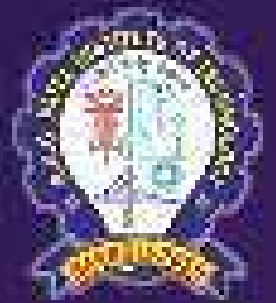




# ESKHA

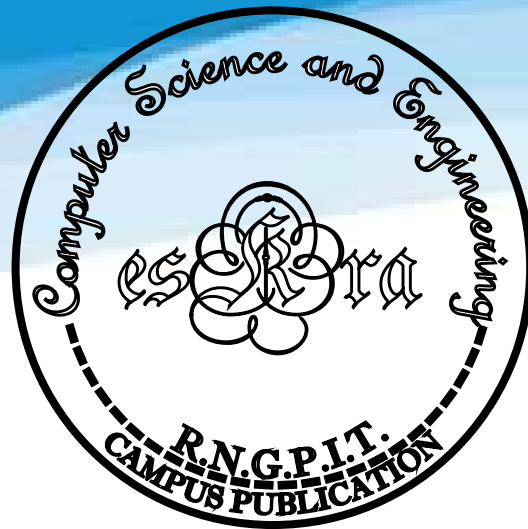


— HE THINKS WE SPARK

CSE Dept. Campus Publication | RINGPIT | ISSUE:01 | AUG-2020



ENTERING THE WORLD  
OF DEVELOPMENT



**Circulation:** **ESKRA** is published frequently by students of Computer Science and Engineering department of R. N. G. Patel Institute of Technology, Bardoli-Navsari road, At: Isroli, Tal: Bardoli - 394620, Dist: Surat, Phone: +91-9427527398, +91-9033707334, +91-9510427118 e-mail: eskra.cse@rngpit.ac.in

**Editorial:** Unless otherwise stated articles, as well as images and description, reflect author's or firm's opinion. Inclusion in **ESKRA** does not necessarily constitute endorsement by **ESKRA** or students of Computer Science and Engineering Department of R.N.G.P.I.T. All submissions are subject to editing for style, clarity and space.

### **Reserve rights and reprint permission :**

Educational or personal use of this material is permitted without fee, provided such use:

(i) is not made for profit.

(ii) includes this notice and full citation to original work on first page of copy, authors are permitted to post accepted version of **ESKRA** copyrighted material on their own web servers without permission, provided that this notice and full citation to original work appear on first screen of posted copy material for commercial, advertising or promotional purposes or for creating new collective works for resale or redistribution must be obtained from Computer Science and Engineering Department of R.N.G.P.I.T copyright©2020.CSE-Department, R.N.G.P.I.T. All rights reserved.

# Message from Editor-in-Chief



**Dr. Madhavi B. Desai**

(HoD, CSE, RNGPIT)

(Editor in Chief, Eskra)

As members of the Computer Science and Engineering department, we are always curious about new developments in the technology. A computer scientist's job entails a lot of development as everything from automatic highway toll systems to smart light-bulbs, multinational web servers, and integrated residential air conditioning systems is developed by Computer Scientists. We've featured articles in this issue that cover topics such as website, application, and game development, as well as testing, automation, and the fundamentals of using various frameworks.

Our mission is to be at the frontline of sharing development and automation strategies to academics and the general public worldwide. "Entering the World of Development" takes us on a tour of development related topics and piques the reader's interest more in learning. Each item in this magazine is written with the goal of providing each reader with a wealth of information.

"What exactly is development, and how does one go about pursuing a career in this field?" is a question that virtually every reader has, and we have attempted to address this issue via our articles.

# Editorial



**Mr. Nikunj Y. Kansara**  
(Asst. Professor, CSE, RONGPIT)  
(Faculty Advisor)



**Mr. Hemang J. Shah**  
(Asst. Professor, CSE, RONGPIT)  
(Faculty Advisor)

Web development can range from developing a simple single static page of plain text to complex web applications, electronic businesses, and social network services. A typical mobile application uses a network connection to work with remote computer resources, and mobile application development is the process of building software applications that operate on a mobile device. As a result, the mobile development process include producing installable software packages (code, binaries, assets, and so on), as well as establishing backend services like data access through an API. Automated Website Testing refers to the process of automating end-user scenarios on a website in order to test its behaviour.

Mobile application development is the act or process by which a mobile app is developed for mobile devices, such as personal digital assistants, enterprise digital assistants or mobile phones. The web development process includes web design, web content development, client-side/server-side scripting and network security configuration, among other tasks. Testing may assist you in avoiding costly mistakes and maximising the value of your web development efforts. It enables you to devise effective solutions for dealing with potential future problems. The necessity for continual testing and maintenance of your website's design will expand as your online presence develops.

# About Authors

Raj Kharvar



Raj Kharvar is a FOURTH year student of CSE, R.N.G.P.I.T. He wrote many articles to enrich the knowledge of his juniors. With his extensive knowledge of various aspects of JavaScript, he shared his expertise with others by providing a glimpse into his thoughts on Deno replacement or Node.js

“I don’t think I need to get a life. I’m a gamer with many lives.” - Deep Mevada, a FOURTH year student of CSE, R.N.G.P.I.T. believes this. With such a wonderful belief in his heart, he wrote an article on Game Development so that other students could also diversify their imaginations and create something where they could be reborn.



Deep Mevada

Rahul Gupta



Rahul Gupta is a FOURTH year student of CSE, R.N.G.P.I.T. He excelled in both his academics and extracurricular activities. His passion for web development is always at the heart of his writings. Dwelling deeply into the web, Rahul wrote an article on Understanding Cypress.io to assist students in broadening their knowledge.

# About Authors

K  
A  
M  
Y  
A  
R  
A  
T  
H  
O  
D



Kanya Rathod, is a FOURTH year student of CSE, R.N.G.P.I.T. She leads the readers with curiosity to read her article “A sneak peek into the world of web dev!”. The use of precise words and concern for the readers demonstrate how she attempted to accompany readers through adversity.

Niva Naik is a FOURTH year student of CSE, R.N.G.P.I.T. When she was in fifth semester, her interest in websites led her to write an article on “How to Make a Simple Website”. With her detailed knowledge of web development, she did her best to lend a hand to other students in starting from scratch and build something.



N  
I  
V  
A  
N  
A  
I  
K

D  
H  
A  
R  
A  
P  
A  
T  
E  
L



Dhara Patel is an ALUMNI of CSE, R.N.G.P.I.T. She is a self motivated, energetic individual who strives for her dreams. Her passion for software development was undeniable. In the eighth semester, she wrote an article on “Introduction to Flutter” to educate students about flutter and its importance in today’s world.

# Table of Contents

1	Basics of Web Development	8-23
2	Introduction to Game Development	24-29
3	Mobile Application Development	30-37
4	Deno replacement or alternative to Node.js	38-45
5	Understanding cypress.io: An automated testing library	46-51

# BASICS OF WEB DEVELOPMENT

- Kamyra Rathod & Niva Naik

(FOURTH year, CSE, R.N.G.P.I.T.)



In today's technological world, almost everyone has internet connection in some way, and a large majority of individuals utilise it on a daily basis. Web development is quickly becoming one of the most appealing and well-paid professions in today's IT sector.



# But what is a web developer and what exactly does one do?

A web developer is someone who converts a web design (made by a client or a design team) into a website. They accomplish this by writing a large amount of complex code in a number of languages. Web developers have a challenging job because they must transform a language that we understand, such as English, into a language that a computer can understand Like HTML.

Even though there are no formal educational requirements, dealing with web developing projects requires those who wish to be referred to as web developers to have advanced knowledge/skills in HTML/XHTML, CSS, JavaScript and jQuery,Server/client side architecture like all or some of the above mentioned, Programming/Coding/Scripting in one of the many server-side languages or frameworks (e.g., Perl, Python, Ruby, PHP, Go, CFML - ColdFusion, Java, ASP, ASP.NET, Node.js), ability to utilize a database and creating single page application with use of front-end tools such as EmberJS, ReactJS or AngularJs.

To comprehend what a web developer is, you must first realise that there are three sorts of web developers: front-end, back-end, and full-stack. Let's have a glimpse at all of the three one by one.

## Importance of web development

First impression is the last impression. If your website doesn't create a good impression on the clients, they are not going to connect with the business.Your website should reflect your business philosophy. It should support you in growing revenues and attracting new customers while reinforcing your credibility and expertise.

You have to gain the outreach as a company while pertaining to the baasic marketing strategy of delivering the better amongst the best. An unresponsive site or one that looks outdated will decrease your chances to attract new customers. Customers want to work with companies that can demonstrate their success and make it easy for them to find what they're looking for.

# Front-end Developer

A front-end developer is a person who takes a client's or design team's website design and creates the code necessary to put it online. A good front-end web developer should be able to code in at least three languages: HTML, CSS, and JavaScript.

**HTML** stands for **Hyper Text Markup Language**. We can create a static website by HTML only. Technically, HTML is a Markup language rather than a programming language. Let's see what is meant by Hypertext Markup Language, and Web page.

**Hyper Text:** Hyper Text simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage.

**Markup language:** Markup language makes text more interactive and dynamic.

**Web Page:** A web page is a document which is commonly written in HTML and translated by a web browser. You can reach to it by entering the URL in search bar.

Now note that HTML document is made of many HTML tags and each HTML tags contains different content. Listed are some basic tags that are used while using html.

