/w production and development build in ReactJS.

The development build is used as the name suggests – for development reasons. You have source maps, debugging and often hot reloading ability in those builds.

The production build, on the other hand, runs in production mode which means this is code running on your client’s machine. The production build runs uglify and builds your source file into one or multiple minimized files. It also extracts CSS and images and of course any other sources you are loading with Webpack. There is also no hot reloading included. Source Maps might be included as separate files depending on your webpack devtool settings.

Assignment 2

1. What is NPM?

NPM is a package manager for the JS programming lang maintained by NPM. NPM is the default package manager for the JavaScript runtime environment node.js. it consists of a command line client also called NPM and an online database of public and paid for private packages called NPM registry. The registry is accessed via client and the available packages can be browsed and searched via the NPM website. The package manager and the registry are managed by NPM inc.

Package manager or package management system is a collection of software tools that automates the process of installing, upgrading configuring and removing computer programs for a computer in a consistent manner. A package manager deals with packages distribution of software and data in archive file. Packages contains metadata, such as software name and version number etc.

1. What is Parcel/webpack? why do we need it?

At its core webpack is a static module bundler for modern JS applications when webpack processes your application, it internally builds a dependency graph from one or more entry points and then combine every module your project needs into one or more bundles which are static assets to serve your content from.

Core concepts –

Entry – an entry point indicates which module webpack should use to begin building out its internal dependency graph.

Output – the output property tells webpack where to emit the bundles it creates and how to name these files. It defaults to ./dist/main.js for the main output file and to ./dist folder for any other generated file.

Loaders – out of the box webpack only understands JS and JSON files. Loaders allow webpack to process other types of files and convert them into valid modules that can be consumed by your application and added to the dependency graph.

Plugins – while loaders are used to transform certain types of modules plugins can be leveraged to perform a wider range of tasks like bundle optimization asset management and injection of environment variables.

Mode – by setting the mode parameter to either development, production or none, you can enable webpack’s built-in optimization that corresponds to each environment. The default value in production.

1. What is .parcel-cache?

Parcel caches everything it builds to disk. If you restart the dev server, parcel will only rebuild files that have changed since the last time it ran. By default cache is stored in .parcle-cache.

1. What is npx?

Node package execute and it comes with npm. It is a npm package runner that can execute any package that you want from the npm registry.

1. Diff b/w dependencies and dev-dependencies?

In every web application project, we have a file called package.json this file contains all the relevant data regarding the project i.e metadata. Starting from all the dependencies used to all the version numbers are present in the file.

Dependencies – in package.json file there is an object called dependencies and it consists of all the packages that are used on the project with its version number. So whenever you install any library that is required in your project that library you can find it in the dependencies object.

Dev dependencies – in package.json file, there is an object called as dev dependencies and it consists of all the packages that are used in the project in its development phase and not in the production or testing environment with its version number.

Ex – bootstrap

1. Tree shaking- it is a tern commonly used in the JS context for dead code elimination. Ex comments and unused functions.
2. Hot module replacement? As you make changes to your code, parcel automatically rebuild the changed files and updates your app in browser. By default parcel fully reloads the page, but in some cases it may perform hot module replacement HMR. HMR improves the development experience by updating modules in the browser at runtime without needing a whole page refresh. CSS changes are automatically applied via HMR with no page reload necessary.
3. Powers of parcel? Minification , bundling ,compression ,caching faster build ,content hashing ,tree shaking.
4. Git-ignore – it is file in fit folder that specifies intentionally untracked files to ignore.
5. Diff b/w package.json and package-lock.json?

Package.json records the minimum version your app needs. If you update the versions of a particular package, the change is not going to be reflected here.

Package-lock.json – records the exact version of each installed package which allow you to reinstall them. Future installs will be able to build an identical dependency tree.

1. Why I shouldn’t modify package-lock.json?

It stores an exact version dependency tree rather than using starred versioning like package.json itself this means you can guarantee the dependencies for other developers or prod releases.

1. What is node-modules?

In simple terms a module is a piece of reusable javascript code. It could be a .js file or a directory containing .js files. You can export the content of these files and use them in other files.

Modules help developers adhere to the DRY principle on programming.

We should not push node modules in github repo.

1. What is dist folder?

When we build a dev of production build of our project resulting files are stored in dist folder. Which are optimized in certain ways.

1. What is browser lists?

Browser list is a tool that allows specifying which browser should be supported in your frontend app.

1. ^(caret) and ~(tilde) in package.json?

~ will update you to all future patch version without incrementing the minor version.

^ will update you to all future minor and patch version without incrementing major version.

1. Script types in html?

The type attribute of the script elements indicate the type of script represented by the element: classic script, a javascript module , an import map, or a data block.

