Table of Contents

[Basics of UI Design 2](#__RefHeading___Toc5498_1421093447)

[Steps for Creating Real Design Project 3](#__RefHeading___Toc5500_1421093447)

[Basics of CSS-3 4](#__RefHeading___Toc5502_1421093447)

[Basic CSS styles 5](#__RefHeading___Toc5504_1421093447)

[CSS Selectors 6](#__RefHeading___Toc5506_1421093447)

[Pseudo Selectors 7](#__RefHeading___Toc5508_1421093447)

[basics of JavaScript 7](#__RefHeading___Toc5510_1421093447)

[JS String methods 8](#__RefHeading___Toc5512_1421093447)

[number methods 8](#__RefHeading___Toc5514_1421093447)

[JS Array methods 9](#__RefHeading___Toc5516_1421093447)

[Js Date 9](#__RefHeading___Toc5518_1421093447)

[JS Objects 10](#__RefHeading___Toc5520_1421093447)

[Regular Expression 11](#__RefHeading___Toc5522_1421093447)

[HTML DOM & j Query 12](#__RefHeading___Toc5524_1421093447)

[JavaScript Best Practices 13](#__RefHeading___Toc5526_1421093447)

[Command Line Interface (CLI) 13](#__RefHeading___Toc5528_1421093447)

[Git & git-hub 14](#__RefHeading___Toc5530_1421093447)

[React Js 17](#__RefHeading___Toc2050_3466779962)

[API Application Programming 18](#__RefHeading___Toc5532_1421093447)

[UI/Ux Websites 18](#__RefHeading___Toc5534_1421093447)

[Development Websites 19](#__RefHeading___Toc5536_14210934471)

[API Websites 19](#__RefHeading___Toc5538_1421093447)

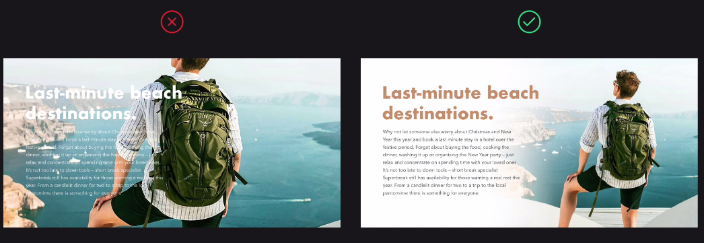
# **Basics of UI Design**

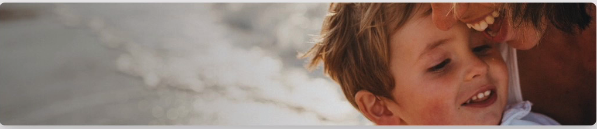
Hierarchy **: -** what is the most important thing you want to show or tell to your audience Mostly we use font weight and font size to make hierarchy.

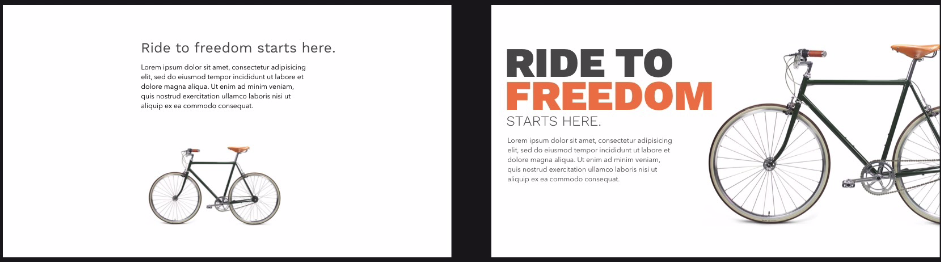
Layout **: -** Alignment everything in our design should be properly aligned to each other

Proximity **: -** things they are related should be closer together

Typography **: -** font weight, Font type (try to use contrasty font for header and paragraph (they shouldn’t be very similar, so may be using serif and sanserif combination give you good result.

Images **: -** when using text image as background you can use over lay to make the background a little bit blackish that can make contrast b/n your background image and your heading.

extreme crop (select the most important thing in the photo and crop and magnify it as much as you can, soft crop)

Consistency **: -** stay consistent for d/f part of your elements (e.g.: similar heading and sub heading for all sections, similar style for paragraphs, buttons etc.). consistency on (colors, shapes, photos)

Contrast **: -** (on text (size, weight, color), on colors (using complimentary color)

Tension **: -** white space, reputation (for similar contents), overlapping, rule of third when possible Reputation.