It is platform independent because it can be displayed on any platform like Windows, Linux, etc.

It facilitates the programmer to add Graphics, Videos, and Sound to the web pages which makes it more attractive and interactive.

**<!DOCTYPE>** defines the document type or it instruct the browser about the version of HTML.

**<html>** tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except **<!DOCTYPE>**.

**<head>** should be the first element inside the **<html>** element, which contains the title of webpage.

**<title>** is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately.

**<body>** Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

**<h1>** describes the first level heading of the webpage.

**<p>** describes the paragraph of the webpage.

**CSS** stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. CSS is used along with HTML and JavaScript in most websites to create user interfaces for web applications and user interfaces for many mobile applications.

You can add new looks to your old HTML documents. You can completely change the look of your website with only a few changes in CSS code.

These are the two major benefits of CSS:

1. Save your time and reduce the length and repetitions in code. Before CSS, tags like font, color, background style, element alignments, border and size had to be repeated on every web page. This was a very long process. For example: If you are developing a large website where fonts and color information are added on every single page, it will become a long and expensive process. CSS was created to solve this problem.

2. It has more features or attributes than HTML. CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

A CSS rule set contains a selector and a declaration block.

**Selector:** Selector indicates the HTML element you want to style. It could be any tag like `<h1>`, `<title>` etc.

**Declaration Block:** The declaration block can contain one or more declarations separated by a semicolon. For the above example, there are two declarations:

```
color: yellow;  
font-size: 11px;
```

Each declaration contains a property name and value, separated by a colon.

**Property:** A Property is a type of attribute of HTML element. It could be color, border etc.

**Value:** Values are assigned to CSS properties. In the above example, value "yellow" is assigned to color property.

**Internal:** An internal CSS is used to define a style for a single HTML page.

**Inline:** An inline CSS is used to apply a unique style to a single HTML element.

**External:** An external style sheet is used to define the style for many HTML pages. With an external style sheet, you can change the look of an entire web site, by changing one file.

# JavaScript

It is a loosely-typed client side scripting language that executes in the user's browser. JavaScript interact with html elements (DOM elements) in order to make interactive web user interface. It basically shows the behavior of a web page.

As a client-side scripting language tailored to the browsing experience it is made to support all major browsers with full integration of HTML/CSS. Its description as a scripting language means code can be written right into HTML and executed at run-time without the need for prior compilation. Ultimately this reduces the load on the server each time you run a project with JavaScript. JavaScript code is also object-oriented, meaning front-end developers can create objects that can be manipulated. Objects contain data that can be modified which helps coders manage the different elements of an application. These unique characteristics all contribute to JavaScript's success.

JavaScript's primary use is to transform web pages from static to dynamic. Think back to the last interactive experience you had on the web.

You can probably credit JavaScript with fabricating their experience.

Adding behavior to web pages is at the core of JavaScript's pervasiveness. Many of these behaviors have been mentioned.

Sliders, dropdown menus, animations, audio and video, can all be categorized as behavior changes.

Here are some other examples of JavaScript's implementation towards web page behavior:

- Showing or hiding information
- Zooming in/out
- Displaying a timer or countdown
- Gallery carousels on homepages

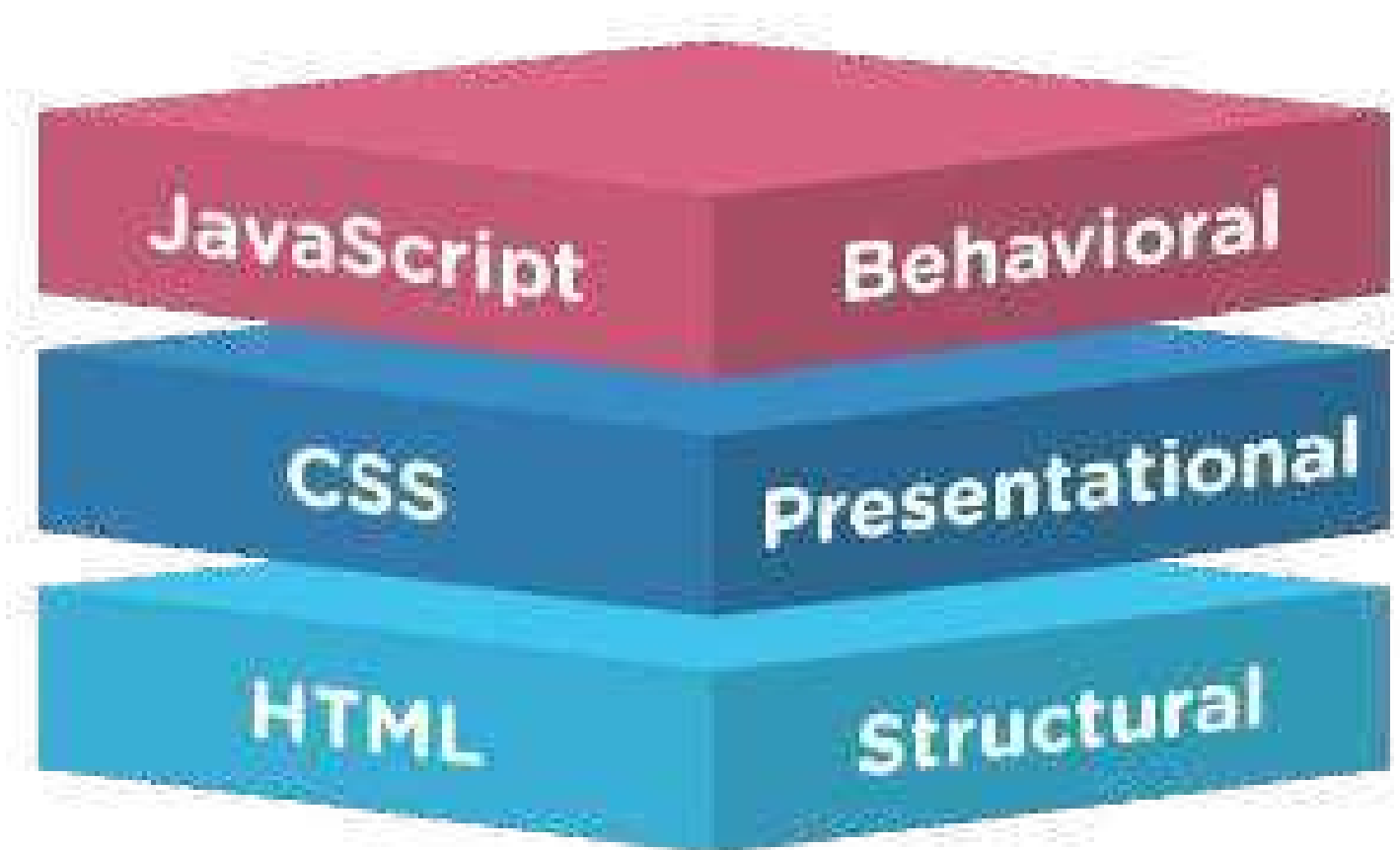
## Relationship between HTML, CSS, JavaScript

Lets consider a web page as a human body. Now let's see what each of these language really do:

HTML is a structural language that build the structure of a website as the skeletal system form the structure of the human body. Heading, paragraphs, images, text all are the part of HTML that creates the basic structure of a website.

CSS is a styling language that gives styling of a website. With the help of front color, background color and border styling CSS furnishes the look of the site as the skin gives look to the human body.

JavaScript is a programming language that gives motion and logics to the website for example a popup window alert. It is just like the motion of human body.



## HTML Tags

### Div Section

`<div class="name" id="id">`

### Headings

`<h1>Heading 1</h1>`  
`<h2>Heading 2</h2>`  
`<h3>Heading 3</h3>`  
`<h4>Heading 4</h4>`  
`<h5>Heading 5</h5>`  
`<h6>Heading 6</h6>`

### Paragraph

**Syntax:** `<p>Text to be displayed</p>`

**Line break:** `<br>`

**Horizontal line:** `<hr>`

### Comment

`<!-- HTML Comment -->`

### Image

``

**Background Image:** `<div style="background-image: url('image.jpg');">`

### Formatting

**Bold text:** `<b>Text to be displayed</b>`

**Italic text:** `<i>Text to be displayed</i>`

**Underlined text:** `<u>Text to be displayed</u>`

**Abbreviation:** `<abbr title="description" data-bbox="275 585 487 596">Text to be displayed</abbr>`

### Attributes

**Image:** `<img alt="description of image" data-bbox="175 655 487 666">`

**Link:** `<a href="http://www.example.com" data-bbox="175 675 487 686">Text to be displayed</a>`

**Title:** `<title>Text to be displayed</title>`

**Style:** `<div style="background-color: red; color: blue;" data-bbox="175 755 487 766">`

**Lang:** `<code data-bbox="175 785 487 796">Text to be displayed</code>`

### Block and Inline

**Block:** `<div style="width: 100%; height: 100px;" data-bbox="175 865 487 876">`

**Inline:** `<span style="display: inline-block; width: 100px;" data-bbox="175 885 487 896">`

## HTML Tags

### Quotation

`<quotation>Text to be displayed</quotation>`

### Video

`<video width="320" height="240" data-bbox="535 205 896 216"><source src="video.mp4" type="video/mp4"></video>`

`<audio src="audio.mp3" type="audio/mp3" data-bbox="535 235 896 246"></audio>`

### Audio

`<audio src="audio.mp3" type="audio/mp3" data-bbox="535 315 934 326"></audio>`

`<table border="1" data-bbox="535 345 934 356"><tr><td>Table content</td></tr></table>`

