

# نکات دوره

## Udemy – The Complete 2022 Web Development Bootcamp

سلام، وقتتون بخیر

این فایل، خلاصه نکات دوره Web Development Bootcamp از سایت Udemy هستش که خودم نوشتم، خواستم تو گیت هاب به اشتراک بزارم تا شاید بقیه هم استفاده کنن و به کارشون بیاد.

می تونید از لینک زیر جزئیات دوره رو از سایت یودمی ببینید:

[/https://www.udemy.com/course/the-complete-web-development-bootcamp](https://www.udemy.com/course/the-complete-web-development-bootcamp)

همچنین می تونید فیلم این دوره رو از سایت زیر دانلود کنید:

<https://downloadly.ir/elearning/video-tutorials/the-complete-2019-web-development-bootcamp-9>

The screenshot shows the course page for 'The Complete 2023 Web Development Bootcamp'. At the top, it says 'Development > Web Development'. The course title is 'The Complete 2023 Web Development Bootcamp'. Below the title, it says 'Become a Full-Stack Web Developer with just ONE course. HTML, CSS, Javascript, Node, React, MongoDB, Web3 and DApps'. A yellow 'Bestseller' badge indicates a 4.7 rating with 275,094 reviews and 931,950 students. It was created by Dr. Angela Yu and last updated on 3/2023. The course is available in English and Arabic [Auto], with 13 more subtitles. The price is \$24.99, discounted from \$124.99, with an 80% off offer. There is a 'Preview this course' button and a large play button icon. The background features a collage of web development icons like JS, React, MongoDB, and a person's face.

## Table of Contents

0. Tips .....	15
1. Front-End Web Development .....	17
4. How to Get the Most Out of the Course.....	17
5.1 12 Rules to Learn to Code eBook [Updated 26.11.18] .....	19
6. How Does the Internet Actually Work.....	19
7. How Do Websites Actually Work.....	20
8. What You'll Need to Get Started - Setup Your Local Web Development Environment .....	20
9. How to Get Help When You're Stuck.....	20
2. Introduction to HTML.....	20
1. Introduction to HTML.....	20
2. The Anatomy of an HTML Tag .....	21
3. What we're building - HTML Personal Site .....	22
4. What is The HTML Boilerplate .....	22
5. How to Structure Text in HTM.....	23
6. HTML Lists.....	24
7. HTML Image Elements .....	24
8. HTML Links and Anchor Tags.....	24
9. How to Ace this Course .....	24
3. Intermediate HTML.....	25
1. HTML Tables .....	25
2. Using HTML Tables for Layout.....	25
3. HTML Tables Code Challenge .....	25
5. HTML Tables Solution Walkthrough .....	26
6. HTML Forms .....	26
7. Forms in Practice - Create a Contact Me Form .....	27
9. Publish Your Website! .....	27
10. Tip from Angela - Habit Building with the Calendar Trick .....	27

4. Introduction to CSS .....	28
1. Introduction to CSS .....	28
2. Inline CSS .....	28
4. Internal CSS.....	28
5. External CSS .....	29
6. How to Debug CSS Code.....	30
7. The Anatomy of CSS Syntax.....	30
8. CSS Selectors.....	31
9. Classes vs. Ids .....	31
11. Tip from Angela - Dealing with Distractions .....	32
5. Intermediate CSS .....	32
1. What We'll Make - Stylised Personal Site .....	32
2. What Are Favicons .....	32
3. HTML Divs .....	33
4. The Box Model of Website Styling.....	33
5. CSS Display Property .....	35
7. CSS Static and Relative Positioning.....	37
8. Absolute positioning .....	39
9. The Dark Art of Centering Elements with CSS.....	40
10. Font Styling in Our Personal Site .....	41
12. Adding Content to Our Website .....	43
13. CSS Sizing .....	43
17. CSS Font Property Challenge Solutions.....	44
18. CSS Float and Clear.....	44
20. Stylised Personal Site Solution Walkthrough.....	45
22. Tip from Angela - Nothing Easy is Worth Doing! .....	45
6. Introduction to Bootstrap 4 .....	45
1. What is Bootstrap .....	45
2. Installing Bootstrap .....	47

3. Web Design 101 – Wireframing .....	48
4. The Bootstrap Navigation Bar .....	51
5. What We'll Make TinDog .....	52
7. Setting Up Our New Project .....	52
8. Bootstrap Grid Layout System.....	53
10. Adding Grid Layouts to Our Website .....	53
12. Bootstrap Containers .....	54
13. Bootstrap Buttons & Font Awesome .....	55
14. Styling Our Website Challenges and Solutions .....	55
16. Solution to Bootstrap Challenge 1.....	56
17. Tip from Angela - How to Deal with Procrastination .....	56
7. Intermediate Bootstrap .....	56
1. The Bootstrap Carousel Part 1.....	56
2. The Bootstrap Carousel Part 2.....	57
3. Bootstrap Cards .....	58
4. The CSS Z-Index and Stacking Order.....	58
5. Media Query Breakpoints .....	59
7. Bootstrap Challenge 2 Solution .....	62
8. How to become a Better Programmer - Code Refactoring .....	63
9. Put it into Practice - Refactor our Website Part 1.....	64
10. Advanced CSS - Combining Selectors .....	65
11. Refactoring our Website Part 2 .....	66
12. Advanced CSS - Selector Priority .....	67
13. Completing the Website.....	67
15. Tip from Angela - Building a Programming Habit.....	68
8. Web Design School - Create a Website that People Love .....	68
1. Introduction to Web Design .....	68
2. Understanding Colour Theory .....	68
3. Understanding Typography and How to Choose a Font .....	72

4. Manage ATTENTION with effective User Interface (UI) Design.....	78
5. User Experience (UX) Design .....	83
6. Web Design in Practice - Let's apply what we've learnt! .....	88
9. Introduction to Javascript ES6.....	91
1. Introduction to Javascript .....	91
2. Javascript Alerts - Adding Behaviour to Websites .....	93
3. Data Types .....	94
4. Javascript Variables.....	94
5. Javascript Variables Exercise Start.....	95
7. Javascript Variables Exercise Solution .....	95
8. Naming and Naming Conventions for Javascript Variables.....	96
10. String Concatenation.....	97
11. String Lengths and Retrieving the Number of Characters.....	97
12. Slicing and Extracting Parts of a String .....	98
13. Challenge Changing Casing in Text .....	98
14. Challenge Changing String Casing Solution.....	99
15. Basic Arithmetic and the Modulo Operator in Javascript .....	99
16. Increment and Decrement Expressions.....	100
18. Functions Part 1 Creating and Calling Functions.....	100
19. Functions Part 1 Challenge - The Karel Robot.....	101
22. Functions Part 2 Parameters and Arguments .....	101
24. Life in Weeks Solution .....	101
25. Functions Part 3 Outputs & Return Values .....	102
26. Challenge Create a BMI Calculator .....	103
28. Challenge BMI Calculator Solution .....	103
30. Tip from Angela - Set Your Expectations .....	103
10. Intermediate Javascript.....	104
1. Random Number Generation in Javascript Building a Love Calculator .....	104
2. Control Statements Using If-Else Conditionals & Logic.....	104

3. Comparators and Equality.....	105
4. Combining Comparators .....	106
6. Introducing the Leap Year Code Challenge.....	106
8. Leap Year Solution .....	106
9. Collections Working with Javascript Arrays .....	108
10. Adding Elements and Intermediate Array Techniques .....	109
12. Who's Buying Lunch Solution .....	110
13. Control Statements While Loops.....	110
15. Control Statements For Loops.....	112
16. Introducing the Fibonacci Code Challenge .....	113
18. Fibonacci Solution .....	113
19. Tip from Angela - Retrieval is How You Learn.....	115
11. The Document Object Model (DOM) .....	115
1. Adding Javascript to Websites.....	115
2. Introduction to the Document Object Model (DOM) .....	116
4. Selecting HTML Elements with Javascript.....	119
5. Manipulating and Changing Styles of HTML Elements with Javascript .....	120
6. The Separation of Concerns Structure vs Style vs Behaviour .....	121
7. Text Manipulation and the Text Content Property.....	122
8. Manipulating HTML Element Attributes.....	122
9. Tip from Angela - The 20 Minute Method .....	123
12. Boss Level Challenge 1 - The Dicee Game .....	123
1. Challenge The Dicee Challenge .....	123
9. The Solution to the Dicee Challenge.....	123
11. Tip from Angela - Learning Before you Eat .....	124
13. Advanced Javascript and DOM Manipulation.....	124
1. What We'll Make Drum Kit.....	124
3. Adding Event Listeners to a Button .....	124
4. Higher Order Functions and Passing Functions as Arguments.....	125

6. How to Play Sounds on a Website.....	127
7. A Deeper Understanding of Javascript Objects .....	128
8. How to Use Switch Statements in Javascript .....	130
9. Objects, their Methods and the Dot Notation.....	131
11. Using Keyboard Event Listeners to Check for Key Presses.....	133
12. Understanding Callbacks and How to Respond to Events.....	134
13. Adding Animation to Websites.....	136
15. Tip from Angela - Dealing with Lack of Progress.....	136
14. jQuery .....	136
1. What is jQuery .....	136
2. How to Incorporate jQuery into Websites.....	137
3. How Minification Works to Reduce File Size .....	138
4. Selecting Elements with jQuery.....	139
5. Manipulating Styles with jQuery .....	139
6. Manipulating Text with jQuery.....	140
7. Manipulating Attributes with jQuery .....	140
8. Adding Event Listeners with jQuery .....	140
9. Adding and Removing Elements with jQuery .....	141
10. Website Animations with jQuery .....	142
11. Tip from Angela - Mixing Knowledge.....	142
15. Boss Level Challenge 2 - The Simon Game .....	143
1. What You'll Make The Simon Game .....	143
25. Tip from Angela - Dealing with Frustration.....	143
16. The Unix Command Line .....	144
2. Command Line Hyper Setup.....	144
3. Understanding the Command Line. Long Live the Command Line! .....	144
4. Command Line Techniques and Directory Navigation .....	145
5. Creating, Opening, and Removing Files through the Command Line .....	146
6. Tip from Angela - Sleep is My Secret Weapon.....	146

17. Backend Web Development.....	147
1. Backend Web Development Explained.....	147
18. Node.js.....	148
1. What is Node.js .....	148
4. The Power of the Command Line and How to Use Node.....	150
5. The Node REPL (Read Evaluation Print Loops).....	151
6. How to Use the Native Node Modules .....	152
7. The NPM Package Manager and Installing External Node Modules .....	153
8. Tip from Angela - Step Up to the Challenge .....	154
19. Express.js with Node.js.....	154
1. What is Express .....	154
2. Creating Our First Server with Express .....	156
3. Handling Requests and Responses the GET Request .....	156
5. Understanding and Working with Routes.....	157
6. What We'll Make A Calculator .....	158
8. Calculator Setup Challenge Solution .....	159
9. Responding to Requests with HTML Files.....	159
10. Processing Post Requests with Body Parser .....	160
12. Solution to the BMI Routing Challenge.....	162
13. Tip from Angela - How to Solidify Your Knowledge .....	163
20. APIs - Application Programming Interfaces .....	164
1. Why Do We Need APIs .....	164
2. API Endpoints, Paths and Parameters .....	166
3. API Authentication and Postman .....	168
4. What is JSON .....	169
5. Making GET Requests with the Node HTTPS Module .....	171
6. How to Parse JSON.....	172
7. Using Express to Render a Website with Live API Data .....	173
8. Using Body Parser to Parse POST Requests to the Server.....	174

9. The Mailchimp API - What You'll Make .....	175
10. Setting Up the Sign Up Page.....	175
11. Posting Data to Mailchimp's Servers via their API .....	176
12. Adding Success and Failure Pages .....	177
13. Deploying Your Server with Heroku .....	178
14. Tip from Angela - Location, Location, Location! .....	180
21. Git, Github and Version Control .....	180
1. Introduction to Version Control and Git .....	180
2. Version Control Using Git and the Command Line .....	181
3. GitHub and Remote Repositories .....	182
5. Gitignore .....	183
6. Cloning .....	184
7. Branching and Merging .....	185
9. Forking and Pull Requests .....	187
10. Tip from Angela - Spaced Repetition .....	190
22. EJS.....	190
1. What We'll Make A ToDoList.....	190
3. Templates Why Do We Need Templates .....	190
4. Creating Your First EJS Templates .....	191
5. Running Code Inside the EJS Template.....	193
6. Passing Data from Your Webpage to Your Server.....	193
7. The Concept of Scope in the Context of Javascript.....	195
8. Adding Pre-Made CSS Stylesheets to Your Website .....	197
9. Understanding Templating vs. Layouts .....	198
10. Understanding Node Module Exports How to Pass Functions and Data between Files.....	200
11. Tip from Angela - Use Accountability in your Favour.....	203
23. Boss Level Challenge 3 - Blog Website .....	203
1. A New Challenge Format and What We'll Make A Blog.....	203

2. Setting Up the Blog Project .....	204
3 – 44 challenges.....	205
45. Tip from Angela - When Life Gives You Lemons .....	208
24. Databases .....	208
1. Databases Explained SQL vs. NOSQL .....	208
25. SQL.....	212
1. SQL Commands CREATE Table and INSERT Data .....	212
2. SQL Commands READ, SELECT, and WHERE .....	214
3. Updating Single Values and Adding Columns in SQL.....	215
4. SQL Commands DELETE.....	217
5. Understanding SQL Relationships, Foreign Keys and Inner Joins .....	217
6. Tip from Angela - Find All the Hard Working People .....	219
26. MongoDB .....	220
1. Installing MongoDB on Mac .....	220
2. Installing MongoDB on Windows .....	220
3. MongoDB CRUD Operations in the Shell Create.....	220
4. MongoDB CRUD Operations in the Shell Reading & Queries.....	221
5. MongoDB CRUD Operations in the Shell Update.....	222
6. MongoDB CRUD Operations in the Shell Delete .....	223
7. Relationships in MongoDB .....	224
8. Working with The Native MongoDB Driver .....	226
10. Tip from Angela - Daily Routines .....	230
27. Mongoose .....	230
1. Introduction to Mongoose .....	230
2. Reading from Your Database with Mongoose .....	234
3. Data Validation with Mongoose .....	234
4. Updating and Deleting Data Using Mongoose.....	236
5. Establishing Relationships and Embedding Documents using Mongoose... ..	238
6. Tip from Angela - Deep Work.....	239

28. Putting Everything Together .....	239
1. Let's take the ToDoList Project to the Next Level and Connect it with Mongoose .....	239
2. Rendering Database Items in the ToDoList App .....	240
3. Adding New Items to our ToDoList Database .....	241
4. Deleting Items from our ToDoList Database.....	242
5. Creating Custom Lists using Express Route Parameters.....	244
6. Adding New Items to the Custom ToDo Lists .....	246
7. Revisiting Lodash and Deleting Items from Custom ToDo Lists .....	247
8. Tip from Angela - One Step at a Time.....	250
29. Deploying Your Web Application.....	250
1. How to Deploy Web Apps with a Database .....	250
2. How to Setup MongoDB Atlas .....	251
3. Deploying an App with a Database to Heroku .....	253
4. Tip from Angela - Discipline Breeds Discipline.....	254
30. Boss Level Challenge 4 - Blog Website Upgrade .....	255
1. Challenge Give your Blog a Database .....	255
8. Tip from Angela - Dealing with Limitations.....	257
31. Build Your Own RESTful API From Scratch.....	257
1. What is REST .....	257
2. Creating a Database with Robo 3T .....	261
3. Set Up Server Challenge .....	262
4. Set Up Server Solution .....	263
5. GET All Articles .....	264
6. POST a New Article.....	266
7. DELTE All Articles .....	269
8. Chained Route Handlers Using Express .....	271
9. GET a Specific Article.....	272
10. PUT a Specific Article.....	275

11. PATCH a Specific Article .....	276
12. DELETE a Specific Article .....	279
14. Tip from Angela - How to Get a Job as Programmer.....	280
32. Authentication & Security .....	280
1. Introduction to Authentication .....	280
2. Getting Set Up.....	284
3. Level 1 - Register Users with Username and Password .....	285
5. Level 2 - Database Encryption .....	288
6. Using Environment Variables to Keep Secrets Safe .....	290
7. Level 3 - Hashing Passwords.....	293
8. Hacking 101.....	299
9. Level 4 - Salting and Hashing Passwords with bcrypt .....	305
10. What are Cookies and Sessions .....	312
11. Using Passport.js to Add Cookies and Sessions .....	317
12. Level 6 - OAuth 2.0 & How to Implement Sign In with Google .....	322
13. Finishing Up the App - Letting Users Submit Secrets .....	336
15. Tip from Angela - How to Work as a Freelancer .....	338
33. React.js .....	338
1. What is React .....	338
2. What we will make in this React module .....	342
3. Introduction to Code Sandbox and the Structure of the Module .....	344
4. Introduction to JSX and Babel .....	346
5. JSX Code Practice .....	349
6. Javascript Expressions in JSX & ES6 Template Literals .....	350
7. Javascript Expressions in JSX Practice.....	351
8. JSX Attributes & Styling React Elements.....	352
9. Inline Styling for React Elements .....	353
10. React Styling Practice .....	354
11. React Components .....	355

12. React Components Practice .....	355
13. Javascript ES6 - Import, Export and Modules .....	356
14. Javascript ES6 Import, Export and Modules Practice.....	356
15. [Windows] Local Environment Setup for React Development.....	356
16. [Mac] Local Environment Setup for React Development.....	356
17. Keeper App Project - Part 1 Challenge.....	356
18. Keeper App Part 1 Solution .....	357
19. React Props .....	357
20. React Props Practice.....	358
21. React DevTools.....	358
22. Mapping Data to Components .....	359
23. Mapping Data to Components Practice.....	359
24. Javascript ES6 MapFilterReduce.....	360
25. Javascript ES6 Arrow functions.....	361
26. Keeper App Project - Part 2 .....	361
27. React Conditional Rendering with the Ternary Operator & AND Operator	361
28. Conditional Rendering Practice .....	362
29. State in React - Declarative vs. Imperative Programming.....	362
30. React Hooks – useState .....	364
31. useState Hook Practice .....	365
32. Javascript ES6 Object & Array Destructuring .....	365
33. Javascript ES6 Destructuring Challenge Solution .....	365
34. Event Handling in React .....	366
35. React Forms .....	367
36. Class Components vs. Functional Components .....	368
37. Changing Complex State .....	369
38. Changing Complex State Practice .....	369
39. Javascript ES6 Spread Operator.....	369
40. Javascript ES6 Spread Operator Practice .....	369

41. Managing a Component Tree .....	370
42. Managing a Component Tree Practice .....	370
43. Keeper App Project - Part 3 .....	370
44. React Dependencies & Styling the Keeper App .....	371
45. Tip from Angela - How to Build Your Own Product.....	372
34. Bonus Module Ask Angela Anything.....	373

mrshaker.ir

## 0. Tips

- لینک ها و سایت های استفاده شده در دوره در لینک زیر موجود می باشد.  
<https://www.appbrewery.co/p/web-development-course-resources>
- سایت codepen.io برای تمرین اجرای کد های front-end می باشد.
- برای جستجوی مفاهیم به صورت مستند می توان از سایت های MDN ، devdocs.io و w3school استفاده کرد. کافیست عنوان مورد نظر + عنوان سایت را در گوگل سرچ کنید تا مستندات مورد نظر پیدا و اطلاعات دلخواه را کسب کنید.
- برای سریع تر شدن تایپ، atom keyboard shortcuts را یاد بگیرید.
- برای سریع تر شدن تایپ، میانبر افزونه های اضافه شده در atom را پیدا کنید.
- توسط کلید میانبر Ctrl + Alt + B می توان توسط پکیج Atom beautify ، برجستگی و زیبا سازی کد را انجام داد.
- به دلیل نصب پکیج Emmet ، برای نوشتتن تگ، کافیست اسم آن المان را آورد تا به صورت خودکار تگ را ایجاد کند.
- توسط کلید میانبر Alt + mouse selection می توان توسط افزونه sublime-style ، به صورت کلی Text را تایپ کرد.
- با استفاده از کلید میانبر / Ctrl می توان خط کد را تبدیل به کامنت نمود.
- توسط کلید میانبر Ctrl + Shift + C می توان صفحه مورد نظر را Inspect گرفت.
- اطلاعات تماس با آنجلاء [wenxiu.yu@gmail.com](mailto:wenxiu.yu@gmail.com) و [twitter: yu\\_angela](#)
- توسط کلید CTRL+F می توان در کدهای نوشته شده جستجو کرد.
- به ورودی های تابع، Argument گفته می شود.
- رفتارها و Properties خصوصیات، Methods object می باشد.

- برای خروج در محیط ترمینال از هر برنامه ای کافیست از `exit`. یا دو بار `Ctrl + C` استفاده کنید و برای پاکسازی ترمینال از `clear` استفاده کنید. همچنین برای پیشنهاد کلمه دوبار از کلید `TAB` استفاده کنید.
- استفاده از دستور `help` برای هر ماژولی در محیط `shell` را فراموش نکنید.
- با دستور `.` در ترمینال می توان، مسیر فعلی را در `atom` به اجرا در آورد. (علامت نقطه یعنی مسیر فعلی، مثلاً `ls` محتوای مسیر فعلی را نشان می دهد).
- علامت "/" نشان دهنده `root rout` یا مسیر صفحه اصلی سرور می باشد.
- استفاده از `vpn` برای خدماتهای مختلف استفاده نشود چون IP ایران در اکثر موقع مسدود می باشد.
- همه ماژول ها و فریم ورک ها دارای `document` می باشند، پس به آنها مراجعه کنید.

# 1. Front-End Web Development

## 4. How to Get the Most Out of the Course

در این قسمت نکات دوره آموزشی برای ماندگاری بیشتر توضیح داده می شود:

- ۱- هر قسمت به گونه ای تنظیم شده است که ۱۰ دقیقه ای باشد. شما نباید همزمان با فیلم، کد بزنید، بلکه بعد تمام شدن هر ۱۰ دقیقه باید سعی کنید نکات فیلم را به یاد بیاورید و آن را درک کرده باشید و در ادامه با فهم خود، بدون کپی کردن، کدنویسی را انجام دهید.
- ۲- برای افزایش سرعت تایپ خود از سایت های آموزش تایپ مانند [keybr.com](http://keybr.com) استفاده کنید.
- ۳- برای داشتن یک نسخه کامل و قابل یادآوری، از [cornell note taking system](#) استفاده کنید که به شکل زیر می باشد. در ابتدا موضوع جلسه را بنویسید. در ادامه نکات درک شده را یادداشت نمایید(به خصوص بعد ۱۰ دقیقه). کلمات کلیدی برای یادآوری یا سوالات پیش آمده برای جستجو را وارد نمایید. در نهایت بعد از فهم کامل خلاصه را به صورت نکته ای جمع بندی کنید.

Topic/Title:		
Keywords/Questions:	Notes:	
Summary:		

۴- می توانید سرعت فیلم ها را در صورت نیاز افزایش و یا کاهش دهید، ولی همه فیلم ها را به ترتیب و پشت هم نگاه کنید.

۵- قسمت هایی را که متوجه نمی شوید، آن قسمت را مجدداً نگاه کنید تا به موضوع پی ببرید ولی اگر باز آن قسمت را درک نکردید، آن قسمت را علامت بزنید و سوال پیش آمده را یادداشت کنید و چند هفته بعد مجدداً به آن مراجعه کنید. حتماً با جستجو و درک برنامه نویسی، به آن پی خواهید برد.

۶- ویدیو ها را سرسری و سطحی نگاه نکنید. تسلط و درک کامل زمانی بر شما حاکم می شود، که چندین بار آن را تکرار کنید و کد بزنید.

۷- اگر در قسمت هایی گیج می شوید، حتماً از گوگل و stack overflow استفاده کنید و دلسرب نشوید.

۸- دنبال کردن مشکلات و رفع آنها هستند که سطح شما را بالا می برد و شما را به یک برنامه نویس حرفه ای تبدیل می کند. پس از حل مسئله نترسید و به یاد داشته باشید که چه رویایی دارید.

## 5.1 12 Rules to Learn to Code eBook [Updated 26.11.18]

حتماً این ۱۲ قانون برای یادگیری برنامه نویسی را هر چند وقت یکبار مطالعه کنید.

- ۱- ذهن خود را فریب دهید و از تکنیک ۲۰ دقیقه استفاده کنید و فقط شروع کنید.
- ۲- برای کدتان حتماً یه پروژه داشته باشید و در هنگام ساخت پروژه به آن زبان مسلط می شوید.
- ۳- هیچ زبان کاملی وجود ندارد، و همه زبان های برنامه نویسی کاربرد و جایگاه خود را دارند.
- ۴- کدهایی را که می نویسید را بفهمید و بافهم کامل کد بزنید. کپی و پیست کردن منوع.
- ۵- اگر چیزی نمی دونید در مورد قضیه برنامه نویسی اشکال نداره. اکثر موقع یه حسی هستش که هیچی نمی دونیم ولی این حس اشتباهه.
- ۶- اگر ایده برای پروژه نداری از ایده های دیگران کپی کن. مثلاً دفتر یادداشت خودت را داشته باش که در این مسیر کمک های زیادی می تونی بگیری.
- ۷- نسبت به یک شخص پاسخگو باشید و کدتan را نشان دهید. مثلاً در یک گروه آنلاین برنامه نویسی عضو شوید و معاشرت داشته باشید. ( دوره های آنلاین چون حضوری نیست، معمولاً افراد از آن دست می کشنند، پس حتماً حواستان باشد)
- ۸- همیشه به یاد گیری مداوم اداهید. از اول شروع کردن نترسید. یادگیری زبان های جدید، ساختاری مانند زبان های دیگر دارد، پس یادگیری آن سخت نخواهد بود.
- ۹- اگه خسته شدی، یا کدت هی خطا میده، باگ داره و مشکل را نمی تونی پیدا کنی، کار رو بازار کنار و به خودت استراحت بده. در اکثر موقع بعد از مراجعه مجدد، مشکل رو پیدا میکنی.
- ۱۰- حتما از یک مشاور برنامه نویسی یا یک برنامه نویس دیگر که به همدیگر کمک می کنند استفاده کنید تا پیشرفت کنید.
- ۱۱- پروژه های بزرگ را به صورت مسئله های کوچک تبدیل کنید و پروژه را به صورت مأذولات حل کنید.
- ۱۲- کد ها و برنامه های دیگران را کپی کنید و اجرا کنید. سعی کنید خط به خط آن را فهمید و به ساختار آن پی ببرید. این عمل باعث می شود به فهم عمیق و کامل در برنامه نویسی برسید و خلاقیت در شما افزایش پیدا خواهد کرد.  
فایل اصلی را برای جزئیات بیشتر، حتماً مطالعه کنید.

## 6. How Does the Internet Actually Work

یادداشت: در این قسمت درباره عملکرد واقعی اینترنت و شبکه جهانی صحبت می شه.

کلمات کلیدی: DNS, ISP, client, server, Ip

## 7. How Do Websites Actually Work

یادداشت: در این قسمت درباره فایل هایی که از سرور یک سایت دریافت می شه صحبت میشه :

Html : در واقع زیربنا و ساختار یک سایت می باشد مانند پنجره و دیوار و اسکلت یک ساختمان. -

Css : در واقع رنگ و طراحی یک سایت می باشد مانند رنگ و دکوراسیون داخلی یک ساختمان. -

Java script : مربوط به قسمت اکشن و قابلیت های زیبایی یک سایت می باشد مانند نور لامپ و یا تلویزیون یک ساختمان. -

كلمات کلیدی: Html, Css, Java script, Inspect element

## 8. What You'll Need to Get Started - Setup Your Local Web Development Environment

یادداشت: در این قسمت برنامه های اصلی مورد نیاز دوره توضیح داده می شود.

كلمات کلیدی: Atom text editor, Chrome, installation

## 9. How to Get Help When You're Stuck

یادداشت: در این قسمت نحوه رفع باغ، در حین دوره آموزشی توضیح داده می شود، برای اینکه هر قسمت بیشتر در ذهن شما بشیند، مراحل زیر را برای رفع خطای برنامه دنبال کنید:

۱- در مرحله اول سعی کنید خطای ایجاد شده را با Stack over flow برطرف کنید.

۲- اگر برطرف نشد، فیلم آن قسمت را مجدداً نگاه کنید و کد خود را اصلاح کنید.

۳- اگر باز برطرف نشد، در مرحله آخر کد مدرس را کپی کنید و آن را اجرا و خطای خود را پیدا کنید.

## 2. Introduction to HTML

### 1. Introduction to HTML

در این قسمت به معرفی اولیه Html پرداخته می شود:

html مخفف hyper text markup language می باشد، که در واقع از استاندارد markup یا

نشانه ها و علامت هایی جهت اجرای دستورات استفاده می کند. مانند زبان xml

از تگ های html به شکل <tag name> برای ایجاد ساختار و اسکلت بندي صفحه مورد نظر استفاده می کنیم.

هر تگی به صورت </tag name> باز و به صورت <tag name> بسته می شود. -

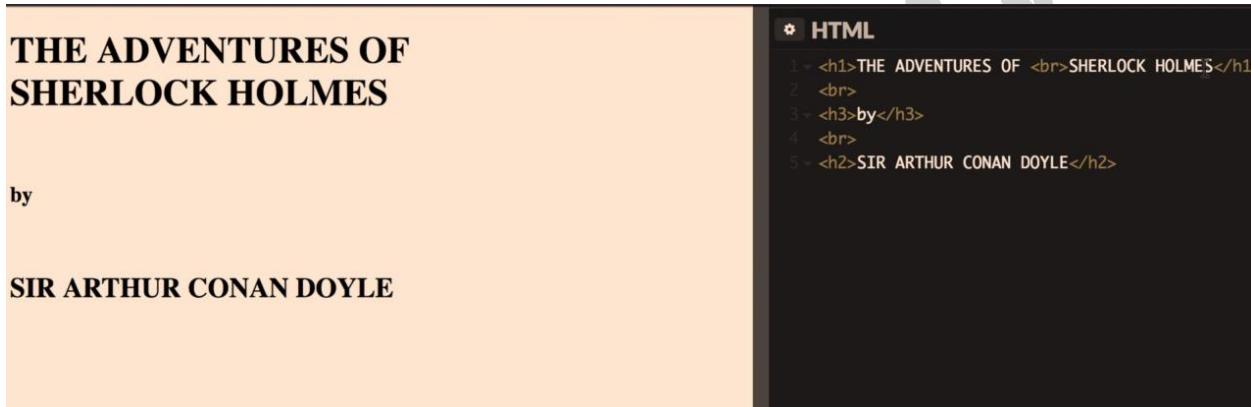
سایت codepen.io برای تمرین اجرای کد های front-end می باشد. -

المنت heading می شود که از تگ <h1> تا <h6> می باشد. -

HTML headings are titles or subtitles that you want to display on a webpage. -

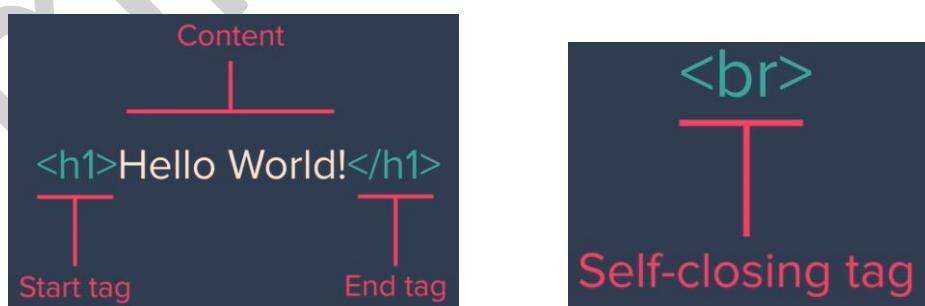
- المنت **line brake** به صورت `<br>` نوشته می شود که مانند **Enter** عمل می کند و نیازی به بسته کردن ندارد.
- برای جستجوی مفاهیم به صورت مستند می توان از سایت های [MDN](#) و [w3school](#) استفاده کرد. کافیست عنوان مورد نظر + عنوان سایت را در گوگل سرچ کنید تا مستندات مورد نظر پیدا و اطلاعات دلخواه را کسب کنید.

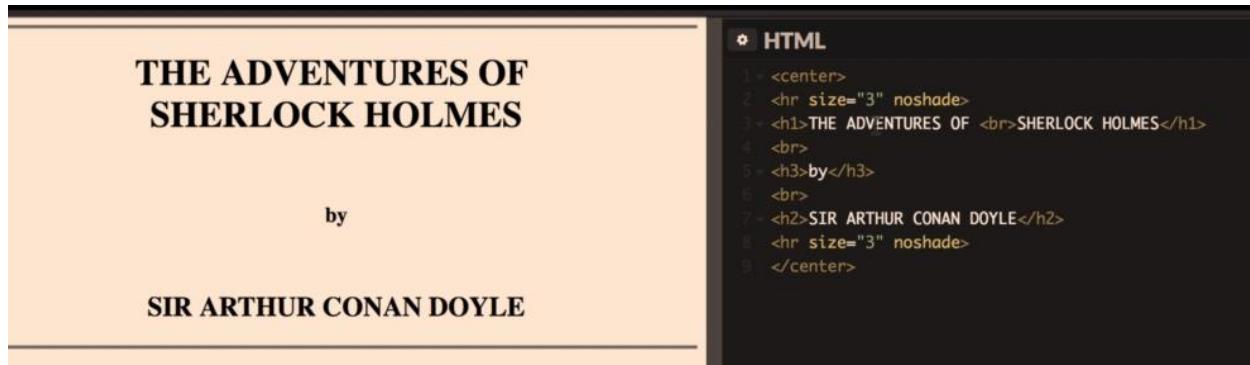
كلمات کلیدی : html heading, line brake, element, tags, search



## 2. The Anatomy of an HTML Tag

- در این قسمت آناتومی html مانند بررسی باز و بسته بودن تگ ها و attribute های هر تگ صحبت می شود:
- المنت **hr** برای ایجاد خط افقی استفاده می شود.
  - Attribute در داخل تگ قرار می گیرد، و می توان رفتارهای المنشی را کنترل نمود.
  - يکسری Attribute ها مشترک و يکسری اختصاصی آن المنشی می باشند.
  - از تگ `<!-- comment text -->` برای نوشتن کامنت در برنامه استفاده می شود.
  - از inspect element می توان برای پی بردن به تگ های استفاده شده در سایت پی برد.





**Key words:** tag omission, inspect element attribute, self-closing tags, comment tag, inspect element use, Attribute.

### 3. What we're building - HTML Personal Site

در این قسمت درباره ساخت یک صفحه شخصی رزومه، که فقط با html ساخته شده و به دهه ۹۰ میلادی مربوط می باشد، توضیحاتی داده می شود. از سایت [web.archive.org](http://web.archive.org) می توان به آرشیو سایت ها مختلف دسترس داشت.

### 4. What is The HTML Boilerplate

در این قسمت به بررسی کد قالب پیش فرض یا html boilerplate یا (توضیحات اصلی در داخل پروژه داده شده است)

شکل زیر standard HTML boilerplate می باشد. -

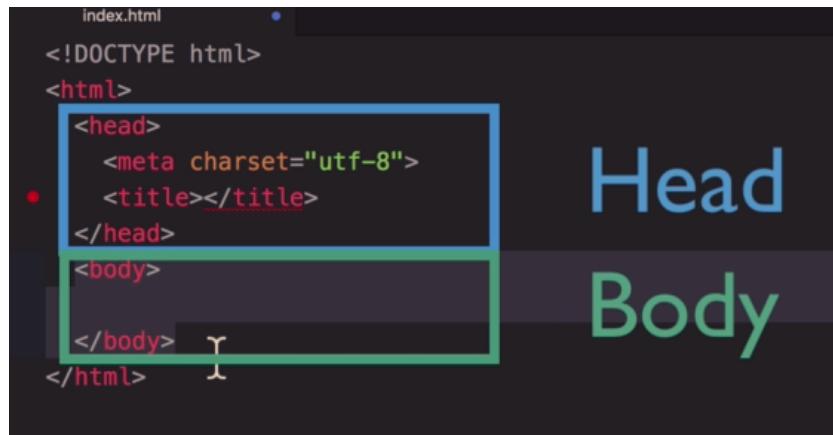
```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    •   <title></title>
  </head>
  <body>

  </body>
</html>

```

- برای سریع تر شدن تایپ، atom keyboard shortcuts را یاد بگیرید.
- برای سریع تر شدن تایپ، میانبر افزونه های اضافه شده در atom را پیدا کنید.



- تأثیر attribute های meta در موتور جستجوی گوگل.

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML, CSS, XML, JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

## MDN Web Docs

<https://developer.mozilla.org/> ▾

The Mozilla Developer Network (MDN) provides information about Open Web technologies including HTML, CSS, and APIs for both Web sites and HTML5 Apps.

You visited this page on 05/11/17.

Key words: standard HTML boilerplate, meta, utf-8, view page source.

## 5. How to Structure Text in HTML

در این قسمت درباره تگ های متنی در المنش body بررسی می شود:

- توسط کلید میانبر Ctrl + Alt + B می توان توسط پکیج Atom beautify ، برجستگی و زیبای سازی کد را انجام داد.
- به دلیل نصب پکیج Emmet ، برای نوشتن تگ، کافیست اسم آن المان را آورد تا به صورت خودکار تگ را ایجاد کند.

- بین *i* و **em** المنشی **em vs. i** را انتخاب کنید که نشان دهنده تاکید در متن پاراگراف می باشد ولی فقط ظاهر italic به متن می دهد.

- بین **b** و **strong** المنشی **strong vs. b** را انتخاب کنید که متن با اهمیت را bold می کند و فقط ظاهر نیست.

**Key words:** p, i, em, b, strong, text element, beautify

## 6. HTML Lists

در این قسمت نحوه افزودن Unordered-list و ordered-list توضیح داده می شود.

**Key words:** ol, ul, li, html list elements

## 7. HTML Image Elements

در این قسمت درباره image element صحبت خواهد شد و به پروژه تصویر اضافه می شود.

**Key words:** img, alt attribute, src image.

## 8. HTML Links and Anchor Tags

در این قسمت نحوه افزودن link به متن مورد نظر توسط المنشی anchor توضیح داده می شود.



- در این قسمت نحوه ایجاد لینک در واقع از HyperText Markup Language نشأت می گیرد، یعنی این قابلیت را داریم که لینک ها و صفحه های مختلف برای هر text ایجاد کنیم.

- برای افزودن لینک از anchor با تگ a استفاده می کنیم.

- از کلمه combak برای ایجاد سریع کامنت استفاده کنید.

**Key words:** anchor, hyperlink, href, combak.

## 9. How to Ace this Course

در این قسمت درباره با انگلیزه ماندن برای ادامه یادگیری صحبت می شود و فرم انگلیزشی تهیه شده است.

### 3. Intermediate HTML

#### 1. HTML Tables

در این قسمت درباره table در html صحبت می شود و اجزای اصلی و فرعی معرفی می گردد:

- درباره قسمت footer ، body ، heading table را شرح می دهد.

- یادتان باشد مفهوم یک کد یا تابع آن، مهم تر از ظاهر خروجی آن است.

- به سایت های فقط html سایت ها دهه ۹۰ میلادی گفته می شود که css و java script وجود ندارند.

<table>
<tr>
<td>Angela</td>
<td>12</td>
</tr>
<tr>
<td>Philipp</td>
<td>14</td>
</tr>
</table>

**Key words:** table, thead, th, tr, td, tbody, table cells, [Deprecated attributes](#),

#### 2. Using HTML Tables for Layout

در این قسمت نشان می دهد که چگونه می توان از Html table به عنوان یک layout یا طرح بندی (که در CSS رایج است) استفاده کرد.

**Key words:** layout with html table.

#### 3. HTML Tables Code Challenge

در این قسمت باید تمرین چالشی را حل کنید.

## 5. HTML Tables Solution Walkthrough

در این قسمت، تمرین قسمت قبل را حل می کند و چند نکته اضافه درباره `table` می گوید.

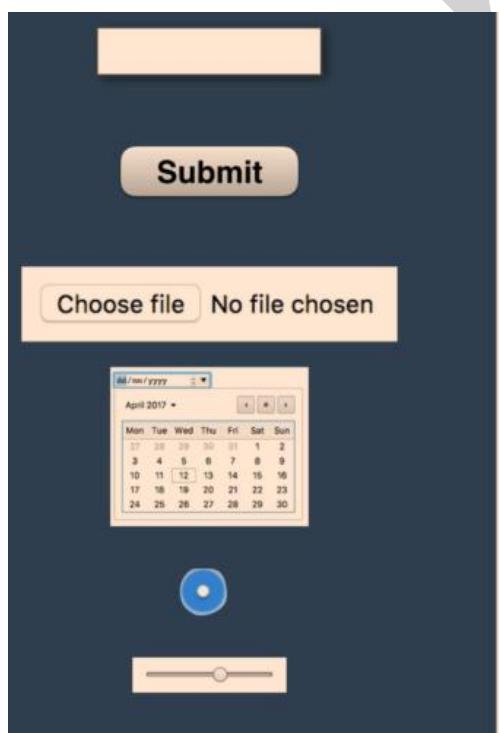
- توسط کلید `Win + .` می توان به اموجی ها دسترسی داشت.
- به دلیل پیچیدگی کار برای ایجاد `layout` با `html` ، `Css` استفاده می شود.

**Key words:** border, cellspacing, nested table, table in table.

## 6. HTML Forms

در این قسمت درباره `form` در `html` صحبت می شود.

```
<form>  
  <label>Name:</label>  
  <input type="text">  
  <input type="submit">  
</form>
```



```
<input type="text">  
  
<input type="submit">  
  
<input type="file">  
  
<input type="date">  
  
<input type="radio">  
  
<input type="range">
```

**Key words:** form, input, label, make forms.

## 7. Forms in Practice - Create a Contact Me Form

در این قسمت یک فرم ثبت ایمیل و نظرات توسط Form ایجاد می شود.

My Contact Details

My Fictional Address  
077263718463  
myemail@gmail.com

Your Name:   
Your Email:   
Your Message:

Submit

**Key words:** Form, textarea, name, action, submit message.

## 9. Publish Your Website!

در این قسمت نحوه host کردن CV ایجاد شده در github توضیح داده می شود که به صورت عمومی قابل دسترسی می باشد.

**Key words:** upload your made CV on Github.

## 10. Tip from Angela - Habit Building with the Calendar Trick

در این قسمت نکات پایانی و انگیزشی این فصل می باشد.

## 4. Introduction to CSS

### 1. Introduction to CSS

در این قسمت به معرفی اولیه CSS یا Cascading Style Sheets می پردازد. CSS فقط برای رنگ و لعاب دادن و زیبا سازی زبان های markup language به وجود آمد و پروسه زیباسازی و چیدمان HTML را راحت تر و وسیع تر می کند. به عبارتی وظیفه style دادن به html را دارد.

**Key words:** CSS Introduction, style.

### 2. Inline CSS

در این قسمت در مورد تغییر رنگ با style در پس زمینه سایت صحبت می شود.

Inline CSS یعنی کد style مربوط به CSS را در داخل تگ المنت html بنویسیم.

```
<body style="background-color: #95E1D3">
```

**Key words:** change background color, color Hexacode, css color, inline.

### 4. Internal CSS

در این قسمت تغییرات internal CSS را توسط style توضیح داده می شود، در واقع برای CSS یک تگ در صفحه مورد نظر ایجاد می کنیم که بهتر از حالت Inline می باشد.

- we have 2 standards, width and height, length and width

```
<style>

    body {
        background-color: #EAF6F6;
    }

    hr {
        border-style: none;
        border-top-style: dotted;
        border-color: grey;
        border-width: 5px;
        width: 5%;
    }

</style>
```

```
hr           display: block;
            margin-top: 0.5em;
            margin-bottom: 0.5em;
            margin-left: auto;
            margin-right: auto;
            border-style: inset;
            border-width: 1px;
```

**Key words:** default CSS browser value, internal CSS, Style, border-style, border-color, border-width.

## 5. External CSS

در این قسمت تغییرات را به روش استاندارد External CSS در پروژه اعمال می کنیم.

```

Project
HTML - Personal Site
  CSS
    styles.css
  images
  .DS_Store
  contact-me.html
  hobbies.html
  index.html

index.html
1 body {
2   background-color: #EAF6F6;
3 }
4
5 •hr {
6   border-style: none;
7   border-top-style: dotted;
8   border-color: grey;
9   border-width: 5px;
10  width: 5%;
11 }

```

**Key words:** External CSS, styles.css, css folder, change styles with CSS in all pages.

## 6. How to Debug CSS Code

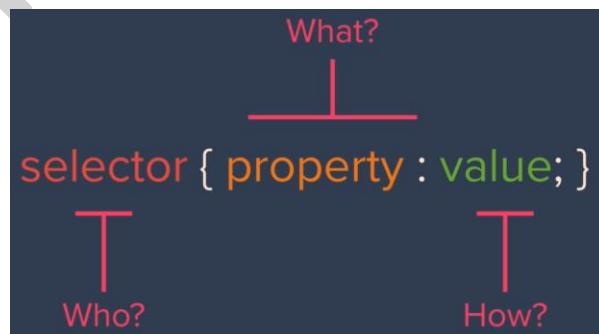
- در این قسمت ۲ کد را دیباگ می کند که از Web developer Tools مرورگر استفاده می کند تا مشکل را پیدا کند.

- در باره حق تقدم یا اولویت CSS به ترتیب inline و internal و کمترین آن external صحبت می کند که inline بالاترین اولویت اجرا در CSS را دارد. در حالت حرفة و کلی بهتر است از حالت external برای کارهایتان استفاده کنید.

**Key words:** priority, debugging, console, inline, internal, external.

## 7. The Anatomy of CSS Syntax

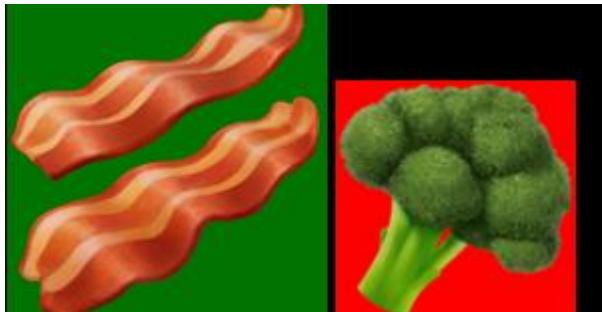
در این قسمت درباره syntax CSS یا همان گرامر نوشتاری زبان CSS صحبت می شود.



**Key words:** CSS syntax or code structure, Basic rule syntax, general properties, I love Bacon.

## 8. CSS Selectors

در این قسمت درباره انتخاب Selector در CSS توسط تعريف Class برای المنش ها و ایجاد تمایز بین المنش ها جهت استفاده در Styles صحبت می شود.



```

```

```
.bacon {
    background-color: green;
}

.broccoli {
    background-color: red;
}
```

**Key words:** Class Selector, Tag Selector, create difference between Elements.

## 9. Classes vs. Ids

در این قسمت تفاوت TAG ، Class و ID به عنوان Selector و حق تقدم آنها مورد بحث می باشد.

- در حالت عادی Default Style از مروگر گرفته می شود، در اولویت بعدی از Tag Selector ، و بعدی از Class Selector و در نهایت بالاترین اولویت ID Selector می باشد.
- Tag selector معمولاً زمانی که بخواهیم تغییرات style را برا کل Tag ها اعمال کنیم استفاده می شود.
- Class Selector معمولاً وقتی بخواهیم که تغییرات Style را برای گروهی از Tag ها اعمال کنیم استفاده می شود. (برای یک تگ می توان چندین Class تعريف کرد).

```
I Love Bacon</h1>
```

- در قسمت styles می توان از pseudo Class که شبه کلاس است استفاده کرد، که معمولاً به شکل Pseudo : می باشد. به عنوان مثال از کلاس hover : می توان زمانی که نشانگر موس بر روی المنت قرار گرفت، تغییرات style دلخواه صورت گیرد.

```
img:hover { /* COMBAK: when  
    background-color: gold;  
}
```

**Key words:** ID, Class, Selector, TAG, CSS, HTML, Differences, pseudo classes.

## 11. Tip from Angela - Dealing with Distractions

نکات انگلیزشی آخر فصل؛ برای افزایش تمرکز حتماً گوشی خود را کنار بگذاردید تا به کارتان عمق دهید.

گروه چت و تبادل نظر: <https://discord.gg/f7BBFBn>

# 5. Intermediate CSS

## 1. What We'll Make - Stylised Personal Site

در این قسمت درباره ایجاد پروژه فصل جدید صحبت می کند که یک سایت شخصی زیبا می باشد.

## 2. What Are Favicons

در این قسمت درباره ایجاد favicon صحبت می شود.

### 3. HTML Divs

در این قسمت درباره المنشی صحبت می شود.

- **Div** یا **divide** وظیفه ایجاد باکس با محتوای دلخواه در **body** را دارد. در واقع با تقسیم، یک گروه از المنشاهای در **body** ایجاد می‌کند که می‌توان تغییرات دلخواه **CSS** را به آنها اعمال نمود.

```
<body>
  <div class="">
    <h1>I'm Shaker</h1>
    <p>A programmer</p>
  </div>
```

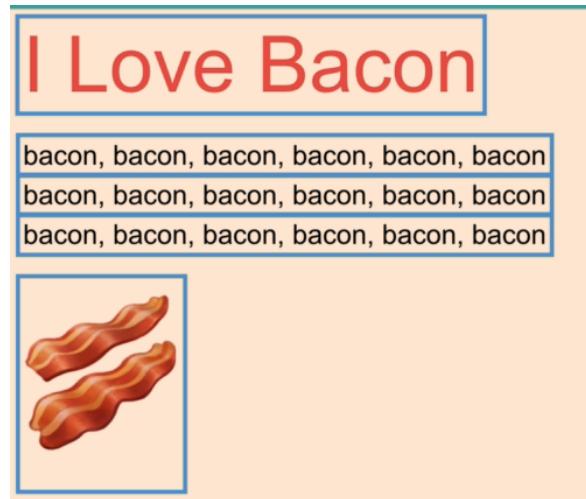
- برای حذف لبه ها یا margin های پیش فرض باید مقادیر آنها را تغییر داد. User agent stylesheet مقادیر پیش فرض CSS می باشد.

```
element.style {  
}  
  
body {  
    margin: 0;  
}  
  
body {  
    display: block;  
    margin: 8px;  
}
```

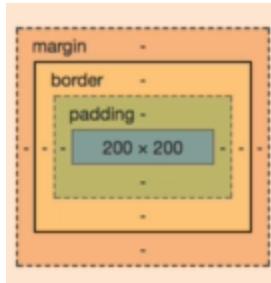
**Key words:** create div, change default margins.

## 4. The Box Model of Website Styling

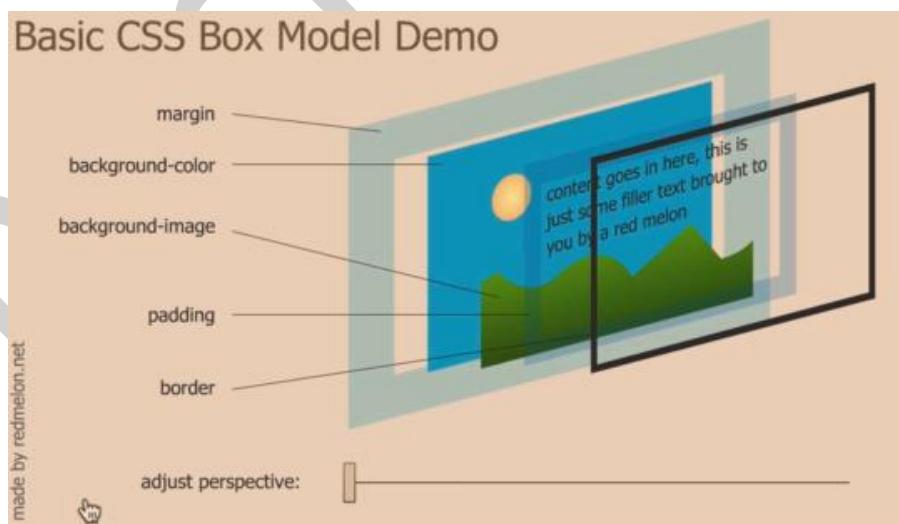
در این قسمت درباره مدل box هر یک از عناصر صحبت می شود.



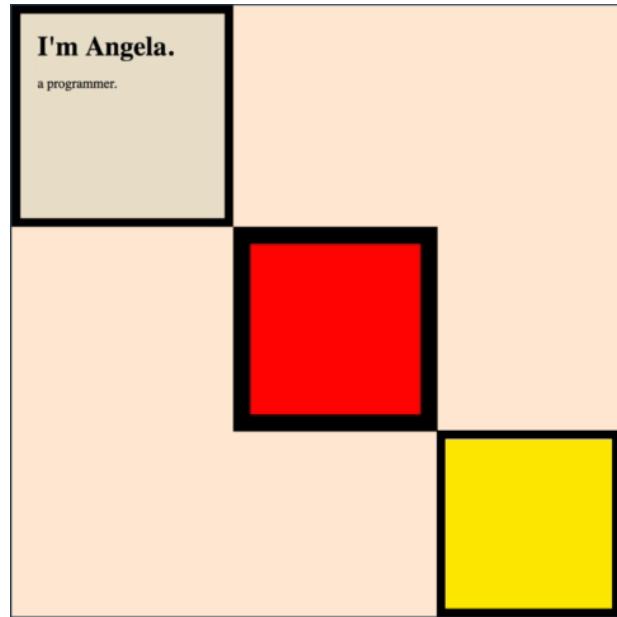
- هر باکس دارای padding ، border ، margin و ابعاد می باشد.



- فاصله باکس نسبت به اطراف آن می باشد، border مرز اطراف باکس می باشد، margin فاصله المنت های داخلی باکس نسبت به لبه های باکس می باشد و در نهایت ابعاد باکس را داریم.



- تغییر margin باعث تغییر ابعاد باکس نمی شود ولی تغییر padding و border باعث تغییر ابعاد باکس می شود.

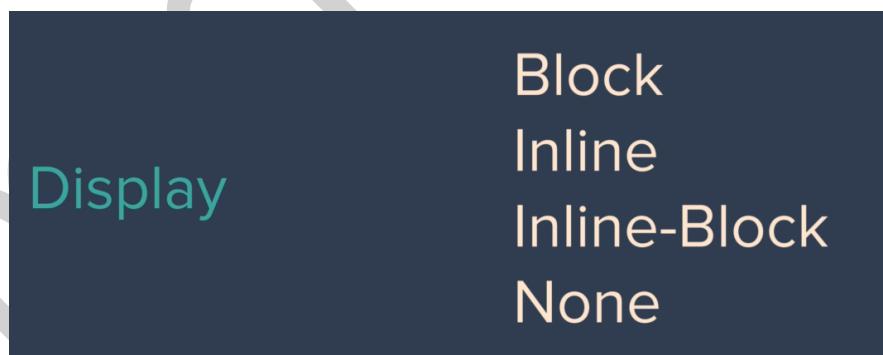


**Key words:** concept, box model, margin, border, padding, dimensions.

## 5. CSS Display Property

در این قسمت درباره خصوصیت **Display** المنشآت صحبت می شود.

در واقع یک خصوصیت در CSS می باشد که تعیین می کند هر المنشآت چگونه در کنار سایر المنشآت قرار گیرد یا چه رفتاری داشته باشد. در شکل زیر چهار رفتار را نشان می دهد.



در شکل زیر مقادیر پیش فرض **display** المنشآت را نشان می دهد.

## Common Block Elements

- Paragraphs (<p>)
- Headers (<h1> through <h6>)
- Divisions (<div>)
- Lists and list items (<ol>, <ul>, and <li>)
- Forms (<form>)

## Common Inline Elements

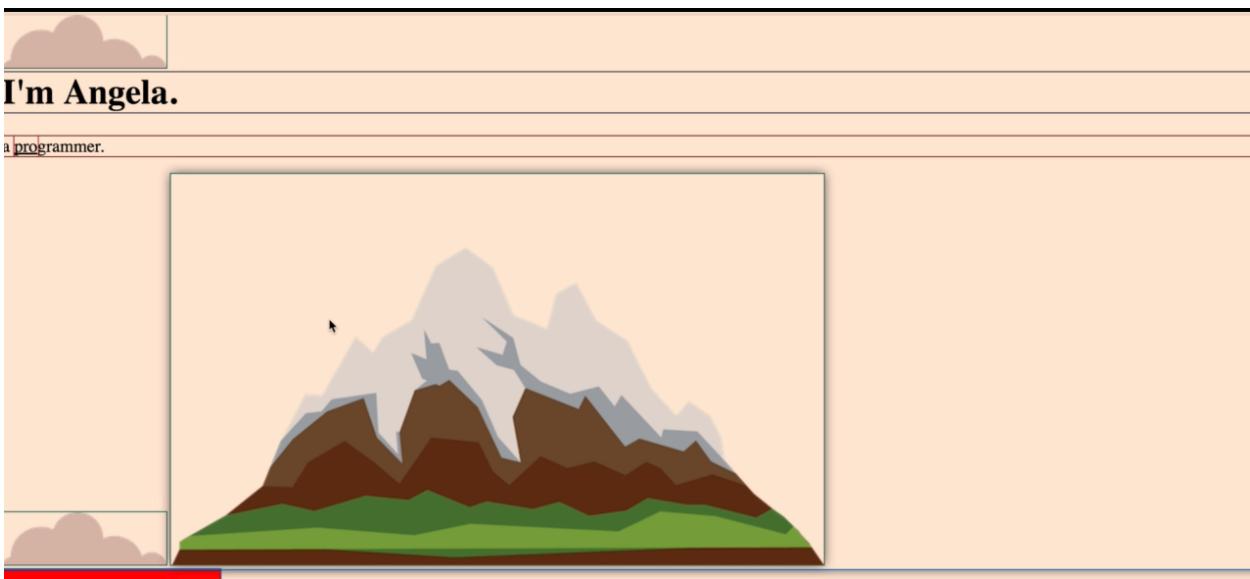
- Spans (<span>)
- Images (<img>)
- Anchors (<a>)

- می توان مقادیر پیش فرض را تغییر داد.

```
* CSS
1 p {
2     background-color: red;
3     width: 100px;
4     display: inline;
5 }
6
7 span {
8     display: block;
9     background-color: blue;
10    width: 100px;
11 }
```

- حالت inline دارای سایز متناسب با محتوا می باشد و خاصیت چسبندگی با یکدیگر دارند و امکان تغییر سایز به اندازه دلخواه باکس وجود ندارد، ولی حالت block سایز ثابت دارد و عرضش کل صفحه را می گیرد و اجازه نمی دهد چیز دیگری در کنارش بچسبد هر چند امکان تغییر سایز به اندازه دلخواه باکس وجود دارد. حالت inline-block ترکیبی از این دو می باشد. حالت none باعث ناپدید شدن المنت می شود و آن را مخفی می کند.

- نکته خاص: img ها با اینکه حالت inline دارند ولی دارای رفتار inline-block می باشند.

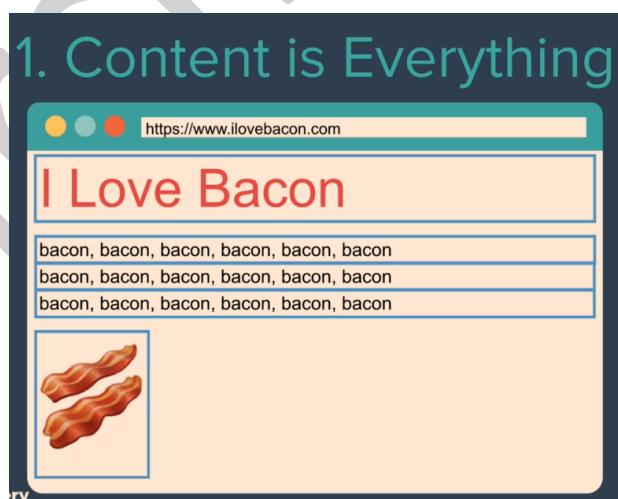


**Key words:** Display modes, display defaults, block, inline, inline-block, none, positions, span vs p elements , none vs visibility.

## 7. CSS Static and Relative Positioning

در این قسمت مفاهیم جدید position در CSS توضیح داده می شود.

- در حالت پیش فرض کدامان از html پیروی می کند که مفاهیم زیر در html برای position باید در نظر بگیریم که در واقع ساختار و اصول اولیه مان می باشد.

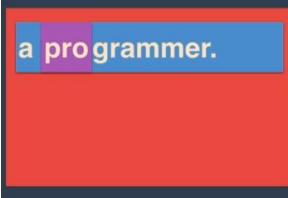


## 2. Order Comes From Code



```
<h1> </h1>
<p> </p>
<p> </p>
<p> </p>
<p> </p>
<img>
```

## 3. Children Sit On Parents

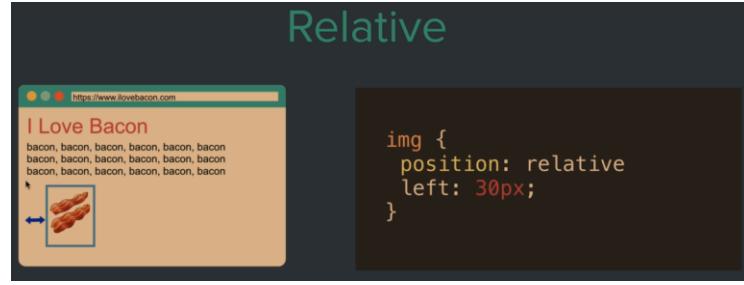


```
<div>
<h1>a <span>pro</span>grammer</h1>
</div>
```

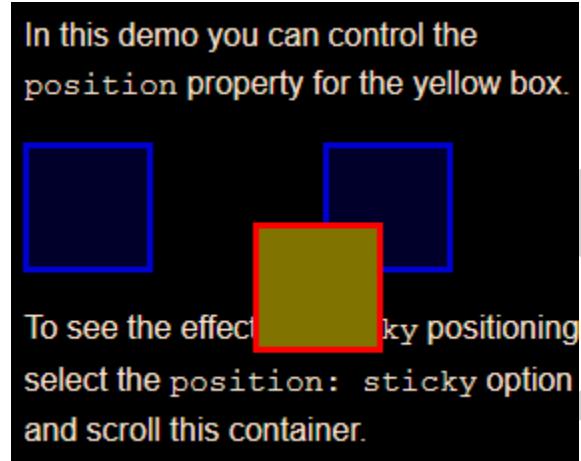
- بعد از مفهوم اولیه Display mode ، مفهوم جدید و متفاوتی به نام Position در CSS را داریم که به شکل زیر می باشد.

- در حالت پیش فرض position: Static می باشد، یعنی Position از ساختار اصلی html و position display mode پیروی می کند. حال در صورت نیاز برای تغییرات موقعیت مکانی بیشتر از coordinates در CSS استفاده می کنیم.





- شکل زیر حالت **relative** را نشان می دهد، که جای خالی خود را حفظ می کند.

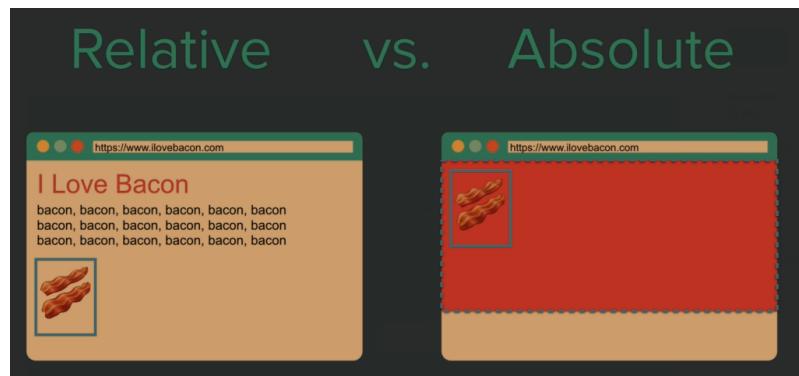


**Key words:** html position, css positions, static and relative positions, new concepts.

## 8. Absolute positioning

این قسمت، ادامه بحث **position** در CSS می باشد که به معرفی **absolute** و **fixed** می پردازد.



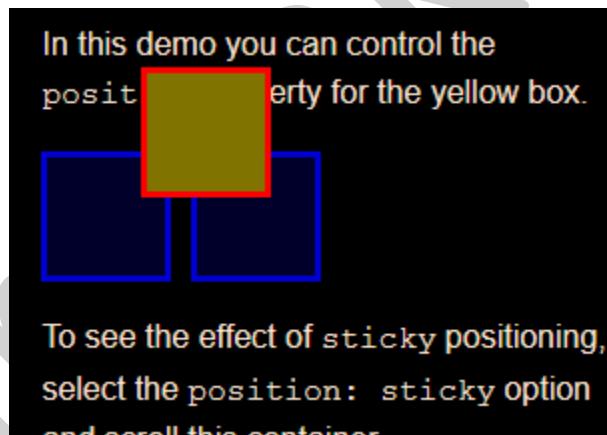


- در Relative مختصات یا coordinates داده شده نسبت به جایگاه قبلی المنشی، به موقعیت های مختلف جا به جا می شود اما در Absolute مختصات داده شده مقدار ثابتی نسبت به اش parent خواهد داشت. (در حالت پیش فرض body به عنوان parent می باشد، در صورت نیاز به تغییر parent)

جهت مقدار دهنده برای relative absolute باید fixed تعريف کنیم)

- در fixed المنشی در یک مختصات ثابت و تغییر ناپذیر خواهد ماند، یعنی وقتی صفحه را scroll کنیم، آن المنشی در جای تعیین شده ثابت خواهد ماند.

- شکل زیر حالت absolute را نشان می دهد.

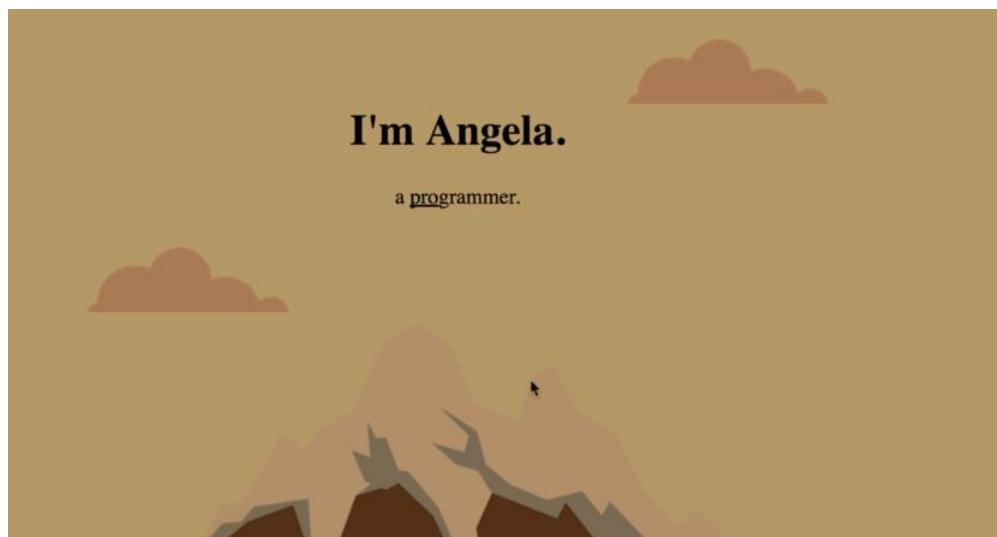


- برای فهم بیشتر position ها در CSS، حتماً در گوگل مطالعه کنید.

**Key words:** css position, absolute and fixed positions, parents, new concepts.

## 9. The Dark Art of Centering Elements with CSS

در این قسمت از position برای مشخص کردن مکان جدید بعضی از المنشی ها و مرتب کردن المنشی ها نشان داده می شود. در واقع برای خارج کردن از ساختار html، باید از position در CSS استفاده کنیم.



**Ker words:** text-align, centering elements, using positions for more.

## 10. Font Styling in Our Personal Site

در این قسمت درباره انواع فونت و نحوه تنظیم کردن آن به عنوان پیش فرض و محبوب ترین و سازگارترین فونت‌ها صحبت می‌شود.

- در انگلیسی ۲ گروه فونت لبه دار و بی لبه زیر را داریم. (در فارسی متفاوت می‌باشد)

### Difference Between Serif and Sans-serif Fonts



- در حالت پیش فرض، سیستم دارای فونت‌های زیر می‌باشد.

```
body {  
    margin: 0;  
    text-align: center;  
    font-family:  
}  
v cursive  
v fantasy  
v inherit  
v monospace  
v sans-serif  
v serif
```

The screenshot shows a portion of a CSS file with a dropdown menu open over the 'font-family' declaration. The menu lists several font families: cursive, fantasy, inherit, monospace, sans-serif, and serif. The 'sans-serif' option is highlighted.

- فونت ها از سیستم و مرورگر Client نمایش داده می شود بنابراین باید از سازگارترین فونت ها در پروژه استفاده کرد. ( فونت Arial از خانواده Sans-serif سازگار ترین فونت بین همه سیستم ها می باشد)
- در شکل زیر در قسمت CSS الیت فونت ها را مشخص کردیم تا در صورت عدم سازگاری از فونت های بعدی به ترتیب استفاده شود.

```
body {  
    margin: 0;  
    text-align: center;  
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif  
}
```

- در نهایت راه میانبر برای استفاده از فونت های دلخواه و قابلیت اجرا در همه سیستم ها ، استفاده از سایت های خدمات فونت مانند fonts.google.com می باشد، که می توان آن را در پروژه اعمال نمود.

```
<link href="https://fonts.googleapis.com/css?family=Merrriweather|Montserrat|Sacramento" rel="stylesheet">
```

```
margin: 0;
text-align: center;
font-family: 'Merriweather', serif;
}

h1 {
margin-top: 0;
font-family: 'Sacramento', cursive;
}

h2 {
font-family: 'Montserrat', sans-serif;
}

h3 {
font-family: 'Montserrat', sans-serif;
}
```

**Key words:** font-family, different fonts, set font for site, google font.

## 12. Adding Content to Our Website

در این قسمت باید محتوای بیشتر و دلخواه را وارد سایت کنیم که در ادامه آن را با CSS زیباتر و حرفه ای تر کنیم.

**Key words:** addin content, Icon, image, Lorem Ipsum text.

## 13. CSS Sizing

در این قسمت درباره تغییر سایز فونت و انواع مقیاس های سایز فونت صحبت می شود.

- برای اینکه سایز فونت استاتیک یا ثابت داشته باشیم از **Px** یا پیکسل (معادل pt در ورد) استفاده می کنیم.
- برای حالت داینامیک، یعنی با تغییر سایز فونت مرورگر، اندازه فونت به مقدار مناسب تغییر کند، از **em** استفاده می کنیم.

- در واقع  $1em=100\% = 16Px$  می باشد و از همین رابطه می توان مقادیر دلخواه **em** را بدست آورد.
- در CSS3 واحد **Root em** یا **rem** وجود دارد، که بدون وابستگی به سایز فونت **parent**، مقدار سایز تغییر می کند.

- در حالت کلی بهتر است از **rem** استفاده شود. چون مستقل عمل می کند و قابل اعتماد تر می باشد و از بقیه موارد گفته شده مناسب تر می باشد.

**Key words:** font size, em, rem, dynamic and static font size.

## 17. CSS Font Property Challenge Solutions

در این قسمت یکسری خصوصیات بیشتر برای فونت ها را نشان می دهد.

**Key word:** font-weight, line-height, font color.

## 18. CSS Float and Clear

در این قسمت ساختار پروژه را به شکل دلخواه تغییر می دهد.

- توسط float می توان شئ مورد نظر را در کنار شئ دیگر قرار داد. مثلاً برای یک تصویر می توان با سمت راست تصویر را خالی کرد تا متن در کنار تصویر قرار گیرد.
- عمل Clear عکس float می باشد، مثلاً Clear: right یعنی سمت راست شئ مورد نظر هیچ چیز نباشد.
- بهتر است در این موارد برای نظم بیشتر از clear و float استفاده کنید و از positions نکنید. در واقع کاربردها را درست استفاده کنید.
- به بحث وراثت المنش ها بیشتر توجه کنید. مثلاً body تشكیل دهنده 100% تصویر می باشد، حال وقتی div را با 50% تغییر می دهیم، در واقع ۵۰ درصد والد خود را می گیرد و به همین ترتیب برای سایر قضیه ها نیز صادق است.



**Key words:** Float, Clear, align the content.

## 20. Stylised Personal Site Solution Walkthrough

در این قسمت پروژه را کامل می کند، و به صورت نهایی و دلخواه در می آورد.

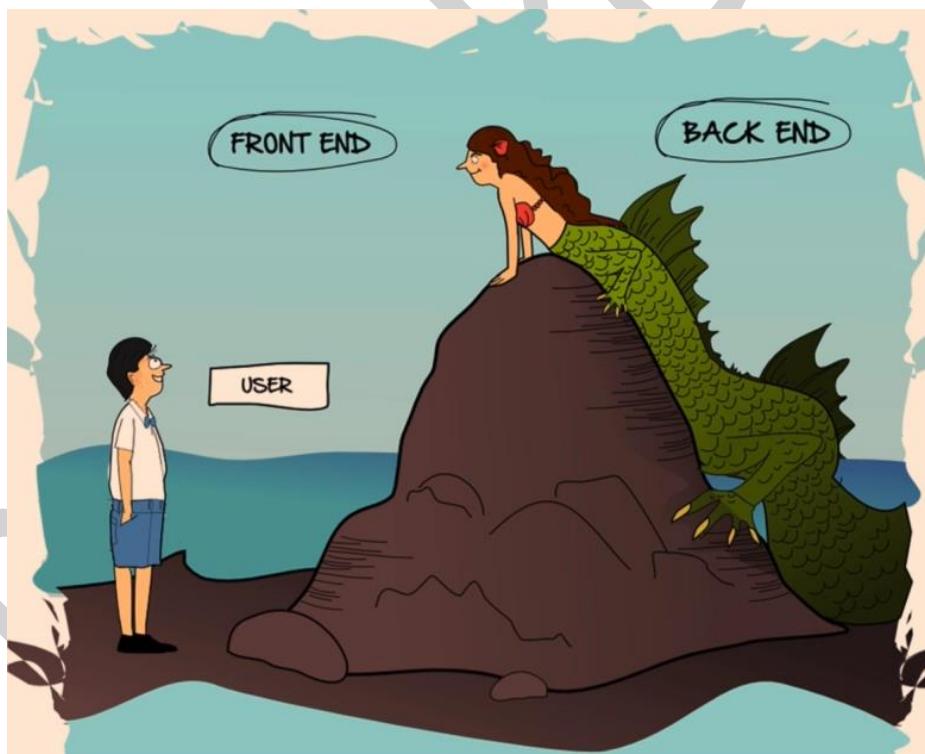
## 22. Tip from Angela - Nothing Easy is Worth Doing!

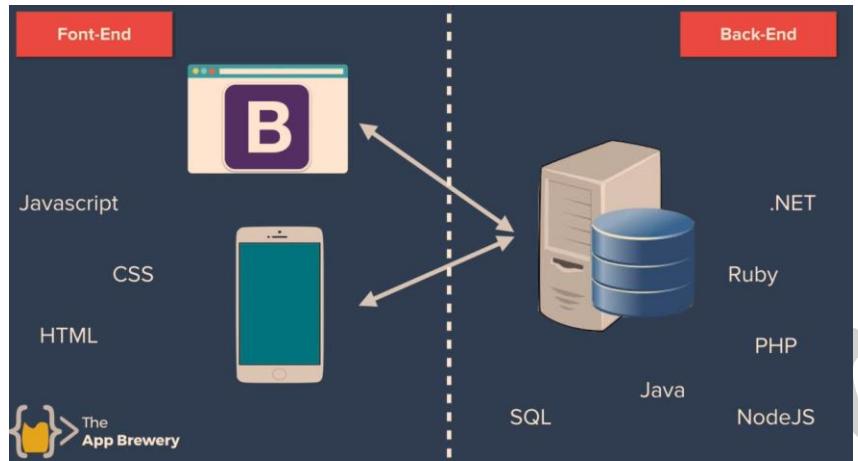
هیچ پاداشی ارزش و لذتی ندارد، مگر برای بدست آوردن آن زحمت کشیده باشد.

## 6. Introduction to Bootstrap 4

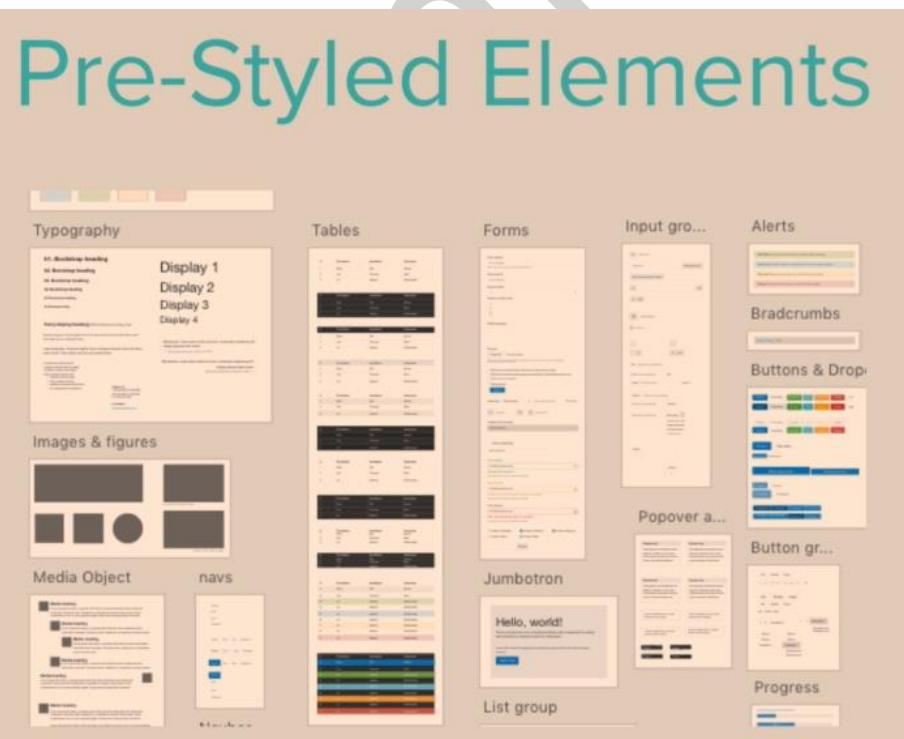
### 1. What is Bootstrap

در این قسمت درباره front end و back end صحبت می شود و به معرفی و کاربرد Bootstrap می پردازد.





- front-end در واقع یک framework یا library می باشد که از مجموعه کدهای bootstrap ، CSS ، HTML و javascript تشکیل شده است که از این کدها کلاس های جدید مختلفی تعریف شده است. مثلا button ها با رنگ های مختلف ، یا جداول با استایل های مختلف و زیبا و یا حتی سایت های از پیش تعیین شده برای کاربرد های مختلف ( مثل سایت فروشگاه، خبری و...) در bootstrap وجود دارد که کافیست از آنها در پروژه مان استفاده کنیم و زیبایی دوچندان به اعمال front-end کنیم.



- اما نکته مهم تر این است که، استفاده از front-end Responsive باعث شدن Bootstrap می شود، یعنی متناسب با کاربر (موبایل، دسکتاپ و...) صفحه نمایش adopt می شود که این مزیت بزرگی در دنیای امروز با توجه به حضور موبایل ها دارد.



- بوت استرپ به توسعه دهندهان کمک می کند تا با سرعت بسیار بالا و با استانداردترین حالت ممکن بخش ظاهری یا فرانت سایت خود را آماده نمایند و هم اکنون bootstrap محبوب ترین فریم ورک موجود CSS در دنیاست.

**Key words:** Bootstrap introduction, front and back end.

## 2. Installing Bootstrap

در این قسمت نحوه نصب یا فراخوانی کتابخانه bootstrap به پروژه توضیح داده می شود.