Components **: -** we can choose component type based on our resource

tabs or accordions **: -**  (if you have many content but you want to display it on small section

Color psychology **: -** every design have a color psychology we want our user to feel.

Red **: -** power, passion, strength and excitement

Orange **: -** draws attention without being as overpowering as red. It means cheerfulness and creativity. Orange can be associated with friendliness, confidence, and courage.

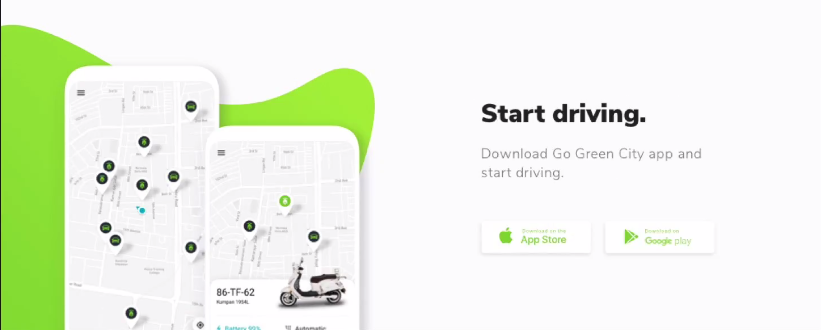
Yellow **: -** is energetic and gives the feeling of happiness and liveliness. Also, it associates with curiosity, intelligence, brightness, etc.

Green **: -** is the color of harmony, nature, life and health, money

Blue **: -** means patience, peace, trustworthiness, and stability, professionalism, trust and honor.

Purple **: -** wisdom, royalty, nobility, luxury, and mystery.

Pink **: -** romance,passivity, care, peace, affection.



# **Steps for Creating Real Design Project**

1. Create project brief by asking your client
2. Create a mode board using figma by using inspirational designs related to your project.
3. Create wire frame
4. create Your assets

3.1 create font asset

font asset (Select your heading and paragraph font), font scaling, letter spacing, line height

3.2 create color asset

create base color(primary, secondary, ternary), create gray color, create lighter version of your base color by adding white

1. use 8 multiple rule for padding and margin e.g mostly I use (heder section margin top 32px, left & right: 64px(on grid), Space b/n elements on the same section: 32px, Space b/n button on hero section and the above element is: 96px, components on the same section: 128px, Space b/n major sections: 256px

# **Basics of CSS-3**

* Centering **: -** margin: auto put the block element center
* box – sizing **: -** padding add extra length to the element to stop that box-sizing: border-box.
* Position **: -** (Static (default), relative (it is relative to its original position, and do not affect the original layout of other elements), fixed (it sticks itself and its relative to the screen and change the layout of the elements after it), absolute (it is relative to its nearest relative parent and change the lay out of the elements after it)
* Float **: -** use float only to wrap text around element we can use clear on the element below the image. Clear (left, right or both): Clear: left: remove left margin from element.
* Display **: -** it possible to set width and Hight for block elements but they do not allow other elements to sit near them. but it is possible to do it if you change the position to inline block.
* Z - index **: -** to change z-index first change the Position of the element from static (default for all elements) to the other.
* Font size **: -** When you specify font size try to use rem instead of px.
* Flex **: -** ([flex-direction](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "flex-direction), [flex-wrap](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "flex-wrap), [flex-flow](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "flex-flow), [justify-content](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "justify-content), [align-items](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "align-items), [align-content](https://www.w3schools.com/css/css3_flexbox_container.asp" \l "align-content))
* Flex item **: -** properties are:([order](https://www.w3schools.com/css/css3_flexbox_items.asp" \l "order), [flex-grow](https://www.w3schools.com/css/css3_flexbox_items.asp" \l "flex-grow), [flex-shrink](https://www.w3schools.com/css/css3_flexbox_items.asp" \l "flex-shrink), [flex-basis](https://www.w3schools.com/css/css3_flexbox_items.asp" \l "flex-basis), [align-self](https://www.w3schools.com/css/css3_flexbox_items.asp" \l "align-self))
* vh / vw **: -** these can be useful any time you want something to be sized relative to the view port, examples including full-height heroes, full-screen app-like interfaces.