### Table Structure

```
<table border="1" data-bbox="535 405 926 416">
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Body 1,1</td>
      <td>Body 1,2</td>
    </tr>
    <tr>
      <td>Body 2,1</td>
      <td>Body 2,2</td>
    </tr>
  </tbody>
</table>
```

### List

**Ordered List:** `<ol data-bbox="535 785 850 796"><li>Item 1</li><li>Item 2</li></ol>`

**Unordered List:** `<ul data-bbox="535 865 850 876"><li>Item 1</li><li>Item 2</li></ul>`

# CSS Basics

## Syntax

selector {  
 property: value;  
 property: value;  
}

## Include External CSS file

<link rel="stylesheet" type="text/css" href="style.css"/>

## Internal Styles

<style type="text/css">  
 selector {property: value;}  
</style>

## Class and ID

id Selector (e.g. #id)  
 Class Selector (e.g. .class)

## Classifiers

id selector {  
 class selector {  
 universal selector {  
 class selector {  
 attribute selector {  
 pseudo-class {  
 pseudo-element {

class selector {  
 id selector {  
 attribute selector {  
 pseudo-class {  
 pseudo-element {

## Selectors

*	All elements
div	div elements
div p	p elements inside div
div p	paragraphs inside div
div p	all p tags, whether they are directly inside div or not
div p	p tags immediately after div
div p	p tags preceded by div
div p	p tags with class
div p	p tags with id
div class="name"	divs with the class name
div class="name"	divs with the class name
div class="name"	divs with the class name
div class="name"	divs with the class name

## Attribute Selectors

a[lang="en"]	all a tags with lang="en" attribute
a[lang="en"]>B[lang="en"]	all B tags with lang="en" attribute inside a tag with lang="en" attribute
a[lang="en"]	all a tags with lang="en" attribute
a[lang="en"]	all a tags with lang="en" attribute
a[lang="en"]	all a tags with lang="en" attribute
a[lang="en"]	all a tags with lang="en" attribute
input[type="text"]	all text input type

# CSS Basics

## Text

font-family	font face (e.g. Helvetica, Arial)
font-size	font size (e.g. 12px, 2em)
font-weight	font weight (e.g. 400, 700, 900)
font-style	font style (e.g. italic, oblique)
font-variant	font variant (e.g. small-caps)
font-size-adjust	font size adjust (e.g. 1.2)
line-height	line height (e.g. 1.2)
letter-spacing	letter spacing (e.g. 0.2em)
text-indent	text indent (e.g. 2em)
text-align	text alignment (e.g. left, right, center)

## Positioning

position	positioning (e.g. absolute, relative)
float	float (e.g. left, right)
clear	clear (e.g. left, right)
bottom	bottom (e.g. 10px)
display	display (e.g. inline-block)
z-index	z-index (e.g. 1, 2, 3)
overflow	overflow (e.g. hidden, scroll)

## Borders and Lists

border	border (e.g. 1px solid black)
border-top	border-top (e.g. 1px solid black)
border-bottom	border-bottom (e.g. 1px solid black)
border-left	border-left (e.g. 1px solid black)
border-right	border-right (e.g. 1px solid black)
border-style	border style (e.g. solid, dashed)
border-width	border width (e.g. 1px, 2px)
list-style-type	list style type (e.g. none, disc, square)
list-style-position	list style position (e.g. inside, outside)

## Everything Else

background-color	background color (e.g. red, blue)
background-image	background image (e.g. url('image.png'))
background-size	background size (e.g. cover, contain)
background-repeat	background repeat (e.g. repeat, no-repeat)
background-attachment	background attachment (e.g. scroll, fixed)
background-position	background position (e.g. top left, center)
background-clip	background clip (e.g. border, content-box)
background-origin	background origin (e.g. border, content-box)
background-size	background size (e.g. cover, contain)
background-repeat	background repeat (e.g. repeat, no-repeat)
background-attachment	background attachment (e.g. scroll, fixed)
background-position	background position (e.g. top left, center)
background-clip	background clip (e.g. border, content-box)
background-origin	background origin (e.g. border, content-box)

## JS Basics

### On Page Script

```
<script language="JavaScript">
</script>
```

### Include External JS file

```
<script src="file name.js"></script>
```

### Delay - 1 second timeout

```
setTimeout(function() {
// Do something
}, 1000);
```

### Functions

```
function addNumbers(a, b) {
return a + b;
}
var addNumbers = addNumbers;
```

### Output

```
console.log(a);
document.write(a);
alert("Hello");
prompt("Enter age", 25);
```

### Comments

```
// This is a
comment //
/* This is a
comment */
```

### If - Else Statement

```
if (age >= 18) {
// do something
} else {
// do something else
}
```

### Switch Statement

```
switch (new Date().getDay()) {
case 0:
// is it Sunday?
break;
case 1:
// is it Sunday?
break;
default:
// is it otherwise?
}
```

## JS Loops

### For Loop

```
for (var i = 0; i < 10; i++) {
document.write(i + " " + i + " ");
}
var sum = 0;
for (var i = 0; i < 10; i++) {
sum += i;
}
// printing an array
var arr = [];
for (var i = 0; i < 10; i++) {
arr.push(i * 2);
}
```

### While Loop

```
var i = 0;
while (i < 10) {
// Do something
}
document.write(i + " ");
```

### Do While Loop

```
var i = 0;
do {
// Do something
} while (i < 10);
```

### Break

```
for (var i = 0; i < 10; i++) {
if (i == 5) {
break;
}
document.write(i + " ");
}
```

### Continue

```
for (var i = 0; i < 10; i++) {
if (i == 5) {
continue;
}
document.write(i + " ");
}
```

## Global Functions()

```
isNaN();
parseFloat();
parseInt();
Math.ceil();
Math.floor();
Math.random();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
encodeURIComponent();
```



## JSON

```

const obj = {
  firstName: 'John',
  lastName: 'Doe',
  age: 40,
  isMarried: true,
  address: {
    street: '123 Main St',
    city: 'New York',
    state: 'NY',
    zip: 10001
  }
};
    
```

### Send

```

fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(obj)
});
    
```

### Storing and Retrieving

```

// Store object to localStorage
localStorage.setItem('obj', JSON.stringify(obj));

// Retrieve object from localStorage
const storedObj = localStorage.getItem('obj');
const obj = JSON.parse(storedObj);
    
```

## Errors

```

try {
  undefinedMethod();
} catch (err) {
  console.log(err.message);
}
    
```

### Throw error

```
throw new Error('message');
```

### Input validation

```

const validateEmail = (email) => {
  if (!/^[\w-]+@[\w-]+\.[\w-]+$/.test(email)) {
    return false;
  }
  return true;
};

const validateForm = (form) => {
  // Check if email is valid
  if (!validateEmail(form.email)) {
    return false;
  }
  // Check if password is strong enough
  if (form.password.length < 8) {
    return false;
  }
  return true;
};
    
```

### Error name values

RangeError	A numeric value out of range
ReferenceError	An illegal reference has occurred
SyntaxError	A syntax error has occurred
TypeError	A type error has occurred
URIError	Pattern (URI) error has occurred

## Events

```

document.getElementById('btn').addEventListener('click', () => {
  // Do something
});
    
```

### Mouse

```

click, dblclick, mouseover, mouseout, mousemove, mousedown, mouseup, contextmenu, drag, dragend, dragstart, dragover, dragleave, drop, mousewheel
    
```

### Keyboard

```

keydown, keyup, keypress
    
```

### Frame

```

load, unload, beforeunload, onreadystatechange, onreadystatechange, onreadystatechange, onreadystatechange
    
```

### Form

```

blur, change, click, focus, input, reset, submit, oninput, onblur, onfocus, onreset, onsubmit
    
```

### Drag

```

drag, dragend, dragstart, dragover, dragleave, drop, dragenter, dragexit
    
```

### Clipboard

```

copy, paste, cut
    
```

### Media

```

abort, error, loadeddata, loadedmetadata, loaded, progress, srcchange, timeupdate, volumechange, play, pause, seeking, stopped, timeupdate, volumechange
    
```

### Animation

```

animationend, animationiteration, animationstart
    
```

### Miscellaneous

```

beforeunload, onbeforeunload, onmouseover, onmouseout, onmousemove, onmousedown, onmouseup, oncontextmenu, ondrag, ondragend, ondragstart, ondragover, ondragleave, ondrop
    
```

# Backend Developer

While front-end developers are responsible for client-side programming, back-end developers have to deal with the server-side. Back-end developers usually look after the databases & servers. Back-end developers use a variety of server-side scripting languages like PHP, Python, Java, and Ruby, while SQL is commonly used to manage and analyze data in website databases.

Back-end developers must have in-depth knowledge about the languages and they must be able to fix the errors and should know how to maintain the websites.

## Django:

Django web development framework has everything that it takes to develop a full-fledged application out-of-the-box as it adopts Python's "batteries included" approach. Since all the essentials are available, you don't need to spend hours in customizing the framework while developing a simple application or a prototype.

However, if you require additional features for building a more complex application, over 4,000 packages are available for Django that cover debugging, profiling, and testing.

The Django framework also leverages tool packages that allow developers to work with cutting-edge technologies such as Artificial Intelligence (AI), machine learning, and data analytics. Also, these tool packages are easy to set up and use in projects especially for math-heavy industries such as FinTech.

