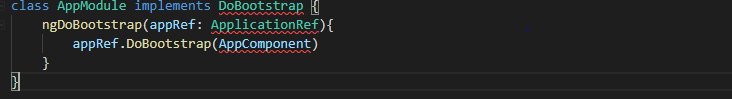
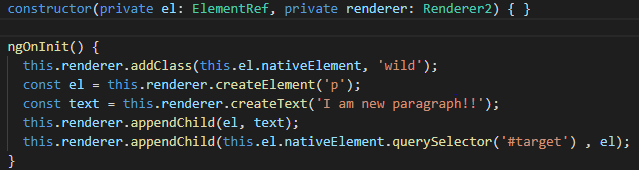
ngDoBootstrap: To Manually bootstrap the application.



Renderer2: This is used to manipulate DOM without having to touch the actual DOM directly and thus helps in developing apps which are to be rendered on environment where DOM is not accessible. For Example: App rendered on server, mobile, etc.



ViewChild**:** provides reference to child on template



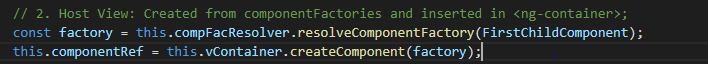
### Creating Views:

#### Embedded View:





#### Host View



## GULP

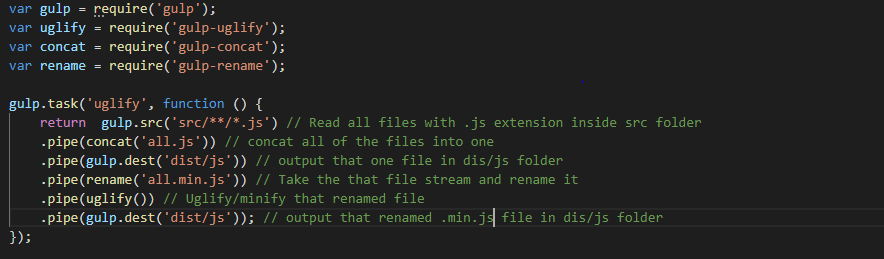
Gulp is an open source JavaScript toolkit and task runner that lets developers automate many development tasks. At a high level, gulp reads files as **streams** and pipes the streams to different tasks.

* Minification of scripts and styles: Removes white space and comments
* Concatenation: Creates one big file from multiple files
* Cache busing: Informs the browser when there is a new version of cached file
* Testing, linting and optimization
* Dev servers

There are single purpose plugins available which perform a single task and output of one task becomes input of another task. Plugin examples:

* gulp-concat
* gulp-rename
* gulp-uglify

Creates two files all.js and all.min.js



### Webpack

Webpack is an application bundler which takes multiple files and bundles them into few large files like one .js, one .css, one .jpg and one .png file.

Webpack has one entry point (for example: main.js) then loads its dependencies according to import statement and then dependencies of those dependencies and thus creating a dependency tree. Then creates few large bundles and then outputs those bundles in the specified outDir location.

In addition webpack provides plugins to do extra tasks like transpiling, minification, uglification, AOT, obfusication

## AOT

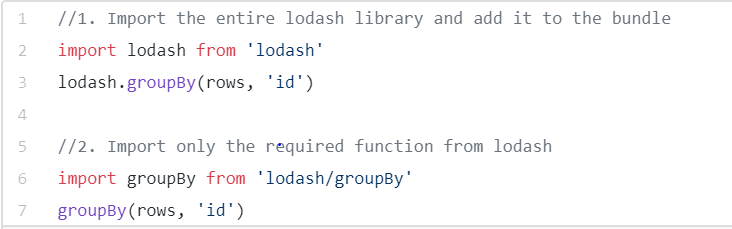
* **Faster rendering:** Application is compiled during the build process only, that means there is no need to have compiler in the final build and thus browser will render application fast
* **Fewer asynchronous requests:** Compiler inlines application HTML templates and CSS style sheets into JavaScript files thus eliminating separate AJAX request for them
* **Smaller angular download size:** As the application is already compiled, there is no need to download the compiler
* **Earlier detection of HTML template errors:** As AOT compiles templates in build process, it give template errors already, without having to run the application

### Ng build --prod

* **AOT**: Ahead of time compilation
* **Production mode**: Deploys production environment
* **Bundling**: Concatenates many application and library files into few bundles
* **Minification**: Removes white space and comments
* **Uglificaiton**: Uglifies the code by using short names and variables
* **Dead code elimination**: Removes unused code

## Reduce Bundle Size

* Compress the code: [compression-webpack-plugin](https://github.com/webpack/compression-webpack-plugin) generates .gz file
* Remove unnecessary imports like



* When files are minified and compressed in bundling process; a sourceMap file is also generated which stores information of original files. For production mode this sourceMap file can be altered to store only minimal information to reduce the size