Attribute is not set (default) an empty string -> indicates that the script is a classic script containing javascript code.

Module – this value causes the code to be treated as a js module the processing of the script contents is deferred. The charset and defer attributes of the script is deferred.

Importmap – this value indicates that the body of the element contains import map the import map is a json object that developers can use to control how the browsers resolves module specifiers when importing js modules.

Assignment 3

1. What is JSX?

JSX is an embeddable XML-like syntax. It is meant to be transformed into valid javascript, through the semantics of that transformation are implementation specific.

Jsx is javascript extension syntax used in react to easily write HTML in js file.

Brower does not understand this jsx because it is not valid js code. This is because we are assigning html tag to a variable.

So to convert it to browser understandable javascript code, we use a tool like babel which is JS compiler / transpiler.

1. Advantages of using JSX in react js?

Help us in keeping our code simpler and elegant when writing large piece of code.

Jsx also allows React to show more useful errors and warning messages.

Faster than normal js as it performs optimization while translating to regular js.

Assignment 4

1. Is JSX mandatory for react? No
2. Is es6 mandatory for react? No babel convert it for older browsers.
3. Diff b/w {tc} {<tc/>} and {<tc></tc>} ? all three can be used to include functional component in JSX.
4. How to write comment in JSX? {/\*comment\*/}.
5. What is react fragment ? it is used to make parent element in jsx.
6. What is virtual DOM?

Virtual DOM is a programming concept where an ideal or virtual representation of a UI is kept in memory and synced with the real DOM by a library such as ReactDOM. The process is called reconciliation. This approach enables the declarative API of react: you tell react what state you want the UI to be in, and it makes sure the DOM matches that state. This abstract out the attribute manipulation event handling and manual DOM updating that you would otherwise have to use to build your app.

Since virtual DOM is more of a pattern than a specific technology people sometimes say it to mean different things. In React world the term VDOM is usually associated with react element since they are the objects representing the user interface. React however uses internal objects called fibers to hold additional information about the component tree. They may also be considered a part of VDOM implementation in React. It is diff from shadow dom.

1. React fiber is new reconciliation engine in React16. Its main goal is to enable incremental rendering of the virtual DOM.
2. What is reconciliation?

React provide a declarative api so that you don’t have to worry about exactly what changes on every update. This makes writing application a lot easier, but it might not be obvious how this is implemented within react. It uses diffing algorithm.

When you use React at a single point in time you can think of the render function as creating a tree of react elements. On the next state or props update that render() function will return a different tree of React elements. React then need to figure out how to efficiently update the UI to match the most recent tree.

There are some generic solutions to this algorithmic problem of generating the minimum number of operations to transform one tree into another. However the state of art algorithms have a complexity in the order of O(N^3) where n is number of elements.

If we used this in react displaying 1000 element will require order of one billon comparisons. This is far too expensive. Instead, react implements a heuristic O(N) algo based on 2 assumptions.

1. Two elements of diff types will produce different trees.
2. The developer can hint at which child elements may be stable across different renders with a key prop.

The diffing algorithm

When diffing two trees react first compares the two root elements. The behavior is different depending on the types of the root elements.

Elements of different types

Whenever the root elements have different types react will tear down old tree and build new tree from scratch.

Dom elements of the same type

When comparing two react DOM elements of the same type, react looks at the attribute of both keeps the same underlying dom node and only updates the changed attribute.

Component elements of the same type

When a component updates the instance stays the same so that state is maintained across renders. React updates the props of underlying component instance to match the new element

Recursing on children

By default when recusing on the children of a DOM node, react just iterates over lists of children at the same time and generates a mutation whenever there’s a difference.

For example, when adding an element at the end of the children, converting between these two trees works well:

<ul>

<li>first</li>

<li>second</li>

</ul>

<ul>

<li>first</li>

<li>second</li>

<li>third</li>

</ul>

React will match the two <li>first</li> trees, match the two <li>second</li> trees, and then insert the <li>third</li> tree.

If you implement it naively, inserting an element at the beginning has worse performance. For example, converting between these two trees works poorly:

<ul>

<li>Duke</li>

<li>Villanova</li>

</ul>

<ul>

<li>Connecticut</li>

<li>Duke</li>

<li>Villanova</li>

</ul>

React will mutate every child instead of realizing it can keep the <li>Duke</li> and <li>Villanova</li> subtrees intact. This inefficiency can be a problem.

Keys

In order to solve this issue this issue react supports a key attribute. When children have keys, react uses the key to match children in the original tree with children in the subsequent tree.

 For example, adding a key to our inefficient example above can make the tree conversion efficient:

Read about named exports and default exports