



# Bootstrap

လို - တို - ရှင်း

အီမောင်

Fairway



## မာတိကာ

- 3      မိတ်ဆက်
- 7      အခန်း (၁) – HTML
- 34     အခန်း (၂) – CSS
- 73     အခန်း (၃) – Bootstrap Intro
- 85     အခန်း (၄) – CSS Components
- 111    အခန်း (၅) – JavaScript Components
- 125    အခန်း (၆) – Bootstrap Layouts
- 139    အခန်း (၇) – Utility Classes
- 151    အခန်း (၈) – Icons
- 162    အခန်း (၉) – Admin Dashboard
- 176    အခန်း (၁၀) – Customizing Bootstrap
- 182    နိဂုံးချုပ်

## မိတ်ဆက်

Bootstrap CSS Framework ဟာ Web Design နဲ့ Web Development လောကကို ကိုင်လှပ်ပြီး တစ်ခေတ်ဆန်းသွားစေခဲ့တဲ့ နည်းပညာတစ်ခုပါ။ Bootstrap မတိုင်ခင်က Web Designer တွေ Web Developer တွေဟာ HTML, CSS, JavaScript တွေကို ကိုယ်တိုင် ချရေးပြီးတော့ ကိုယ့်စိတ်ကူးဥပဒေရှုရင် ရှိသလို ဝတ်ဆိုက်တွေကို ဖန်တီးခဲ့ကြပါတယ်။ Bootstrap ထွက်ပေါ်လာပြီး နောက်မှာတော့ ကိုယ်တိုင် အကုန်လုပ်စရာ မလိုတော့ဘဲ Bootstrap က ပေးထားတဲ့ အသင့်သုံး Components နဲ့ Layouts လုပ်ဆောင်ချက်တွေကို အသုံးချပြီးတော့ ဖန်တီးနိုင်လာကြပါတယ်။ ကိုယ့်စိတ်ကူးဥပဒေနဲ့ တိတွင်ချင်ရင် လည်း လုံးဝကွဲပြား ဆန်းပြားတဲ့ ဖန်တီးမှုတွေထက် Bootstrap ကို အခြေခြားထားတဲ့ ဖန်တီးမှုတွေကို ပိုပြီး တော့ လုပ်လာကြပါတယ်။

Web Developer တစ်ဦးအနေနဲ့ Bootstrap ပေါ်ခါစက လုံးဝမကြိုက်ပါဘူး။ ဘာကြောင့်လဲ မပြောခင် ကြားဖြတ်ပြီး အနည်းငယ် ရောထွေးနေတဲ့ Web Designer နဲ့ Web Developer ဆိုတဲ့ အသုံးအနှစ်ခု အကြောင်းကို ပြောချင်ပါတယ်။ ဒီကိစ္စနဲ့ပက်သက်ရင် အားလုံးက တစ်ညီထဲ သဘောတူလက်ခံထားကြတဲ့ အမိပိုယ်ဖွင့်ဆိုချက်တော့ မရှိပါဘူး။ ဒါပေမယ့် အကြမ်းဖျဉ်းအားဖြင့် ဒီလိုပါ။

Web Designer ဆိုတာဟာ ဝတ်ဆိုက်တစ်ခုရဲ့ User တွေ တွေ့မြင်ထိတွေ့ အသုံးပြုရတဲ့ အသွင်အပြင်စိုင်းကို ရေးဆွဲဖန်တီးရတဲ့ သူတွေပါ။ ဒီလိုဖန်တီးဖို့အတွက် Adobe Photoshop တို့ Adobe Illustrator တို့ Sketch တို့လို Graphic Design ပိုင်းဆိုင်ရာ ဆော့ဖို့တွေကို အသုံးပြုတာလည်း ဖြစ်နိုင်ပါတယ်။ Pencil တို့ Figma တို့လို UI Prototype နည်းပညာမျိုးကို အသုံးပြုတာလည်း ဖြစ်နိုင်ပါတယ်။ ဒါမှမဟုတ် HTML/CSS ကို သုံးပြီး ဖန်တီးတာလည်း ဖြစ်နိုင်ပါသေးတယ်။ HTML/CSS ကို တိုက်ရှိက်သုံးတာမျိုး ဖြစ်နိုင်သလို၊ Photoshop တို့နဲ့ Sketch တို့နဲ့ ဒီဇိုင်းအရင်ဆွဲပြီးတော့မှ အဲဒီဒီဇိုင်းကို HTML/CSS နဲ့ Template ပြန်ပြောင်းတာလည်း ဖြစ်နိုင်ပါတယ်။ ဒီထက်တစ်ဆင့် မြင့်လာတဲ့အခါ JavaScript တို့ jQuery တို့ React