Python and Django are intertwined but not the same. Python is a programming language that's used for many different applications: artificial intelligence, machine learning, desktop apps, etc. On the other hand, Django is a Python framework for full-stack web application development and server development.

## **Node.js:**

Node.js is a so-called framework, a runtime environment for running JavaScript code on the back of a browser, or outside of it. It is a server framework which is free license technology and suitable for different platforms.

Node.js is great for real-time working with a large amount of information. Also, as HTTP requests and responses are basically a stream of large files, Node.js enables collecting and visualizing data as dashboards. For example, GraphQL can be used for this purpose. It works along with many programming languages but exactly Node.js made GraphQL a new standard. Generally, Node.js backend development is so concise because it provides many features that are impossible with other event-based environments.

### **Companies using Node.js:**

Airbnb, Codecademy, HotelTonight, eBay, Square, Asana, Uber.

## **PHP:**

PHP is a server-side scripting language designed specifically for webdevelopment. Since PHP code executed on the server-side, so it is called a server-side scripting language.

PHP can be used to Collect form data,Generate dynamic page content,Send and receive cookies,Write command line scripting, Write server-side scripting,Write desktop applications.

Since PHP is dynamically typed, it means you're able to come up with a variety of solutions and workarounds for one problem. It also means that the same bit of code can mean something different depending on the context, which makes programs written in PHP tricky to scale and sometimes slow to run.

### **Companies using PHP:**

Facebook, Lyft, Mint, Hootsuite, Viber, BufferDocuSign.

# Fullstack Developer

A full stack developer has understanding of both Front-end and back-end processes. They can oversee the whole process of a website and should be able to implement both. For becoming a full stack developer, you should have in-depth knowledge of both front-end & back-end. Job opportunities for a full stack developer increases more than a front-end & back-end developer.

## MEAN Stack:

Experts consider MEAN technology the best for web development due to its various benefits. It's comprised of MongoDB (a NoSQL DB), Express.js (a backend web framework), Angular (a front-end framework), and Node.js (an open-source cross-platform server), and can be used for developing complex mobile and responsive web applications.

The single language used throughout this stack is JavaScript. Its components are JSON savvy and excellent in data transmission with free module library access. This means web developers can reuse this code across the whole app without reinventing the wheel. Knowledge of JavaScript is sufficient to work with this web development technology stack. The stack also aids in developing fast, highly efficient, and scalable software applications.

### Advantages of MEAN

- Highly flexible.
- Cost-effective.
- Open-source.
- Easily switch between client and server.
- Excellent for real-time web applications.
- Time-saving.

### Disadvantages of MEAN

- Lack of widespread support.
- Security exploits.

## **MERN stack:**

MERN is nearly identical to MEAN with a bit of technological change, where Angular is exchanged with React. The main benefit of using MERN is the integration of React and its powerful library and capability to use code simultaneously on servers and browsers. Additionally, it has phenomenal full-stack development (front-end and backend) possibilities. React utilizes JavaScript XML and Virtual DOM, and these components work and implement changes seamlessly.

React is a popular framework known for its flexibility and performance oriented approach, enabling the building of top-end single-page apps with interactive interfaces. The MERN technology stack comes with an extensive suite of testing tools and is open source with community backing. It is the second most popular web technology stack of 2021.

### **Advantages of MERN**

- UI rendering and performance.
- Cost-effective.
- Open-source.
- Easily switch between client and server.

### **Disadvantages of MERN**

- Not fit for large scale applications.
- Lower productivity.

## **METEOR.js Stack:**

Meteor.js is an open-source website development platform and makes JavaScript app development much faster for desktop, mobile, and web. You can integrate it with popular frameworks, tools, and technologies that you are already using. You can use the same code to develop apps for iOS, Android, desktop, and the web.

With Meteor.js, you get an integrated JavaScript tech stack that ranges from the applications' database to the viewer's screen. It means that what one can achieve in a few lines might not have been possible in more than 100 lines with another framework. The majority of web developers across the globe are using Meteor.js to develop scalable applications. This stack is supported by a large community of developers.

### Advantages of Meteor.js

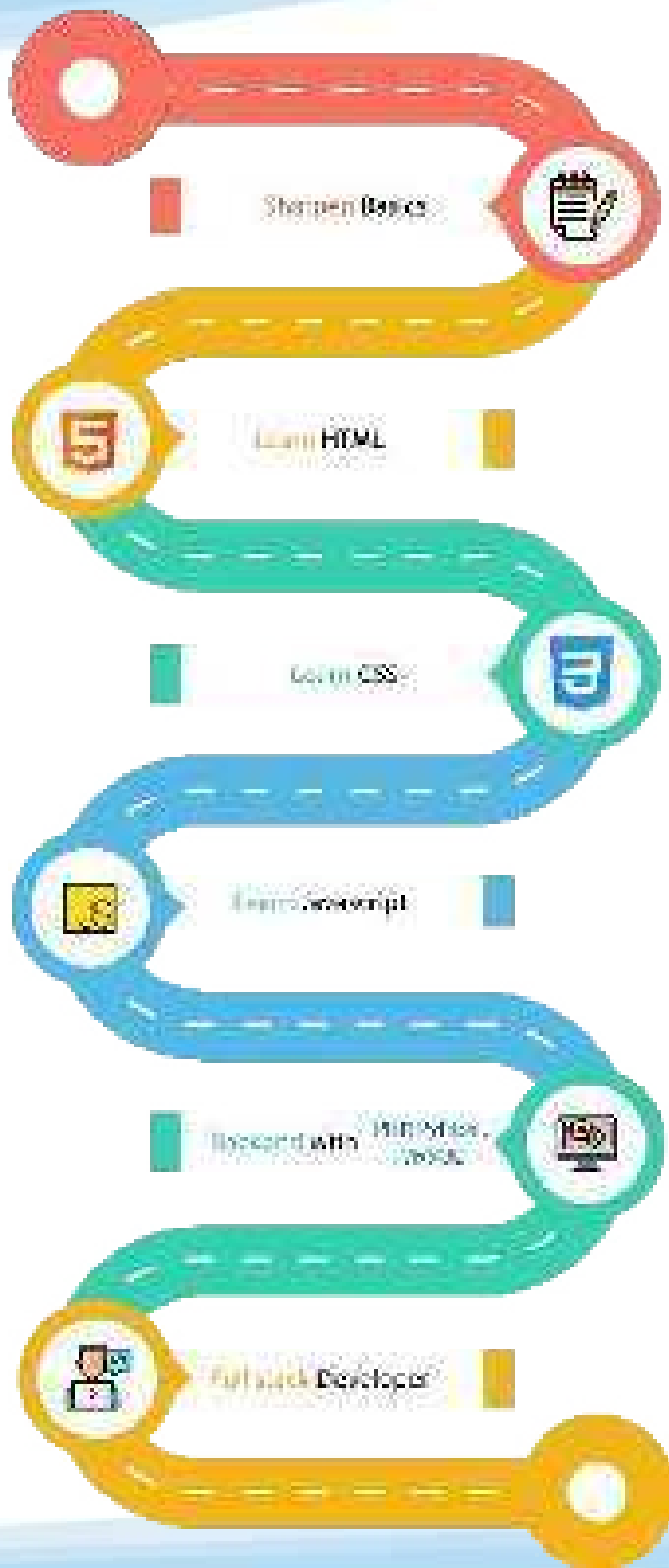
- Simplicity.
- Seamless client-server communication.
- Real-time testing tools.
- Debugging.
- An abundance of packages and libraries.

### Disadvantages of Meteor.js

- Lacks SSR support.
- Problems with data integrity.
- Absence of a native widget library.
- Lacks built-in support for PWAs.



# Web-development Roadmap



# Introduction to Game Development

- Deep Mevada

(FOURTH year, CSE, R.N.G.P.I.T.)



## What is Game Development??

Game Development is the art of creating games and describes the design, development and release of a game. It may involve concept generation, design, build, test and release. While you create a game, it is important to think about the game mechanics, rewards, player engagement and level design.

Game Development can be undertaken by a large Game Development Studio or by a single individual. It can be as small or large as you like. As long as it lets the player interact with content and is able to manipulate the game's elements, you can call it a 'game'.



To get involved in the Game Development process, you do not need to write code. Artists may create and design assets, while a Developer might focus on programming a health bar. A Tester may get involved to see that the game works as expected.

Have you ever considered creating video games? Have you ever played a game and thought to yourself, "I could do better."? Have you ever wished to create fantastic games despite having no interest in coding?

Well, you are in the right place to start your game development journey.

The total process of making a video game is known as game development. And if you think designing a video game is as simple as playing one, think again!!! Many elements come together in the creation of a video game, such as the story, characters, audio, art, lighting, and so on, to create an entirely new universe. But first, decide what you want to be

Game Development industry offers a lot of roles to select from.

**Game Designers** creates game-play, rules and structure of the game.

**Game Artists** creates game art

**Game Programmer** writes logic within the game

**Level Designer** creates different levels and designs them

**Game Tester** Who tests the flow of the game and finds bugs  
And Many more...

Above are the most basic roles, but these roles comprised of roles inside them as well. For example, Game Designer consists of roles like Lead Designers, Content Designers, System Designers, UI Designers, Environment Designers, etc.

Game Designers are often confused as people who create game assets like art, models, textures, etc. Game Artists are the people who create the assets given above.