- بهترین کار برای نصب bootstrap در هر پروژه ای، استفاده از لینک مخصوص bootstrap می باشد که در ابتدای پروژه تعریف می شود. این لینک خاصیت Content Delivery Network یا CDN را دارد، یعنی فایل مورد نظر در سرورهای مختلف در کل جهان وجود دارد، در نتیجه سرعت دانلود فایل bootstrap بالا رفته و صفحه سایت سریع تر لود می شود. (هر چند به دلیل مراجعه کاربران به سایتها مختلف، معمولاً این فایل ها در کش مرورگر موجود می باشد)

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5Bf9tH/B/KMkx7oJyJGzZ7W1wUj7uK4uKuXqCewyRr4gXyvDQmHdG3Xc" crossorigin="anonymous">
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-KJ3o2DKtIkVYIK3L" crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScFdeoEjEqKJqsU5MfKtC1FfTC+g9p" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U024aKJb0TjNPabZPlD8dgacxDichaHUhGzIKw0WZgUPWq" crossorigin="anonymous"></script>
  </body>
</html>
```

- در حالت کلی برای پیکربندی بر اساس bootstrap به لینک <https://getbootstrap.com/docs> مراجعه کنید و از راهنمای آن استفاده کنید.  
- روش دیگر نصب، دانلود فایل bootstrap و قرار دادن به عنوان فولدر محلی در پروژه، که این روش توصیه نمی شود. (چون مرورگر آن را به عنوان فایل جدید دانلود می کند و سرعت کاهش می یابد)

**Key words:** bootstrap installation, CDN, network speed.

### 3. Web Design 101 – Wireframing

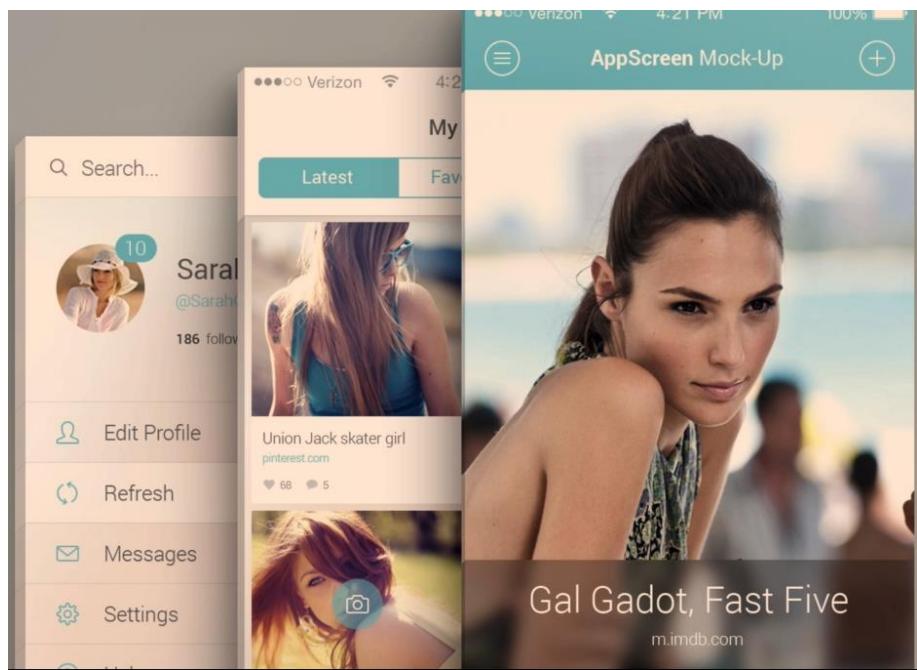
در این قسمت درباره انواع طراحی سایت قبل از شروع کردن به کدزنی پروژه صحبت می شود.

- بايد قبل از شروع کدزنی سایت یا اپلیکیشن باید ظاهر آن را طراحی کنیم و چینش قرارگیری اشیاء را مشخص کنیم. مانند طراحی ساختمان ، قبل از ساختن آن.

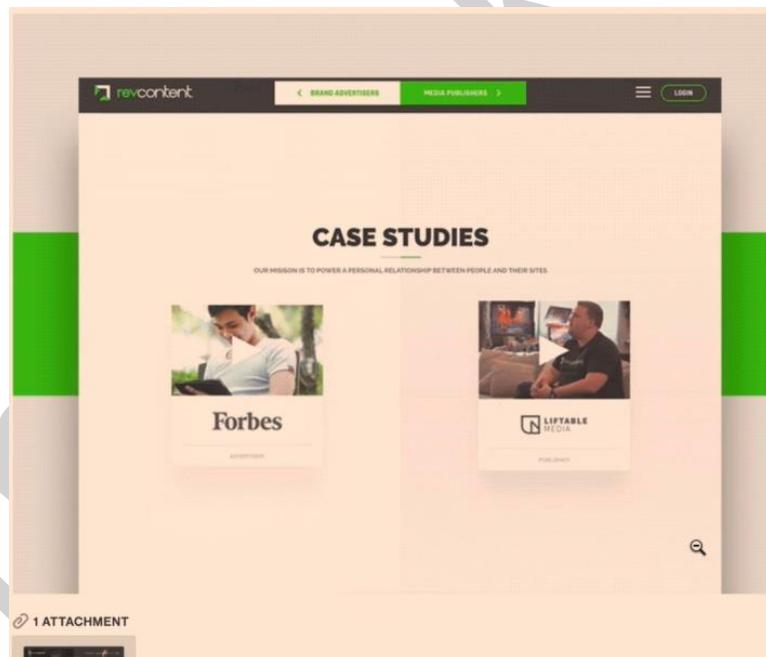
- سریع ترین و متداول ترین طراحی، طراحی wireframing می باشد که به صورت ساده و با یک مداد و صفحه خالی موارد نیاز را پیاده می کنیم. همچنین می توان از سایت های طراحی نیز استفاده کرد.



- طراحی بعدی به روش mock-up می باشد، که توسط نرم افزارهای گرافیکی مانند فتوشاپ جزئیات سایت را ایجاد می کنیم. (روش اختیاری و در صورت نیاز استفاده می شود)



- طراحی آخر روش **prototype** می باشد، که در واقع یک انیمیشن یا فیلم از خروجی مورد نظر که قرار است ایجاد کنیم، می سازیم. (در بعضی پروژه ها از این طراحی استفاده می شود)



- ولی در نهایت طراحی **wifeframing** مهم ترین طراحی هر پروژه می باشد.



- به سایت های طراحی معرفی شده در این بخش مراجعه کنید.
- با استفاده از کلید میانبر / Ctrl + می توان خط کد را تبدیل به کامنت نمود.

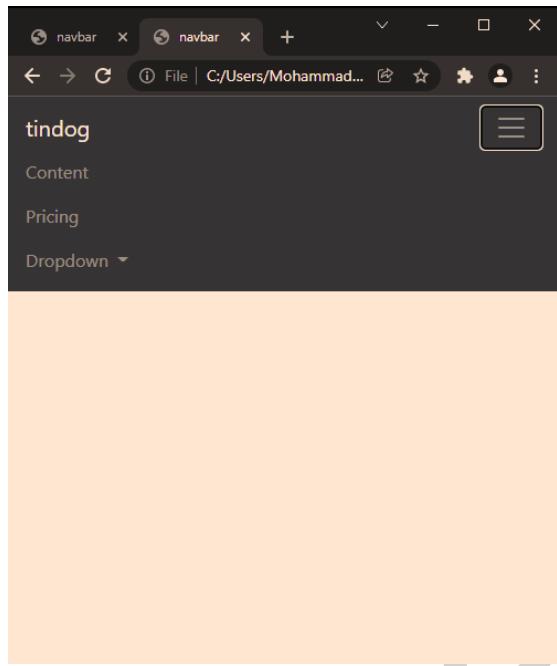
**Key words:** Web design, wireframing, mock-up, prototype, UI design.

#### 4. The Bootstrap Navigation Bar

در این قسمت به معرفی المنت nav یا navbar می پردازد که توسط کتابخانه های bootstrap آن را حرفه ای می کند.

- در این قسمت از کلاس های مختلف nav در bootstrap استفاده می کند تا قسمت بالای سایت طراحی شده را ایجاد کند.

```
1- <nav class="navbar navbar-expand-lg navbar-light bg-danger">
2-   <ul class="navbar-nav">
3-     <li class="nav-item">
4-       <a class="nav-link" href="">Contact</a>
5-     </li>
6-     <li class="nav-item">
7-       <a class="nav-link" href="">Pricing</a>
8-     </li>
9-     <li class="nav-item">
10-       <a class="nav-link" href="">Download</a>
11-     </li>
12-   </ul>
13- </nav>
```



- می توان کلاس ها را به گونه ای تغییر داد، که با تغییر ابعاد از دسکتاپ به گوشی، گزینه toggle یا همبرگر ایجاد شود.
- معمولا از anchor و list در اجزای nav استفاده می شود. در واقع nav وظیفه جابه جایی در جاهای مختلف یک صفحه را دارد. مثلاً می توان از بالای صفحه به وسط صفحه با یک کلیک جا به جا شد.
- تمام اطلاعات از اسناد bootstrap در سایت <https://getbootstrap.com/docs/4.0/components/navbar> موجود می باشد.

Key words: navbar element, making navbar-toggler, changing navbar with Bootstrap frameworks, navbar-expand-lg, navbar-dark, bg-dark, dropdown.

## 5. What We'll Make TinDog

در این قسمت درباره سایتی که قرار است در این فصل درست شود نشان داده خواهد شد (mock-up). در واقع یک سایت یک صفحه ای استارتاپی می باشد که در آن bootstrap و CSS advanced را در طول این فصل یاد خواهیم گرفت.

## 7. Setting Up Our New Project

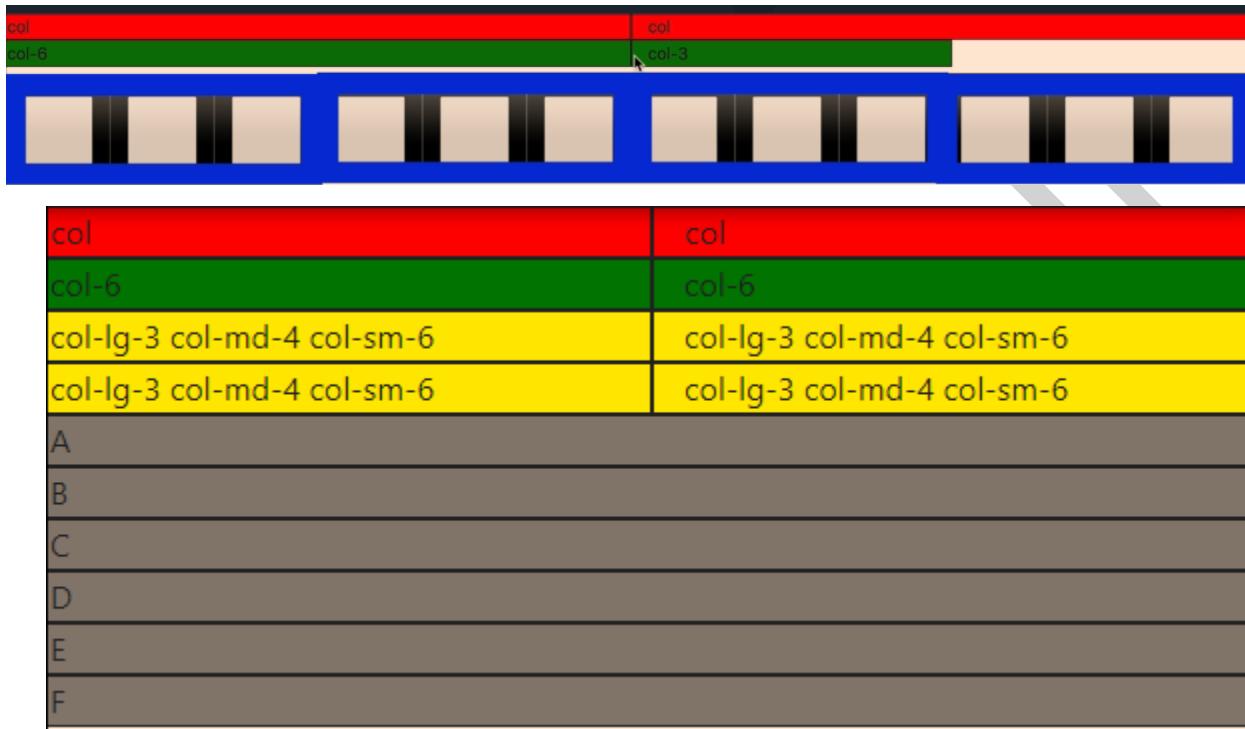
در این قسمت نحوه پیکربندی اولیه پروژه Tindog را توضیح می دهد و نحوه معرفی کتابخانه bootstrap و java script را نشان می دهد.

Key words: adding bootstrap framework.

## 8. Bootstrap Grid Layout System

در این قسمت به معرفی Grid layout system می پردازد که یکی از ویژگی های در responsive می باشد، یعنی محتوای ردیفی و ستونی ایجاد شده، متناسب با viewport تغییر می کند.

- هر row می تواند ۱۲ تا col بپذیرد. طبق همین قانون می توان Grid layout را ایجاد کرد.

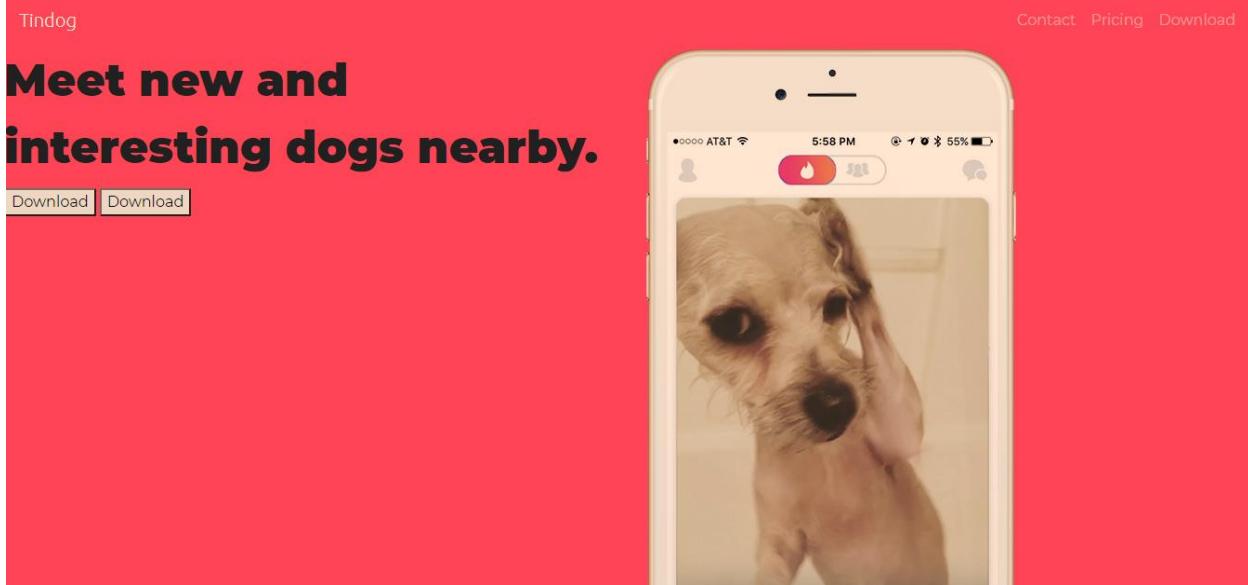


```
<div class="row">
  <div class="col-lg-3 col-md-4 col-sm-6" style="background-color:yellow; border: 1px solid">
    col-lg-3 col-md-4 col-sm-6
  </div>
  <div class="col-lg-3 col-md-4 col-sm-6" style="background-color:yellow; border: 1px solid">
    col-lg-3 col-md-4 col-sm-6
  </div>
  <div class="col-lg-3 col-md-4 col-sm-6" style="background-color:yellow; border: 1px solid">
    col-lg-3 col-md-4 col-sm-6
  </div>
  <div class="col-lg-3 col-md-4 col-sm-6" style="background-color:yellow; border: 1px solid">
    col-lg-3 col-md-4 col-sm-6
  </div>
</div>
```

**Key words:** Grid Layout, making responsive table, responsive viewport, row and col classes.

## 10. Adding Grid Layouts to Our Website

در این قسمت grid layout در عمل پیاده سازی شد و یکسری تمرینات انجام شد.



**Key words:** grid layout, change fonts, sections.

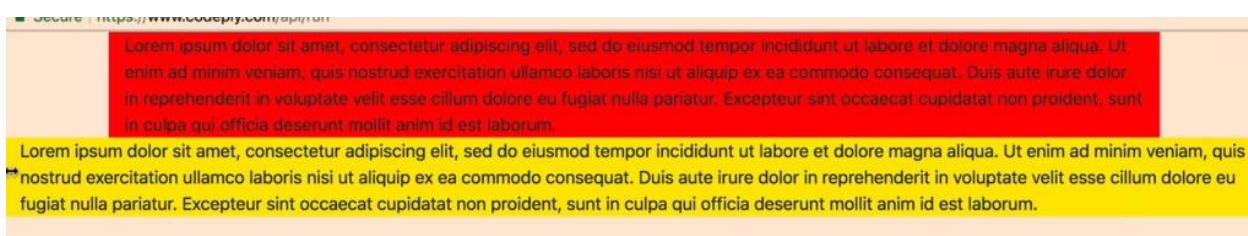
## 12. Bootstrap Containers

در این قسمت درباره container در bootstrap صحبت می شود. Container یک کلاسی می باشد که در واقع می توان گروهی از component ها در آن قرار داد. مثلاً container همه اجزا را به صورت خودکار در وسط صفحه نگه می دارد و با تغییر viewport همه اجزا در وسط صفحه خواهند بود و به صورت خودکار محتوا را جایه جا می کند.

در شکل زیر تفاوت container و container-fluid را نشان می دهد. محتوای هر دو متناسب با تغییر می کند. container دارای مارجین متغیر می باشد ولی container-fluid دارای مارجین ثابت.

```

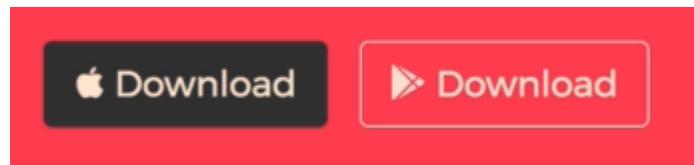
1- <div class="container" style="background-color: red;">
2-   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
      cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
3- </div>
4-
5- <div class="container-fluid" style="background-color: red;">
6-   Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco
      laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
      cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
7- </div>
```



**Key words:** container, container-fluid.

### 13. Bootstrap Buttons & Font Awesome

در این قسمت از button های موجود در bootstrap استفاده می شود و همچنین Icon های مورد نظر از سایت font awesome وارد پروژه می شود.

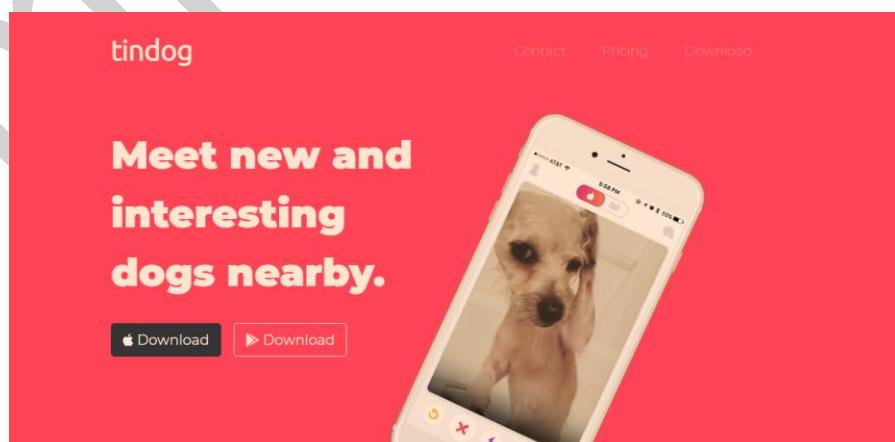


**Key words:** stylish button, Font Awesome site, i or span for Insert icons in wild.

### 14. Styling Our Website Challenges and Solutions

در این قسمت تغییرات style را برای جزئیات بیشتر و تغییرات دلخواه، توسط کدهای مستقیم CSS به پروژه اعمال می کنیم. در واقع در هر پروژه بعد از استفاده از Bootstrap باید با استفاده از CSS جزئیات کار را زیباتر نمود.

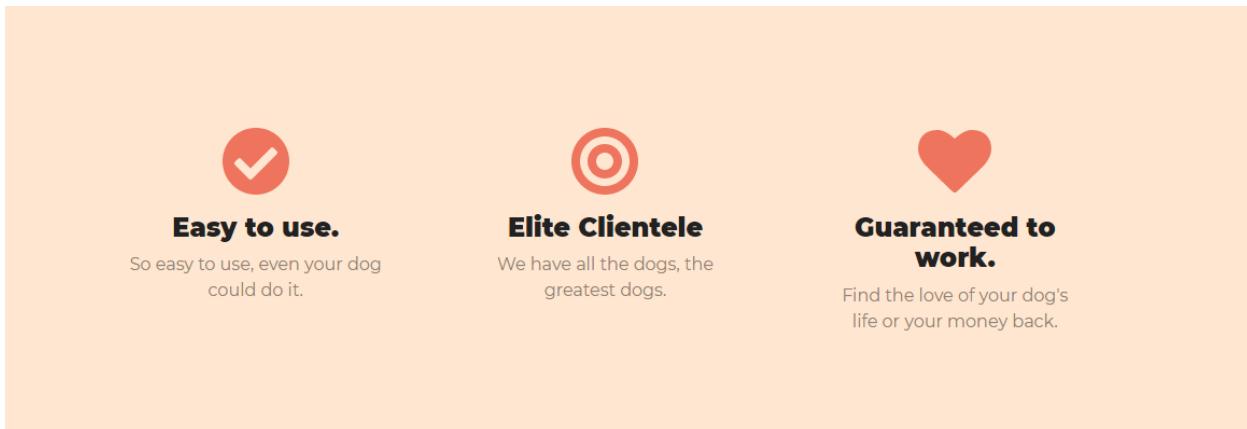
- در طراحی رابط کاربری سعی کنید، کل اجرا در یک خط عمودی قرار گیرند و کمتر از تو رفتگی استفاده کنید.
- در طراحی رابط کاربری، به واسطه سایز و حجم فونت ها می توان هدف مورد نظر را به کاربر رساند. مثلاً در شکل زیر ابتدا متن، سپس دکمه ها و در نهایت لوگو به چشم کاربر خواهد خورد.



**Key words:** CSS styling, UI concepts, beautifying.

## 16. Solution to Bootstrap Challenge 1

در این قسمت به حل چالش محول شده پرداخت، و قسمت دیگری از پروژه را تکمیل نمود.



- برای اضافه کردن تغییرات CSS جدید، از کلاس های bootstrap استفاده نکنید و کلاس های جدید ایجاد کنید.

**Key words:** designing UI, nice padding and margin, bootstrap components.

## 17. Tip from Angela - How to Deal with Procrastination

نکات انگیزه ای پایان فصل: از تکنیک پومودورو استفاده کنید، و بعد از هر پومودورو به خود جایزه دهید.

## 7. Intermediate Bootstrap

### 1. The Bootstrap Carousel Part 1

در این قسمت به تکمیل کردن پروژه و زیباسازی section های بعدی پروژه انجام می شود.



- توسط کلید میانبر Alt + mouse selection می توان توسط افزونه sublime-style ، به صورت کلی Text را تایپ کرد.

```

2 -   <div class="carousel-inner">
3 -     <div class="carousel-item active" style="background-color: red;">
4 -       
5 -     </div>
6 -     <div class="carousel-item" style="background-color: yellow;">
7 -       
8 -     </div>
9 -     <div class="carousel-item" style="background-color: blue;">
10 -       
11 -     </div>
12 -   </div>
13 - </div>
```

Name	Type	Default	Description
interval	number	5000	The amount of time to delay between automatically cycling an item. If false, carousel will not automatically cycle.
keyboard	boolean	true	Whether the carousel should react to keyboard events.
pause	string   boolean	"hover"	If set to "hover", pauses the cycling of the carousel on <code>mouseenter</code> and resumes the cycling of the carousel on <code>mouseleave</code> . If set to <code>false</code> , hovering over the carousel won't pause it.  On touch-enabled devices, when set to "hover", cycling will pause on <code>touchend</code> (once the user finished interacting with the carousel) for two intervals, before automatically resuming. Note that this is in addition to the above mouse behavior.
ride	string	false	Autoplays the carousel after the user manually cycles the first item. If "carousel", autoplays the carousel on load.
wrap	boolean	true	Whether the carousel should cycle continuously or have hard stops.

**Key words:** carousel component, adding carousel buttons.

### 3. Bootstrap Cards

در این قسمت از bootstrap Cards در پروژه استفاده می شود.

**A Plan for Every Dog's Needs**

Simple and affordable price plans for your and your dog.

Dog Breed	Price	Features
Chihuahua	<b>Free</b>	5 Matches Per Day 10 Messages Per Day Unlimited App Usage
Labrador	<b>\$49 / mo</b>	Unlimited Matches Unlimited Messages Unlimited App Usage
Mastiff	<b>\$99 / mo</b>	Priority Listing Unlimited Matches Unlimited Messages Unlimited App Usage

**Key words:** Bootstrap Cards.

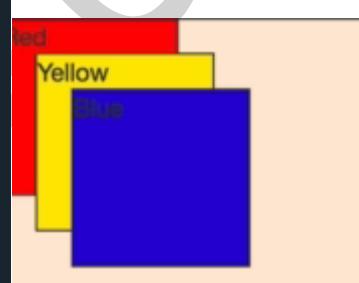
### 4. The CSS Z-Index and Stacking Order

در این قسمت درباره Stacking order یا همان اولویت ترتیب نمایش المنت صحبت می شود.

- یک المنت علاوه بر موقعیت مکانی  $y$ ,  $x$  دارای موقعیت مکانی  $z$  یا Stacking order می باشد.

- مقدار z یا z-index در حالت عادی برابر صفر و از ساختار html پیروی می کند، حال برای تغییر stacking خارج از html ، باید ۲ شرط CSS تغییر position (fixed, absolute, relative ) و تغییر مقدار order را اعمال کنیم.

- از فلوچارت تهیه شده این قسمت برای درک بیشتر استفاده کنید.



```
<style CSS>
1 - div {
2     height: 100px;
3     width: 100px;
4     border: 1px solid;
5     position: absolute;
6 }
7
8 - .red {
9     background-color: red;
10    z-index: -1;
11 }
12
13 - .yellow {
14     background-color: yellow;
15     left: 20px;
16     top: 20px;
17     z-index: -1;
18 }
19
20 - .blue {
21     background-color: blue;
22     left: 40px;
23     top: 40px;
24     z-index: -1;
25 }
```

**Key words:** CSS Z-Index, Stacking order, change positions, new rule and concepts.

## 5. Media Query Breakpoints

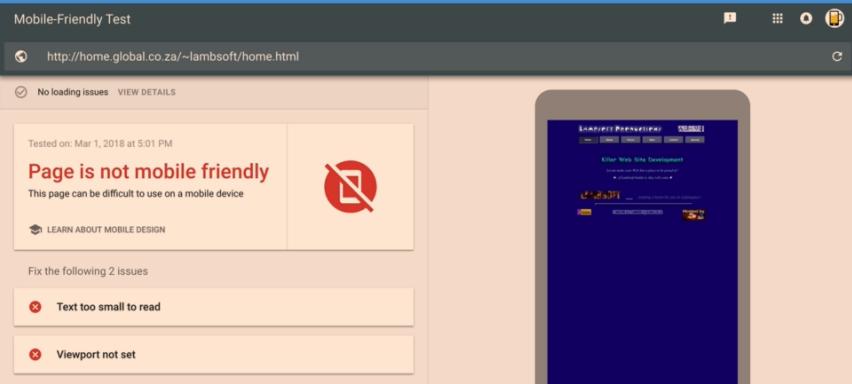
در این قسمت به معرفی CSS Media Query پرداخته می شود که برای responsive بودن بیشتر، به خصوص در موبایل ها، از آن استفاده می کنیم.

- وب گردی با موبایل بیشتر از دسکتاپ می باشد، بنابراین باید سایت ها mobile friendly باشند، تا هم کاربرد راحت باشد و هم در موتور گوگل دارای امتیاز باشد.
- باعث responsive شدن می شود، ولی برای جزئیات بیشتر در Bootstrap نیاز به استفاده از Media Query داریم.

# Mobile-First



## 1. Do Nothing



## 2. Make a Separate Mobile Site



Brought to you  
London App B

## 3. Make it Responsive!



Media Query يعني پرس و جو یا گذاشت شرط درست یا غلط، یعنی زمانی که شرط های Query برقرار باشد، آن کدهای CSS اعمال خواهند شد.

### Media Query

```
@media <type> <feature>
```

```
@media screen (min-width: 900px) {  
  
    //Change Something  
  
}
```

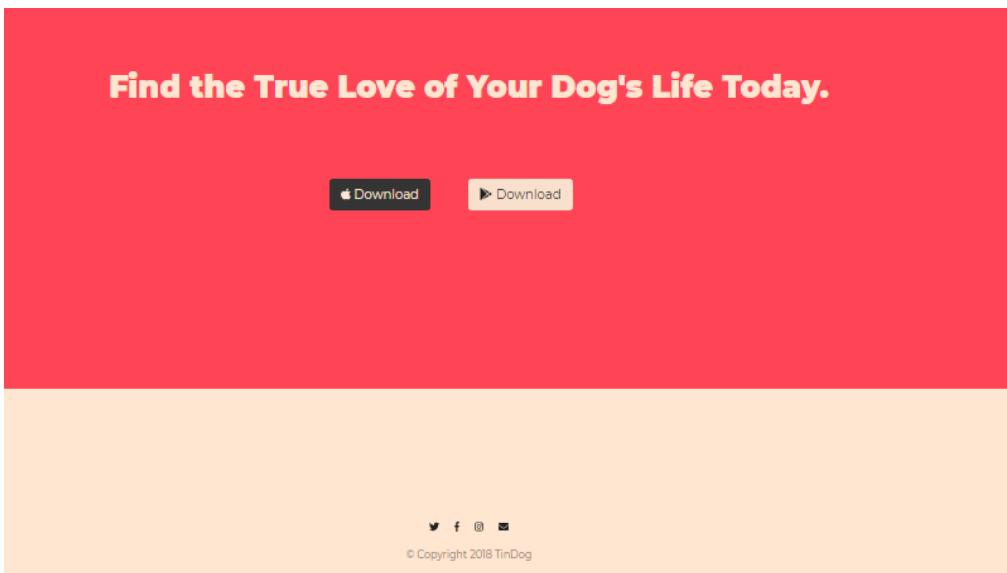
```
<style>  
1 h1 {  
2     font-size: 30px;  
3 }  
4  
5 @media (min-width: 900px) and (max-width: 1000px) {  
6  
7 h1 {  
8     font-size: 60px;  
9 }  
10 }  
11  
12 }
```

- برای طراحی UI دو روش وجود دارد، از طراحی دسکتاپ به طراحی موبایل بررسیم و یا از طراحی موبایل به دسکتاپ بررسید. درست و غلطی وجود ندارد، با هر کدام که راحتی‌تر طراحی مورد نظر را انجام دهید.

**Key words:** CSS Media Query, more responsive, mobile-friendly, new concepts.

## 7. Bootstrap Challenge 2 Solution

در این قسمت به حل و تکمیل قسمت های پایین صفحه می پردازد و همچنین از id Sections برای آدرس دهی استفاده می کند(استفاده مناسب از id در پروژه)



**Key words:** using id in href, complete footer and bottom sections.

## 8. How to become a Better Programmer - Code Refactoring

در این قسمت درباره بازنویسی کد جهت بهینه شدن کد صحبت می شود، یعنی چه قواعدی را رعایت کنیم که کدی بهینه و استاندارد داشته باشیم:

- به ترتیب اولویت خوانا بودن کد (یعنی استفاده از نام گذاری درست متغیرها و کامنت گذاری) و مازولات بودن (هر چه بیشتر مازولات باشد، راحت تر می توان باگ را برطرف نمود) از اهمیت بالایی برخوردار هستند.
- در ادامه کارآمدی (سرعت پردازش) و مقدار کد (طول خطوط کد) اهمیت دارند (البته نباید به ۲ مورد اول لطمه ای بزند).

# Code Refactoring

- 
1. Readability
  2. Modularity
  3. Efficiency
  4. Length

**Key words:** Code Refactoring, DRY: Don't Repeat Yourself, WET: We Enjoy Type, axe murder.

## 9. Put it into Practice - Refactor our Website Part 1

در این قسمت کدهای نوشته شده پروژه را در عمل Refactor می کند.

- در حالت کلی باید از ابتدای نوشتگی کد بهینه سازی کد با ۴ مؤلفه ذکر شده، هماهنگ باشد تا نیاز به refactoring مجدد نباشد.
- توسط کلید F CTRL+ F می توان در کدهای نوشته شده جستجو کرد.
- به کلاس ها و ای دی های موجود در CSS انتخاب گر یا Selector گفته می شود.
- در شکل زیر combine selector را نشان می دهد، که می توان ویژگی های یک کلاس را با ویژگی های جدید جایگزین نمود.

```
#title .container-fluid {  
    padding: 3% 15% 7%;  
}
```

**Key words:** beautify, Refactoring codes, combine selector, emerge repeated codes, convert to more modular code.

## 10. Advanced CSS - Combining Selectors

در این قسمت به معرفی ۳ روش ترکیب کردن Selector ها در CSS توضیح داده می شود.

- می توان به روش های زیر عمل combine کردن را انجام داد که هر کدام رفتار خود را دارند.
- در multiple selector می توان خصوصیات جدید را به مجموعه ای از Selector ها اعمال کرد.

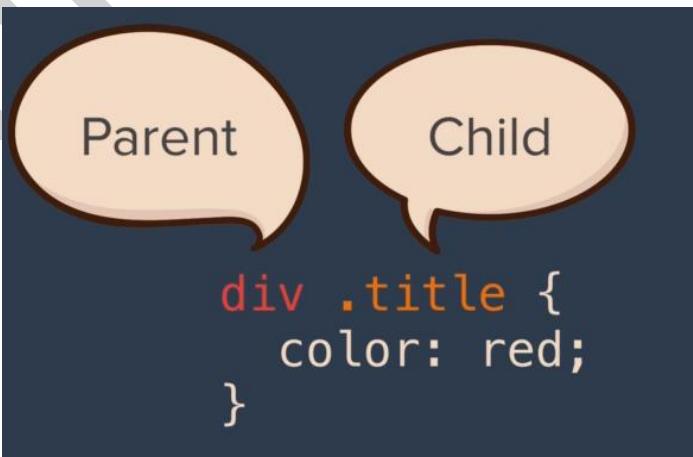
### Multiple Selectors

```
selector1, selector2 {  
}  
  
h1, h2, h3, h4, h5, h6 {  
    font-family: "Montserrat-Bold";  
}
```

در Hierarchical selector - می توان با توجه به بحث وراثت، خصوصیات را به یک گروهی از Selector ها اعمال کرد.

### Hierarchical Selectors

```
selector1 selector2 {  
}  
  
#title .container-fluid {  
    padding-top: 3%;  
    text-align: left;  
}
```



```
#title .container-fluid {  
    padding-top: 3%;  
    text-align: left;  
}
```

- در selector می توان خصوصیات را فقط به المنشت هایی اعمال کرد که آن selector -  
ها را دارا باشند.



- در hierarchical بر اساس وراثت می باشد ولی در combined بر اساس انتخاب می باشد.



- به فاصله بین selector ها توجه کنید.

**Key words:** new concepts, Selectors in CSS, Multiple Selector, Hierarchical Selector, Combined Selector.

## 11. Refactoring our Website Part 2

در این قسمت در ادامه refactoring ، رنگ بندی اجزای صفحه را مازوچاتر می کنیم.

```
.colored-section {
    background-color: #ff4c68;
    color: #fff;
}

.white-section {
    background-color: #fff;
}
```

```
#testimonials {
    padding: 7% 0;
    background-color: #ef8172;
}

.colored-section {
    background-color: #ff4c68;
    color: #fff;
}
```

**Key words:** Refactoring for page colors, Selector Priority.

## 12. Advanced CSS - Selector Priority

در این قسمت درباره اولویت selector ها در CSS صحبت می شود. به این نکته دقت کنید، با توجه به فصل ۴ قسمت ۹، حق تقدم Selector ها در CSS به ترتیب ID، Class و در نهایت html element selector (html element selector) می باشد، یعنی اگر خصوصیتی تکراری و با مقداری جدید توسط ID در CSS اضافه کنیم، مقدار جدید جایگزین مقادیر Class و Tag خواهد شد. (مانند شکل زیر) البته مقادیر Inline بالاترین اولویت در CSS را همیشه دارند.

```
<body>
1 <h1 id="heading" class="title" style="color: orange;">Hello World</h1>

h1 {
  color:red;
}

.title {
  color: yellow;
}

#heading {
  color: blue;
}
```

برای جلوگیری از ایجاد مشکل در کد نویسی توسط قوانین اولویت Selector ها موارد زیر را رعایت کنید: ۱- همیشه از class ها استفاده کنید (مگر مجبور به استفاده از ID باشید) ۲- سعی کنید برای هر المنت html از کمترین تعداد class استفاده کنید. ۳- به هیچ وجه از کدهای inline برای استفاده نکنید.

**Key words:** Selector Priority, html element selector, Id and class selectors, important rules.

## 13. Completing the Website

در این قسمت جمع بندی نهایی سایت ساخته شده را انجام می دهد و نکات تکمیلی گفته می شود.

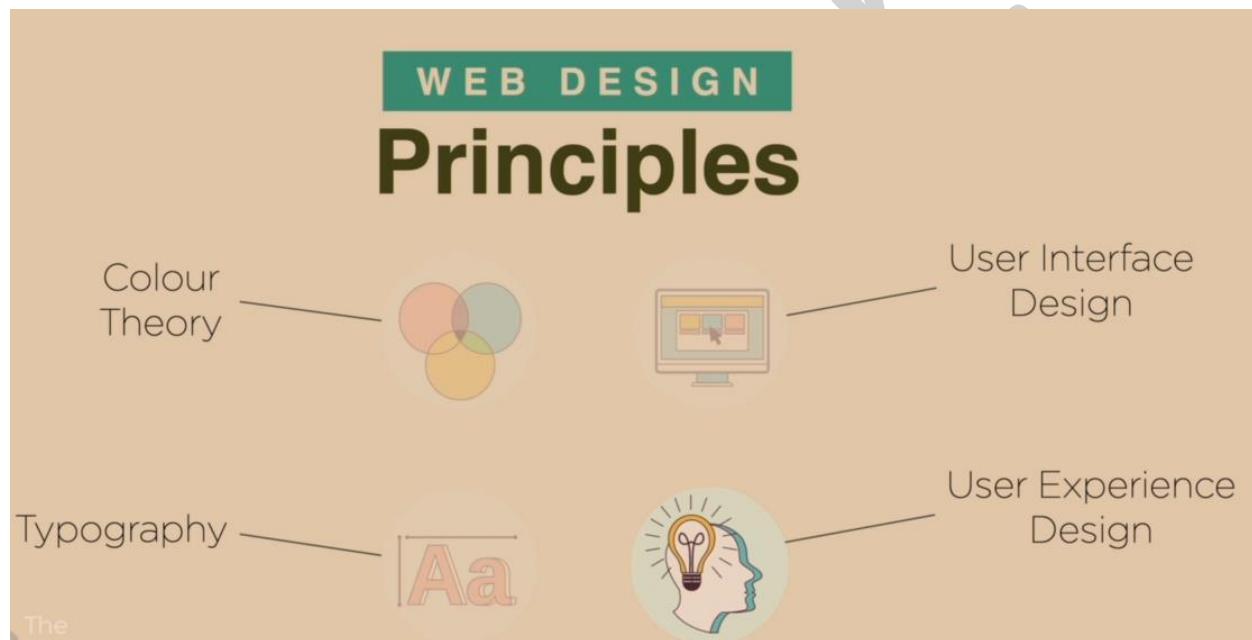
## 15. Tip from Angela - Building a Programming Habit

نکات انگیزشی پایان فصل: برای ایجاد عادات جدید مستمر و پایدار، سعی کنید عادات جدید را بعد از عادات قدیمی انجام دهید. مثلاً خود را اجبار کنید که هر صبح بعد از مسواک زدن (عادت قدیمی)، ۲۰ دقیقه مدیتیشن (عادت جدید) کنید. این گونه عادات جدید راحت‌تر شکل خواهند گرفت.

## 8. Web Design School - Create a Website that People Love

### 1. Introduction to Web Design

در این فصل به نحوه طراحی یک وب سایت کاربرپسند مطابق با قوانین و اصول مورد نظر، توضیح داده خواهد شد.



**Key words:** new concept, web Design Principles.

### 2. Understanding Colour Theory

در این قسمت به تئوری انتخاب رنگ ها و ترکیب استاندارد آنها صحبت می شود که باید در طراحی گرافیک مد نظر باشد.

*flowers*  
SHOW YOUR LOVE

# PAINTBALL

for the daring

*flowers*  
SHOW YOUR LOVE

# PAINTBALL

for the daring

- هر رنگ حالت و کاربردی دارد. در واقع باید به صورت آگاهانه و هدفمند رنگ ها را انتخاب کنیم، چون رنگ ها پیام مخصوصی یا پروژه ما را می رسانند.

## Moods

RED



LOVE. ENERGY.  
INTENSITY.

YELLOW



JOY. INTELLECT.  
ATTENTION.

GREEN



FRESHNESS. SAFETY.  
GROWTH.

BLUE



STABILITY. TRUST.  
SERENITY.

PURPLE



ROYALTY. WEALTH.  
FEMININITY.

- معمولاً برای طراحی ها از ترکیب ۲ یا ۳ رنگ استفاده می شود، که ترکیب کردن آنها، علمی ترین قسمت انتخاب رنگ ها می باشد.

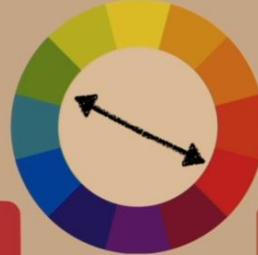


- می توان از انتخاب ۲ رنگ در همسایگی هم، در چرخ رنگ (color wheel)، یک هارمونی و هماهنگی ایجاد کرد. این انتخاب معمولاً در پس زمینه های سایت ویا navigation bar و یا برای لوگو و پس زمینه اش استفاده می شود.

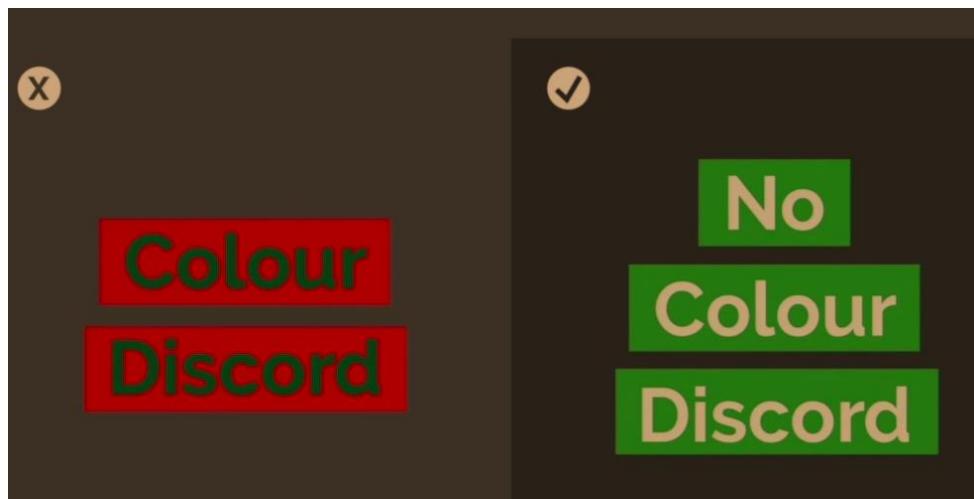


- می توان از انتخاب ۲ رنگ در جهت مخالف، ترکیب مکمل را ایجاد کرد که حالت انفجار یا pop در نگاه اول برای مخاطب ایجاد می کند و ماندگاری بالاتری در ذهن خواهد گذاشت. مثل حس تازگی قرمزی گوشت در کنار رنگ سبز.

## Complementary Colours



- از ترکیب رنگ مکمل برای آیکون و لوگوها و جاهایی که نیاز به تمایز و برجستگی بیشتر می باشد استفاده کنید و به هیچ وجه در متن و پس زمینه اش استفاده نکنید چون باعث تاری متن می شود.



- در حالت کلی می توان از روش های مختلف ترکیب رنگ استفاده کرد. همچنین می توان از سایت های مورد نظر ترکیبات رنگ محبوب و مورد نظر را انتخاب کرد.



**Key words:** select appropriate color, color combination, new concepts.

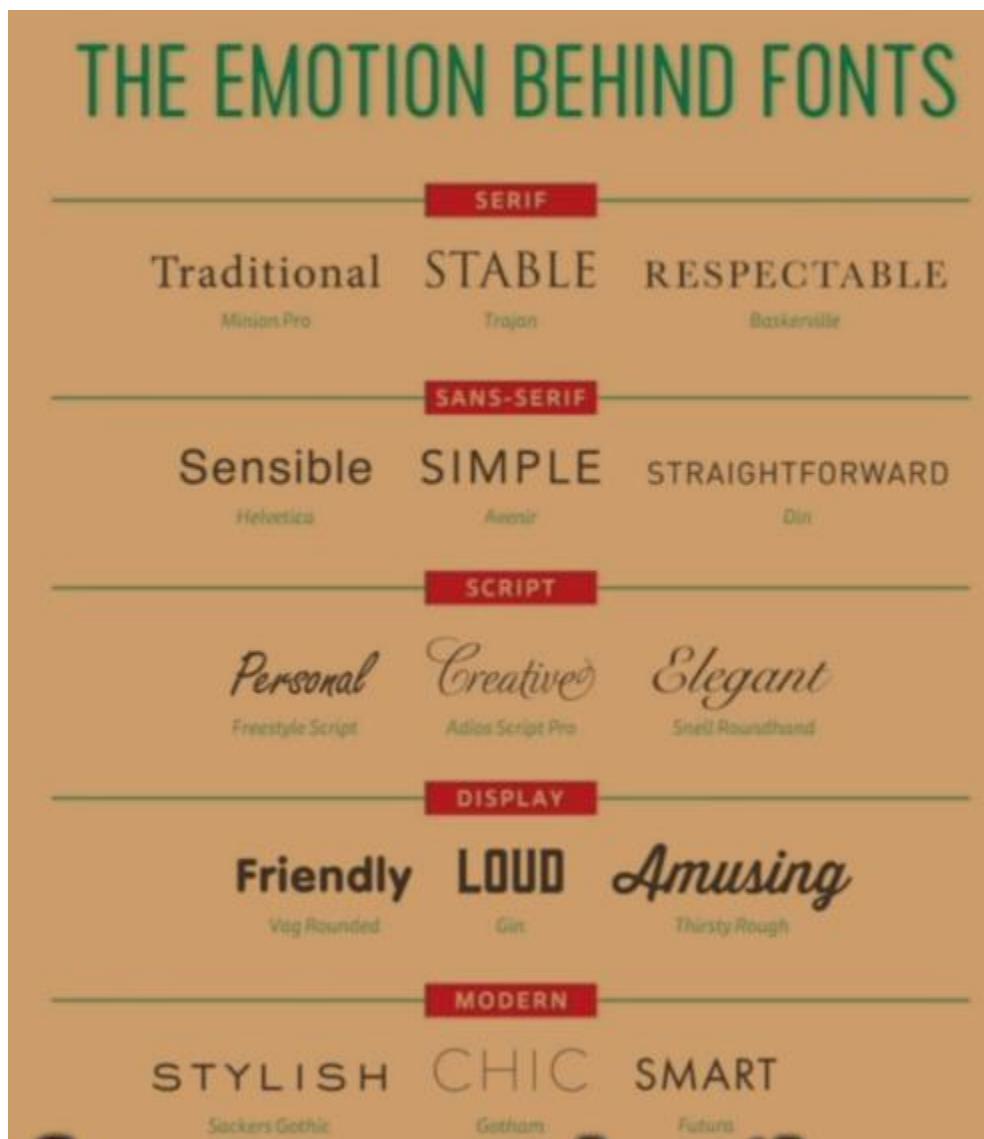
### 3. Understanding Typography and How to Choose a Font

در این فصل به نحوه انتخاب استاندارد فونت و ترکیب آنها صحبت می شود.

- انتخاب فونت و ترکیب آن (typography) نیز مانند انتخاب رنگ، باید از روی آگاهی و مطابق هدف محصول و یا سایت باشد.



- فونت ها نیز دارای MOOD و روانشناسی می باشد و باید در هرجایی از فونت مناسب استفاده شود.



- فونت خانواده **Serif** (دارای پا در حروف) از فونت های قدیمی می باشد، و معمولاً در جاهای رسمی، یا قدیمی یا برای احترام زیاد ، استفاده می شود. نمونه های مدرن تر و جدیدتر **Serif** نیز ایجاد شده است.

Crimson Text  
is a **serif**  
typeface.

Adobe Jenson  
is an **Old Style**  
serif typeface

Baskerville  
is a **Transitional**  
serif typeface

Didot  
is a **Modern**  
serif typeface

American Typewriter  
is a **Slab-Serif**  
typeface

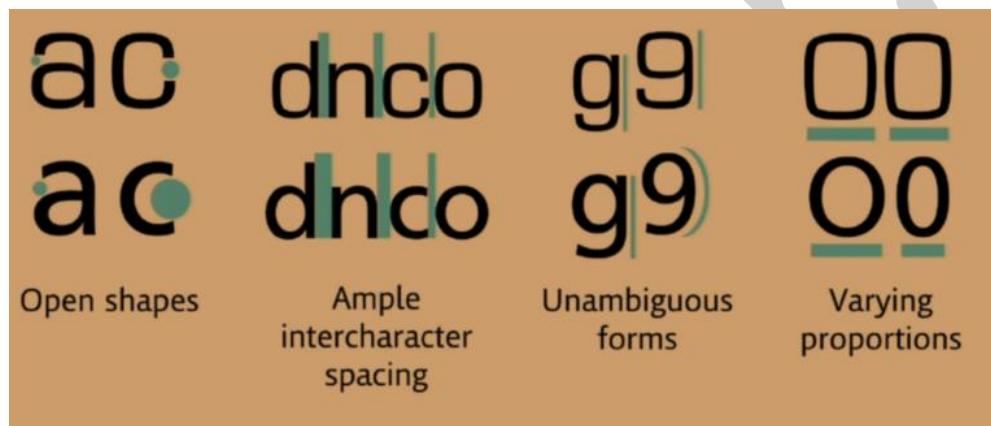
- فونت خانواده Sans-Serif می باشد و معمولاً دوستانه تر، صمیمی تر و ساده تر می باشد و در چنین جاهایی استفاده می شود. همچنین خوانا تر و واضح تر می باشد و معمولاً در متن های طولانی ( body ) از آنها استفاده می شود.

## San-Serifs

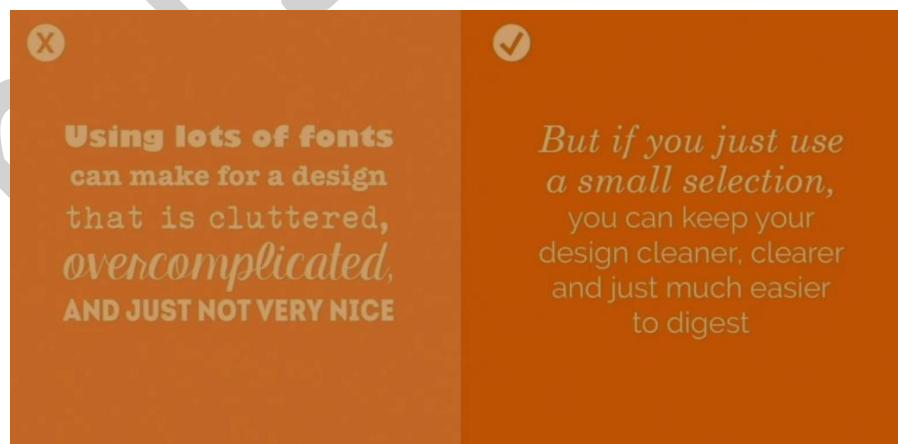
**Delicious** is a  
**sans-serif**  
typeface.



- در شکل زیر مفهوم خوانایی و واضح بودن (legible) را نشان می دهد.



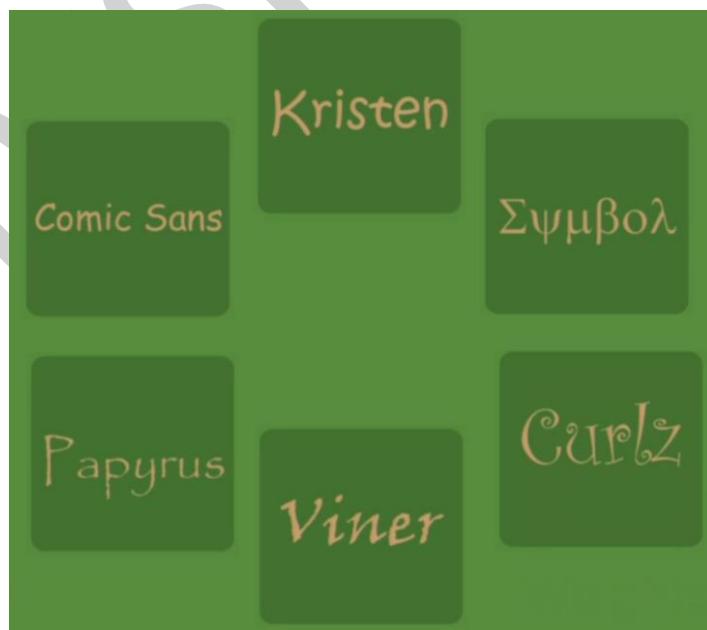
- در هر طراحی (پوستر، سایت و...) از ۲ نوع فونت استفاده کنید و استفاده از چندین فونت باعث آشفتگی کاربر می شود.



- برای استفاده از ۲ فونت، قوانین زیر را رعایت کنید تا کیفیت طراحی بیشتر شود: ۱- حالات و استایل فونت در یک زمینه باشد. ۲- فونت های برای یک دوره یا عصر باشند. ۳- با انتخاب serif و san-serif تمایز ایجاد کنید. ۴- با ایجاد وزن light و bold تمایز ایجاد کنید.



- از نکات گفته شده برای انتخاب فونت استفاده کنید و به هیچ وجه از فونت های ناخوانای زیر استفاده نکنید و حرفه ای باشید.



**Key words:** Select appropriate fonts, new concepts, Serif and San-Serif typeface, readable and legible.

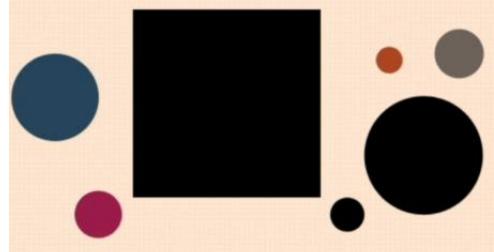
#### 4. Manage ATTENTION with effective User Interface (UI) Design

در این قسمت درباره طراحی رابط کاربری، و اصول آن در طراحی ها صحبت خواهد شد.



- به معرفی ۵ اصل در طراحی رابط کاربری می پردازیم.
- اصل اول رعایت Hierarchy یا سلسله مراتب می باشد، یعنی آن مواردی که چشم ما در نگاه اول می بیند.

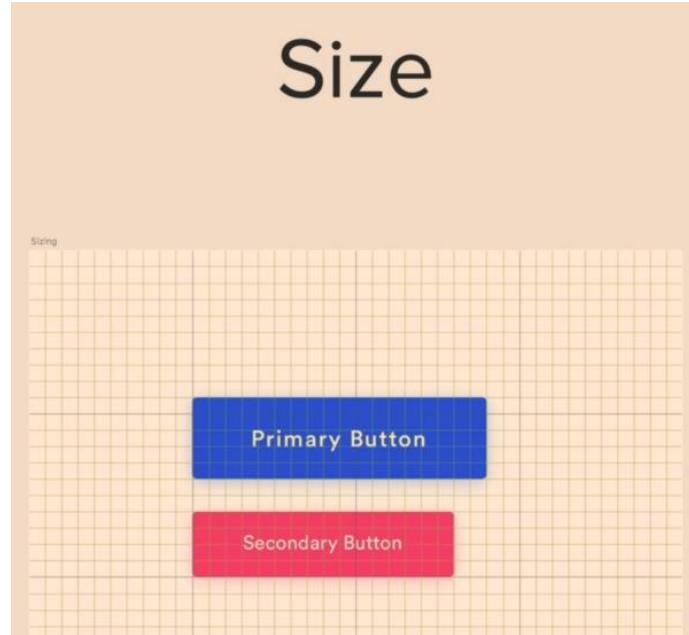
## 1. Hierarchy



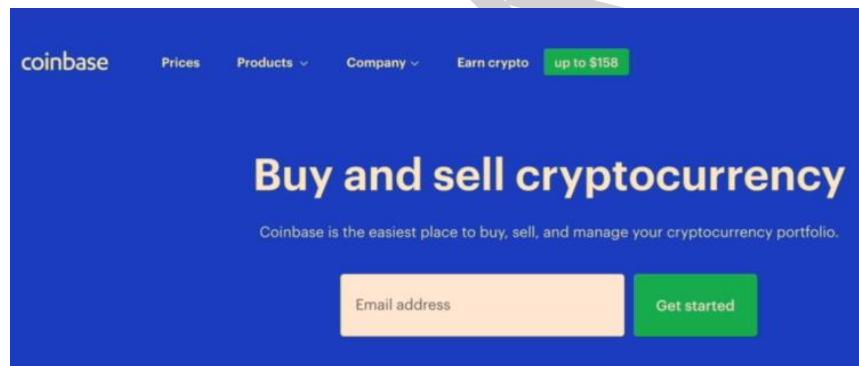


برای برقراری Hierarchy باید به رنگ، اندازه اشیا توجه نمود. -





- به عنوان نمونه در طراحی زیر با توجه به Hierarchy ، در نگاه اول، ابتدا نوشته درشت، سپس رنگ سبز و در مرحله بعد سایر موارد به چشم خواهد خورد.



- اصل دوم layout یا چیدمان درست اشیا در طراحی می باشد. مثلاً به جای استفاده از بلاک های بزرگ از بلاک ها با متن های متوسط(۳۴ حرف در هر خط) استفاده کنید.

## 2. Layout

The diagram illustrates two contrasting layout approaches. On the left, a vertical column of horizontal lines represents a layout where text is presented in a single column. On the right, a grid structure with vertical columns and horizontal rows represents a layout designed for better readability and visual hierarchy.

**X**

This block of text can get a little bit tedious to read after a while, and this may be due to the long line lengths. These sentences have an average of about 57 characters (including) spaces in them whereas the sweet spot is at about 30-40.

too long

too short

On the other hand, this block of text has an average of about 18 characters per line, which is too short, making the sentences choppy and a bit awkward to read.

**✓**

just right

This block of text averages about 34 characters per line and around 6 words per line, making it the most comfortable line length for the eye to read. Keep your line lengths short, people.

- اصل سوم برقراری درست Alignment یا نسبت و فاصله اشیا نسبت به یکدیگر می باشد.

## 3. Alignment

The diagram compares two text blocks based on alignment. The left block, marked with a red 'X', shows text aligned to the left, with a small amount of space on the right. The right block, marked with a green checkmark, shows text centered, with equal space on both sides.

**✗**

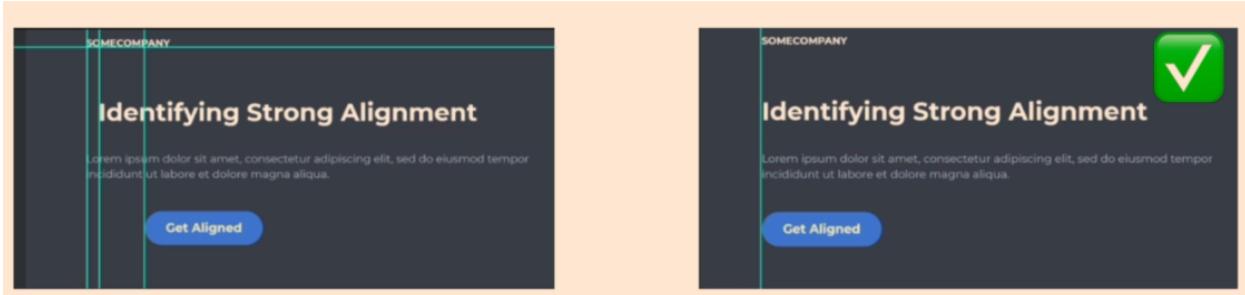
**✓**

**Left Alignment:**

**Text:** Lorem ipsum dolor sit amet.  
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusanti doloremque laudantium.

**Right Alignment:**

**Text:** Lorem ipsum dolor sit amet.  
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusanti doloremque laudantium.



- اصل چهارم ایجاد white space برای اشیای مهم در محصول می باشد. ایجاد پس زمینه سفید برای هر محصول یا آیکونی باعث ایجاد حس ارزشمند بودن و شکوه به محصول می دهد.

## 4. White Space

- طراحی سمت راست شکل تر و با شکوه تر نسبت به سمت چپ می باشد.



- اصل ۵ و مهم، توجه به Audience یا مخاطبان مورد نظر می باشد. یعنی باید طراحی شما برای هر زمینه ای متفاوت باشد. مثلاً طراحی برای کودکان با طراحی برای بزرگسالان متفاوت می باشد. باید انعطاف پذیری در طراحی داشته باشید و طراحیتان مناسب کاربران آن محصول باشد.

## 5. Audience



**Key words:** UI Design, new concepts, UI rules, Hierarchy, Layout, Alignment, white space, Audience.

## 5. User Experience (UX) Design

در این قسمت به معرفی تجربه کاربری یا UX در طراحی صحبت می شود.



UX یعنی تجربه کاربر از محصول، و اینکه کاربر چگونه می تواند تعامل خوب و راحتی داشته باشد. UI چیزی است که فکر می کنیم خوب است، ولی UX چیزی است حتماً خوب است.



- ۵ اصل که باید در UX رعایت کنیم تا کاربر حس خوبی داشته باشد.
- اصل اول سادگی یا simplicity می باشد، یعنی هر محصول را ساده و به دور از پیچیدگی طراحی کنید.



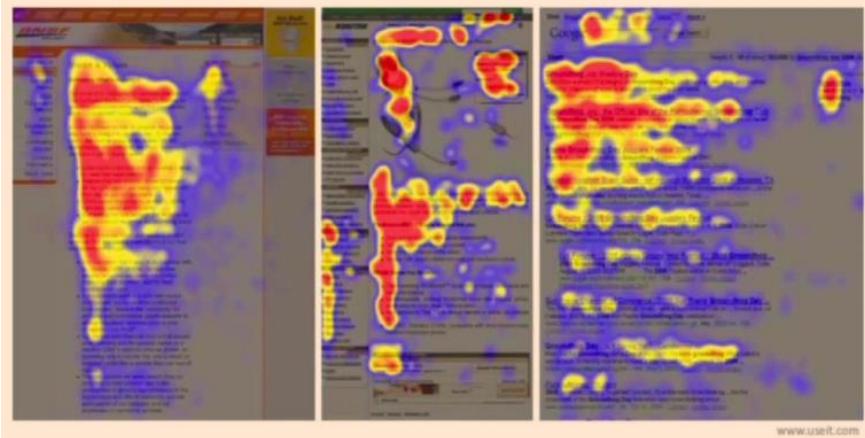
- دو نمونه از سادگی و پیچیدگی در شکل زیر مشخص است.

- اصل دوم، ثبات طراحی یا consistency می باشد. یعنی علاوه بر responsive بودن باید در سایر موارد مشابه ثبات داشته باشد. مثلاً سایت های مختلف یک برنده، باید دارای طراحی ثابتی باشند و هر سایت دارای طراحی های مختلف نباشد تا باعث سردرگمی کاربر نشود.



- اصل سوم، توجه به reading pattern یا نحوه بررسی کردن کاربر در نگاه اول می باشد. در شکل زیر انواع بررسی کردن در نگاه اول را نشان می دهد که مدل F و مدل Z، یکی از متداول ترین ها می باشد.

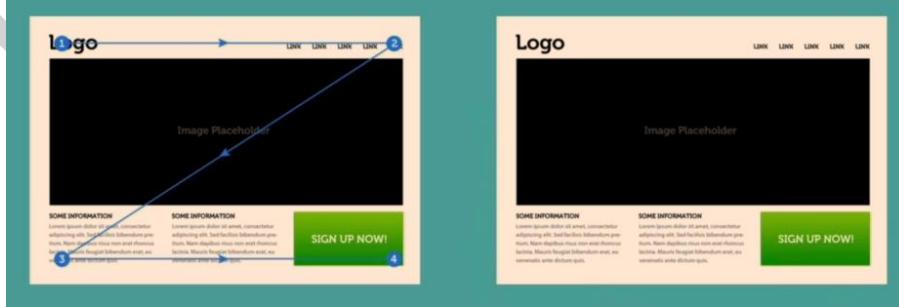
# 3. Reading Patterns

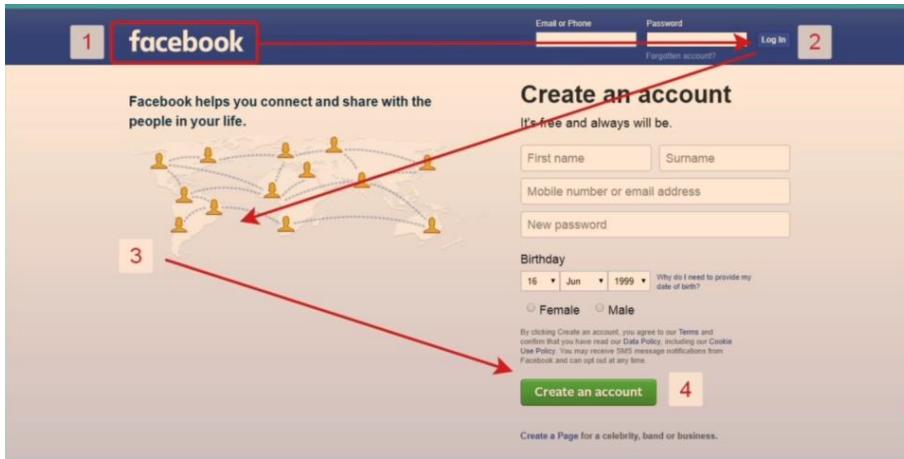


## Use the F-Layout



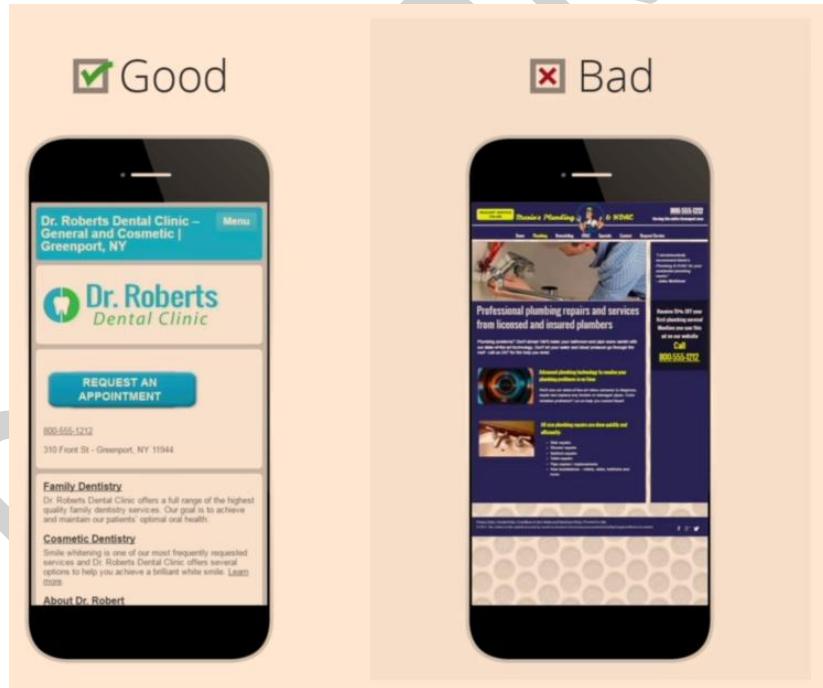
## Or use the Z-Layout





- اصل ۴ توجه به all platform Design می باشد، یعنی علاوه بر responsive بودن، حالت مطلوبی جهت نمایش محتوا برای کاربران در همه پلتفرم ها داشته باشد.

## All Platform Design



- اصل پنجم، عدم استفاده موزیانه و خبیثانه طراحی، جهت گول زدن کاربر می باشد. به این طراحی يا dog pattern يا گویند که کاربر عمل ناخواسته ای را انجام می دهد که شما میخواهید و باعث نارضایت کاربر می شوید.

## 5. Don't Use Your Powers for Evil

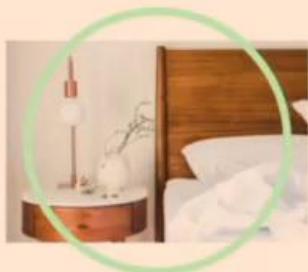


- UX ایده آل و بهینه به مرور زمان ایجاد می شود که کاربر می تواند تجربه و حس خوبی از محصول شما داشته باشد.

**Key words:** UX Design, new concept, UX rules.

## 6. Web Design in Practice - Let's apply what we've learnt!

در این قسمت نکات طراحی گفته شده ، به صورت عملی مورد استفاده قرار می گیرد و طراحی نهایی توسط سایت canva انجام می شود و همچنین سایت های طراحی معرفی می شود.



### Beautiful Food

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc convallis mauris sed magna iaculis ultrices.



### Beautiful Rooms

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc convallis mauris sed magna iaculis ultrices.



### Beautiful Pool

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc convallis mauris sed magna iaculis ultrices.

A large orange shadow effect is cast over the bottom section of the slide, covering the three columns of images and text.



- نمونه ایجاد شده برای حل چالش.

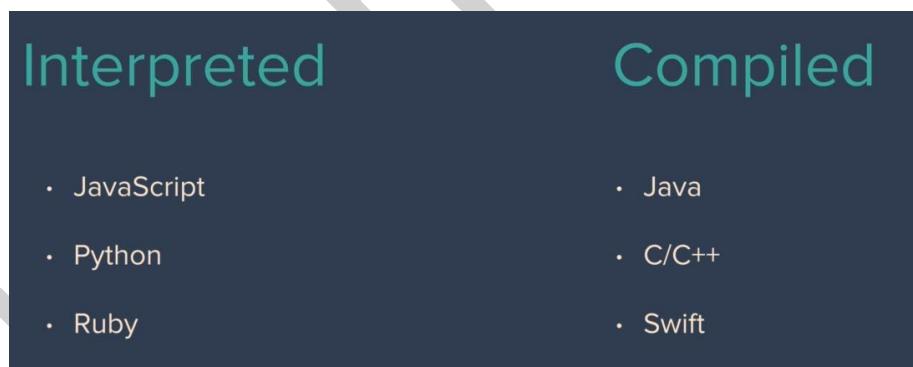


**Key words:** Web Designing in wild, Canva.com.

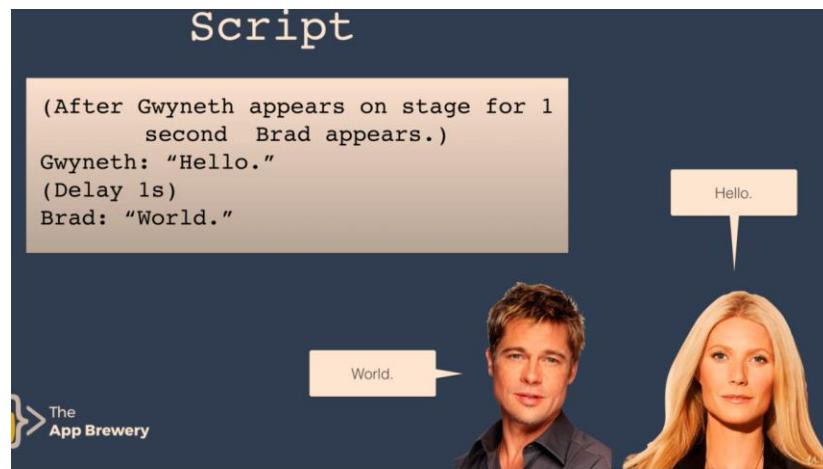
## 9. Introduction to Javascript ES6

### 1. Introduction to Javascript

در این قسمت به معرفی جاوا اسکریپت و ماهیت آن پرداخته می شود و تاریخچه آن گفته می شود.  
جاوا اسکریپت، یک زبان مفسری می باشد، که برای ایجاد حرکت یا اکشن در المنت های Html ساخته شده است که حالات گرافیکی را ایجاد می کند. اما امروزه کاربردهای آن گسترش یافته و در جاهای مختلف استفاده می شود.



- اسکریپت یعنی نوشته یا متن یا نمایشنامه، که هر دستورالعمل به صورت متن در آن می باشد(به دو شکل زیر توجه کنید). اسم گذاری جاوا اسکریپت، ارتباطی با جاوا ندارد، و به دلیل محبوبیت جاوا در آن دوره، اسم آن جاوا اسکریپت شد.



- خیلی از سایت های مطرح بدون جاوا اسکریپت اجرا نمی شوند. در شکل زیر یک مثال بدون جاوا اسکریپت و با جاوا اسکریپت را مشاهده می کنید.

SECTION SEARCH

ENGLISH 中文 (CHINESE) ESPAÑOL

Subscribe to debate, not division.

**The New York Times**

Monday, April 2, 2018 | Today's Paper | Video | 33°F | S. & P. 500 -0.63%

SUBSCRIBE NOW LOG IN

Get The Times for just £1 a week.

World U.S. Politics N.Y. Business Opinion Tech Science Health Sports Arts Books Style Food Travel Magazine T Magazine ALL

**Kushners Saw Hope in White House, but It Was a Mirage**

By SHARON LaFRANIERE and KATIE BENNER

• Charlie Kushner felt redeemed when his son Jared was named senior White House adviser last

**N.C.A.A. TOURNAMENT**

**EDITORIAL**

Facebook Is Not the Problem. Lax Privacy Rules Are.

Internet companies fear privacy regulations. They ought not to.

Can Europe Lead on Privacy?

By TOM WHEELER

**Opinion**

A New Black American Dream

By DIONENE MILLNER

We all want our children to have a better life than we did. But statistics say they won't.

Blow Stephen Clark: Rhythms of Tragedy

There Is a Middle Ground on Guns



**Key words:** Javascript introduction, history, application, interpreted vs compiled language.

## 2. Javascript Alerts - Adding Behaviour to Websites

در این قسمت درباره نوشتن کد جاوااسکریپت و توابع و syntax و ساختار کد، نشان داده می شود.

```

Network Snippets >
+ New snippet
*index.js
1 alert("Hello!");
2 alert("World!");
3

```

```

Console was cleared
< undefined
> alert("Hello");
alert("World!");
< undefined
>

```

- علامت مربوط به string در کد double quotation می باشد نه نقل قول. (در کدها دقت کنید)



- علامت single quotation و double quotation در جاوااسکریپت یکی می باشد، ولی باید از double quotation استفاده کنید.

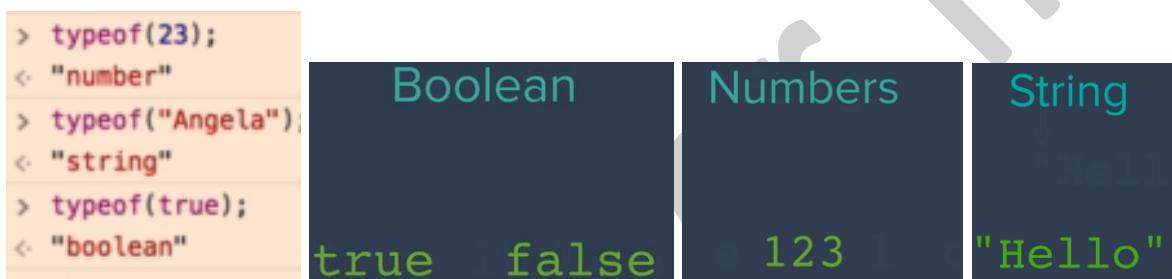
```
alert( 'Hello' );
```

- به استاندارد نویسی در جاوا اسکریپت دقت کنید و گرامر کد را رعایت کنید تا کد تمیز و حرفه ای داشته باشید. می توان به <https://github.com/rwaldron/idiomatic.js> مراجعه نمود.

**Key words:** first code javascript, console and snippet in browser, create pop-up, javascript syntax and instruction.

### 3. Data Types

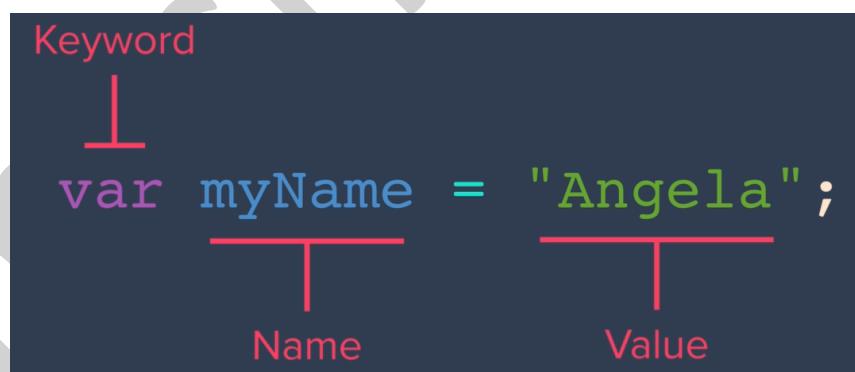
در این قسمت ۳ نوع داده مهم در جاوا اسکریپت را معرفی می کند.



**Key words:** important data type in Javascript, string, number, Boolean, foundation block.

### 4. Javascript Variables

در این قسمت درباره ایجاد متغیر در جاوا اسکریپت توسط دستور `var` و استفاده از آن صحبت می شود.



```
var myName = "Angela";
var yourName = prompt("What is your name?");
alert("My name is " + myName + ", welcome to my course " + yourName + "!");
```

```
index.js* x
1 var gameLevel = 1;
2 gameLevel = 2;
3 gameLevel = 3;
4 alert("Your level is currently: " + gameLevel);
```

**Key words:** Variable, var instruction, create new variable, new concept.

## 5. Javascript Variables Exercise Start

در این قسمت، یک تمرین مربوط به **variable** گفته می شود.

```
1 | var a = "3";
2 | var b = "8";
```

So that the variable **a** holds the value "8".

And the variable **b** holds the value "3".

When the code is run, it should output:

**a is 8**

**b is 3**

Do **NOT** change any of the existing code.

You are **NOT** allowed to type any numbers.

You should **NOT** redeclare the variables a and b.

Hint: Use [this code playground](#) to run your code and see if it matches your expectations.

Hint: The solution is just 3 lines of code.

index.js

```
1 * function test() {
2     var a = "3";
3     var b = "8";
4
5 * //*****Do not change the code above ⌘ *****/
6 //Write your code on lines 7 - 9:
7
8
9
10 * //*****Do not change the code below ⌘ *****/
11
12     console.log("a is " + a);
13     console.log("b is " + b);
14 }
15
```

## 7. Javascript Variables Exercise Solution

در این قسمت راه حل تمرین را بیان می کند.