တို့လို နည်းပညာမျိုးတွေနဲ့ ထိတွေ့အသုံးပြုနိုင်တဲ့ အဆင့်ထိ ဖန်တီးကြပါတယ်။ တစ်ယောက်နဲ့တစ်ယောက် သွားတဲ့အဆင့်တွေ မတူကြပါဘူး။ တစ်ချို့က မြင်ရတဲ့ ဒီဇိုင်းပိုင်း ဆဲပြီးရင် Web Designer တစ်ဦးရဲ့လုပ်ငန်း ပြီဆုံးပြီလို သဘောထားကြပါတယ်။ တစ်ချို့ကတော့ မြင်ရယုံနဲ့ မပြီးသေးဘူး၊ ပရောဂျက်ထဲမှာ လက်တွေ့ထည့် အသုံးချလိုရတဲ့ Template တွေ Code တွေထိ တစ်ခါတည်း ရေးပေးရတယ်လို သဘောထားကြပါတယ်။ လိုရင်းအနှစ်ချုပ်ကတော့ သတ်မှတ်ထားတဲ့အဆင့် မတူကြပေမယ့် Web Designer တစ်ဦးရဲ့အလုပ်က ဝဘ်ဆိုက်တွေမှာ တွေ့မြင့်ထိတွေ့နိုင်တဲ့ အပိုင်းကို ဖန်တီးပေးခြင်း ဖြစ်ပါတယ်။ တစ်ချို့ Web Designer တွေက Code ရေးနိုင်ပြီး၊ တစ်ချို့ မရေးနိုင်ကြပါဘူး။ Code ရေးနိုင်ခြင်း မရေးနိုင်ခြင်းက Web Designer ကောင်း ဟုတ်ခြင်း၊ မဟုတ်ခြင်းနဲ့ မဆိုင်ပါဘူး။ တစ်ယောက်နဲ့တစ်ယောက် ချဉ်းကပ်ပုံ မတူကြတာသာ ဖြစ်ပါတယ်။