# **Basic CSS styles**

Responsive image

Img,picture,video,svg{

Width: 100%; : it grow and shrink with the parent container

hight: auto; : the hight will grow and shrink based on the width

Object-fit:contsin /\* preserve a nice aspect ratio\*/

}

Relative Path

<img src="../images/dog.jpg"> ../get out from current folder get in to images

<img src="./images/dog.jpg"> ./ god practice to start this way although they are on the same folder it tel to find the image relative to current location

Sticky nav

nav scrole up to some point and stick at top 0

<div id=”top”></div>

<div id =”down”></div> // #down{position: sticky; top: 0}

Growing field

input grow when user type

input, textarea, select {

field-sizing: content;

}

Cropped text

if texts is out of the box or cropped ?

tetext {

overflow: hidden;

white-space: nowrap;

text-overflow: ellipsis;

}

// List Image

list-style-img: url()

2D Transform

transform: translate (x, y): transform: rotate (angle) : transform: scale (x, y):

transform: skewx(angle) : transform: skewy (angle) : skew(x-ang, y-ang)

3D Transform

transform: rotateX (angle): transform: rotateY (angle) : transform: rotateZ (angle):

transform: scale3d (x, y, z):

# **CSS Selectors**

child selector: every element w/c is found before the closing tag of other element is a chilled of that element

li:first-child: li w/c is the first child of its parent

li:last-child: li w/c is the last child of its parent

li:nth-child(3,4,even,odd): li w/c is the third,fourth,even,odd child of its parent

:nth-of-type(n) N-th sibling of its type

:last-of-type Last sibling of its type

:first-of-type First sibling of its type

General Selector

\*{} Universal Selector

#id {} ID Selector

class {} Class Selector

h1, h2{} Type Selector

h+p {} Adjacent Sibling Selector (the first “p” inside “h”)

hi ~ p {} General Sibling Selector (select all “p” inside “hi” on our document)

p a{} Descendant Selector ( “a” found after “p” )

div[att="val"]{} Attribute Selector

# **Pseudo Selectors**

|  |  |
| --- | --- |
| :active active element  : focus focus element  :hover hover element  : link link element  :disabled disabled element  :enabled enabled element  :checked checked element | :empty Element with no children  :root Root element  :not(x) Element not matching ‘x’  :target Target element specified by a URI  ::before I nsert content before an element(h2::after{content: "some text"  Color:red;  }) |

# **basics of JavaScript**

|  |  |
| --- | --- |
| **ternary operater**  variablename = (condition) ? value1:value2  age = Number(age);  **Template literals**  const name = 'Alice'; const greeting = `Hello, ${name}!`;  // 'Hello, Alice!'  **Self-Invoking Functions**  (started) automatically, without being called.  (function () {  let x = "Hello!!"; // I will invoke myself  })(); // anonymous self-invoking function  Function Rest Parameter  function sum(...args) | **Async awit**  async function getData() {  try {  const response = await fetchData();  console.log(response);  } catch (error) {  console.error(error);  }} // For...of Loop (user for to iterate over elements of array, object, string, sets)  const colors = ['red', 'green', 'blue'];  for (const color of colors) {  console.log(color);  } |

# **JS String methods**

“DEVELOPER” .toLowerCase() “developer”

“developer” .toUpperCase() “DEVELOPER”

“developer” .length 9

“developer” .charAt(1) “e"

“Sloba’” .includes(“ba") true

“Sloba" .endsWith(“Co") false

“Codewith” .concat("Sloba") “CodewithSloba’

“CodewithSloba” .slice(“O,4") “Code”

“Follow,Sloba" .split(“,”) ["Follow’,"Sloba”]

“the man of” .match(/man/g) “man”

“welcome” .replace("welcom","go"); “go”

# **number methods**

|  |  |
| --- | --- |
| math.floor()  math.random()  x=7.787 | x.toFixed(2) = 7.78  parseInt("10 years") = 10 |

# **JS Array methods**

[“orange”, “apple”] .include(“Orange”) true

[“orange”] .splice(2, 0, "Kiwi") [“orange”,2,0,”Kiwi”]

[1,2,3] .map((num) => num \* 2) [2,4,6]

[1,2,3] .find()

[1,2,3] .filter((num) => num % 2 === 0); [2]

[1,2,3] .push(4);  [1,2,3,4]

[“orange”, “apple”] .fruits.pop( );  [“orange”]

[“orange”, “apple”] .shift( );  [“apple”]

[1,2,3] .unshift(4);  [4,1,2,3]

const users = [ { id: 1, name: 'Alice' }, { id: 2, name: 'Bob' }, { id: 3, name: 'Charlie' },];  
const user = users.find((user) => user.id === 2); // { id: 2, name: 'Bob' }  
const userIndex = users.findIndex((user) => user.id === 2); /