```

1 * function test() {
2     var a = "3";
3     var b = "8";
4
5 * //*****Do not change the code above ⚡ *****/
6 //Write your code on lines 7 - 9:
7     var c = a;
8     a = b;
9     b = c;
10 * //*****Do not change the code below ⚡ *****/
11
12     console.log("a is " + a);
13     console.log("b is " + b);
14 }
15

```

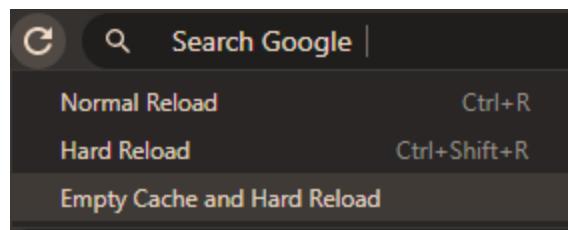
Line 9, Column 11 Saving changes...



## 8. Naming and Naming Conventions for Javascript Variables

در این قسمت نحوه نام گذاری متغیرهای جاوااسکریپت و کاراکترها مجاز برای نام گذاری را نشان می دهد.

- برای نام گذاری متغیرها از استاندارد Camel (چون کوهان شتر بالاتر از سرش قرار دارد) استفاده می کنیم، یعنی کلمه اول حروف کوچک و حروف اول بقیه کلمه ها بزرگ می باشد مانند .myNameIsAli
- برای نام گذاری فقط حروف و اعداد و علامت \_ (Under score) و \$ (Dollar sign) مجاز می باشد.
- توسط کلید Shift+Enter می توان تو رفتگی در خط کد در console ایجاد کرد.
- توسط راست کلیک بر روی reload ، می توان دیتا را پاک نمود.



**Key words:** clear screen, clean variables or hard reload, standard of naming variables in Javascripts, Camel naming.

## 10. String Concatenation

در این قسمت درباره نحوه الحاق کردن چند کلمه در کنار هم به همراه فضای خالی صحبت می شود.

```

index.js* x
1 var message = "Hello";
2 var name = "Angela";
3
4 //Write your code below this line:
5
6 alert(message + " there," + " " + name);

```

**Key words:** String Concatenation, string spaces.

## 11. String Lengths and Retrieving the Number of Characters

در این قسمت نحوه بدست آوردن طول کاراکترهای String توسط دستور length را نشان می دهد و یک برنامه برای شمارش کاراکترهای توییتر ایجاد می کند.

```

index.js* x
1
2 var tweet = prompt("Compose your tweet:");
3 var tweetCount = tweet.length;
4 alert("You have written " + tweet.length + " characters, you have " + (140 - tweet.length) + " characters remaining.");
5
6
7
8
9
10 //You have written 182 characters, you have -42 characters left.

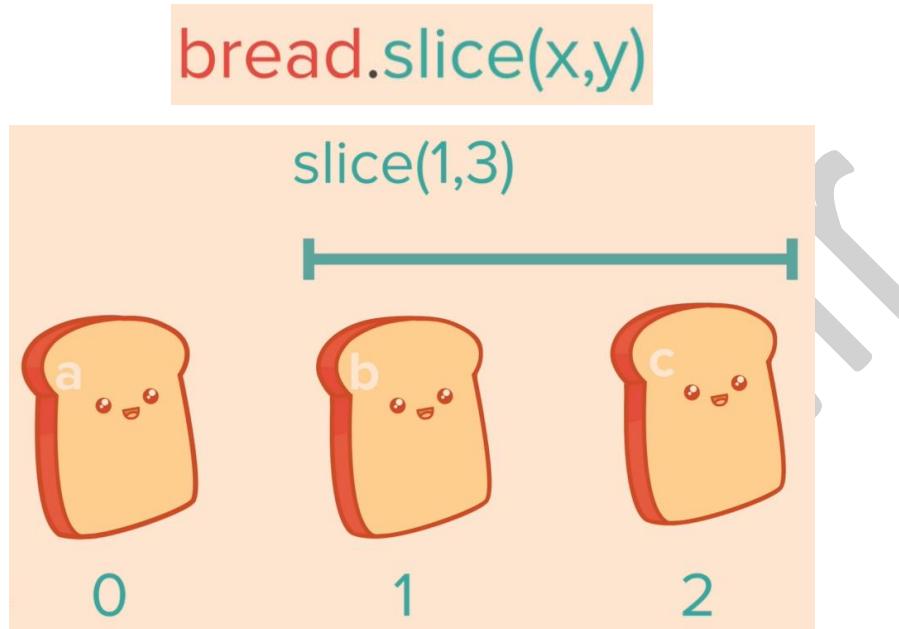
```

- با // می توان یک لاین را comment نمود و با /\* \*/ می توان چند خط را comment نمود.

**Key words:** length attribute in Javascript, make new code, how to comment in Javascript.

## 12. Slicing and Extracting Parts of a String

در این قسمت درباره خصوصیت `slice` برای `string` صحبت می شود که می توان محدوده کاراکترهای دلخواه را برداشت کرد.



- در کد زیر نحوه برش توییت به مقادیر کوچکتر را نشان می دهد.

```
index.js* x
1
2 var tweet = prompt("Compose your tweet:");
3 var tweetUnder140 = tweet.slice(0,140);
4 alert(tweetUnder140);
5
6
```

- خلاصه تر کد بالا.

```
index.js* x
1
2
3 alert(prompt("Compose your tweet:").slice(0,140));
```

**Key words:** slicing string, selecting string.

## 13. Challenge Changing Casing in Text

در این قسمت درباره دستورات `LowerCase` و `UpperCase` برای `string` صحبت می شود و همچنین چالشی برای حل کردن داده می شود.

```
index.js* x
1 var name = "Angela";
2 name = name.toUpperCase();
3 name = name.toLowerCase();
```

**Key words:** toUpperCase, toLowerCase, challenge.

#### 14. Challenge Changing String Casing Solution

در این قسمت مسئله مورد نظر را با موارد درس داده شده، حل می کند و چالش را به مسائل کوچکتر تقسیم می کند. تقسیم هر مسئله به مسائل کوچکتر مهمترین اصل در برنامه نویسی می باشد.

```
index.js* x
1
2
3 //1 Create a var that stores the name that user enters via prompt.
4
5 var name = prompt("What is your name?");
6
7 //2 Capitalise the first letter of their name.
8
9 //a isolate the first char
10
11 //b Turn the first char to upper case
12
13 //c Isolate the rest of the name
14
15 //d concactenate the first char with the rest of the char
16
17 //3 We use the capitalised version of their name to greet them using an alert.
18
19
20 //Hello, Angela.
```

**Key words:** resolve challenge, break down your problem.

#### 15. Basic Arithmetic and the Modulo Operator in Javascript

در این قسمت با انواع عملگرهای ریاضی و حق تقدم آنها صحبت می شود.

```
index.js* x
1 var dogAge = prompt("How old is your dog?");
2 var humanAge = ((dogAge - 2) * 4) + 21;
3 alert("Your dog is " + humanAge + " years old in human years.");
```

**Key words:** Math in Javascript.

## 16. Increment and Decrement Expressions

در این قسمت یکسری علامت میانبر برای عملیات افزایش یا کاهش در مقدار نشان داده می شود.

```
var x = 5;  
x-- ; //4
```

```
var x = 5;  
x++; //6
```

```
var x = 5;  
var y = 3;  
x += y ; //8
```

+=  
-=  
\*=  
/=

**Key words:** Increment and Decrement Expressions.

## 18. Functions Part 1 Creating and Calling Functions

در این قسمت نحوه ایجاد تابع و فراخوانی آن توضیح داده می شود.

Creating the function

```
function getMilk( ) { }
```

```
alert leaveHouse  
alert moveRight  
alert moveRight  
alert moveUp  
alert moveUp  
alert moveUp  
alert moveUp  
alert moveRight  
alert moveRight  
alert buyMilk  
alert moveLeft  
alert moveLeft  
alert moveDown  
alert moveDown  
alert moveDown  
alert moveDown  
alert moveLeft  
alert moveLeft  
alert enterHouse
```

Calling the function

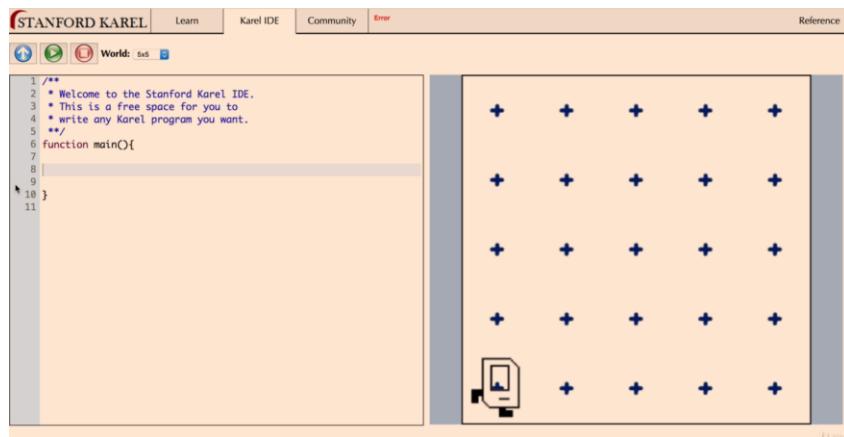
```
getMilk();
```



**Key words:** making and calling function, console.log().

## 19. Functions Part 1 Challenge - The Karel Robot

در این قسمت برای اینکه فهم بیشتری از تابع داشته باشیم با ربات karel سرگرم می شویم.



**Key words:** Karel IDE, more understand about function.

## 22. Functions Part 2 Parameters and Arguments

در این قسمت درباره تابعی که ورودی می گیرد صحبت می شود که می توان پارامتری برای آن تعریف کرد.

The diagram illustrates functions with parameters and arguments. It shows two parts: 'Creating the function' and 'Calling the function'.

**Creating the function:**

```
function getMilk (bottles) {
    var cost = bottles * 1.5;
    //Do something with cost
}
```

**Calling the function:**

```
getMilk(2);
```

To the right of the text, there is an image of a chocolate ice cream cone.

**Key words:** vanilla and chocolate functions, advanced function.

## 24. Life in Weeks Solution

در این قسمت مسئله ای داده می شود، و آن را حل می کند.

## Life in Weeks Coding Exercise

I was reading this article by Tim Urban - Your Life in Weeks and realised just how little time we actually have.

In this challenge, you are going to create a function that tells us how many days, weeks and months we have left if we live until 90 years old.

It will take your current age as the input and console.logs a message with our time left in this format:

You have x days, y weeks, and z months left.

Where x, y and z are replaced with the actual calculated numbers.

For this challenge, assume there are **365** days in a year, **52** weeks in a year and **12** months in a year.

```
1 - function lifeInWeeks(age) {  
2  
3     var yearsRemaining = 90 - age;  
4     var days = yearsRemaining * 365;  
5     var weeks = yearsRemaining * 52;  
6     var months = yearsRemaining * 12;  
7  
8     console.log("You have " + days + " days, " + weeks + " weeks, and " + months + " months left  
9     .");  
10    |  
11    }  
12
```

**Key words:** resolving task.

## 25. Functions Part 3 Outputs & Return Values

در این قسمت به معرفی تابعی که هم ورودی و هم خروجی دارد می پردازد.

Creating the function

```
function getMilk (money) {  
    return money % 1.5;  
}
```

Calling the function

```
var change = getMilk(4);
```



**Key words:** strawberry function, function with output.

## 26. Challenge Create a BMI Calculator

در این قسمت یک مسئله جدید داده می شود.

$$BMI = \frac{weight(kg)}{height^2(m^2)}$$

Your challenge is to create a function that takes height and weight as inputs and gives the calculated BMI value as an output.

```
index.js
1 //Create your function below this line.
2
3
4
5
6 /* If my weight is 65Kg and my height is 1.8m, I should be able to call your function like this:
7
8 var bmi = bmiCalculator(65, 1.8); |   |
9   bmi should equal around 20 in this case.
10
11 */
12
13
```

**Key words:** Challenge.

## 28. Challenge BMI Calculator Solution

در این قسمت راه حل مسئله داده می شود.

```
1 //Create your function below this line.
2
3
4 function bmiCalculator(weight, height) {
5   var bmi = weight / (height * height);
6   return Math.round(bmi);
7 }
8
```

**Key words:** challenge solution.

## 30. Tip from Angela - Set Your Expectations

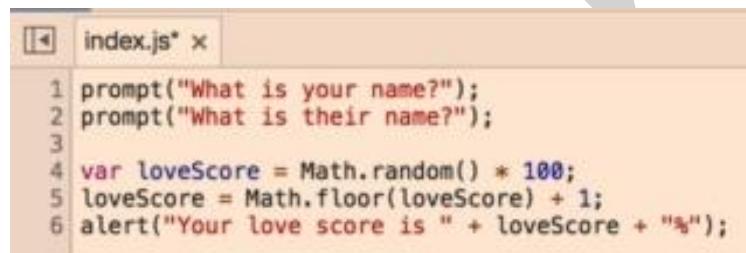
برنامه نویس نه خیلی سخته، نه خیلی آسون. دقیقاً مثل هر کار دیگه ای هستش و باید زمان گذاشت و به نتیجه رسید.

## 10. Intermediate Javascript

### 1. Random Number Generation in Javascript Building a Love Calculator

در این قسمت نحوه ایجاد عدد تصادفی در محدوده دلخواه به صورت عدد صحیح را نشان می دهد که در راستای پروژه Love Calculator می باشد.

var n = Math.random();	0.3647382746318429
n = n * 6;	2.18842964779
n = Math.floor(n);	2



```
index.js* x
1 prompt("What is your name?");
2 prompt("What is their name?");
3
4 var loveScore = Math.random() * 100;
5 loveScore = Math.floor(loveScore) + 1;
6 alert("Your love score is " + loveScore + "%");
```

**Key words:** pseudo random, round number, create Love Calculator.

### 2. Control Statements Using If-Else Conditionals & Logic

در این قسمت درباره حالات شرطی یا control flow صحبت خواهد شد و در پروژه به کار برده خواهد شد.

```
if (track === "clear") {
  goStraight();
} else {
  turnRight();
}
```

```

index.js* x
1 prompt("What is your name?");
2 prompt("What is their name?");
3
4 var loveScore = Math.random() * 100;
5 loveScore = Math.floor(loveScore) + 1; //1-100
6
7 if (loveScore > 70) {
8     alert("Your love score is " + loveScore + "%" + "You love each other like Kanye loves Kanye.");
9 } else {
10    alert("Your love score is " + loveScore + "%");
11 }
12

```

**Key words:** control flow, if and else conditions.

### 3. Comparators and Equality

در این قسمت درباره عملگرهای مقایسه‌ای صحبت می‌شود و همچنین تفاوت عملگر ۲ مساوی با ۳ مساوی را بیان می‌کند.

<b>====</b>	Is equal to
<b>!==</b>	Is not equal to
<b>&gt;</b>	Is greater than
<b>&lt;</b>	Is lesser than
<b>&gt;=</b>	Is greater or equal to
<b>&lt;=</b>	Is lesser or equal to

- علامت ۲ مساوی فقط از نظر مقدار مقایسه می‌کند، ولی علامت ۳ مساوی علاوه بر مقدار به نوع داده مورد نظر هم توجه می‌کند.

```

> var a = 1;
var b = "1";
< undefined
> typeof(a);
< "number"
> typeof(b);
< "string"
> if (a === b) {
    console.log("yes");
} else {
    console.log("no")
}

```

**Key words:** comparators sign, different between 2 equal sign and 3 equal sign in Javascript.

#### 4. Combining Comparators

در این قسمت ۳ نوع عملگر برای استفاده در شرط ها معرفی می شود و از آن در پروژه استفاده می شود.



```
index.js* x
1 if (loveScore > 70) {
2     alert("Your love score is " + loveScore + "%" + "You love each other like Kanye loves Kanye.");
3 }
4
5 if (loveScore > 30 && loveScore <= 70) {
6     alert("Your love score is " + loveScore + "%");
7 }
8
9 if (loveScore <= 30) {
10     alert("Your love score is " + loveScore + "%" + " You go together like oil and water.");
11 }
```

**Key words:** Combining Comparators for conditions, OR, AND, NOT.

#### 6. Introducing the Leap Year Code Challenge

در این قسمت چالش جدیدی مطرح می شود که سال های کبیسه یا ۳۶۶ روزه را پیدا کنید.

##### Leap Year Challenge

👉 This is a Difficult Challenge 👈

Write a program that works out whether a given year is a leap year. A normal year has 365 days, leap years have 366, with an extra day in February. The reason why we have leap years is really fascinating, [this video](#) goes into more detail.

This is how to work out whether a particular year is a leap year:

*on every year that is evenly divisible by 4*

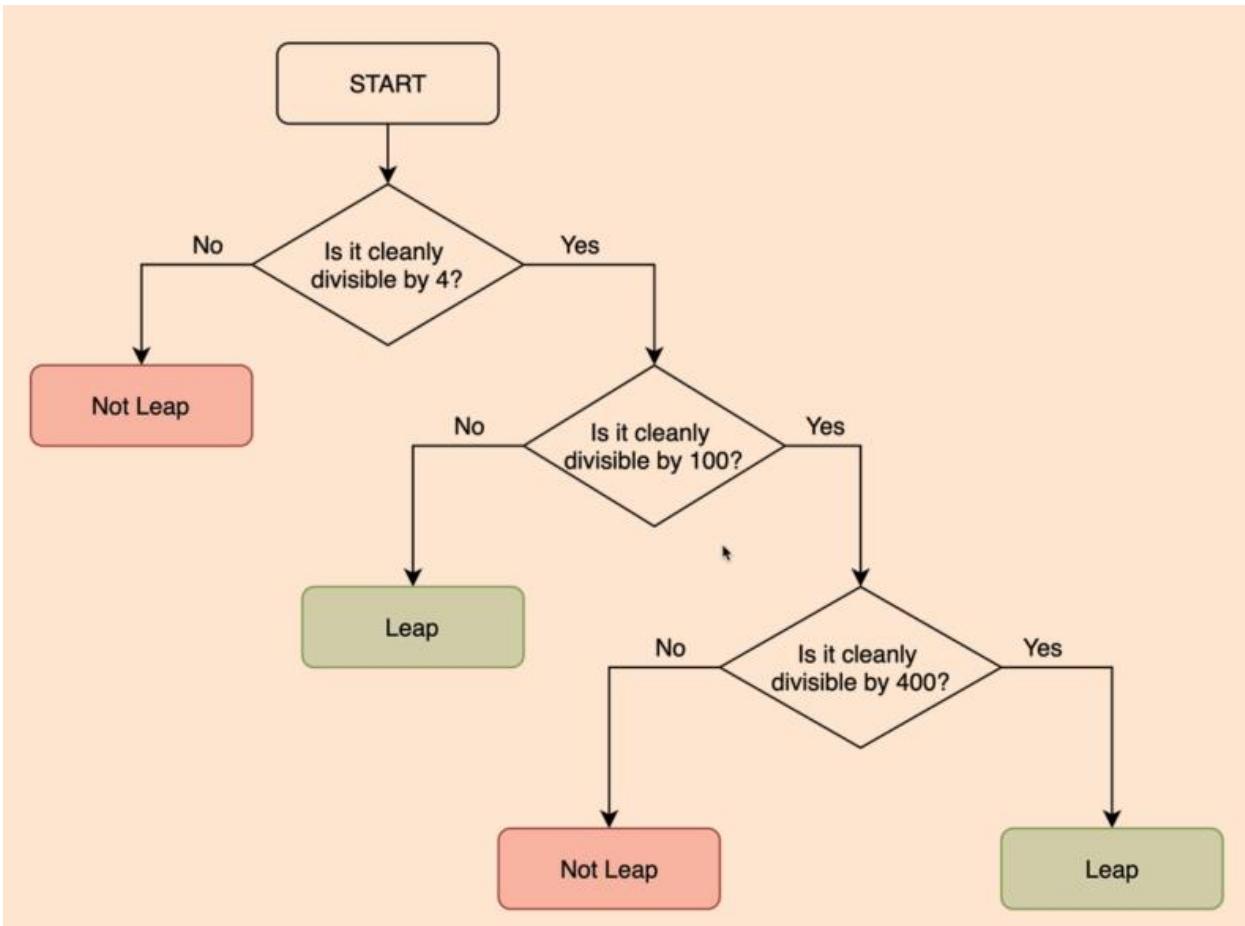
*except every year that is evenly divisible by 100*

*unless the year is also evenly divisible by 400*

**Key words:** Leap year challenge.

#### 8. Leap Year Solution

در این قسمت به حل چالش به همراه فلوچارت شرط ها می پردازد.



```

1+ function isLeap(year) {
2
3+   if (year % 4 === 0) {
4+     if (year % 100 === 0) {
5+       if (year % 400 === 0) {
6+         return "Leap year.";
7+       } else {
8+         return "Not leap year.";
9+       }
10+    } else {
11+      return "Leap year.";
12+    }
13+  } else {
14+    return "Not leap year.";
15+  }
16+
17+ }
  
```

**Key words:** Leap Year Solution.

## 9. Collections Working with Javascript Arrays

در این قسمت به معرفی آرایه ها و نحوه ایجاد و استفاده از آنها پرداخته می شود.

The first card shows code: `var myEgg = eggs[1];` with a red arrow pointing to the index `eggs[1]`. Below it is another card showing `eggs.length; 5`. The third card shows `eggs.includes()` with a green checkmark and a red arrow pointing to the last egg in the array. The fourth card shows the full array `var eggs = [ ... ]` with a red arrow pointing to the last egg.

`var myEgg = eggs[1];`

`var eggs = [ , , , ,  ]`

`eggs.length; 5`

`var eggs = [ , , , ,  ]`

`eggs.includes()` ✓

`var eggs = [ , , , ,  ]`

`index.js*`

```
1 var guestList = ["Angela", "Jack", "Pam", "James", "Lara", "Jason"];
2 var guestName = prompt("What is your name?");
3 if (guestList.includes(guestName)) {
4     alert("Welcome!");
5 } else {
6     alert("Sorry, maybe next time.");
7 }
```

**Key words:** making and using Arrays.

## 10. Adding Elements and Intermediate Array Techniques

در این قسمت خصوصیات بیشتری درباره آرایه ها توضیح داده می شود و نحوه افزودن و یا حذف آیتم به آرایه توضیح داده می شود.



Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.

```

1  //index.js* x
2
3  function fizzBuzz() {
4      //write code here.
5
6      if (count % 3 === 0 && count % 5 === 0) {
7          output.push("FizzBuzz");
8      }
9      else if (count % 3 === 0) {
10         output.push("Fizz");
11     }
12     else if (count % 5 === 0) {
13         output.push("Buzz");
14     }
15     else {
16         output.push(count);
17     }
18
19
20
21     count++;
22

```

**Key words:** Adding and removing Items from array with Push and pop.

## 12. Who's Buying Lunch Solution

در این قسمت چالش انتخاب اسم تصادفی، در یک آرایه را حل می کند.

### Who's Buying Lunch? Code Challenge

You are going to write a function which will select a random name from a list of names. The person selected will have to pay for everybody's food bill.

**Important:** The output should be returned from the function and you do not need **alert**, **prompt** or **console.log**. The output should match the example output exactly, including capitalisation and punctuation.

#### Example Input

```
1 | ["Angela", "Ben", "Jenny", "Michael", "Chloe"]
```

#### Example Output

```
1 | Michael is going to buy lunch today!
```

index.js	<pre> 1+ function whosPaying(names) { 2 3     var numberOfPeople = names.length; 4     var randomPersonPosition = Math.floor(Math.random() * numberOfPeople); 5     var randomPerson = names[randomPersonPosition]; 6 7     return randomPerson + " is going to buy lunch today!"; 8 9 }</pre>
----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

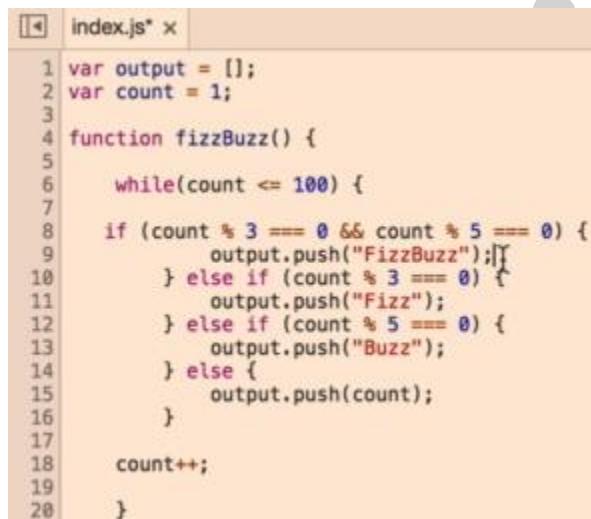
**Key words:** Challenge and solution.

## 13. Control Statements While Loops

در این قسمت نحوه ایجاد و استفاده از حلقه **while** را توضیح می دهد. همچنین یک چالش جدید نیز برای حل کردن دارد.

# While Loops

```
while (something is true) {  
    //Do something  
}
```



```
index.js* x  
1 var output = [];  
2 var count = 1;  
3  
4 function fizzBuzz() {  
5  
6     while(count <= 100) {  
7  
8         if (count % 3 === 0 && count % 5 === 0) {  
9             output.push("FizzBuzz");}  
10        } else if (count % 3 === 0) {  
11            output.push("Fizz");}  
12        } else if (count % 5 === 0) {  
13            output.push("Buzz");}  
14        } else {  
15            output.push(count);}  
16        }  
17  
18        count++;  
19    }  
20}
```

## Lyrics of the song 99 Bottles of Beer

99 bottles of beer on the wall, 99 bottles of beer.  
Take one down and pass it around, 98 bottles of beer on the wall.

98 bottles of beer on the wall, 98 bottles of beer.  
Take one down and pass it around, 97 bottles of beer on the wall.

97 bottles of beer on the wall, 97 bottles of beer.  
Take one down and pass it around, 96 bottles of beer on the wall.

96 bottles of beer on the wall, 96 bottles of beer.  
Take one down and pass it around, 95 bottles of beer on the wall.

95 bottles of beer on the wall, 95 bottles of beer.  
Take one down and pass it around, 94 bottles of beer on the wall.

راه حل مسئله:

```

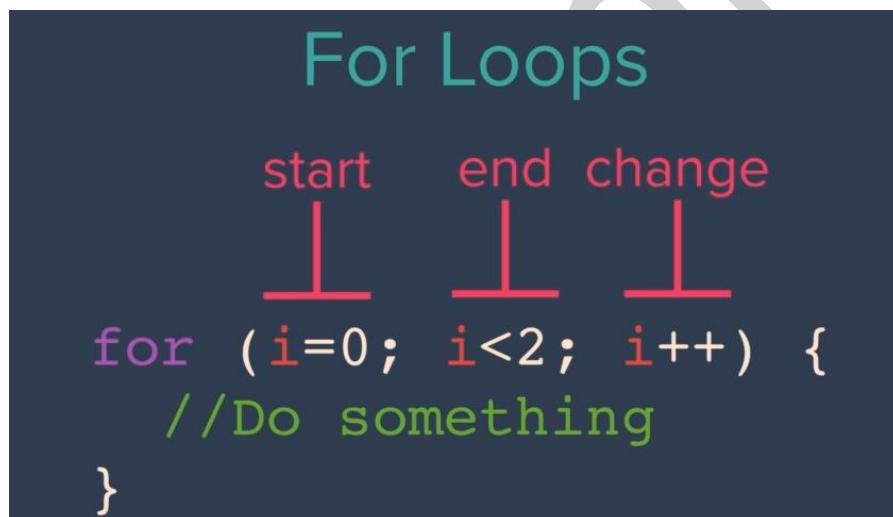
var numberOfBottles = 99
while (numberOfBottles >= 0) {
  var bottleWord = "bottle";
  if (numberOfBottles === 1) {
    bottleWord = "bottles";
  }
  console.log(numberOfBottles + " " + bottleWord + " of beer on the wall");
  console.log(numberOfBottles + " " + bottleWord + " of beer,");
  console.log("Take one down, pass it around,");
  numberOfBottles--;
  console.log(numberOfBottles + " " + bottleWord + " of beer on the wall.");
}

```

**Key words:** making and using While Loops, optimize FizzBuzz code.

## 15. Control Statements For Loops

در این قسمت نحوه ایجاد و استفاده از حلقه For و کاربرد آن گفته می شود.



```

1
2
3
4 function fizzBuzz() {
5
6   for(var count = 1; count < 101; count++) {
7     if (count % 3 === 0 && count % 5 === 0) {
8       output.push("FizzBuzz");
9     } else if (count % 3 === 0) {
10       output.push("Fizz");
11     } else if (count % 5 === 0) {
12       output.push("Buzz");
13     } else {
14       output.push(count);
15     }
16   }
17
18   console.log(output);
19 }
20

```

```

while (something is true) {
    //Do something
}

for (i=0; i<2; i++) {
    //Do something
}

```

State

Iterate

**Key words:** making and using For Loops, application of For and While, more optimize FizzBuzz code.

## 16. Introducing the Fibonacci Code Challenge

در این قسمت مسئله فیبوناچی مطرح می شود که باید آن را ایجاد کنید.

### Fibonacci Challenge

Fibonacci was an Italian mathematician who came up with the Fibonacci sequence:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 ...

Where every number is the sum of the two previous ones.

e.g. 0, 1, 1, 2, 3, 5 comes from

0 + 1 = 1

1 + 1 = 2

1 + 2 = 3

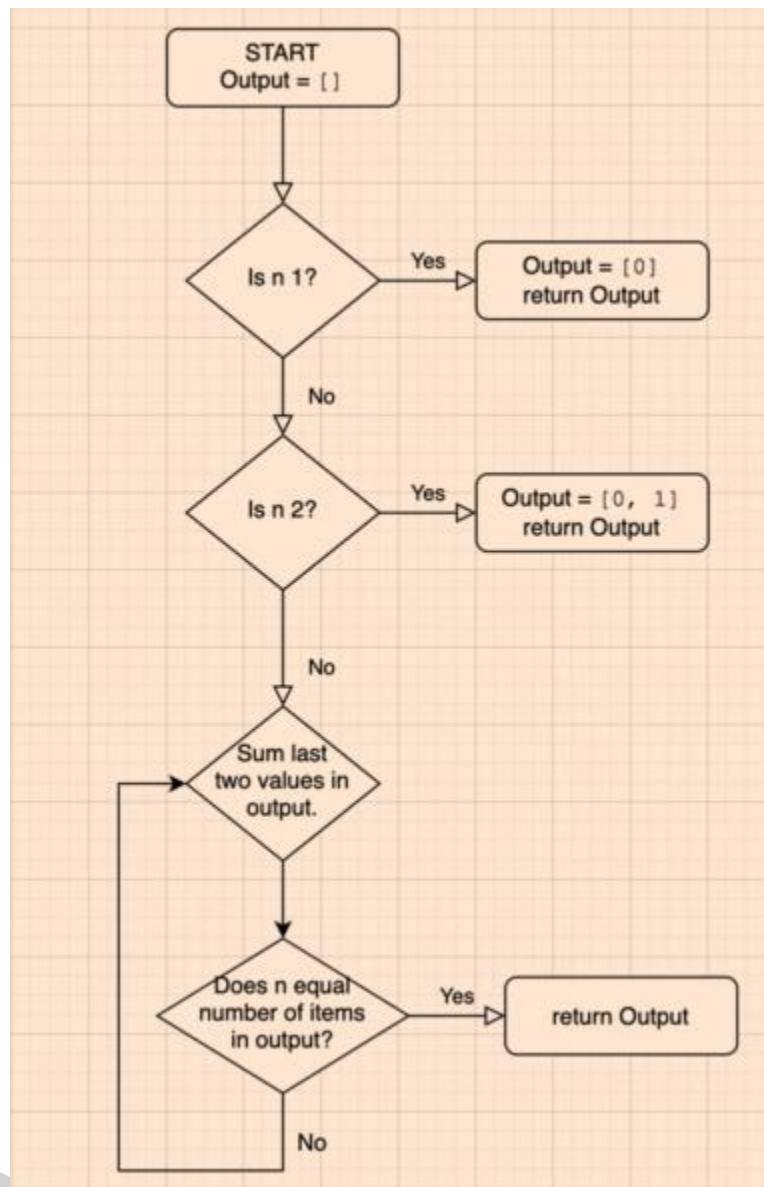
2 + 3 = 5

etc.

**Key words:** Challenge.

## 18. Fibonacci Solution

در این قسمت مسئله فیبوناچی را حل می کند.



```
index.js      ── saving...
1
2   function fibonacciGenerator (n) {
3
4     var output = [];
5     if (n === 1) {
6       output = [0];
7     }
8     else if (n === 2) {
9       output = [0, 1];
10    }
11    else {
12      output = [0, 1];
13
14      for (var i = 2; i < n; i++) {
15        output.push(output[output.length - 2] +
16          output[output.length - 1]);
17      }
18
19    return output;
20  }
21
22  output = fibonacciGenerator(10);
23  console.log(output)  I
```

**Key words:** Solution.

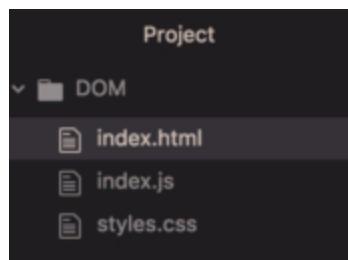
### 19. Tip from Angela - Retrieval is How You Learn

یکی از مشکلات یادگیری این است که فقط یاد می گیریم، و تمرین نمی کنیم. برای اینکه مطلبی را به صورت کامل درک کنید، باید با حل تمرین و چالش آن را تقویت کنید، در غیر این صورت چیزی یاد نخواهید گرفت.

## 11. The Document Object Model (DOM)

### 1. Adding Javascript to Websites

در این قسمت نحوه و مکان و روش افزودن جاوااسکریپت به وبسایت را نشان می دهد.



- شکل های زیر به ترتیب نشان دهنده، افزودن به روش، inline و internal و external را نشان می دهد.

```
<body onload="alert('Hello');">
  <h1>Hello</h1>
</body>
```

```
<body>
  <h1>Hello</h1>

  <script type="text/javascript">

    alert("Hello");

  </script>

</body>
```

index.html	index.js
1	
2      alert("Hello");	
3	

- باید دقت کنید، که کدهای جawa اسکریپت باید در خط های آخر قرار گیرند، یعنی بعد از بارگذاری html و CSS، چونکه هدف ایجاد تعامل در سایت را دارند.

**Key words:** How to add Javascript to Website, inline, internal, outline.

## 2. Introduction to the Document Object Model (DOM)

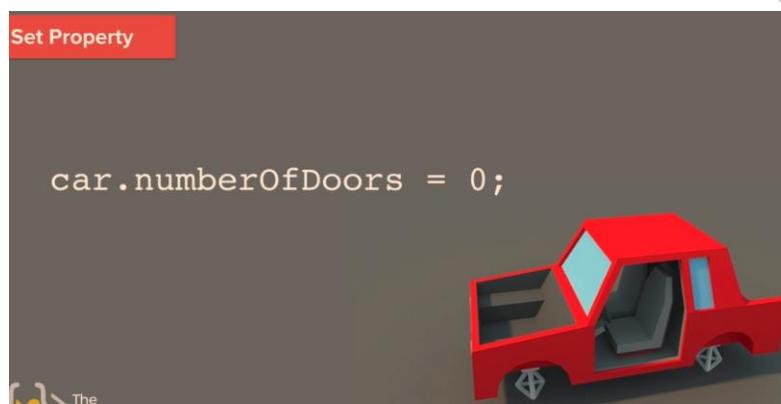
در این قسمت به معرفی DOM و نحوه استفاده از آن و همچنین نحوه استفاده از خصوصیات و تابع های هر المنت در وب صحبت می شود.

DOM در واقع یک مدل می باشد، که سلسله مراتب المنشت های موجود در صفحه را به صورت درختی مستند می کند و می توان از راهنمایی آن، برای دسترسی به المنشت ها و تغییر آن توسط Javascript اقدام نمود.



- هر شئ یا المنشت دارای properties و Methods می باشد. نشان دهنده مشخصات و Method نشان دهنده تابع های مخصوص آن شئ می باشد.





## Properties

- innerHTML
- style
- firstChild

**button**

## Methods

- click()
- appendChild()
- setAttribute()

- در کدهای زیر نحوه فراخوانی مقادیر `DOM` یا `Html` و استفاده از آن توسط `Javascript` را نشان می دهد.

The screenshot shows a browser window with developer tools open. On the left, there's a sidebar with a list of items: "Click Me", "Google", "Second", and "Third". Below this is a toolbar with "Elements", "Console" (which is selected), "Sources", "Network", "Performance", "Memory", "Application", and "Security". The main content area displays an "HTML Tree Generator" diagram and a "Node Details" panel. The tree diagram shows the structure of the page: HTML > HEAD (META, TITLE, LINK) > BODY (H1, INPUT, BUTTON, UL, SCRIPT). The UL node has three LI children, which each have an A child. The "Node Details" panel shows "Tag Name: SCRIPT", "Node ID: 0", and "# of Children: 0". Below the tree is a "Brought to you by Joel Saupe" link. At the bottom, the JavaScript console shows the following code and its execution:

```
> var heading = document.firstChild.lastElementChild.firstElementChild;  
  
< undefined  
> heading  
<  <h1>Hello</h1>  
> heading.innerHTML = "Good Bye";  
< "Good Bye"  
> heading.style.color = "red";  
< "red"
```

**Key words:** What is DOM, new concept, Properties and Method in objects, getter and setter, call method.

#### 4. Selecting HTML Elements with Javascript

در این قسمت روش های انتخاب یا `select` المنت ها توسط دستورات مختلف بررسی می شود.

- می توان با `TAG` ، `Class` و یا `ID` توسط دستورات زیر آنها را انتخاب نمود و تغییرات دلخواه را اعمال کرد.

```

▶ top
Filter

> document.getElementsByTagName("li");
< ▶ HTMLCollection(3) [li.item, li.item, li.item]
> document.getElementsByTagName("li").style.color = "purple";
✖ ▶ Uncaught TypeError: Cannot set property 'color' of undefined
  at <anonymous>:1:49
> document.getElementsByTagName("li")[2].style.color = "purple";
< "purple"
> document.getElementsByTagName("li").length;
< 3
> document.getElementsByClassName("btn");
< ▶ HTMLCollection [button.btn]
> document.getElementsByClassName("btn").style.color = "red";
✖ ▶ Uncaught TypeError: Cannot set property 'color' of undefined
  at <anonymous>:1:52
> document.getElementsByClassName("btn")[0].style.color = "red";|


> document.getElementById("title");
< <h1 id="title">Hello</h1>
> document.getElementById("title").innerHTML = "Good Bye";
< "Good Bye"

```

- در حالت متداول و برای راحتی، از دستورات `querySelector` برای دسترسی به کلاس ها و ID ها و تگ ها استفاده می شود کافیست به سلسله مراتب المنت ها دقیق کنید.

```

> document.querySelector("h1");
< <h1 id="title">Good Bye</h1>
> document.querySelector("#title");
< <h1 id="title">Good Bye</h1>
> document.querySelector(".btn");
< <button class="btn" style="color: red;">Click Me</button>
> document.querySelector("li a");

> document.querySelector("#list a");
< <a href="https://www.google.com">Google</a>
> document.querySelector("#list .item");
< ▶<li class="item">...</li>
> document.querySelectorAll("#list .item");
< ▶ NodeList(3) [li.item, li.item, li.item]
> document.querySelectorAll("#list .item")[2].style.color = "blue";
< "blue"

```

**Key words:** Selecting Element with Javascript, `querySelector`.

## 5. Manipulating and Changing Styles of HTML Elements with Javascript

در این قسمت نحوه فراخوانی خصوصیات موجود در CSS برای جاوااسکریپت را نشان می دهد.

- کدهای CSS در Javascript به صورت کد Camel می شوند.

```
> document.querySelector("h1").style.color = "red";
< "red"
> document.querySelector("h1").style.fontSize = "10rem"

> document.querySelector("button").style.backgroundColor = "yellow";
< "yellow"
>
```

Secure | [https://www.w3schools.com/jsref/dom\\_obj\\_style.asp](https://www.w3schools.com/jsref/dom_obj_style.asp)

CSS JAVASCRIPT SQL PHP BOOTSTRAP HOW TO JQUERY MORE ▾ REFERENCES ▾ EX/

The Style object represents an individual style statement.

### Style Object Properties

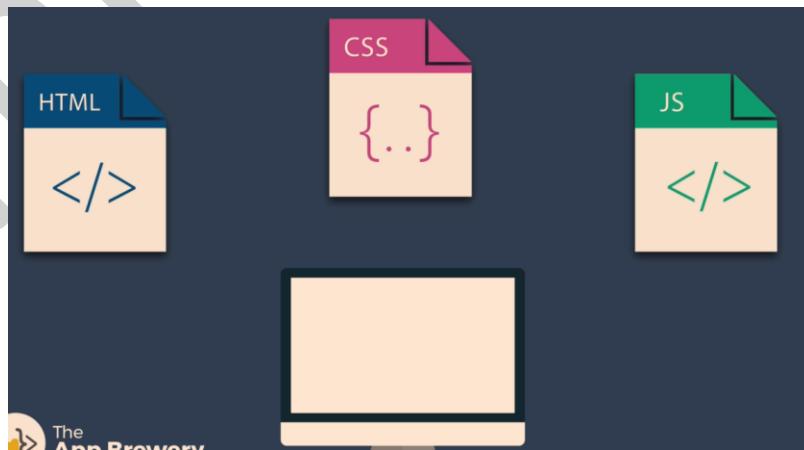
The "CSS" column indicates in which CSS version the property is defined (CSS1, CSS2, or CSS3).

Property	Description	CSS
<a href="#">alignContent</a>	Sets or returns the alignment between the lines inside a flexible container when the items do not use all available space	3
<a href="#">alignItems</a>	Sets or returns the alignment for items inside a flexible container	3
<a href="#">alignSelf</a>	Sets or returns the alignment for selected items inside a flexible container	3
<a href="#">animation</a>	A shorthand property for all the animation properties below, except the animationPlayState property	3

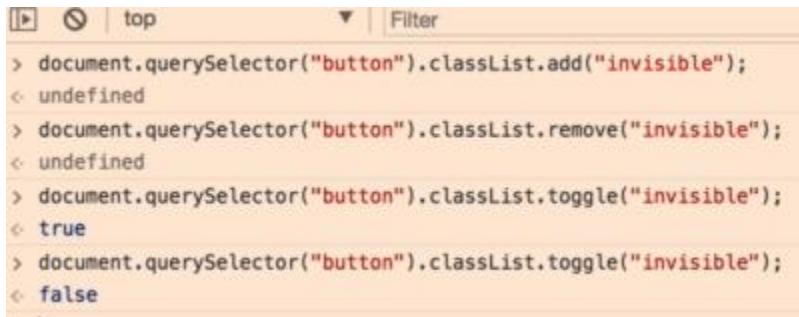
**Key words:** changing Style with Javascript, CSS camel mode in Javascript.

## 6. The Separation of Concerns Structure vs Style vs Behaviour

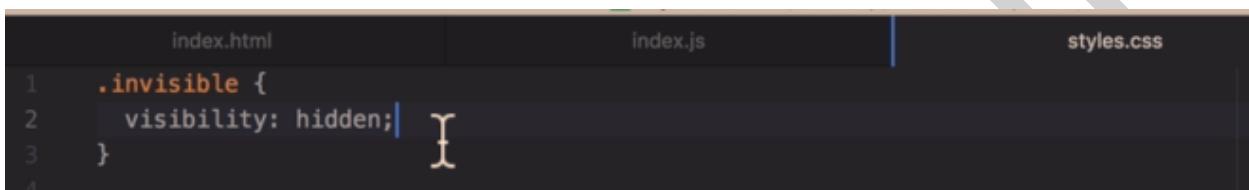
در این قسمت روش ایجاد تمایز بین کدهای javascript با HTML و CSS را نشان می دهد.



- برای ایجاد تمایز، می توان کلاس های مجزا برای javascript ایجاد کرد، که بتوان در موقع دلخواه آن را فعال و غیر فعال نمود و به ساختار CSS و HTML اصلی کاری نداشته باشیم.



```
> document.querySelector("button").classList.add("invisible");
< undefined
> document.querySelector("button").classList.remove("invisible");
< undefined
> document.querySelector("button").classList.toggle("invisible");
< true
> document.querySelector("button").classList.toggle("invisible");
< false
```

```
index.html index.js styles.css
1 .invisible {
2   visibility: hidden;
3 }
```

**Key words:** make a Separation of Structure vs Style vs Behaviour, classList, add, remove, toggle.

## 7. Text Manipulation and the Text Content Property

در این قسمت نحوه ایجاد تغییر متن Html با دستورات Javascript و innerHTML و textContent را بررسی می کند و تفاوت آنها را بیان می کند.

```
> document.querySelector("h1").innerHTML;
< "<strong>Hello</strong>"
> document.querySelector("h1").textContent;
< "Hello"
> document.querySelector("h1").innerHTML = "<em>Good Bye</em>";
< "<em>Good Bye</em>"
> document.querySelector("h1").innerHTML = "<em>Good Bye</em>";
```

**Key words:** Manipulation the Text Html with Javascript, innerHTML vs textContent.

## 8. Manipulating HTML Element Attributes

در این قسمت دستورات مورد نیاز برای تغییر attributes Javascript را نشان می دهد.

- به محتوای بین tag ها مانند href, class, Id و ... attribute گفته می شود.

```
> document.querySelector("a");
<   <a href="https://www.google.com">Google</a>
> document.querySelector("a").attributes;
<   ▶ NamedNodeMap {0: href, href: href, length: 1}
> document.querySelector("a").getAttribute("href");
<   "https://www.google.com"
> document.querySelector("a").setAttribute("href", "https://www.bing.com");
```

**Key words:** Manipulating HTML Element Attributes, get and set attributes.

## 9. Tip from Angela - The 20 Minute Method

همیشه شروع کردن هر کاری سخت می باشد، بنابراین بهترین کار اینه که شروع کنیم و حداقل ۲۰ دقیقه برای آن کار زمان بگذاریم، بعد خواهید دید که این ۲۰ دقیقه، انگیزه شما را زیاد می کند و بیشتر ادامه خواهید داد. در واقع انگیزه ای که از شروع کار بدست می آید، شما را به جلو حل می دهد.

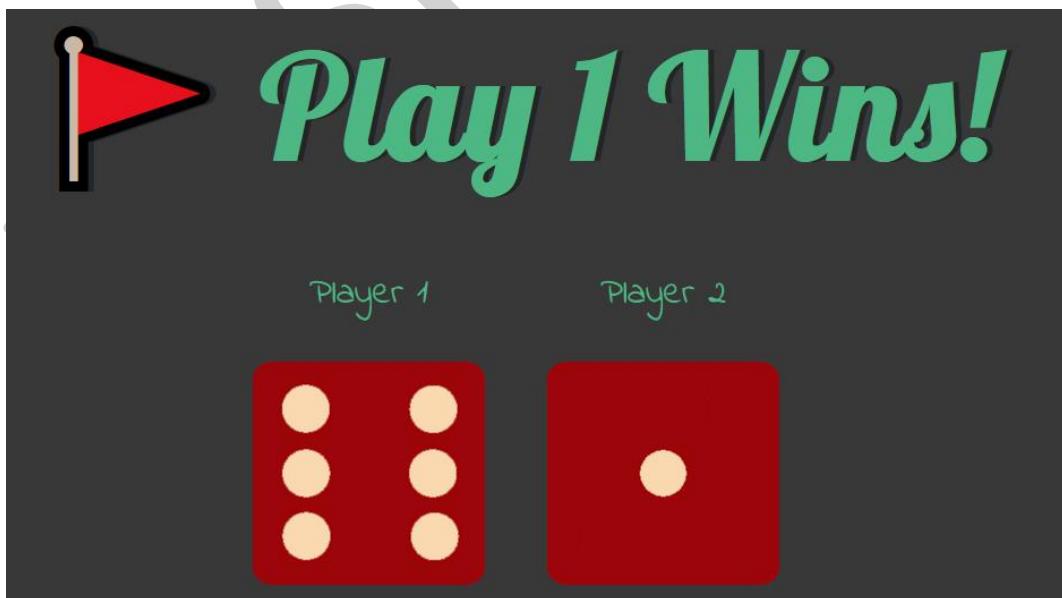
## 12. Boss Level Challenge 1 - The Dicee Game

### 1. Challenge The Dicee Challenge

در این قسمت صورت مسئله بیان می شود که با پد توسط دانشجو حل شود.

### 9. The Solution to the Dicee Challenge

در این قسمت راه حل مسئله بیان می شود و روش حل چالش را نشان می دهد.



## 11. Tip from Angela - Learning Before you Eat

برای افزایش تمرکز و عدم خواب آلدگی در حین یادگیری، سعی کنید، از خوردن غذاهای سنگین جلوگیری کنید و رژیم غذایی سبکی داشته باشید. برای مثال ناهار سنگین نخورید تا خواب آلد نشوید. در حالت کلی وقتی معده سبک باشد، مغز بهتر کار می کند.

## 13. Advanced Javascript and DOM Manipulation

### 1. What We'll Make Drum Kit

در این قسمت درباره پروژه نهایی این فصل و هدف این فصل صحبت می شود و اینکه چگونه می توان قابلیت تعامل با دکمه های کیبورد و موس در سایت ایجاد کرد.



**Key words:** describing about this season.

### 3. Adding Event Listeners to a Button

در این قسمت نحوه افزودن عملکرد کلیک موس بر روی دکمه های سایت را نشان می دهد که می توان بعد از هر کلیک، کدی یا تابعی در جاوااسکریپت اجرا شود.

- فراخوانی تابع به دو صورت می باشد، حالت اول `functionName ()` که مستقیماً تابع را فراخوانی می کند. حالت دوم `functionName` که بدون پرانتز می باشد و برای فراخوانی تابع برای بعد می باشد، به عنوان مثال زمانی که کلیک می کنیم رخ دهد.

```
• document.querySelector("button").addEventListener("click", handleClick);

function handleClick() {
  alert("I got clicked!");
}
```

- anonymous function ای که نام ندارد و به صورت () می باشد، function () به گفته می شود.

```
index.html index.js

1 var numberOfDrumButtons = document.querySelectorAll(".drum").length;
2
3 for (var i = 0; i < numberOfDrumButtons; i++) {
4
5   • document.querySelectorAll(".drum")[i].addEventListener("click", function () {
6     alert("I got clicked!");
7
8     //What to do when click detected.
9   });
10
11 }
12
13 }
```

**Key words:** Adding Event Listeners to a Button, anonymous function.

#### 4. Higher Order Functions and Passing Functions as Arguments

در این قسمت درباره تابع هایی که به ورودی تابع های دیگر داده می شوند صحبت می شود.

- در شکل زیر در حالت استفاده از anonymous function و استفاده از تابع معمولی به عنوان ورودی های تابع دیگر را نشان می دهد.

```
$0.addEventListener("click", function() {
  console.log("I got clicked");
});
```

```
$0.addEventListener("click", respondToClick);

function respondToClick() {
  console.log("I got clicked");
}
```

- در واقع `addEventListener` یک تابع Higher Order می باشد چون در خود، تابع دارد. یعنی زمانی که `input1` رخ داد، تابع `input2` را اجرا کن.

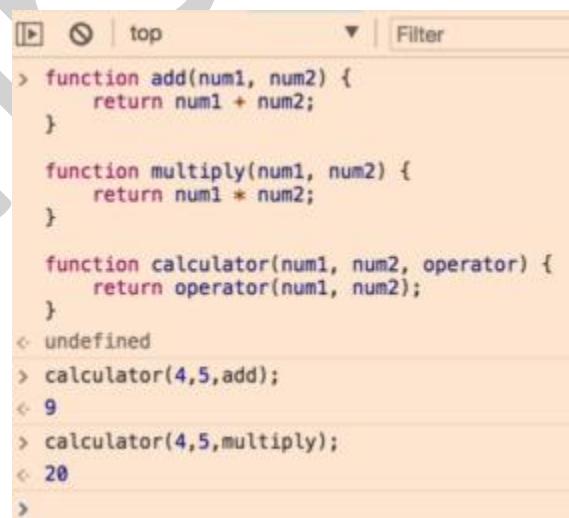
```
$0.addEventListener(input1, input2);

function respondToClick() {
  console.log("I got clicked");
}
```

- در شکل های زیر نحوه ایجاد تابع Higher order برای ایجاد یک ماشین حساب را نشان می دهد.

## Higher Order Functions

“Higher order functions are functions that can take other functions as inputs.”



```
function add(num1, num2) {
  return num1 + num2;
}

function multiply(num1, num2) {
  return num1 * num2;
}

function calculator(num1, num2, operator) {
  return operator(num1, num2);
}

calculator(4,5,add);
// 9

calculator(4,5,multiply);
// 20
```

- برای دیباگ کردن تابعی، می توان به شکل زیر عمل کرد.

```
> debugger;
calculator(3,4,multiply);
< 12
```

The screenshot shows a browser's developer tools debugger. The status bar at the bottom says "Debugger paused". The code pane shows a line starting with '> debugger;' followed by a call to 'calculator(3,4,multiply);' and the result '12'.

**Key words:** Higher Order Function, Function as Arguments or input, debugging.

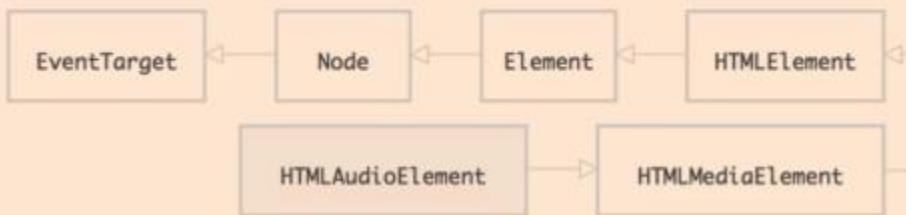
## 6. How to Play Sounds on a Website

- توسط کد زیر می توان متغیر صدا را ایجاد و play نمود.

```
var audio = new Audio("sounds/tom-1.mp3");
audio.play();
```

- در شکل زیر وراثت شی Audio را نشان می دهد که از بچه های HTMLMediaElementn می باشد و رفتار Play را از آن به ارث برده است.

The **HTMLAudioElement** interface provides access to the properties of <audio> elements, as well as methods to manipulate them. It derives from the **HTMLMediaElement** interface.



- برای ایجاد background از کد زیر در CSS استفاده می شود.

```
.w {
  background-image: url("images/tom1.png");
}
```

- شکل زیر نحوه استفاده از this را نشان می دهد. This یک متغیر موقت می باشد که هر جایی باشد، اطلاعات آن مکان را در خود نگه می دارد. در اینجا اطلاعات Button، کلیک شده را در خود نگه می دارد، در نتیجه می توان تغییرات دلخواه را بر شی اعمال نمود.

```
this.style.color = "white";
```

```
for (i = 0; i < numberDrumButtons; i++) {  
  
    document.querySelectorAll(".drum")[i].addEventListener("click", function() {  
  
        console.log(this);  
  
    })  
  
    <button class="w drum">W</button>  
    <button class="a drum">A</button>  
    <button class="s drum">S</button>  
    <button class="d drum">D</button>  
    <button class="j drum">J</button>  
    <button class="k drum">K</button>  
    <button class="l drum">L</button>
```



**Key words:** play sound, inheritance, “this” property.

## 7. A Deeper Understanding of Javascript Objects

در این قسمت درباره شی گرایی و نحوه ایجاد object در جاوا اسکریپت صحبت می شود.

# JavaScript Objects

Dominating the Web, One Object at a Time.

- برای ایجاد یک متغیر Bell Boy که دارای خصوصیت یا Properties می باشد، باید مراحل زیر را طی کنیم.

```
var bellBoy1Name = "Timmy";  
var bellBoy1Age = 19;  
var bellBoy1HasWorkPermit = true;  
  
var bellBoy1Languages = [  
    "French",  
    "English"  
]
```

- حال اگر بخواهیم چند Bell Boy مطابق جدول ایجاد کنیم، اگر بخواهیم مطابق بالا پیش برویم، کلی کد باید بنویسیم.

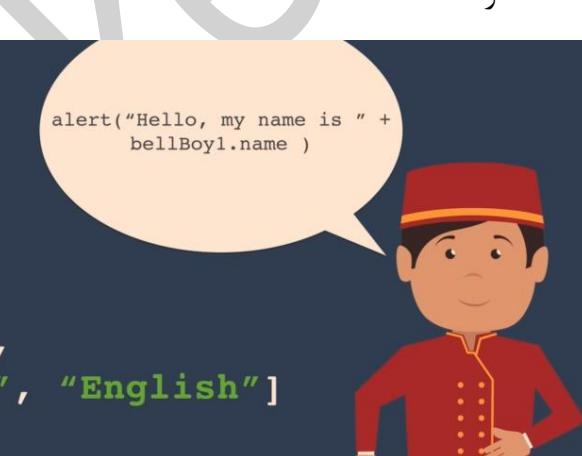
Properties	bellBoy1	bellBoy2	bellBoy3
name	"Timmy"	"Jimmy"	"Bitty"
age	19	21	18
hasWorkPermit	TRUE	FALSE	TRUE
languages	["French", "English"]	["English", "German"]	["English"]



- در نتیجه می توان از object استفاده کرد، و مطابق شکل زیر یک object برای Bell Boy ایجاد کرد.

## Object

```
var bellBoyl = {
  name: "Timmy",
  age: 19,
  hasWorkPermit: true,
  languages: ["French", "English"]
}
```



```
alert("Hello, my name is " + bellBoyl.name )
```

- در ادامه برای راحتی بیشتر در ایجاد object از constructor function استفاده می کنیم.

## Constructor Function

```
function BellBoy (name, age, hasWorkPermit, languages) {
  this.name = name;
  this.age = age;
  this.hasWorkPermit = hasWorkPermit;
  this.languages = languages;
}
```

- و برای ایجاد و دادن مقدار به خصوصیات object ، مطابق شکل زیر عمل می کنیم.

## Initialise Object

```
var bellBoy1 = new BellBoy("Timmy", 19, true, ["French", "English"]);  
var bellBoy1 = new BellBoy("Jimmy", 21, false, ["English", "German"]);
```

•  
•  
•

- نکته: نام گذاری و فراخوانی Constructor Object با حروف اول بزرگ می باشد.

- نکته: برای فراخوانی کافیست، از نقطه به شکل ObjectName.Properties استفاده کنیم.

- رفتارها و Properties خصوصیات object Methods می باشد.

**Key words:** create an object, object oriented, properties, constructor object.

## 8. How to Use Switch Statements in Javascript

در این قسمت در ادامه پروژه فصل، برای به صدا در آوردن تک تک طبل ها، از Switch استفاده می کند.

- در اینجا استفاده از switch راحت تر و بهینه تر از if else می باشد.

- عبارت default مانند else می باشد.

```

var buttonInnerHTML = this.innerHTML;

switch (buttonInnerHTML) {
    case "w":
        var tom1 = new Audio("sounds/tom-1.mp3");
        tom1.play();
        break;

    case "a":
        var tom2 = new Audio("sounds/tom-2.mp3");
        tom2.play();
        break;

    default:
}

```

**Key words:** Switch Statement instead if else statement.

## 9. Objects, their Methods and the Dot Notation

در این قسمت درباره نحوه افزودن method یا رفتار به object صحبت می شود.

- هر object می تواند رفتارها یا Methods داشته باشد که در واقع تابع های مختص آن object می باشد.

## Methods

```

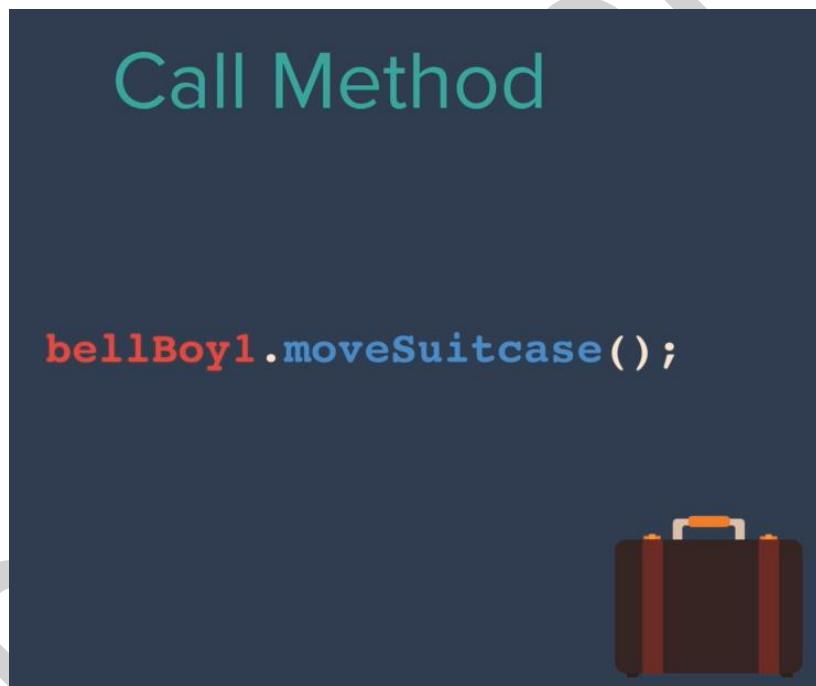
function moveSuitcase {
    alert("May I take your suitcase?");
    pickUpSuitcase();
    move();
}

```

- در شکل زیر نحوه افزودن method به object را نشان می دهد.

```
var bellBoy1 = {
    name: "Timmy",
    age: 19,
    hasWorkPermit: true,
    languages: ["French", "English"],
    moveSuitcase: function() {
        alert("May I take your suitcase?");
        pickUpSuitcase();
        move();
    }
}
```

- نحوه فراخوانی آن به شکل زیر می باشد.



- همچنین می توان مطابق شکل زیر، از Constructor function برای ایجاد method استفاده نمود.

# Constructor Function

```
function BellBoy (name, age, hasWorkPermit, languages) {  
    this.name = name;  
    this.age = age;  
    this.hasWorkPermit = hasWorkPermit;  
    this.languages = languages;  
    this.moveSuitcase = function() {  
        alert("May I take your suitcase?");  
        pickUpSuitcase();  
        move();  
    }  
}
```

- در نتیجه object های می توانند از قبل ساخته شده باشند مانند Audio در شکل زیر، یا می توان object هایی با خصوصیات و رفتار دلخواه ایجاد کرد.

# Constructor Function

```
function Audio (fileLocation) {  
    this.fileLocation = fileLocation;  
    this.play = function() {  
        //Tap into the audio hardware  
        //Check the file at fileLocation exists  
        //Check the file at fileLocation is a sound file  
        //Play the file at fileLocation  
    }  
}  
  
var tom1 = new Audio("sounds/tom-1.mp3");  
tom1.play();
```

**Key words:** create method for objects.

## 11. Using Keyboard Event Listeners to Check for Key Presses

در این قسمت نحوه بدست آوردن دکمه کیبورد فشرده شده، را نشان می دهد.

- کافیست کل صفحه را توسط **keydown** فعال کنیم، در ادامه پس از فشردن دکمه کیبورد، اطلاعات را ذخیره می کند و پاس می دهد (Callback function) که حرف مورد نظر در قسمت قرار دارد که به ورودی تابع داده می شود.

```
// detecting key
document.addEventListener("keydown", function(event) {
    makeSound(event.key);
});
```

**Key words:** Checking key presses with Listener.

## 12. Understanding Callbacks and How to Respond to Events

در این قسمت درباره **Callbacks function** صحبت می شود:

- همان طور می که می دانیم، تابعی که در ورودی اش تابع بپذیرد، به عنوان **Higher order** شناخته می شود که **addEventListener** یک نمونه از آن می باشد.

## Higher Order Function

```
document.addEventListener("keypress", respondToKey(event));

function respondToKey(event){
    console.log("Key pressed.");
}
```

- حال اگر بتوان یک ورودی مانند **event** (که در صورت فشردن دکمه ای (**keypress**) پاس داده شده است) برای تابع ورودی در نظر گرفت، به آن **callback function** می گویند. یعنی **callback function** (درخواست) می باشد و **call "keypress"** (جواب درخواست) یک **callback function** (درخواست) می باشد.

# Callback Function

```
document.addEventListener("keypress", respondToKey(event));  
  
function respondToKey(event){  
    console.log("Key pressed.");  
}
```

- در شکل زیر یک نمونه callback function ایجاد شده است تا مفهوم آن قابل درک باشد و بتوان از آن استفاده و یا آن را برای موارد دلخواه ایجاد کرد.

```
> function anotherAddEventListener(typeOfEvent, callback) {  
    //Detect Event Code...  
  
    var eventThatHappened = {  
        eventType: "keypress",  
        key: "p",  
        durationOfKeypress: 2  
    }  
  
    if (eventThatHappened.eventType === typeOfEvent) {  
        callback(eventThatHappened);  
    }  
}  
}  
< undefined  
> anotherAddEventListener("keypress", function(event) {  
    console.log(event);  
});  
► {eventType: "keypress", key: "p", durationOfKeypress: 2}  
< undefined  
>
```

- در شکل زیر یک نمونه دیگر از این تابع ایجاد و فراخوانی شده است. (به respond() و respond() توجه کنید). اسم های متغیر دلخواه می باشد.

```
Script snippet #1 × (index) lazy_load.js  
1 function callback (action, respond){  
2  
3     if (action === 2){  
4         respond(true);  
5     }  
6 }  
7  
8 callback (2, function(myRespond){  
9     console.log(myRespond);  
10});  
11
```

**Key words:** Callback function, new and important concept.

### 13. Adding Animation to Websites

در ادامه پروژه فصل، در این قسمت با تعویض کلاس المنت، باعث تعویض آن می شود و پس از یک Delay به حالت قبل بر می گردد. در نتیجه باعث ایجاد انیمیشن بر روی دکمه ها می شود.

```
function buttonAnimation(currentKey) {  
  
    var activeButton = document.querySelector("." + currentKey);  
  
    activeButton.classList.add("pressed");  
  
    setTimeout(function() {  
        activeButton.classList.remove("pressed");  
    }, 100);  
}
```

**Key words:** adding delay or Timeout, adding animation to object.

### 15. Tip from Angela - Dealing with Lack of Progress

در این مرحله ممکن است فکر کنید که خیلی چیزهای درس های اولیه را فراموش کردید. این طبیعی است و شما کافیست درس ها اولیه را مرور کنید تا در ذهن شما بشینند. مرور و تمرین است که باعث یادگیری و ماندگاری می شود و هیچ راه میانبری برای آن وجود ندارد. یادمان باشد ما همه چی می دانیم ولی هیچی نمی دانیم.

## 14. jQuery

### 1. What is jQuery

jQuery یکی از کتابخانه های معروف javascript می باشد که باعث راحت تر و کوتاه تر و خواناتر شدن دستورات javascript می شود.

# jQuery

The JavaScript Library that will Prevent Our Fingers From Breaking

jQuery("h1")

document.querySelector("h1")

Vs.

\$("h1")

**Key words:** what is jQuery.

## 2. How to Incorporate jQuery into Websites

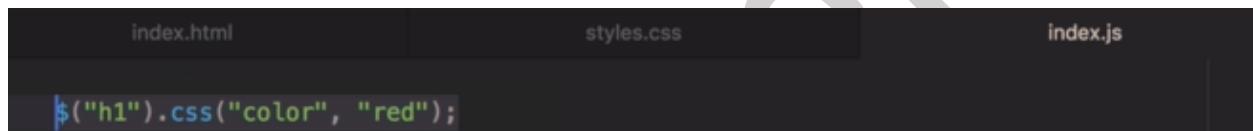
در این قسمت نحوه افزودن کتابخانه jQuery به پروژه را نشان می دهد.

- کافیست از طریق افزودن لینک [jQuery google CDN](#) به انتهای کد `html` و قبل از کتابخانه `javascript` ، `jquery` را به پروژه مان اضافه کنیم. ( حتماً باید قبل از `javascript` قرار بگیرد)

```

1  <!DOCTYPE html>
2  <html lang="en" dir="ltr">
3  <head>
4      <meta charset="utf-8">
5      <title>jQuery</title>
6      <link rel="stylesheet" href="styles.css">
7  </head>
8  <body>
9
10 <h1>Hello.</h1>
11 <button>Click Me</button>
12 <button>Click Me</button>
13 <button>Click Me</button>
14 <button>Click Me</button>
15 <button>Click Me</button>
16
17     <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
18     <script src="index.js" charset="utf-8"></script>
19
20 </body>
21 </html>

```

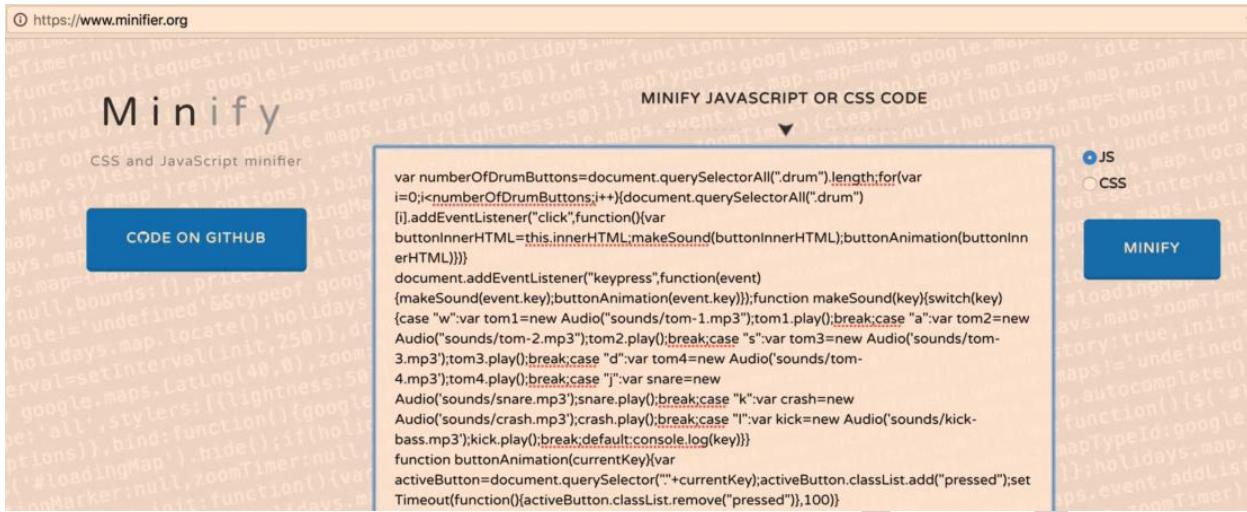


**Key words:** Add jQuery to our project.

### 3. How Minification Works to Reduce File Size

در این قسمت درباره Minify کردن صحبت می شود. یعنی می توانیم سایز فایل های CSS و JS و HTML خود را با حذف فاصله های موجود در کد، کاهش دهیم تا سرعت بارگذاری سایت افزایش یابد.

- برای Minify کردن می توان از سایت [www.minifier.org](http://www.minifier.org) استفاده نمود.
- نکته: اگر لینک کتابخانه های bootstrap و jQuery را باز کنید، متوجه Minify خواهید شد.



**Key words:** Minify JS and CSS and HTML files to reduce file size.

#### 4. Selecting Elements with jQuery

- نحوه انتخاب در jQuery به شکل زیر می باشد و از \$ و نام المنت متوجه (مورد نظر) مثل نام کلاس، id و... همانند قبل استفاده می کند.

```
document.querySelector("h1");
$("h1");

• document.querySelectorAll("button");
$("#button");
```

**Key words:** Selecting Elements with jQuery.

#### 5. Manipulating Styles with jQuery

- با دستور `$("h1").css("color");` می توان مقدار یک خصوصیت یا style مثل رنگ، فونت و غیره را از CSS بدست آورد.

- با دستور `$(“h1”).css(“color”, “green”);` می توان به آن خصوصیت مقدار جدید داد.

- همان طور که می دانیم، تغییر style باید در CSS صورت گیرد نه در جاوا اسکریپت. در نتیجه از

`$(“h1”).addClass(“big-title margin-50”);` دستور همانند قبل کلاس های مورد نظر را به المنت متوجه اضافه می کنیم.

- با دستور `$(“h1”).hasClass(“margin-50”);` می توان وجود کلاس مورد نظر در یک المنت را بررسی نمود.

**Key words:** Manipulating Styles with jQuery.

## 6. Manipulating Text with jQuery

- توسط دستور `$("h1").text("Bye");` می توان متن یک المنت یا چندین المنت را تغییر داد.  
- به جای دستور `$("button").html("<em>Hey</em>");` درjQuery استفاده می شود. (به تفاوت `html` و `text` دقت کنید)

**Key words:** Manipulating Text with jQuery.

## 7. Manipulating Attributes with jQuery

در این قسمت درباره تغییر attributes in HTML توسط jQuery صحبت می شود.

- برای انجام دستور `getter` به شکل `$("a").attr("href");` و برای انجام `setter` به شکل `$("a").attr("href", "https://www.yahoo.com");` در attributes اقدام می کنیم.

**Key words:** Manipulating Attributes of HTML

## 8. Adding Event Listeners with jQuery

- توسط کد شکل زیر می توان event listener click مخصوص `jQuery` را ایجاد کرد.

```
$("h1").click(function() {  
    $("h1").css("color", "purple");  
});
```

- در شکل زیر قبل و بعد از استفاده از `jQuery` برای event listener `click` مجموعه ای از button ها را نشان می دهد.

```

1
2   for (var i = 0; i<5; i++) {
3     document.querySelectorAll("button")[i].addEventListener("click", function() {
4       document.querySelector("h1").style.color = "purple";
5     });
6   }
7
8   $("button").click(function() {
9     $("h1").css("color", "purple");
10  });

```

- در شکل زیر keypress event listener مختص صفحه را نشان می دهد.

<code>\$("body").keypress(function(event) {   console.log(event.key); });</code>	<code>\$document.KeyPress(function(event) {   \$("h1").text(event.key); });</code>
------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------

- همچنین برای event listener می توان از تابع `on(yourEvent, callbackFunction)` استفاده نمود.

```

$("h1").on("mouseover", function() {
  $("h1").css("color", "purple");
});

```

**Key words:** Adding Event Listeners with jQuery.

## 9. Adding and Removing Elements with jQuery

- برای افزودن المتن جدید، قبل از یا بعد از یا داخل یک المتن دیگر همانند شکل زیر عمل می کنیم.