Game Designers are the people who create the concepts of the game, they define rules and flow of the game.

# So, How do you make Games ?

To make games you don't require any programming knowledge, but it is guided that you have some knowledge.

We'll discuss both aspects of game development — Without Programming and With Programming.

## Game Development without any Programming Knowledge!

To start making games, without any programming knowledge there are some tools that help people to start making games rather than worrying about programming.

These tools are nothing but **Game Engines**.

Game Engine is a tool that helps you build and create games, rather than worrying about basic stuff like memory management, input, graphics rendering, etc.



**CONSTRUCT 3**

It has drag and drop features along with a lots of rules: for example, walk, run, attack, etc. Along with that they provide a lot of good tutorials on how to make a game without any coding knowledge. The best feature is that it works on the browser, yes you heard me right, No need to install it on your desktop or anything.

It's an open-source and free game engine that helps you create games without any coding knowledge. It works online in the browser.



It is a popular game engine that helps you build game, without even writing a single line of code. It has a drag-and-drop feature, used by a large community, and has lots of tutorials. It has an optional scripting feature that lets developers with experience to tweak a little bit around.

# Game Development with any Programming Knowledge!

I am going to assume that you have some knowledge in programming, and you wouldn't be scared if I said some technical terms.

The good thing about knowing programming is that you can personalize your game as you like.

We will be using game engines, because it makes game development even easier by doing all the stuff like memory management, rendering graphics, garbage collection and many other stuff in the background rather than letting you worry about it.



Unity is the most popular game engine for beginners in game development mainly because of its interactive and beginner friendly interface. It uses the C#

language as its scripting language for writing game logic. It has many in-built tools like Animation, Sprite Editor, etc. The vast community of developers makes this engine the best choice for beginners in game development. It also has the great collection of assets which helps developers to get assets for free and focus more on development.



**UNREAL  
ENGINE**

Epic Games has given us many things for free. They also provide this game engine for free. This game engine is very useful for creating wonderful 3D games with stunning graphics that may melt your machine (Just kidding). This engine uses C++ as its programming language for writing game logic. It also has a great tool called Blueprints which is nothing but Visual Scripting.

What it does is that without writing a line of code, you can create visually stunning and great games. It also has a great community of developers and artists. The Unreal Marketplace is a great place to find assets for your game. Games made using Unreal : **PUBG, Batman Arkham City, Bioshock, etc.**

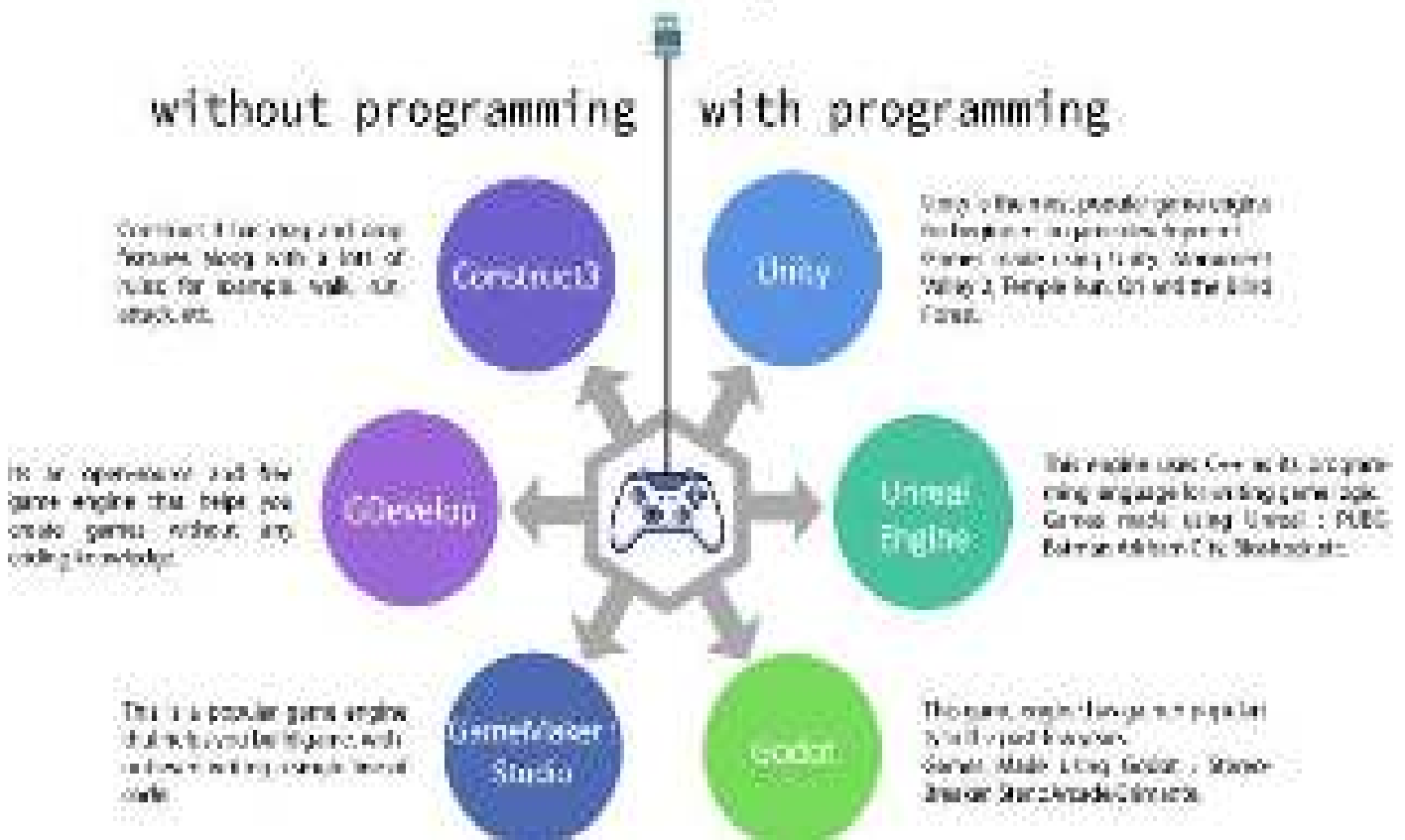
This game engine has gained popularity in the past few years. It is free and open-source, which makes it a good choice for beginners. It also has its own built-in scripting language, GDScript, a high-level, dynamically typed programming language very similar to Python. Learn more about GDScript here. It is beginner-friendly and very helpful for developers with experience. Some Games Made using Godot : **StereoBreaker**, **StenoArcade**, **Grimante**.



# GODOT

Game engine

## Game Development



“The Game gives you a Purpose. The Real Game is to Find a Purpose.”

- Vinçet Raj Kapoor

# Mobile Application Development



**DIGITALIZATION AT YOUR  
FINGER TIPS**

# What is Mobile application development?

Mobile application development is the process of creating software applications that run on a mobile device, and a typical mobile application utilizes a network connection to work with remote computing resources. Hence, the mobile development process involves creating installable software bundles (code, binaries, assets, etc.) , implementing backend services such as data access with an API, and testing the application on target devices.



Mobile applications are becoming increasingly sophisticated, interactive and user friendly as companies line up to join the mobile revolution. To meet demand, application framework developers have stepped up their game, providing superior technologies for app development.

There are two dominant platforms in the modern smartphone market. One is the iOS platform from Apple Inc. The iOS platform is the operating system that powers Apple's popular line of iPhone smartphones. The second is Android from Google. The Android operating system is used not only by Google devices but also by many other OEMs to build their own smartphones and other smart devices.

Although there are some similarities between these two platforms when building applications, developing for iOS vs. developing for Android involves using different software development kits (SDKs) and different development toolchain. While Apple uses iOS exclusively for its own devices, Google makes Android available to other companies provided they meet specific requirements such as including certain Google applications on the devices they ship. Developers can build apps for hundreds of millions of devices by targeting both of these platforms.

# What is Android application development?

Android software development is the process by which applications are created for devices running the Android operating system. Google states that[3] “Android apps can be written using Kotlin, Java, and C++ languages” using the Android software development kit (SDK), while using other languages is also possible. All non-JVM languages, such as Go, JavaScript, C, C++ or assembly, need the help of JVM language code, that may be supplied by tools, likely with restricted API support.

Some programming languages and tools allow cross-platform app support (i.e. for both Android and iOS). Third party tools, development environments, and language

support have also continued to evolve and expand since the initial SDK was released in 2008. The official Android app distribution mechanism to end users is Google Play; it also allows staged gradual app release, as well as distribution of pre-release app versions to testers.

Mobile applications are becoming increasingly sophisticated, interactive and user friendly as companies line up to join the mobile revolution. To meet demand, application framework developers have stepped up their game, providing superior technologies for app development.

## What is an Application Framework?

An Android application framework is a software toolkit that enables app developers to piece together a finished product that meets the requirements of its proprietor. A framework provides the bones of an application, to be fleshed out with graphics, animation, special features and functionality. Application frameworks are designed to simplify the app development process, and make it easy to manage, modify and fix bugs down the road. Some of the major application framework are **React native by Facebook, Xamarin by Microsoft and Flutter by Google**. Hereby we will have a little look at the Flutter.



# Creating with Flutter:

**-Dhara Patel**

(ALUMNI, CSE, R.N.G.P.I.T.)

I've been hearing about how amazing Flutter is and I've decided to try it out to learn something new.