Web Developer ဆိုတာကတော့ ဝဘ်ဆိုက်တွေကို လက်တွေ့အလုပ်လုပ်ပြီး အများသုံးလိုရအောင် ရွှေ့တင်တဲ့အထိ HTML/CSS Template တွေ၊ Server-side နည်းပညာတွေ၊ Database နည်းပညာတွေနဲ့ ပေါင်းစပ်ပြီး ဖန်တီးရေးသားတဲ့ သူတွေပါ။ ဒီနေရာမှာ Web Designer ကြိုတင်ရေးဆွဲပေးထားတဲ့ ဒီဇိုင်းကို အသုံးပြုပြီး ဆက်လက်ဖန်တီးတာဖြစ်နိုင်သလို၊ ကိုယ်တိုင် ဒီဇိုင်းနဲ့ Template တွေကအစ ဖန်တီးရတာ မျိုးလည်း ဖြစ်နိုင်ပါတယ်။ တစ်ချို့ Team တွေမှာ Web Designer နဲ့ Web Developer ကို သူတာဝန်နဲ့သူ ခွဲထားပြီး တစ်ချို့ Team တွေမှာတော့ ခွဲမထားပါဘူး၊ Web Developer လိုပြောရင် အကုန်တာဝန်ယူကြရတာတွေလည်း ရှိပါတယ်။ ဒီလိုသီးခြား Web Designer မရှိတဲ့ Team တွေကလုပ်တဲ့ ပရောဂျက်တွေဟာ ဒီဇိုင်းအသွင်အပြင် အားနည်းကြလေ့ရှိပါတယ်။ ဘာသာရပ်နှစ်ခုလုံးဟာ သူဟာနဲ့သူ ကျယ်ပြန်လို ခေါင်းစဉ်တစ်မျိုးတည်းအောက်မှာ နှစ်ခုလုံးကို ကျွမ်းကျင်ပိုင်နိုင်ဖို့ဆိုတာ ခက်ပါတယ်။ နှစ်ခုလုံးကျွမ်းကျင်သူ မရှိဘူးမဟုတ်ပါဘူး၊ ရှိတော့ရှိတယ်၊ ရှားတယ်ဆိုတာမျိုးပါ။ တစ်ချို့ Team တွေကျတော့ Web Designer လို ခေါင်းစဉ်တပ်ပြီး ဒီဇိုင်းဆွဲတာရော လက်တွေ့လွှေ့တင်နိုင်တဲ့ အဆင့်ထိရော အကုန်တာဝန်ယူကြပြန်ပါတယ်။ ဒီလိုအခါမျိုးမှာ တစ်ချို့ကလည်း အသင့်သုံး Content Management System (CMS) နည်းပညာတွေကို အခြေခံပြီး ဖန်တီးကြလို CMS Developer ဆိုတဲ့ ခေါင်းစဉ်လည်း ကြားထဲမှာ ရှိလာပြန်ပါတယ်။ ဒါကြောင့် ဒီအခေါ်အဝေါနှစ်ခု ရောထွေးနေယုံမက တစ်ခြား Web Master, Front-end Developer, Back-end Developer စဖြင့် အခေါ်အဝေါတွေနဲ့လည်း ရောထွေးနေပါသေးတယ်။ ဒါတွေကြောင့် ခေါင်းရှုပ်သွားရင် စိတ်မပူပါနဲ့၊ စာဖတ်သူမှုမဟုတ်ပါဘူး၊ ဘယ်သူမှ ရှင်းလင်းတဲ့ အဓိပါယ်ဖွင့်ဆိုချက်ကို မသိကြတာပါ။ တစ်ယောက်ကိုမေးရင် တစ်မျိုးပြောကြပါလိမ့်မယ်။

အကျဉ်းချုပ်ကို ဒီလိုမှတ်နိုင်ပါတယ်။ Web Designer ဆိတာ User တွေမြင်ထိတွေရတဲ့ အသွင်အပြင် ဒီဇိုင်း ရေးဆွဲဖန်တီးသူဖြစ်ပြီး၊ Web Developer ဆိတာကတော့ အဲဒါ ဒီဇိုင်းကို အသုံးပြုပြီး လက်တွေ လွှဲတင်လိုရတဲ့ ဝဘ်ဆိုက်တွေ ဖန်တီးသူဖြစ်တယ် လို မှတ်နိုင်ပါတယ်။

စာရေးသူကိုယ်တိုင်ကတော့ Web Developer တစ်ဦးဖြစ်ပါတယ်။ Web Design ပိုင်းကို မကျမ်းပါဘူး။ ဒီ စာအုပ်မှာလည်း Bootstrap အကြောင်းကို Web Development ရှုထောင့်ကနေ ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။ အထက်မှာ Web Developer တစ်ဦးအနေနဲ့ Bootstrap ပေါ်ခါစက မကြိုက်ဘူးလို ပြောခဲ့ပါတယ်။ ဘာဖြစ်လိုလဲဆိုတော့ ဒီလိုပါ။