```

> $("h1").before("<button>New</button>");
< ► w.fn.init [h1.big-title.margin-50, prevObject: w.fn.init(1)]
> $("h1").after("<button>New</button>");
< ► w.fn.init [h1.big-title.margin-50, prevObject: w.fn.init(1)]
> $("h1").prepend("<button>New</button>");
< ► w.fn.init [h1.big-title.margin-50, prevObject: w.fn.init(1)]
> $("h1").append("<button>New</button>");
< ► w.fn.init [h1.big-title.margin-50, prevObject: w.fn.init(1)]

```

- برای حذف المتن به شکل `> $("button").remove();` عمل می کنیم.

**Key words:** Adding and Removing Elements with jQuery.

## 10. Website Animations with jQuery

در این قسمت درباره ایجاد اینیمیشن بر روی المنت های مورد نظر توسط jQuery صحبت می شود.

- برای ایجاد حالات اینیمیشن از دستورات jQuery Effect استفاده می شود.

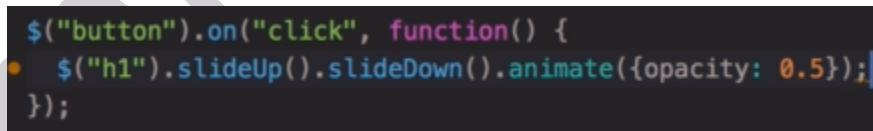
The screenshot shows the left sidebar of the jQuery website with the 'jQuery Effects' category highlighted. The main content area is titled 'jQuery Effect Methods' and contains a table listing various animation methods:

Method	Description
<a href="#">animate()</a>	Runs a custom animation on the selected elements
<a href="#">clearQueue()</a>	Removes all remaining queued functions from the selected elements
<a href="#">delay()</a>	Sets a delay for all queued functions on the selected elements
<a href="#">dequeue()</a>	Removes the next function from the queue, and then executes the function
<a href="#">fadeIn()</a>	Fades in the selected elements
<a href="#">fadeOut()</a>	Fades out the selected elements
<a href="#">fadeTo()</a>	Fades in/out the selected elements to a given opacity

- در شکل های زیر چند نمونه از نحوه استفاده از این دستورات را مشاهده می کنید.



- می توان چندین دستور را به ترتیب اجرا ، پشت هم نوشت.



**Key words:** jQuery Effects.

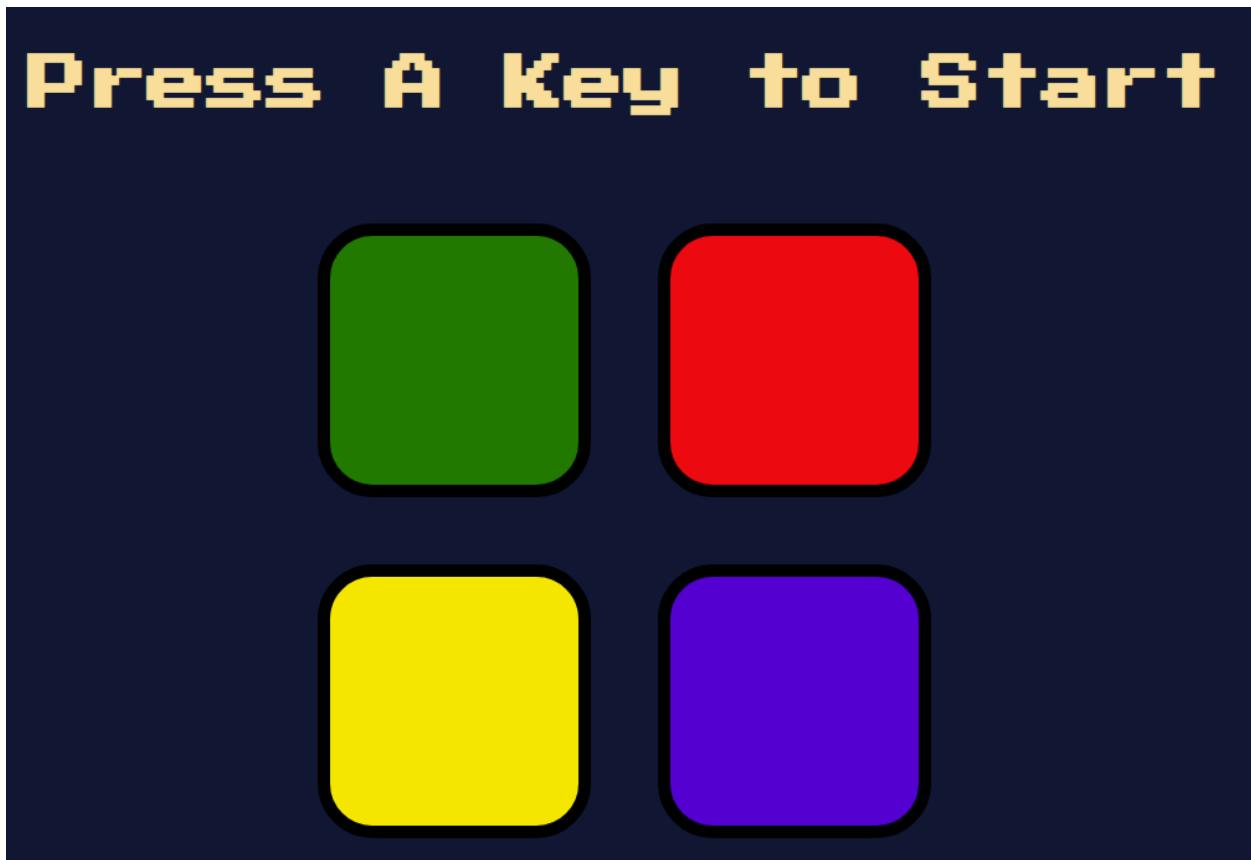
## 11. Tip from Angela - Mixing Knowledge

هر فصل این دوره، دارای مفهوم های مجزا می باشد. برای اینکه بتوان این مفهوم ها را به هم وصل کرد و باعث یکپارچه سازی در ذهنتان شود، باید خود را به چالش بکشید و سعی کنید از مفاهیم مختلف در پروژه استفاده کنید. در نتیجه باعث ماندگاری و یکپارچه سازی کل مفاهیم در ذهنتان خواهد شد.

## 15. Boss Level Challenge 2 - The Simon Game

### 1. What You'll Make The Simon Game

در این فصل یک چالش دیگر برای جمع بندی و درک مفاهیم ایجاد می شود که در نهایت چالش با موفقیت انجام شد.



### 25. Tip from Angela - Dealing with Frustration

اگر در پروژه و کدهایتان گیر کرده اید و دچار به هم ریختگی و نا امیدی شده اید، بهترین کار این است که نفس عمیق بکشید و به خودتان استراحت دهید و بعد از مدتی دوباره برگردید یا اصلاً یک روز دیگر برای حل این مشکل وقت بگذارید. در این صورت به نتیجه خواهید رسید و گیر کردن برای مدت طولانی در کد، مشکلی را برطرف نخواهد کرد. بهترین کار استراحت مغز می باشد.

## 16. The Unix Command Line

### 2. Command Line Hyper Setup

در این قسمت نحوه نصب و پیکربندی git و hyper terminal در ویندوز را نشان می دهد.

**Key words:** installation Git and Hyper terminal.

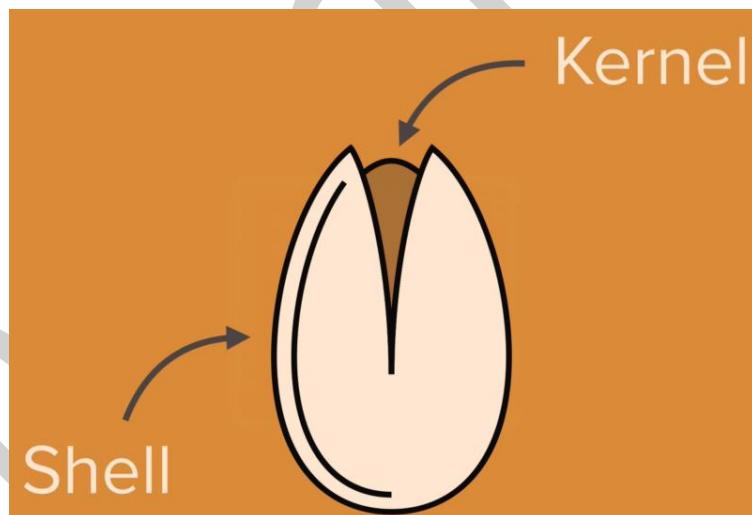
### 3. Understanding the Command Line. Long Live the Command Line!

در این قسمت علت استفاده از ترمینال یا command line را توضیح می دهد.

- هر سیستم عاملی برای دسترسی به سخت افزار، یک هسته یا kernel دارد و معروف ترین آن Unix kernel می باشد که در سیستم عامل های لینوکس، اندروید، مک و سرورهای سایت ها و... از این هسته استفاده می شود.

- هر kernel دارای shell می باشد. Shell قابلیت کنترل و دسترسی به هسته را می دهد. حال استفاده از shell به دو صورت command line (ترمینالی) و گرافیکی (GUI) (مانند کارهایی که در ویندوز می کنیم) می باشد.

- برای یک برنامه نویس، ضروری است که به صورت command line با هسته ارتباط داشته باشد. چونکه ۱- قابلیت انعطاف پذیری بیشتری دارد، ۲- سرعت بیشتری دارد، ۳- دسترسی بیشتری دارد.



- از آنجا که سرورها در UNIX kernel هستند، باید از shell ترمینالی آن که Bash نامیده می شود استفاده کنیم.

- از آنجا که سیستم عامل ویندوز بر روی UNIX سوار نیست، در نتیجه از Git Bash برای شبیه سازی هسته UNIX استفاده می کنیم. ولی مثلاً در سیستم مک نیازی به نصب Git نیست، چون هسته آن UNIX و در نتیجه ترمینال اصلی آن Bash می باشد.

# Bash = Bourne Again Shell

یک افزونه یا برنامه اضافی برای Bash می باشد که قابلیت کنترل و انعطاف در محیط ترمینال را برای ما بیشتر می کند.



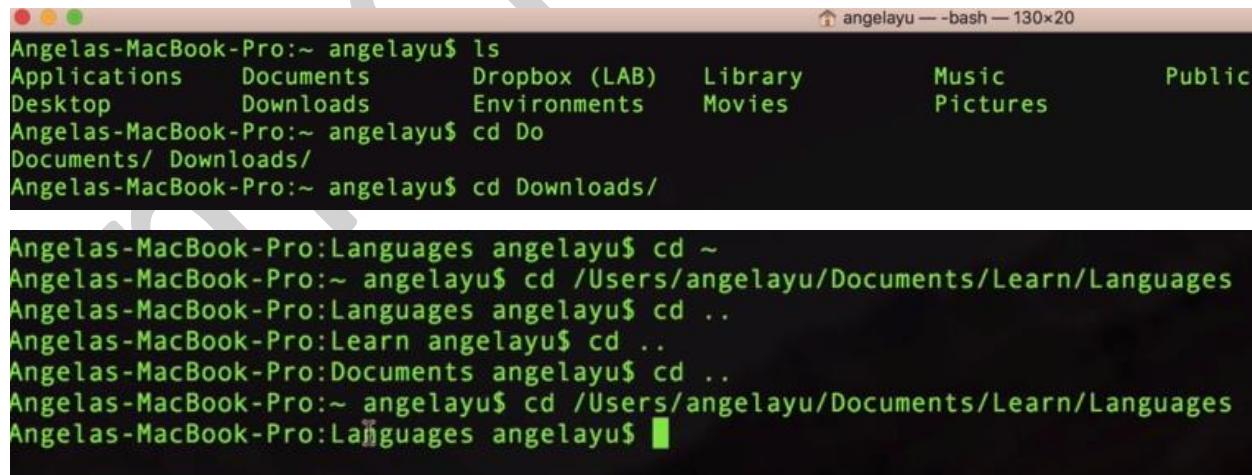
```
OnlineCourse Desktop — -bash — 80x24
Angelas-MacBook-Pro:desktop angelayu$ ls -a
.
..
.DS_Store .SecretStuff App Development
Angelas-MacBook-Pro:desktop angelayu$ mkdir .Secrets
Angelas-MacBook-Pro:desktop angelayu$ ls -a
.
..
.DS_Store .Secrets
..
.SecretStuff App Development
Angelas-MacBook-Pro:desktop angelayu$
```

**Key words:** command line, git bash, UNIX kernel, new concepts.

## 4. Command Line Techniques and Directory Navigation

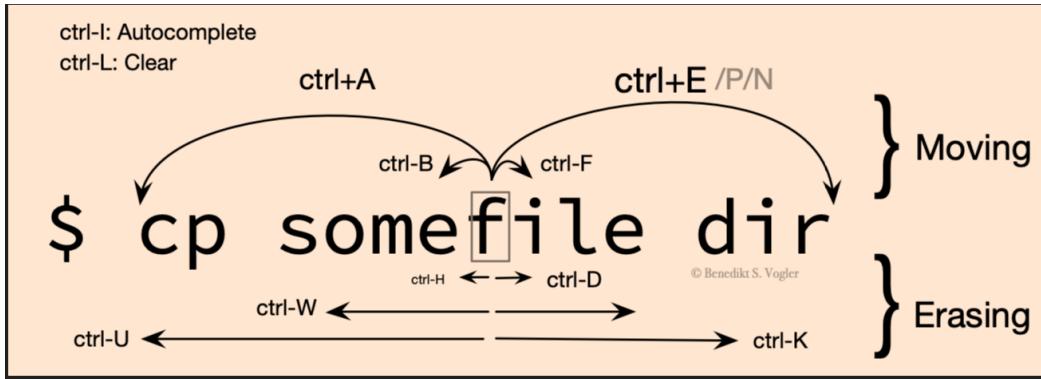
در این قسمت دستورات تغییر مکان یا جا به جایی در bash توضیح داده می شود.

- در حالت پیش فرض مسیر اصلی ترمینال به صورت user/userName می باشد که به این مسیر root می گویند که علامت آن "~" یا tilde می باشد.
- با دستور ls لیست فایل های موجود در مکان فعلی و با دستور cd می توان جهت جابه جایی مکانی استفاده کرد مانند ~ cd و .. که به ترتیب برای برگشت به root و برگشتن به یک مرحله قبل استفاده می شود.



```
OnlineCourse angelayu — -bash — 130x20
Angelas-MacBook-Pro:~ angelayu$ ls
Applications Documents Dropbox (LAB) Library Music Public
Desktop Downloads Environments Movies Pictures
Angelas-MacBook-Pro:~ angelayu$ cd Do
Documents/ Downloads/
Angelas-MacBook-Pro:~ angelayu$ cd Downloads/
Angelas-MacBook-Pro:Languages angelayu$ cd ~
Angelas-MacBook-Pro:~ angelayu$ cd /Users/angelayu/Documents/Learn/Languages
Angelas-MacBook-Pro:Languages angelayu$ cd ..
Angelas-MacBook-Pro:Learn angelayu$ cd ..
Angelas-MacBook-Pro:Documents angelayu$ cd ..
Angelas-MacBook-Pro:~ angelayu$ cd /Users/angelayu/Documents/Learn/Languages
Angelas-MacBook-Pro:Languages angelayu$
```

- از کلید میانبر های TAB برای نوشتن بقیه دستور، از پیکان های بالا و پایین جهت دسترسی به دستورات قبلی استفاده می شود. سایر کلیدهای میانبر ترمینال در شکل زیر نشان داده شده است.



**Key words:** Command Line Techniques and Directory Navigation.

## 5. Creating, Opening, and Removing Files through the Command Line

در این قسمت دستورات ایجاد، بازکردن، حذف کردن فایل یا پوشه در Bash توضیح داده می شود.

- دستور `pwd` برای بدست آوردن مسیر فعلی استفاده می شود. دستورات `start`, `touch` برای ایجاد فایل و باز کردن آن استفاده می شود. دستور `rm *` همه فایل های موجود را پاک می کند.
- دستور `* rm -r` همه فولدرها را پاک می کند. دستور `mkdir` فولدر ایجاد می کند.

MohammadReza@MohammadReza MINGW64 ~ \$ pwd /c/Users/MohammadReza	MohammadReza@MohammadReza MINGW64 ~/shaker \$ mkdir s1
MohammadReza@MohammadReza MINGW64 ~/shaker \$ rm -r *	MohammadReza@MohammadReza MINGW64 ~/shaker \$ touch tesss.txt

**Key words:** Creating, Opening, and Removing Files through the Command Line.

## 6. Tip from Angela - Sleep is My Secret Weapon

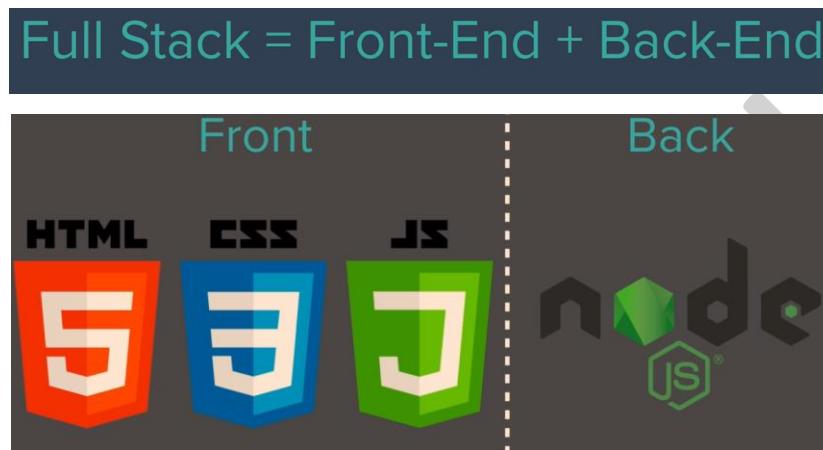
یکی از مسائل مهم در انرژی و میزان خلاقیت در طول روز، داشتن خواب کافی و موثر در طول شب می باشد. بنابراین سعی نکنید که کمتر بخوابید تا زمان بیشتری داشته باشید، بلکه خواب کافی یکی از اصول اولیه ترمیم معز و بدن می باشد و هیچ دارویی مانند خواب وجود ندارد. یکی از راه حل های خواب خوب، داشتن ساعت خواب می باشد، یعنی یک ساعت قبل از خوابیدن ساعت هشدار دهد و شما باید آماده خوابیدن شوید و مانیتور و سایر مشکلات را کنار بگذارید.

## 17. Backend Web Development

### 1. Backend Web Development Explained

در این قسمت درباره full stack development صحبت می شود.

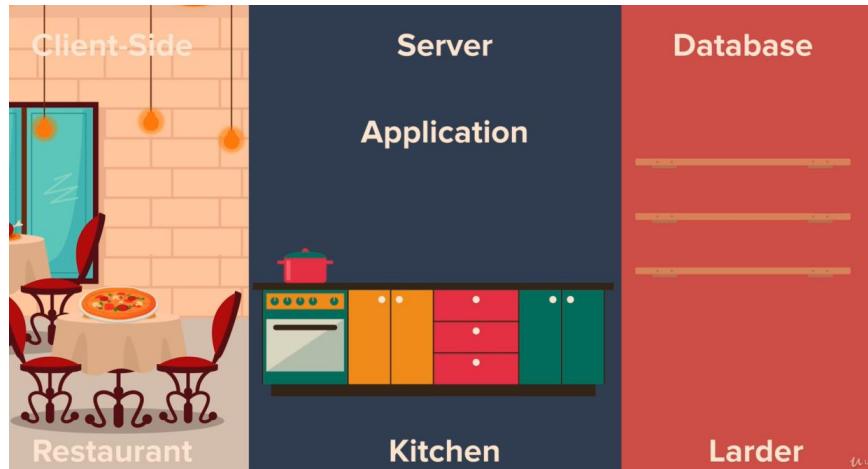
- فضولی که تا کنون درس داده شد، برای front-end بود و در ادامه فضول به بخش دوم دوره یعنی Back-end پرداخته می شود.



Shامل ۳ بخش application و database server می شود -



وظیفه نگهداری اطلاعات را دارد. Application همان منطق یا تابع های مخصوص سایت می باشد. Server وظیفه اجرای عملیات application را دارد که آن را تحويل کاربر نهایی یا front-end می دهد.



برای application در back-end از زبان های برنامه نویسی مختلفی (مثل java, python, php و...) استفاده می شود. در این دوره ما از محیط node.js برای back-end استفاده می کنیم که از javascript استفاده می کند.



**Key words:** backend concept.

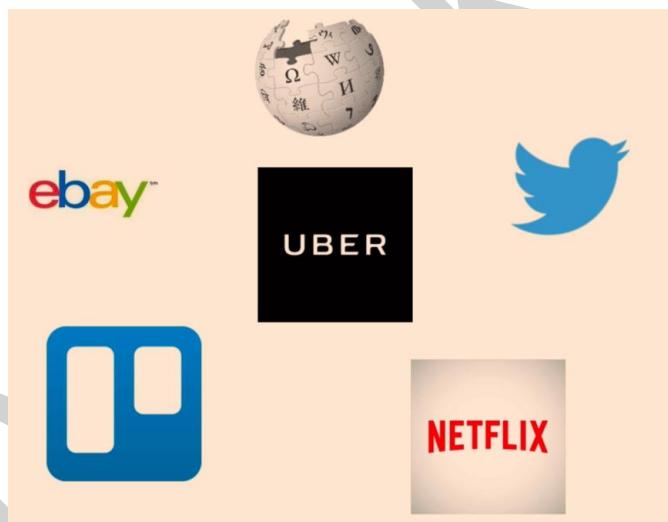
## 18. Node.js

### 1. What is Node.js

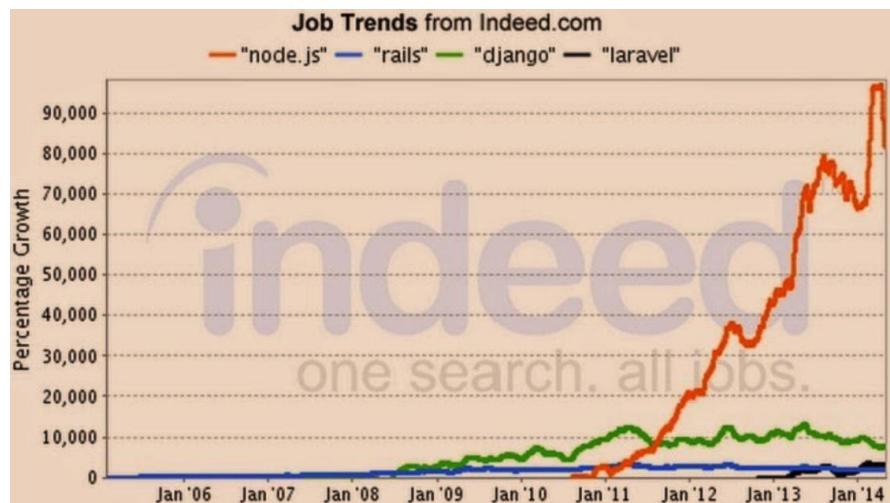
در این قسمت علت برتری Back-end Node.js در سایر زبان ها را مورد بررسی قرار می دهد.



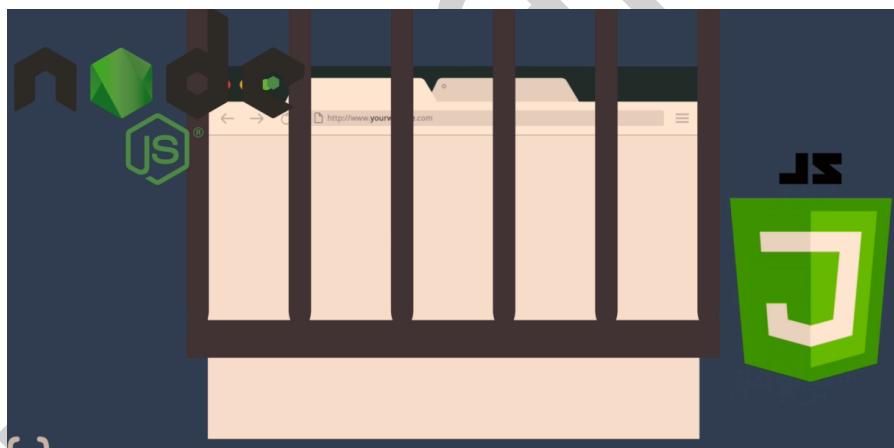
- از آنجا که node.js بر پایه جاوا اسکریپت می باشد، در نتیجه نیاز به یادگیری زبان جدیدتر نداریم.
- Node.js یک زبان برنامه نویسی نیست، بلکه یک محیط برای اجرای javascript می باشد که این قابلیت را می دهد، در هر سیستمی که Node.js نصب شود، قابلیت اجرای Javascript وجود خواهد داشت. در واقع node.js برای آزاد کردن javascript از مرورگرها ابداع شد.



- جزء ترندها در چند سال اخیر می باشد.



فقط مختص مرورگر و تعامل با کاربر می باشد، در حالی که در زبان های back-end این امکان وجود دارد که به فایل های محلی و سخت افزار دسترسی مستقیم داشته باشیم. در نتیجه به کمک node.js می توان برنامه سیستم عامل نوشت. مثلاً برنامه Atom توسط node.js ساخته شده است. در نهایت می توان این application را در سمت سرور قرار داد تا به فایل و دیتابیس دسترسی داشته باشد و قابلیت اجرا داشته باشد و اطلاعات را به front انتقال دهد.



**Key words:** Node.js benefits and survey.

#### 4. The Power of the Command Line and How to Use Node

در این قسمت دستورات bash را مرور کلی می کند و در نهایت یک فایل JS را توسط node در ترمینال اجرا می کند.

```

Angelas-MacBook-Pro:~ angelayu$ pwd
/Users/angelayu
Angelas-MacBook-Pro:~ angelayu$ cd Desktop/
Angelas-MacBook-Pro:Desktop angelayu$ pwd
/Users/angelayu/Desktop
Angelas-MacBook-Pro:Desktop angelayu$ ls
App Development Web Development
Angelas-MacBook-Pro:Desktop angelayu$ mkdir intro-to-node
Angelas-MacBook-Pro:Desktop angelayu$ cd intro-to-node/
Angelas-MacBook-Pro:intro-to-node angelayu$ ls
Angelas-MacBook-Pro:intro-to-node angelayu$ touch index.js
Angelas-MacBook-Pro:intro-to-node angelayu$ ls
index.js
Angelas-MacBook-Pro:intro-to-node angelayu$ node index.js
Hello, world!
Angelas-MacBook-Pro:intro-to-node angelayu$ 

```

**Key words:** run the javascript code with node.js in shell.

## 5. The Node REPL (Read Evaluation Print Loops)

یکی از ماثول های node می باشد که همانند شکل زیر می توان کدهای جاوا اسکریپت را به صورت مفسری در ترمینال نوشته و اجرا نمود. ( همانند chrome developer console در

```

Angelas-MacBook-Pro:intro-to-node angelayu$ node
> console.log("Hey there!");
Hey there!
undefined
> 3+5
8
> "Angela " + "Yu"
'Angela Yu'
> con
const      continue

console

constructor

```

- در ترمینال کافیست node را تایپ کنید تا وارد REPL شوید و دستورات جاوا اسکریپتی را اجرا کنید.
- برای خروج در محیط ترمینال از هر برنامه ای کافیست از exit. یا دو بار Ctrl + C استفاده کنید و برای پاکسازی ترمینال از clear استفاده کنید. همچنین برای پیشنهاد کلمه دوبار از کلید TAB استفاده کنید.

```

Angelas-MacBook-Pro:intro-to-node angelayu$ node
> .exit
Angelas-MacBook-Pro:intro-to-node angelayu$ node
>
> (To exit, press ^C again or type .exit)
>
Angelas-MacBook-Pro:intro-to-node angelayu$ 

```

**Key words:** The Node REPL (Read Evaluation Print Loops).

## 6. How to Use the Native Node Modules

در این قسمت درباره نحوه فراخواندن و استفاده از ماثول های آماده Node را نشان می دهد.

The screenshot shows the Node.js v17.4.0 documentation website. On the left, there's a sidebar with links like 'About this documentation' and 'Usage and example'. The main content area has a large title 'Node.js v17.4.0 documenta' (partially cut off) and a 'Table of contents' section. Under 'File system', it lists several items: 'Promise example', 'Callback example', 'Synchronous example', 'Promises API', 'Class: FileHandle' (which further lists 'Event: 'close'' and 'filehandle.appendFile(data[, options])').

- ابتدا باید ماثول را به شکل `require("moduleName")` فراخوانیم یا ایجاد کنیم. در ادامه دستورات و استفاده دلخواه را از آن ماثول انجام می دهیم.(مانند کپی کردن فایل در شکل زیر)

```
index.js
1 //jshint esversion:6
2
3
4 const fs = require("fs");
5
6 fs.writeFileSync("file1.txt", "Hello World!");
7
8 fs.writeFileSync("file2.txt", fs.readFileSync("file1.txt"));
```

- سایر ماثول های آماده و نحوه استفاده آن در سایت node موجود می باشد.

**Key words:** the Native or internal Node Modules.

## 7. The NPM Package Manager and Installing External Node Modules

علاوه بر پیکچ های آماده داخلی node ، می توان از پیکچ هایی که توسط دیگران نوشته شده است نیز استفاده کنیم. برای این منظور از NPM استفاده می کنیم که انواع پکیج های دلخواه را می توان در آن پیدا کرد.



- در شکل زیر ماژول یا پیکچ خود را پیکربندی می کنیم که در package.json قرار می گیرد و همچنین dependency یا ماژول های دیگران را توسط NPM در محل مورد نظر نصب می کنیم.

```
MohammadReza@MohammadReza MINGW64 ~/Desktop/intro-to-node
$ npm init
```

```
MohammadReza@MohammadReza MINGW64 ~/Desktop/intro-to-node
$ npm install supervillains
```

```
{
  "name": "intro-to-node",
  "version": "1.0.0",
  "description": "is the first module.",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "mrshaker",
  "license": "ISC",
  "dependencies": {
    "supervillains": "^3.0.0"
  }
}
```

- در نهایت کد خود را نوشته و از پکیج دیگران در NPM استفاده می کنیم و پروژه را اجرا می کنیم.

The screenshot shows a terminal window with two command-line sessions. The first session runs \$ node index.js and outputs "Grim Reaper". The second session runs \$ node index.js again and outputs "Answer".

```

Project
  intro-to-node
    node_modules
      index.js
      package-lock.json
      package.json

index.js
1 const supervillains = require ("supervillains");
2
3 var nameSupervillains = supervillains.random();
4
5 console.log(nameSupervillains);
6

MohammadReza@MohammadReza MINGW64 ~/Desktop/intro-to-node
$ node index.js
Grim Reaper

MohammadReza@MohammadReza MINGW64 ~/Desktop/intro-to-node
$ node index.js
Answer

```

**Key word:** external node package, npm init, dependency in package.json.

## 8. Tip from Angela - Step Up to the Challenge

اگر فکر می کنید ضعیف هستید، یا چیزی نمی دانید، نترسید. فقط ادامه بدید و یاد بگیرید به مرور زمان همه چی برایتان راحت تر می شود. چالش ها را پذیرا باشید و آن ها را حل کنید. خودتان را دست کم نگیرید. هر چالشی راه حلی دارد.

## 19. Express.js with Node.js

### 1. What is Express

Express یکی از فریم ورک های محبوب و مطرح Node.js در سمت back-end می باشد که از آن برای ایجاد web application ها و سمت سرور استفاده می شود.



- در جاهای مختلفی مورد استفاده قرار می گیرد یکی از کاربردهای آن در سرور می باشد.  
در نتیجه با استفاده از Express می توان سرعت برنامه نویس در سمت سرور را افزایش داد.



**Key words:** express framework, express in web application.

## 2. Creating Our First Server with Express

در این قسمت نحوه افزودن express به پروژه، و استفاده از آن برای راه اندازی سرور فایل محلی (localhost:3000) توضیح داده می شود و اسکلت بندی اصلی یک سرور کوچک را نشان می دهد.

The screenshot shows a terminal window with several parts:

- A top bar with code: 

```
"dependencies": {  
    "express": "^4.16.3"  
}
```

 and a command: `$ npm install express`.
- A central area showing a file structure:
  - Project folder: my-express-server
  - node\_modules
  - package-lock.json
  - package.json
  - server.js
- A code editor view showing the contents of `server.js`:

```
//jshint esversion:6  
  
const express = require("express");  
  
const app = express();  
  
app.listen(3000, function(){  
    console.log("Server started on port 3000");  
});
```
- A bottom terminal window showing the command `$ node server.js` and the output `Server started on port 3000`.

**Key words:** a simple server with Express, initials Express.

## 3. Handling Requests and Responses the GET Request

در این قسمت نحوه گرفتن درخواست از مرورگر کاربر و پاسخ دادن به آن بررسی می شود.

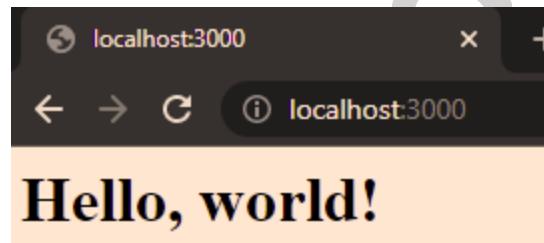
- `App.listen()` ، پورت مورد نظر را فعال و آماده تعامل می کند. در واقع سرور راه اندازی می شود، ولی این سرور پاسخی برای ارائه ندارد.
- در ادامه توسط `app.get()` ، در صورت `request` یا تعامل `response` یا پاسخ به مرورگر ارسال می شود..
- علامت "/" نشان دهنده `root rout` یا مسیر صفحه اصلی سرور می باشد.

The screenshot shows a code editor interface. On the left, there's a 'Project' sidebar with a tree view of a 'my-express-server' directory containing 'node\_modules', 'package-lock.json', 'package.json', and 'server.js'. The main area has two tabs: 'package.json' and 'server.js'. The 'server.js' tab displays the following code:

```
//jshint esversion:6
const express = require("express");
const app = express();

app.get("/", function(request, response){
  response.send("<h1>Hello, world!</h1>");
});

app.listen(3000, function(){
  console.log("Server started on port 3000");
});
```



**Key words:** rout or home page address, app.get, response to request on rout.

## 5. Understanding and Working with Routes

- توسط `app.get()` می توان مسیرها یا route های جدید برای سرور ایجاد کرد و برای هر کدام پاسخی تعریف کرد.

```

Project
  my-express-server
    node_modules
    package-lock.json
    package.json
    server.js

server.js
1 //jshint esversion:6
2
3 const express = require("express");
4
5 const app = express();
6
7 app.get("/", function(req, res){
8   res.send("<h1>Hello, world!</h1>");
9 });
10
11 app.get("/contact", function(req, res){
12   res.send("Contact me at: angela@gmail.com");
13 });
14
15 app.get("/about", function(req, res){
16   res.send("My name is Angela and I love beer and code.");
17 });
18
19 app.get("/hobbies", function(req, res){
20   res.send("<ul><li>Coffee</li><li>Code</li><li>Beer</li></ul>");
21 });
22
23 app.listen(3000, function(){
24   console.log("Server started on port 3000");
25 });
26

```

- برای راحتی کار و تعامل با سرور، nodemon را نصب کنید.

```

Angelas-MacBook-Pro:my-express-server angelayu$ nodemon server.js
[nodemon] 1.17.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: ***!
[nodemon] starting `node server.js`
Server started on port 3000

```

**Key words:** make and interaction with new routes, install nodemon.

## 6. What We'll Make A Calculator

در اینجا به معرفی چالش ایجاد یک ماشین حساب، توسط express توضیح داده می شود.

- این اولین web application دوره می باشد یعنی سرور در Back-end کار محاسباتی می کند و آن را تحويل کاربر می دهد. ولی در front-end فایل های css, html, javascript فقط در مرورگر کاربر اجرا می شود و یا اصطلاحاً استاتیک می باشد.

**Key words:** first web application.

## 8. Calculator Setup Challenge Solution

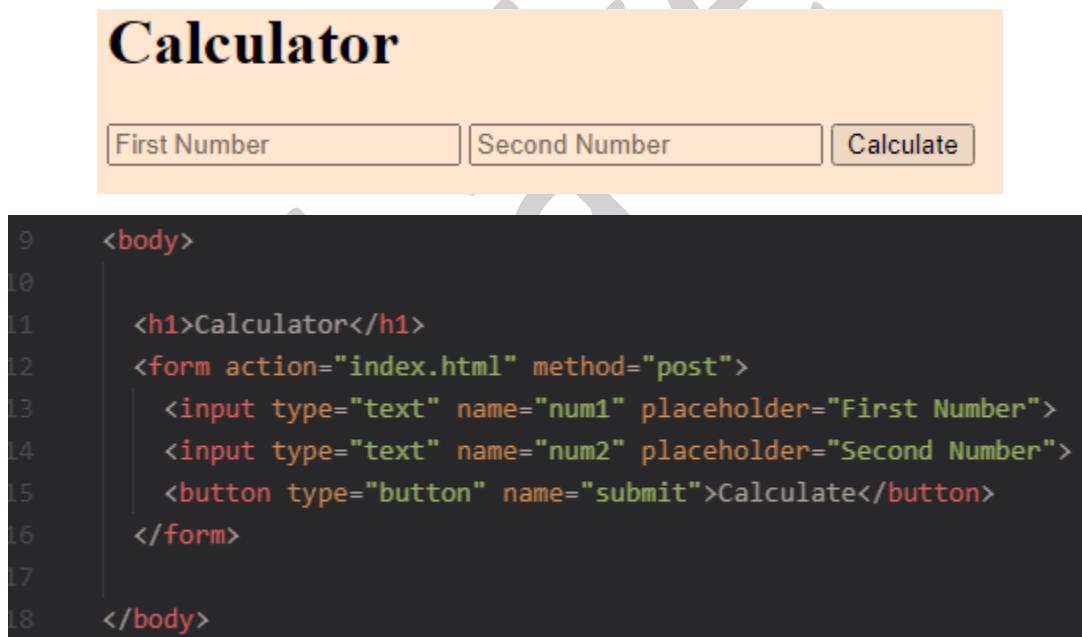
در این قسمت جمع بندی مطالب این فصل می باشد، که چگونه تنظیمات اولیه سرور مورد نظر را ایجاد کنیم تا برای پروژه ماشین حساب در ادامه استفاده کنیم.

- با دستور `atom` در ترمینال می توان، مسیر فعلی را در `atom` به اجرا در آورد. (علامت نقطه `.` یعنی مسیر فعلی، مثلًا `ls` محتوای مسیر فعلی را نشان می دهد).

**Key words:** recap of setup a simple server.

## 9. Responding to Requests with HTML Files

در این قسمت `html` ماشین حساب ایجاد می شود، و در ادامه توسط `res.sendFile()` فایل `html` برای کاربر ارسال می شود.



The screenshot shows a web browser displaying a simple calculator application. The page title is "Calculator". It features two input fields: "First Number" and "Second Number", and a "Calculate" button. Below the browser window is a code editor showing the HTML code for this application.

```
9 <body>
10
11     <h1>Calculator</h1>
12     <form action="index.html" method="post">
13         <input type="text" name="num1" placeholder="First Number">
14         <input type="text" name="num2" placeholder="Second Number">
15         <button type="button" name="submit">Calculate</button>
16     </form>
17
18 </body>
```

```

calculator.js                                         index.htm
1
2     const express = require("express");
3
4     const app = express();
5
6     app.get("/", function(req, res){
7         res.sendFile(__dirname + "/index.html");
8     });
9
10    app.listen(3000, function(){
11        console.log("server started on port 3000");
12    });

```

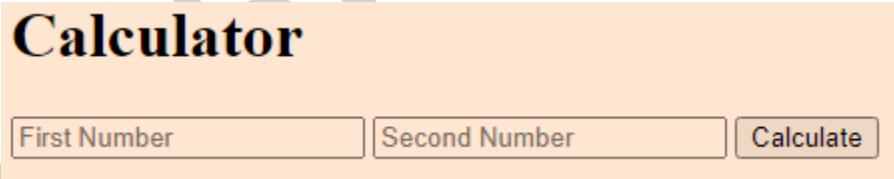
- توسط `__dirname` می توان به مکان فعلی فایل ها دسترسی داشت. بنابراین می تواند در سرورها و جاهای دیگر کار را ساده کند و نیازی به تکرار تعریف مسیر نباشد.

`console.log(__dirname);`

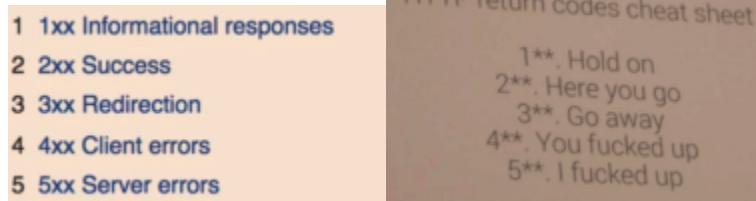
**Key words:** send html to web, `__dirname`, make calculator html.

## 10. Processing Post Requests with Body Parser

در این قسمت درباره نحوه گرفتن دیتا از سمت کاربر و استفاده از آن در سمت سرور صحبت می شود.



- دستور `app.get` برای بارگذاری صفحات بود، حال در ادامه برای گرفتن دیتا از سمت کلاینت باید از دستور `app.post` استفاده شود که توسط روش `html form` در `HTTP request` ایجاد شده است.
- در شکل زیر حالت های ایجاد شده، برای `HTTP request` را نشان می دهد. در صورت عدم استفاده از `app.post` با خطای `404` مواجه خواهیم شد.



- در نتیجه از طریق form روش و محل ارسال اطلاعات (action) را مشخص می کنیم.

```
<form action="/" method="post">

    <h1>Calculator</h1>
    <input type="text" name="n1" placeholder="First Number">
    <input type="text" name="n2" placeholder="Second Number">
    <button type="submit">calculate</button>

</form>
```

- در ادامه برای تجزیه و به دست آوردن اطلاعات، باید از express body-parser در استفاده کنیم.

Terminal command: `npm install body-parser`

Browser screenshot showing Form Data:

- num1: 2
- num2: 3
- submit:

- استفاده از نوع داده urlencoded برای داده های ارسالی از طریق html استفاده می شود.

```
const express = require("express");
const bodyParser = require("body-parser");

const app = express();
app.use(bodyParser.urlencoded({extended: true}));
```

#### extended

The `extended` option allows to choose between parsing the URL-encoded data with the `querystring` library (when `false`) or the `qs` library (when `true`). The "extended" syntax allows for rich objects and arrays to be encoded into the URL-encoded format, allowing for a JSON-like experience with URL-encoded. For more information, please see the `qs` library.

Defaults to `true`, but using the default has been deprecated. Please research into the difference between `qs` and `querystring` and choose the appropriate setting.

- در نهایت داده دریافتی را به عدد تبدیل کرده و استفاده می کنیم.

```
app.post("/", function(req, res){

  var num1 = Number(req.body.n1);
  var num2 = Number(req.body.n2);

  var result = num1 + num2;

  res.send("The result of the calculation is " + result);
});
```

- نکته: Get and Post requests ، درخواست های کاربر برای گرفتن صفحه (get) و ارسال کردن اطلاعات (post) نامیده می شوند.

**Key words:** html form element, how to use Post Request, HTTP requests status, how to setup and use body-parser.

## 12. Solution to the BMI Routing Challenge

در این قسمت در ادامه پروژه ، route جدیدی ایجاد، و از get و post استفاده می کند تا BMI را محاسبه کند.

localhost:3000/bmicalculator

# BMI Calculator

Weight  height

```
<body>
  <h1>BMI Calculator</h1>
  <form action="/bmicalculator" method="post">

    <input type="text" name="w" placeholder="Weight">
    <input type="text" name="h" placeholder="height">
    <button type="submit">Calculate BMI</button>
  </form>

</body>
```

```
app.get("/bmicalculator", function(req, res) {
  res.sendFile(__dirname + "/bmicalculator.html");
});

app.post("/bmicalculator", function(req, res) {
  var weight = parseFloat(req.body.w);
  var height = parseFloat(req.body.h);
  var bmi = weight / (height * height);

  res.send("Your BMI is " + bmi);
})
```

**Key words:** using get and post request in new route of server, completing challenge.

### 13. Tip from Angela - How to Solidify Your Knowledge

مطلوبی و مفاهیم را که یاد می گیرید، برای اینکه در حافظهتان مستحکم و ماندگار شود، سعی کنید آن را به صورت نکته بنویسید، یا آن را برای دیگران توضیح دهید. یاد دادن به دیگران و یاد گرفتن از دیگران، یک مؤلفه اساسی در فهم یک موضوع می باشد. وقتی کسی از شما درباره کدی می پرسد، شما سعی می کنید آن را مرور

کنید و توضیح دهید، یا اگر ندانید مجبور می شوید که سرج کنید، تا بفهمید. در هر صورت باعث افزایش فهم شما می شود.

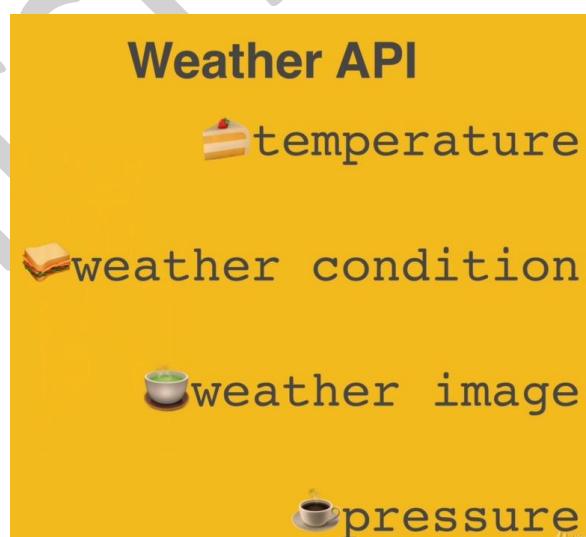
## 20. APIs - Application Programming Interfaces

### 1. Why Do We Need APIs

در این فصل درباره API صحبت می شود.

- API خدماتی می باشد، که توسط شرکت ها یا افراد دیگر (به صورت رایگان یا پولی)، در اختیار برنامه نویسان قرار می گیرد تا در پروژه خود مورد استفاده قرار دهند.

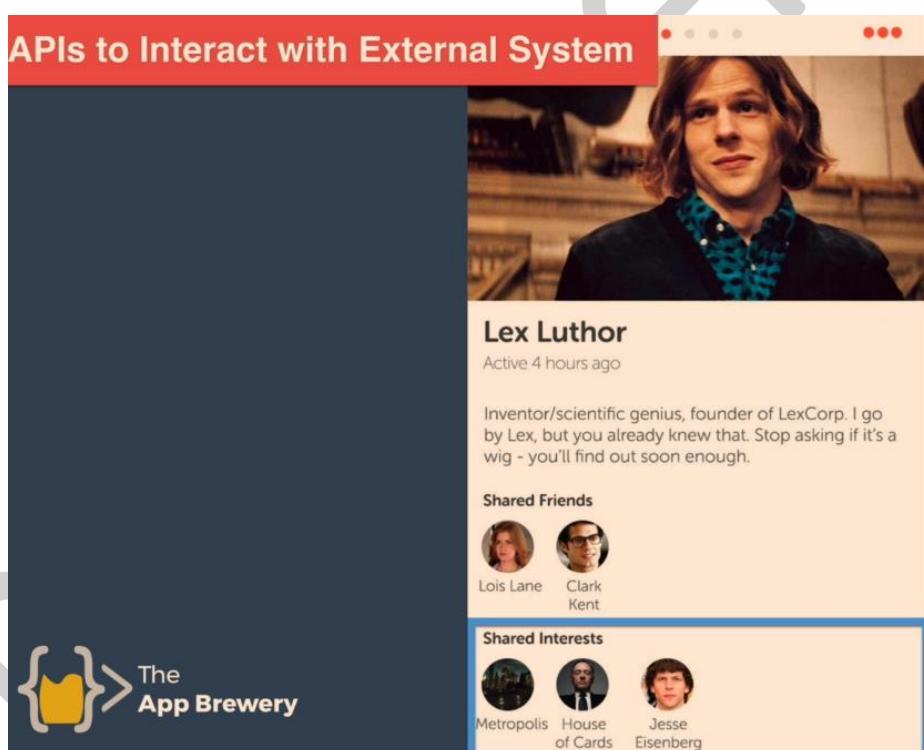
An Application Programming Interface (API) is a set of commands, functions, protocols, and objects that programmers can use to create software or interact with an external system.



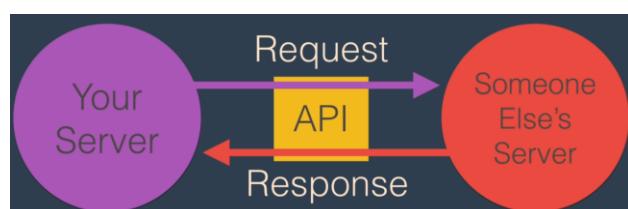
- jQuery یک API رایگان برای ایجاد نرم افزار می باشد که از تابع های آن در پروژه استفاده می شود.



- یکسری سایت ها مثل API دارای facebook می باشند، که دیتای آن برای استفاده در پروژه های دیگر به کار می رود.



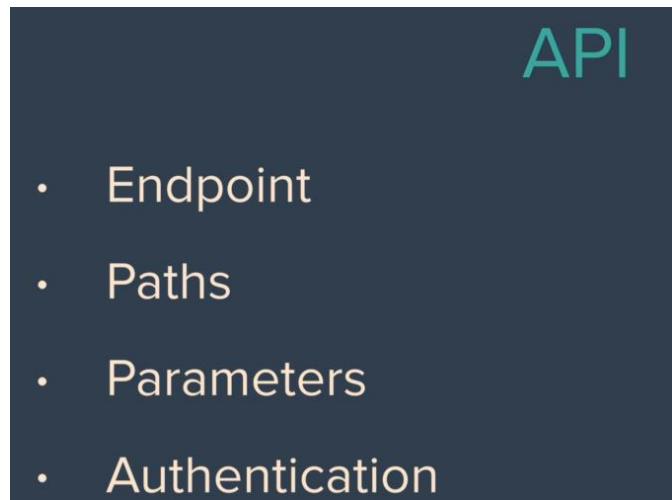
- در حالت کلی باید Document هر API را مطالعه و از آن در پروژه استفاده نمود.



**Key words:** API definitions and concepts.

## 2. API Endpoints, Paths and Parameters

در این قسمت درباره نحوه استفاده از API صحبت می شود.



- هر API دارای قسمت های مختلفی می باشد که جهت استفاده از آن باید آن اصول را رعایت نمود.
- مسیر اصلی یا rout هر API نامیده می شود. بعد از آن Path و سپس بعد از علامت ؟ ، Parameters قرار می گیرد. Path برای محدود کردن و مشخص کردن نوع دیتای درخواستی مورد نظر می باشد. در واقع نوعی فیلتر برای API تعیین می کنیم.

sv443.net/jokeapi/v2

## Menu

### JokeAPI Documentation

Try it out here:

Select category / categories:  Any  Custom:  Programming  Miscellaneous  Dark

Select flags to blacklist: (optional)  nsfw  religious  political  racist  sexist

Select response format:  default (json)  xml  yaml

Select at least one joke type:  single  twopart

Search for a joke that contains this search string: (optional)

Search for a joke in this ID range: (optional) From: 0 To: 162

**Endpoint**

URL: <https://sv443.net/jokeapi/v2/joke/Any>

Reset Form Send Request >

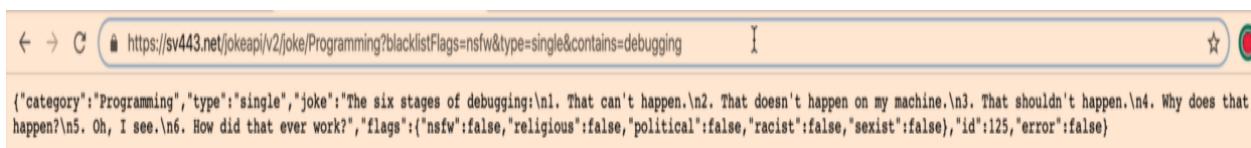
## Path

**Endpoint**

<https://sv443.net/jokeapi/v2/joke/Programming>

## Parameter

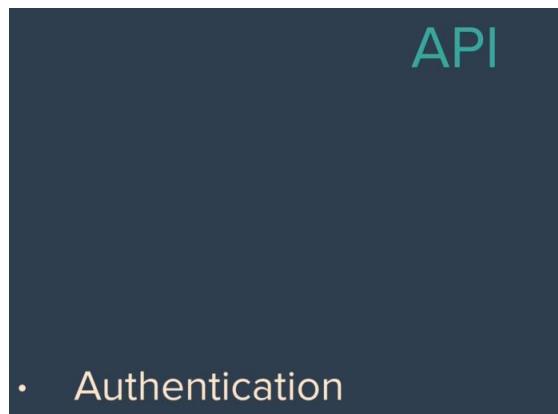
<https://sv443.net/jokeapi/v2/joke/Any?contains=debugging&type=single>



**Key words:** API Endpoints, Paths and Parameters.

### 3. API Authentication and Postman

در این قسمت به نحوه Authentication یا احراز هویت در API صحبت می شود.



- پس از ثبت نام در سایتی که خدماتی API می دهد، با توجه به app id یا کلید اختصاصی شما، احراز هویت توسط سایت انجام می شود و در نهایت می توان از خدمات API رایگان تا پولی را انتخاب کرد.

	Free	Startup	Developer	Professional	Enterprise
Price per month	Free Price is fixed, no other hidden costs (VAT is not included)	40 USD / month	180 USD / month	470 USD / month	2,000 USD / month
Subscribe	<a href="#">Get API key and Start</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>	<a href="#">Subscribe</a>
Calls per minute (no more than)	60	600	3,000	30,000	200,000
Current weather API	✓	✓	✓	✓	✓
4 days/hourly forecast API <small>NEW</small>	-	-	✓	✓	✓
5 days/3 hour forecast API	✓	✓	✓	✓	✓
16 days/daily forecast API	-	✓	✓	✓	✓

- API را در API به عنوان یکی از parameter ها وارد می کنیم.
- در API ترتیب قرارگیری پارامترها مهم نیست.

API call:

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={your api key}
```

api.openweathermap.org/data/2.5/weather?q=London,uk&appid=e72ca729af228beabd5d20e3b7749713

- از نرم افزار Postman جهت سهولت در ویرایش و مدیریت API استفاده می شود.

The screenshot shows the Postman application interface. At the top, there's a header bar with icons for New, Import, Runner, My Workspace, Invite, Refresh, and Upgrade. Below the header, a search bar contains the URL "GET api.openweathermap.org/data/2.5/weather?q=London&appid=e72ca729af228beabd5d20e3b7749713". To the right of the search bar are buttons for "Send" and "Save". The main area is titled "Untitled Request". Underneath, a "Params" tab is selected, showing a table for "Query Params". The table has columns for KEY, VALUE, and DESCRIPTION. It contains four rows with checked checkboxes for "q" (value: London), "appid" (value: e72ca729af228beabd5d20e3b7749713), and "units" (value: metric). There is also a row for "Key" with a "Value" column and a "Description" column. Other tabs like Authorization, Headers, Body, Pre-request Script, Tests, Settings, Cookies, and Code are visible but not selected.

**Key words:** API Authentication and Postman.

#### 4. What is JSON

در این قسمت درباره JSON صحبت می شود.



- JSON یک فرمت ذخیره دیتا می باشد، که الگوی ذخیره سازی آن مانند ایجاد Object در جاوااسکریپت می باشد که می تواند دارای چندین Object تو در تو باشد.
- در شکل زیر یک object در جاوااسکریپت را نشان می دهد، که در موقع ارسال توسط API برای فشرده سازی به فرمت JSON تبدیل می شود.

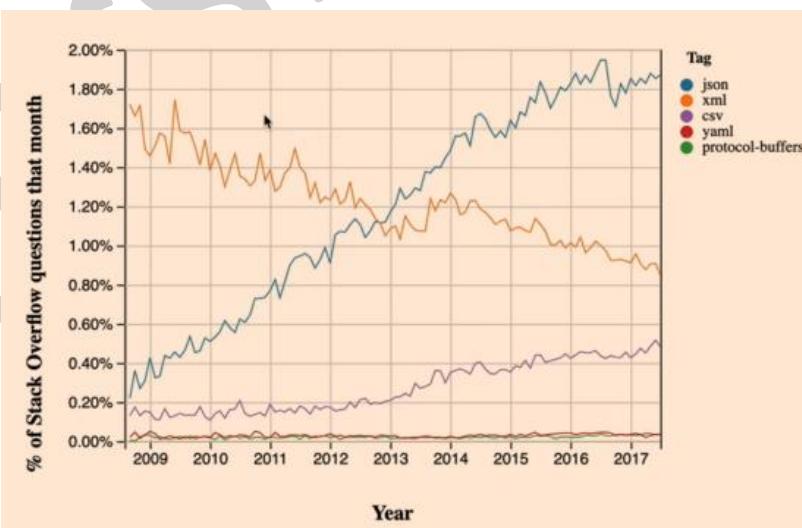


- اطلاعات JSON به صورت string می باشد، که می توان آن را به صورت object جاوااسکریپت ، تبدیل کرد.



- JSON محبوبیت و خوانایی بیشتری نسبت به سایر فرمت ها دارد.

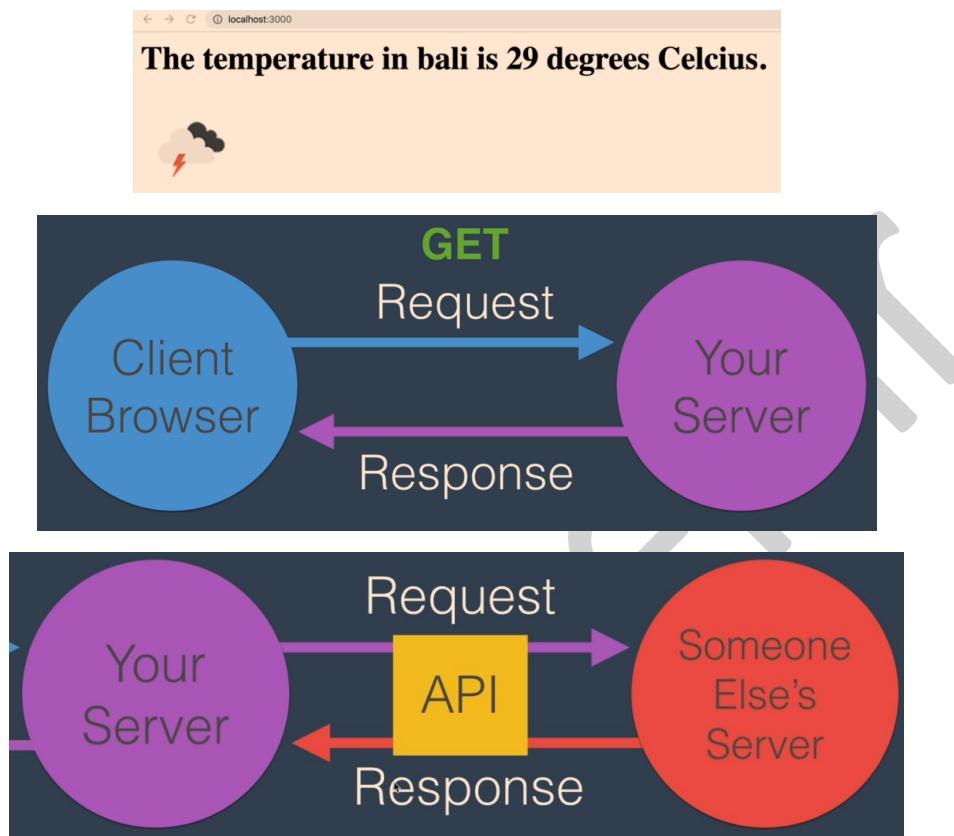
XML is much more difficult to parse than JSON.  
JSON is parsed into a ready-to-use JavaScript object.



**Key words:** JSON definition, JSON structure.

## 5. Making GET Requests with the Node HTTPS Module

در این قسمت نحوه ایجاد Get request برای بدهست آوردن دیتا از سرور دیگر یا API توضیح داده می شود.



برای Get request به منظور API می توان از کتابخانه های Node مثل `axios` و `request` استفاده کرد. ولی ما در اینجا از خود ماثول اصلی و بومی Node به اسم `https` استفاده می کنیم.

Node.js      `https.get(url[, options][, callback])`

```
app.get("/", function(req, res){  
  const url = "https://api.openweathermap.org/data/2.5/weather?q=tehran&appid=...";  
  
  https.get(url, function(response){  
    console.log(response);  
  })  
  
  res.send("Server is up and running.")  
})
```

**Key words:** Making GET Requests with the Node HTTPS Module, Weather Project, 5 ways to making http requests.

## 6. How to Parse JSON

در این قسمت نحوه دسترسی به دیتای API ارسالی JSON پرداخته می شود.

- از `response.statusCode` می توان وضعیت HTTP request را بررسی نمود که اگر 200 باشد یعنی API اطلاعات را به کاملی داده است.

## HTTP response status codes

Web technology for developers > HTTP > HTTP response status codes

### On this Page

Information responses  
Successful responses  
Redirection messages  
Client error responses  
Server error responses  
Browser compatibility  
See also

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

1. Informational responses (100–199),
2. Successful responses (200–299),
3. Redirects (300–399),
4. Client errors (400–499),
5. and Server errors (500–599).

- JSON در واقع فرمتی است که object های جاوا اسکریپت را به صورت string ذخیره می کند. شکل زیر (Buffer) نشان دهنده کدهای هگزا دسیمال ذخیره شده JSON در API ، response ، `JSON.parse(data)` در می باشد.

```
<Buffer 7b 22 63 6f 6f 72 64 22 3a 7b 22 6c 6f 6e 22 3a 2d 3
0 2e 31 33 2c 22 6c 61 74 22 3a 35 31 2e 35 31 7d 2c 22 77 6
5 61 74 68 65 72 22 3a 5b 7b 22 69 64 ... 410 more bytes>
```

- برای دسترسی به دیتای JSON (Buffer) از دستوری مانند `response.on("data")` استفاده می کنیم. در ادامه توسط دستور `JSON.parse(data)` آن را به object جاوا اسکریپت تبدیل می کنیم. ( دستور `JSON.stringify(data)` برعکس دستور می باشد و object را به تبدیل می کند).



**Key words:** Parse JSON, unwrap JSON, access to JSON data from API.

## 7. Using Express to Render a Website with Live API Data

در این قسمت در ادامه پروژه هواشناسی، پس دریافت اطلاعات مورد نظر، از آن استفاده کرده و آن را توسط express در سایت نمایش می دهیم.



```
response.on("data", function(data){  
  const weatherData = JSON.parse(data);  
  const temp = weatherData.main.temp;  
  const weatherDescription = weatherData.weather[0].description;  
  const icon = weatherData.weather[0].icon;  
  const imageURL = "http://openweathermap.org/img/wn/" + icon + "@2x.png";  
  res.write("<p>The weather is currently " + weatherDescription + ".</p>");  
  res.write("<h1>The temperature in London is " + temp + " degrees Celcius.</h1>");  
  res.write("")  
  res.send();  
  console.log(temp);  
});
```

**Key words:** show live API Data for weather, res.write().

## 8. Using Body Parser to Parse POST Requests to the Server

در این قسمت در ادامه پروژه هواشناسی، توسط Post request قابلیت انتخاب شهر توسط کاربر را ایجاد می‌کنیم.

A screenshot of a web browser window titled 'localhost:3000'. It shows a simple HTML form with a text input field labeled 'City Name:' and a 'Go' button. The background of the page is light orange.

```
<body>  
  <form action="/" method="post">  
    <label for="cityInput">City Name: </label>  
    <input id="cityInput" type="text" name="cityName">  
    <button type="submit">Go</button>  
  </form>  
</body>
```

```
app.post("/", function(req, res){  
  const query = req.body.cityName;  
  const apiKey = "e72ca729af228beabd5d20e3";  
  const unit = "metric";  
  const url = "https://api.openweathermap.org/data/2.5/weather?q=" + query + "&appid=" + apiKey + "&units=" + unit;  
  request(url, function(error, response, body){  
    if(error){  
      res.send("Error: " + error);  
    } else {  
      res.send(body);  
    }  
  });  
});
```

**Key words:** Using Body Parser to Parse POST Requests to the Server, select different cities for predict weather.

## 9. The Mailchimp API - What You'll Make

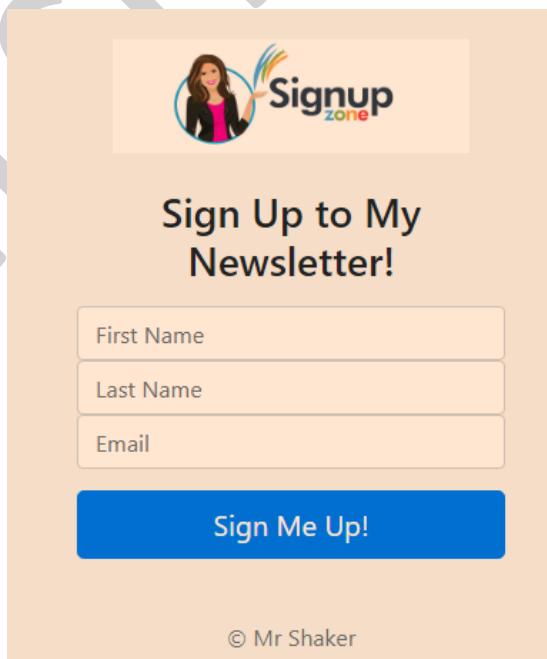
در این قسمت درباره پروژه ای که قرار است در ادامه ایجاد شود صحبت می شود و نحوه کار آن را توضیح می دهد.

- این پروژه، یه صفحه ثبت ایمیل برای دریافت اخبار و غیره می باشد که از API استفاده می کنیم.

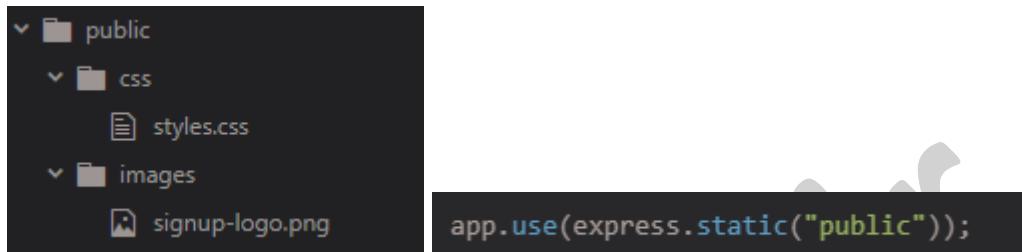
**Key words:** The Mailchimp API description.

## 10. Setting Up the Sign Up Page

در این قسمت صفحه ثبت نام پروژه مورد نظر توسط bootstrap ایجاد می شود، و در ادامه با back-end پیکربندی می شود.



- برای ارسال فایل های ثابت مانند CSS و تصاویر غیره باید پوشه به نام public ایجاد شود، و در پروژه توسط static آن مکان معرفی شود. در نهایت باید مسیر آنها را در href وارد نمود تا پس از ارسال فایل html ، توسط Get ، صفحه به همراه images و styles بارگذاری شود.



**Key words:** create and config Sign Up page, using public folder with "static" commad.

## 11. Posting Data to Mailchimp's Servers via their API

در این قسمت، در سایت mailchimp ثبت نام، و قوانین کار با آن بررسی می شود. در نهایت توانایی ثبت مشخصات در پایگاه داده mailchimp وجود خواهد داشت.

Email Address	First Name	Last Name	Address	Phone Number	Birthday	Tags	Email Marketing
hfgh@gfg.com	fyui	asddf					Subscribed
niun@gggs.com	alex	ninu					Subscribed
mohammadshaker36@gmail....	mamal	shaker	mrshui khadof squa				Subscribed

- از دستورات https.request و روش های جدید استفاده می شود.
- برای فهم برنامه و کارکرد آن حتماً کد پروژه را مطالعه کنید.

```
// making member with mailchimp rules for JSON
const data = {
  members: [
    {
      email_address: email,
      status: "subscribed",
      merge_fields: {
        FNAME: firstName,
        LNAME: lastName
      }
    }
  ]
};
```

```
// necessary settings for https.request
const url = "https://us14.api.mailchimp.com/3.0/lists/b8dce31d04";
const option = {
  method: "POST",
  auth: "shki:1d817b267dc95d9bb2438a1125b9f6cd-us14" // api authin
}

// post request with https.request for mailchimp
const request = https.request(url, option, function(response) {

  response.on("data", function(data) {
    console.log(JSON.parse(data));
  });

});

request.write(jsonData); // writing new member data in mailchimp
request.end(); // end of writing sign.
```

**Key words:** Posting Data to Mailchimp's Servers via their API, new rules, https.request, making data for api and set in Mailchip.

## 12. Adding Success and Failure Pages

در این قسمت صفحه موفقیت آمیز بودن یا نبود ثبت نام، به پروژه افزوده می شود.

# Uh Oh!

There is a problem, please try again or contact with developer!

Try Again!

# Awesome!

You've been successfully signed up to the newsletter, look forwards to lots of awesome content!

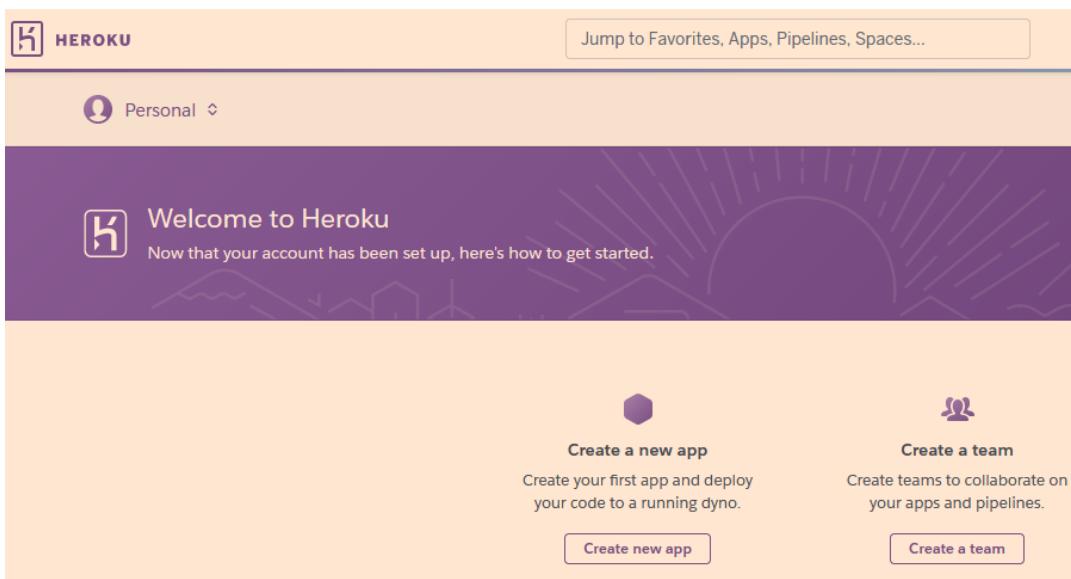
```
if(response.statusCode === 200){  
  res.sendFile(__dirname + "/success.html");  
} else {  
  res.sendFile(__dirname + "/failure.html");  
}  
  
app.post("/failure", function(req, res){  
  res.redirect("/");  
});
```

**Key words:** Adding Success and Failure Pages, response.statusCode, redirect to other pages.

## 13. Deploying Your Server with Heroku

در این قسمت اپلیکیشن نوشته شده را در سرورهای Heroku راه اندازی می کنیم.

- دقت کنید که صفحات ما static نیستند، و دارای تعامل با سرور هستند، بنابراین باید از سایت های خدماتی مانند Heroku استفاده کرد.



- برای راه اندازی پروژه در سرورهای Heroku ، از اسناد آن در سایت مورد نظر استفاده کنید.

## Getting Started on Heroku with Node.js

[Introduction](#)

[Set up](#)

[Prepare the app](#)

[Deploy the app](#) View logs

[Define a Procfile](#)

[Scale the app](#)

[Declare app dependencies](#)

[Run the app locally](#)

## Deploy the app

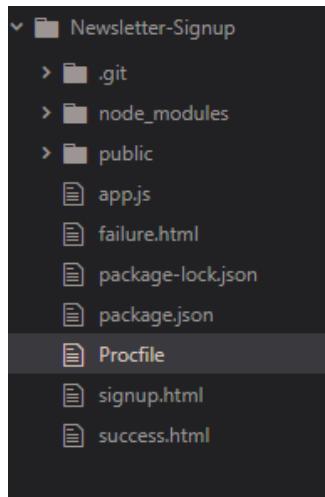
In this step you will deploy the app to Heroku.

Create an app on Heroku, which prepares a git repository for you.

```
$ heroku create
Creating sharp-rain-871... done
http://sharp-rain-871.herokuapp.com
Git remote heroku added
```

When you create an app, a git remote is automatically added to your app's git repository.

Heroku generates a random name (you can use the --name parameter to specify your own name).



```
77 app.listen(process.env.PORT || 3000, function() { // config for Localhost and Heroku
78   | console.log("server is running");
79 });
80
```

```
Angelas-MacBook-Pro:Newsletter-Signup angelayu$ git init
Reinitialized existing Git repository in /Users/angelayu/Desktop/Newsletter-Signup/.git/
Angelas-MacBook-Pro:Newsletter-Signup angelayu$ git add .
Angelas-MacBook-Pro:Newsletter-Signup angelayu$ git commit -m "First Commit"
```

**Key words:** Deploying Your Server with Heroku, launch your app in server.

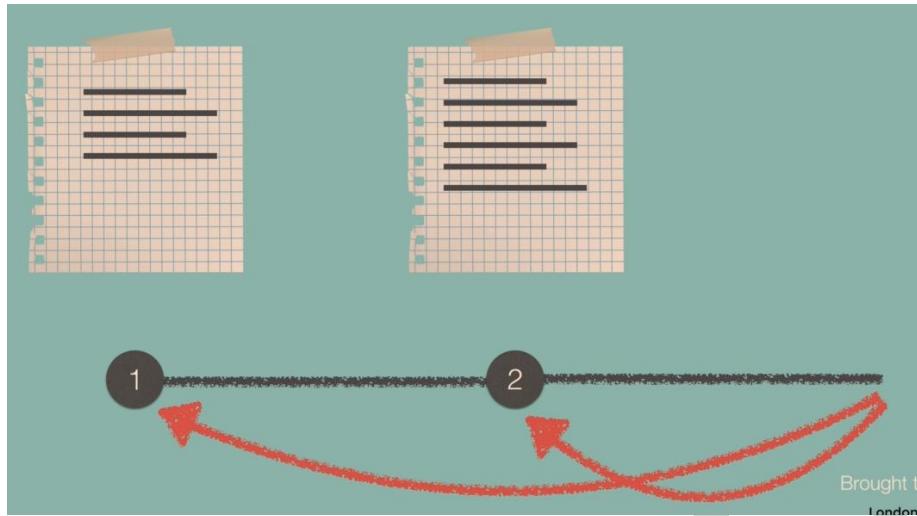
#### 14. Tip from Angela - Location, Location, Location!

همیشه بعد از دیدن هر قسمت، از آن برای خود نکته برداری کنید، و حتماً آن مفاهیم را کد بزنید و تمرین کنید، تا در حافظه شما بنشینند. در بعضی مواقع شما حس می کنید که خیلی می دانید و آن کار آسان است، و در بعضی مواقع حس می کنید که چیزی نمی فهمید و کار سخت است. در هر دو حالت باید به تمرین و پیشروی ادامه دهید تا به درجه تعالی و استادی برسید.

## 21. Git, Github and Version Control

### 1. Introduction to Version Control and Git

در این قسمت سرفصل های این فصل توضیح داده می شود و کاربری git را نشان می دهد.



**Key words:** Introduction to Version Control and Git, what is Git.

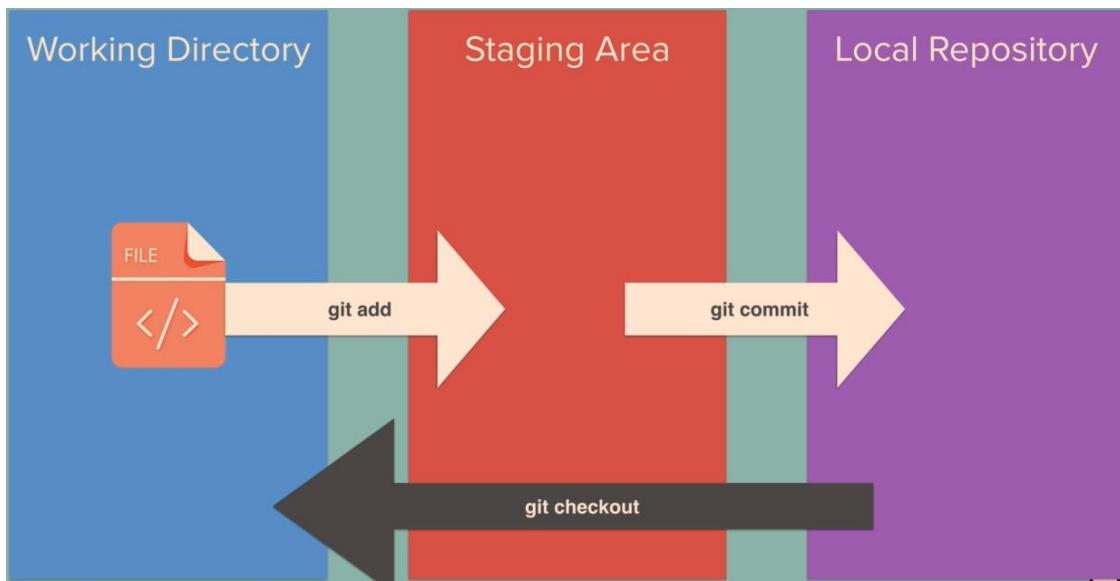
## 2. Version Control Using Git and the Command Line

در این قسمت نحوه ایجاد و استفاده از Git در local command line توضیح داده می شود.

- توسط git init مخزن یا repository در پوشه پروژه به صورت hidden file ایجاد می شود.
- در ادامه توسط git add می توان فایل یا فایل های مورد نظر را به staging area انتقال داد تا به عنوان فایل یا فایل های مورد ردگیری در git شناخته شود.
- در ادامه توسط git commit –m “your message version” ، فایل ها را در مخزن قرار می دهیم. (در پیام ها بهتر است از فعل زمان حال استفاده کنیم تا فعل زمان گذشته)
- هر پیام دارای Hash می باشد که برچسب commit انجام شده است. علامت HEAD نشان دهنده آخرین نسخه commit در پروژه می باشد.

```
commit c722ec4459a95e1036ab237ddf8221ed0e60c614 (HEAD -> master)
```

- هر لحظه توسط git status می توان وضعیت را پیگیری کرد و توسط git log می توان commit را مشاهده نمود.

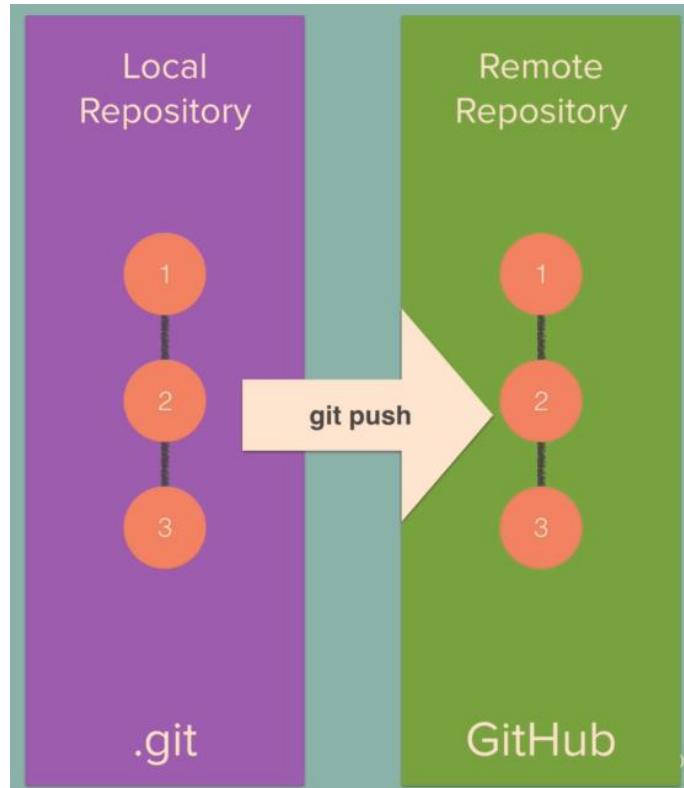


- توسط `git diff` می توان تفاوت فایل ویرایش شده با آخرین Commit را مقایسه نمود و در صورت نیاز توسط `git checkout` فایل را به آخرین Commit بازگرداند.

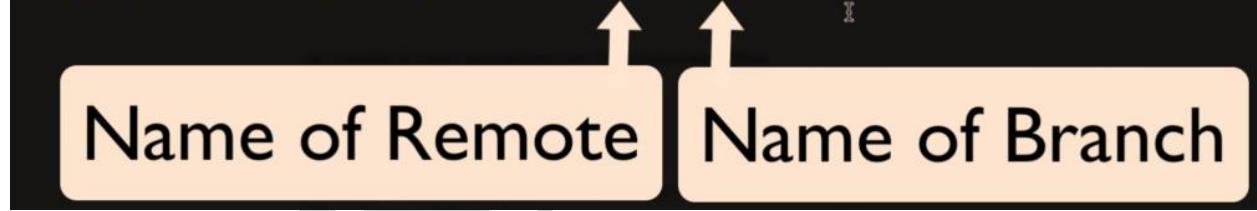
**Key words:** Version Control Using Git and the Command Line, git commands.

### 3. GitHub and Remote Repositories

- در این قسمت نحوه ایجاد repository در github و ارسال فایل را بررسی می کنیم.
- github remote repository برای git local Repository -  
شناخته می شود.
- شاخه master یا main، مسیر اصلی پروژه می باشد.



```
Angelas-MacBook-Pro:Story angelayu$ git remote add origin https://github.com/angelabauer/Story.git
Angelas-MacBook-Pro:Story angelayu$ git push -u origin master
```

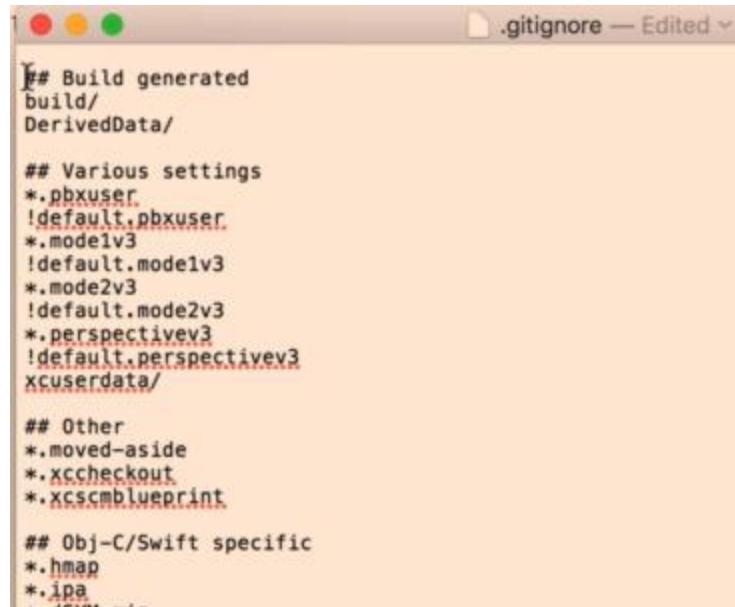


**Key words:** GitHub and Remote Repositories, difference between git and github.

## 5. Gitignore

- توسط ایجاد فایلی به نام `.gitignore` می توان اسمی فایل هایی را که نمی خواهیم توسط `git` مورد ردگیری قرار گیرد را وارد می کنیم. مثل فایل های دارای `api` یا فایل های غیر ضروری.

```
Angelas-MacBook-Pro:Project angelayu$ ls -a
.           .DS_Store      file1.txt      file3.txt
..          .gitignore     file2.txt      secrets.txt
```



```
# Build generated
build/
DerivedData/

## Various settings
*.pbxuser
!default.pbxuser
*.mode1v3
!default.mode1v3
*.mode2v3
!default.mode2v3
*.perspectivev3
!default.perspectivev3
Xcuserdata/

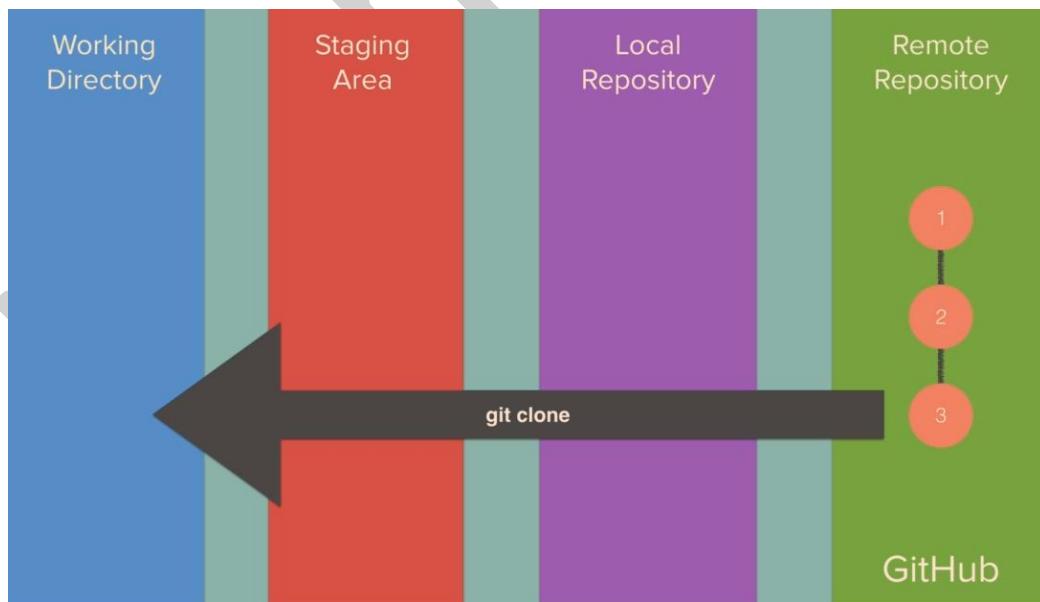
## Other
*.moved-aside
*.xccheckout
*.xcscmblueprint

## Obj-C/Swift specific
*.hmap
*.ipa
```

**Key words:** Gitignore, making except between files for push to remote, making more security.

## 6. Cloning

- توسط git clone می توان هر پروژه ای در github (پروژه خود و یا پروژه دیگران) را به local system منتقال داد و از آن استفاده کرد.