# **Js Date**

|  |  |
| --- | --- |
| **curretn date and time formted**  const d = new Date(); let text = d.toDateString();  // out put Thu Mar 28 2024'  **Set interval function**  var intervalId = setInterval(function() {  num++;  console.log("num:", num);  if (num === 3) {  clearInterval(intervalId);  }}, 1000); | **time interval function**  var timerId = setTimeout(function() {  console.log("This function runs in 30 seconds");  }, 30000);  clearTimeout(timerId); |

# **JS Objects**

|  |  |
| --- | --- |
| const person = { name: 'John', age: 30,**};**  // Access property  const name = person.name; // 'John'  // Add a new property  person.city = 'New York';  **Destructuring**  const person = { name: 'Bob', age: 25 };  const { name, age } = person; // name = 'Bob', age = 25 const colors = ['red', 'green', 'blue'];  const [first, second] = colors; // first = 'red', second = 'green' | If you want to create 30 objects which have similar property it will be hard using normal object but if we use a constructer fuction we can easly creat as many as we want  function Data(name,gread,age){  this.name = name;  this.gread=gread;  this.age = age;  }  Person1 = new Data("abel",7,13);  Person2 = new Data("selam",11,23); |

# **Regular Expression**

// Regular expression to match email addresses

const emailRegex = /^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/;

// Test string

const testEmail = "example@email.com";

// Test if the email matches the regex

const isValidEmail = emailRegex.test(testEmail);

// Output the result

console.log(isValidEmail); // true

Explanation:/^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/

/: Delimiter to indicate the start of the regular expression.

^: Anchors the match at the start of the string.

[a-zA-Z0-9.\_%+-]+: Matches one or more characters that are alphanumeric, or one of the special characters .

, \_, %, +, or -.@: Matches the @ symbol.

[a-zA-Z0-9.-]+: Matches one or more characters that are alphanumeric or .

, -.\.: Escapes the dot character . to match a literal dot.

[a-zA-Z]{2,}: Matches two or more characters that are alphabetic.

$: Anchors the match at the end of the string.

/: Delimiter to indicate the end of the regular expression.

const testEmail = "example@email.com";Define a test string representing an email address.const isValidEmail = emailRegex.test(testEmail);

Use the test() method of the regular expression object to check if the test string matches the regular expression.console.log(isValidEmail); // trueOutput the result, indicating whether the test email is valid according to the regular expression. In this case, it will log true.

# **HTML DOM & j Query**

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes and all css selectrs.

$(“li”).html: return content inside li element.

$(“li”).text: return only the text inside li elment.

Note that using text is prefered than html(for security)

$("\*") Selects all elements

$(this) Selects the current HTML element(for list e,g button li)

$(":button") Selects all <button> elements and <input> elements of type="button"

$("tr:even") Selects all even <tr> elements

$("tr:odd") Selects all odd <tr> elements

$(“a”).attr(“href”): return the value of the href atribut of a

$(“a”).setAttribute("id", "theDiv");

$(“a”).removeAttribute("id")

$(“a”).attr(“href”, “google.com”): replace the value of the href atribut to google

$(“li”).addClasslist(big, live): add big class and live class to li element.

$(“li”).removeClass(big): remove big class from li element.

div.classList.toggle("active")

append() - Inserts content at the end of the selected elements

prepend() - Inserts content at the beginning of the selected elements

after() - Inserts content after the selected elements

before() - Inserts content before the selected elements

remove()

empity()

# **JavaScript Best Practices**

\* "use strict"; to write secure js code

\* Always Declare Local Variables avoide global variables

\* Initialize Variables

\* Declare Arrays, function , regexp & Objects with const

\* Use === Comparison

\* Use Parameter Defaults. function myFunction(x=1, y=1) { }

\* Undefinde should be first if (typeof myObj !== "undefined" && myObj !== null)

\* Try to reduce code inside a loop. let l = arr.length; for (let i = 0; i < l; i++) { }

\* If you expect to access a DOM element several times, access it once, and use it as a local variable: const obj = document.getElementById("demo");

obj.innerHTML = "Hello";

\* Putting your scripts at the bottom of the page body lets the browser load the page first. An alternative is to use defer="true" in the script tag.

\* Objects are addressed by reference, not by value.

# **Command Line Interface (CLI)**

touch: to creat a file

Echo >> index.html "heloo world": write helow world on index.html

Cat index.html: to see the content of index.html inside our terminal

Ctrl+C: enable us to get out of some modes in our terminal.