တစ်ခြား Software အမျိုးအစားတွေနဲ့ယုံ့ရင် Web ရဲ့ အားသာချက်ကတော့ HTML, CSS, JavaScript ရဲ့ အကူးအညီနဲ့ အကန့်အသတ်ဘောင် တော်တော်နည်းပြီး စိတ်ကူးညာ၏ရှုရင်ရှိသလောက် ထူးခြားဆန်းပြား တဲ့ User Interface တွေဖန်တီးနိုင်ခြင်းပဲ ဖြစ်ပါတယ်။ ၂၀၀၅-၂၀၁၀ ကြားကာလဟာ Web ရဲ့ ရွှေ့ခေတ်ပါပဲ။ Web Designer တွေ Web Developer တွေ အပြိုင်အဆိုင် လက်စွမ်းတွေပြပြီး တိတွင်လိုက်ကြတာမှ အပြိုင်အဆိုင်ပါပဲ။ ဒါပေမယ့် အဲဒီအားသာချက်ကပဲ Web ရဲ့ အားနည်းချက်လည်း ဖြစ်နေပြန်ပါတယ်။ စိတ်ကူးရှိသလို ဖန်တီးလိုရတာ မှန်ပေမယ့်၊ အကုန်ကိုယ့်ဘာသာ ဖန်တီးနေရပါတယ်။ ထပ်ခါထပ်ခါ ပြန်ပြီးတော့ ဖန်တီးရပါတယ်။ တစ်ယောက်ကို တစ်မျိုးစီ ထွင်ကြတော့၊ သုံးတဲ့ User က၊ ဟိုနေရာမှာတစ်မျိုး၊ ဒီနေရာမှာတစ်မျိုး၊ မျက်စီတွေ လည်ကြပါတယ်။

Bootstrap ပေါ်လာတဲ့အခါ သူမှာ User Interface တွေတည်ဆောက်ဖို့ အသင့်သုံးနိုင်တဲ့ Layouts တွေ Components တွေ ပါဝင်လာပါတယ်။ တော်တော်လေး အဆင်ပြေတဲ့အတွက် အချိန်တိုအတွင်း လူကြိုက်များပြီး လူသုံးများသွားပါတယ်။ ဒီတော့ ဟိုဝဘ်ဆိုက် ကြည့်လိုက်လဲ ဒီပုံစံ၊ ဒီဝတ်ဆိုက် ကြည့်လိုက်လဲ ဒီပုံစံ၊ ပုံစံတူတွေ များလာတော့တာပါပဲ။ အဲဒါကို မကြိုက်ခဲ့တာပါ။ မူလ Web နည်းပညာရဲ့ လွပ်လွပ်လပ်လပ် ဖန်တီးတဲ့အလေ့အကျင့်တွေ တစ်ဖြည်းဖြည်း နည်းပါးပျောက်ကွယ်သွားပြီလို မြင်ခဲ့တာပါ။

အချိန်ကာလတစ်ခု ရောက်လာတော့မှ အဲဒါကသာလျှင် ပိုကောင်းတဲ့နည်းဆိုတာကို သိလာခဲ့ရတာပါ။ အသုံးပြုသူ User က ရှုချင်စွယ်၊ သုံးချင်စွယ်ဖြစ်တာကို လိုချင်ပေမယ့် အလွန်အမင်း ဆန်းပြားတာကို တော့မလိုချင်ပါဘူး။ Consistence ဖြစ်တာကို လိုချင်တာပါ။ သုံးရလွယ်ကူတာကို လိုချင်တာပါ။ Bootstrap ထွက်ပေါ်လာပြီးနောက်မှာတော့ Menu ရဲ့ဖွဲ့စည်းပုံ၊ ပါဝင်တဲ့ Component တွေရဲ့ ဖွဲ့စည်းပုံ၊

Layout ရဲဖွံ့စည်းပုံ၊ ဒါတွေဟာ ခပ်ဆင်ဆင်တွေ ဖြစ်လာတော့ User အတွက်က တော်တော် အဆင်ပြေပါတယ်။ ဘယ်သွားသွား အရောင်အသွေးနဲ့ ဖွံ့စည်းပုံသာ ကွဲပြားသွားမယ်၊ အသုံးပြုနည်းက အသစ်အဆန်းမဟုတ်တော့ဘဲ သိရှိကျမ်းဝင်ပြီး ဖြစ်တဲ့ ပုံစံကိုသာ ရရှိမှာဖြစ်ပါတယ်။