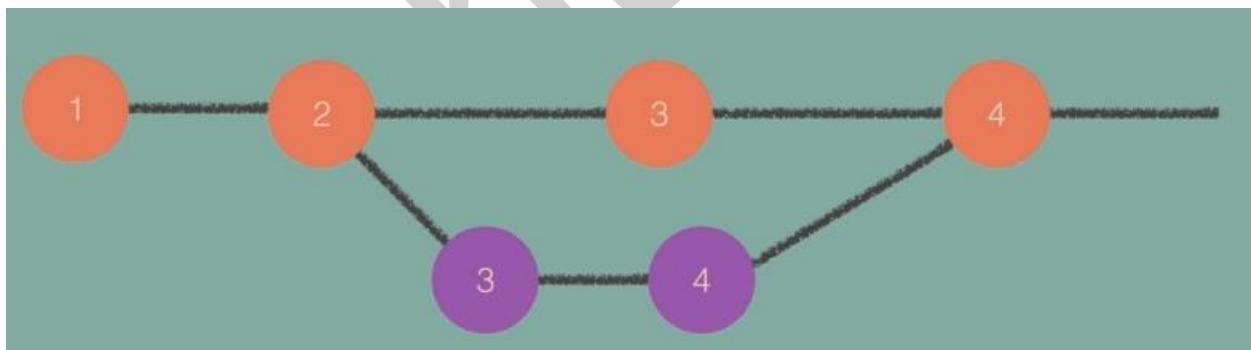


```
Angelas-MacBook-Pro:Desktop angelayu$ git clone https://github.com/austinzheng/swift-2048.git
```

**Key words:** Clone or download from Github repository to local system.

## 7. Branching and Merging

Branches نشان دهنده مسیر انجام پروژه می باشد. مثلاً پروژه اصلی در branch Main قرار می گیرد، حال اگر بخواهیم یک ویژگی جدیدی برای پروژه ایجاد کنیم، یک branch با نام جدید ایجاد می کنیم، و پس از ایجاد ویژگی های جدید پروژه، و تست کردن آن می توان آن را با مسیر Merge ، master را با مسیر Merge ، master کرد.



- از git branch برای ایجاد branch جدید یا مشاهده همه branch استفاده می شود.  
- از git checkout برای سوییچ بین branch ها استفاده می کنیم.

```
Angelas-MacBook-Pro:Story angelayu$ git branch alien-plot
Angelas-MacBook-Pro:Story angelayu$ git branch
  alien-plot
* master
Angelas-MacBook-Pro:Story angelayu$ git checkout alien-plot
Switched to branch 'alien-plot'
```

HEAD نشان دهنده وضعیت فعلی یا آخرین commit پروژه می باشد. نشان دهنده Master Github branch در local system branch می باشد.

```
(HEAD -> master, origin/master)

commit 172a45c03db59c86c077932fa24c1f95cd200389 (HEAD -> alien-plot)
Author: Angela Yu <angelayu@Angelas-MacBook-Pro.local>
Date:   Wed Sep 13 14:57:34 2017 +0100

    modify chapter 1 and 2 to have alien theme

commit e623df44bd52083efdc66153b33f246a3f608d28 (origin/master, master)
Author: Angela Yu <angelayu@Angelas-MacBook-Pro.local>
Date:   Tue Sep 12 15:14:25 2017 +0100

    complete chapter 2 and 3
```

- Git merge می توان branch فعلی را با mord نظر ادغام کرد.

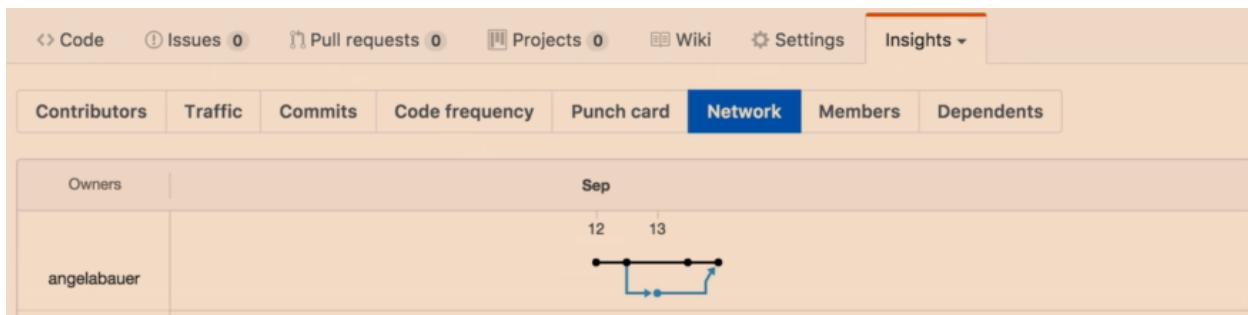
```
Angelas-MacBook-Pro:Story angelayu$ git merge alien-plot
```

- از طریق Github می توان تغییرات را به صورت گرافیکی دنبال کرد.

The screenshot shows a GitHub repository page for 'anglabauer / Story'. The repository has 0 stars and 0 forks. The 'Code' tab is selected. A dropdown menu shows 'Branch: master'. Below it, there are two sections of commits:

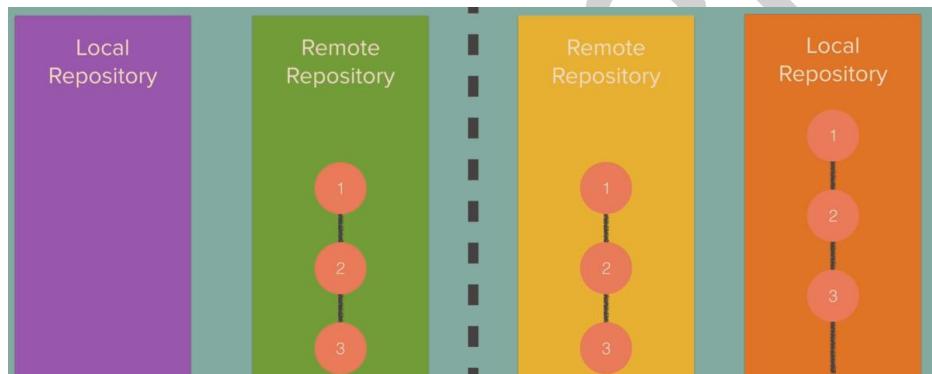
- Commits on Sep 13, 2017:**
  - Merge branch 'alien-plot' by Angela Yu committed 2 minutes ago (commit c743e7a)
  - add chapter 4 by Angela Yu committed 6 minutes ago (commit e887449)
  - modify chapter 1 and 2 to have alien theme by Angela Yu committed 8 minutes ago (commit 172a45c)
- Commits on Sep 12, 2017:**
  - complete chapter 2 and 3 by Angela Yu committed a day ago (commit e623df4)
  - Complete Chapter 1 by Angela Yu committed a day ago (commit c722ec4)

- به هر تعداد دلخواهی می توان branch ایجاد کرد و در پروژه های بزرگ چندین وجود دارد.

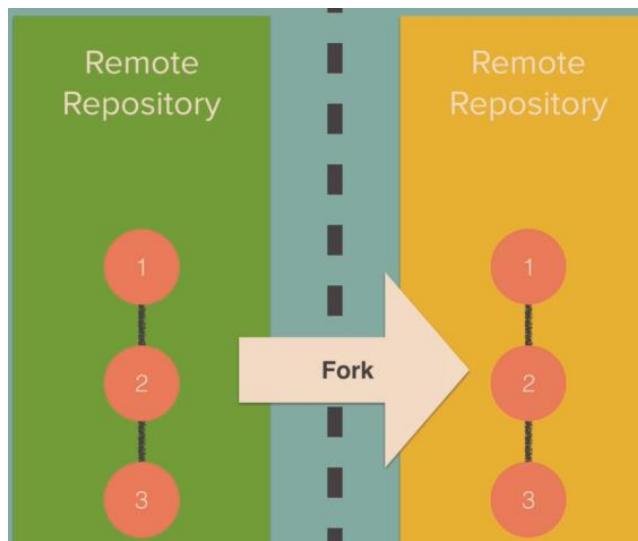


**Key words:** Branching and Merging.

## 9. Forking and Pull Requests

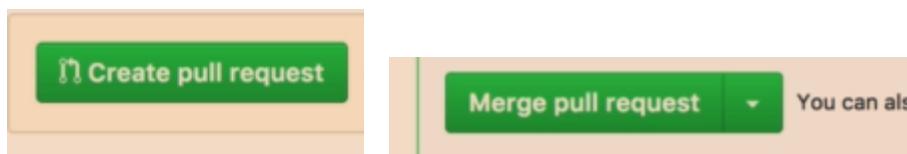
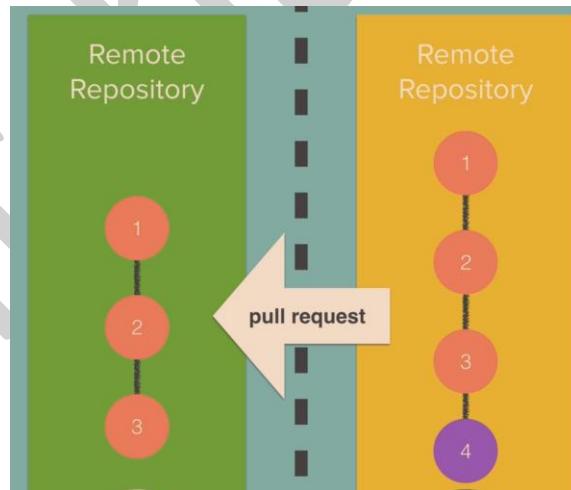
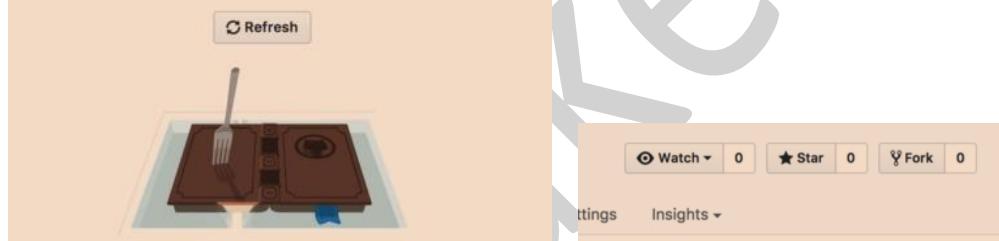


- در صورتی که بخواهیم در پروژه های دیگران در Github سهیم باشیم، می توان با استفاده از Fork یک کپی از آن پروژه در repository خود ایجاد کرد، سپس بعد از انجام تغییرات دلخواه، پروژه را برای مالک اصلی پروژه pull می کنیم. در ادامه در صورتی که مالک پروژه راضی باشد می توان درخواست pull شما را قبول، و آن را با پروژه خود merge کند.



### Forking angelabauer/Story

It should only take a few seconds.

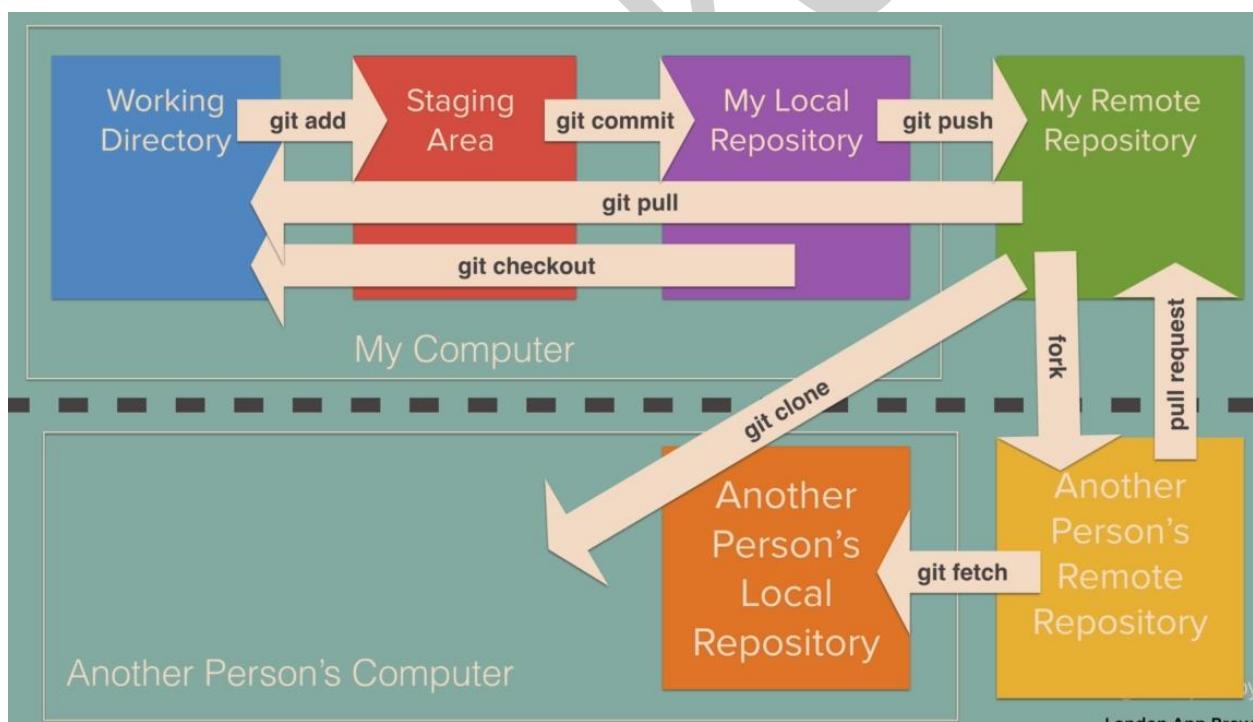


- در شکل زیر نشان دهنده مشارکت، یک developer branch دیگر در پروژه می باشد.



git clone means you are making a copy of the repository in your system. git fork means you are copying the repository to your Github account. **git pull means you are fetching the last modified repository.**

- در شکل زیر معماری کلی دستورات Git را نشان می دهد.



**Key words:** Forking and Pull Requests, contribution in other people project on Github.

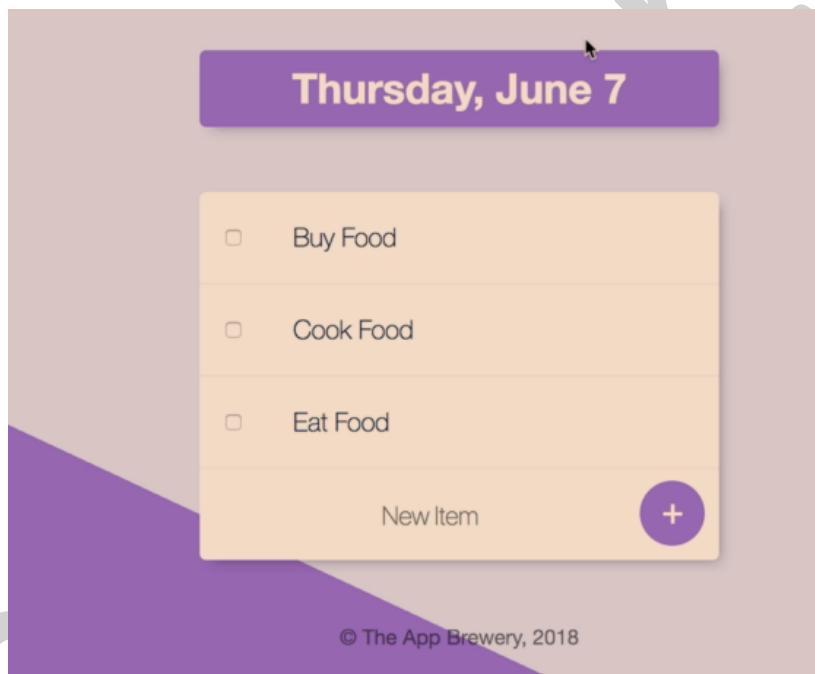
## 10. Tip from Angela - Spaced Repetition

مطلوبی را که میخوانید و می بینید به مرور زمان فراموش می کنید. برای اینکه فصل های یاد گرفته شده را به حافظه بلند مدت بسپارید، سعی کنید برنامه هفتگی، ماهانه، و یا سالانه برای مرور داشته باشید تا واقعاً آن را یاد بگیرید و در آن استاد شوید. (میتوانید نکاتی را که نوشتید سریع مرور کنید، و یا فیلم ها را با سرعت بیشتر مرور کنید).

## 22. EJS

### 1. What We'll Make A ToDoList

در این قسمت به معرفی پروژه این فصل که قرار است انجام شود صحبت می شود که یک ToDoList می باشد.



**Key words:** introduce the project, partials and templates.

### 3. Templates Why Do We Need Templates

در این قسمت مقدمات اولیه پروژه را آماده می کند و علت نیاز به template html توضیح می دهد.

- یک سری html هایی که استفاده می کنیم برای سایت، با محتوای تکراری می باشد، در نتیجه باید یک قالب یا template html داشته باشیم، و با تغییر دلخواه آن ، سرعت کار را افزایش دهیم.

```

app.get("/", function(req, res){

  var today = new Date();
  var currentDay = today.getDay();

  if (currentDay === 6 || currentDay === 0) {
    res.write("<h1>Yay it's the weekend!</h1>");
  } else {
    res.sendFile(__dirname + "/index.html");
  }
});

```

**Key words:** Templates Why Do We Need Templates, today.getDay.

#### 4. Creating Your First EJS Templates

- یکی از مازول های npm برای تغییر html template در استفاده از EJS یا Embedded Javascript templating می باشد که توسط آن می توان المنش های Html را توسط Javascript تغییر داد در نتیجه از ایجاد صفحات زیاد و تکراری برای Html جلوگیری می شود.

### What is EJS?

What is the "E" for? "Embedded?" Could be. How about "Effective," "Elegant," or just "Easy"? EJS is a simple templating language that lets you generate HTML markup with plain JavaScript. No religiousness about how to organize things. No reinvention of iteration and control-flow. It's just plain JavaScript.

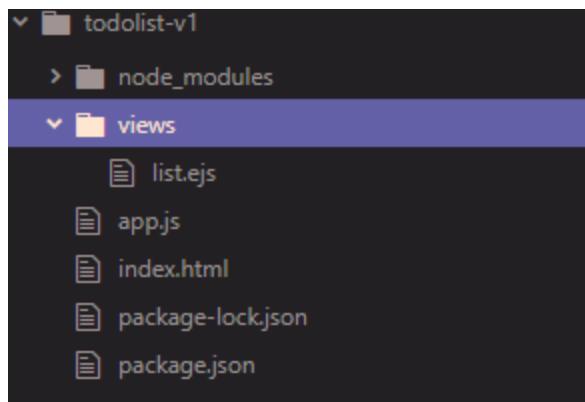
- در واقع توسط EJS می توان متغیرهای موجود در javascript را به صفحه html به اسم .ejs پاس داد و آن را مقدار داد و از ایجاد صفحات html تکراری جلوگیری کرد.

- برای استفاده از EJS باید پوشه views و فایل های .ejs. ایجاد گردد و در نهایت با استفاده از syntax مخصوص خودش، در html استفاده نمود.

<%= EJS %>

\$ npm install ejs

Embedded JavaScript templating.



```
if(currentDay === 6 || currentDay === 0){  
  day = "weekend";  
} else {  
  day = "weekday";  
}  
  
res.render("list", {kindOfDay: day});
```

```
list.ejs  
1  <!DOCTYPE html>  
2  <html lang="en" dir="ltr">  
3    <head>  
4      <meta charset="utf-8">  
5      <title>To Do List</title>  
6    </head>  
7    <body>  
8      <h1>It's a <%= kindOfDay %>! </h1>  
9    </body>  
10   </html>
```

حتماً document برای راه اندازی EJS و استفاده از آن را بخوانید.

**Key words:** EJS or Embedded Javascript Templating, Creating Your First EJS Templates, make views and file.ejs, use res.render, passing javascript var to html structure with EJS.

## 5. Running Code Inside the EJS Template

- توسط tag های ejs می توان در html ، کدهای javascript را به صورت محدود استفاده نمود.
- نکته: حواستان باشد، که html مخصوص محتوا متنی می باشد، بنابراین javascript استفاده شده در آن باید فقط به منظور تغییر context باشد، و logic های اصلی برنامه باید سمت سرور انجام شود.

### Tags

- <% 'Scriptlet' tag, for control-flow, no output
- <%\_ 'Whitespace Slurping' Scriptlet tag, strips all whitespace before it
- <%= Outputs the value into the template (HTML escaped)
- <%- Outputs the unescaped value into the template
- <%# Comment tag, no execution, no output
- <%% Outputs a literal '<%'
- %> Plain ending tag
- -%> Trim-mode ('newline slurp') tag, trims following newline
- \_%> 'Whitespace Slurping' ending tag, removes all whitespace after it

- مثلاً در کد زیر از عبارات شرطی در template استفاده شده است، و در هر خط که مربوط به است از tag javascript مربوطه استفاده شده است.

```
<body>
  |
  <% if (kindOfDay === "Thursday") { %>
    <h1 style="color: purple"><%= kindOfDay %> ToDo List</h1>
  <% } else { %>
    <h1 style="color: blue"><%= kindOfDay %> ToDo List</h1>
  <% } %>
```

**Key words:** Running Code Inside the EJS Template, EJS Tags uses, using javascript in html.

## 6. Passing Data from Your Webpage to Your Server

- در ادامه پروژه todo list ، در این قسمت نحوه افزودن لیست در صفحه اصلی را نشان می دهد.
- متغیرهای res.render در قسمت ejs ، باید در ابتدا تعیین شوند، سپس در ادامه توسط هر بار post request ، آرایه های items اصلاح می شود.



```
9   <body>
10  <h1> <%= kindOfDay %> </h1>
11  <ul>
12  <% for(var i=0; i<newListItems.length; i++){ %>
13  <li> <%= newListItems[i] %> </li>
14  <% } %>
15  </ul>
16
17  <form action="/" method="post">
18    <input type="text" name="newItem" placeholder="To Do">
19    <button type="submit">Add</button>
20  </form>
21
22  </body>
```

```
var items = ["Buy Food", "Cook Food", "Eat Food"];
```

```

app.get("/", function(req, res) {

    var today = new Date();
    var options = {
        day: "numeric",
        month: "long",
        weekday: "long"
    };

    var day = today.toLocaleDateString("en-US", options);

    res.render("list", { // we can use just one time render.
        kindOfDay: day,
        newListItems: items
    });

});

```

```

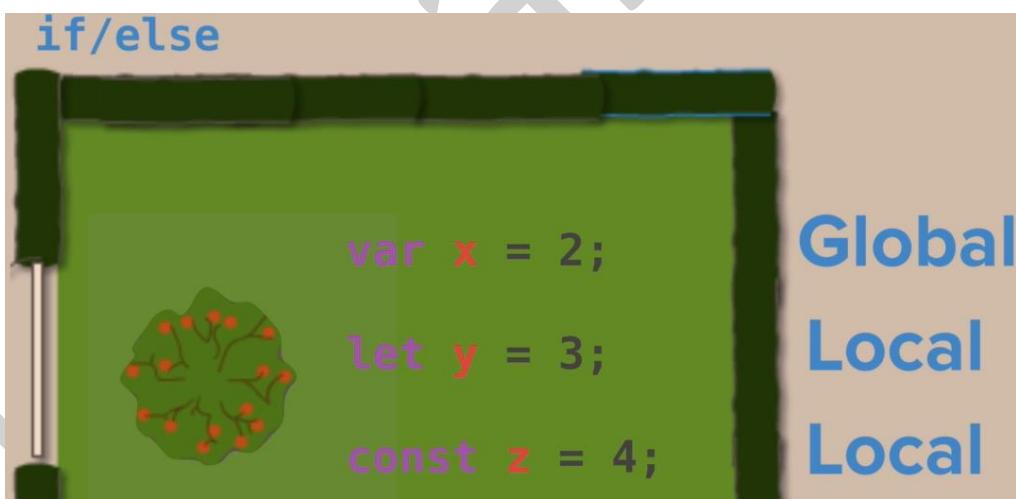
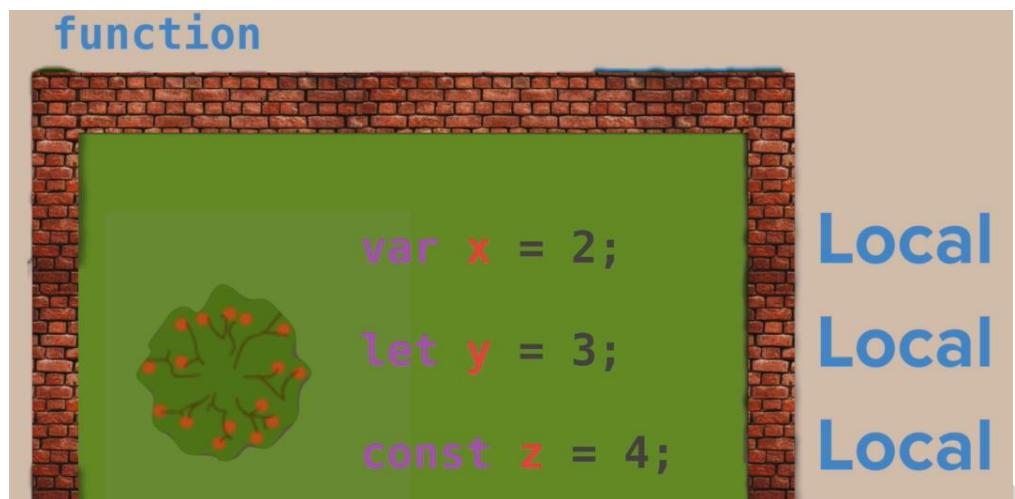
app.post("/", function(req, res) {
    var item = req.body newItem;
    //console.log(item);
    items.push(item);
    res.redirect("/"); // to refresh new html with new List Item
})

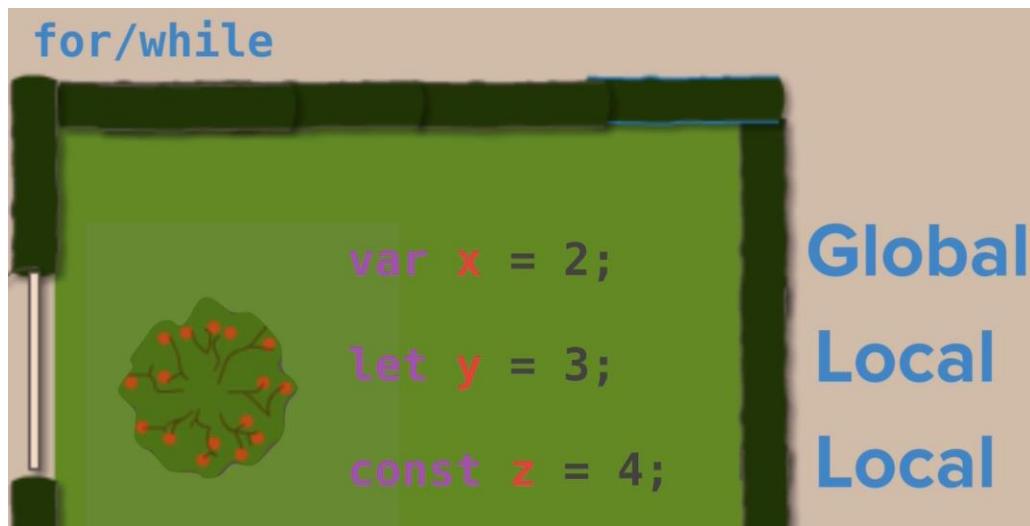
```

**Key words:** Passing Data from Your Webpage to Your Server and change html list, new concept, make array of list items with EJS template, use just one res.render.

## 7. The Concept of Scope in the Context of Javascript

- Scope یا رنج متغیرهای ایجاد شده به ۲ صورت local or global variable می باشد.
- در javascript var، نوع const و let و نوع globally-scoped در صورت if/else و (for/while) می باشند.
- بنابراین به مکان و نوع متغیری که در برنامه ایجاد می کنید توجه کنید و با توجه به اینکه خصوصیت های زیادی دارد، به جای آن در برنامه تان از let استفاده کنید. برای مقادیر ثابت نیز از استفاده کنید.
- انتخاب نوع درست و مناسب متغیر، باعث تمیزتر شدن کد و افزایش امنیت برنامه را به دنبال دارد.





## var

The `var` statement declares a function-scoped or globally-scoped variable, optionally initializing it to a value.

## let

The `let` statement declares a block-scoped local variable, optionally initializing it to a value.

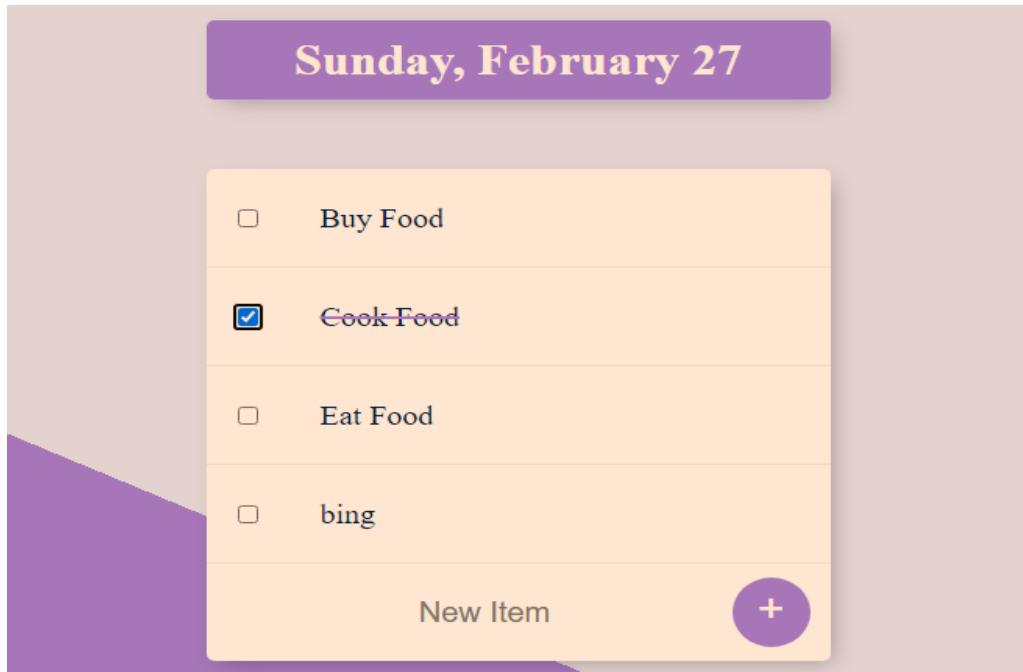
## const

Constants are block-scoped, much like variables declared using the `let` keyword. The value of a constant can't be changed through reassignment (i.e. by using the [assignment operator](#)), and it can't be redeclared (i.e. through a [variable declaration](#)). However, if a constant is an [object](#) or [array](#) its properties or items can be updated or removed.

**Key word:** The Concept of Scope in the Context of Javascript, new concept, var and let and const differences, local or global variable.

## 8. Adding Pre-Made CSS Stylesheets to Your Website

- در این قسمت CSS از قبل ساخته شده به قالب پروژه اعمال می شود.



- در این قسمت از css selector های جدید استفاده شد. Css selector ها خیلی زیاد می باشند، که باید در صورت نیاز به آن مراجعه کنید.

## CSS Selectors

In CSS, selectors are patterns used to select the element(s) you want to style.

Use our [CSS Selector Tester](#) to demonstrate the different selectors.

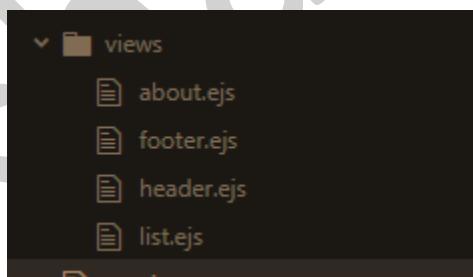
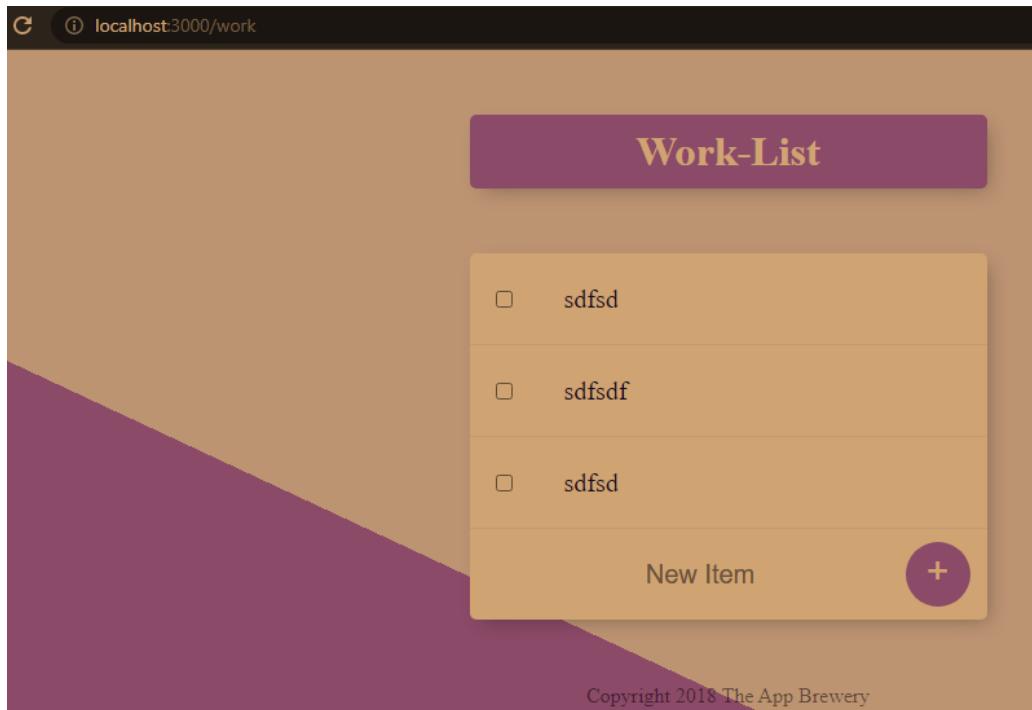
Selector	Example	Example description
<u>.class</u>	.intro	Selects all elements with class="intro"
<u>#id</u>	#firstname	Selects the element with id="firstname"
*	*	Selects all elements
<u>element</u>	p	Selects all <p> elements
<u>element,element</u>	div, p	Selects all <div> elements and all <p> elements
<u>element element</u>	div p	Selects all <p> elements inside <div> elements

**Key words:** Adding Pre-Made CSS Stylesheets to Your Website, new CSS selectors.

## 9. Understanding Templating vs. Layouts

- برای فهم بیشتر این قسمت حتماً کدهای اصلی را در فایل پروژه بررسی کنید.  
- توسط template می توان الگوی های پویا و متغیر برای html برای صفحات مختلف ایجاد کرد.

- توسط `layouts` می توان، کدهای تکراری `html` که در همه صفحات وجود دارد توسط `ejs` به صورت فایل های جدا در آورد. مثلاً فایل های `footer.ejs` و `header.ejs` کدهای تکراری ابتدایی و انتهایی همه صفحات پروژه می باشد. بنابراین `layout` `header` و `footer` به عنوان `html` شناخته می شوند.



```
header.ejs
1  <!DOCTYPE html>
2  <html lang="en" dir="ltr">
3
4  <head>
5    <meta charset="utf-8">
6    <title>To Do List</title>
7    <link rel="stylesheet" href="css/styles.css">
8  </head>
9
10 <body>
11
```

The code editor shows the content of the "header.ejs" file. It contains the opening HTML tag, the head section with meta tags and a title, and the body section which is currently empty.

```

    footer.ejs
1 </body>
2
3
4 <footer>
5   Copyright 2018 The App Brewery
6 </footer>
7
8 </html>
9

about.ejs
1 <%- include("header"); -%>
2
3 <h2>This is the about page</h2>
4
5 <p>as;dfjk a;sdfj;l 'sfd'k'; ;lsjdflsjf;lsk ;ksjf;sjf</p>
6
7 <%- include("footer"); -%>
8

```

**Key words:** creating new pages with ejs, what is ejs Layouts, header and footer layouts.

## 10. Understanding Node Module Exports How to Pass Functions and Data between Files

- در Node.js این قابلیت وجود دارد، که module اختصاصی خودمان را در فایل js. جداگانه ایجاد، و در جاهای مختلف پرورش از method و properties آن استفاده کرد.
- در یک module می توان method و properties های مختلف ایجاد کرد ، و سپس توسط export.methodOrProperties دادا و یا تابع ها در مکان فراخوانی شده ارسال کرد.
- یک module در واقع یک object می باشد و ویژگی آن را به ارث برده است.

## Node.js

About these Docs

Usage & Example

Assertion Testing

Async Hooks

Buffer

C++ Addons

C/C++ Addons - N-API

Child Processes

Cluster

Command Line Options

Console

# The `module` Object

Added in: v0.1.16

- <Object>

In each module, the `module` free variable is a reference to the `module` object. It exports module-global. `module` is not actually a global but

## module.children

Added in: v0.1.16

- <module[]>

The module objects required for the first time by this one.

## module.exports

Added in: v0.1.16

- <Object>

```
Module {
  id: '/Users/angelayu/Documents/LAB/OnlineCourse',
  exports: {},
  parent: null,
  Module {
    id: '.',
    exports: {},
    parent: null,
    filename: '/Users/angelayu/Documents/LAB/Onli
    loaded: false,
    children: [ [Object], [Object], [Circular] ],
    paths:
      [
        '/Users/angelayu/Documents/LAB/OnlineCourse'
      ]
  }
}

module.exports = "Hello World";
```

- Require کردن مازول ایجاد شده، در زمان فراخوانی آن لازم می باشد.

```
const date = require(__dirname + "/date.js");
```

## Properties of the Object constructor ↗

`Object.length`

Has a value of 1.

`Object.prototype`

Allows the addition of properties to all objects of type Object.

## Methods of the Object constructor ↗

`Object.assign()`

Copies the values of all enumerable own properties from one or more source objects to a target object.

## 6 ways to declare JavaScript functions

Dmitri Pavlutin | 22 Jun 2016

- ۳ شکل زیر نحوه export کردن در module را نشان می دهد، که شکل آخر خلاصه ترین کد آن می باشد.

```
module.exports.getDate = getDate;
```

```
var getDate = function() {
```

```
module.exports.getDate = function() {
```

```
exports.getDate = function() {
```

- کد زیر فراخوانی یک methoe در پروژه را نشان می دهد.

```
const day = date.getDate();
```

- شکل زیر ویژگی های داده Const را نشان می دهد. مقادیر Const به صورت مستقیم قابل تغییر نیستند ولی می توان با استفاده از method های مخصوص خودش آن را تغییر داد. (برای امنیت بیشتر سعی کنید به نوع انتخاب خود، یعنی Const ، let و var توجه کنید)

```
// throws an error - Uncaught SyntaxError: Missing initializer in const declaration
const FOO;

// const also works on objects
const MY_OBJECT = {'key': 'value'};

// Attempting to overwrite the object throws an error - Uncaught TypeError: Assignment to constant variable
MY_OBJECT = {'OTHER_KEY': 'value'};

// However, object keys are not protected,
// so the following statement is executed without problem
MY_OBJECT.key = 'otherValue'; // Use Object.freeze() to make object immutable

// The same applies to arrays
const MY_ARRAY = [];
// It's possible to push items into the array
MY_ARRAY.push('A'); // ["A"]
// However, assigning a new array to the variable throws an error - Uncaught TypeError: Assignment to constant variable
MY_ARRAY = ['B'];
```

**Key words:** Create your module and use it, Node Module Exports, Pass Functions and Data between Files , Const specifics in javaScript, making clean code.

## 11. Tip from Angela - Use Accountability in your Favour

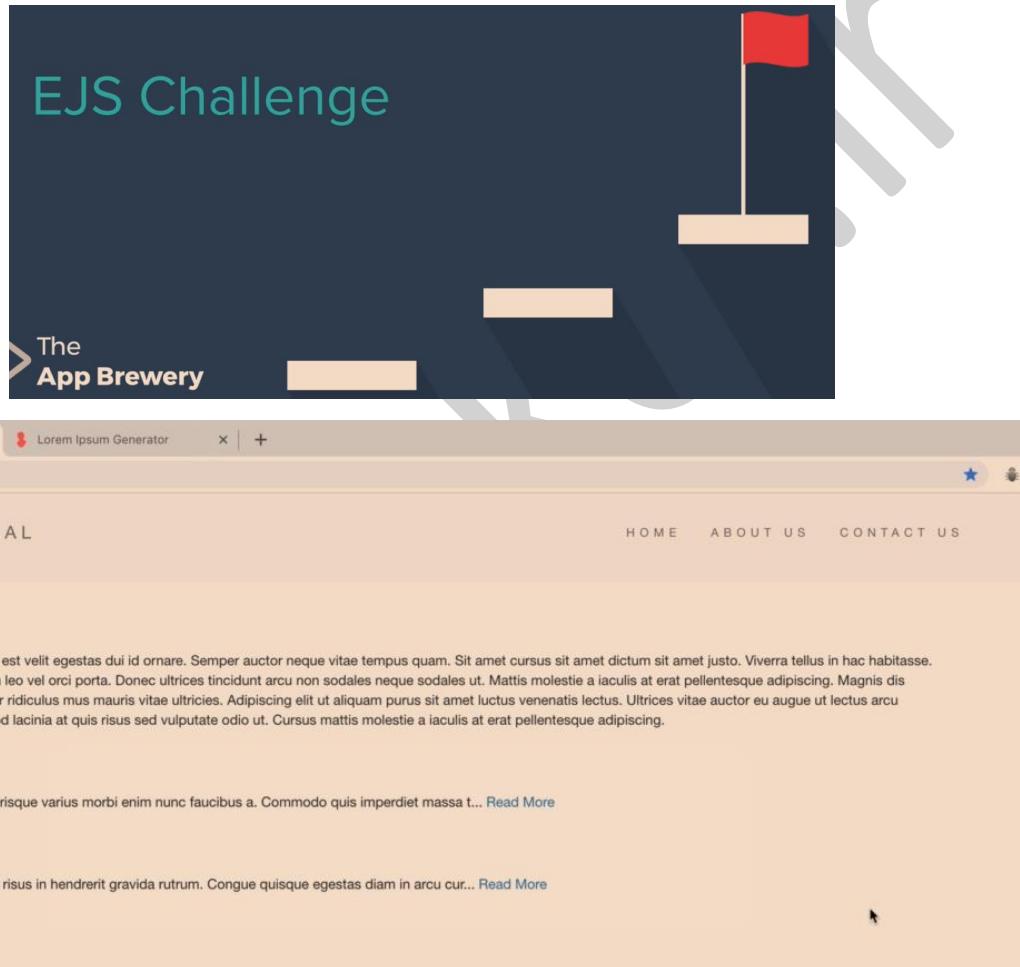
یکی از راه های مجبور شدن و مسئول بودن نسبت به انجام هدفی که تعیین کرده اید، این است که آن را به همه اعلان کنید، در این صورت برای اینکه نشان دهید آدم قوی ای هستید و کنار نمی کشید، آن هدف را دنبال خواهید کرد. مثلاً هدف خود را به دوستان و خانواده بگویید یا اینکه در شبکه های اجتماعی آن را اعلام کنید، در این صورت هدفتان را دنبال خواهید کرد و مسئولیتان بیشتر می شود.

## 23. Boss Level Challenge 3 - Blog Website

### 1. A New Challenge Format and What We'll Make A Blog

- در این قسمت پروژه ای که قرار است در پایان این چالش ایجاد شود را نشان می دهد.

- پروژه نهایی، یک بلاگ شخصی با صفحات خاص می باشد و این قابلیت را دارد که بتوان پست جدید نیز برای بلاگ ایجاد کرد.
- این چالش مرکز بر روی EJS می باشد تا چیزی که یاد گرفته اید را در عمل پیاده کنید.
- چالش ها را با مرکز حل کنید، و اگر مجبور شدید به راه حل ها مراجعه کنید. درگیر چالش شوید و طعم حل مسأله را بچشید. (خیلی سریع به راه حل مراجعه نکنید، حداقل یک ساعت درگیر شوید برای هر قسمتی که نمی توانید حل کنید).

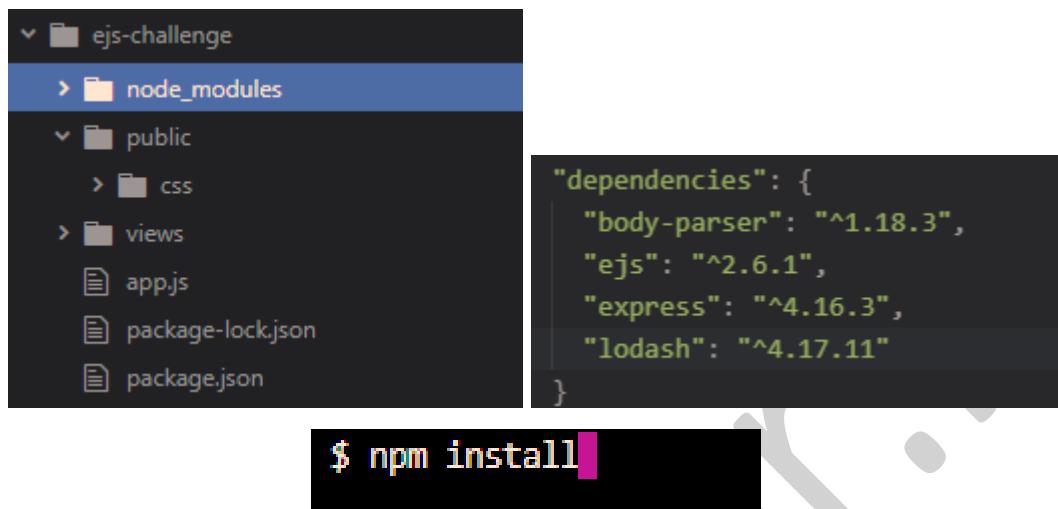


**Key words:** EJS challenges, creating Blog.

## 2. Setting Up the Blog Project

- در این قسمت فایل های اولیه پروژه را نشان می دهد.
- در حالت کلی وقتی پروژه ای را از `node_modules` می کنیم، پوشش `clone` ، `github` به صورت `dependencies` می باشد(یعنی وجود ندارد) تا حجم فایل های پروژه سنگین نشود، ولی `gitignore`

آن مشخص است تا بتوان node\_modules مورد نظر را نصب کرد. کافیست بعد از clone کردن پروژه، از دستور npm install در ترمینال استفاده کنیم تا مازول های مورد نیاز نصب شود.



**Key words:** Setting Up the Blog Project, npm install for Github clone.

3 – 44 challenges

## Array.prototype.forEach()

Jump to: Syntax Description Examples Polyfill Specifications Browser compatibility See also

Web technology for developers > JavaScript > JavaScript reference > Standard built-in objects > Array > Array.prototype.forEach()

Related Topics

Standard built-in objects

Array

Properties

Array.length

Array.prototype

The `forEach()` method executes a provided function once for each array element.

**JavaScript Demo: Array.forEach()**

```
1 var lettersArray = ['a', 'b', 'c'];  
2  
3 lettersArray.forEach(function(letter) {  
4   console.log(letter);  
5 });  
6  
7 // expected output: "a"  
8 // expected output: "b"  
9 // expected output: "c"  
10
```

```

MohammadReza@MohammadReza MINGW64 ~
$ nodemon app.js
[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs
[nodemon] starting `node app.js`
Server started on port 3000
rs
[nodemon] starting `node app.js`
Server started on port 3000

```

Express

Home Getting started **Guide** API reference Advanced topics Resources

### Route parameters

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys.

```

Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }

```

To define routes with route parameters, simply specify the route parameters in the path of the route as shown below.

```

app.get('/users/:userId/books/:bookId', function (req, res) {
  res.send(req.params)
})

```

The name of route parameters must be made up of "word characters" ([A-Za-z0-9\_]).

https://lodash.com

**Lodash** | A modern JavaScript utility library delivering modularity, performance & extras.

Documentation FP Guide

```

_.defaults({ 'a': 1 }, { 'a': 3, 'b': 2 });
// → { 'a': 1, 'b': 2 }
_.partition([1, 2, 3, 4], n => n % 2);
// → [[1, 3], [2, 4]]

```

Star 34,961 Fork 3,626 Follow @bestiejs Tweet

### Download

- Core build (~4kB gzipped)
- Full build (~24kB gzipped)

### Example

```
__.replace('Hi Fred', 'Fred', 'Barney');
// => 'Hi Barney'
```

### Example

```
__.lowerCase('--Foo-Bar--');
// => 'foo bar'

_.lowerCase('fooBar');
// => 'foo bar'

_.lowerCase('__FOO_BAR__');
// => 'foo bar'
```

## JavaScript String substring() Method

[◀ Previous](#)

[JavaScript String Reference](#)

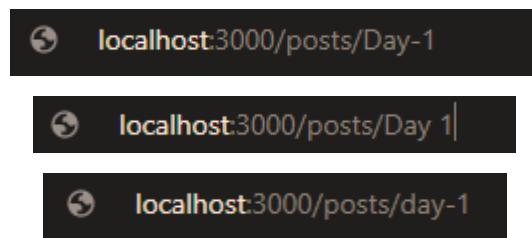
### Example

Extract characters from a string:

```
var str = "Hello world!";
var res = str.substring(1, 4);
```

- در url یا route یا `route` علامت - ، \_ و فاصله ، به یک معنا می باشد و همچنین کوچکی یا بزرگی حروف بی تأثیر می باشد. یعنی عبارات (A b) و (a-b) و (a\_b) و (a-b) و (a-b) برابر می باشند. علت آن تابع `lowercase` در مژول `lodash` می باشد، که خط تیره ها را به فاصله و حروف بزرگ را به کوچک تبدیل می کند.

```
_lowerCase(req.params.postName); // LowerCase ignores the "-" or "_" or " " in postName.
```



**Key words:** review, survey EJS, lodash module, forEach function, “rs” in command line, lowercase properties, using substring in javascript, and etc.

## 45. Tip from Angela - When Life Gives You Lemons

زندگی و شرایط دنیا هیچ وقت عادلانه نیست و نخواهد بود. همیشه یکسری مشکلات و اتفاقات ناخواسته پیش می آید که دست شما نیست. ولی همین اتفاقات باعث رشد و تعالی شما خواهد شد. در حالت کلی شما می توانید یکسری چیزها را تغییر دهید و یکسری چیزها را نمی توانید. حرف مردم، اذیت شدن توسط دیگران، شرایط محیطی نا مناسب و... چیزهایی هستند نمی توانید تغییر دهید، اما دید شما و نحوه برخورد شما با مسائل قابل تغییر است. شما می توانید همه مشکلات و اتفاقات را با دید و ذهنیت مناسب برطرف کنید و به سمت جلو حرکت کنید. یک ذهنیت مناسب در هر شرایطی پیروز است.

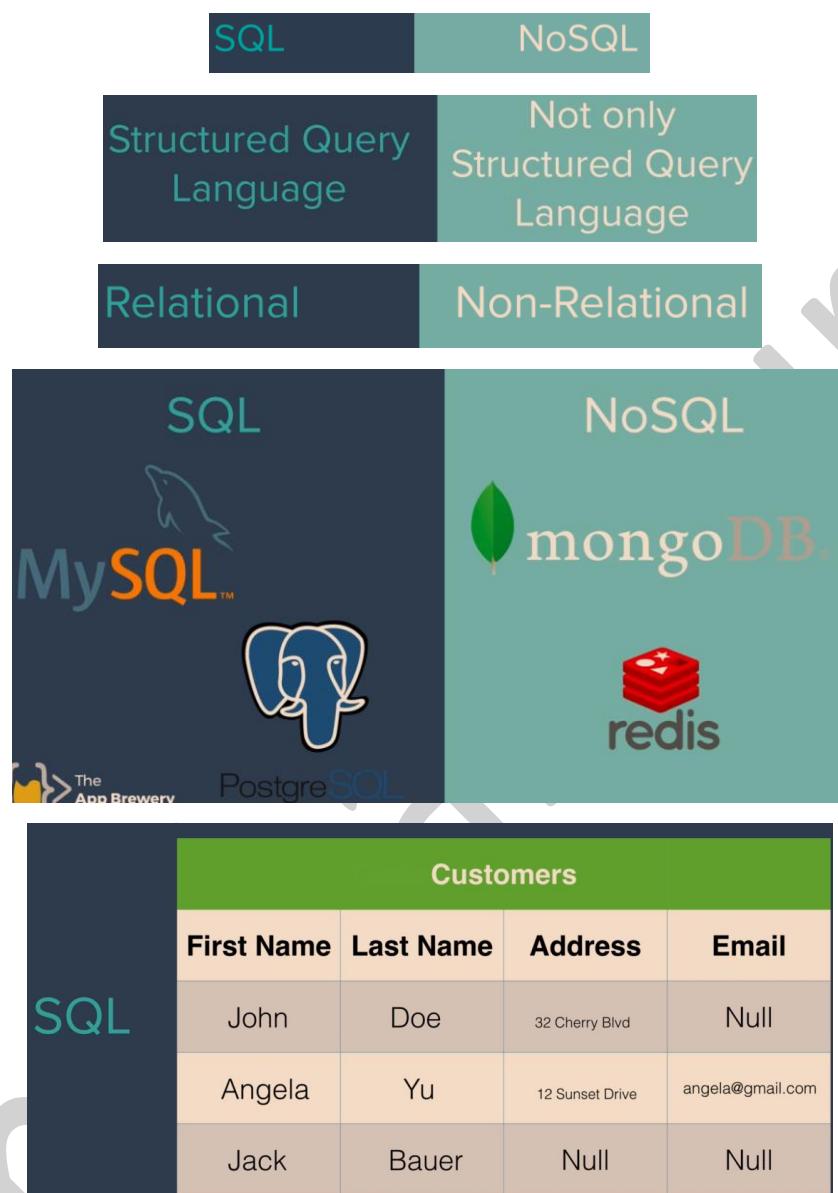
## 24. Databases

### 1. Databases Explained SQL vs. NOSQL

برای ذخیره سازی اطلاعات از database یا پایگاه داده ها استفاده می شود. -  
Database ها برندهای مختلفی دارند، ولی همه آن ها در دو دسته SQL و NoSQL قرار می گیرند. -  
SQL نام دیگر Sequel می باشد و NoSQL یعنی هرچیزی به غیراز SQL می باشد. -  
SQL ساختاری قدیمی و مقرراتی دارد. ساختار آن به صورت Table می باشد. Relational می باشد. -  
از قبل باید نیازها یا scheme مشخص باشد. برای دیتای حجمی نیاز به سخت افزار قوی دارد. MySQL می باشد. -  
یکی از این پایگاه ها می باشد.

NoSQL ساختاری جدید و امروزی دارد. ساختار آن استنادی می باشد. برای Relation وسیع مناسب نیست. دیتای آن به صورت چند فایل یا distributed می باشد. تغییر مقادیر آن راحت و انعطاف پذیر می باشد. مناسب برنامه هایی که نیاز های اولیه آن مشخص نیست و قرار است در آینده توسعه یابد. -  
یکی از این پایگاه ها می باشد. MongoDB

- در حالت کلی هر دو مزایای خود را دارند و باید با توجه به نیاز پروژه SQL یا NoSQL را انتخاب کرد.



## NoSQL

```
{  
  first_name: "John",  
  last_name: "Doe",  
  address: "32 Cherry Blvd"  
}  
  
{  
  first_name: "Angela",  
  last_name: "Yu",  
  address: "12 Sunset Drive",  
  email: "angela@gmail.com"  
}  
  
{  
  first_name: "Jack",  
  last_name: "Bauer"  
}
```

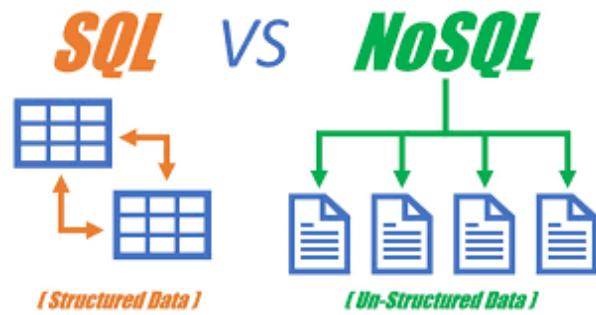
I'm SQL

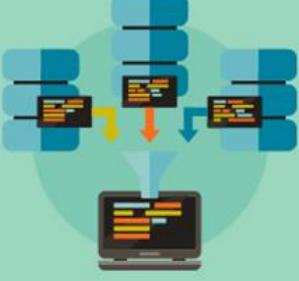


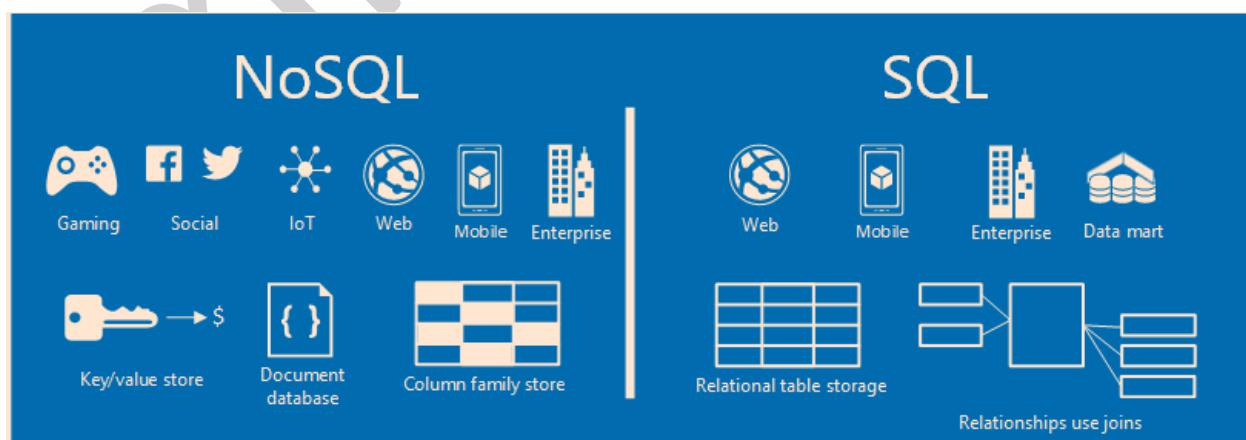
I'm NoSQL



MySQL	MongoDB
More Mature	Shiny and New
Table Structure	Document Structure
Requires a Schema	More Flexible to Changes
Great with Relationships	Not Great with Complex Relationships
Scales Vertically	Horizontally Scalable



<h3>SQL</h3>  <h4>Relational Data Model</h4> <p><b>Pros</b></p> <ul style="list-style-type: none"> <li>&gt; Easy to use and setup.</li> <li>&gt; Universal, compatible with many tools.</li> <li>&gt; Good at high-performance workloads.</li> <li>&gt; Good at structure data.</li> </ul> <p><b>Cons</b></p> <ul style="list-style-type: none"> <li>&gt; Time consuming to understand and design the structure of the database.</li> <li>&gt; Can be difficult to scale.</li> </ul>	<h3>NoSQL</h3>  <h4>Document Data Model</h4> <p><b>Pros</b></p> <ul style="list-style-type: none"> <li>&gt; No investment to design model.</li> <li>&gt; Rapid development cycles.</li> <li>&gt; In general faster than SQL.</li> <li>&gt; Runs well on the cloud.</li> </ul> <p><b>Cons</b></p> <ul style="list-style-type: none"> <li>&gt; Unsuitable for interconnected data.</li> <li>&gt; Technology still maturing.</li> <li>&gt; Can have slower response time.</li> </ul>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



**Key words:** SQL or sequel databases, NoSQL Databases, what is database, cons and pros of databases, mySql vs mongoDB.

## 25. SQL

### 1. SQL Commands CREATE Table and INSERT Data

- در ابتدای کار با هر Database ای باید CRUD را یاد بگیرد تا بر آن مسلط شوید.

The diagram features a large dark blue square at the top containing the words "Create", "Read", "Update", and "Destroy" stacked vertically in light blue and white text. Below this is a smaller dark blue rectangle containing the text "Create This with SQL" in light blue. Underneath that is a table titled "Products" with three columns: "id" (with a key icon), "name", and "price". A single row is shown with values: id=1, name=Pen, price=1.20.

- توسط کد زیر می توان یک جدول جدید، با نام های ستون و نوع داده آن ایجاد کرد.

```
1 CREATE TABLE products (
2   id INT NOT NULL,
3   name STRING,
4   price MONEY,
5   PRIMARY KEY (id)
6 )
```

## The SQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

### Syntax

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ...
);
```

## MySQL Data Types

In MySQL there are three main data types: text, number, and date.

## SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a database table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only one primary key, which may consist of single or multiple fields.

## SQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

برای هر جدول ضروری است، و نشان دهنده شناسه هر ردیف در جدول می باشد. Primary key -

# The SQL INSERT INTO Statement

The `INSERT INTO` statement is used to insert new records in a table.

## INSERT INTO Syntax

It is possible to write the `INSERT INTO` statement in two ways.

The first way specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

توسط `INSERT` می توان به جدول مقادیر جدید وارد نمود. -

```
1 INSERT INTO products
2 VALUES (1, "Pen", 1.20)
```

برای فهم بیشتر به [w3school](#) مراجعه کنید. -

**Key words:** SQL Commands CREATE Table and INSERT Data.

## 2. SQL Commands READ, SELECT, and WHERE

برای خواندن دیتا از دستور `Select` مطابق شکل زیر استفاده می شود. (علامت \* یعنی تمام جدول). -

# The SQL SELECT Statement

The `SELECT` statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

## SELECT Syntax

```
SELECT column1, column2, ...
FROM table_name;
```

```
1 | SELECT * FROM 'products';
```

توسط **where** می توان ردیف با مقدار یا شرط مورد نظر را پیدا کرد.

## The SQL WHERE Clause

The **WHERE** clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

### WHERE Syntax

```
SELECT column1, column2, ...
  FROM table_name
 WHERE condition;
```

```
1 | SELECT * FROM products WHERE id=1
```

**Key words:** SQL Commands READ, SELECT, and WHERE.

### 3. Updating Single Values and Adding Columns in SQL

توسط **Update** می توان به جاهای خالی در جدول مقدار داد.

## Update a Single Value

Products		
<b>id</b> 	<b>name</b>	<b>price</b>
1	Pen	1.20
2	Pencil	0.8

# The SQL UPDATE Statement

The `UPDATE` statement is used to modify the existing records in a table.

## UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

```
1 UPDATE products
2 SET price = 0.80
3 WHERE id=2|
```

توسط `Alter` می توان ستونی اضافه و یا حذف نمود. -

## Update by Adding a Column

Products			
<i>id</i> 	<i>name</i>	<i>price</i>	<i>stock</i>
1	Pen	1.20	32
2	Pencil	0.80	12

## ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

```
1 ALTER TABLE products  
2 ADD stock INT
```

**Key words:** Updating Single Values and Adding Columns in SQL.

#### 4. SQL Commands DELETE

توسط Delete می توان، یک مقدار یا همه مقادیر جدول را حذف نمود.

### The SQL DELETE Statement

The **DELETE** statement is used to delete existing records in a table.

#### DELETE Syntax

```
DELETE FROM table_name WHERE condition;
```

```
1 DELETE FROM products  
2 WHERE id = 2
```

**Key words:** SQL Commands DELETE.

#### 5. Understanding SQL Relationships, Foreign Keys and Inner Joins

در این قسمت توسط Inner Join بین، جداول relation یا لینک ایجاد می کنیم.

# Create This with SQL

Orders			
<b>id</b> 	<b>order_number</b>	<b>customer_id</b> 	<b>product_id</b> 
1	4362	2	1
2	3254	1	1

- توسط **foreign key** می توان جدول را با **primary key** جدول دیگر ارتباط داد.

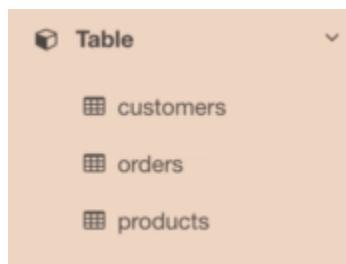
## SQL FOREIGN KEY Constraint

The **FOREIGN KEY** constraint is used to prevent actions that would destroy links between tables.

A **FOREIGN KEY** is a field (or collection of fields) in one table, that refers to the **PRIMARY KEY** in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

```
PRIMARY KEY (id),  
FOREIGN KEY (customer_id) REFERENCES customers(id),  
FOREIGN KEY (product_id) REFERENCES products(id) ]
```



- توسط **Inner Join** می توان ۲ جدولی که توسط **primary key** و **Foreign key** مرتبط شدن را، در

یک جدول جدید بسط داد و اطلاعات بیشتری را مشاهده نمود.

- به همین علت **SQL relation** در SQL راحت می باشد.

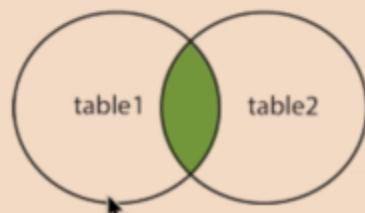
# SQL INNER JOIN Keyword

The INNER JOIN keyword selects records that have matching values in both tables.

## INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2 ON table1.column_name = table2.column_name;
```

INNER JOIN



```
1 SELECT orders.order_number, customers.first_name, customers.last_name, customers.address
2 FROM orders
3 INNER JOIN customers ON orders.customer_id = customers.id
```

order_number	first_name	last_name	address
4362	Angela	Yu	12 Sunset Drive

**Key words:** Understanding SQL Relationships, Foreign Keys and Inner Joins.

## 6. Tip from Angela - Find All the Hard Working People

یکی از راه کارهای افزایش انگیزه، بودن در محیطی می باشد که افراد در حال تلاش و کوشش می باشند. مثلاً اگر در خانه نمی توانید کار کنید، می توانید به کتابخانه بروید و وقتی ببینید دیگران سخت مشغولند، شما نیز سخت مشغول خواهید شد. یا می توانید رفیق داشته باشید که به یکدیگر برای پیشرفت کمک کنید.

## 26. MongoDB

### 1. Installing MongoDB on Mac

### 2. Installing MongoDB on Windows

### 3. MongoDB CRUD Operations in the Shell Create

- برای استفاده از هر پایگاه داده ای، باید ابتدا عملیات CRUD آن را یاد بگیریم.
- برای راه اندازی سرورهای mongoDB در local host از دستور mongod استفاده کنیم.
- در ادامه در ترمینال دیگر با دستور mongo وارد محیط shell mongo می شود.
- دستور dbs ، پایگاه داده های موجود را نشان می دهد.
- با دستور use DataBaseName می توان یک DB جدید ایجاد یا به آن سوئیچ کرد.
- دستور db موقعیت فعلی(یعنی حضور در پایگاه داده مورد نظر) را در shell نشان می دهد.
- با دستور db.collection.insertOne(fields and values) می توان یک collection (ها) در پایگاه داده مورد نظر ایجاد کرد و یک یا چند document (را) وارد collection کرد.

```
> use shopDB  
switched to db shopDB
```

```
> show dbs  
admin    0.000GB  
config   0.000GB  
local    0.000GB
```

- db.collection.insertOne() New in version 3.2
- db.collection.insertMany() New in version 3.2

In MongoDB, insert operations target a single collection. All write operations in MongoDB are of a single document.

```
db.users.insertOne(collection  
{  
  name: "sue", field: value  
  age: 26, field: value  
  status: "pending" field: value } document  
}  
)
```

Products			
<b>id</b> 	<b>name</b>	<b>price</b>	<b>stock</b>
1	Pen	1.20	32
2	Pencil	0.80	12

```
> db.products.insertOne({_id: 1, name: "Pen", price: 1.20})
{ "acknowledged" : true, "insertedId" : 1 }
```

> db  
shopDB

> show collections  
products

- هر DB ایجاد شده، از collection ها، و هر document از documents تشكیل شده است.
- استفاده از mongo doc و help درک بیشتر فراموش نشود.

**Key words:** MongoDB CRUD Operations in the Shell Create, collection or table, record or document.

#### 4. MongoDB CRUD Operations in the Shell Reading & Queries

توسط db.collection.find() می توان دیتا را خواند، و توسط queries و projection می توان دیتای مورد نظر را فیلتر کرد.

##### Read Operations

Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- db.collection.find()

You can specify query filters or criteria that identify the documents to return.

```
db.users.find(
  { age: { $gt: 18 } },
  { name: 1, address: 1 }
).limit(5)
```

← collection  
← query criteria  
← projection  
← cursor modifier

## Definition

```
db.collection.find(query, projection)
```

### MONGO SHELL METHOD:

This page documents the `mongo` shell method, and does *not* refer to the MongoDB Node.js driver (or any other driver) method. For corresponding MongoDB driver API, refer to your specific [MongoDB driver](#) documentation instead.

Selects documents in a collection or view and returns a [cursor](#) to the selected documents.

Parameter	Type	Description
<code>query</code>	document	Optional. Specifies selection filter using <a href="#">query operators</a> . To return all documents in a collection, omit this parameter or pass an empty document ( <code>{}</code> ).
<code>projection</code>	document	Optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit this parameter. For details, see <a href="#">Projection</a> .

```
> db.products.find()
{ "_id" : 1, "name" : "Pen", "price" : 1.2 }
{ "_id" : 2, "name" : "Pencil", "price" : 0.8 }
```

```
> db.products.find({name: "Pencil"})
{ "_id" : 2, "name" : "Pencil", "price" : 0.8 }
```

```
> db.products.find({_id: 1}, {name: 1})
{ "_id" : 1, "name" : "Pen" }
> db.products.find({_id: 1}, {name: 1, _id: 0})
{ "name" : "Pen" }
```

**Key words:** MongoDB CRUD Operations in the Shell Reading & Queries.

## 5. MongoDB CRUD Operations in the Shell Update

- در این قسمت نحوه `Update` کردن `collection` را نشان می دهد، و ستون `stock` به آن اضافه می شود.

```
db.collection.updateOne(filter, update, options) ¶
```

```
db.users.updateMany(  
  { age: { $lt: 18 } },  
  { $set: { status: "reject" } }  
)
```

collection ←  
update filter ←  
update action ←

```
> db.products.updateOne({_id: 1}, {$set: {stock: 32}})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

```
> db.products.find()  
{ "_id" : 1, "name" : "Pen", "price" : 1.2, "stock" : 32 }  
{ "_id" : 2, "name" : "Pencil", "price" : 0.8 }
```

```
> db.products.updateOne({_id: 2}, {$set: {stock: 12}})  
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }  
> db.products.find()  
{ "_id" : 1, "name" : "Pen", "price" : 1.2, "stock" : 32 }  
{ "_id" : 2, "name" : "Pencil", "price" : 0.8, "stock" : 12 }
```

**Key words:** MongoDB CRUD Operations in the Shell Update.

## 6. MongoDB CRUD Operations in the Shell Delete

- در این قسمت نحوه حذف collection از document را نشان می دهد.

## Delete Operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- `db.collection.deleteOne()` *New in version 3.2*
- `db.collection.deleteMany()` *New in version 3.2*

In MongoDB, delete operations target a single collection. All write operations in MongoDB are **atomic** on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These **filters** use the same syntax as read operations.

```
db.users.deleteMany(  
  { status: "reject" } )
```



```
> db.products.deleteOne({_id: 2})  
{ "acknowledged" : true, "deletedCount" : 1 }  
> db.products.find()  
{ "_id" : 1, "name" : "pen", "price" : 1.2, "stock" : 32 }
```

**Key words:** MongoDB CRUD Operations in the Shell Delete.

## 7. Relationships in MongoDB

- برای ایجاد **relationship** در روش اول، در شکل زیر در داخل **collection** یک آرایه ای از **reviews** ایجاد می شود و یک **document** **reviews** بین محصول، و نظرات محصول ایجاد می شود.

```
db.products.insert(  
  {  
    _id: 3,  
    name: "Rubber",  
    price: 1.30,  
    stock: 43,  
    reviews: [  
      {  
        authorName: "Sally",  
        rating: 5,  
        review: "Best rubber ever!"  
      },  
      {  
        authorName: "John",  
        rating: 5,  
        review: "Awesome rubber!"  
      }  
    ]  
  })
```

- روش دیگر relationship collection ایجاد جدیدی می باشد، که در آن ، به document های collection orders اشاره کند. (مثلاً ایجاد collection به نام orders و اشاره به id مخصوصات در آن).

```
{  
    _id: 1,  
    name: "Pen",  
    price: 1.20,  
    stock: 32  
}  
  
{  
    _id: 2,  
    name: "Pencil",  
    price: 0.80,  
    stock: 12  
}  
  
{  
    orderNumber: 3243,  
    productsOrdered: [1, 2]  
}
```

**Key words:** Relationships in MongoDB, 2 method to create relations in mongoDB, embedded document.

## 8. Working with The Native MongoDB Driver

- چیزی که تا کنون گفته شد، دسترسی مستقیم به MongoDB از طریق shell بود. حال برای دسترسی به MongoDB از طریق سایر زبان های برنامه نویسی Driver هایی وجود دارد که واسطه آن زبان با MongoDB می باشد.

- از آن جا که ما با Node.JS کار می کنیم می توانیم از native mongoDB driver مخصوص استفاده کنیم که در این قسمت خلاصه ای از نحوه استفاده از آن را نشان می دهد.  
- روش دیگر استفاده از MongoDB از طریق node.js استفاده از ماژول محبوب mongoose می باشد، که از کدهای کمتری استفاده می کند که در فصل بعد به توضیح روش استفاده از آن می پردازیم.

# MongoDB with Node.JS



## Start Developing with MongoDB

Connect your application to your database with one of our official libraries.

The following libraries are officially supported by MongoDB. They are actively maintained, support new MongoDB features, and receive bug fixes, performance enhancements, and security patches.

C	C++	C#
Go	Java	Node.js
PHP	Python	Ruby
Rust	Scala	Swift

`npm install mongodb --save`

`mongod`

`node app.js`

همان درخواست کننده برای ارسال اطلاعات می باشد و عملیات ارسال در آن انجام می شود -  
یک جور اعتبارسنج یا validation برای دیتای ارسالی می باشد - Assert

- عمل `client.close()` بعد از پایان ارسال دیتا، برای قطع ارتباط استفاده می شود.

## Connect to MongoDB

Create a new `app.js` file and add the following code to try out some basic CRUD operations using the MongoDB driver.

Add code to connect to the server and the database `myproject`:

```
const MongoClient = require('mongodb').MongoClient;
const assert = require('assert');

// Connection URL
const url = 'mongodb://localhost:27017';

// Database Name
const dbName = 'myproject';

// Create a new MongoClient
const client = new MongoClient(url);

// Use connect method to connect to the Server
client.connect(function(err) {
  assert.equal(null, err);
  console.log("Connected successfully to server");

  const db = client.db(dbName);

  client.close();
});
```

## Insert a Document

Add to `app.js` the following function which uses the `insertMany` method to add three documents to the `documents` collection.

```
const insertDocuments = function(db, callback) {
  // Get the documents collection
  const collection = db.collection('documents');
  // Insert some documents
  collection.insertMany([
    {a : 1}, {a : 2}, {a : 3}
  ], function(err, result) {
    assert.equal(err, null);
    assert.equal(3, result.result.n);
    assert.equal(3, result.ops.length);
    console.log("Inserted 3 documents into the collection");
    callback(result);
  });
}
```

## Find All Documents

Add a query that returns all the documents.

```
const findDocuments = function(db, callback) {
  // Get the documents collection
  const collection = db.collection('documents');
  // Find some documents
  collection.find({}).toArray(function(err, docs) {
    assert.equal(err, null);
    console.log("Found the following records");
    console.log(docs);
    callback(docs);
  });
}
```

- در صورت برخورد خطاهای مختلف در `log` ترمینال، می توان آن را گوگل و مشکل را برطرف نمود.

A screenshot of the Stack Overflow website. The top navigation bar includes links for Home, PUBLIC, and Stack Overflow. A search bar is present. The main content area displays a question with the title "Avoid 'current URL string parser is deprecated' warning by setting useNewUrlParser to true".

You can use CTRL + C in your Terminal to shut down your mongod connection.

If you have closed down Terminal or Hyper and forgot to close down your mongod connection, you might get an error that says:

```
2018-11-04T16:17:53.473+1300 E STORAGE [initandlisten] Failed to set up listener: SocketException: Address already in use  
2018-11-04T16:17:53.474+1300 I CONTROL [initandlisten] now exiting  
2018-11-04T16:17:53.474+1300 I CONTROL [initandlisten] shutting down with code:48
```

In this case, you'll have to follow these steps to manually shut down the mongod process:

1. Open up a fresh Hyper Terminal tab

2. Paste the command below into Hyper:

```
sudo pkill -f mongod
```

3. Now enter the password that you use to log on to the Mac.

4. Open a new Hyper terminal, you should now be able to run the mongod command again.

**Key words:** The Native MongoDB Driver vs mongoose, the special connection between mongoDB and Node.js, CRUD in node.js.

## 10. Tip from Angela - Daily Routines

یکی از روش های انجام دادن کارهای روزانه، داشتن برنامه و روتین مشخص روزانه می باشد. یعنی در حالت عادی ( به غیر از زمان های خاص مثل مسافرت، جلسه و...) شما باید به آن روتین پایبند باشید. در این صورت وقتی که از خواب بیدار می شوید، می دانید که باید چه کارهایی را امروز انجام دهید. مسئله بعدی این است که کار و وظیفه سخت را در اولویت قرار دهید و آن را زودتر از بقیه امور انجام دهید. در این صورت احساس رضایتی به دست خواهد آورد که باعث می شود، بقیه کارها را به راحتی انجام دهید. یادتان باشد اهمال کاری خر است.

# 27. Mongoose

## 1. Introduction to Mongoose

یک ابزار برای کار با Object Document Mapper یا ODM Mongoose -

(documentDB) می باشد، و کار با mongoDB را سریع تر و راحت می کند.

توسط دستور mongo shell در dropDatabase می توان، Database را حذف نمود. -

```
> show dbs
admin      0.000GB
config     0.000GB
fruitsDB   0.000GB
local      0.000GB
shopDB     0.000GB
> use fruitsDB
switched to db fruitsDB
> db.dropDatabase()
{ "dropped" : "fruitsDB", "ok" : 1 }
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
shopDB     0.000GB
>
```

- توسط connect می توان به پایگاه داده وصل شد و اگر وجود نداشت آن را ایجاد می کند.
- توسط Schema ، ساختار دیتای هر record یا document را مشخص می کنیم.
- توسط collection یک model ایجاد می شود و هم یک متغیر برای ایجاد record در آن خواهیم داشت.

```
2  const mongoose = require("mongoose");
3
4  mongoose.connect("mongodb://localhost:27017/fruitsDB"); // connect to the database
5
6  const fruitSchema = mongoose.Schema({ // create a Schema for our fruit
7    name: String,
8    rating: Number,
9    review: String
10   });
11
12  const Fruit = mongoose.model("Fruit", fruitSchema); // this creates our model
13
14  const fruit = new Fruit({
15    name: "apple",
16    rating: 7,
17    review: "It's sweet apple!"
18  });
19
20  fruit.save();
```

- در شکل زیر نحوه ذخیره یک دسته از record را نشان می دهد.