Flutter is an open-source cross-platform mobile development framework developed by Google. It is used to develop applications for Android, iOS, Windows, Mac, Linux, Google Fuchsia, and the web.

I noticed that reading the documentation on Dart, Flutter, and all of its widgets wouldn't be a good idea as it would be too time-consuming. So, I thought it would be amazing to have a short guide for the beginners to get started with flutter.

## ***Installations:***

One of the best things about Flutter is that it lets you use your favourite IDE for development. Flutter is relatively easy to set up and depending on what OS you're using; you can check out the steps in this official Flutter tutorial: <https://flutter.dev/docs/get-started/install>. It has first-class support for **Android Studio, IntelliJ IDEA, and VS Code**.

## ***Creating the First App with Flutter:***

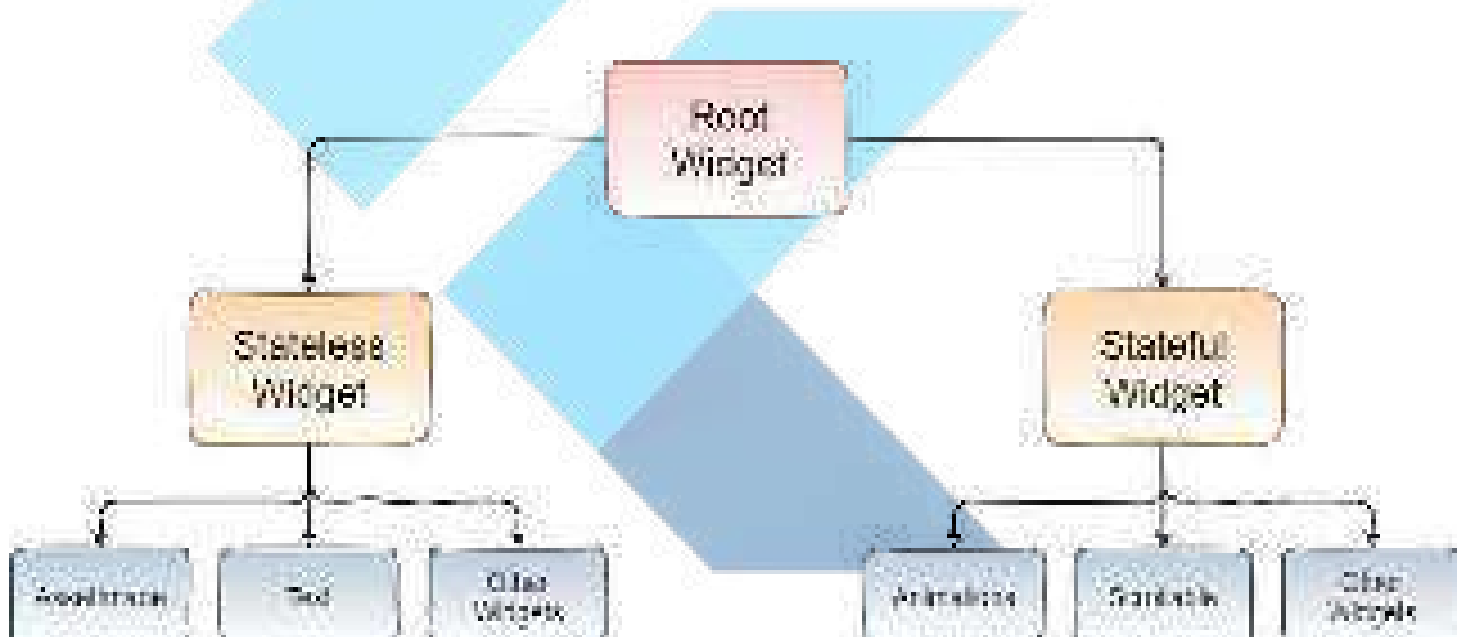
1. Invoke View > Command Palette.
2. Type "flutter", and select the Flutter: New Project.
3. Enter a project name, such as FlutterDemo, and press Enter.
4. Create or select the parent directory for the new project folder.
5. Wait for project creation to complete and the main.dart file to appear.

## Widgets in flutter:

*Remember, in Flutter everything is a Widget.*

Widgets are the basic building blocks of a Flutter app's user interface. Widgets describe what their view should look like given their current configuration and state. When the state of the widget changes, widget rebuilds its description to show the minimal changes in the widget tree to transition from one state to another.

Widgets form a hierarchy based on the composition they work with. Each widget can be nested inside another widget and inherits properties from its parent widget. The root widget handles the state of the app.



## Creating our first application:

CODE

```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(MyApp());
5 }
6
7 class MyApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10    return Scaffold(
11      appBar: AppBar(
12        title: Text('Flutter Basics'),
13      ),
14      body: Container(
15        margin: EdgeInsets.all(16),
16        color: Colors.blueGrey,
17        child: Center(child: Text('Welcome!')),
18      ),
19    );
20  }
21 }
```

OUTPUT



That's It !!!  
WOHOOO you have  
created your first flutter app!!!

# What is iOS application development?

According to IBM, iOS application development is the process of making mobile applications for Apple hardware, including iPhone, iPad and iPod Touch. The software is written in the Swift programming language or Objective-C and then deployed to the App Store for users to download.

If you're a mobile app developer, you may have had reservations about iOS development. For example, each developer needs a Mac computer—and Macs are generally more expensive than their Windows-based counterparts. In addition, once you complete your app, it faces a stringent quality review process before it can be distributed through the App Store.

Nevertheless, if your organization's employees, customers or partners are among the hundreds of millions of Apple iPhone and iPad users around the world, you have obvious reasons to engage in iOS app development. And despite potentially high barriers to entry, developing an iOS app can be as easy as (in some cases easier than) developing for Android. With proper planning and the right resources, you can join the ranks of iOS app developers.

## What is swift?

According to apple, Swift is a powerful and intuitive programming language for iOS, iPadOS, macOS, tvOS, and watchOS. Writing Swift code is interactive and fun, the syntax is concise yet expressive, and Swift includes modern features developers love. Swift code is safe by design, yet also produces software that runs lightning-fast.

Swift can open doors to the world of coding. In fact, it was designed to be anyone's first programming language, whether you're still in school or exploring new career paths. For educators, Apple created free curriculum to teach Swift both in and out of the classroom. First-time coders can download Swift Playgrounds—an app for iPad that makes getting started with Swift code interactive and fun.

Creating ios application using xcode: The Xcode IDE is at the center of the Apple development experience. Tightly integrated with the

Cocoa and Cocoa Touch frameworks, Xcode is an incredibly productive environment for building apps for Mac, iPhone, iPad, Apple Watch, and Apple TV.

Creating first project: Launch Xcode, then click “Create a new Xcode project” in the Welcome to Xcode window or choose File > New > Project. In the sheet that appears, select the target operating system or platform and a template under Application. In the following sheets, fill out the forms and choose options to configure your project. You must provide a product name and organization identifier because they are used to create the bundle identifier that identifies your app throughout the system. Also enter an organization name. If you don’t belong to an organization, enter your name.

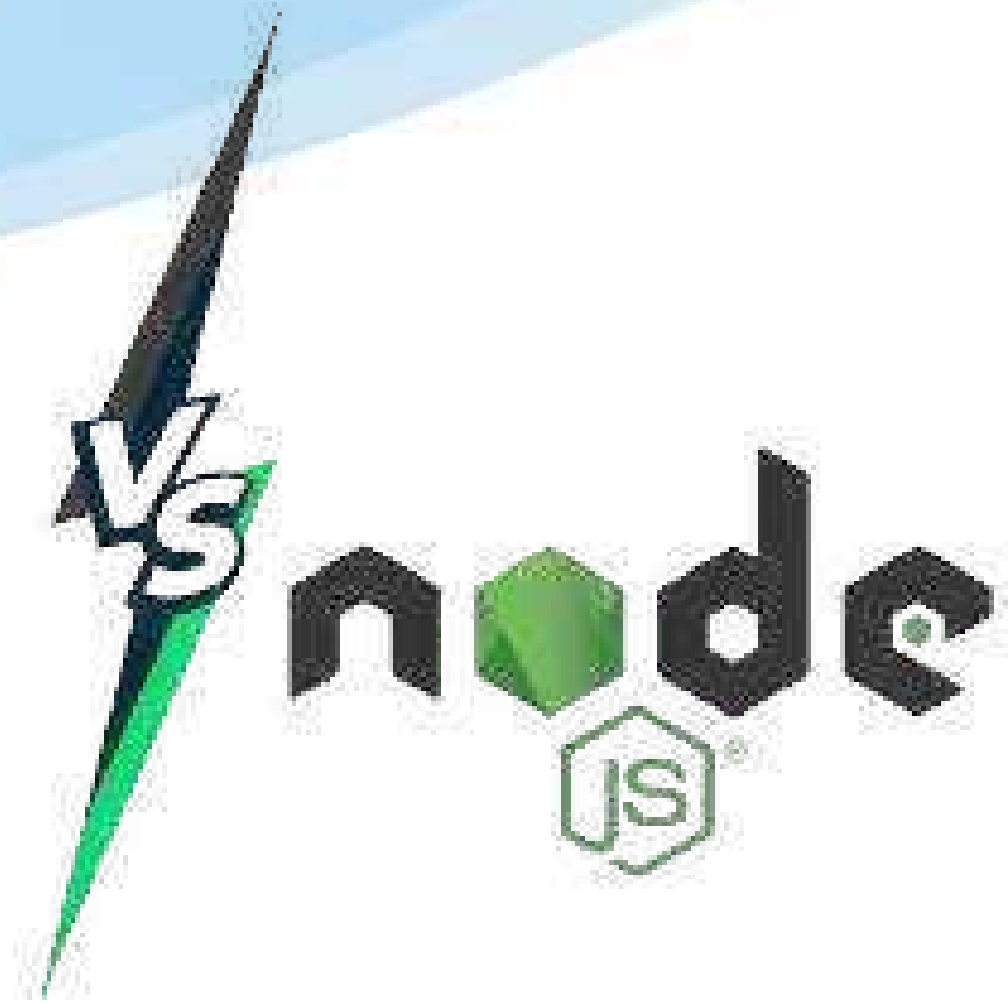


To develop for all platforms and see an interactive preview of your layout, choose SwiftUI as the user interface before you click Next on this sheet.