ဒီသဘောသဘာဝကြောင့်ပဲ Bootstrap ဟာ လက်ရှိမှာလူသုံးအများဆုံး နည်းပညာတစ်ခုဖြစ်နေတာပါ။ Bootstrap နဲ့ အပြိုင် Foundation လို အလားတူ နည်းပညာတွေ ရှိသေးပေမယ့် Bootstrap ကသာလျှင် အဓိကနည်းပညာ ဖြစ်လာခဲ့ပါတယ်။ အခုနောက်ပိုင်းမှာ Bootstrap နဲ့ ရည်ရွယ်ချက်တူပေမယ့် သဘောသဘာချင်း မတူတော့တဲ့ Tailwind လိုပေါ်တဲ့ နည်းပညာတစ်ခု ခေတ်စားစ ပြုနေပါတယ်။ Bootstrap ကို ကျော်ဖြတ်ပြီး အဓိကနည်းပညာနေရာကို ယူသွားမလားဆိုတာတော့ ပြောဖို့စောပါသေးတယ်။ စောင့်ကြည့်ကြရှိုးမှာပါ။

ဒီစာအုပ်ဟာ ရေးလက်စဖြစ်တဲ့ လိုတိုရှင်း စာအုပ်တွေထဲမှာ (င) အုပ်မြောက်ဖြစ်ပါတယ်။ ရှုံးပိုင်းကရေးခဲ့တဲ့ React လိုတိုရှင်း၊ Laravel လိုတိုရှင်း၊ API လိုတိုရှင်း၊ စတဲ့ စာအုပ်တွေက Intermediate အဆင့် စာအုပ်တွေလို ဆိုနိုင်ပါတယ်။ အခြေခံ အထိက်အလျောက်ရှိပြီးမှသာ အဲဒီစာအုပ်တွေကို ဖတ်ရှုလေ့လာရတာ အဆင်ပြေမှာပါ။ ဒီစာအုပ်ကိုတော့ လူတိုင်းဖတ်လိုရတဲ့ အခြေခံအဆင့် Basic စာအုပ်တစ်အုပ် ဖြစ်စေချင်ပါတယ်။ စာဖတ်သူက အခြေခံတွေ ကြိုတင်လေ့လာထားခြင်း မရှိသေးဘူးလို သဘောထားပြီးတော့ ရေးသားဖော်ပြသွားမှာပါ။ ဒါကြောင့် HTML လို တော်တော်လေး အခြေခံကျတဲ့ နည်းပညာမျိုးကအစထည့်သွင်းဖော်ပြသွားမှာပဲ ဖြစ်ပါတယ်။

## အခန်း (၁) - HTML

### Markup Language

HTML ဟာ Markup Language တစ်ခုဖြစ်ပါတယ်။ သူနာမည် အပြည့်အစုံက Hypertext Markup Language ပါ။ Markup Language ဆိုတာကို လိုတိရှင်း ဒီလိုမှတ်ပါ။ ကွန်ပျူးတာက နားလည်အလုပ်လုပ်နိုင်တဲ့ Content Structure တည်ဆောက်ရသော နည်းပညာဖြစ်ပါတယ်။ အခြားသော Markup Language တွေ ရှိပါသေးတယ်။ XML, YAML, Markdown စသည်ဖြင့်ပါ။ Language မတူလို့ ရေးသားပုံ တွေ မတူပေါမယ့် Markup Language အားလုံးရဲ့ ရည်ရွယ်ချက်က အတူတူပါပဲ။ ကွန်ပျူးတာက နားလည် အလုပ်လုပ်နိုင်တဲ့ Content Structure တည်ဆောက်ဖို့ပဲ ဖြစ်ပါတယ်။ ဥပမာ - ဒီစာလေးကို လေ့လာ ကြည့်ပါ။