```

35   ///////////////////////////////// Create a bunch of fruits///
36
37   const kiwi = new Fruit ({
38     name: "Kiwi",
39     rating: 10,
40     review: "oh, it's so soft."
41   });
42
43   const orange = new Fruit ({
44     name: "Orange",
45     rating: 6,
46     review: "locious for me!"
47   });
48
49   const banana = new Fruit ({
50     name: "Banana",
51     rating: 10,
52     review: "My morning fruit."
53   });
54
55   Fruit.insertMany([kiwi, orange, banana], function(err){
56     if (err){
57       console.log(err);
58     } else {
59       console.log("Successfully saved all the fruits to fruitsDB");
60     }
61   });
62

```

```

> show collections
fruits
people
> db.fruits.find()
{
  "_id" : ObjectId("62283bfc9a6e459c06b3821f"),
  "name" : "apple",
  "rating" : 7,
  "review" : "It's sweet apple!",
  "__v" : 0
},
{
  "_id" : ObjectId("62283d15ee6d05dcc4e291f5"),
  "name" : "apple",
  "rating" : 7,
  "review" : "It's sweet apple!",
  "__v" : 0
},
{
  "_id" : ObjectId("622843ca2bf889a946ac6c48"),
  "name" : "Kiwi",
  "rating" : 10,
  "review" : "oh, it's so soft.",
  "__v" : 0
},
{
  "_id" : ObjectId("622843ca2bf889a946ac6c49"),
  "name" : "Orange",
  "rating" : 6,
  "review" : "locious for me!",
  "__v" : 0
},
{
  "_id" : ObjectId("622843ca2bf889a946ac6c4a"),
  "name" : "Banana",
  "rating" : 10,
  "review" : "My morning fruit.",
  "__v" : 0
},
{
  "_id" : ObjectId("622843ca2bf889a946ac6c46"),
  "name" : "apple",
  "rating" : 7,
  "review" : "It's sweet apple!",
  "__v" : 0
}

```

**Key words:** Introduction to Mongoose, ODM or object document mapper, create connection to mongoDB, create and save records with mongoose.Schema, mongoose.model.

## 2. Reading from Your Database with Mongoose

- در شکل زیر از collection مورد نظر دیتا خوانده می شود، و مانند یک javascript object در می توانیم از آن استفاده کنیم.
- بعد از پایان تبادل دیتا، باید ارتباط با database را close کنیم.

```
64 Fruit.find(function(err, fruits){  
65     if (err){  
66         console.log(err);  
67     } else {  
68         mongoose.connection.close(); // it will disconnect the app.js on mongoDB  
69  
70         fruits.forEach(function(fruit){  
71             console.log(fruit.name);  
72         });  
73     }  
74 };  
75});
```

```
Angelas-MacBook-Pro:FruitsProject angelayu$ node app.js  
Apple  
Kiwi  
Orange  
Banna  
Angelas-MacBook-Pro:FruitsProject angelayu$
```

**Key words:** Reading from Your Database with Mongoose, closing connection.

## 3. Data Validation with Mongoose

- توسط validation می توان قوانینی برای record ها تعیین کرد، تا مقادیر وارد شده، مقادیر موردنیاز ما باشند، در غیر این صورت دیتا ذخیره نمی شود و خطای validation error ارسال می گردد.

## Validation



Before we get into the specifics of validation syntax, please keep the following rules in mind:

- Validation is defined in the `SchemaType`
- Validation is `middleware`. Mongoose registers validation as a `pre('save')` hook on every schema by default.
- You can disable automatic validation before save by setting the `validateBeforeSave` option
- You can manually run validation using `doc.validate(callback)` or `doc.validateSync()`
- You can manually mark a field as invalid (causing validation to fail) by using `doc.invalidate(...)`
- Validators are not run on undefined values. The only exception is the `required` validator.
- Validation is asynchronously recursive; when you call `Model#save`, sub-document validation is executed as well. If an error occurs, your `Model#save` callback receives it
- Validation is customizable

- در کد زیر توسط `name`، وجود `rating` ضروری است، همچنین `rating` باید بین ۱ تا ۱۰ باشد، تا اطلاعات مورد تأیید باشد.

```
5  const fruitSchema = mongoose.Schema({ // create a Schema for records in collection
6    name: {
7      type: String, // data validation
8      required: [true, "Please check your data entry, no name specified!"]
9    },
10   rating: {
11     type: Number,
12     min: 1, // data validation
13     max: 10
14   },
15   review: String
16 });
17
18 const Fruit = mongoose.model("Fruit", fruitSchema); // this will create collection
19
20 const fruit = new Fruit({
21   name: "peach",
22   rating: 10,
23   review: "It's ok fruit!"
24 });
25
26 fruit.save();
```

در شکل زیر `validation error` را نشان می دهد.

```
properties: {
  validator: [Function (anonymous)],
  message: 'Path `rating` (25) is more than maximum allowed value (10).',
  type: 'max',
  max: 10,
  path: 'rating',
  value: 25
},
kind: 'max',
path: 'rating',
value: 25,
reason: undefined,
[Symbol(mongoose:validatorError)]: true
}
},
_message: 'Fruit validation failed'
}
```

```
properties: {
  validator: [Function (anonymous)],
  message: 'Path `name` is required.',
  type: 'required',
  path: 'name',
  value: undefined
},
kind: 'required',
path: 'name',
value: undefined,
reason: undefined,
[Symbol(mongoose:validatorError)]: true
}
},
_message: 'Fruit validation failed'
}
```

Validator های گفته شده، تنها بخشی از آن بود، برای اطلاعات بیشتر به document مراجعه کنید.

**Key words:** Data Validation with Mongoose, saving correct and clean data, create rule for receiving data with validators.

#### 4. Updating and Deleting Data Using Mongoose

در این قسمت mongoose و update توسط delect را نشان می دهد.

## Updating

Each `model` has its own `update` method for modifying documents in the database without returning them to your application. See the [API](#) docs for more detail.

```
Tank.updateOne({ size: 'large' }, { name: 'T-90' }, function(err, res) {
  // Updated at most one doc, `res.nModified` contains the number
  // of docs that MongoDB updated
});
```

```
/////////////////// Update Database

Fruit.updateOne({name: "Kiwi"}, {rating: 1}, function(err){
  if (err){
    | | console.log(err);
  } else {
    | | console.log("Successfully Updated one document.");
  }
});
```

## Deleting

Models have static `deleteOne()` and `deleteMany()` functions for removing all documents matching the given `filter`.

```
Tank.deleteOne({ size: 'large' }, function (err) {
  if (err) return handleError(err);
  // deleted at most one tank document
});
```

```
/////////////////// delete one record
Fruit.deleteOne({name: "apple"}, function(err){
  if (err){
    | | console.log(err);
  } else {
    | | console.log("Successfully deleted one record.");
  }
});
```

```
////////////////// delete many records
Person.deleteMany({name: "shaker"}, function(err){
  if (err){
    console.log(err);
  } else {
    console.log("Successfully deleted many documents.");
  }
})
```

**Key words:** Updating and Deleting Data Using Mongoose.

## 5. Establishing Relationships and Embedding Documents using Mongoose

در این قسمت با تغییر ساختار record ، ساختار جدید برای personSchema در Scheme ایجاد می کنیم، در نتیجه می توان Relationship favouriteFruit را در آن ادغام کرد و نمود.

```
const fruit = new Fruit({
  name: "mango",
  rating: 10,
  review: "It's sweet!"
});
```

```
const personSchema = mongoose.Schema({
  name: String,
  age: Number,
  favouriteFruit: fruitSchema
});

const Person = mongoose.model("Person", personSchema);

const person = new Person({
  name: "shaker",
  age: 25,
  favouriteFruit: fruit
});

person.save();
```

```
> db.people.find()
{ "_id" : ObjectId("62287d34bc4e96696b39ca30"), "name" : "shaker", "age" : 25, "favouriteFruit" : { "name" : "mango", "rating" : 10, "review" : "It's sweet!", "_id" : ObjectId("62287d34bc4e96696b39ca2f") }, "_v" : 0 }
>
```

**Key words:** Establishing Relationships and Embedding Documents using Mongoose, make new structure for Schema, create Relationship.

## 6. Tip from Angela - Deep Work

موقع برنامه نویسی، به کار عمیق احترام بگذارید. یعنی موقع هر کاری فقط آن کار را انجام دهید. مثلاً موقع کار فقط به کار توجه کنید و تمرکزتان فقط روی این موضوع باشد، و نگذارید افکارتان به سمت موضوعات انحرافی و غیرمرتبط حرکت کند. ذهنتان را در کنترل بگیرید و به کار عمیق و متمرکز پایبند باشید. کیفیت زمانی، مهمتر از کیمیت زمانی می باشد.

## 28. Putting Everything Together

### 1. Let's take the ToDoList Project to the Next Level and Connect it with Mongoose

- پروژه todoList که بدون پایگاه داده بود، و با ری استارت شدن سرور اطلاعات پاک می شد، در این قسمت اطلاعات را توسط mongoose در mongoDB ذخیره می کند.

```

mongoose.connect("mongodb://localhost:27017/todolistDB", {useNewUrlParser: true});

const itemsSchema = {
  name: String
};

const Item = mongoose.model("Item", itemsSchema);

const item1 = new Item({
  name: "Welcome to your todolist!"
});

const item2 = new Item({
  name: "Hit the + button to add a new item."
});

const item3 = new Item({
  name: "<-- Hit this to delete an item."
});

const defaultItems = [item1, item2, item3];

Item.insertMany(defaultItems, function(err) {
  if (err) {
    console.log(err);
  } else {
    console.log("Successfully saved default items to DB.");
  }
});

```

**Key words:** config todoList project to mongoDB with mongoose.

## 2. Rendering Database Items in the ToDoList App

- در این قسمت نحوه خواندن اطلاعات از MongoDB، و همچنین select دیتای مورد نظر، و جلوگیری از تکرار ذخیره دیتا در MongoDB نشان داده می شود.

```
app.get("/", function(req, res) {

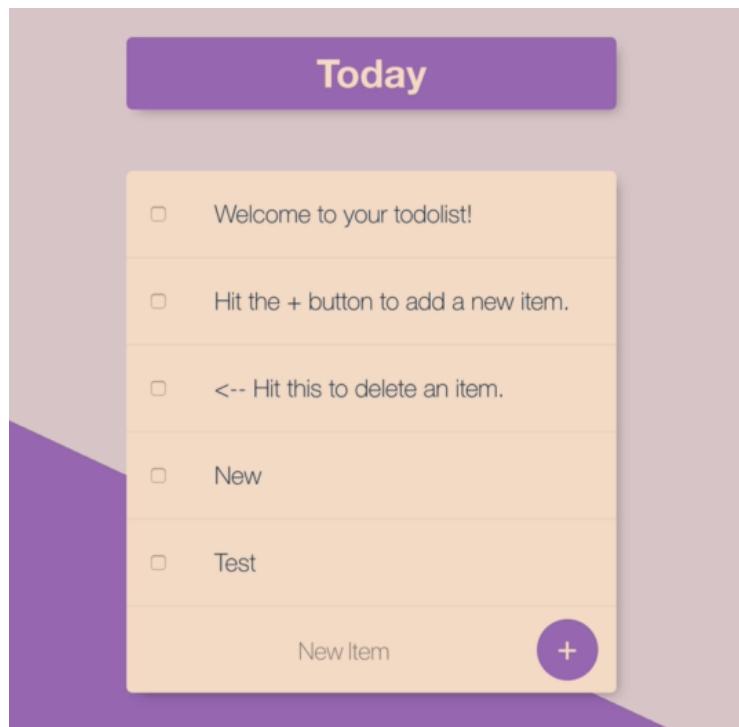
  Item.find({}, function(err, foundItems) {
    if (foundItems.length === 0) {

      Item.insertMany(defaultItems, function(err) {
        if (err) {
          console.log(err);
        } else {
          console.log("Successfully saved default items to DB.");
        }
      });
      res.redirect("/");
    } else {
      res.render("list", {
        listTitle: "Today",
        newListItems: foundItems
      });
    }
  });
});
```

**Key words:** Rendering Database Items in the ToDoList App.

### 3. Adding New Items to our ToDoList Database

- در این قسمت افزودن آیتم جدید توسط post request را در mongoDB ذخیره می کنیم.

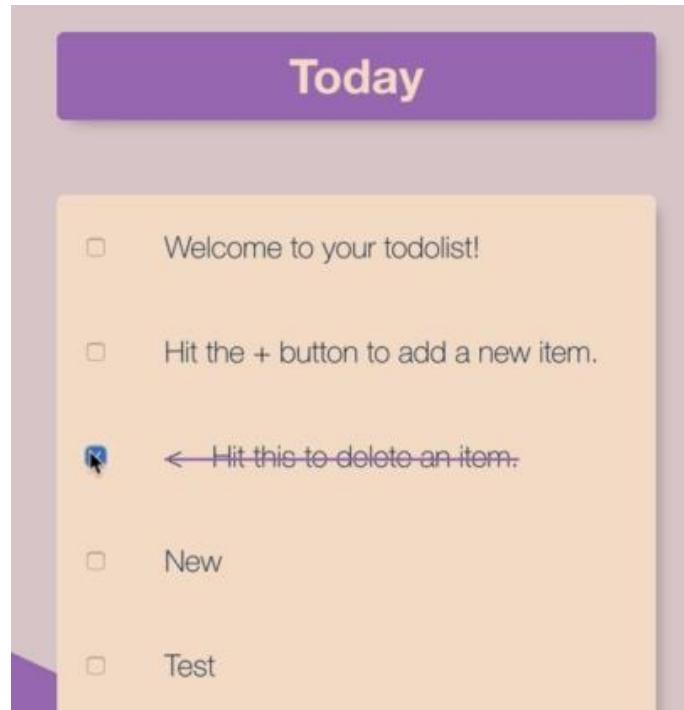


```
app.post("/", function(req, res) {  
  
    const itemName = req.body newItem;  
  
    const item = new Item({  
        name: itemName  
    });  
  
    item.save();  
  
    res.redirect("/");  
});
```

**Key words:** Adding New Items to our ToDoList Database.

#### 4. Deleting Items from our ToDoList Database

- در این قسمت نحوه حذف آیتم انتخاب شده توسط checkbox ، در MongoDB را نشان می دهد.



▲ In plain JavaScript simply put  
46      `onChange="this.form.submit()"`  
▼ into the form/input tag.

## Mongoose findByIdAndRemove()

```
<ModelName>.findByIdAndRemove(<Id>, function(err){  
  //Handle any errors or log success.  
});
```

- در کد زیر، به هر checkbox ، مربوط به آیتم اختصاص می یابد تا بتوان موقع Post آن آیتم را شناسایی کرده و حذف کنیم.  
- از checkbox به عنوان onChange استفاده شده است.

```

<% newListItems.forEach(function(item){ %>
  <form action="/delete" method="post">
    <div class="item">
      <input type="checkbox" name="checkbox" value="<%= item._id %>" onchange="this.form.submit()">
      <p><%= item.name %></p>
    </div>
  </form>
<% }); %>

```

```

app.post("/delete", function(req, res){
  const checkedItemId = req.body.checkbox;
  console.log(checkedItemId);

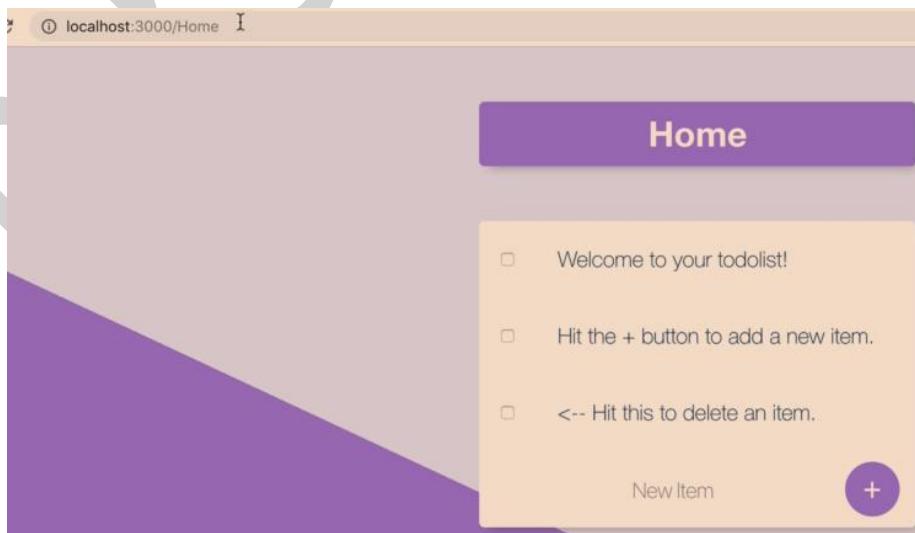
  Item.findByIdAndRemove(checkedItemId, function(err){
    if(!err){
      console.log("Successfully deleted checked item.");
      res.redirect("/");
    }
  });
});

```

**Key words:** Deleting Items from our ToDoList Database.

## 5. Creating Custom Lists using Express Route Parameters

- در این قسمت نحوه ایجاد لیست جدید، در route جدید را نشان می دهد.
- برای فهم بیشتر کد اصلی پروژه را مطالعه نمایید.



- توسط دستور زیر می توان collection دلخواه را حذف نمود.

```
> show collections
items
lists
> db.lists.drop()
true
> show collections
items
```

## Express Route Parameters

```
app.get("/category/:<paramName>", function(req, res){
    //Access req.params.paramName
});
```

## Mongoose findOne()

```
<ModelName>.findOne({conditions}, function(err, results){
    //Use the found results docs.
});
```

- Collection جدیدی برای لیست به اسم List ایجاد می شود.

```
const listSchema = mongoose.Schema({
  name: String,
  items: [itemsSchema]
});

const List = mongoose.model("List", listSchema);
```

- در ادامه در صورت موجود نبودن customListName یک record جدید ایجاد می کند و در صورت موجود بودن آن را نمایش می دهد.

```

app.get("/:customListName", function(req, res) {
  const customListName = req.params.customListName;

  List.findOne({
    name: customListName
  }, function(err, foundList) {
    if (!err) {
      if (!foundList) {
        //Create a new list
        const list = new List({
          name: customListName,
          items: defaultItems
        });
        list.save();
        res.redirect="/" + customListName); // it will rerun this get request.
      } else {
        // show an existing list
        res.render("list", {
          listTitle: foundList.name,
          newListItems: foundList.items
        });
      }
    }
  });
});
});

```

**Key words:** Creating Custom Lists using Express Route Parameters.

## 6. Adding New Items to the Custom ToDo Lists

- در این قسمت برای post request ارسالی، از طریق مقدار دادن به value برای button، تمایز ایجاد کردیم.

```

<form class="item" action="/" method="post">
  <input type="text" name="newItem" placeholder="New Item" autocomplete="off">
  <button type="submit" name="list" value=<%= listTitle %>></button>
</form>

```

- در ادامه با گذاشتن شرط، آیتم جدید در لیست مربوطه ذخیره، و در route مربوطه نمایش داده می‌شود.

```

if (listName === "Today"){
  item.save();
  res.redirect("/");
} else {
  List.findOne({name: listName}, function(err, foundList){
    foundList.items.push(item);
    foundList.save();
    res.redirect("/" + listName);
  });
}

```

**Key words:** Adding New Items to the Custom ToDo Lists.

## 7. Revisiting Lodash and Deleting Items from Custom ToDo Lists

با استفاده از `hidden` می توان، `value` های بیشتری در یک `post` request ارسال کرد.

**<input type="hidden">**

```

<% newListItems.forEach(function(item){ %>
<form action="/delete" method="post">
  <div class="item">
    <input type="checkbox" name="checkbox" value="<%= item._id %>" onchange="this.form.submit()">
    <p><%= item.name %></p>
  </div>
  <input type="hidden" name="listName" value="<%= listTitle %>"></input>
</form>

```

برای حذف یک عضو آرایه از پایگاه داده روش های مختلفی وجود دارد، اما بهینه ترین روش ترکیب دستورات mongoose(`findOneAndUpdate`) و mongo shell (`$pull`) می باشد.

Mongoose delete array element in document and save

## \$pull

The **\$pull** operator removes from an existing array all instances of a value or values that match a specified condition.

The **\$pull** operator has the form:

```
{ $pull: { <field1>: <value|condition>, <field2>: <value|condition>, ... } }
```

copy

## Mongoose findOneAndUpdate()

```
<ModelName>.findOneAndUpdate(  
  {conditions},  
  {$pull: {field: {query}}},  
  function(err, results){}  
);
```

- در نهایت با برقراری شروط زیر، حذف آیتم از لیست دلخواه صورت می گیرد.

```

app.post("/delete", function(req, res) {
  const checkedItemId = req.body.checkbox;
  const listName = req.body.listName;

  if (listName === "Today") {
    Item.findByIdAndRemove(checkedItemId, function(err) {
      if (!err) {
        res.redirect("/");
      }
    });
  } else {
    List.findOneAndUpdate({
      name: listName
    }, {
      $pull: {
        items: {
          _id: checkedItemId
        }
      }
    }, function(err, foundList) {
      if (!err) {
        res.redirect("/" + listName);
      }
    });
  }
});

```

در کد زیر با استفاده از lodash حروف اول تمام کلمات را، بزرگ می کنیم تا در ذخیره سازی و rout مشکلی نداشته باشد.

```

_.capitalize('FRED');
// => 'Fred'

```

```

app.get("/:customListName", function(req, res) {
  const customListName = _.capitalize(req.params.customListName);

```

**Key words:** Deleting Items from Custom ToDo Lists, Revisiting Lodash, `_capitalize()`.

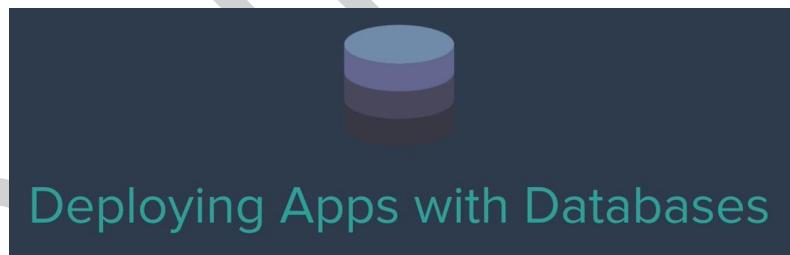
## 8. Tip from Angela - One Step at a Time

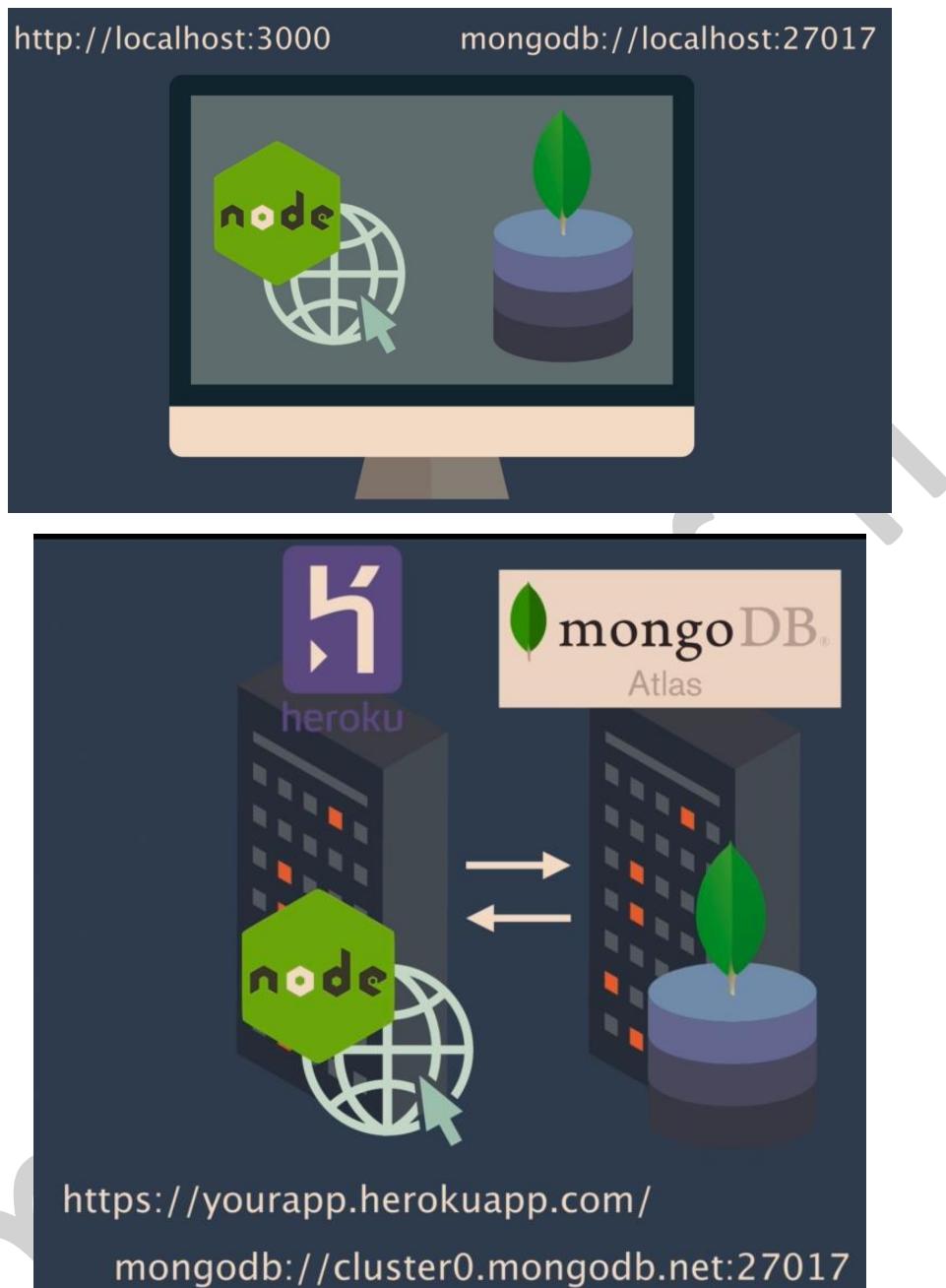
وقتی فکر می کنید که با مقصد و هدف نهایی خیلی فاصله دارید، دلسوز می شوید. در این زمان بهترین حالت این است که به هدف فکر نکنید و فقط به قدم های کوچکی که پیش رویتان است توجه کنید و قدم به قدم پیش بروید. بعد از گذشت مدتی متوجه خواهید شد که به هدف نزدیک شده اید و این راز رسیدن به اکثر هدف هاست. در واقع تغییرات بزرگ در اثر استمرار عادت های کوچک روزانه ایجاد می شود. بنابراین هر روز قدمی رو به جلو بردارید و در بلند مدت اثر آن را خواهید دید.

# 29. Deploying Your Web Application

## 1. How to Deploy Web Apps with a Database

- در این قسمت به ضرورت وجود سرور برای سایت، و دیتابیس توضیح داده می شود.
- از `localhost` برای تست و آزمایش استفاده می کنیم.
- برای پیاده سازی باید از سرورها استفاده کنیم. برای پیاده سازی `node` از سرورهای Heroku و برای پیاده سازی database از MongoDB Atlas استفاده می کنیم.

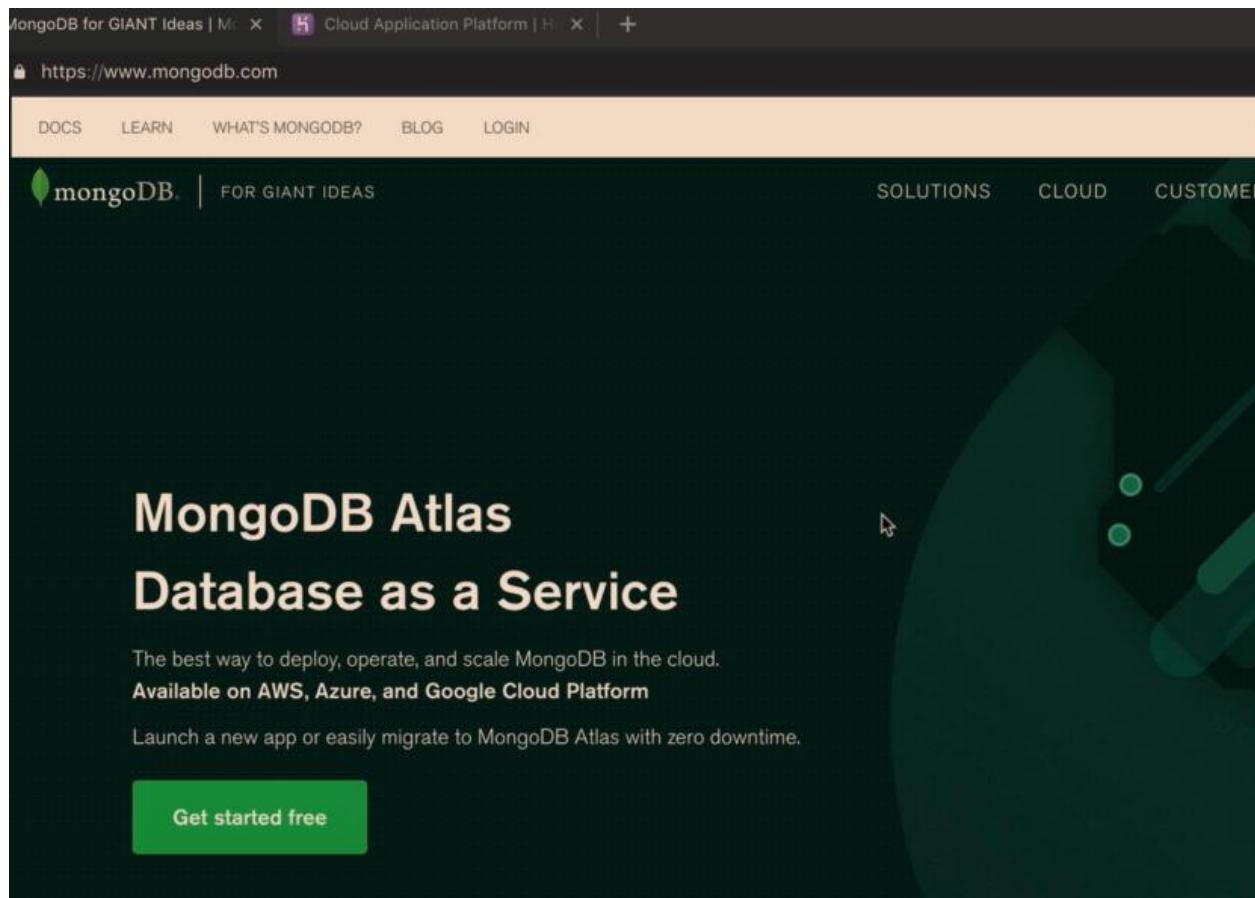




**Key words:** How to Deploy Web Apps with a Database.

## 2. How to Setup MongoDB Atlas

در این قسمت نحوه راه اندازی و پیکربندی سیستم ابری MongoDB Atlas را نشان می دهد، تا بتوان از این Database در بستر اینترنت استفاده نمود.



The screenshot shows the MongoDB Atlas landing page. At the top, there's a navigation bar with links for DOCS, LEARN, WHAT'S MONGODB?, BLOG, and LOGIN. Below the navigation is the MongoDB logo and the tagline "FOR GIANT IDEAS". To the right are links for SOLUTIONS, CLOUD, and CUSTOMER. The main heading "MongoDB Atlas" is prominently displayed, followed by "Database as a Service". A subtext below it reads: "The best way to deploy, operate, and scale MongoDB in the cloud. Available on AWS, Azure, and Google Cloud Platform. Launch a new app or easily migrate to MongoDB Atlas with zero downtime." A green button labeled "Get started free" is visible.

### Cloud Provider & Region



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster size.

Cluster Tier					M0 (Shared RAM, 512 MB Storage) 
					Encrypted
Base hourly rate is for a MongoDB replica set with 3 data bearing servers.					
Shared Clusters 					
 M0	Shared RAM	512 MB Storage	Shared vCPUs	FREE	
 M2	Shared RAM	2 GB Storage	Shared vCPUs	from \$0.012/hr <b>ONLY \$9 / MONTH</b>	
 M5	Shared RAM	5 GB Storage	Shared vCPUs	from \$0.035/hr <b>ONLY \$25 / MONTH</b>	

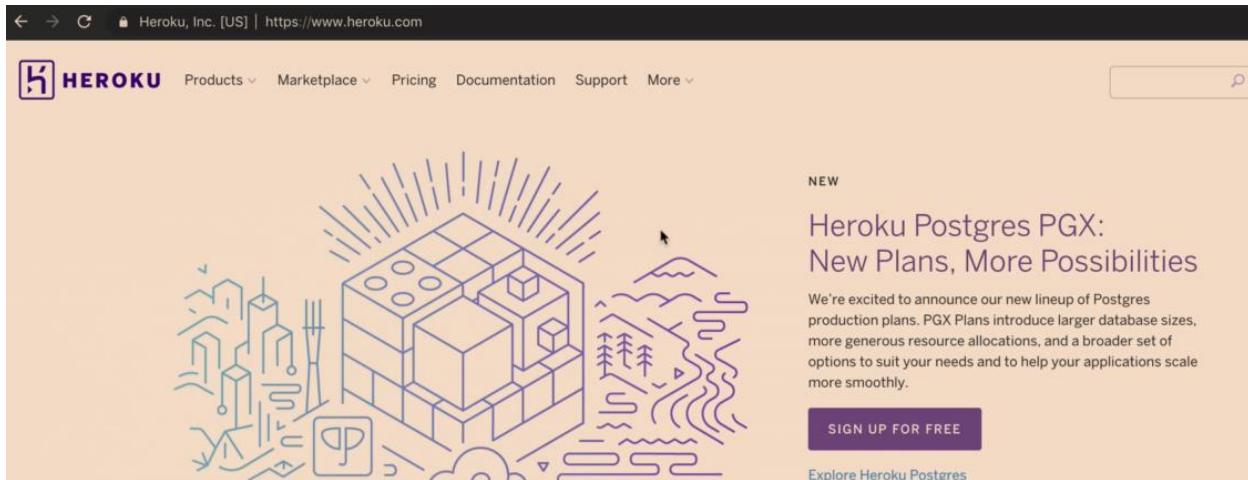


The screenshot shows the MongoDB Atlas interface. On the left, there's a sidebar with project navigation and various management links like Stitch Apps, Alerts, Backup, etc. The main area is titled 'Clusters' and shows the 'Security' tab selected under the 'MongoDB Users' section. A progress bar at the top indicates a deployment is in progress: 'We are deploying your changes: 0 of 3 servers complete (current action)'. Below the progress bar, it shows 'ANGELA'S ORG - 2018-10-22 > PROJECT 0'.

**Key words:** How to Setup MongoDB Atlas.

### 3. Deploying an App with a Database to Heroku

- در این قسمت، نحوه استفاده از سرورهای Heroku برای راه اندازی wep application و اتصال آن به MongoDB Atlas را نشان می دهد.



Heroku, Inc. [US] | https://devcenter.heroku.com/articles/getting-started-with-nodejs#prepare-the-app

## Getting Started on Heroku with Node.js

**introduction**

**Setup**

**Prepare the app** (selected)

Deploy the app

View logs

Define a Procfile

Scale the app

Declare app dependencies

Run the app locally

Push local changes

Provision add-ons

Start a console

Define config vars

Provision a database

Next steps

### Prepare the app

In this step, you will prepare a sample application that's ready to be deployed to Heroku.

**Info** If you are new to Heroku, it is recommended that you complete this tutorial using the Heroku-provided sample application.

However, if you have your own [existing application](#) that you want to deploy instead, see [this article](#) to learn how to prepare it for Heroku deployment.

To clone a local version of the sample application that you can then deploy to Heroku, execute the following commands in your local command shell or terminal:

```
$ git clone https://github.com/heroku/node-js-getting-started.git  
$ cd node-js-getting-started
```

You now have a functioning Git repository that contains a simple application as well as a `package.json` file, which is used by Node's dependency manager.

[Report a problem](#) [I cloned the app source code](#)

**Key words:** Deploying an App with a Database to Heroku.

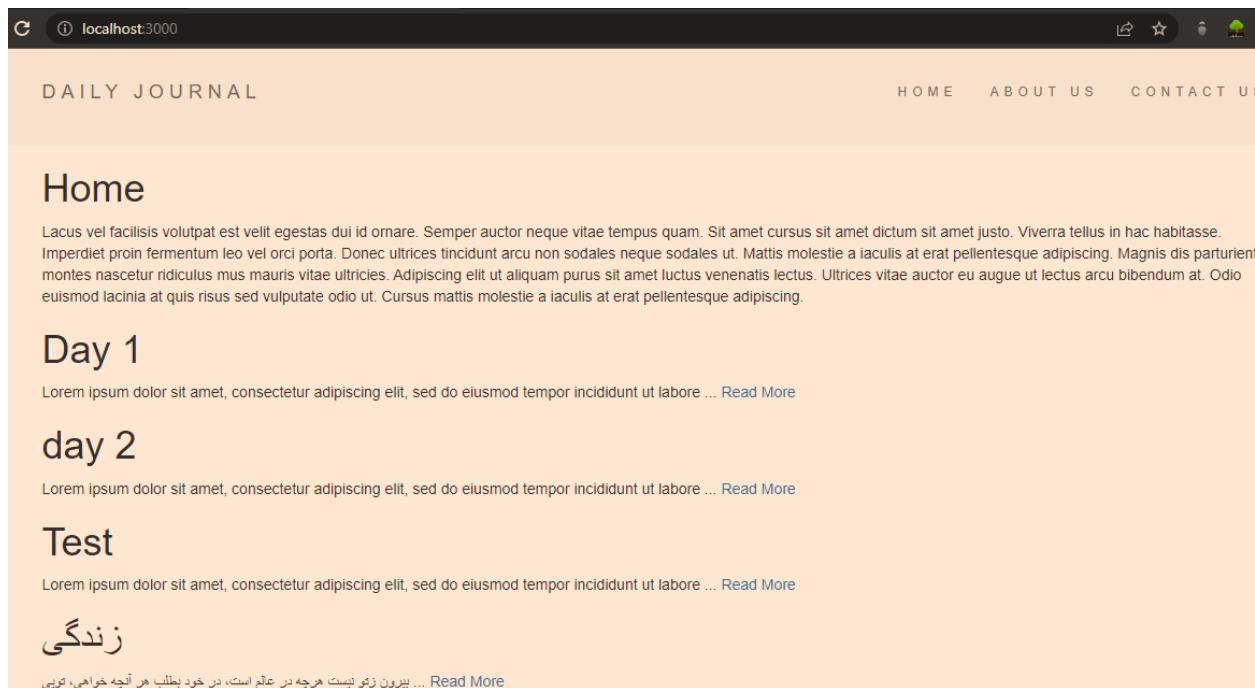
#### 4. Tip from Angela - Discipline Breeds Discipline

- بعضی روزها انگیزه انجام وظایف در راستای اهدافتان را ندارید. در این موقع چرایی و علت اهدافتان را مرور کنید تا انگیزه تان زنده شود و شما را به عمل وادار کند. یادتان باشد، نظم باعث پیروزی نظم می‌شود، یعنی اگر در راستای اهدافتان، هر روز به طور منظم قدم بردارید، روزهای بعد نیز این کار را خواهید کرد و رفته رفته برایتان عادت شده و به راحتی آن وظایف را انجام می‌دهید.

## 30. Boss Level Challenge 4 - Blog Website Upgrade

### 1. Challenge Give your Blog a Database

- در این فصل هدف ایجاد Database برای پروژه Blog Website می باشد.



- برای بررسی بیشتر پروژه به فایل اصلی تکمیل شده پروژه مراجعه کنید.

- کدهای زیر در بروزرسانی پروژه ، جهت ایجاد Database اعمال شده است.

```
const postSchema = {  
    title: String,  
    content: String  
};  
  
const Post = mongoose.model("Post", postSchema);
```

```
app.get("/", function(req, res){  
  Post.find({}, function(err, posts){  
    res.render("home", {  
      startingContent: homeStartingContent,  
      posts: posts  
    });  
  });  
});
```

```
app.post("/compose", function(req, res){  
  const post = new Post({  
    title: req.body.postTitle,  
    content: req.body.postBody  
  });  
  
  post.save(function(err){  
    if (!err){  
      res.redirect("/");  
    }  
  });  
});
```

```
app.get("/posts/:postId", function(req, res){  
  
  const requestedPostId = req.params.postId;  
  
  Post.findOne({_id: requestedPostId}, function(err, post){  
    res.render("post", {  
      title: post.title,  
      content: post.content  
    });  
  });  
});
```

## 8. Tip from Angela - Dealing with Limitations

یکی از مشکلاتی که ذهن میمونی ما دارد، این است که مدام از همه چی ایراد بگیرد و همیشه دنبال بهانه تراشی و غر زدن باشد. حتی وقتی واقعاً همه چیز خوب است، باز از شرایط ناراضی است. یادتان باشد، مهم نیست چند سالtan است، زن یا مرد هستید، کجا زندگی می کنید و... این ها جزوی از محدودیت های شما هستند، ولی محدودیت واقعی در ذهن شما شکل گرفته است. به جای غر زدن، شما می توانید بیشتر تلاش کنید، زودتر از خواب بیدار شوید، بیشتر کتاب بخوانید، بیشتر یاد بگیرید و در نتیجه ذهنی قوی پرورش دهید که محدودیتی ندارد. یادتان باشد هرچه با چالش های بزرگتری برخورد کنید و آن را حل کنید، حل چالش های کوچکتر راحتتر خواهد بود.

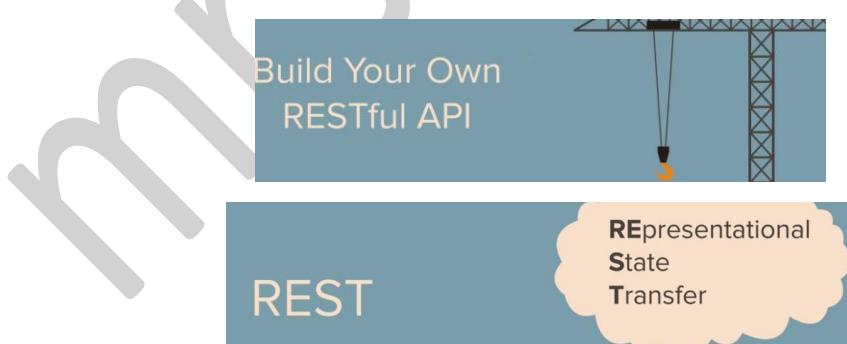
## 31. Build Your Own RESTful API From Scratch

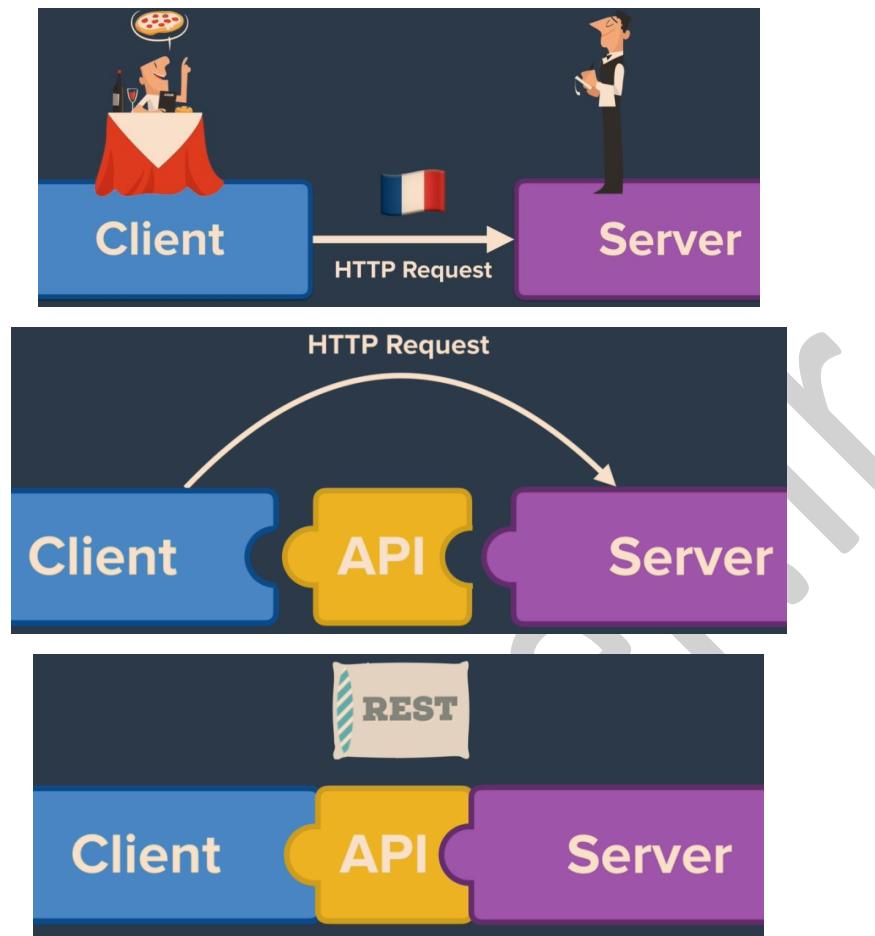
### 1. What is REST

- زبان های مختلی در شبکه، مانند HTTP, FTP, HTTPS و... موجود می باشد، ولی برای وب از استفاده می شود.

- یکسری قوانین استاندار برای طراحی API می باشد که توسط HTTP Request Verbs می توان با زبان قوانین REST ، تعامل برقرار کرد. مثلاً Http request client با REST متناسب با API Server می تواند درخواست مورد نظر خود را از API Server بخواهد.

- در واقع با RESTful API ، هر کاربری می تواند با Server تعامل کند، و اطلاعات مورد نظرش در Database را اصلاح کند . مثلاً در توییتر توئیت می کنیم، یا در تلگرام پیام می دهیم ویا پیام را پاک می کنیم و کلی کار دیگه.





- علاوه بر REST استانداردهای مختلفی وجود دارد، ولی REST بهترین استاندارد برای WEB می باشد.  
در واقع استاندارد سازی باعث راحتی و سرعت برای کل کاربران جهان می شود.

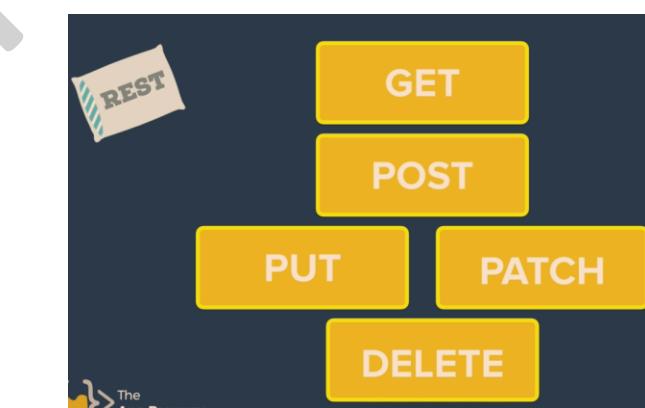
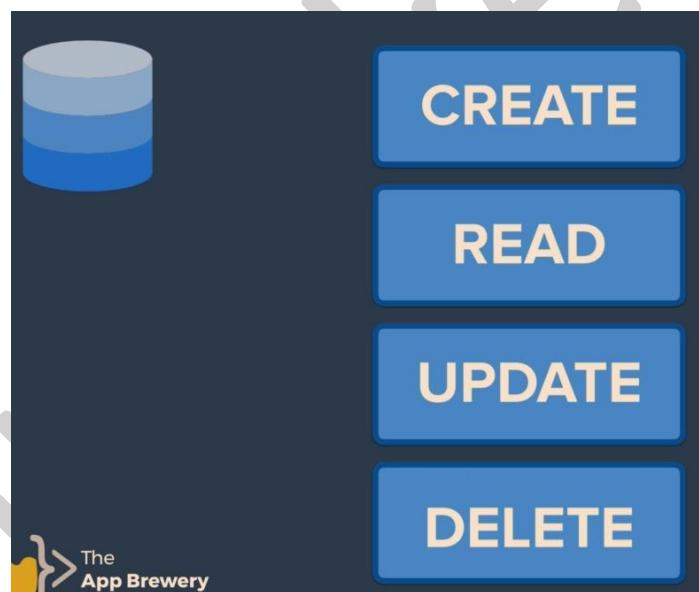


# Use HTTP Request Verbs

Use Specific Pattern of Routes/Endpoint URLs



HTTP Request Verbs - را می توان با CRUD معادل سازی کرد، تا به کاربرد هر کدام پی برد.



- معادل تعویض کل دیتا، ولی Patch معادل تغییر بخشی از دیتا می باشد.

GET	READ	POST	CREATE
app.get(function(req, res){	app.post(function(req, res){		
PUT	PATCH	UPDATE	DELETE
app.patch(function(req, res){		app.delete(function(req, res){	

- شکل زیر قوانین HTTP Request Verbs را توسط RESTful Routes نشان می دهد.

## RESTful Routing

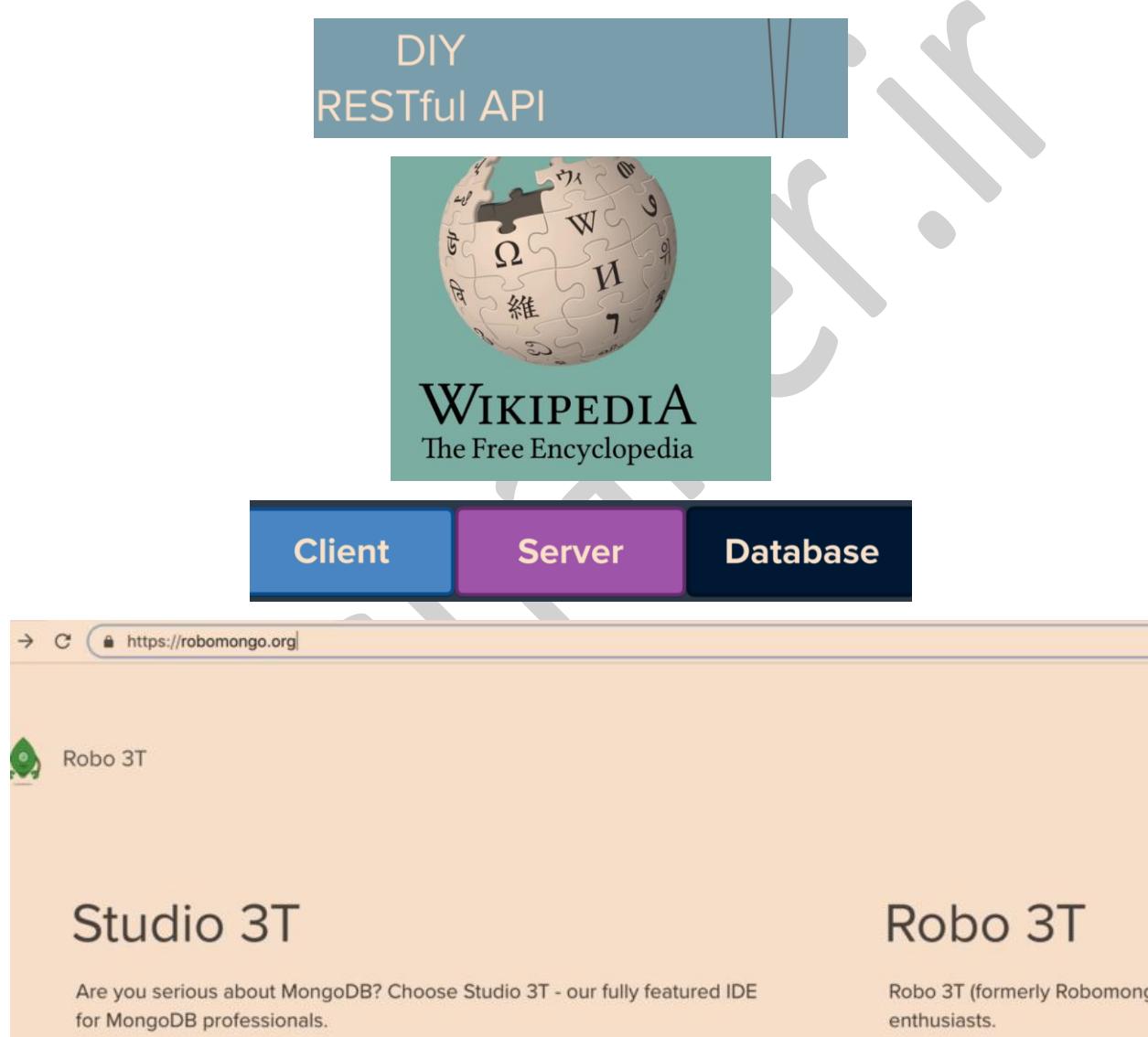
HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

**Key words:** what is REST, different language requests, http request verbs, RESTful standard rules.

## 2. Creating a Database with Robo 3T

- در این قسمت، قصد ایجاد یک پروژه با Database همانند wikipedia داریم، که چیزهایی را که یاد گرفته اید در این دوره، به صورت document وارد mongoDB کنید و مقدمه ایجاد RESTful API شود.

- برای وارد کردن و بررسی اطلاعات به جای استفاده از mongoDB shell ، از نرم افزار گرافیکی Robo 3T استفاده می کنیم.



The screenshot shows the Robo 3T interface. On the left, the sidebar lists connections (New Connection (5), System, blogDB, config, wikiDB) and collections (Collections (1), articles, Indexes, Functions, Users). The main area shows a query: db.getCollection('articles').find(). The results table has columns Key and Value. There are four documents:

Key	Value
▶ (1) ObjectId("5c1e1892998bdb3b3d355bf")	{ 3 fields }
▶ (2) ObjectId("5c139771d79ac8eac11e754a")	{ 3 fields }
▶ (3) ObjectId("5c1398aad79ac8eac11e7561")	{ 3 fields }
▶ (4) ObjectId("5c1398ecd79ac8eac11e7567")	{ 3 fields }

```
{
    "_id" : ObjectId("5c139771d79ac8eac11e754a"),
    "title" : "API",
    "content" : "API stands for Application Programming Interface. It is a set of subroutine definitions, communication protocols, and tools for building software. In general terms, it is a set of clearly defined methods of communication among various components. A good API makes it easier to develop a computer program by providing all the building blocks, which are then put together by the programmer."
}

{
    "_id" : ObjectId("5c1398aad79ac8eac11e7561"),
    "title" : "Bootstrap",
    "content" : "This is a framework developed by Twitter that contains pre-made front-end templates for web design"
}

{
    "_id" : ObjectId("5c1398ecd79ac8eac11e7567"),
    "title" : "DOM",
    "content" : "The Document Object Model is like an API for interacting with our HTML"
}
```

**Key words:** create our Wikipedia Database, using Robo 3T GUI for our mongoDB.

### 3. Set Up Server Challenge

- در این قسمت چالش ایجاد سرور، با توجه به لیست زیر خواسته می شود.

# CodeWiki-API

- Create new Directory called Wiki-API
- Initialise NPM and install body-parser, mongoose, ejs and express
- Create a new file called app.js
- Inside app.js add server code (Write/Copy)
- Setup MongoDB:
  - DB name is wikiDB
  - Collection name is articles
  - Document has 2 fields: title and content

**Key words:** Set Up Server Challenge.

## 4. Set Up Server Solution

- در این قسمت، راه حل چالش مورد نظر ارائه می شود.

```
3  const express = require("express");
4  const bodyParser = require("body-parser");
5  const ejs = require("ejs");
6  const mongoose = require('mongoose');
7
8  const app = express();
9
10 app.set('view engine', 'ejs');
11
12 app.use(bodyParser.urlencoded({
13   extended: true
14 }));
15 app.use(express.static("public"));
16
17
18 mongoose.connect("mongodb://localhost:27017/wikiDB", {
19   useNewUrlParser: true
20 });
21
22
23 const articleSchema = {
24   title: String,
25   content: String
26 };
27
28 const Article = mongoose.model("Article", articleSchema);
29
```

## 5. GET All Articles

در این قسمت خواندن همه اطلاعات از RESTful API با استاندارد Database را ایجاد می کنیم.

# RESTful

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer



```
app.get(route, function(req, res){  
});
```

READ

A graphic element featuring a grey hand icon pointing towards a blue cylinder representing a database. To the right is a blue rounded rectangle containing the word "READ" in white capital letters.

```
<ModelName>.find({conditions}, function(err, results){  
    //Use the found results docs.  
});
```

```

app.get("/articles", function(req, res){
  Article.find({}, function(err, foundArticles){
    if(!err){
      res.send(foundArticles);
    } else {
      res.send(err);
    }
  });
});

```

**Key words:** GET All Articles from Database with RESTful API standard.

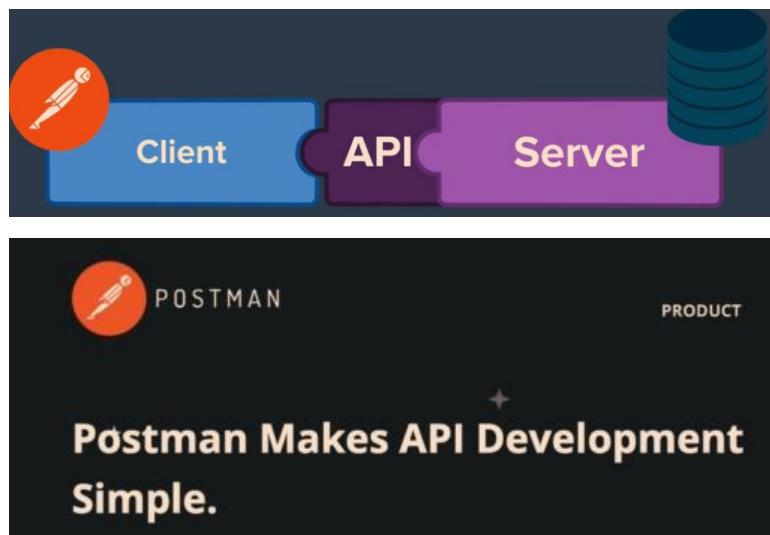
## 6. POST a New Article

در این قسمت نحوه ثبت اطلاعات در RESTful API توسط Database را نشان می دهد که برای این منظور از Post Request استفاده می شود.

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

برای ارسال Post Request از سمت client، علاوه بر دستورات input HTML، می توان در زمانی که UI نداریم، از روش سریع تر یعنی از نرم افزار Postman استفاده کنیم.





- در شکل زیر نحوه post request و تعریف مقادیر آن در Postman را نشان می دهد.

localhost:3000/articles

POST localhost:3000/articles

Params	Authorization	Headers (1)	Body (1)	Pre-request Script	Tests									
<input type="radio"/> none	<input type="radio"/> form-data	<input checked="" type="radio"/> x-www-form-urlencoded	<input type="radio"/> raw	<input type="radio"/> binary										
			<table border="1"> <thead> <tr> <th>KEY</th> <th>VALUE</th> <th>DES</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> title</td> <td>Jack Bauer</td> <td></td> </tr> <tr> <td><input checked="" type="checkbox"/> content</td> <td>Jack Bauer once stepped into quicksand. The quicksand cou...</td> <td></td> </tr> </tbody> </table>	KEY	VALUE	DES	<input checked="" type="checkbox"/> title	Jack Bauer		<input checked="" type="checkbox"/> content	Jack Bauer once stepped into quicksand. The quicksand cou...			
KEY	VALUE	DES												
<input checked="" type="checkbox"/> title	Jack Bauer													
<input checked="" type="checkbox"/> content	Jack Bauer once stepped into quicksand. The quicksand cou...													

- در سمت سرور، اطلاعات دریافتی از post request را در Database ذخیره می شود.





## CREATE

```
const <constantName> = new <ModelName> ({

    <fieldName> : <fieldData>,

    ...

});

<constantName>.save();
```



New Connection localhost:27017 wikiDB

db.getCollection('articles').find({})

articles 0.01 sec.

Key	Value
▶ (1) ObjectId("5c18e1892998bdb3b3d355bf")	{ 3 fields }
▶ (2) ObjectId("5c139771d79ac8eac11e754a")	{ 3 fields }
▶ (3) ObjectId("5c1398aad79ac8eac11e7561")	{ 3 fields }
▶ (4) ObjectId("5c1398ecd79ac8eac11e7567")	{ 3 fields }
▶ (5) ObjectId("5c18f35cde40ab6cc551cd60")	{ 4 fields }
└ _id	ObjectId("5c18f35cde40ab6cc551cd60")
└ title	Jack Bauer
└ content	Jack Bauer once stepped into quicksand. The quicksand cou..
└ __v	0

```
app.post("/articles", function(req, res){
  const newArticle = new Article({
    title: req.body.title,
    content: req.body.content
  });

  newArticle.save(function(err){
    if (!err){
      res.send("Successfully added a new article.");
    } else {
      res.send(err);
    }
  });
});
```

**Key words:** POST a New Article with RESTful API, use Postman for post request.

## 7. DELTE All Articles

- در این قسمت نیز با استاندارد API RESTful ، اطلاعات را از Database پاک می کنیم. کل collection پاک می شود.

# RESTful

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer



**DELETE**

```
app.delete(route, function(req, res){  
})
```



**DELETE**

```
<ModelName>.deleteMany(  
  {conditions},  
  function(err){  
  }  
) ;
```

The screenshot shows the Postman interface with a successful DELETE request to `localhost:3000/articles`. The response status is 200 OK, time is 24 ms, and size is 239 B. The response body is "Successfully deleted all articles." Below the screenshot is a snippet of Node.js code demonstrating how to implement this logic using Express.js:

```

app.delete("/articles", function(req, res){
  Article.deleteMany(function(err){
    if (!err){
      res.send("Successfully deleted all articles.");
    } else {
      res.send(err);
    }
  });
});|
```

**Key words:** DELTE All Articles with RESTful API.

## 8. Chained Route Handlers Using Express

- با دستور `app.route()` در `express` می توان `Http Request` ها را به صورت یک زنجیر و پشت هم نوشت. در نتیجه باعث خوانایی و کارآمدی و راحتی کد نوشته شده می شود.

## app.route()

You can create chainable route handlers for a route path by using `app.route()`. Because the path is specified at a single location, creating modular routes is helpful, as is reducing redundancy and typos. For more information about routes, see: [Router documentation](#).

Here is an example of chained route handlers that are defined by using `app.route()`.

```
app.route('/book')
  .get((req, res) => {
    res.send('Get a random book')
  })
  .post((req, res) => {
    res.send('Add a book')
  })
  .put((req, res) => {
    res.send('Update the book')
  })
```

**Key words:** Chained Route Handlers Using Express, `app.route()`.

## 9. GET a Specific Article

- در این قسمت نیز با استاندارد RESTful API ، اطلاعات با عنوان خاص را از Database می خوانیم.

The diagram illustrates a RESTful API endpoint for articles. At the top, the word "RESTful" is written in large blue letters. Below it is a table with three columns: "HTTP Verbs", "/articles", and "/articles/jack-bauer". The table rows correspond to the five HTTP methods: GET, POST, PUT, PATCH, and DELETE. The "GET" row under "/articles" is highlighted in green and describes fetching all articles. The "GET" row under "/articles/jack-bauer" is also highlighted in green and describes fetching the article for author "jack-bauer". The "POST" row under "/articles" is highlighted in green and describes creating one new article. The "PUT" row under "/articles" is highlighted in orange and describes updating the article for author "jack-bauer". The "PATCH" row under "/articles" is highlighted in orange and describes updating the article for author "jack-bauer". The "DELETE" row under "/articles" is highlighted in green and describes deleting all articles. Below the table, there is a dark blue box containing a yellow "GET" button and some code. To the left of the box, there is a small icon of a document with the word "REST" on it.

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

```
app.route("route")
  .get(function(req, res){
    })
  
```

## READ

```
<ModelName>.findOne(  
    {conditions},  
    function(err, result){  
        //Use the found result.  
    }  
);
```

### Route parameters

Route parameters are named URL segments that are used to capture the values specified at their position in the URL. The captured values are populated in the `req.params` object, with the name of the route parameter specified in the path as their respective keys.

```
Route path: /users/:userId/books/:bookId  
Request URL: http://localhost:3000/users/34/books/8989  
req.params: { "userId": "34", "bookId": "8989" }
```

To define routes with route parameters, simply specify the route parameters in the path of the route as shown below.

```
app.get('/users/:userId/books/:bookId', function (req, res) {  
  res.send(req.params)  
})
```

```
app.route("/articles/:articleTitle")  
  .get(function(req, res){  
  
    const articleTitle = req.params.articleTitle;  
    console.log(articleTitle);  
    Article.findOne({title: articleTitle}, function(err, foundArticle){  
  
      if (foundArticle){  
        res.send(foundArticle);  
      } else {  
        res.send("Not fonund Article.")  
      }  
    });  
  });
```

GET    localhost:3000/articles/REST

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies

در URL Encoding برای کاراکترها از UTF-8 استفاده می شود. مثلاً به جای ارسال از ۲۰ space در URL استفاده می شود. (این کار را معمولاً مرورگر به صورت پیش فرض انجام می دهد)

## HTML URL Encoding Reference

URL encoding converts characters into a format that can be transmitted over the Internet.

## ASCII Encoding Reference

Your browser will encode input, according to the character-set used in your page.

The default character-set in HTML5 is UTF-8.

Character	From Windows-1252	From UTF-8
space	%20	%20
!	%21	%21
"	%22	%22
#	%23	%23
\$	%24	%24
%	%25	%25
&	%26	%26
'	%27	%27
/	%2F	%2F

GET    localhost:3000/articles/Jack%20Bauer

Send

**Key words:** GET a Specific Article, Express route params, url encoded for characters.

## 10. PUT a Specific Article

در این قسمت نیز با استاندارد RESTful API ، یک document با عنوان خاص در Database را با document جدید جایگزین می کنیم.

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

PUT

```
app.put(route, function(req, res){  
});
```

UPDATE



```
<ModelName>.update(  
  {conditions},  
  {updates},  
  {overwrite: true}  
  function(err, results){}  
);
```

```

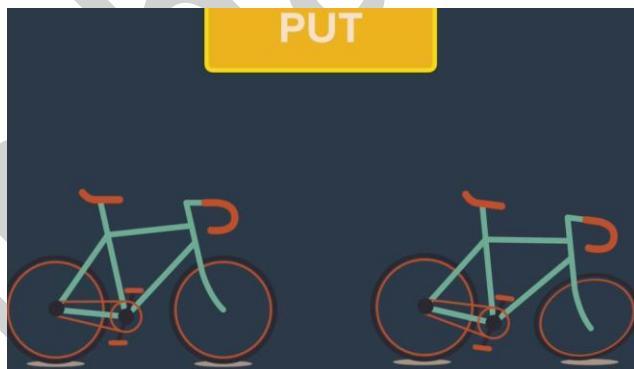
Article.update(
  {title: req.params.articleTitle},
  {title: req.body.title, content: req.body.content},
  {overwrite: true},
  function(err){
    if(!err){
      res.send("Successfully updated the selected article.");
    }
  }
);
})

```

PUT      localhost:3000/articles/Jack%20Bauer      Send

Params	Authorization	Headers (1)	<b>Body</b> ●	Pre-request Script	Tests
<input type="radio"/> none	<input type="radio"/> form-data	<input checked="" type="radio"/> x-www-form-urlencoded	<input type="radio"/> raw	<input type="radio"/> binary	
<input checked="" type="checkbox"/> title			Chuck Norris		
<input checked="" type="checkbox"/> content			The First rule of Chuck Norris is: you do not talk about Chuc...		

- باعث می شود، مجدداً آن document نوشته شود، یعنی اثری از Overwrite - باقی نماند چون وظیفه PUT جایگزینی می باشد.



**Key words:** PUT a Specific Article, overwrite a new document in Database.

## 11. PATCH a Specific Article

- در این قسمت نیز با استاندارد RESTful API ، یک document با عنوان خاص در Database را با مقادیر جدید تعویض می کنیم.

# RESTful

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

**PATCH**

```
app.patch(route, function(req, res){  
})
```

**UPDATE**



```
<ModelName>.update(  
  {conditions},  
  {$set: updates},  
  function(err, results){}  
);
```

```

.patch(function(req, res){

  Article.update(
    {title: req.params.articleTitle},
    {$set: req.body},
    function(err){
      if(!err){
        res.send("Successfully updated article.");
      } else {
        res.send(err);
      }
    }
  );
})

```

PATCH    localhost:3000/articles/Jack%20Bauer    Send

Params    Authorization    Headers (1)    **Body**    Pre-request Script    Tests

none     form-data     x-www-form-urlencoded     raw     binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> title	Chuck Norris	
Key	Value	Description

**\$set**

The `$set` operator replaces the value of a field with the specified value.

The `$set` operator expression has the following form:

```
{ $set: { <field1>: <value1>, ... } }
```

- در شکل زیر نمونه عناصر موجود در یک body را نشان می دهد.

```

req.body = {
  title: "TEST",
  content: "TEST"
}

```

**Key words:** PATCH a Specific Article, change a document in Database.

## 12. DELETE a Specific Article

در این قسمت نیز با استاندارد RESTful API ، یک document با عنوان خاص در Database را حذف می کنیم.

HTTP Verbs	/articles	/articles/jack-bauer
GET	Fetches <b>all</b> the articles	Fetches <b>the</b> article on jack-bauer
POST	Creates <b>one</b> new article	-
PUT	-	Updates <b>the</b> article on jack-bauer
PATCH	-	Updates <b>the</b> article on jack-bauer
DELETE	Deletes <b>all</b> the articles	Deletes <b>the</b> article on jack-bauer

DELETE

```
app.delete(route, function(req, res){  
})
```

DELETE



```
<modelName>.deleteOne(  
  {conditions},  
  function(err){}  
);
```

```

.delete(function(req, res){

  Article.deleteOne(
    {title: req.params.articleTitle},
    function(err){
      if (!err){
        res.send("Successfully deleted the corresponding article.");
      } else {
        res.send(err);
      }
    }
  );
});

```



**Key words:** DELETE a Specific Article or Document in Database.

#### 14. Tip from Angela - How to Get a Job as Programmer

برای اینکه بتوانید در شرکت های برنامه نویسی استخدام شوید قوانین زیر را رعایت کنید:

- رزومه خوبی برای خود درست کنید.
- به **Linkedin** بروید و مهارت های مورد نیاز کارفرماها را یاد بگیرید و در آن راستا برای خود پروژه انجام دهید تا رزومه شما قوی شود.
- به **stackoverflow** بروید و به دیگران کمک کنید و از دیگران یاد بگیرد. یک برنامه نویس قوی باید بداند چگونه در یک تیم کار کند و با بالادستی و پایین دستی خود چگونه تعامل کند.

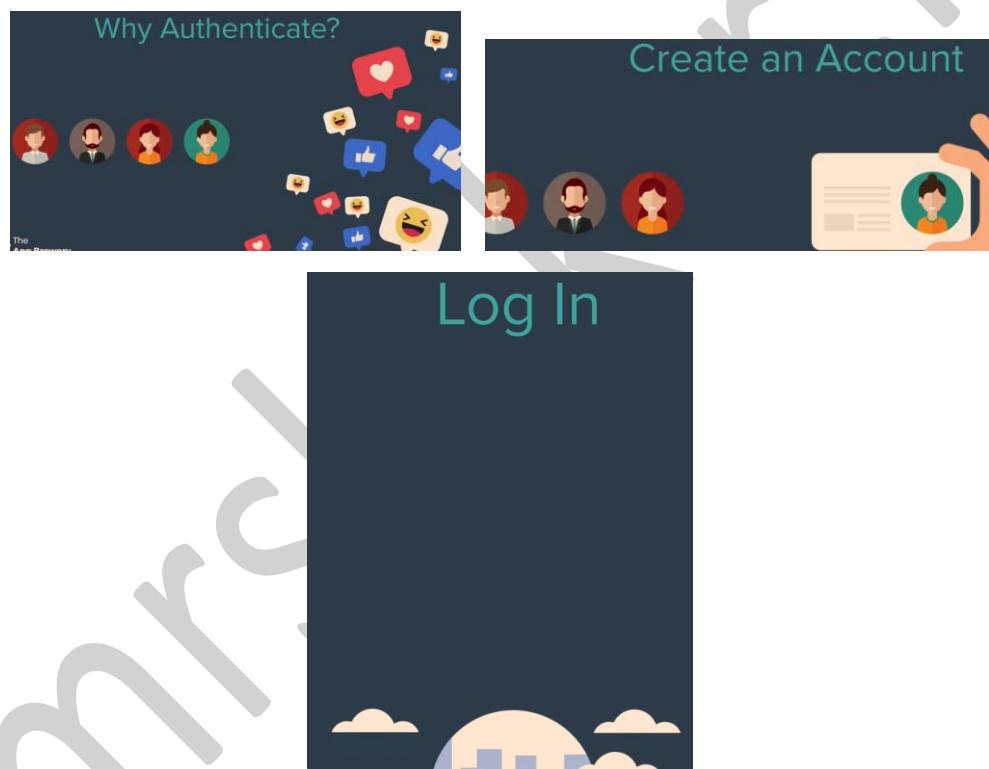
## 32. Authentication & Security

### 1. Introduction to Authentication

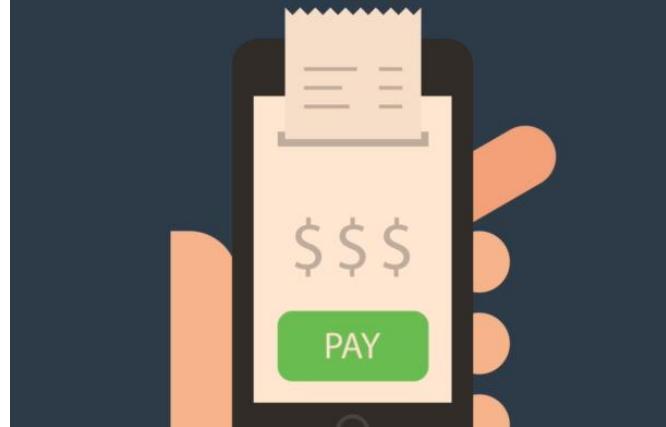
- در این قسمت درباره **Authentication** یا احراز هویت صحبت می شود و مزایای آن را بیان می کند.



- احراز هویت برای هر سایت لازم است، ۱- باعث حفظ حریم شخصی می شود ۲- امکان ایجاد سطح دسترسی برای کاربران را فراهم می کند.



## Restrict Access



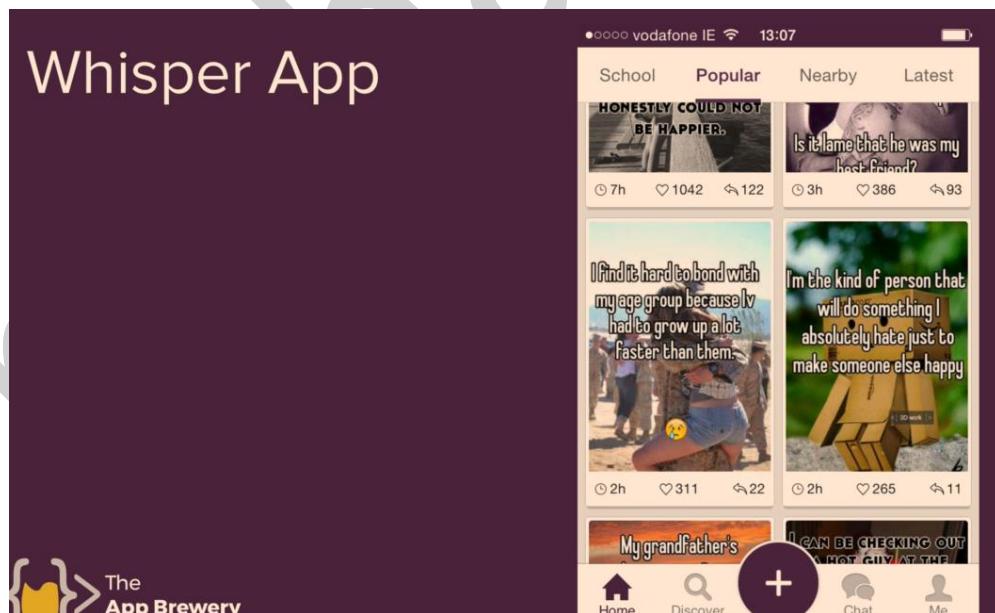
- در نتیجه احراز هویت باعث افزایش امنیت و یکپارچگی سایت می شود.



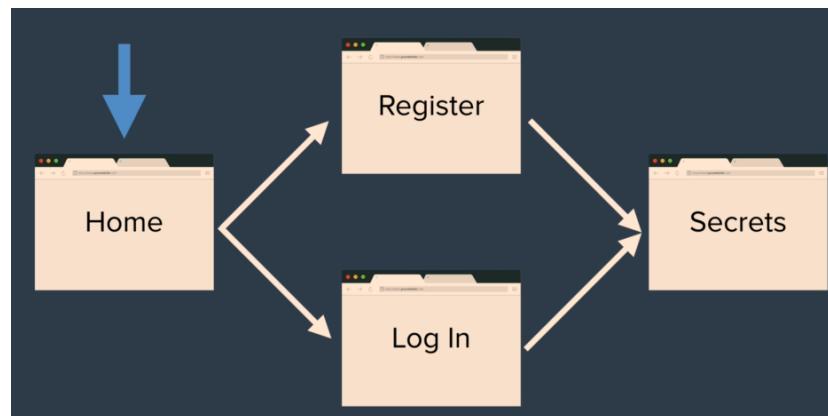
- در این فصل ۶ سطح برای رسیدن به امنیت توسط Authentication به ترتیب از پایین ترین به بالاترین سطح توضیح داده خواهد شد. همچنین درک شما را درباره امنیت سایت بالا خواهیم برد.



برای درک Authentication - که مکانی برای نوشتن متن به صورت ناشناس می باشد ایجاد کنیم.



- در ساختار زیر نقشه سایت یا پروژه ای که قرار است ایجاد شود را نشان می دهد.



- برای دسترسی کامل به هر مرحله پروژه، و مشاهده آن توسط Git می‌توان از لینک زیر به پروژه دسترسی داشت و commit های مختلف را مشاهده نمود.

<https://github.com/londonappbrewery/Authentication-Secrets>

Then you can use `git log` to see all the commits. You should see something like this:

```

commit 98ce8958d78eadfeb37cc7fb95c41cadf86fb4f
Level 6 - Google OAuth 2.0 Authentication

commit 4e8349702f16a5570f9ff9b80f7a3740ddd8b108
Level 5 - Cookies and Sessions

commit d3b3b3a908fc01e72b99616db45e2c28f8975369
Level 4 - Hashing and Salting with bcrypt

commit 17696f8cfe68c8f91082a98e9750d45e9e176bc3
Level 3 - Hashing with md5

commit 92a07aa559eb29e5c9c0f50304e7b5e0674a25d1
Add Environment Vars

commit 1702e1d3f75bfbeb0e43848c8bd921863ea21147
Level 2 - Encryption

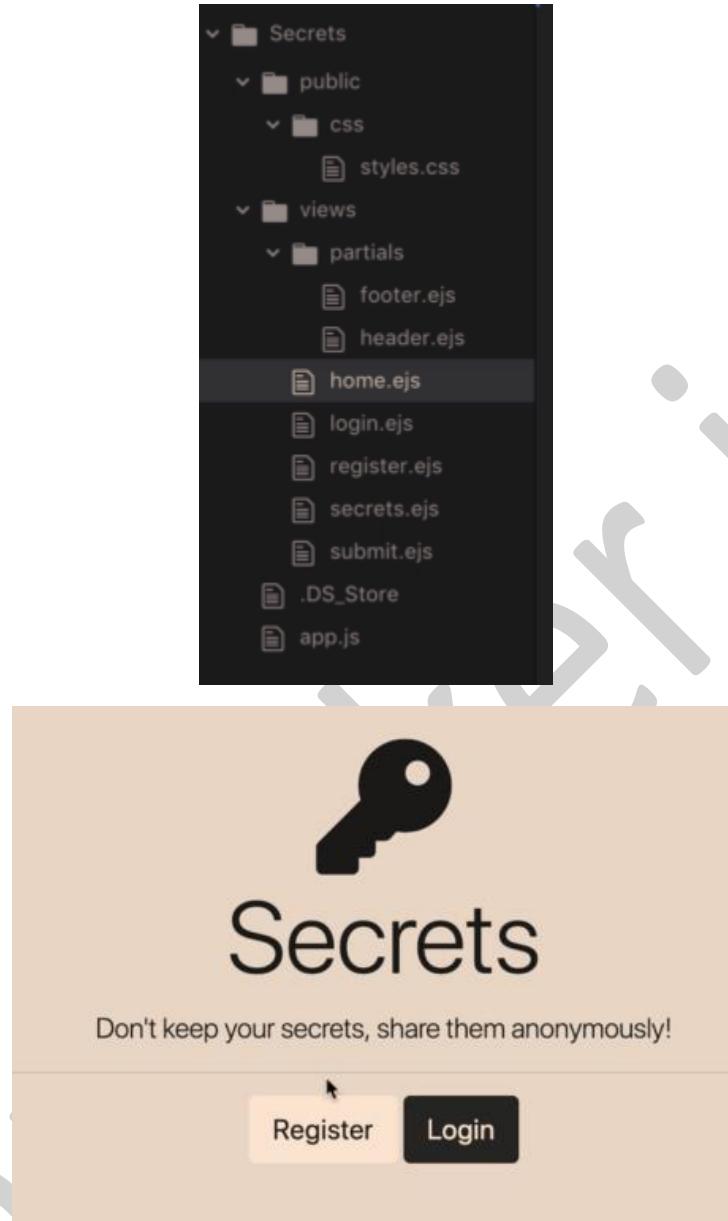
commit 7078af837299a4ff50121d67afe17d9fa522ec68
Level 1 - Username and Password Only

```

**Key words:** Introduction to Authentication, security.

## 2. Getting Set Up

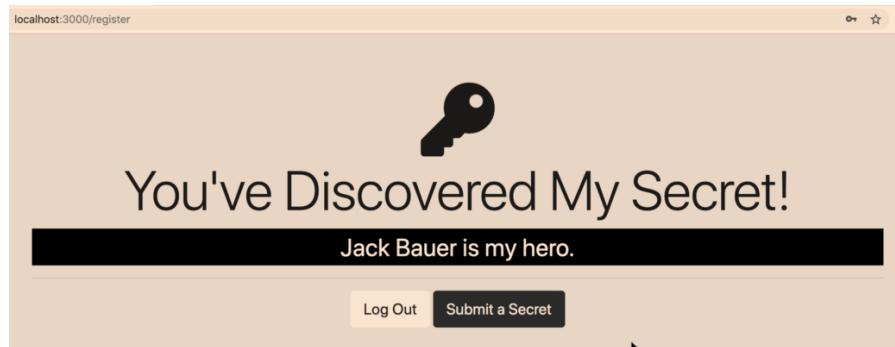
- در این قسمت فایل های پروژه را شرح می‌دهد و پیکربندی اولیه برای پروژه را انجام می‌دهد.



**Key words:** Getting Set Up.

### 3. Level 1 - Register Users with Username and Password

- در این قسمت در ادامه پروژه فصل، سطح اول Authentication، یعنی ثبت نام کاربر صورت می گیرد  
و اطلاعات در دیتابیس ذخیره می شود.  
در ادامه برای login از Database اطلاعات را می خوانیم. -



localhost:3000/register

## Register

Email

Password

**Register**

localhost:3000/login

## Login

Email

Password

**Login**

users    0.008 sec.

Key	Value
✓ (1) ObjectId("624053dbc787a449123f97f4")	{ 4 fields }
_id	ObjectId("624053dbc787a449123f97f4")
email	1@2.com
password	123
__v	0

```
app.post("/register", function(req, res) {
  const newUser = new User({
    email: req.body.username,
    password: req.body.password
  });

  newUser.save(function(err) {
    if (err) {
      console.log(err);
    } else {
      res.render("secrets");
    }
  });
});
```

```
app.post("/login", function(req, res) {
  const username = req.body.username;
  const password = req.body.password;

  User.findOne({
    username: username
  }, function(err, foundUser) {
    if (err) {
      console.log(err);
    } else {
      if (foundUser) {
        if (password === foundUser.password) {
          res.render("secrets");
        }
      }
    }
  });
});
```

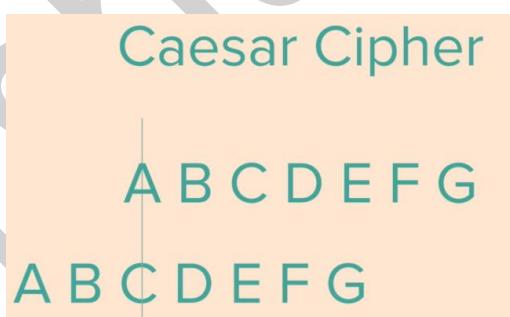
**Key words:** Level 1, Register Users with Username and Password.

## 5. Level 2 - Database Encryption

- در سطح بعدی امنیت، رمزگاری اطلاعات مورد نظر مثل پسوردها در پایگاه داده می باشد. با این کار اول اینکه در صورت هک شدن پایگاه داده، اطلاعات مهم برای هکر ناخوانا خواهد بود. دوم کارمندانی که با پایگاه داده کار می کنند، نمی توانند اطلاعات مهم کاربران را مشاهده کنند.

- برای رمزگاری Database از plugin به اسم `mongoose-encryption` استفاده می شود.

- برای ایجاد Schema از `mongoose.Schema` استفاده می کنیم تا قابلیت افزودن plugin را داشته باشیم.



## [mongoose-encryption](#)

2.0.1 • [Public](#) • Published 6 months ago

[Readme](#)

[7 Dependencies](#)

[14 Dep](#)

# [mongoose-encryption](#)

[npm package](#) [2.0.1](#) [build](#) [passing](#) [license](#) [MIT](#)

Simple encryption and authentication for mongoose documents. Relies on the Node `crypto` module. Encryption and decryption happen transparently during save and find. Rather than encrypting fields individually, this plugin takes advantage of the BSON nature of mongoDB documents to encrypt multiple fields at once.

## How it Works

Encryption is performed using `AES-256-CBC` with a random, unique initialization vector for each operation. Authentication is performed using `HMAC-SHA-512`.

### Basic

By default, all fields are encrypted except for `_id`, `__v`, and fields with indexes

```
var mongoose = require('mongoose');
var encrypt = require('mongoose-encryption');

var userSchema = new mongoose.Schema({
  name: String,
  age: Number
  // whatever else
});
```

### Secret String Instead of Two Keys

For convenience, you can also pass in a single secret string instead of two keys.