# Deno replacement or alternative to Node.js

- Raj Kharvar  
(FOURTH year, CSE, R.N.G.P.I.T.)



## Let's see

Before comparing Deno and Node.js, Let's talk about some history of Javascript.

JavaScript was created by **Brendan Eich** in **1995** during his time at Netscape Communications. Javascript was the language that only runs on the web browser. There was no runtime environment that could allow you to run the Javascript code without a web browser. In **2009**, **Ryan Dahl** introduced the Javascript run-time environment - **Node.js**.

Node.js is an open-source, cross-platform, providing Javascript runtime environment that executes Javascript code outside of the web browser. Node.js was written in C++.

Node.js was built on top of Google's V8 Javascript engine, the same engine that Chrome uses. V8 compiles JavaScript source code to native machine code at runtime.

With the launch of Node.js, there was no need to learn a separate language to write backend code. Javascript code can be deployed to a server with the help of Node.js. In January 2010, a package manager was introduced for Node.js environment called npm.

## With just Javascript and Node.js. You can create:

**Web applications** using Vue, React, and Angular.

Cross-Platform **Mobile Application** using React-Native, Ionic, and Nativescript.

Cross-Platform **Desktop Application** could be developed using Electron.js.

Training and Deploying of **Machine Learning** possible due to libraries like Tensorflow.js.

And much more in the field **Robotics** and **IoT (Internet of Things)**.

Pretty much everything that you can do while learning two or more languages and frameworks that is possible with just learning Javascript and Node.js.



## System requirements for the Node.js agent

Before installing the Node.js agent confirm you can meet the following requirements:

- There is a deployed application to be analyzed, and the web application technology is supported by Contrast.
- The agent has network connectivity with the Contrast server.

Using the Node.js agent requires increasing the application's available CPU and memory due to the increased processing and analysis of inbound information. Using the Node.js agent will use more resources than your application on its own. CPU load will also increase but this heavily influenced by the specific application architecture and existing CPU usage profile.

## Some of the problems that Node.js have are:

### **Security issues:**

As soon as you install a node program or add the library from npm. It has all the access to I/O, disk, and network.

### **Centralized Package manager:**

npm provides all the packages that are totally centralized. If any of the servers are down you won't be able to download any package.

### **Incompatibility with browser:**

The browser uses the ES Modules System, while Node.js uses CommonJS as a default module system which is not compatible with the browser.

### **No built-in support for Typescript:**

Node.js doesn't provide support for the Typescript out of the box. So a compiler would be needed to convert Typescript to Javascript to run with Node.js.

### **Huge node\_modules bundle size:**

Once project codebase grows this results in huge node\_modules which end up creating a gigantic bundle size.

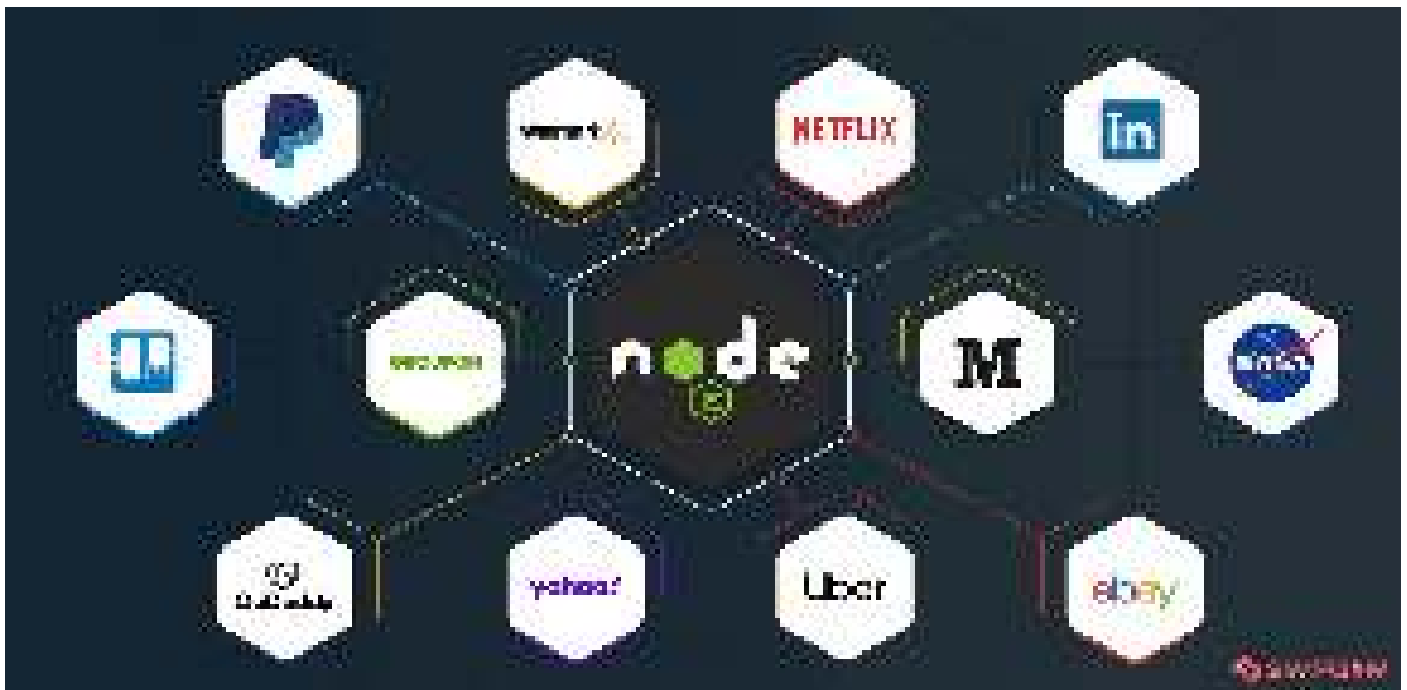


## Companies using node.js

In fact, Node.js is so popular, quite a few major business enterprises are well-acquainted with the software.

Companies that use Node.js include the following:

1. LinkedIn
2. Netflix
3. Uber
4. Trello
5. PayPal
6. NASA
7. eBay
8. Medium
9. Groupon
10. Walmart
11. Mozilla
12. GoDaddy



# Now, let's Talk about Deno



## What is deno?

### ***Rise of Deno***

On May 13, 2018, Ryan Dahl the same person who invented Node.js announced the initial release of another Javascript and Typescript runtime which was Deno. This time it was made in Rust and not in C++ because of the memory safety that Rust provides is out of the box. It is built on top of the Tokio platform and Google's V8 engine that Node.js was built on. Two years later on the same date May 13, 2020, Deno 1.0 was released.

### **Some of the features that Deno provides are:**

#### **Sandbox code:**

- All the code that you write in Deno runs in a sandbox which means that there is no access to I/O, Disk and Network
- If you want to access any of the resources you need to manually grant permission via flag during runtime.

- Some of the flags available are:

```
--allow-env -> Allow environment access
--allow-hrtime -> Allow high resolution time measurement
--allow-net -> Allow network access
--allow-plugin -> Allow loading plugins
--allow-read -> Allow file system read access
--allow-run -> Allow running subprocesses
--allow-write -> Allow file system write access
--allow-all -> Allow all the above permissions
```

## **Built-in support for Typescript:**

Deno provides built-in support for Typescript out of the box. So you don't need an additional compiler setup to convert Typescript to Javascript.

## **Decentralized package manager:**

Unlike npm, you don't need to rely on a centralized package manager. Instead you need to import modules via URL where the code is being hosted. Anyone can host a package just like anyone can host any type of file on the web. When you first import package from URL it is being fetched from URL and stored in the local cache. So no need to pull every time the code changes or being executed.

## **Browser compatibility:**

Deno provides browser compatibility with ES modules. So you don't need webpack anymore to bundle your code.

## **Security:**

With Deno, a developer can provide permission to scripts using flags like `--allow-net` and `--allow-write`. Deno offers a sandbox security layer through permissions. A program can only access the permissions set to the executable as flagged by the user. You're probably asking yourself, "How will I know which flags I have to add to execute the server?" Don't worry; you will get a message in the console log asking you to add a given flag.

## Deno's built-in tooling:

Deno is an entire ecosystem in itself, complete with runtime and its own module/package managing system. This gives it a much greater scope to have all its own tooling built-in.

## Top-level await:

Normally, when using `async/await` in Node.js, you have to wrap your awaits inside of an asynchronous function, and you have to label it `async`. Deno makes it possible to use the `await` function in the global scope without having to wrap it inside an `async` function, which is a great feature.