#### Plain Text

HTML ဖြစ်ပေါ်လာပုံ

HTML ကို Tim Berners-Lee အမည်ရှု ကွန်ပျူးတာသိပ္ပံာပညာရှင်က ၁၉၉၁ ခုနှစ်တွင် WorldWideWeb နည်းပညာ၏ အစိတ်အပိုင်းတစ်ရပ်အဖြစ် ချုပြုခြင်းဖြစ်သည်။ ငြင်း WorldWideWeb နည်းပညာကို လက်တွေ့ စမ်းသပ်နိုင်ရန် အောက်ပါတို့ကိုလည်း ပူးတွဲတို့ထွင်ခဲ့သည်။

#### Web Browser

HTTP Server

ဒီစာကိုလူတစ်ယောက် ဖတ်ကြည့်ရင် ဘယ်ဟာက ခေါင်းစဉ်၊ ဘယ်ဟာက စာကိုယ်၊ ဘယ်ဟာက စာရင်း စသဖြင့် အလိုလို သိပါတယ်။ ဘာကိုကြည့်ပြီး သိတာလဲဆိုတော့ ရေးထားတဲ့ စာမျာပါတဲ့ အကြောင်းအရာ

ကို ဖတ်ကြည့်ပြီး သိတာပါ။ ကွန်ပျူးတာကတော့ လူတစ်ယောက်လို အဲဒီစာကို ဖတ်ရှုသိရှိနိုင်စွမ်း ရှိမှာ မဟုတ်ပါဘူး။ အခုနောက်ပိုင်း AI နည်းပညာတွေ ပေါ်လာလို သိရှိနိုင်စွမ်း ရှိလာပေမယ့် ဒါတွေကအခုမှ အစပဲရှုပါသေးတယ်။ ပါတဲ့အကြောင်းအရာကို ကြည့်ပြီးတော့ ခေါင်းစဉ်လား၊ စာကိုယ်လား၊ စာရင်းလား ခဲ့နိုင်မှာ မဟုတ်ပါဘူး။ အဲဒါကို ခွဲနိုင်အောင် အမှတ်အသား လုပ်ပေးတဲ့ နည်းပညာကို Markup Language လိုခေါ်တာပါ။ ဒီစာကိုပဲ HTML နဲ့ ရေးမယ်ဆိုရင် အခုလို ရေးပေးရမှာပါ။

## HTML

```
<h1>HTML ဖြစ်ပေါ်လာပံ</h1>
```

```
<p>
```

HTML ကို Tim Berners-Lee အမည်ရကွန်ပျူးတာသိပ္ပါယ်ပညာရှင်က ၁၉၉၁ ခုနှစ်တွင် WorldWideWeb နည်းပညာ၏ အစိတ်အပိုင်းတစ်ရပ်အဖြစ် ချုပြုခြင်းဖြစ်သည်။ ငြင်းWorldWideWeb နည်းပညာကို လက်တွေ့စမ်းသပ်နိုင်ရန် အောက်ပါတို့ကို ပူးတွဲတိတွင်ခဲ့သည်။

```
</p>
```

```
<ul>
```