```
var secret = process.env.SOME_LONG_UNGUESSABLE_STRING;
userSchema.plugin(encrypt, { secret: secret });
```

### Encrypt Only Certain Fields

You can also specify exactly which fields to encrypt with the `encryptedFields` option. This overrides the defaults and all other options.

```
of any other options. name and _id will be left unencrypted
, { encryptionKey: encKey, signingKey: sigKey, encryptedFields: ['age'] });
```

The screenshot shows the mongoose official website. On the left, there's a sidebar with links like 'Version 5.4.6', 'Quick Start', 'Guides', 'Schemas', and 'SchemaTypes'. The main content area has a header 'Plugins' and a 'Sponsor' banner for Slack. Below that, it says 'Schemas are pluggable, that is, they allow for applying pre-packaged capabilities to extend their functionality. This is a very powerful feature.' A code editor window displays the following JavaScript code:

```
const userSchema = new mongoose.Schema({
  email: String,
  password: String
});

const secret = "Thisisourlittlesecret.";
userSchema.plugin(encrypt, {
  secret: secret,
  encryptedFields: ["password"]
});

const User = new mongoose.model("User", userSchema);
```

**Key words:** Level 2, Database Encryption, mongoose-encryption.

## 6. Using Environment Variables to Keep Secrets Safe

- در قسمت قبل Database رمزنگاری شد اما همان طور که دیدیم اگر کسی سرور app.js را هک کند، به راحتی می توان به کلید یا Secret دسترسی داشته و در ادامه اطلاعات database را بدست آورد.
- برای جلوگیری از این منظور، باید از environment variables استفاده کنیم تا متغیرهای حساس را در آن محیط نگهداری کنیم. برای این منظور از ماژول dotenv با ایجاد فایل .env. استفاده می کنیم. همچنین باید فایل .env. در موارد .gitignore نیز قرار دهیم.

**dotenv**

6.2.0 • Public • Published 2 months ago

Readme 0 Dependencies 7,967 Depend

## dotenv

Dotenv is a zero-dependency module that loads environment variables from a `.env` file into `process.env`. Storing configuration in the environment separate from code is based on [The Twelve-Factor App](#) methodology.

build passing ⚡ build failing npm v6.2.0 code style standard coverage 100%

**Install**

به نحوه کردن و ساختار `variables` در `.env` توجه کنید. -

## Usage

As early as possible in your application, require and configure dotenv.

```
require('dotenv').config()
```

Create a `.env` file in the root directory of your project. Add environment-specific variables on new lines in the form of `NAME=VALUE`. For example:

```
DB_HOST=localhost  
DB_USER=root  
DB_PASS=simpl3
```

[US] | <https://github.com/github/gitignore/blob/master/Node.gitignore>

Branch: master ▾ **gitignore / Node.gitignore**

 shiftkey Merge pull request #2908 from lmedson/env\_test\_ignore

44 contributors 

81 lines (57 sloc) | 1.12 KB [Raw](#) [B](#)

```
1  # Logs
2  logs
3  *.log
4  npm-debug.log*
5  yarn-debug.log*
6  yarn-error.log*
7
8  # Runtime data
9  pids
10 *.pid
11 *.seed
12 *.pid.lock
13
14 # Directory for instrumented libs generated by jscoverage/JSCover
15 lib-cov
16
17 # Coverage directory used by tools like istanbul
18 coverage
19
20 # nyc test coverage
21 .nyc_output
22
23 # Grunt intermediate storage (https://gruntjs.com/creating-plugins#storing-task-files)
24 .grunt
```

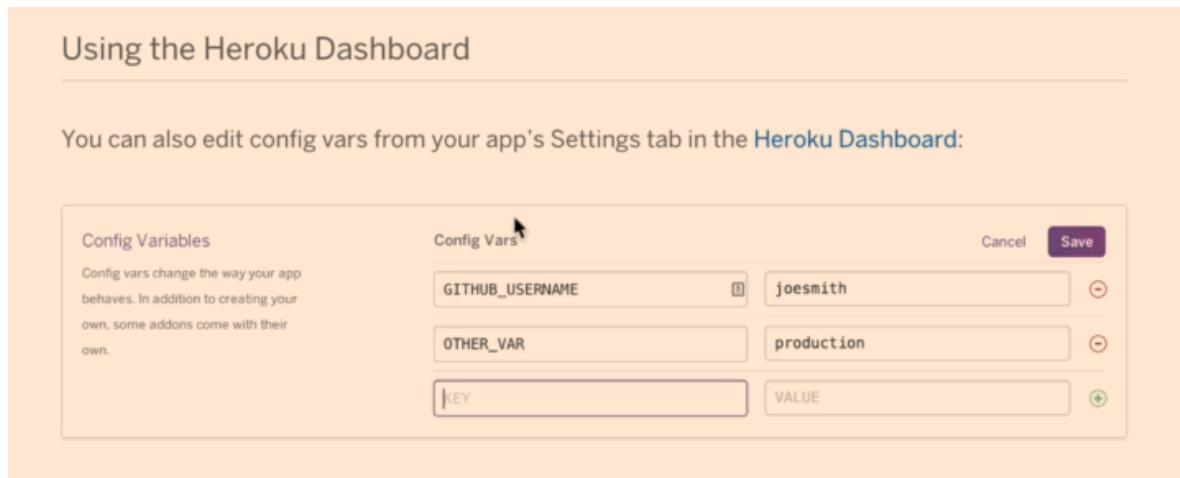


```
require("dotenv").config();
const express = require("express");
const bodyParser = require("body-parser");
const ejs = require("ejs");
const mongoose = require("mongoose");
const encrypt = require("mongoose-encryption");
```

app.js		.env
SECRET=Thisisourlittlesecret.	-----	
API_KEY=oiweruweiuerowur	-----	

```
userSchema.plugin(encrypt, { secret: process.env.SECRET, encryptedFields: ["password"] });
```

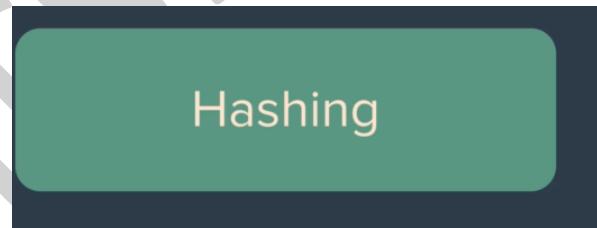
- هنگام ایجاد پروژه در Heroku مکانی برای نگهداری متغیرهای مهم وجود دارد که همانند `.env` عمل می‌کند.

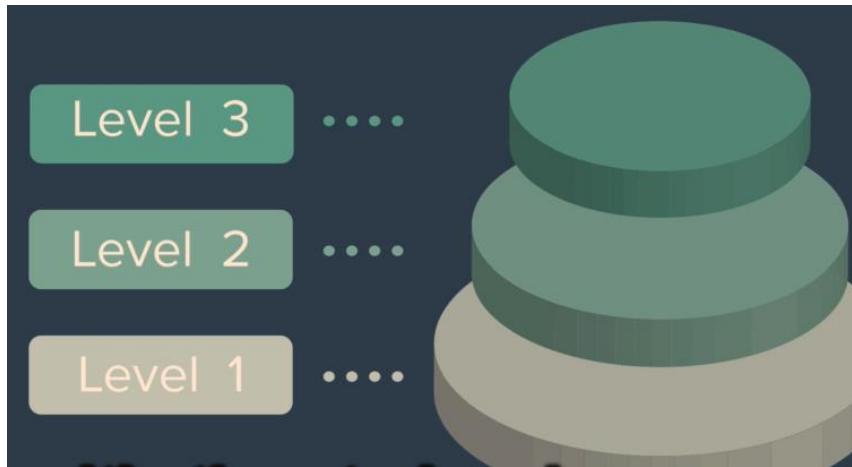


**Key words:** Using Environment Variables to Keep Secrets Safe, dotenv module, `.env` file.

## 7. Level 3 - Hashing Passwords

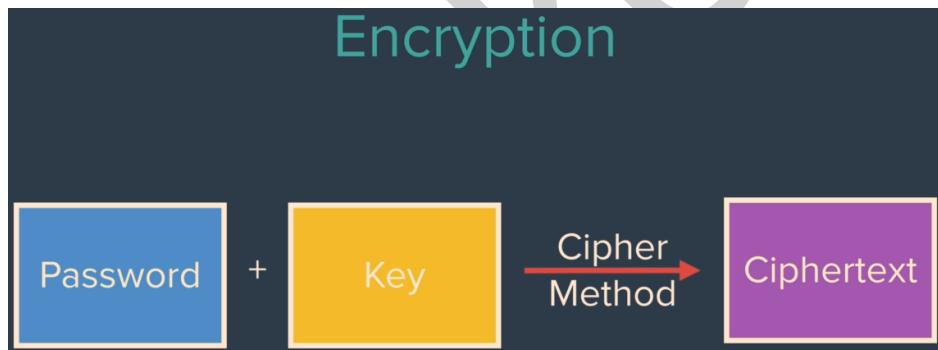
- در این قسمت به سطح ۳ برای افزایش امنیت اطلاعات می‌رویم و درباره Hashing توضیح داده می‌شود.





- در شکل زیر روش Encryption یا رمزگاری و بر عکس آن یعنی Decryption یا رمزگشایی را نشان می دهد.

- در روش مرسوم رمزگاری، اگر Key فاش شود، هکر می تواند کل اطلاعات را Decryption کرده و اطلاعات را بدست آورد. در روش های جدید رمزگاری از Hash Function استفاده می شود.





- در Hashing قابلیت سریع رمزنگاری وجود دارد، ولی عکس آن یا رمزگشایی با سخت افزارهای عصر حاضر کاری زمانبر خواهد بود. به عبارتی در Hashing رمزنگاری وجود دارد، ولی رمزگشایی عملی سخت می باشد. همچنین key وجود ندارد که هر اطلاعات را هک کند.
- در واقع به جای key و cipher method از Hash Function استفاده می شود.
- خروجی Hash Function یا کد رمز شده به روش Hashing می باشد.



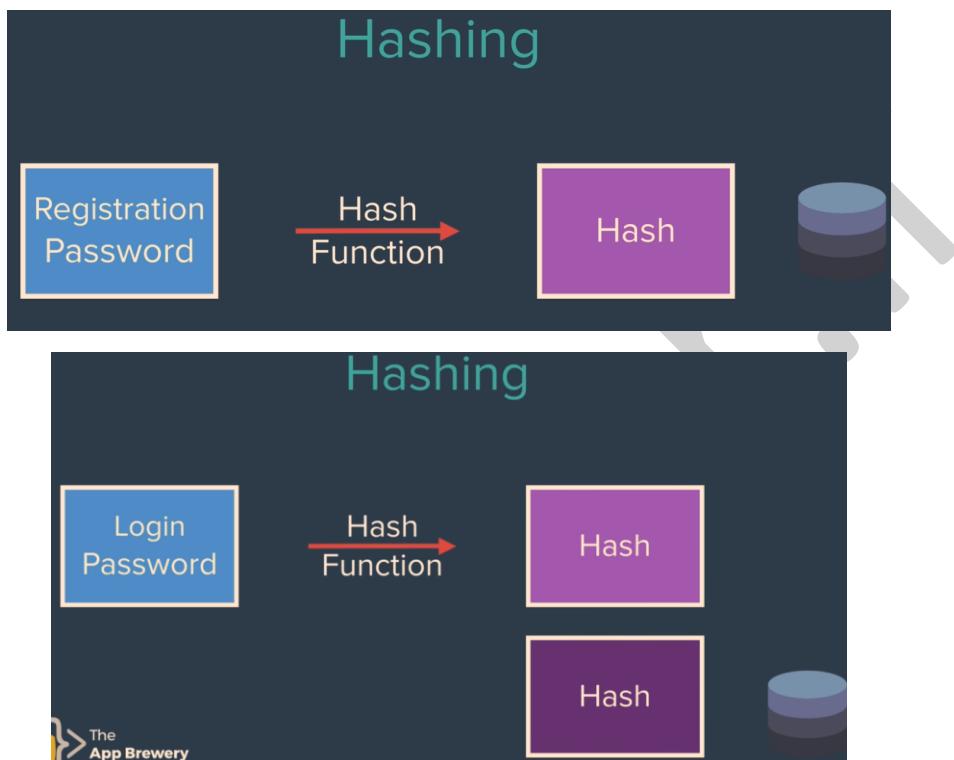
What are the factors of 377  
other than 1 and 377?

$$\frac{377}{1} = 377$$

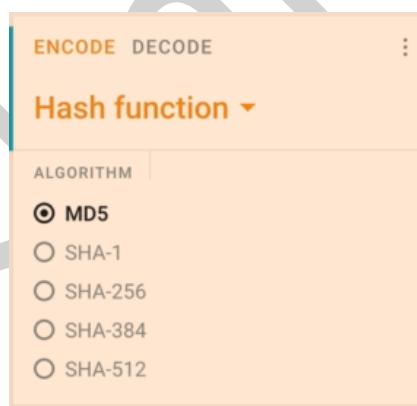
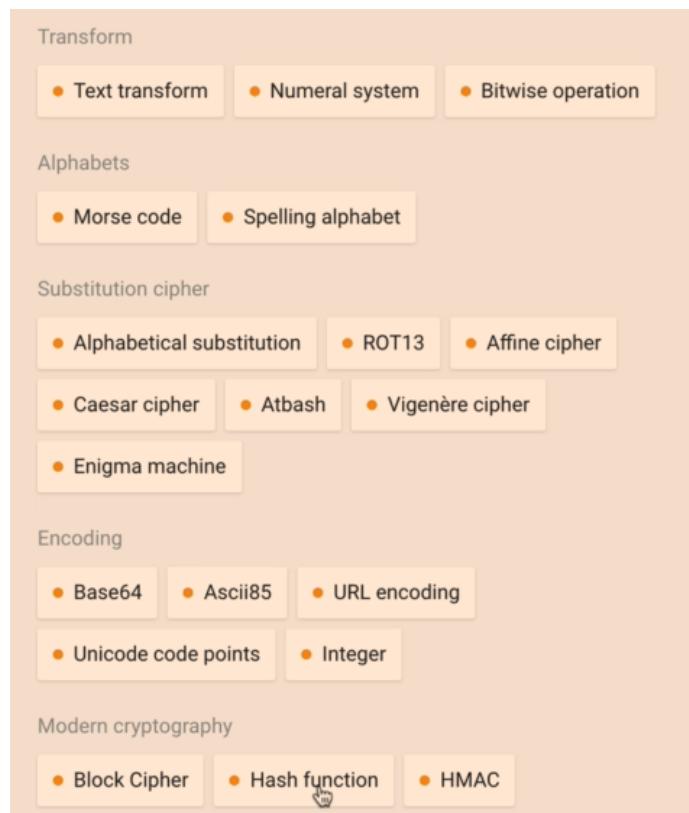
$$\frac{377}{13} = 29$$

Multiply 13 by 29  
 $13 \times 29 = 377$

- در عمل Hash ایجاد شده در Database ذخیره شده، و برای login در قسمت Authentication در Database ذخیره شده، و برای Hash باید Hash ایجاد شده و Hash موجود در Database برابر باشند. به عبارتی فقط کاربر یا صاحب رمز توانایی دسترسی به حساب خود را دارد و حتی مالک سایت یا password Database نمی تواند را بفهمد.



- شکل زیر روش های مدرن و سنتی رمزگاری را نشان می دهد.



- یکی از روش های Hashing به اسم md5 می باشد که در پروژه خود استفاده می کنیم.

**md5**

2.2.1 • Public • Published 2 years ago

Readme 3 Dependencies 2,231

## MD5

build passing

a JavaScript function for hashing messages with MD5.

**Node.js:**

```
npm install md5
```

**API**

`md5(message)`

- message -- String or Buffer
- returns String

**Usage**

```
var md5 = require('md5');

console.log(md5('message'));
```

This will print the following

```
78e731027d8fd50ed642340b7c9a63b3
```

```
app.post("/register", function(req, res){
  const newUser = new User({
    email: req.body.username,
    password: md5(req.body.password)
});
```

```
app.post("/login", function(req, res){
  const username = req.body.username;
  const password = md5(req.body.password);
```

```
console.log("weak password hash: " + md5("123456"));
console.log("strong password hash: " + md5("sjkhdfsd8f7jhsd$%$sdfsdfHJKHSJFHD$F78324"));
```

- به صورت Hash ذخیره شده است.

Key	Value
▶ (1) ObjectId("5c43153a6401980f1f11c4a4")	{ 5 fields }
▶ (2) ObjectId("5c4838d57f39e25f146a0788")	{ 5 fields }
▼ (3) ObjectId("5c485ea734190f633d6380e3")	{ 4 fields }
□ _id	ObjectId("5c485ea734190f633d6380e3")
□ email	user@hash.com
□ password	e10adc3949ba59abbe56e057f20f883e
□ __v	0

**Key words:** Level 3, Hashing Passwords, md5.

## 8. Hacking 101

- در این قسمت درباره هک سطح کلاس ۱۰۱ صحبت می شود. در واقع hacking 101 برای هک اخلاقی استفاده می شود یعنی بتوانیم با روش های هک و امنیت آشنا شده و از آن برای جلوگیری از هکرهای استفاده کنیم. ( کتاب Hacker 101 نیز موجود می باشد)



- یکسری سایت ها هستند، که ایمیل و پسورد شما را به صورت plain text برای شما ایمیل می کنند. این نشان دهنده ضعیف بودن امنیت سایت است ، چون همانطور که می دانیم، در صورت استفاده از Hash امکان بازیابی آن وجود ندارد. بنابراین از این سایت ها استفاده نکنید و اکانت خود را پاک کنید.

C ⓘ Not Secure | plaintextoffenders.com

# Plain Text Offenders

Did you just email me back my own password?

[About](#)

[FAQ](#)

[Developers FAQ](#)

[Offenders List](#)

[3rd Party Tools](#)

[Reformed Offenders](#)

[Archive](#)

[Talk To Us](#)

[Submit a post](#)

NOTE: Tumblr's search feature is broken and therefore disabled. Please use the list at [plaintextoffenders.com/offenders](http://plaintextoffenders.com/offenders) to search for any domain.

January 31st, 2019 at 6:

Dear [REDACTED]

Thank you for registering as job seeker with Get Linux Jobs.

Your account information is as follows:

Email: [REDACTED]  
Password: [REDACTED]

[You can sign in and manage your profile and resume here](#)

\*IMPORTANT - Please keep this information in a secure place.

Once again thank you again for registering and for being part of our community!

To your success,

Get Linux Jobs Team  
GetLinuxJobs.com  
[contact@getlinuxjobs.com](mailto:contact@getlinuxjobs.com)  
<https://www.getlinuxjobs.com>

getlinuxjobs.com

Job listings

- در سایت زیر می توان ایمیل خود را وارد کرده و بررسی کنید که آیا ایمیل شما در Database هایی که هک شده وجود دارد یا خیر؟ اگر وجود داشت، password خود را تغییر دهید.

s://haveibeenpwned.com

';-)

[Home](#) [Notify me](#) [Domain search](#) [Who's been pwned](#) [Passwords](#) [API](#) [About](#) [Donate ⌃ P](#)

## ';-have i been pwned?

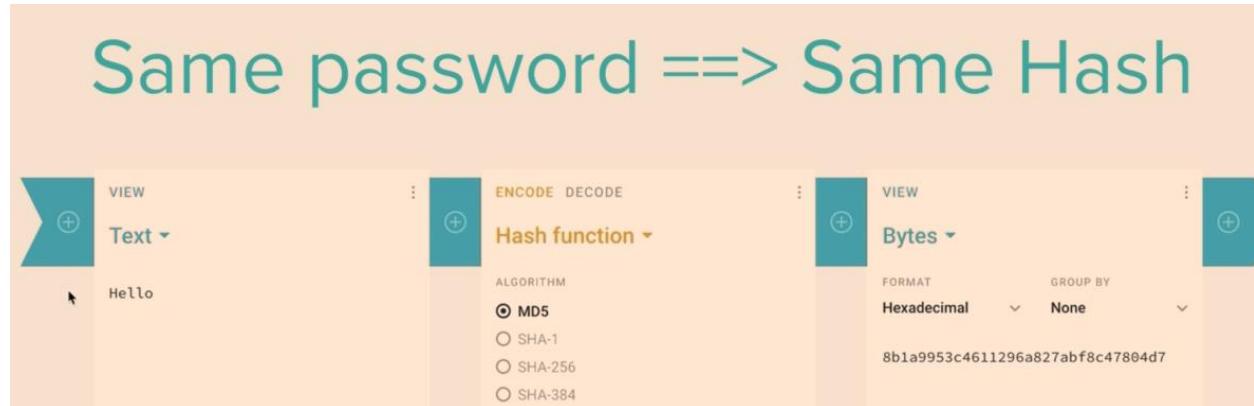
Check if you have an account that has been compromised in a data breach

email address

pwned?

- یادتان باشد، در روش hashing ، ورودی های یکسان، hash های یکسان خواهد داشت. با همین اصل می توان با داشتن hash کاربران توسط هکر، با توجه به کلمات و رمزهای پرطرفدار ، می توان رمز کاربرانی که رمز ضعیف دارند را پیدا کرد.

## Same password ==> Same Hash



Username	Hash	Password	Hash
angela@gmail.com	d8578edf8458ce06fbc5 bb76a58c5ca4	123456	e10adc3949ba59abbe5 6e057f20f883e
john @gmail.com	a0ea63ddd9dc071f8c73 19d391b31b88	qwerty	d8578edf8458ce06fbc5 bb76a58c5ca4
tony@gmail.com	d8578edf8458ce06fbc5 bb76a58c5ca4	password	5f4dcc3b5aa765d61d83 27deb882cf99
emily@gmail.com	d8578edf8458ce06fbc5 bb76a58c5ca4	111111	96e79218965eb72c92a5 49dd5a330112

- در حملات Dictionary attack یا Brute-force ، هکر با ایجاد دیکشنری کامل از کلمات، اعداد دفترچه تلفنی، و کاراکترهای پرکاربرد، و رمزهای پرکاربرد و تکراری، می تواند اکثر رمزها را امتحان کند.  
- در نوع دیگر حمله، به اسم Hash table ، با ایجاد Hash table از دیکشنری ذکر شده، با داشتن می تواند رمزهای کاربران را حدث بزند.

# Let's Make a Hash Table!

- All words from a dictionary ( $\approx 150,000$ )
  - All numbers from a telephone book ( $\approx 5,000,000$ )
  - All combinations of characters up to 6 places (19,770,609,664)
- 

19'775'759'664

- روش md5 سریع ترین روش ایجاد hash می باشد، بنابراین یک گرافیک معمولی به راحتی در یک ثانیه می تواند ۲۰ میلیارد Hash را ایجاد و تست کند.

20,000,000,000 MD5 Hashes/Second

19,775,759,664

---

0.9s

Most common passwords list

#:	Password	MD5	Length	L	U	N	Meter
1	password	5f4dcc3b5aa765d61d8327deb882cf99	8	8	0	0	check
2	123456	e10adc3949ba59abbe56e057f20f883e	6	0	0	6	check
3	12345678	25d55ad283aa400af464c76d713c07ad	8	0	0	8	check
4	1234	81dc9bdb52d04dc20036dbd8313ed055	4	0	0	4	check
5	qwerty	d8578edf8458ce06fbcb5bb76a58c5ca4	6	6	0	0	check
6	12345	827ccb0eea8a706c4c34a16891f84e7b	5	0	0	5	check
7	dragon	8621ffdbc5698829397d97767ac13db3	6	6	0	0	check
8	pussy	acc6f2779b808637d04c71e3d8360eeb	5	5	0	0	check
9	baseball	276f8db0b86edaa7fc805516c852c889	8	8	0	0	check
10	football	37b4e2d82900d5e94b8da524fbbe33c0	8	8	0	0	check
11	letmein	0d107d09f5bbe40cade3de5c71e9e9b7	7	7	0	0	check

- در سایت زیر می توان قدرت password را تست کرد و مدت زمان کشف آن توسط کارت گرافیک های مختلف را نشان می دهد.
- در نتیجه بهترین رمز باید: ۱- طولانی باشد، ۲- از کلمات متداول استفاده نشده باشد ۳- ترکیبی از حروف، اعداد و کاراکترها باشد.

## Password Checker Online

Verifier

Password: itkdl



Strength: 20%

Evaluation: Low!

### Password properties

Property	Value	Comment
Password length:	6	SHORT
Numbers:	0	NOT USED
Letters:	6	USED
Uppercase Letters:	0	NOT USED
Lowercase Letters:	6	USED
Symbols	0	NOT USED
Charset size	26	LOW (a-z)
TOP 10000 password	NO	Password is NOT one of the most frequently used passwords.

### Brute-force attack cracking time estimate

Machine	Time
Standard Desktop PC	3 seconds
Fast Desktop PC	1 second
GPU	0 seconds
Fast GPU	0 seconds
Parallel GPUs	0 seconds
Medium size botnet	0 seconds

password-checker.online-domain-tools.com

### Verifier

### Password Checker Online

list to  
ation  
s per task  
urting at  
til  
liable  
**pay to**  
0 USD  
USD  
USD

Property	Value	Comment
Password length:	12	OK
Numbers:	0	NOT USED
Letters:	12	USED
Uppercase Letters:	0	NOT USED
Lowercase Letters:	12	USED
Symbols	0	NOT USED
Charset size	26	LOW (a-z)
TOP 10000 password	NO	Password is NOT one of the most frequently used passwords.

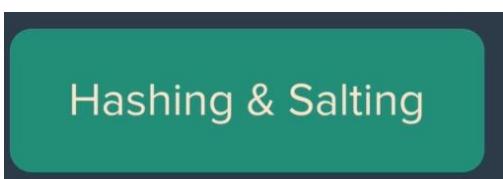
Brute-force attack cracking time estimate

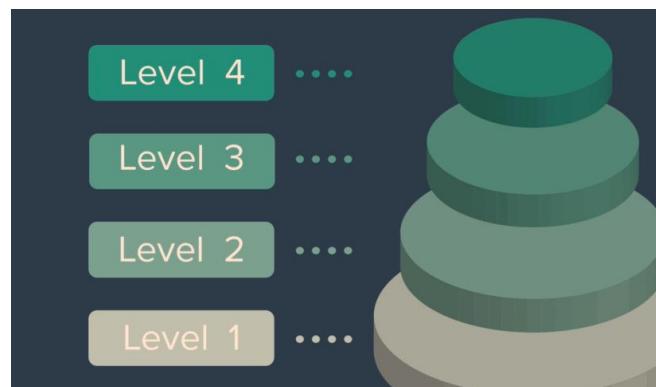
Machine	Time
Standard Desktop PC	About 31 years
Fast Desktop PC	About 8 years
GPU	About 3 years
Fast GPU	About 2 years
Parallel GPUs	About 2 months
Medium size botnet	About 16 minutes

**Key words:** Hacking 101, brut-force attack or dictionary attack, hash table attack.

## 9. Level 4 - Salting and Hashing Passwords with bcrypt

- در سطح ۴ امنیت، از Salting استفاده می کنیم -





- در Hashing یک مقدار تصادفی به اصطلاح Salt در کنار password قرار گرفته و بعد عمل صورت می گیرد. با این کار هم طول و هم تنوع کاراکترهای password بیشتر می شود و در نتیجه خروجی در مقابل حملات hash table مصون تر خواهد شد.



- برای عمل salting از مازول bcrypt استفاده می کنیم که در شکل زیر تفاوت پردازش Salting و Hashing تنها را در یک گرافیک معمولی مشاهده می کنید.

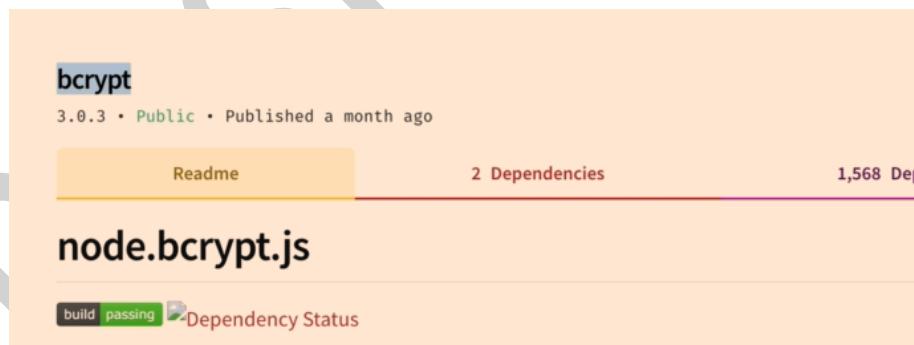
20,000,000,000 MD5 Hashes/Second
17,000 bcrypt Hashes/Second

- در مرحله بعد برای افزایش امنیت و افزایش زمان پردازش Hashing از Salt Rounds استفاده می کنیم.  
به این شکل که بعد از ایجاد اولین Hash به تعداد Salt Round عمل Hashing انجام می شود در نتیجه مدت زمان پردازش بیشتر می شود، و هکر ناتوان تر خواهد شد.



- در شکل زیر مقدار تصادفی Salt و Hash خروجی و Round را نشان می دهد.

Username	Salt	Hash x 10
angela@gmail.com	hjs95dfHAs72Gv3o	1cb8d9a17982c7e25e6d4ae868356bb0
john@gmail.com	73gHJTusdf92bsdf	e826ce28c7c7862517872dbb15336327
tony@gmail.com	sdf8sdfjhsdfbIUHF	47075ccd564621206c659590ee05e491
emily@gmail.com	74hgjJHVBSfvasd	dbdfca8c97a28066bcde4cc1b38d51aa



- برای استفاده از bcrypt به سازگاری آن به ورژن Node.js توجه کنید. برای این منظور از NVM یا استفاده می کنیم که قابلیت نصب نسخه های مختلف node.js را به ما می دهد.

**Version Compatibility**

Node Version	Bcrypt Version
0.4	<= 0.4
0.6, 0.8, 0.10	>= 0.5
0.11	>= 0.8
4	< 2.1
8	>= 1.0.3
10	>= 3

Branch: master ▾ nvm / README.md

waldyrious README: clarify how to specify default packages

31 contributors

710 lines (482 sloc) | 28.1 KB

Raw Bl

Node Version Manager build passing version v0.34.0 cli best practices passing

### Table of Contents

- Installation
  - Install script
    - Ansible

```
Angelas-MacBook-Pro:~ angelayu$ nvm --version
0.34.0
Angelas-MacBook-Pro:~ angelayu$ nvm install 10.15.0
```

- در صورت داشتن خطأ و هشدا در نصب هر ماژولی ، کافیست به قسمت Issue در github رفته و مشکل خود را برطرف کنید.

The screenshot shows the GitHub repository page for `kelektiv/node.bcrypt.js`. At the top, there are navigation links: Why GitHub? (dropdown), Enterprise, Explore (dropdown), Marketplace, and Pricing (dropdown). Below the header, the repository name is displayed. A navigation bar includes tabs for Code, Issues (22, highlighted), Pull requests (3), Projects (0), and Wiki. A search bar contains the query `is:issue is:open`. Below the search bar are buttons for Labels and Milestones. The main content area shows statistics: 22 Open and 552 Closed issues. A list of three open issues is shown:

- ① "npm install bcrypt" failing with "Error: Cannot find module 'nan'" #697 opened 3 days ago by kissenger
- ② How to avoid bcrypt generating hashes with a slash? #696 opened 5 days ago by deemeetree
- ③ Bcrypt Installation error `node-pre-gyp install --fallback-to-build` #695 opened 5 days ago by rigalpatel001

## Usage

### async (recommended)

```
const bcrypt = require('bcrypt');
const saltRounds = 10;
const myPlaintextPassword = 's0/\u2f4\$\$w0rD';
const someOtherPlaintextPassword = 'not_bacon';
```

- در شکل زیر مدت زمان پردازش با توجه به مقدار `round` را نشان می دهد و هر سال با پیشرفت پردازندۀ ها باید مقدار `round` را افزایش دهیم.

## A Note on Rounds

A note about the cost. When you are hashing your data the module will go through a series of rounds to give you a secure hash. The value you submit there is not just the number of rounds that the module will go through to hash your data. The module will use the value you enter and go through  $2^{\text{rounds}}$  iterations of processing.

From @garthk, on a 2GHz core you can roughly expect:

```
rounds=8 : ~40 hashes/sec
→ rounds=9 : ~20 hashes/sec
rounds=10: ~10 hashes/sec
rounds=11: ~5 hashes/sec
rounds=12: 2-3 hashes/sec
rounds=13: ~1 sec/hash
rounds=14: ~1.5 sec/hash
rounds=15: ~3 sec/hash
rounds=25: ~1 hour/hash
rounds=31: 2-3 days/hash
```

در شکل زیر نحوه استفاده از bcrypt را نشان می دهد که Hash خروجی ترکیبی از round و password می باشد.

Technique 2 (auto-gen a salt and hash):

```
bcrypt.hash(myPlaintextPassword, saltRounds, function(err, hash) {
  // Store hash in your password DB.
});
```

To check a password:

```
// Load hash from your password DB.
bcrypt.compare(myPlaintextPassword, hash, function(err, res) {
  // res == true
});
```

```
const bcrypt = require("bcrypt");
const saltRounds = 10;
```

```
app.post("/register", function(req, res){  
  
    bcrypt.hash(req.body.password, saltRounds, function(err, hash) {  
        const newUser = new User({  
            email: req.body.username,  
            password: hash  
        });  
        newUser.save(function(err){  
            if (err) {  
                console.log(err);  
            } else {  
                res.render("secrets");  
            }  
        });  
    });  
  
});
```

```
app.post("/login", function(req, res){  
    const username = req.body.username;  
    const password = req.body.password;  
  
    User.findOne({email: username}, function(err, foundUser){  
        if (err) {  
            console.log(err);  
        } else {  
            if (foundUser) {  
                bcrypt.compare(password, foundUser.password, function(err, result) {  
                    if (result === true) {  
                        res.render("secrets");  
                    }  
                });  
            }  
        }  
    });  
});  
});
```

▼ (4) ObjectId("5c49b0a48022737b7f5b98ee")	{ 4 fields }	Object
↳ _id	ObjectId("5c49b0a48022737b7f5b98ee")	ObjectID
↳ email	user@bcrypthash.com	String
↳ password	\$2b\$10\$Pv4gs/g4OdBkIcgUkrSJZu5X47oEYKLupC8F3ak9R...	String
↳ __v	0	Int32

**Key words:** Level 4, Salting and Hashing Passwords with bcrypt, nvm module, how to issue in Github.

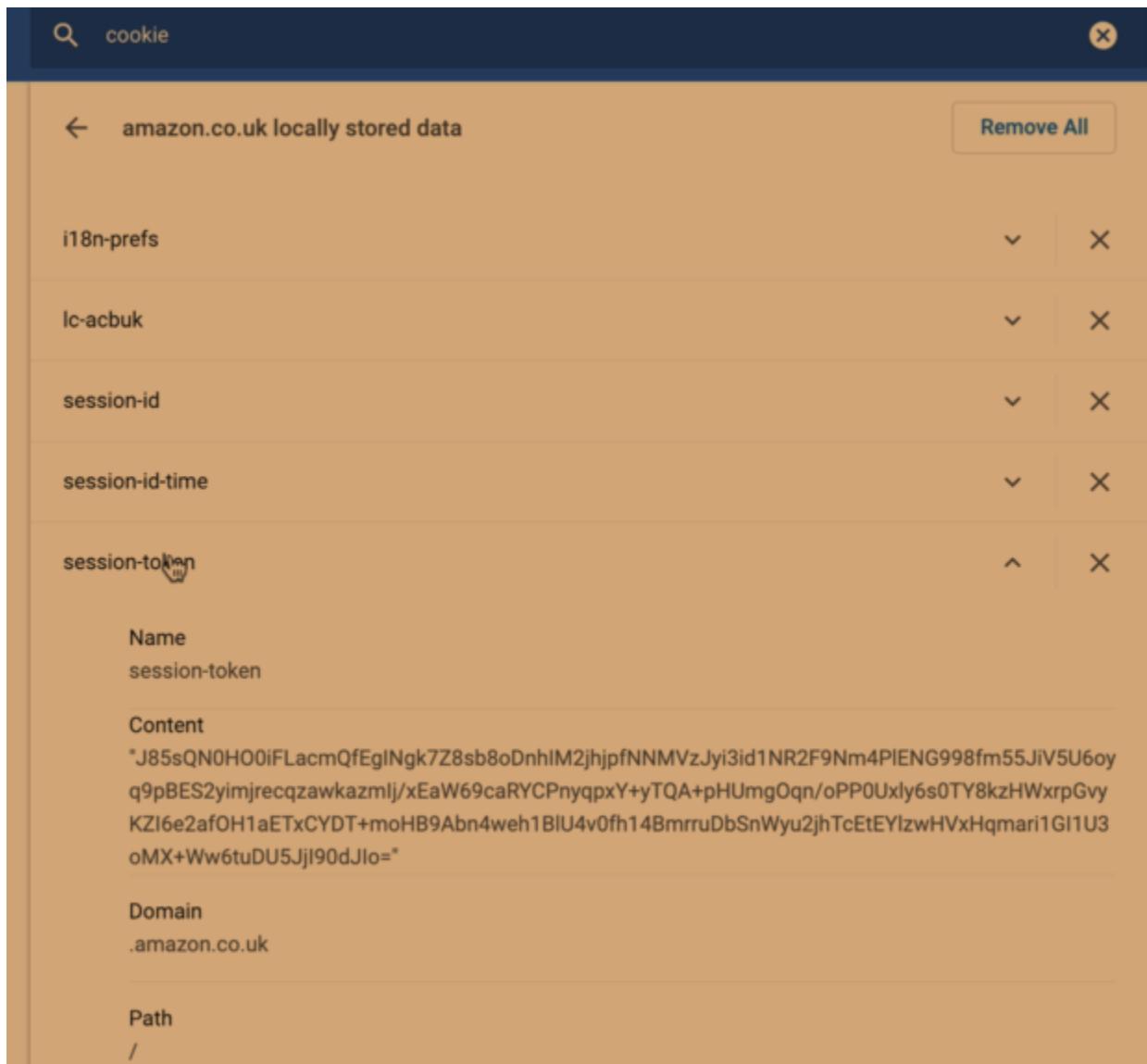
## 10. What are Cookies and Sessions

- در این قسمت درباره Cookies و Sessions که امنیت level 5 برای Authentication می باشد صحبت می کنیم.

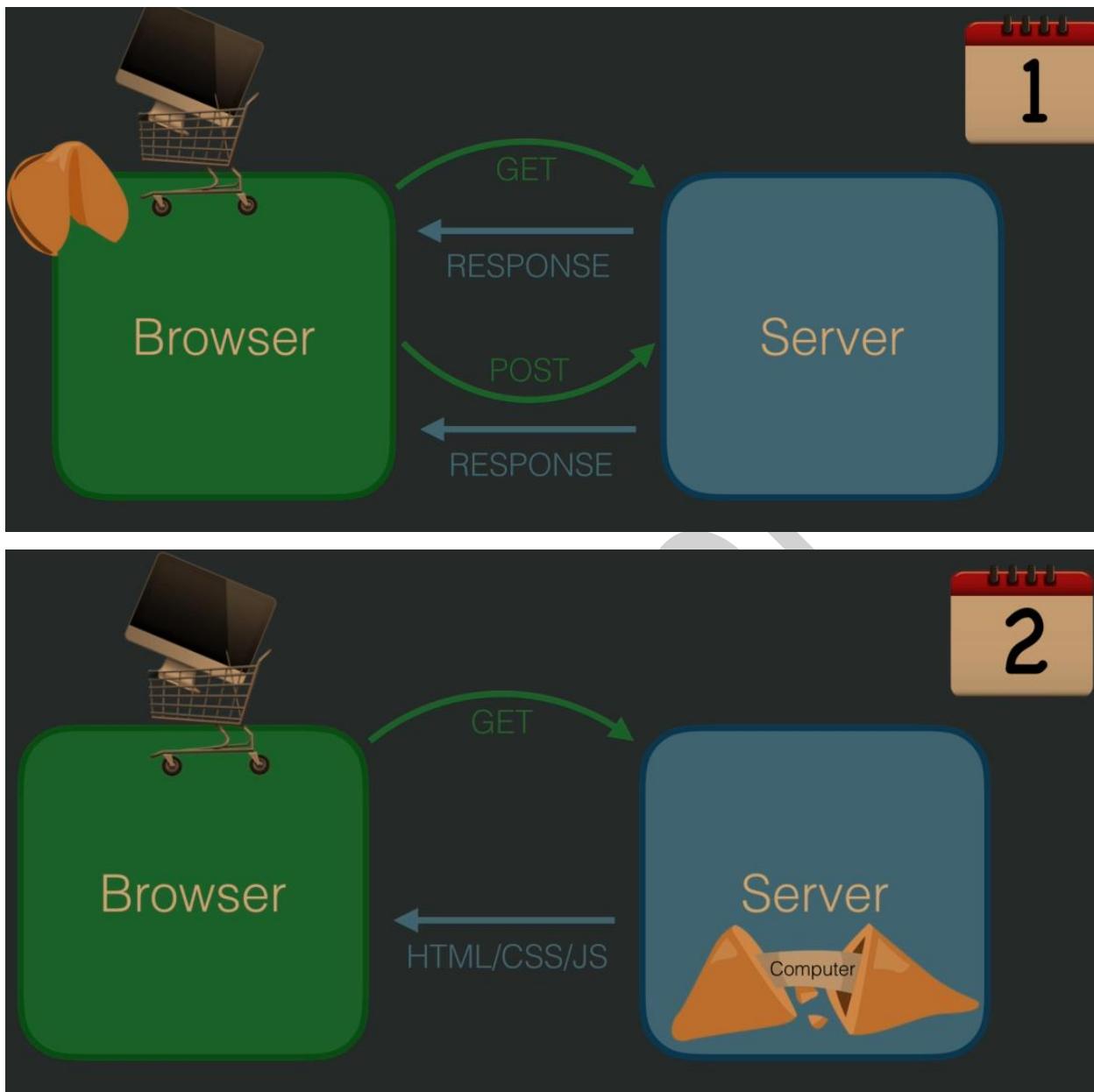
### Cookies & Sessions



- اطلاعات موقتی می باشد، که با توجه به تعاملاتی که کاربر در Session ها با سایت ها دارد در مرورگر کاربر ذخیره می شود. مثلاً برای سایت فروشگاهی، افزودن خریدهایی که می کنیم، به عنوان cookies در مرورگر کاربر ذخیره می شود، تا در مراجعه بعدی آن خریدها حذف نشده باشد. Session یا جلسه یا نشست، به معنی مراجعه هر باره ما به آن سایت می باشد، مثلاً وقتی صفحه سایتی را می بندیم، در واقع Session را می بندیم.

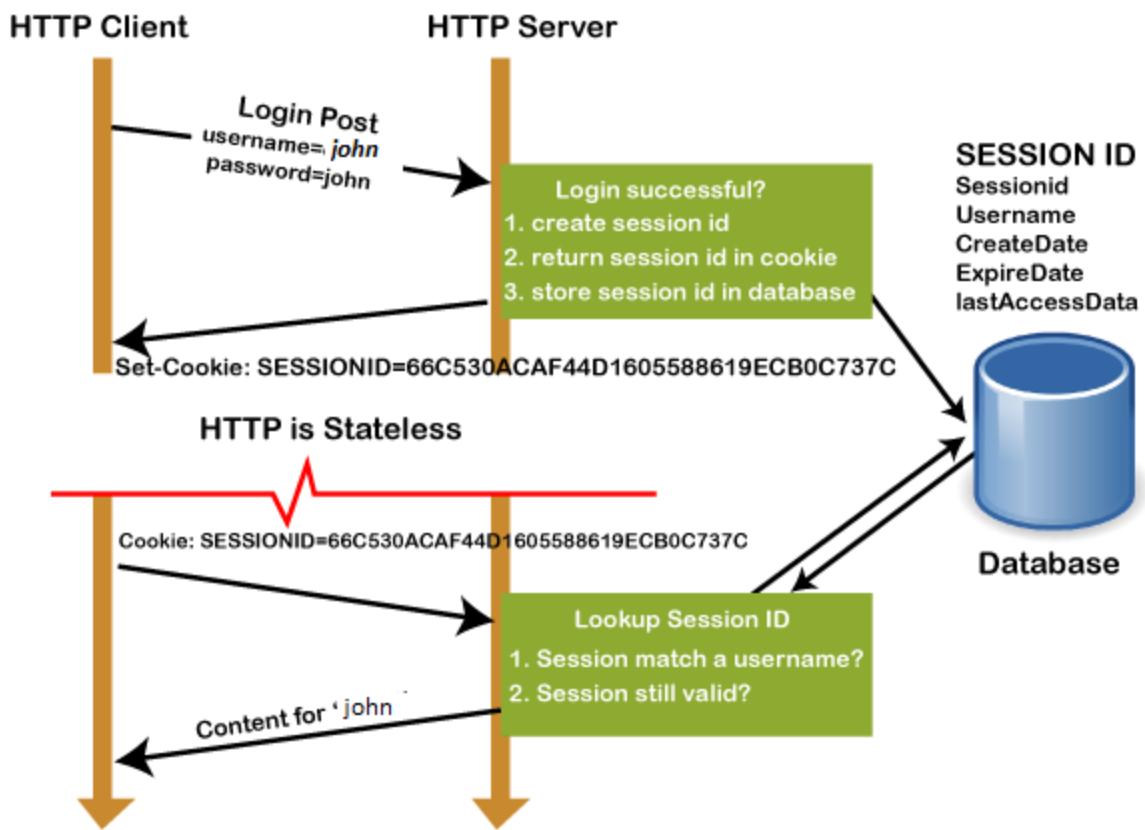


- در ۲ شکل زیر، در Session اول اطلاعات به صورت `cookies` در مرورگر کاربر ذخیره شده و در مراجعه بعدی یا Session دوم، آن `cookies` از طرف کاربر به سرور ارائه شده، و کاربر سایت مورد نظر را با تنظیمات یا تعاملاتی که از قبل انجام داده مشاهده خواهد کرد. در واقع `cookies` از تکرار اطلاعات جلوگیری می کنند.



- یکی از کاربردهای authentication برای cookies و Session می باشد. در شکل زیر وقتی cookies معتبر شروع می شود و اطلاعات authenticated و تعاملات در Log In Session می کنیم. حال در ادامه اگر Log out کنیم، آن Session پایان یافته و همچنانین ذخیره می شود، (ذخیره اطلاعات authencatied data شامل browsing session id) نیز از بین خواهد رفت. (ذخیره اطلاعات Authentication نام کاربری و پسورد به صورت cookies باعث می شود که با مرور یا در صفحات مختلف سایت، نیاز به هر بار Log in نباشد یا به عبارتی Session معتبر داریم).



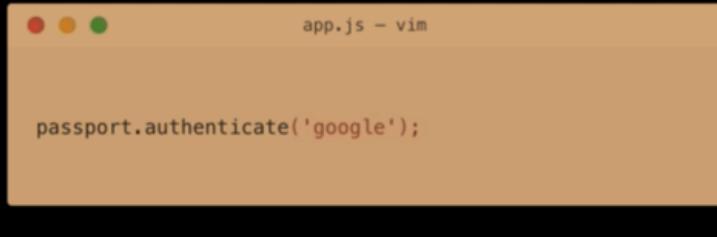


- مازول **Sessions** و **cookies** قابلیت **Authentication** در **node.js** با استفاده از **Passport** دارد.

# Passport

Simple, unobtrusive authentication for Node.js

Passport is authentication middleware for Node.js. Extremely flexible and modular, Passport can be unobtrusively dropped in to any Express-based web application. A comprehensive set of strategies support authentication using a username and password, Facebook, Twitter, and more.



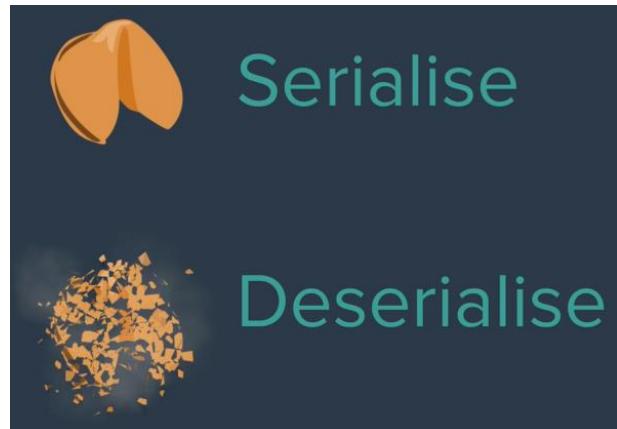
**Key words:** Level 5, What are Cookies and Sessions.

## 11. Using Passport.js to Add Cookies and Sessions

- در این قسمت برای Authentication توسط cookies و Session باید مازول های زیر را در پروژه نصب کنیم. (دقت کنید که مازول ها به ما کمک می کنند کدهای کمتری بنویسیم)
- مازول Passport قابلیت Authentication را می دهد، ولی با نصب مازول های دیگر، خط کد کمتر و کار راحت تری با Sessions و cookies برای استفاده در passport خواهیم داشت.
- مازول Passport-local-mongoose قابلیت ایجاد salt و hash را می دهد و آن را در Database ذخیره می کند(نیازی به استفاده از bcrypt و مراحل قسمت ها قبل نیست).
- مازول passport-local-mongoose نیاز Passport-local مورد نیاز passport-local به صورت مستقیم استفاده نمی شود.
- مازول Express-session Express-session مورد نیاز برقراری Session در Express می باشد.

## npm install

- passport
- passport-local
- passport-local-mongoose
- express-session NOT express-sessions



- حتماً کد اصلی پروژه را بررسی کنید، و به ترتیب قرار گیری کدها توجه و آن را رعایت کنید.
- حتماً document هر مازول را بررسی و نحوه استفاده از آن را بفهمید.
- فیلم این قسمت را برای فهم بیشتر مشاهده نمایید.

در کد زیر مازول ها را require می کنیم. -

```
const session = require('express-session');
const passport = require("passport");
const passportLocalMongoose = require("passport-local-mongoose");
```

در کد زیر استفاده از express session را فعال می کنیم. -

```
app.use(session({
  secret: "Our little secret.",
  resave: false,
  saveUninitialized: false
}));
```

- در کد زیر مژول passport را به منظور Authentication در Express فعال می کنیم.

```
app.use(passport.initialize());
app.use(passport.session());
```

- در کد زیر مژول userSchema را برای passport-local-mongoose فعال می کنیم.

```
userSchema.plugin(passportLocalMongoose);
```

- در کد زیر پیکربندی اولیه و ایجاد cookie برای مدل یا کالکشن User به منظور authentication را فعال می کنیم.

- انجام می دهد.(زمانی که Session شروع یا پایان میابد).

```
passport.use(User.createStrategy());

passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());
```

- در کد زیر عمل register توسط passport-local-mongoose انجام شده، و در صورت موفقیت توسط authentication عمل cookie در قالب passport انجام می شود.

```
app.post("/register", function(req, res){

  User.register({username: req.body.username}, req.body.password, function(err, user){
    if (err) {
      console.log(err);
      res.redirect("/register");
    } else {
      passport.authenticate("local")(req, res, function(){
        res.redirect("/secrets");
      });
    }
  });
});
```

- در کد زیر نیز، عمل log in برای authenticaton توسط passport می باشد.

```
app.post("/login", function(req, res){  
  
    const user = new User({  
        username: req.body.username,  
        password: req.body.password  
   });  
  
    req.login(user, function(err){  
        if (err) {  
            console.log(err);  
        } else {  
            passport.authenticate("local")(req, res, function(){  
                res.redirect("/secrets");  
            });  
        }  
    });  
});
```

در کد زیر معتبر بودن session یا همان authentication را برای دفعات بعدی چک می کند.(در صورت عدم log out یا نبستن صفحه، Session معتبر است)

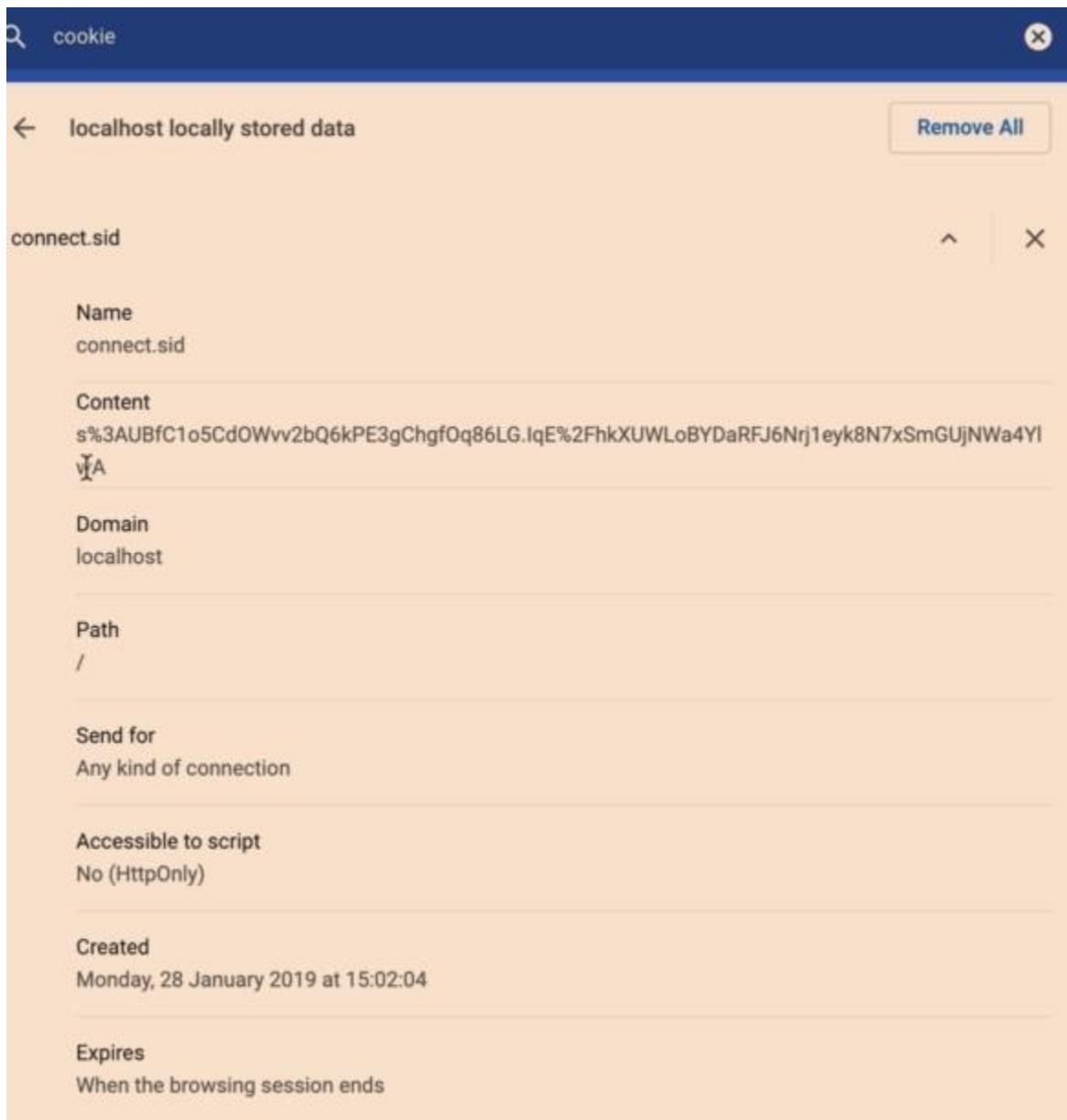
```
app.get("/secrets", function(req, res){  
    if (req.isAuthenticated()){  
        res.render("secrets");  
    } else {  
        res.redirect("/login");  
    }  
});
```

در کد زیر نیز با session ، log out نامعتبر شده و cookie از بین می روید

```
app.get("/logout", function(req, res){  
    req.logout();  
    res.redirect("/");  
});
```

▼ (5) ObjectId("5c4f196b30c91e8d6ba11ce0")	{ 5 fields }	Object
_id	ObjectId("5c4f196b30c91e8d6ba11ce0")	ObjectId
username	user@passportlocalmongoose.com	String
salt	6f42803db1920181b9ea8659d8bddbd1a94a78687170a55b...	String
hash	50415bbc486d528640d7c0a2d154241bb32002884e76803...	String
__v	0	Int32

cookie		
google.co.uk	2 cookies	▶
google.com	6 cookies	▶
krxd.net	1 cookie	▶
licensebuttons.net	1 cookie	▶
localhost	1 cook <b>o</b>	▶
mookie1.com	2 cookies	▶

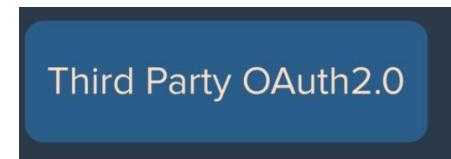


**Key words:** Using Passport.js to Add Cookies and Sessions.

## 12. Level 6 - OAuth 2.0 & How to Implement Sign In with Google

- اگر دقت کرده باشید، ثبت نام سایت ها هم به صورت Register و هم با استفاده از اکانت های دیگر کاربر می باشد.
- در این قسمت نحوه احراز هویت توسط Third Party یا شخص سوم صحبت می شود. در واقع به جای Register کردن به صورت نام و پسورد در Database پروژه، می توان با Third Party احراز هویت

را توسط کمپانی های بزرگ مثل گوگل، فیس بوک و... انجام داد. چون که امنیت این کمپانی ها فوق العاده بالا می باشد، دیگران نگران هک شدن نام و پسورد در Database پروژه نخواهیم.



Register

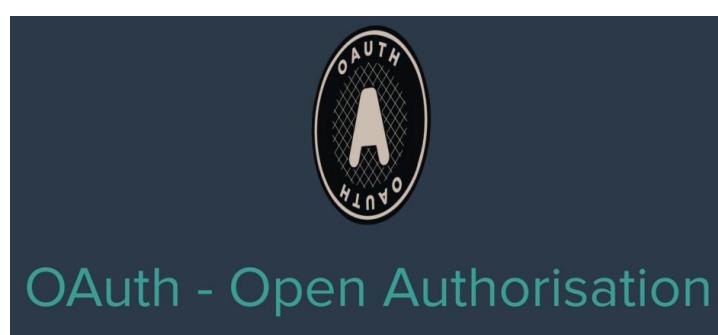
Email

Password

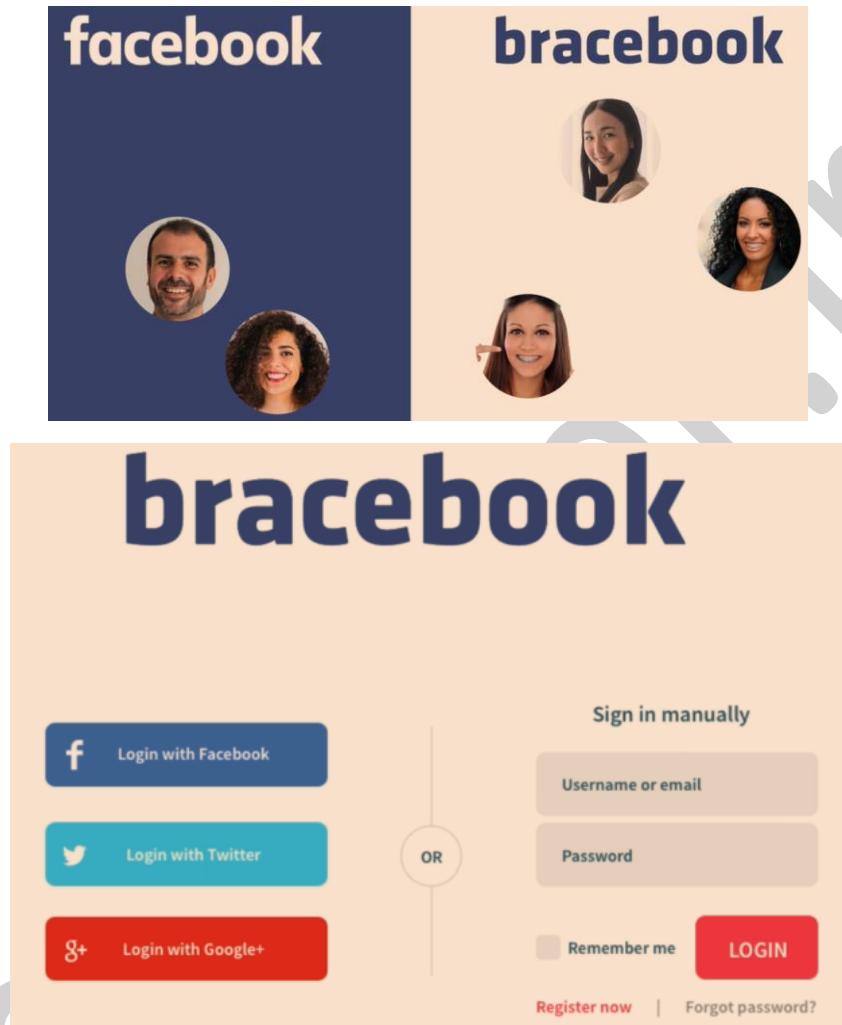
Register

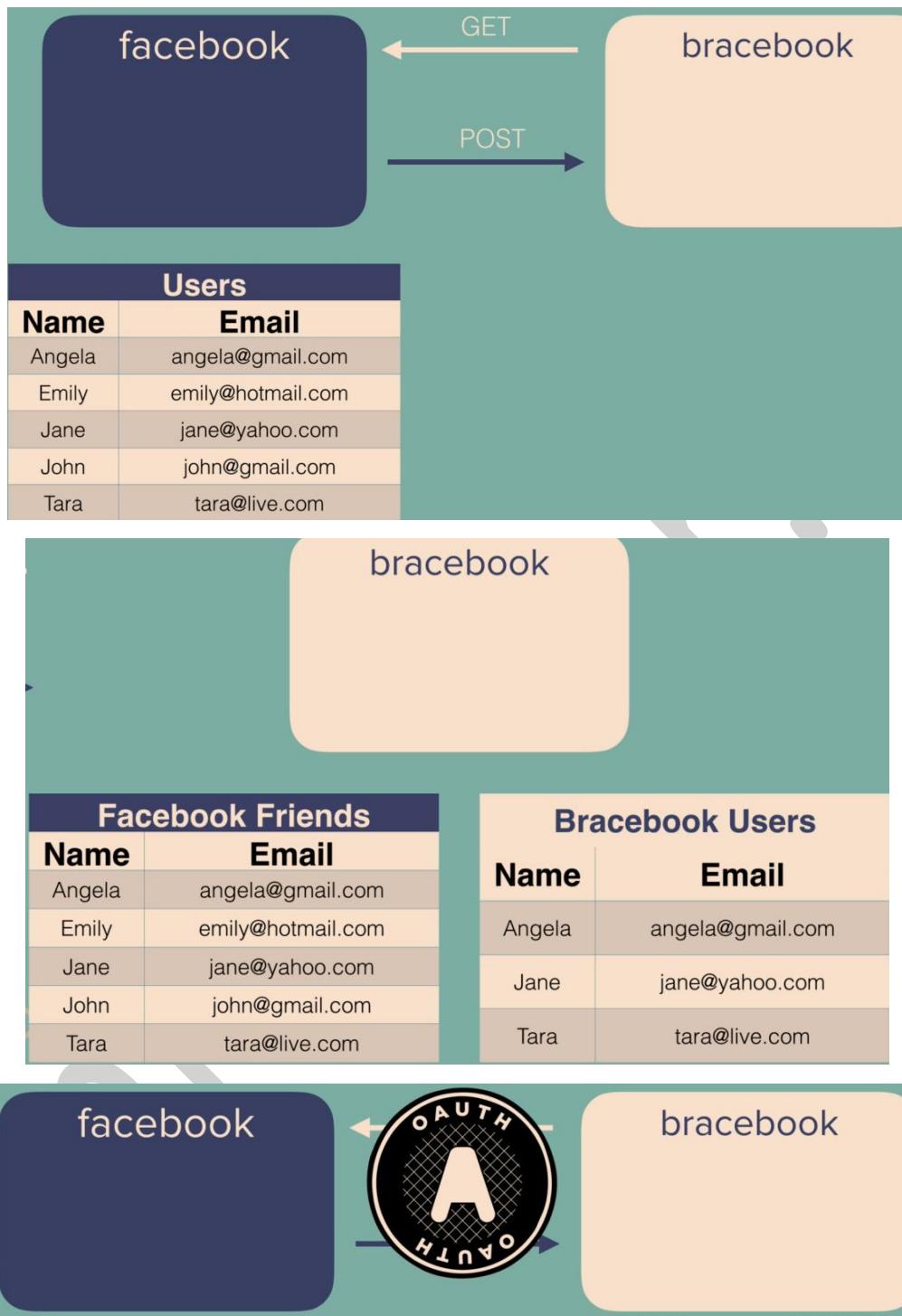
G Sign Up with Google

- برای این منظور باید از استانداردی به نام OAuth استفاده کنیم که یکی از معروف ترین و محبوبترین می باشد.



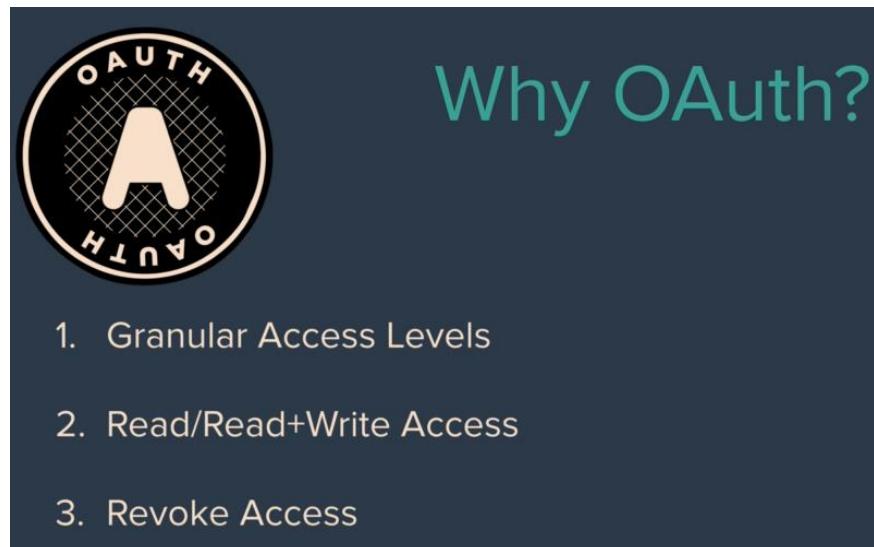
- فرض کنید قرار است سایتی به نام bracebook ایجاد کنید. شما می توانید با OAuth عمل احراز هویت کاربر را انجام داده و همچنین اطلاعات خاص مانند gmail یا لیست دوستان و... را از سایت احراز کننده، مثل facebook دریافت کرد.





- با توجه به شکل زیر، به سه علت OAuth نسب به بقیه محبوب تر است: ۱- قابلیت مشخص کردن اطلاعات دریافتی از کاربر را داریم (مثل، نام، ایمیل، لیست دوستان و...). ۲- می‌توان دسترسی فقط قابلیت خواندن یا قابلیت خواندن و نوشتمن برای اطلاعات کاربر داشت (مثلاً خواند پست فیس بوک یا نوشتمن

پست در فیس بوک کاربر). ۳- کاربر می تواند هر زمانی دسترسی برنامه را قطع کند(مثلاً دیگر آن برنامه نمی تواند اطلاعات بگیرد یا به عبارتی آن برنامه را در مثلاً فیس بوک log out کند).



- در شکل های زیر مراحل احراز هویت توسط OAuth را نشان می دهد.

Step 1 - Set Up Your App

Dashboard

IS\_FEDERATED\_APP\_1

APP ID: 543101382542779 | View Analytics

API Version: v2.7 | App ID: 543101382542779

App Secret: \*\*\*\*

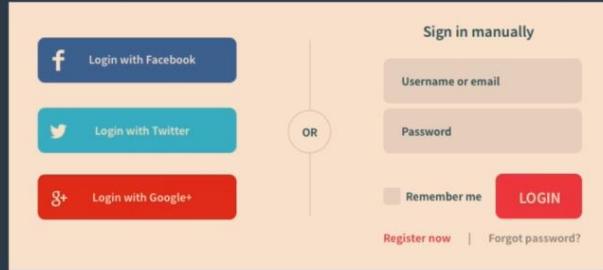
Get Started with the Facebook SDK

Facebook Login

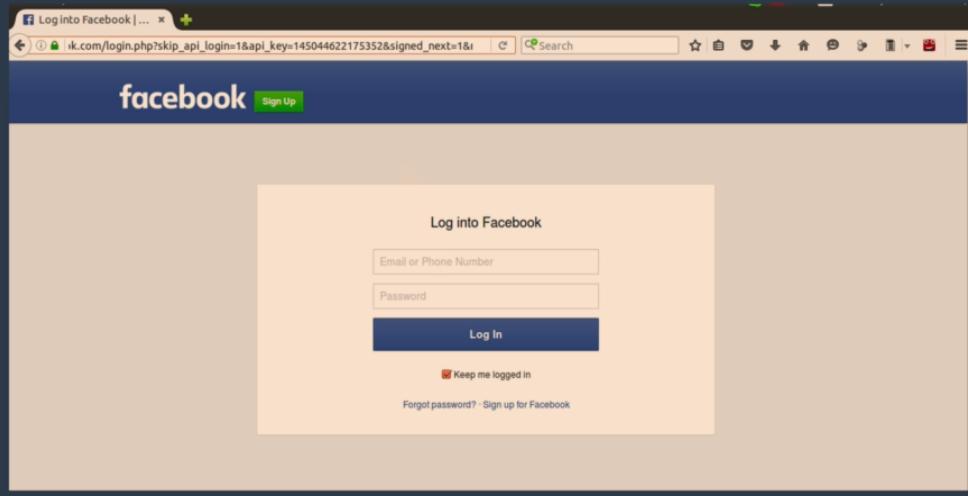
Active Login Users Trend

Monthly Active Users Weekly Active

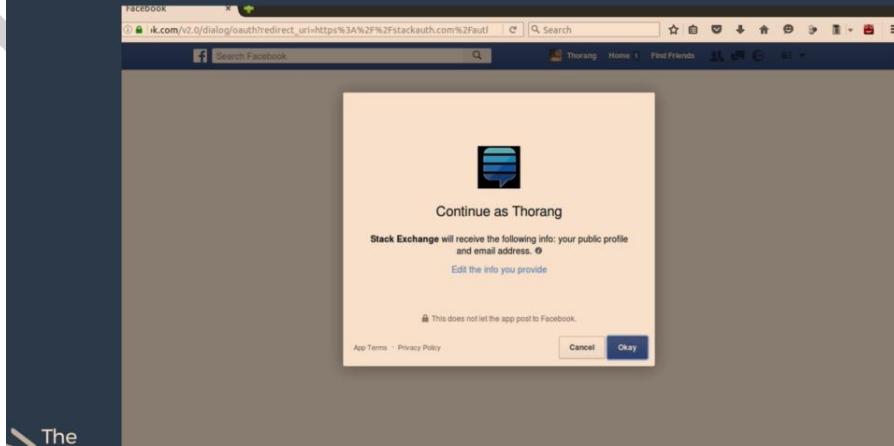
## Step 2 - Redirect to Authenticate



## Step 3 - User Logs In



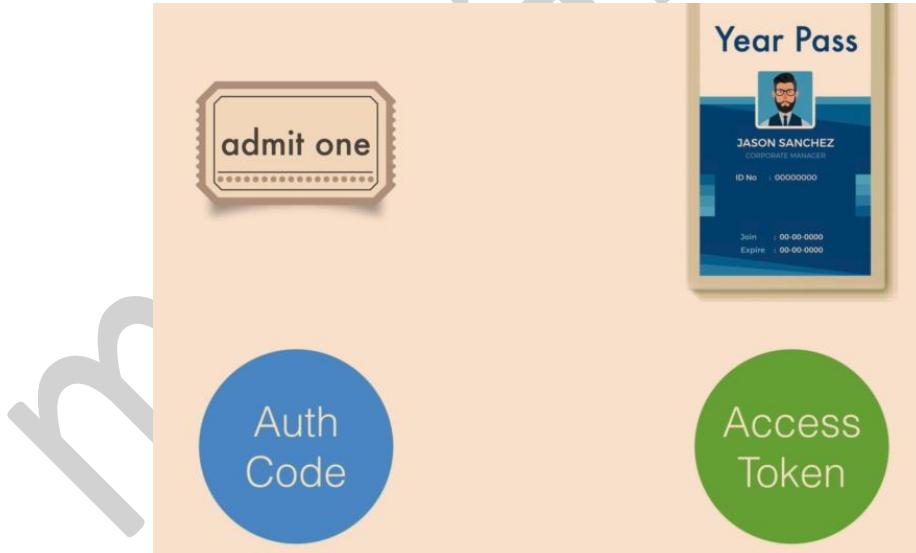
## Step 4 - User Grants Permissions



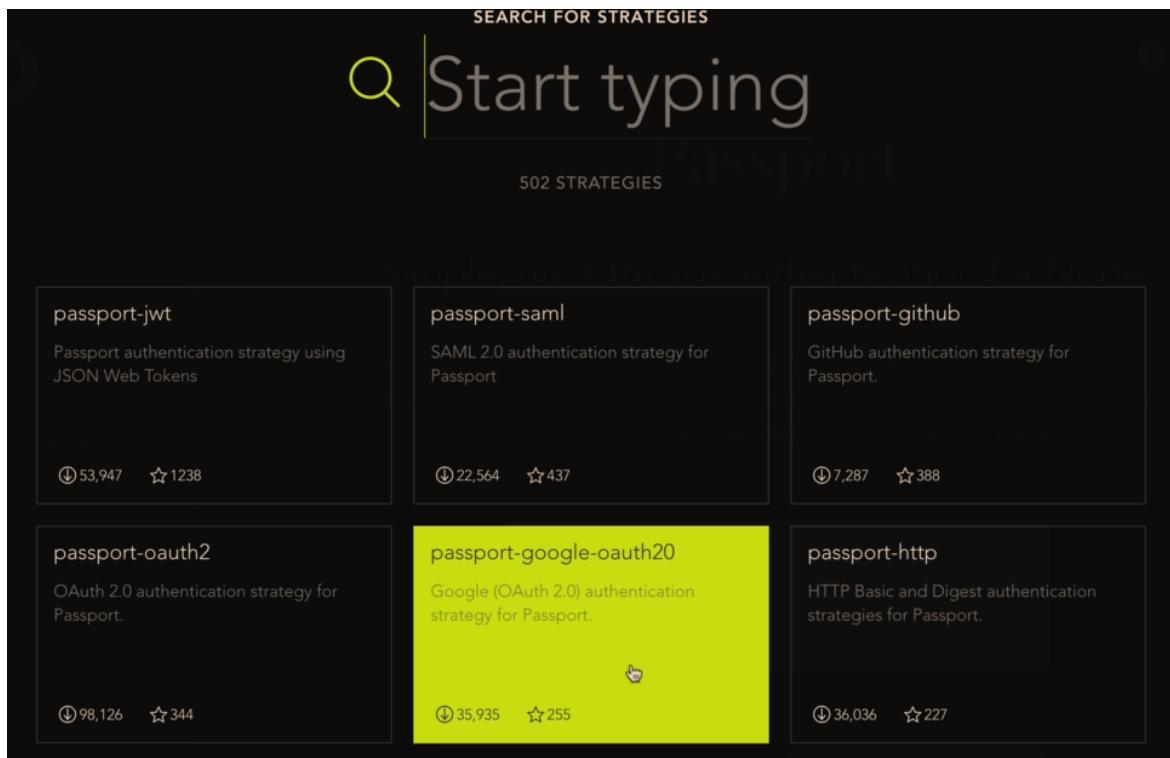
## Step 5 - Receive Authorisation code



## Step 6 - Exchange AuthCode for Access Token



- از افزونه passport-google-oauth20 برای این منظور در پروژه استفاده می کنیم و با توجه به آن پیش می رویم.



## passport-google-oauth20

[build failing](#) [coverage 100%](#) [quality](#) [not found](#) [dependencies up to date](#)

Passport strategy for authenticating with [Google](#) using the OAuth 2.0 API.

This module lets you authenticate using Google in your Node.js applications. By plugging into Passport, Google authentication can be easily and unobtrusively integrated into any application or framework that supports [Connect](#)-style middleware, including [Express](#).

### Install

```
$ npm install passport-google-oauth20
```

- برای پروژه مان باید در google API ثبت نام و کلیدها و اعتبارات و تنظیمات مورد نظر را ثبت کنیم.

<https://console.developers.google.com/apis/credentials/consent?project=secret-230310&supportedpurview=apis>

≡ **Google APIs** ⚙ Secret ▾

**API APIs & Services**

**Credentials**

Scopes allow your application to access your user's private data. [Learn more](#)

If you add a sensitive scope, such as scopes that give you full access to Gmail or Drive, Google will verify your consent screen before it's published.

behav...  
Let us know  
experience.

Dashboard

Library

**Credentials**

email

profile

openid

**Add scope**

**Authorized domains** ⓘ

To protect you and your users, Google only allows applications that authenticate using OAuth to use Authorized Domains. Your applications' links must be hosted on Authorized Domains. [Learn more](#)

example.com

Type in the domain and press Enter to add it

**Application Homepage link**

Shown on the consent screen. Must be hosted on an Authorized Domain.

https:// or http://

**Application Privacy Policy link**

Shown on the consent screen. Must be hosted on an Authorized Domain.

https:// or http://

**Application Terms of Service link (Optional)**

Shown on the consent screen. Must be hosted on an Authorized Domain.

https:// or http://

**Save** **Submit for verification** **Cancel**

## Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

### Application type

- Web application
- Android [Learn more](#)
- Chrome App [Learn more](#)
- iOS [Learn more](#)
- Other

### Name

Secrets

### Restrictions

Enter JavaScript origins, redirect URIs, or both [Learn More](#)

Origins and redirect domains must be added to the list of Authorized Domains in the OAuth consent settings.

#### Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (`https://*.example.com`) or a path (`https://example.com/subdir`). If you're using a nonstandard port, you must include it in the origin URI.

`http://localhost:3000` 

`https://www.example.com` 

Type in the domain and press Enter to add it

#### Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

`http://localhost:3000/auth/google/secrets`

Type in the domain and press Enter to add it

**Create**

**Cancel**



```
app.js          .env
CLIENT_ID=560768197808-mur1p64ibglu85dndtbrf7qhdioa82i6.apps.googleusercontent.com
CLIENT_SECRET=700LrGhqBgWo01XBVBXhG08q
```

The screenshot shows the Google API console interface. The top navigation bar includes "Google APIs", "Secret", and ".env". The main area is titled "Credentials" under the "APIs & Services" tab. It shows a table for "OAuth 2.0 client IDs" with one entry:

Name	Creation date	Type	Client ID
Secrets	Jan 31, 2019	Web application	560768197808-mur1p64ibglu85dndtbrf7qhdioa82i6.apps.googleusercontent.com

```
const GoogleStrategy = require('passport-google-oauth20').Strategy;
const findOrCreate = require('mongoose-findorcreate');
```

```

passport.use(new GoogleStrategy({
  clientID: process.env.CLIENT_ID,
  clientSecret: process.env.CLIENT_SECRET,
  callbackURL: "http://localhost:3000/auth/google/secrets",
  userProfileURL: "https://www.googleapis.com/oauth2/v3 userinfo"
},
function(accessToken, refreshToken, profile, cb) {
  console.log(profile);

  User.findOrCreate({ googleId: profile.id }, function (err, user) {
    return cb(err, user);
  });
}
));

```

```

userSchema.plugin(passportLocalMongoose);
userSchema.plugin(findOrCreate);

```

```

<div class="col-sm-4">
  <div class="card">
    <div class="card-body">
      <a class="btn btn-block btn-social btn-google" href="/auth/google" role="button">
        <i class="fab fa-google"></i>
        Sign Up with Google
      </a>
    </div>
  </div>
</div>

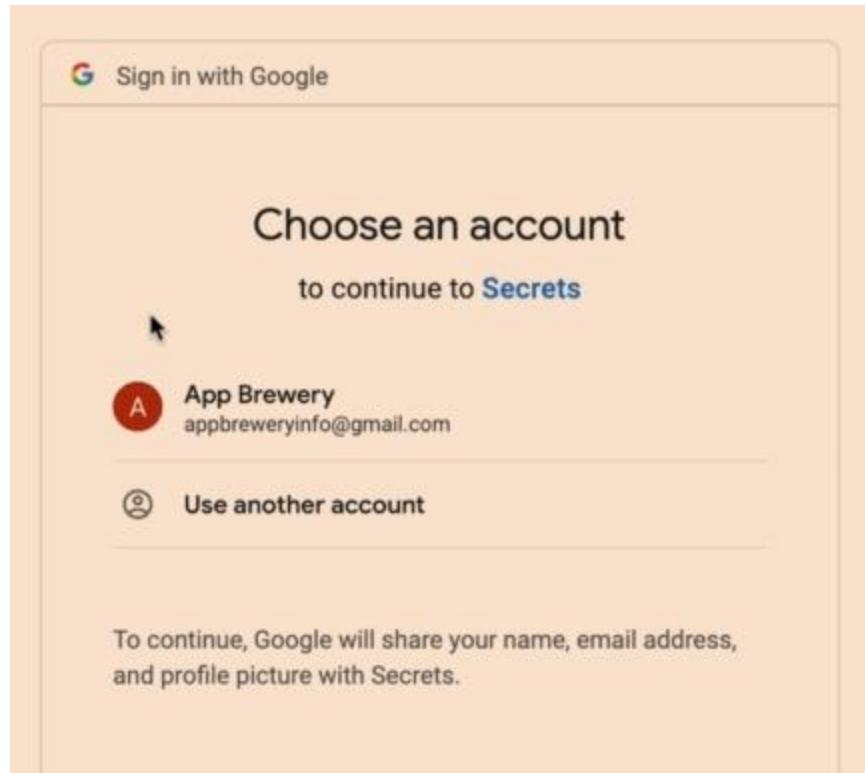
```

- کد زیر کاربر را به صفحه احراز هویت google هدایت می کند که مختص پروژه مان است.