Ctrl+shift+C/Ctrl+shift+V: copy and past on cli

Ls/ls -a: list file incurrent folder/list file incurrent folder including hiden files

pwd – show current directory

mkdir dir – create a directory dir

rm -r dir – delete directory dir  
rm -f file – force remove file  
rm -rf dir – force remove directory dir \*  
cp file1 file2 – copy file1 to file2  
cp -r dir1 dir2 – copy dir1 to dir2; create dir2 if it  
doesn't exist  
mv file1 file2 – rename or move file1 to file2  
if file2 is an existing directory, moves file1 into  
directory file2

Tab: auto fill the started comand

code my\_awesome\_project/: open vs code project

Cd folder: to get in to folder

Cd ~ : to return to our root folder

Code . : open the curent folder on vs code as a project folder.

# **Git & git-hub**

after you install git bash on your window pc configure these

**User Configuration** : git config --global user.name "Your Name", git config --global user.email "yourname@example.com"

**Master to main:** git config --global init.defaultBranch main

**branch reconciliation :** to merging: git config --global pull.rebase false

**Default Editor :** vs code: git config --global core.editor "code"

**git init :** initialize git repository on our local project folder

**Git status:** tell us the changes made but not staged files of the project

**git add:** add the changed file to staging area

**git commit:** save change

**git show (commit hash code ) :** if you modified some file on this commit the show command show you how you modified the file

**git diff hash1 hash2 :** show the d/f b/n two commits + (added content) & - (deleted content)"

**Subject of git commit:** A brief summary of the change you made to the project.

**Body of git commit:** Describe the problem your commit solves and how.

E.g : subject: Add missing link and alt text to the company's logo

body: Screen readers won't read the images to users with disabilities without this information

**git log:** list commits

**git log --one line:** list commit short way

**connect your local repo to remote :**

1. create remote repo by going to your github acc(after you create remote reop you will get a url for that repo) and you have to create personal access token for security
2. To get personal access tokken: On your github acc click on your profile - > seting - > developer - > personal access tooken(clasic)
3. git remote add origin [https:](https://personal/)[//personalaccesstokn@remotelink](mailto://personalaccesstokn@remotelink)(with out https)

e.g: git remote add origin https://ghp\_5N7cy9hlJQylgF  
oF1hy3DBXpl2Fqq92aCK5j@github.com/josephgeberu/FullStack-Notes.git

1. git remote -v: To check the connected url

**Delete remote :** git -d remote origion: delete connected url (if you clone other person project and if you are not contributor so you cant push b/c your connected url is not yours

**pushing for the first time :** from the new locally created branch

git push remote -u origin main b/c the remote don’t know about the new branch you have to use this

**Git checkout :** to restore deleted file(rm file)

**Git colaboration**

i**nvite collaborator to your remote repo**

Open repo - > seting tab - > colaborator - > add people - > username(email of the person)If he accept your invitation on his email your repo will copied in to his github acc

**clone the project :**  after clicking the newly added repository he has to click on code tab and copy the link

git clone http://github..... after you clone you can push no need to git init if you are contributor your push update your remote repo and a person who invite you repo at the same time

**cloning specific branch :** Git clone -b origin branch2 -s : if you use the above clone command by default you are cloning from main branch but this enable you to clone from any specified branch

**git pull:** to copy change from your remote repository to your local repo always make sure you pull first when you want to push to update your local repo.

**git push -f :** if the change on remote repo is not correct you can push force fully with out pull but not advisable the command is this will replace the remote repo files by your local repo file

**git branch:** To list all branches found on local tepo

**Git branch -r:** To list all branches found on remote repo

**Git branch -a:** To list all branches found on remote and local repo

**git branch <branch\_name>:** To create a new branch(the new branch copy all the files frome the parent branch)

**git branch -d <branch\_name:** To delete a branch

**git checkout <branch\_name>:** to switch between branches

**git checkout -b <your\_new\_branch\_name>:** create and checkout to a branch

**Git merge :** Git merge source-branch: to copy change from source brach, after merging there is no need to add and commit simply push to remote

**git diff <source branch> <target branch> :** to compare two branches:

**git branch –abort :** if we want to cancel merging.

**git request-pull:** if you clone the project from other person git hub account and you made a change on it, if you want to merge your change to the main project you have to ask the project owner to merge your change using git hub.