- <li>Web Browser</li>
- <li>HTTP Server</li>

```
</ul>
```

HTML က သတ်မှတ်ထားပါတယ်။ <h1> ဆိုတဲ့အမှတ်အသားနဲ့ </h1> ဆိုတဲ့အမှတ်အသား ကြားထဲမှာ ရှိတဲ့ အကြောင်းအရာကို ခေါင်းစီးအဆင့် (၁) လို မှတ်ယူရမယ် တဲ့။ ဒါကိုရေးသားသူ ကျွန်တော်တို့က နားလည်သိရှိလို ရေးပေးလိုက်သလို၊ HTML ကို နားလည်တဲ့ ကွန်ပျူးတာစနစ်တွေလည်း သိနိုင်သွားပါပြီ။ ဘာရေးထားလည်း ဖတ်တတ်စရာမလိုဘူး၊ <h1> အမှတ်အသားနဲ့ </h1> အမှတ်အသားကြားထဲက ဟာကို ခေါင်းစီးအဆင့် (၁) မှန်း သိနိုင်သွားပါပြီ။ HTML ကို နားလည်တဲ့ ကွန်ပျူးတာစနစ်လို ပြောလိုက်တာကိုသတိပြုပါ။ HTML ကိုနားမလည်တဲ့ ကွန်ပျူးတာ စနစ်တွေလည်း ရှိနိုင်တာပါပဲ။ HTML ကို နားလည်တဲ့ ကွန်ပျူးတာစနစ်တွေထဲမှာ အဓိကအကျခုံးကတော့ ကျွန်တော်တို့တွေ နေ့စဉ် အင်တာနက်သုံးဖို့ အသုံးပြုနေကြတဲ့ Google Chrome, Mozilla Firefox, Microsoft Edge စတဲ့ Web Browser တွေပါပဲ။ ဒီ Web Browser တွေက HTML ကိုနားလည်ကြပါတယ်။ ဒါကြောင့် HTML အမှတ်အသားတွေကိုသုံးပြီး ရေးထားတဲ့ Document ကို အမှတ်အသား သတ်မှတ်ချက်နဲ့အညီ ဖော်ပြုအလုပ်လုပ် ပေးနိုင်ကြပါတယ်။

## Basic Structure & Elements

အထက်မှာပေးခဲ့တဲ့ နမူနာကို ပြည့်စုံအောင် ရေးမယ်ဆိုရင် ဒီလိုရေးပေးရမှာပါ။

### HTML

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>HTML ဆိုသည်မှာ</title>
</head>
<body>
    <h1>HTML ဖြစ်ပေါ်လာပုံ</h1>

    <p>
        HTML ကို Tim Berners-Lee အမည်ရကွန်ပျု၍ တာသိပ္ပါယ်ပညာရှင်က ၁၉၉၀ ခုနှစ်
        တွင် WorldWideWeb နည်းပညာ၏ အစိတ်အပိုင်းတစ်ရပ်အဖြစ် ချုပြုခဲ့ခြင်း
        ဖြစ်သည်။ ငွေး WorldWideWeb နည်းပညာကို လက်တွေ့စမ်းသပ် နိုင်ရန်
        အောက်ပါတို့ကို ပူးတွဲတို့ထွင်ခဲ့သည်။
    </p>