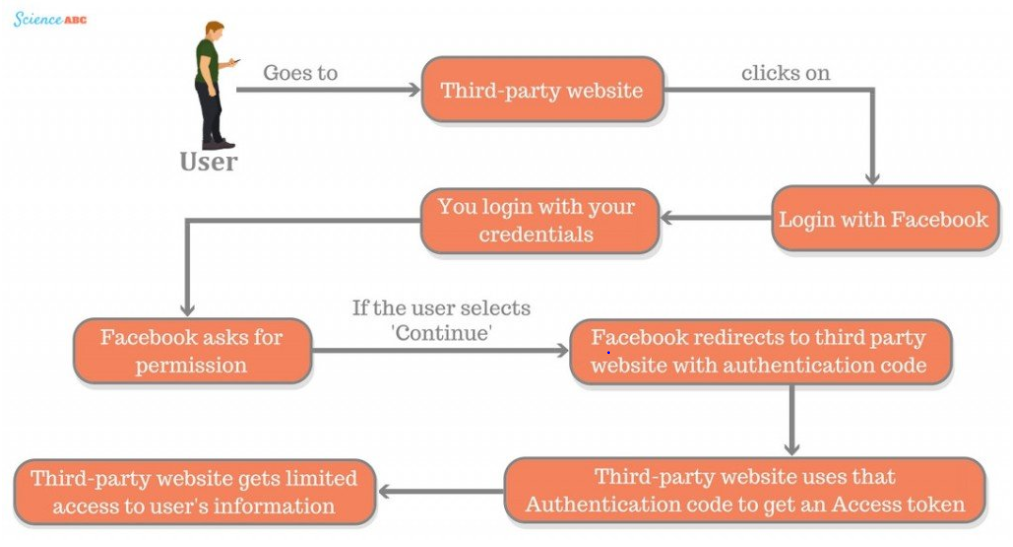
This setting can be altered in webpack config to “**eval-source-map”**

## HTTP Status code

* 1xx: Informational: It means the request has been received and the process is continuing
* 2xx: Success: It means the action was successfully received, understood, and accepted
* 3xx: Redirection: It means further action must be taken in order to complete the request
* 4xx: Client Error: It means the request contains incorrect syntax or cannot be fulfilled
* 5xx: Server Error: It means the server failed to fulfill an apparently valid request

## OAuth

OAuth is a protocol that helps ‘big’ websites (websites that have a very large number of users, such as Google, Facebook etc.) grant access to its users’ information to third-party websites or applications without sharing the users’ passwords and other private, sensitive details.



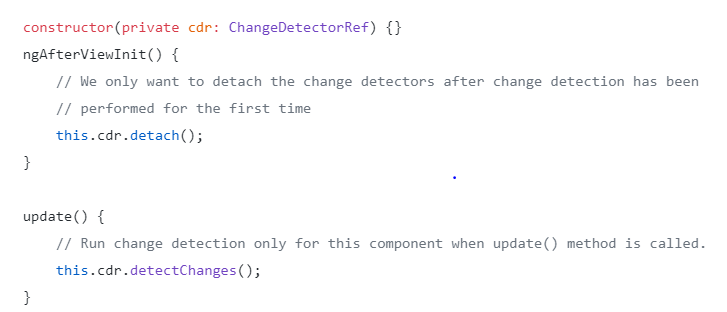
## Optimize Node.js code

1. Run asynchronous tasks in parallel
2. Cache application
3. All errors should be logged
4. Optimized query and better NOSQL DB like MongoDB
5. Cluster application with child processes
6. Use gZip for compression
7. Keep code base small and light by importing only the required modules

* https://medium.com/skyshidigital/6-tricks-to-speed-up-and-improve-your-node-js-performance-fadc06d15cbe
* <https://www.monitis.com/blog/top-7-node-js-performance-tips-you-can-adopt-today/>

## Optimize Angular application

* Use changedetection strategy.onPush()
* Detach changedetector and detect changes when required



* Use trackBy for lists in ngFor
* Lazy loading
* Use webworkers

# Rx/Js

import { of, from } from 'rxjs';

import { map, delay, switchAll, switchMap, mergeMap, concatMap } from 'rxjs/operators';

const getData = (param) => {

return of(`retrieved new data with param ${param}`).pipe(

delay(1000)

)

};

// From returns one observable at a time from [1,2,3,4] which is to be subscribed. getData additionally returns another observable which is to be subscribed as shown below

from([1, 2, 3, 4]).pipe(

map(param => getData(param))

).subscribe(val => val.subscribe(data => console.log(data)));

// mergeMap comes over the problem of subscribing twice(mergeMap = map + mergeAll). It subscribes to the first array element and calls getData. Never waits for the previous observable-chain to finish and goes ahead with the nextchain

from([1, 2, 3, 4]).pipe(

mergeMap(param => {

console.log('--')

return getData(param)