**Git conflict:** conflict happen when we pull and merge the reason is when one file on our local repo has some difference from remote repo

e.g 1.txt (hello welcome) on remote repo but in our local repp 1.txt(helo there) so when we tray to merge or to pull, conflict will raise.

so we have to open 1.txt on our local repo and this file will show us both changes after if we want both changes we have to delete only git added things(head,hash,>>>>), if we want our change delete all except ours then we can add, commit and push.

There are two types of merge fast for ward(when the changes only from one branch) 3-way(when there is a change from both branches)

**Git rebase** : is similar to git merge but there difference is when they create graph of merging

when git merge create straight forwarding merge (merging branches have no many difference) its graph is straight but when it create 3-way (the merging branches have many difference) merge its graph is not straight.

to make 3-way merging straight graph we can use git rebase but it is very bulky to fix many conflict. so it is not advisable we can(skip(to skip conflict), continue after we fix one conflict we can git rebase continue

But if we merge using 3-way merge we have to deal only one conflict

**Git reflog** : instad of using hash code of a commite this command tell as the position of acomite with respekt to the head.

git show head~7: tel as what we did on commit at position 7

**Ditched head :** created when we want to caret experimental commits. git checkout commit hash move us to specific commit that we want to start to experiment with after this all commits we made will follow this path if we want that experiment commits to be real we can create and check out new branch

**Git fetch :** help to copy remote branches to your local e.g if you delete a branch by accident on your local.

Git fetch remote origin a(branch from remote) : a(create branch a on local and copy a(remote) on a(local) - >

**.gitignore** : is a very important file in the Git world because it prevents extraneous files from being submitted to version control.

# **React Js**

const products = [

{ title: 'Cabbage', isFruit: false, id: 1 },

{ title: 'Garlic', isFruit: false, id: 2 },

{ title: 'Apple', isFruit: true, id: 3 },

];

export default function ShoppingList() {

const listItems = products.map(product =>

<li

key={product.id}

style={{ // if want to style based on condition we can use style attribute of jsx

color: product.isFruit ? 'magenta' : 'darkgreen'

}}

>

{product.title}

</li>

);

return (

<ul>{listItems}</ul>

);

}

often you’ll need components to share data and always update together.

To make both MyButton components display the same count and update together, you need to move the state from the individual buttons “upwards” to the closest component containing all of them.

import { useState } from 'react';

export default function MyApp() {

const [count, setCount] = useState(0);

function handleClick() {

setCount(count + 1);

}

return (

<div>

<h1>Counters that update together</h1>

<MyButton count={count} onClick={handleClick} />

<MyButton count={count} onClick={handleClick} />

</div>

);

}

function MyButton({ count, onClick }) {

return (

<button onClick={onClick}>

Clicked {count} times

</button>

);

}

# **API Application Programming**

API a way to use service of others who host some service on there server and let us use the service they provide on our website. Thy may let us to use there server fully or they may want authentication in order to use there service. The flow is ( our client ask service form our server, using our API we send our client request to the API server)

API: has 4 parts: endpoint, Path, Parameters, Authentication

# **UI/Ux Websites**

|  |  |
| --- | --- |
| Design inspiration  [awwwards.com](https://www.awwwards.com/websites/)  [Dribbble.com](https://dribbble.com/shots/popular/web-design)  Design resource  [Flat icon](https://www.flaticon.com/)  [Unsplash.com](http://Unsplash.com/)  [Pixels.com](http://Pixels.com/)  [unblast](https://unblast.com/):mocups  [human illustrations.](https://blush.design/collections/humaaans/humaaans) | [Fontjoy](https://fontjoy.com/) : font combinations.  l[eanardo(AI image)](https://app.leonardo.ai/)  [freepick](https://www.freepik.com/): (ilustration, icon, photo)  [shapfest](https://www.shapefest.com/): 3D packs:  [smashing logo](https://smashinglogo.com/) : Logo design Development Websites [CSS Button](https://css3buttongenerator.com/) [Generator:](https://css3buttongenerator.com/)  [Readme.so](http://Readme.so/) **:** create a ReadMe |

# **API Websites**

[ipinfo.io](http://ipinfo.io/) : specified IP address, such as geological info, company, and carrier name.

[developers.notion.com](http://developers.notion.com/) **:** (workspace for your tasks, notes, etc), you're gonna have fun with their API.

[stripe.com/docs/api](http://stripe.com/docs/api) **:** payment tool that powers up Online businesses.

[developers.google.com/people](http://developers.google.com/people) **:** to get detail abut autenticated user.

[api.adviceslip.com](http://api.adviceslip.com/) **:** This API Provides random pieces of advice.