```

app.get("/auth/google",
  passport.authenticate('google', { scope: ["profile"] })
);

```



```
app.get("/auth/google/secrets",
  passport.authenticate('google', { failureRedirect: "/login" }),
  function(req, res) {
    // Successful authentication, redirect to secrets.
    res.redirect("/secrets");
});
```

```
passport.serializeUser(function(user, done) {
  done(null, user.id);
});

passport.deserializeUser(function(id, done) {
  User.findById(id, function(err, user) {
    done(err, user);
  });
});
```

```

const userSchema = new mongoose.Schema ({
  email: String,
  password: String,
  googleId: String
});

```

- در نتیجه در Database فقط Googleid را ذخیره می کنیم و نیازی به نگهداری نام کاربری و پسورد نیست (البته اختیاری است). این googleid مختص اکانت کاربر در سایت ما می باشد. یعنی اگر کاربر از هر سیستمی با گوگل در سایت ما وارد شود، google این id را برای ما ارسال می کند و ما متوجه می شویم که کدام کاربر وارد شده است.

6 ObjectId("5c52d485157fb5abec4592f5")	{ 3 fields }	Object
_id	ObjectId("5c52d485157fb5abec4592f5")	ObjectId
googleId	110269538995256656990	String
__v	0	Int32

- برای قالب های button مختص احراز هویت مثل گوگل، فیسبوک وغیر، از قالب زیر استفاده کنید.

The screenshot shows the README page of a GitHub repository titled "Social Buttons for Bootstrap". The page has a red header with the title "Social Buttons for Bootstrap" and a subtitle "Social Sign-In Buttons made in pure CSS based on Bootstrap and Font Awesome!". Below the header are two buttons: "View on GitHub" and "Download". The repository structure on the left includes "css" (bootstrap-social.css, styles.css), "views", and ".DS\_Store". The code editor at the bottom shows the HTML and CSS for a "Sign Up with Google" button.

The screenshot shows a "Sign Up with Google" button. The button is red with a white "G" icon from Google and the text "Sign Up with Google". A cursor is hovering over the button, indicating it is interactive.

**Key words:** Level 6 , OAuth 2.0 & How to Implement Sign In with Google.

### 13. Finishing Up the App - Letting Users Submit Secrets

- در این قسمت امکان ارائه Secret یا همان متن برای کاربر معتبر فراهم می شود که در ذخیره می شود.
- همچنین امکان مشاهده همه Secrets برای همه کاربران امکان پذیر خواهد بود.

The image shows two screenshots of a web application interface. The top screenshot displays a form with a large key icon and the word 'Secrets'. Below the form is a button that says 'Don't keep your secrets, share them anonymously!'. The bottom screenshot shows the result of a submission, featuring a key icon and the message 'You've Discovered My Secret!'. It lists two secrets: 'My favourite colour is blue.' and 'Jack Bauer is my hero.'. At the bottom are 'Log Out' and 'Submit a Secret' buttons. A code block at the bottom shows the Node.js code for the '/submit' route.

```
app.get("/submit", function(req, res){  
  if (req.isAuthenticated()){  
    res.render("submit");  
  } else {  
    res.redirect("/login");  
  }  
});
```

```
const userSchema = new mongoose.Schema ({
  email: String,
  password: String,
  googleId: String,
  secret: String
});
```

```
app.post("/submit", function(req, res){
  const submittedSecret = req.body.secret;

  //Once the user is authenticated and their session gets
  // console.log(req.user.id);

  User.findById(req.user.id, function(err, foundUser){
    if (err) {
      console.log(err);
    } else {
      if (foundUser) {
        foundUser.secret = submittedSecret;
        foundUser.save(function(){
          res.redirect("/secrets");
        });
      }
    }
  });
});
```

```
app.get("/secrets", function(req, res){
  User.find({secret: {$ne: null}}, function(err, foundUsers){
    if (err){
      console.log(err);
    } else {
      if (foundUsers) {
        res.render("secrets", {usersWithSecrets: foundUsers});
      }
    }
  });
});
```

```
<% usersWithSecrets.forEach(function(user){ %>
|   <p class="secret-text"><%=user.secret%></p>
<% }) %>
```

**Key words:** Finishing Up the App, Letting Users Submit Secrets.

### 15. Tip from Angela - How to Work as a Freelancer

برای فریلنسر شدن، در ابتدا باید رزومه خود را قوی کرده و نمونه کارهای خوبی درست کنید تا اعتبار و شهرت بدست بیاورید. برای شروع می توانید از سایت های فریلنسری رایگان یا با دست مزد کم مثل Fiverr شروع کنید. در ادامه بعد از گذشت یک سال یا بیشتر (بسته به خودتان) با مشتری های راضی، دارای اعتبار خواهید شد. در نهایت می توانید وارد دنیای حرفه ای ها شده و در سایت هایی فریلنسری مطرح مانند Upwork, Freelancer, Guru مشغول به کار شوید.

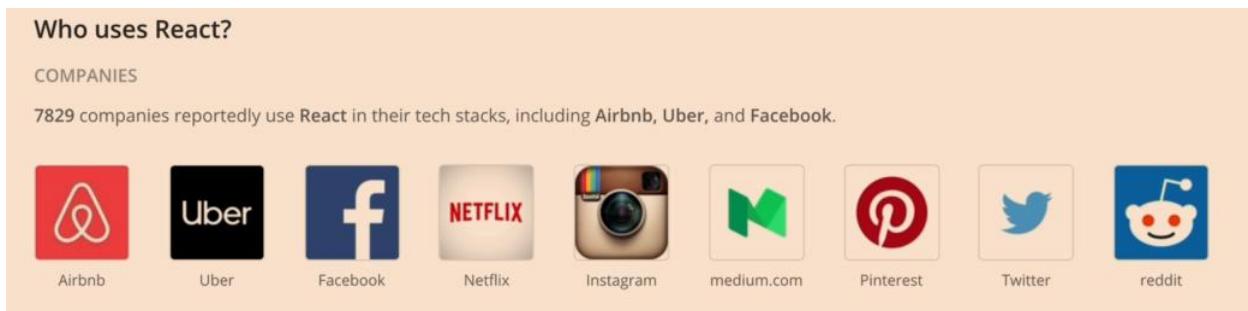
## 33. React.js

### 1. What is React

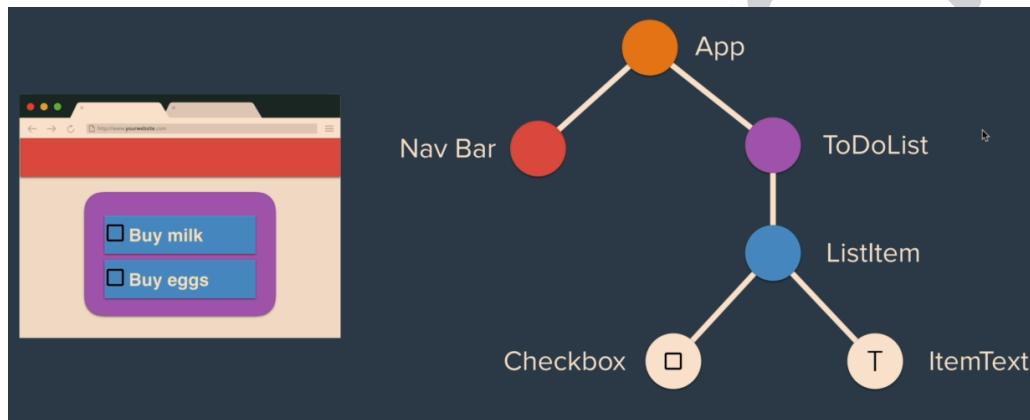
یک framework React - javascript محبوب و پر تقاضای می باشد که برای ساخت UI استفاده می شود و قابلیت ها و راحتی های زیادی در ایجاد UI فراهم می کند.







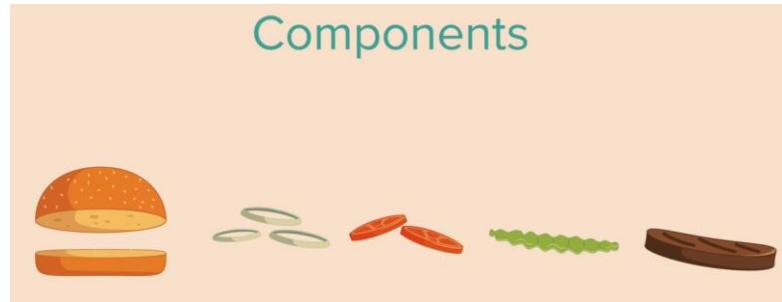
- در از React از component tree استفاده می کنیم و می توان کل اجزا یا components یک سایت را به شکل زیر تحلیل کرد.



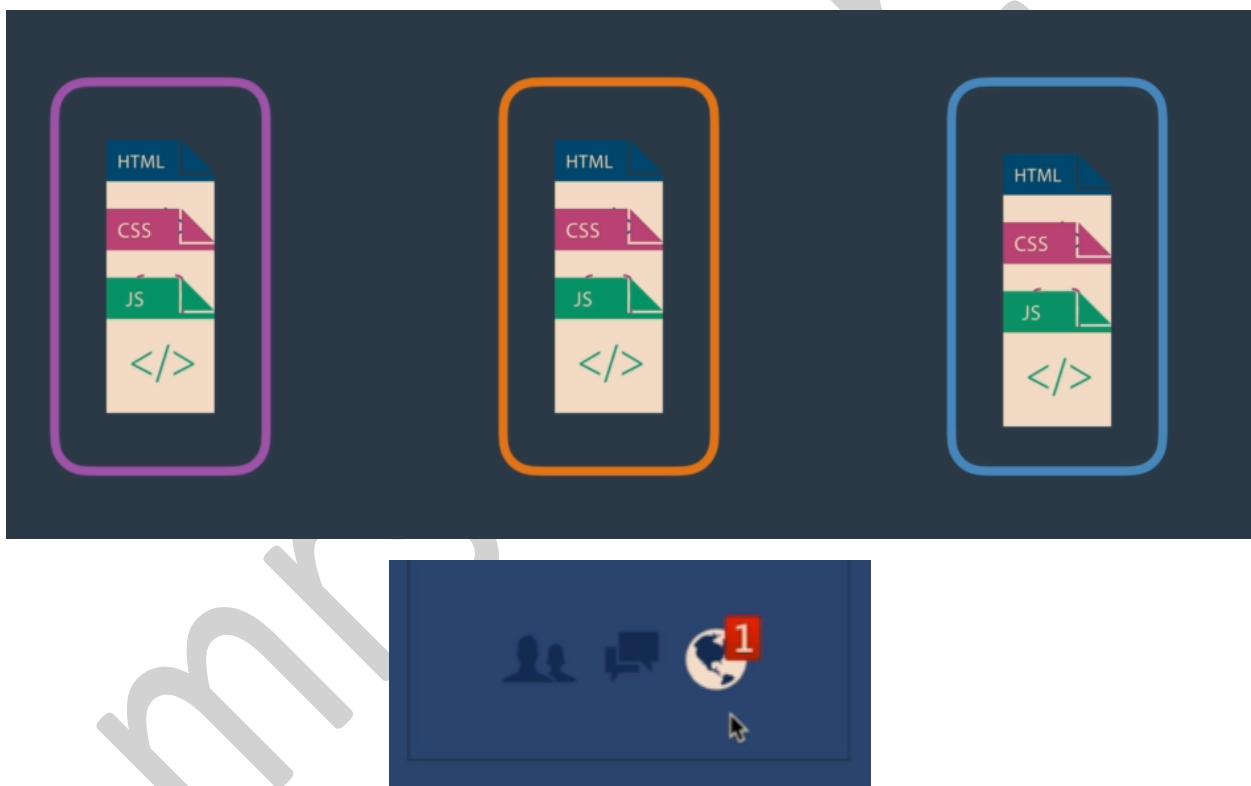
- می توان HTML کدهای اختصاصی ایجاد کرد، و در نتیجه باعث مازوچیتی و clean code شدن پروژه می شود.

```
<body>
  <MyHeader />
  <PageContent />
  <MyFooter />
</body>
```

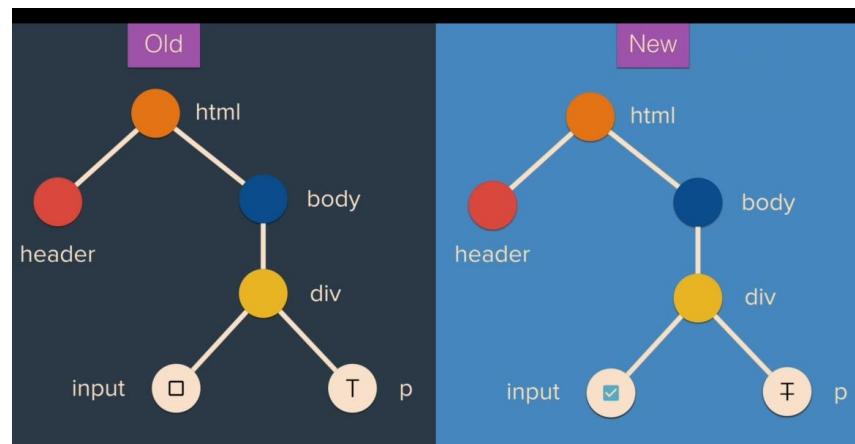
- می توان components را از هم جدا و مجدداً روی هم سوار کرد.



- هر component دارای HTML, CSS, JS جداگانه برای خود دارد در نتیجه عملکردی مستقل نسبت به سایر component می‌تواند داشته باشد. مثلاً هر component به طور مستقل با سرور ارتباط داشته باشد مثل دریافت نوتیفیکیشن در لحظه در اینستاگرام و فیس بوک.(در حالی که قبلاً برای دریافت اطلاعات جدید باید سایت را Refresh می‌کردیم).



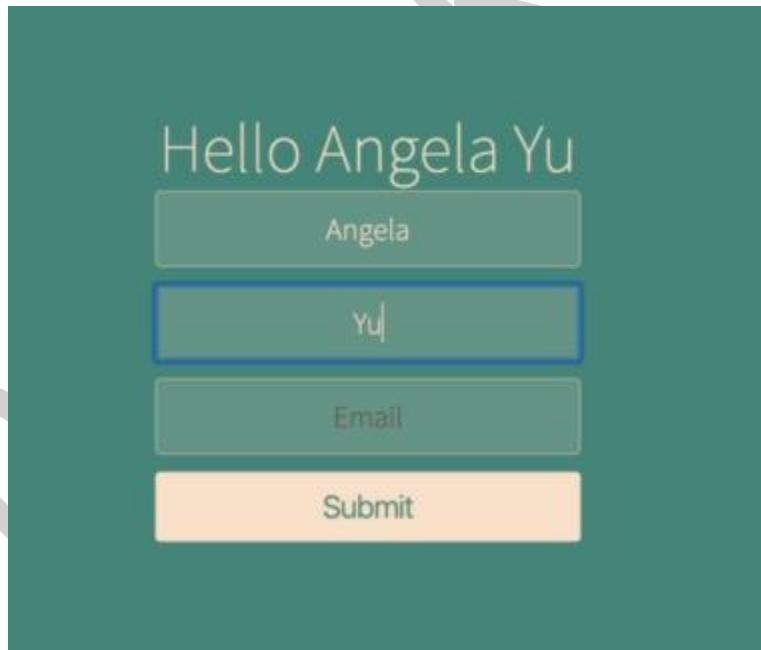
- قابلیت مقایسه تغییرات اعمال شده نسبت به حالت فعلی و قبلی در component را دارد. به این ترتیب فقط آن component را تغییر می‌دهد و کل سایت را تغییر نمی‌دهد. در نتیجه بارگذاری مجدد اطلاعات تکراری نخواهیم داشت و سرعت کار بالاتر می‌رود.



**Key words:** what is React, React Features.

## 2. What we will make in this React module

- در این قسمت پروژه هایی که قرار است در این فصل توسط React پیاده سازی شود را نشان می دهد.



# emojipedia



## Tense Biceps

"You can do that!" or "I feel strong!" Arm with tense biceps. Also used in connection with doing sports, e.g. at the gym.



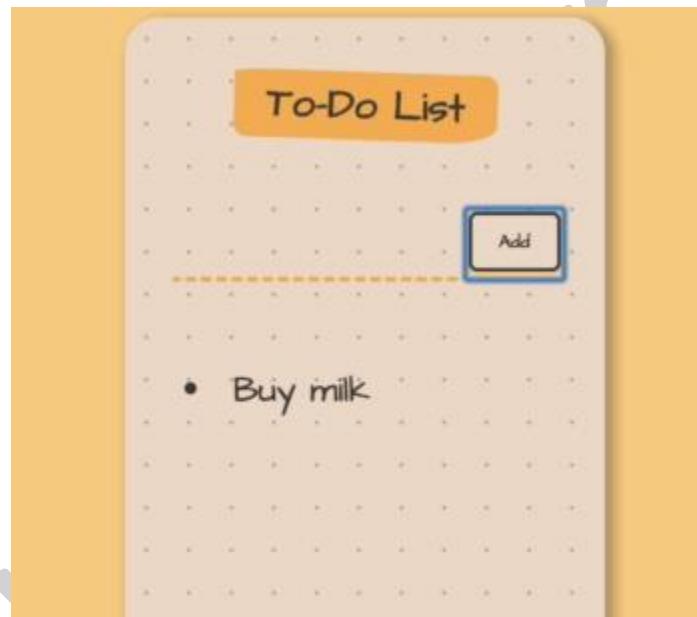
## Person With Folded Hands

Two hands pressed together. Is currently very introverted, saying a prayer, or hoping for enlightenment. Is also used as a "high five" or to say thank you.



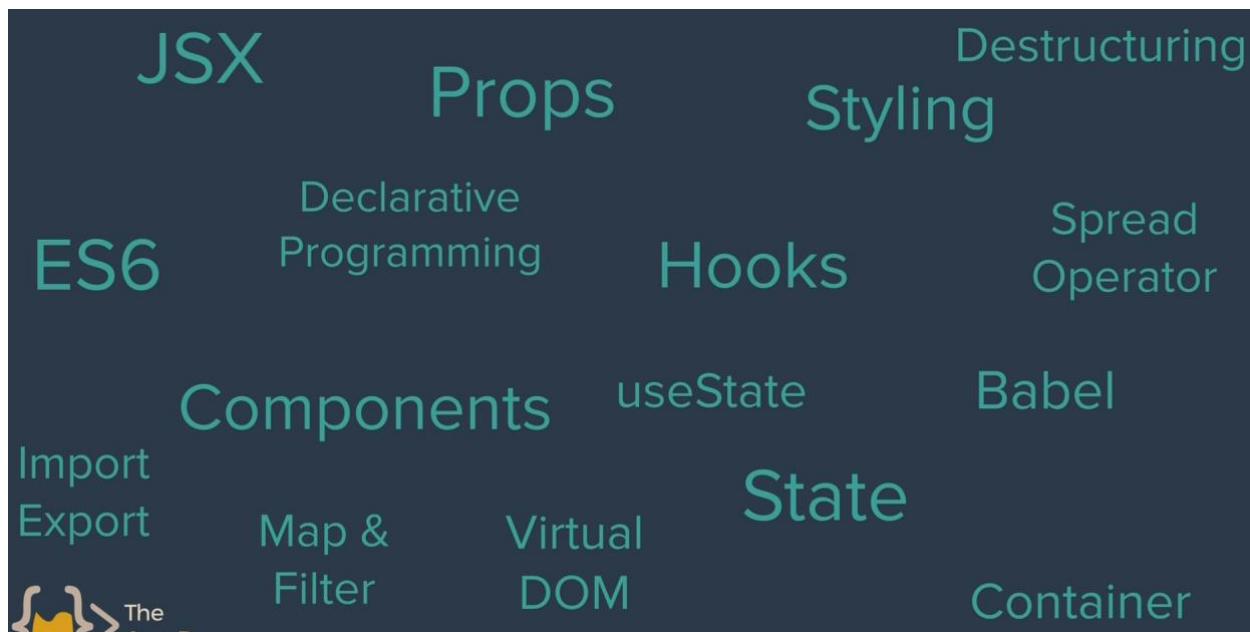
## Rolling On The Floor, Laughing

This is funny! A smiley face, rolling on the floor, laughing. The face is laughing boundlessly. The emoji version of "rofl". Stands for „rolling on the floor, laughing".



The screenshot shows the Google Keep application interface. At the top, there's a navigation bar with a menu icon, a lightbulb icon labeled 'Keep', and a search bar. Below the navigation bar, there are two main categories: 'Notes' (which is highlighted in yellow) and 'Reminders'. Under 'Notes', there's a note card with the text 'Hello' and 'This is a new note.' To the left of the note card, there's a section for 'LABELS' with an 'Edit labels' button. At the bottom, there are buttons for 'Archive' and other navigation options.

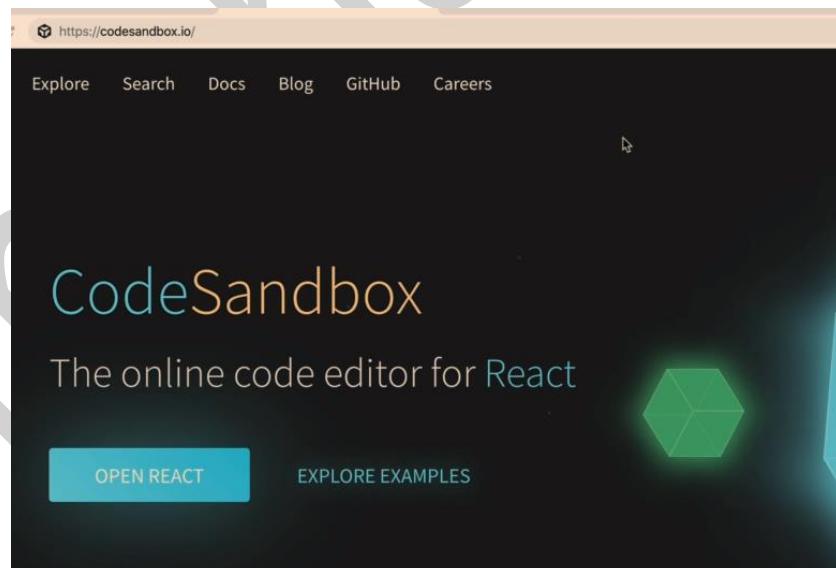
- در این فصل برای کار با React با مفاهیم و ابزارهای زیر آشنا خواهیم شد.



**Key words:** What we will make in this React module.

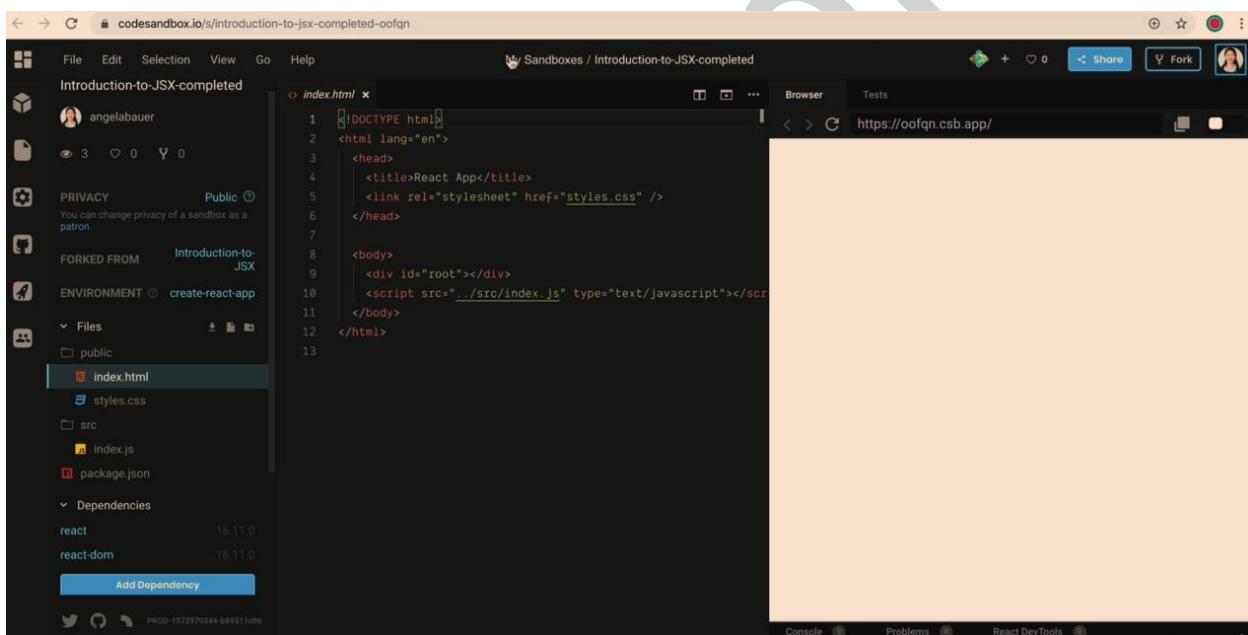
### 3. Introduction to Code Sandbox and the Structure of the Module

در این قسمت به معرفی سایت [codeSandbox](https://codesandbox.io/) می پردازد که قرار است پروژه های React را در این محیط کد زده و تست کنیم.



قابلیت نصب افزونه و npm package نیز وجود دارد و قابلیت ها فراوانی دارد. در نتیجه کارهای تکراری و اولیه برای نصب پکیج ها و غیره برای هر پروژه را راحت می کند. (در انتهای فصل نحوه پیکربندی برای استفاده در VS Code یا Atom را نشان می دهیم).

 Export to GitHub	 Static File Hosting	 Integrated DevTools	 patron
All sandboxes can easily be exported to a GitHub repository.	The development server will serve all files statically from the public folder, depending on the template.	The preview window has integrated DevTools, like a console, test view and a problem viewer.	You can set a sandbox to private or unlisted to make sure others cannot see or find it.
 Externally Hosted Previews	 Monaco Editor	 Hot Module Reloading	 Error Overlay
You can open your sandbox preview with a separate URL, while still keeping Hot Module Reloading.	We use the same editor as VSCode, which gives us "Go to Definition", "Replace Occurrences" and more!	Hot Module Reloading is built in, so you won't have to press refresh for every change.	We show a user friendly error overlay for every error, sometimes with suggestions on how to solve it.
 Automatic Type Acquisition	 TypeScript	 Prettier	 ESLint
typings are automatically downloaded for every dependency, so you always have autocompletions.	Thanks to Monaco we show TypeScript autocompletions and diagnostics for TS sandboxes.	Code automatically gets prettified on save according to your own Prettier preferences.	All code is linted automatically using latest version of ESLint, with full ES6 support.



The screenshot shows the CodeSandbox web interface. On the left, there's a sidebar with project details: 'Introduction-to-JSX-completed' by 'angelabauer', 'Public', 'FORKED FROM Introduction-to-JSX', and 'ENVIRONMENT create-react-app'. Below this are sections for 'Files' (with 'index.html' selected), 'Dependencies' (listing 'react' and 'react-dom'), and a footer with social icons and a PR ID. The main area shows the file structure and code for 'index.html':

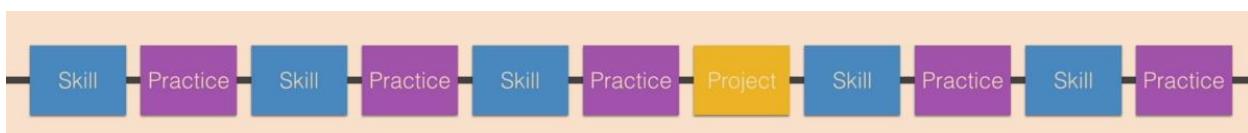
```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <div id="root"></div>
    <script src="src/index.js" type="text/javascript"></scr
  </body>
</html>

```

To the right, a browser window shows the live preview at <https://oofqn.csb.app/>. The browser toolbar includes 'Share', 'Fork', and user profile icons.

- در شکل زیر روش یادگیری و تمرین این فصل را نشانی می دهد. ابتدا مفهوم را در یک **sandbox** یاد می گیریم و در **sandBox** دیگر آن را تمرین می کنیم.



**Key words:** Introduction to Code Sandbox and the Structure of the Module.

## 4. Introduction to JSX and Babel

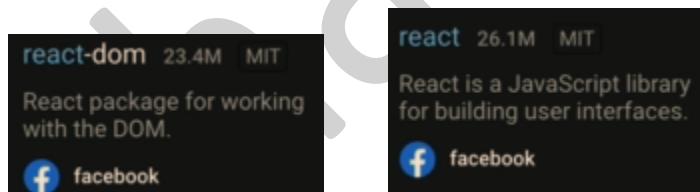
### JSX چیست؟

افزونه‌ای برای ری اکت است که به شما کمک می‌کند کدها را مانند کد HTML بنویسید. قطعه کد زیر را ببینید:

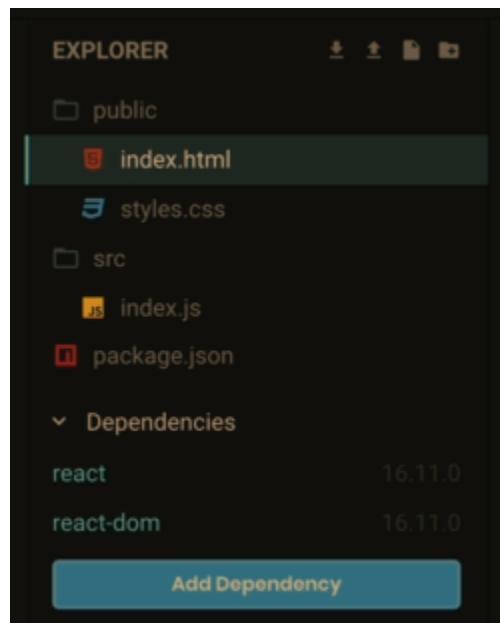
```
const element = <h1>Hello, world!</h1>;
```

این خط، یک کد HTML نیست بلکه سینتکسی متفاوت و مبتنی بر HTML/XML از زبان جاوا اسکریپت دارد و به همراه کدهای JSX استفاده می‌شود. JSX یک قالب یا فریمورک جدید نیست بلکه همان زبان جاوا اسکریپت و فقط با ساختار دستوری دیگری است که در ایجاد کامپوننت‌ها مختلف برای React DOM (Element) و المان‌های (Component) کاربرد دارد.

- در واقع JSX این قابلیت را می‌دهد که تگ‌های HTML را مستقیماً در JavaScript استفاده کنیم.



- در شکل زیر متریال‌های موجود برای کد نویسی به کمک React را نشان می‌دهد.



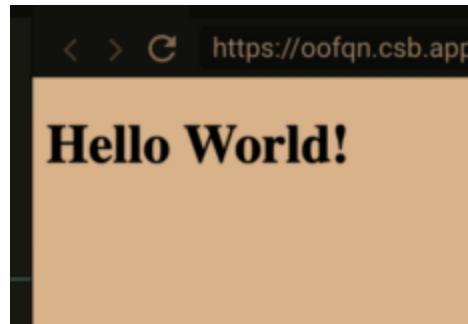
```
<body>
  <div id="root"></div>
  <script src="../../src/index.js" type="text/javascript"></script>
</body>
</html>
```

- در کد زیر نیاز استفاده از ReactDom و React را نشان می دهد.

```
1 var React = require("react");
2 var ReactDOM = require("react-dom");
3
4 ReactDOM.render(WHAT TO SHOW, WHERE TO SHOW IT);
```

- قابلیت دسترسی به Dom را برای React فراهم می کند.

```
index.html    js index.js
1 var React = require("react");
2 var ReactDOM = require("react-dom");
3
4 ReactDOM.render(<h1>Hello World!</h1>, document.getElementById("root"));
```



- در React، کامپایلری به نام Babel وجود دارد. کار این کامپایلر، تبدیل کردن نسخه های از ES6 و بعد از آن، به نسخه خوانا و سبک جاوا اسکریپت، یعنی ES5 می باشد که قابلیت اجرا در همه مرورگرها و حتی مرورگرهای قدیمی را نیز دارد. Babel همیچنین وظیفه تشخیص JSX را نیز دارد و آن را به نسخه ساده Javascript تبدیل می کند. ( اکثر مرورگرها از ES5 که خیلی شبیه به ES3 یا javascript vanilla می باشد پشتیبانی می کنند. از نسخه ES6 تغییرات محسوسی در javascript vanilla اعمال شد).
- جاوا اسکریپت نام نسخه تجاری ES می باشد. در حالت کلی جاوا اسکریپت عمومی و سبک همان نسخه ES5 می باشد.

Put in next-gen JavaScript	Get browser-compatible JavaScript out
<pre>var obj = {   shorthand,   method() {     return "☀️";   } };</pre>	<pre>var obj = {   shorthand: shorthand,   method: function method() {     return "☀️";   } };</pre>

- در کد زیر hello world ۲، یکی با استفاده از React و ES6 و JSX و دیگری بدون آنها را نشان می دهد. سادگی و راحتی مورد اول مشهود است.

The screenshot shows a code editor with two tabs: 'index.html' and 'js index.js'. The 'js index.js' tab contains the following code:

```
1 var React = require("react");
2 var ReactDOM = require("react-dom");
3
4 ReactDOM.render(<h1>Hello World!</h1>, document.getElementById("root"));
5
6 var h1 = document.createElement("h1");
7 h1.innerHTML = "Hello World!";
8 document.getElementById("root").appendChild(h1);
```

To the right of the code editor is a browser window displaying the output. The browser title bar says 'Browser'. The address bar shows 'https://oofqr'. The page content is 'Hello World!' repeated twice.

- در کد از دستور Import در ES6 استفاده کردیم. همچنین توسط div المنش ها را افرایش دادیم.

The screenshot shows a code editor with two tabs: 'index.html' and 'js index.js'. The 'js index.js' tab contains the following code:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 ReactDOM.render(
5   <div>
6     <h1>Hello World!</h1>
7     <p>This is a paragraph.</p>
8   </div>,
9   document.getElementById("root")
10 );
11
```

To the right of the code editor is a browser window displaying the output. The browser title bar says 'Browser'. The address bar shows 'https://oofqr'. The page content is 'Hello World!' followed by 'This is a paragraph.'

**Key words:** React, JSX, Babel, ES6 and ES5.

## 5. JSX Code Practice

در این قسمت تمرين مورد نظر را حل کردیم.

```
import React from "react";
import ReactDOM from "react-dom";

ReactDOM.render(
  <div>
    <h1>My Favorite routine</h1>
    <ul>
      <li> coffee </li>
      <li> reading book </li>
      <li> coding </li>
    </ul>
  </div>,
  document.getElementById("root")
);
```



**Key words:** using JSX concept in practice.

## 6. Javascript Expressions in JSX & ES6 Template Literals

In this lesson you will know how to add expression with curly braces in HTML Tag in ReacDom.



```
import React from "react";
import ReactDOM from "react-dom";

const fName = "Angela";
const lName = "Yu";
const num = 7;

ReactDOM.render(
  <div>
    <h1>Hello {fName + " " + lName}!</h1>
    <p>Your lucky number is {num}</p>
  </div>,
  document.getElementById("root")
);
```

Hello Angela Yu!

Your lucky number is 7

expression ↗

vs.

statement ➤ ➤ ➤ ➤ ➤

**Key words:** difference between expression and statements, ES6 Template.

## 7. Javascript Expressions in JSX Practice

In this lesson, you will code below codes:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 const name = "Angela";
5 const currentDate = new Date();
6 const year = currentDate.getFullYear();
7
8 ReactDOM.render(
9   <div>
10     <p>Created by {name}</p>
11     <p>Copyright {year}</p>
12   </div>,
13   document.getElementById("root")
14 );
15
```

Created by Angela  
Copyright 2023

**Key words:** JSX Code practice.

## 8. JSX Attributes & Styling React Elements

In this lesson, you will learn how to use HTML attributes in JSX and styling element with CSS.

```
import React from "react";
import ReactDOM from "react-dom";

const img = "https://picsum.photos/200";

ReactDOM.render(
  <div>
    <h1 className="heading">My Favourite Foods</h1>
    <img alt="random" src={img + "?grayscale"}/>
    
    
    
  </div>,
  document.getElementById("root")
);
```

https://kkyer.csb.app/

# My Favourite Foods

A collage of three food images. The top image shows a stack of bacon strips. The middle image shows a slice of ham (jamon). The bottom image shows a bowl of noodle soup with garnishes.

## 9. Inline Styling for React Elements

In this lesson you will learn how to use inline styling instead class styling.

A screenshot of a code editor and a browser window. The code editor shows a file named `index.js` with the following content:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3
4 const customStyle = {
5   color: "red",
6   fontSize: "20px",
7   border: "1px solid black"
8 };
9
10 customStyle.color = "blue";
11
12 ReactDOM.render(
13   <h1 style={customStyle}>Hello World!</h1>,
14   document.getElementById("root")
15 );
```

The browser window shows a single `<h1>` element with the text "Hello World!" in blue font, indicating that the CSS styles have been applied.

## 10. React Styling Practice

In this lesson, we apply the style for heading with inline and css ways.

A screenshot of a code editor and a browser window. The code editor shows a file named `index.js` with the following content:

```
import React from "react";
import ReactDOM from "react-dom";

const date = new Date();
const currentTime = date.getHours();

let greeting;

const customStyle = {
  color: ""
};

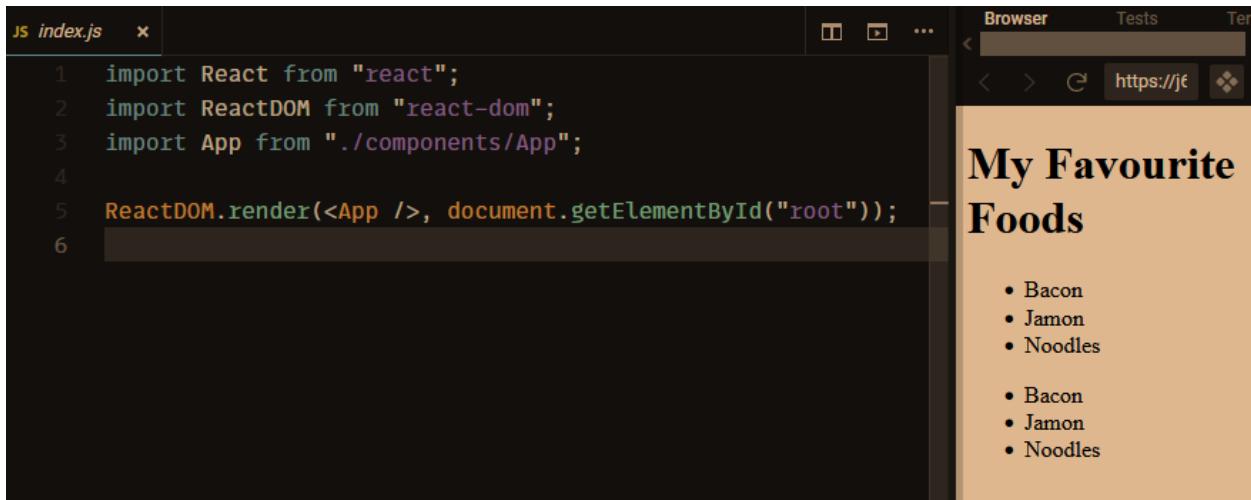
if (currentTime < 12) {
  greeting = "Good Morning";
  customStyle.color = "red";
} else if (currentTime < 18) {
  greeting = "Good Afternoon";
  customStyle.color = "green";
} else {
  greeting = "Good Night";
  customStyle.color = "blue";
}

ReactDOM.render(
  <h1 className="heading" style={customStyle}>
    {greeting}
  </h1>,
  document.getElementById("root")
);
```

The browser window shows a single `<h1>` element with the text "Good Afternoon" in green font, indicating that the CSS styles have been applied.

## 11. React Components

In this lesson, you will know how to make React component like HTML elements and how to modular your component with import and export.



The screenshot shows a code editor with an `index.js` file and a browser window. The code in `index.js` is:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import App from "./components/App";
4
5 ReactDOM.render(<App />, document.getElementById("root"));
6
```

The browser window displays the title "My Favourite Foods" and a list of items:

- Bacon
- Jamon
- Noodles

This demonstrates a simple React application where the `App` component is being rendered into the `root` element.

## 12. React Components Practice

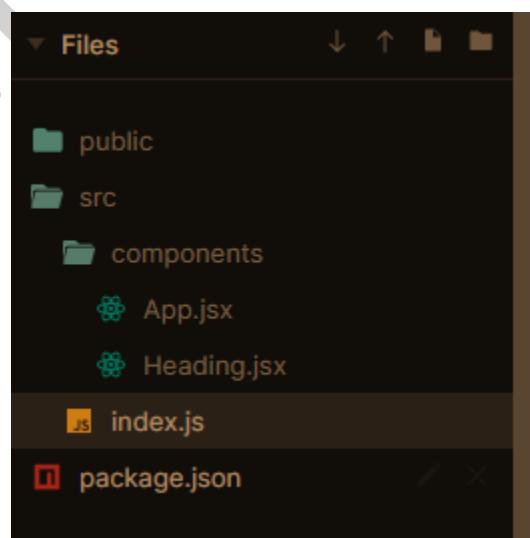
In this practice, you will create a modular react with `App.jsx`



The screenshot shows a code editor with an `index.js` file and a browser window. The code in `index.js` is:

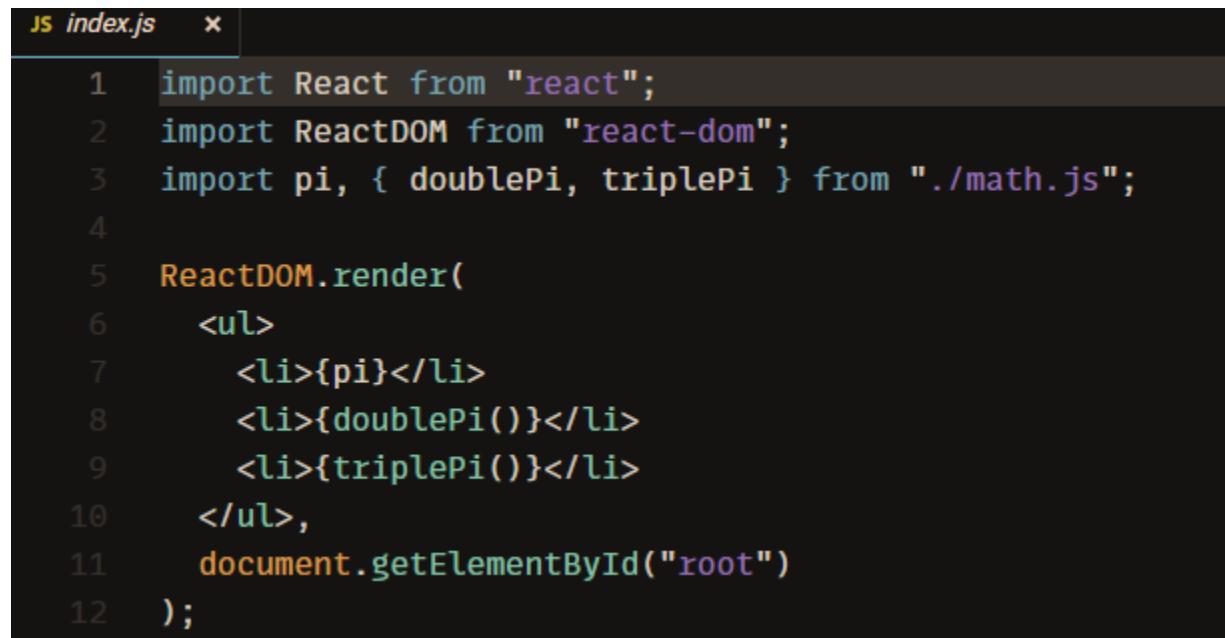
```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import App from "./components/App";
4
5 ReactDOM.render(<App />, document.getElementById("root"));
6
```

The browser window displays the text "Good Night" in a large blue font.



### 13. Javascript ES6 - Import, Export and Modules

In this lesson you will learn how to import and export in ES6.



The screenshot shows a code editor window with a dark theme. The file is named 'index.js'. The code imports React and ReactDOM from their respective modules, and then imports pi, doublePi, and triplePi from a module named 'math.js'. It then uses ReactDOM.render to render an 

 element containing three - elements with the values of pi, doublePi(), and triplePi() respectively, into a DOM element with id 'root'.

```
js index.js x
1 import React from "react";
2 import ReactDOM from "react-dom";
3 import pi, { doublePi, triplePi } from "./math.js";
4
5 ReactDOM.render(
6   <ul>
7     <li>{pi}</li>
8     <li>{doublePi()}</li>
9     <li>{triplePi()}</li>
10   </ul>,
11   document.getElementById("root")
12 );
```

### 14. Javascript ES6 Import, Export and Modules Practice

It's just a practice to import and export.

### 15. [Windows] Local Environment Setup for React Development

In this lesson you will learn how to config Reac for windows.

### 16. [Mac] Local Environment Setup for React Development

### 17. Keeper App Project - Part 1 Challenge

You have a challage to make the Keeper app.

# Keeper

This is the note title  
This is the note content

## 18. Keeper App Part 1 Solution

It's the solution.

## 19. React Props

In this lesson you will learn about React Props and how to use them in components.

## HTML Attributes

```
document.getElementById("email")
  .id
  .placeholder
  .value
```

```
<input
  id="email"
  placeholder="Your email"
  value="angela@email.com"
/>
```

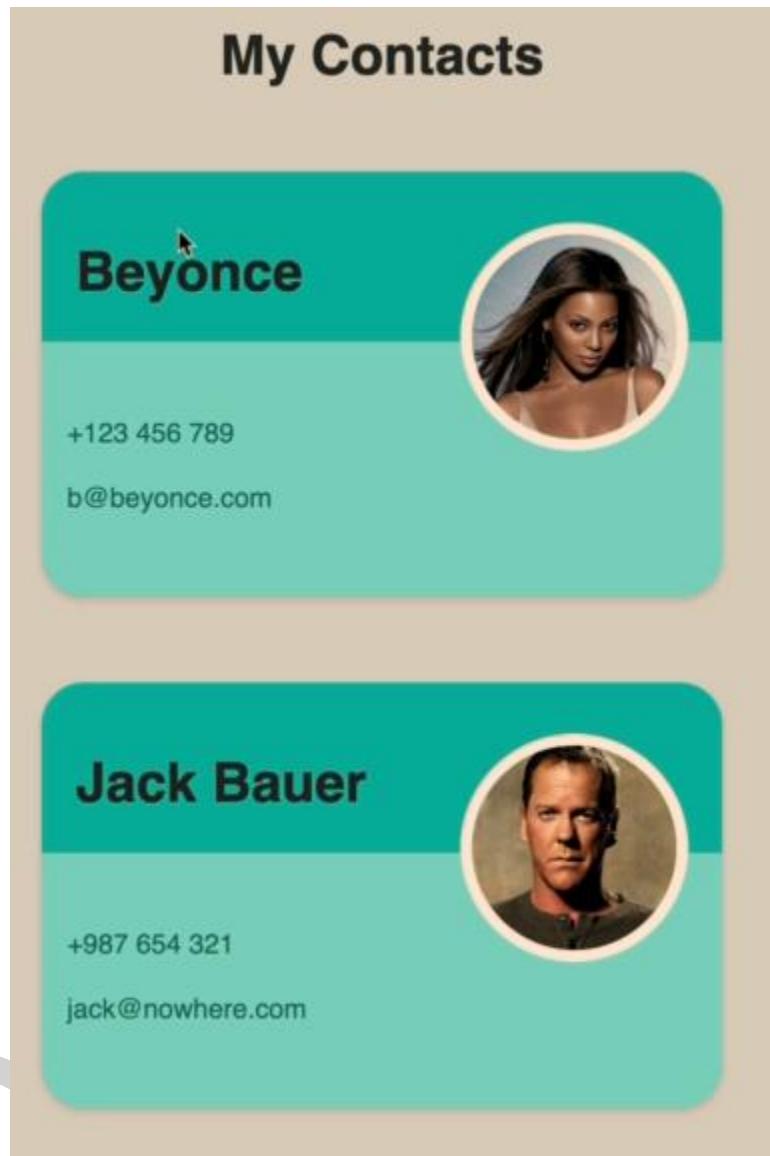
## React Props

```
function Card(props) {
  return <div>
    <h2>{props.name}</h2>
    <p>{props.tel}</p>
    <p>{props.email}</p>
  </div>;
}
```

```
<Card
  name="Beyonce"
  tel="+123456789"
  email="b@beyonce.com"
/>
```

## 20. React Props Practice

In this lesson you will practice about React component and its props.



## 21. React DevTools

In this lesson you will learn how to work with React DevTool to speed up your coding and debugging.



## 22. Mapping Data to Components

In this lesson you will learn how to map data to component with `map()` feature.

```
function createCard(contact) {
  return (
    <Card
      key={contact.id}
      name={contact.name}
      img={contact.imgURL}
      tel={contact.phone}
      email={contact.email}
    />
  );
}

function App() {
  return (
    <div>
      <h1 className="heading">My Contacts</h1>
      {contacts.map(createCard)}
    </div>
  );
}
```

## 23. Mapping Data to Components Practice

In this lesson you will practice about `.map()`.

# emojipedia



## Tense Biceps

"You can do that!" or "I feel strong!" Arm with tense biceps. Also used in connection with doing sports, e.g. at the gym.



## Person With Folded Hands

Two hands pressed together. Is currently very introverted, saying a prayer, or hoping for enlightenment. Is also used as a "high five" or to say thank you.



## Rolling On The Floor, Laughing

This is funny! A smiley face, rolling on the floor, laughing. The face is laughing boundlessly. The emoji version of "rofl". Stands for „rolling on the floor, laughing".

```
21 function createEntry(emojiTerm) {
22     return (
23         <Entry
24             key={emojiTerm.id}
25             emoji={emojiTerm.emoji}
26             name={emojiTerm.name}
27             description={emojiTerm.meaning}
28         />
29     );
30 }
31
32 function App() {
33     return (
34         <div>
35             <h1>
36                 <span>emojipedia</span>
37             </h1>
38             <dl className="dictionary">{emojipedia.map(createEntry)}</dl>
39         </div>
40     );
41 }
```

## 24. Javascript ES6 MapFilterReduce

In this lesson, you will learn how to use `.map()` , `.filter()` , `.reduce()` , `.find()` , `.findIndex()` , to work with arrays in ES6.

Don't forget that all of them, use `.forEach()` in their methods.

## 25. Javascript ES6 Arrow functions

In this lesson you will learn, how to use Arrow function in ES6 and update your old functions to Arrow functions.

## 26. Keeper App Project - Part 2

In this lesson you practice about .map() and arrow function.

## 27. React Conditional Rendering with the Ternary Operator & AND Operator

In this lesson you will learn how to use Ternary & AND Operators.

## Ternary Operator

```
CONDITION ? DO IF TRUE : DO IF FALSE
```

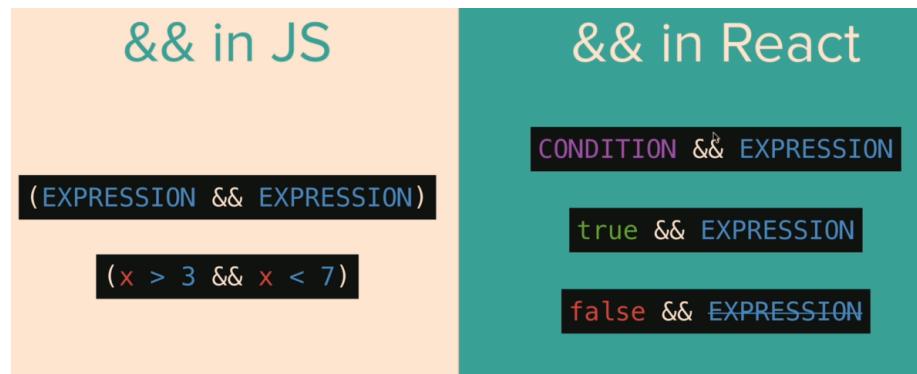
```
isCloudy === true ? bringUmbrella() : bringSunscreen()
```

## && in JS

```
(EXPRESSION && EXPRESSION)
```

```
var x = 5;
```

```
(x > 3 && x < 7)
```



## 28. Conditional Rendering Practice

In this lesson you will practice about Conditional operators.

**Form.jsx**

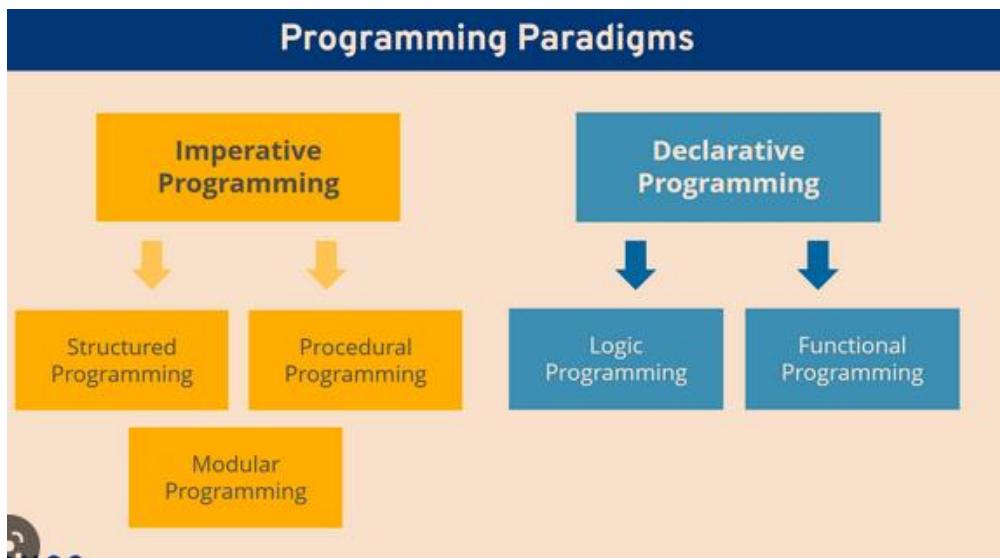
```

1 import React from "react";
2
3 function Form(props) {
4   return (
5     <form className="form">
6       <input type="text" placeholder="Username" />
7       <input type="password" placeholder="Password" />
8       {!props.isRegistered && (
9         <input type="password" placeholder="Confirm Password" />
10      )}
11
12       <button type="submit">{props.isRegistered ? "Login" : "Register"}</button>
13     </form>
14   );
15 }
16
17 export default Form;
18 
```

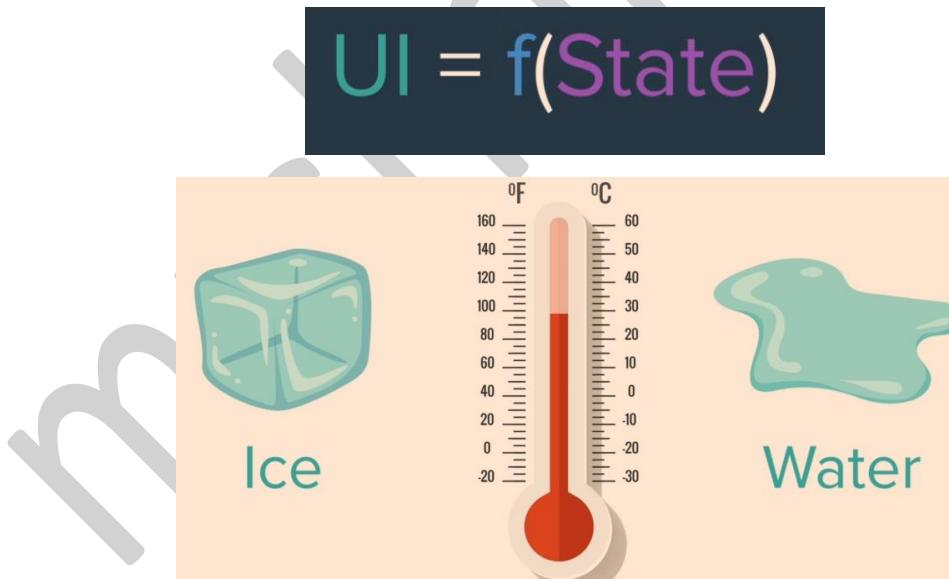
The screenshot shows a code editor with the file `Form.jsx`. The code defines a functional component `Form` that takes a prop `isRegistered`. It renders a form with two inputs and a button. If `isRegistered` is false, it adds a third input field for confirming the password. The rendered output in the browser shows a form with two inputs labeled "Username" and "Password", and a button labeled "Login".

## 29. State in React - Declarative vs. Imperative Programming

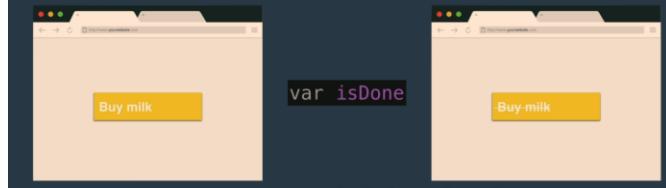
In this lesson you will learn about Difference between Declarative and Imperative programming.



- 1- Imperative is more complicated and deals with details, but Declarative is more simple.
- 2- In React we have useState hook module to change the state of component that is Declarative, instead working with old html DOM that is Imperative.



## Declarative Programming



## Imperative Programming

```
document.getElementById("root").style.textDecoration = "line-through";
```

### 30. React Hooks – useState

useState is a React Hooks that prepared a interactive UI for us. You can interact with component in UI and change them.

```
App.jsx
1 import React, { useState } from "react";
2
3 function App() {
4   const [count, setCount] = useState(0);
5
6   function increase() {
7     setCount(count + 1);
8   }
9
10  function decrease() {
11    setCount(count - 1);
12  }
13
14  return (
15    <div className="container">
16      <h1>{count}</h1>
17      <button onClick={decrease}>-</button>
18      <button onClick={increase}>+</button>
19    </div>
20  );
21}
22
23 export default App;
```

### 31. useState Hook Practice

In this lesson you will practice about useState.

### 32. Javascript ES6 Object & Array Destructuring

Destructuring is really helpful, just google it and find it. It's like doing query in database.

```
1 import animals from "./data";
2
3 console.log(animals);
4 const [cat, dog] = animals;
5 console.log(cat);
6
7 const { name, sound } = cat;
8 console.log(sound);
```

```
Console was cleared
▶ [Object, Object]
▶ Object {name: "cat", sound: "meow"}
meow
```

```
const { name: catName, sound: catSound } = cat;
```

```
const { name = "Fluffy", sound = "Purr" } = cat;
```

```
const { name, sound, feedingRequirements: {food, water} } = cat;
```

### 33. Javascript ES6 Destructuring Challenge Solution

You will practice about destructuring.

```
const [honda, tesla] = cars;

const {
  speedStats: { topSpeed: hondaTopSpeed }
} = honda;
const {
  speedStats: { topSpeed: teslaTopSpeed }
} = tesla;

const {
  coloursByPopularity: [hondaTopColour]
} = honda;
const {
  coloursByPopularity: [teslaTopColour]
} = tesla;
```

### 34. Event Handling in React

In this lesson you will learn how to handle the Events from UI in React.

```
function App() {
  const [headingText, setHeadingText] = useState("Hello");
  const [isMousedOver, setMouseOver] = useState(false);

  function handleClick() {
    setHeadingText("Submitted");
  }

  function handleMouseOver() {
    setMouseOver(true);
  }

  function handleMouseOut() {
    setMouseOver(false);
  }

  return (
    <div className="container">
      <h1>{headingText}</h1>
      <input type="text" placeholder="What's your name?" />
      <button
        style={{ backgroundColor: isMousedOver ? "black" : "white" }}
        onClick={handleClick}
        onMouseOver={handleMouseOver}
        onMouseOut={handleMouseOut}
      >
        Submit
      </button>
    </div>
  );
}
```

## 35. React Forms

In this lesson you will learn how to work with Form elements in React and how to control them with useState(). (refer to google and search “controlled components in React”).

In Html property of elements are belong to it and can't change it in back-end. But in React you have access and you can change those properties.

# Controlled Components

In HTML, form elements such as `<input>`, `<textarea>`, and `<select>` typically maintain their own state and update it based on user input. In React, mutable state is typically kept in the `state` property of components, and only updated with `setState()`.

We can combine the two by making the React state be the "single source of truth". Then the React component that renders a form also controls what happens in that form on subsequent user input. An input form element whose value is controlled by React in this way is called a "controlled component".

Uncontrolled(without value property)

```
<input onChange={handleFName} name="fName" placeholder="First Name" />
<input onChange={handleFName} name="lName" placeholder="Last Name" />
```

Controlled( with value property)

```
<input onChange={handleFName} name="fName" placeholder="First Name" value={fullName.fName}/>
<input onChange={handleFName} name="lName" placeholder="Last Name" value={fullName.lName} />
```

## 36. Class Components vs. Functional Components

In React we have to type of component, Class and Functional components.

For controlling State in old React, Class component has been used but in newer version of React the Hook or Functional component has used that is easier than Class.

```
Class
React Tutorials / Class Components vs. Hooks

App.js
FunctionalComponent.jsx

1 import React, { useState } from "react";
2
3 function FunctionalComponent() {
4   const [count, setCount] = useState(0);
5
6   function increase() {
7     setCount(count + 1);
8   }
9
10  return (
11    <div>
12      <h1>{count}</h1>
13      <button onClick={increase}>+</button>
14    </div>
15  );
16}
17
18 export default FunctionalComponent;
19
```

```
ClassComponent.jsx

1 import React from "react";
2
3 class ClassComponent extends React.Component {
4   constructor() {
5     super();
6     this.state = {
7       count: 0
8     };
9     this.increase = this.increase.bind(this);
10   }
11
12   increase() {
13     this.setState({ count: this.state.count + 1 });
14   }
15
16   render() {
17     return (
18       <div>
19         <h1>{this.state.count}</h1>
20         <button onClick={this.increase}>+</button>
21       </div>
22     );
23   }
24 }
```

## 37. Changing Complex State

In this lesson your will learn how to Change and control State when it goes more complicated. (like when we have some Input components.)

## 38. Changing Complex State Practice

It's just a practice about working with State.

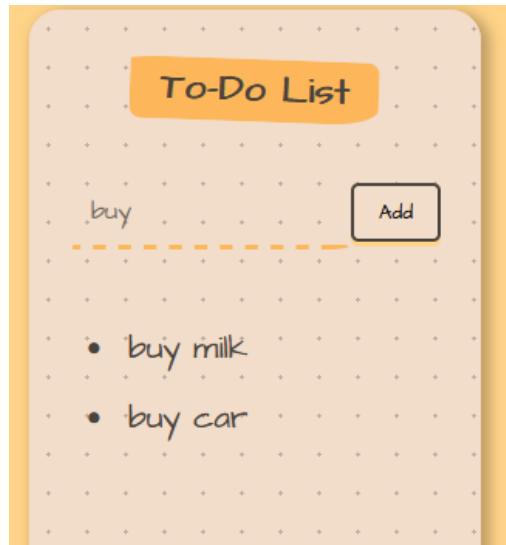
## 39. Javascript ES6 Spread Operator

In this lesson you will learn how to use Spread syntax in ES6 for Array and Object.

# Spread syntax (...)

## 40. Javascript ES6 Spread Operator Practice

In this practice with To-Do list project, you will remember what you have learnt like State, Spread etc.



## 41. Managing a Component Tree

In this lesson you will learn a new tips about how to manage components in React.

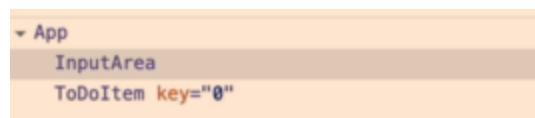
```
import React from "react";

function ToDoItem(props) {
  return (
    <div
      onClick={() => {
        props.onChecked(props.id);
      }}>
      <li>{props.text}</li>
    </div>
  );
}
```

## 42. Managing a Component Tree Practice

In this practice you will find more about managing component tree.

In fact it make your code more clear and modular and you will how to pass data in parent component to child and reverse.



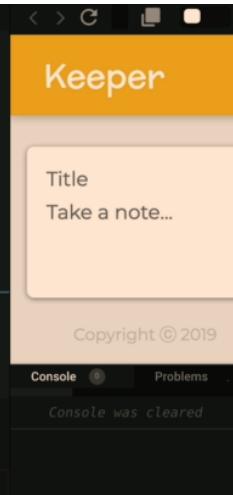
## 43. Keeper App Project - Part 3

This is a practice for what you have learnt till now.

```

3 import App from "./components/App";
4
5 ReactDOM.render(<App />, document.getElementById("root"));
6
7 //CHALLENGE:
8 //1. Implement the add note functionality.
9 //-- Create a constant that keeps track of the title and content.
10 //-- Pass the new note back to the App.
11 //-- Add new note to an array.
12 //-- Take array and render separate Note components for each item.
13
14 //2. Implement the delete note functionality.
15 //-- Callback from the Note component to trigger a delete function.
16 //-- Use the filter function to filter out the item that needs deletion.
17 //-- Pass a id over to the Note component, pass it back to the App when deleting.
18

```



## 44. React Dependencies & Styling the Keeper App

In this lesson you will learn how to use Material UI library for use pre-built React component in your front-end and have material design concept in your project.

MUI is like bootstrap library but prepared for React.

## Material Icons

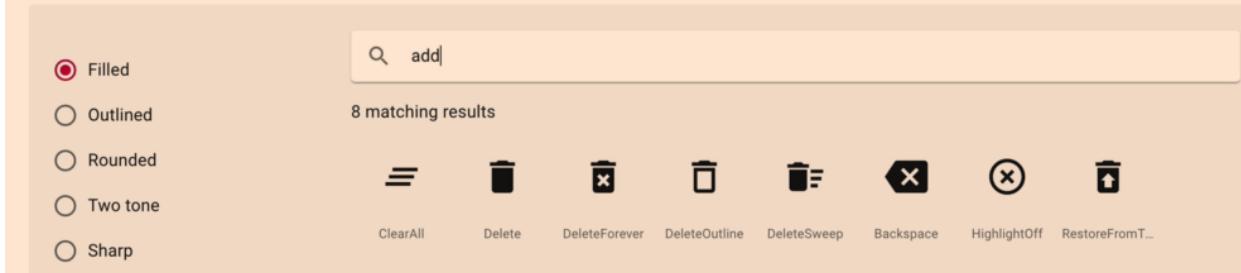
EDIT THIS

1,000+ React Material icons ready to use from the official website.



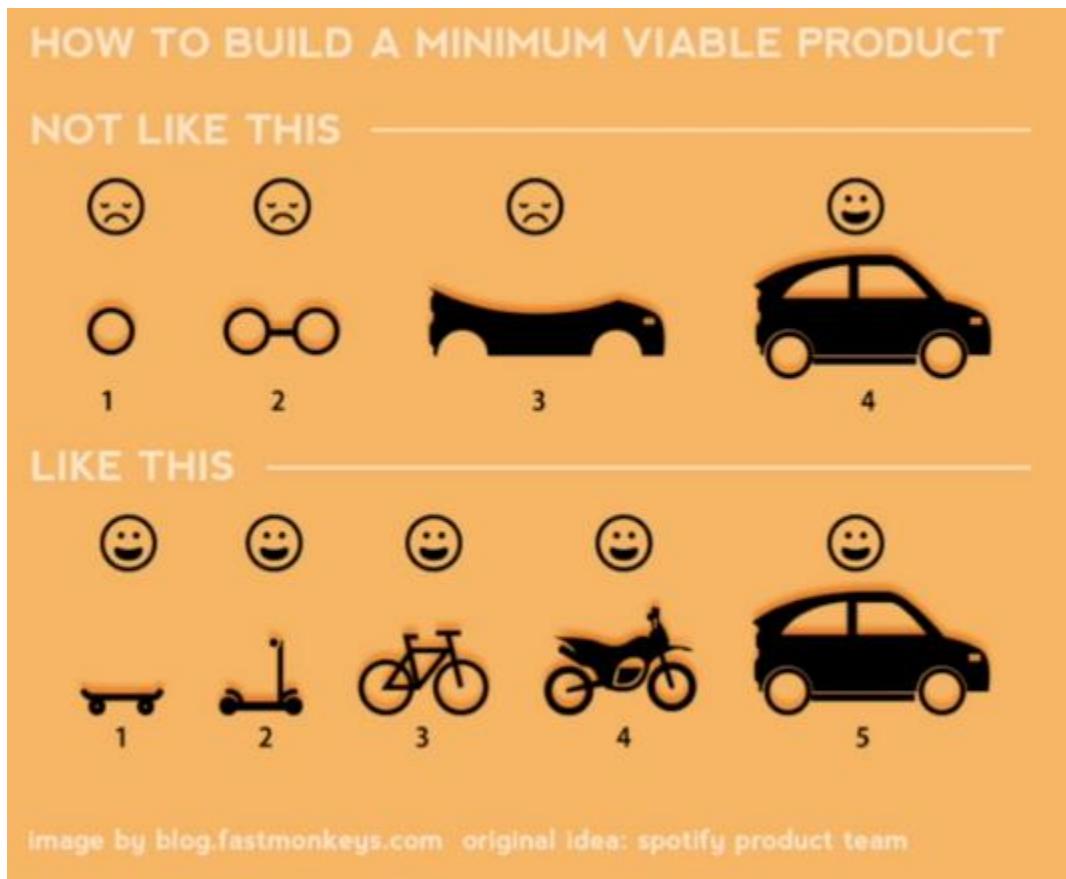
ScaffoldHub - Automate  
building your full-stack  
Material-UI web-app.  
ad by Material-UI

The following npm package, [@material-ui/icons](#), includes the 1,000+ official Material icons converted to `SvgIcon` components.



### 45. Tip from Angela - How to Build Your Own Product

بعد از دیدن این دوره احتمالا میخواهید رو پروژه یا اپ خود کار کنید یا استارتاپ خود را راه اندازی کنید. چیزی که مهمه ابتدا برآورد کنید که اپ شما به چه چیزهایی نیاز دارد. یعنی معماری اپ خود و موارد لازم آن را در ابتدای کار فراهم کنید. در ادامه ساده ترین نسخه برنامه خود را پیاده سازی و اجرا کنید. قرار نیست از اول اپ شما کامل باشد. به مرور زمان و فیدبک گرفتن از کاربران برنامه خود را بهبود و توسعه می دهید مانند فیس بوک و اینستا و غیره. پس شروع کنید.



### 34. Bonus Module Ask Angela Anything

- برای یادگیری هر چیزی ابتدا کلیات را بفهمید بعد در ادامه جزئیات را درک کنید.
- برای شروع برنامه نویسی، ابتدا از سایت های فریلنسری پروژه با درآمد کم بگیرید تا هم پول بگیرید، هم نمونه کار ایجاد کنید و هم با مشکلات پروژه های واقعی آشنا شوید.
- سعی کنید همیشه در حال یادگیری باشید. یه مدت از روز را برای یادگیری چیزهای جدید اختصاص دهید. با این کار ذهنتان همیشه آماده خواهد بود و نخواهد مرد. این سختی را بپذیرید و پیشرفت کنید.
- یه مدت از روز را برای کار بالارزش خود اختصاص دهید و آن را متمرکز و به دور از هر حواس پرتی انجام دهید.
- با مشکلات رو برو شوید و به جای تغییر دنیا بیرون، دنیای درون خود را کنترل کنید.
- سعی کنید مفاهیم و الگوریتم ها را بفهمید در ادامه یادگیری هر زبان برنامه نویسی راحت خواهد بود.
- برای فریلنسری از سایت upwork یا fiver شروع کنید(حتما قبلش یک pet project یعنی پروژه ای که حداقل برای یک نفر قابل استفاده است را ایجاد کنید تا نمونه کارتان شود). پروژه های ارزان قیمت و کوچک که کمتر از یک هفته زمان می برد را قبول کنید و آنها را انجام دهید. چیزی که مهم

است این است که مشتری هایتان را راضی نگه دارید و نظرات خوبی در صفحه خود بگیرید. در ادامه شما پروژه های زیادی انجام دادید و تجربه کسب کرید و نمونه کار ساخته اید و درآمد کوچکی ایجاد کرده اید و از همه مهمتر مشتری های راضی دارید که در ادامه با شما همکاری بیشتری خواهند داشت و شبکه قوی ایجاد خواهد کرد. پس در شروع حتما پروژه های کوچک را قبول کنید و طمع پول نکنید.

You need to make a portfolio, you need to show the world, not tell the -  
world what you can do.

امیدوارم این فایل به کارتان آمده باشد.