})

).subscribe(val => console.log('mergeMap:', val));

// switchMap(switchMap = subscribe + switch + map) subscribes to the arrayelement and calls getData and then unsubscribes to previous observable-chain if another observable is emitted

from([1, 2, 3, 4]).pipe(

switchMap(param => {

console.log('--')

return getData(param)

})

).subscribe(val => console.log('switchMap', val));

// concatMap subscribes to the arrayElement and calls getData and waits for it to finish and then subscirbes to the next arrayElement

from([1, 2, 3, 4]).pipe(

concatMap(param => {

console.log('--')

return getData(param)

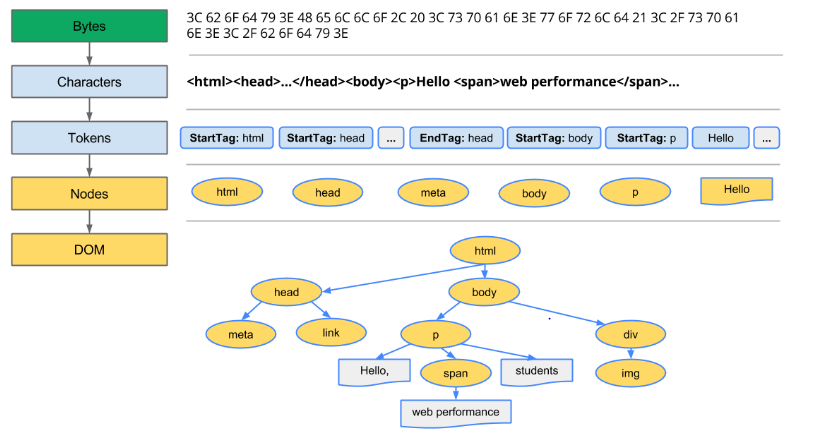
})

).subscribe(val => console.log('concatMap:', val));

## Paint Cycle

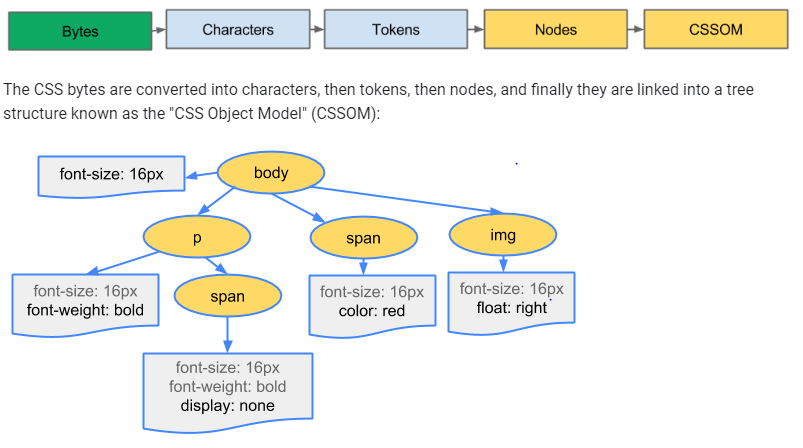
<https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-tree-construction>

1. Document Object Model Tree is created:



Every time the browser processes HTML markup, it goes through all of the steps above: convert bytes to characters, identify tokens, convert tokens to nodes, and build the DOM tree. This entire process can take some time, especially if we have a large amount of HTML to process.

1. CSS Object Model is created:



1. Render –Tree is created:

DOM Tree and CSS Trees are combined into one tree which has all the node information including their style properties. This tree doesn’t include nodes which are not to be rendered on the UI example: display none nodes, script tag, head tag, etc.

1. Layout/Reflow stage:

The output of this stage is a box model which has information of size and exact position of each node in pixels

1. Paint:

Final stage where layout stage’s information is used to paint the actual pixels on the screen.

## MAP

Collection of key and values. Keys are not converted to string unlike objects. 1 and “1” are considered different keys.

a = new Map();

a.set(1, “nitin”);

a.set(“1”, “Rawat”);

## Set

Set is a collection with unique values;

a = new Set();

a.add(1);

a.add(2);

a.add(1);

Second entry of 1 is ignored.

## WeakMap and WeakSet

If we put an object into an array/map/set, then while the array/map/set is alive, the object will be alive as well, even if there are no other references to it (even if we set object as undefined or null).

That object will not be garbage collected. To overcome this problem weakMap and weakSet are used.

WeakMap and WeakSet only take objects and no primitive types are allowed.

a= {b: 1};

c = [a];

a = undefined;

cosole.log(c[0]); // Return {b: 1}

## Design Patterns:

<https://medium.com/better-programming/javascript-design-patterns-25f0faaaa15>