## Deno was built with:

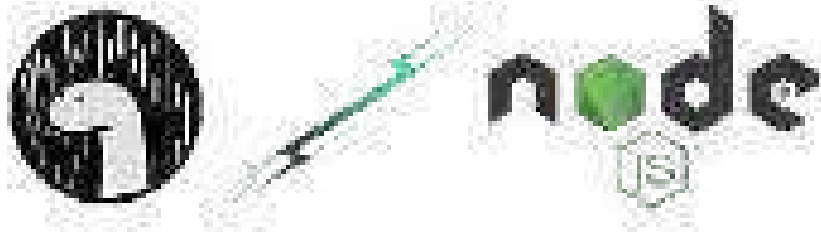
1. **Rust** (Deno's core was written in Rust, Node's in C++)
2. **Tokio** (the event loop written in Rust)
3. **TypeScript** (Deno supports both JavaScript and TypeScript out of the box)
4. **V8** (Google's JavaScript runtime used in Chrome and Node, among others)

## 10 companies reportedly use Deno in their tech stacks:

- Cloudless.
- The Loner Foundation.
- Appwrite.
- Stack goals.
- The Tribe.
- SOLID engineer.
- DenoScript.
- POLAR COP.

## Comparison between Deno js and Node js

- Deno's core written in Rust programming language (core emphasis on safety) and Tokyo. Nodejs had been on C++.
- Typescript a javascript superset designed to remove the programming errors before executing code.
- Typescript is suitable for broad applications.
- ES6 import declarations a decentralized approach to the management of dependencies, whereas Node.js uses CommonJS
- No npm, meaning it doesn't use any package manager like npm and doesn't have the package.json for the module handling.
- Deno removes unmanaged exceptions which are not the case for NodeJS.
- Deno needs to have a clear device, network, and system access permissions.



Category	Node.js	Deno
Language	JavaScript	TypeScript, JavaScript
Packages	NPM	.js or .JS modules
Security & Permissions	No Mechanisms	CU Flags
Code Engine	V8 JavaScript	V8 JavaScript
Security	Full Access	Permissioned Access
TypeScript Support	Not Built In	Built In
Community Support	Strong and Huge	Relatively New
Importing Packages	CommonJS style	ES Modules
Async	Callbacks	Promises
Browser Support	Vague	Support Provided

# Understanding Cypress.io: An Automated Testing Library

- Rahul Gupta  
(FOURTH year, CSE, R.N.G.P.I.T.)



## Introduction:

One of the most important phase of software development is its comprehensive Testing. Writing and fully implementing an end-to-end test comes with many benefits, the most important one being able to determine whether or not a web application works as intended for the user. Additionally, end-to-end tests ensure that new functionality does not break previously developed features. These concerns are crucial to every enterprise web application and shouldn't be overlooked.

To perform End-to-End test of an application manually can be cumbersome, this is why most of developers tend to skip or ignore it. There needs to be an automated testing system which can save developer from this long and slow process, that's when Cypress.io comes into picture.

## What is Cypress.io ?

Cypress is a JavaScript end-to-end testing framework that allows you to write tests that run in a browser much like Selenium. The key difference is that it is easy to set up, has no dependencies and

is a pleasure to write tests with and use. What you get with Cypress is a tool that makes it simple to set up, write, run, and debug tests. Cypress is built and optimized as a tool for local development. If you're familiar with tools like Mocha and Chai, you've already got a head start. Cypress makes use of these tools under the hood but packs on many more features. With Cypress, you get to test a web application from the perspective of your end users.

## Getting Started:

Lets take an example of a TODO application and run some test to check its functionality. Run following command in your CLI at root folder of your project.

## Install Cypress

```
$ npm install cypress --save-dev
```



After running the above command a folder named cypress and cypress.json file will be added to your root folder. Now let's understand its components.

### Fixtures:

Fixtures are used as external pieces of static data that can be used by your tests. You would typically use them with the `cy.fixture()` command and most often when you're stubbing Network Requests.

### Plugins :

It is used to load plugins, by default Cypress will automatically include the plugins file before every single spec file it runs.

### Integration :

Test files are located in this folder. It contains some default tests. To write custom test, create a new file like `app_spec.js` inside this folder. You'll spend most of time in this folder only.

### Support :

The support file is a great place to put reusable behavior such as Custom Commands or global overrides that you want applied and available to all of your spec files.



## **cypress.json :**

Here you can define configurations like `projectId`, `baseUrl`, `defaultCommandTimeout`, `nodeVersion` and many more.

## **Features:**

Cypress comes fully baked, batteries included. Here is a list of things it can do that no other testing framework can:

### **Time Travel:**

Cypress takes snapshots as your tests run. Hover over commands in the Command Log to see exactly what happened at each step.

### **Debuggability:**

Stop guessing why your tests are failing. Debug directly from familiar tools like Developer Tools. Our readable errors and stack traces make debugging lightning fast.

### **Automatic Waiting:**

Never add `waits` or `sleeps` to your tests. Cypress automatically waits for commands and assertions before moving on. No more `async hell`.

### **Screenshots and Videos:**

View screenshots taken automatically on failure, or videos of your entire test suite when run from the CLI.

### **Spies, Stubs, and Clocks:**

Verify and control the behavior of functions, server responses, or timers. The same functionality you love from unit testing is right at your fingertips.

### **Network Traffic Control:**

Easily control, stub, and test edge cases without involving your server. You can stub network traffic however you like.

### **Consistent Results:**

Our architecture doesn't use Selenium or WebDriver. Say hello to fast, consistent and reliable tests that are flake-free.

### **Cross browser Testing:**

Run tests within Firefox and Chrome-family browsers (including Edge and Electron) locally and optimally in a Continuous Integration pipeline.

## Writing tests:

Here we'll write test to check two functionality of our application.

1. Load Website
2. Add a todo

### Load Website:

It is a very basic test to check whether our site gets loaded or not. Create a new file named `loadTodo.spec.js` (you can name whatever you want) inside integration folder and open it.

From the code snippet above, the first line simply defines our test suite as "Load Page" — feel free to make it descriptive as you deem fit. After that, we describe what we want to do, which is to show content as a placeholder in this case "TODO loaded".

The `cy.visit()` method was used to specify the URL that we want Cypress to visit. This test is expected to pass unless wrong url is provided.

### Running test:

Run the following command in CLI.

Note: Ensure that your application is running in local server.

```
npx cypress open
```

After running this command, following window will pop-up. To run a particular test you've to just click on it (in our case `loadTodo.spec.js`).



## Add a TODO:

To test this functionality, create another file named `addTodo.spec.js` and open it.

`cy.get()` method helps us to grab/get a html element by its class name (`.new-todo`) which in our case is text area for entering TODO items.

`type()` enters the text "Write a Technical Article" automatically in text area.

`type("{enter}")` triggers the `onEnter` event and todo item is added in list.

`should()` is an Assertion that checks whether list has one item or not.  
Browser runs `addTodo.spec` test

Our program has passed all the test so far. One of the neat features of Cypress is that you can actually hover over the commands and you will see a snapshot of when that command ran. It also shows before and after option for an event. Here you can see when before is selected it show todo item in text area(not submitted) and when after is selected todo item is added on the list.

# Student Coordinators

BATCH 2019-2023

RUSHIL PATEL

URVI  
SUKHARAMWALA

DHRUV MOJILA

KRISHNA  
TAMAKUWALA

DHAIRYA PATEL

VINITI  
CHANGAWALA

MANAN CHORARIA

NAVPREET KOUR

MAHIMNA PANDYA

MANSI SHAH

DIVYESH CHHATRANI

ISHA LAD

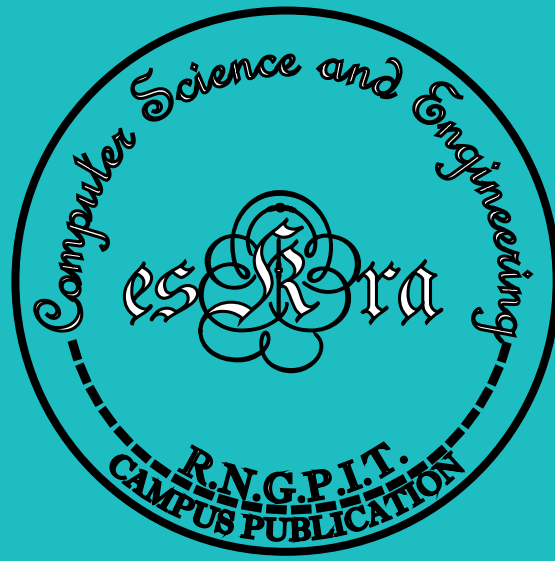
DINKEL MISTRY

VIDHI PATEL

YASHI MARVADI

“Getting a quality website is not an expenses but rather an investment.”

- Dr. Christopher DayaGoag



[www.eskra.rngpit.ac.in](http://www.eskra.rngpit.ac.in)

A Magazine by:  
ESKRA - Campus publication  
Computer Science and Engineering,  
R.N.G. Patel Institute of Technology,  
Surat

You can provide feedback on  
[eskra.cse@rngpit.ac.in](mailto:eskra.cse@rngpit.ac.in)



[eskra.cse@rngpit.ac.in](mailto:eskra.cse@rngpit.ac.in)



[cserngpit](https://www.facebook.com/cserngpit)



[cserngpit](https://www.instagram.com/cserngpit)



+91-9427527398  
+91-9033707334  
+91-9510427118