    <ul>
        <li>Web Browser</li>
        <li>HTTP Server</li>
    </ul>

    <p>
        HTML နှင့် ဆက်စပ်နည်းပညာများမှာ အောက်ပါအတိုင်း ဖြစ်ပါသည်။
    </p>

    <ol>
        <li>HTTP</li>
        <li>CSS</li>
    </ol>
</body>
</html>
```

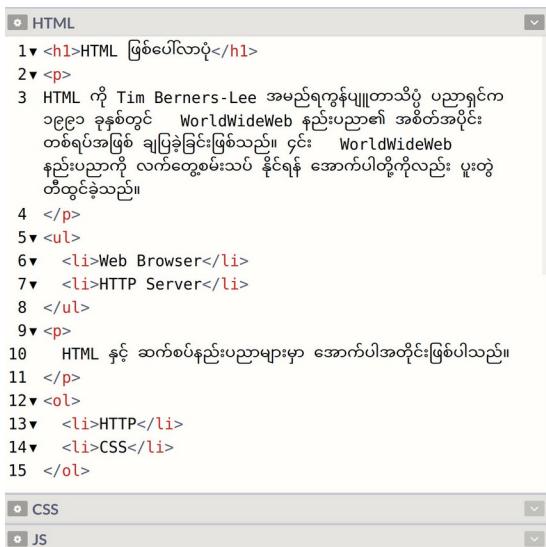
ဒီကုဒ်ကို ကိုယ်တိုင်လည်း ကူးရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။ နှစ်သက်ရာ Code Editor ကိုသုံးပြီး ရေးလိုရ ပါတယ်။ လက်ရှိမှာ VS Code ကတော့ လူကြိုက်အများဆုံး Code Editor ဖြစ်နေလို့ မရှိသေးရင် ဒီမှာ Download ရယူနိုင်ပါတယ်။

- <https://code.visualstudio.com>

နမူနာကုဒ်တွေကို ကူးရေးပြီးရင် ရလဒ်ကို Chrome, Firefox စသဖြင့် နှစ်သက်ရာ Web Browser နဲ့ဖွင့်ပြီး စမ်းကြည့်လို့ရပါတယ်။

ဒီစာအုပ်မှာ ရေးပြသမှု ကုဒ်အတိအစလေးကအစ အကုန်လုံးကို လိုက်စမ်းကြည့်ဖို့ တိုက်တွန်းပါတယ်။ ဒီတော့မှ တစ်ခါတည်းမြင်ပြီး တစ်ခါတည်းရသွားမှာပါ။ ဒီလိုစမ်းကြည့်တဲ့အခါ အမြန်ဆုံးနည်းကတော့ **Code Pen** လို့ နည်းပညာကို အသုံးပြုခြင်းပါဝါ။ ချက်ခြင်းရေးပြီး ချက်ခြင်းရလဒ်မြင်ရလို့ အဆင်ပြေပါတယ်။ Code Editor နဲ့ရေးလိုက်၊ သိမ်းလိုက်၊ Browser နဲ့ ပြန်ဖွင့်လိုက် လုပ်နေစရာ မလိုတော့ပါဘူး။ အခွန် လိုင်းကနေ တိုက်ရှိက်ရေးစမ်းတာမြို့လို့ အင်တာန်က်အဆက်အသွယ်ရှိဖို့တော့ လိုပါတယ်။ ဒီမှာရေးရမှာပါ။

- <https://codepen.io/pen>



```

HTML
1▼ <h1>HTML ဖြစ်ပေါ်လာပုံ</h1>
2▼ <p>
3 HTML ကို Tim Berners-Lee အမည်ရကွန်ပျူောဘာသီပုံ ပညာရှင်က
  ၁၉၉၀ ခုနှစ်တွင် WorldWideWeb နည်းပညာ၏ အစိတ်အပိုင်း
  ထစ်ရပ်အဖြစ် ချုပြုခြင်းဖြစ်သည်။ ငြင်း WorldWideWeb
  နည်းပညာကို လက်တွေ့စွာသပ် နိုင်ရန် အောက်ပါတို့ကိုလည်း ပူးတွဲ
  တိုထွင်ခဲ့သည်။
4 </p>
5▼ <ul>
6▼   <li>Web Browser</li>
7▼   <li>HTTP Server</li>
8 </ul>
9▼ <p>
10   HTML နှင့် ဆက်စပ်နည်းပညာများမှာ အောက်ပါအတိုင်းဖြစ်ပါသည်။
11 </p>
12▼ <ol>
13▼   <li>HTTP</li>
14▼   <li>CSS</li>
15 </ol>

```

**CSS**

**JS**

## HTML ဖြစ်ပေါ်လာပုံ

HTML ကို Tim Berners-Lee အမည်ရကွန်ပျူောဘာသီပုံ ပညာရှင်က ၁၉၉၀ ခုနှစ်တွင် WorldWideWeb နည်းပညာ၏ အစိတ်အပိုင်း ထစ်ရပ်အဖြစ် ချုပြုခြင်းဖြစ်သည်။ ငြင်း WorldWideWeb နည်းပညာကို လက်တွေ့စွာသပ် နိုင်ရန် အောက်ပါတို့ကိုလည်း ပူးတွဲ တိုထွင်ခဲ့သည်။

- Web Browser
- HTTP Server

HTML နှင့် ဆက်စပ်နည်းပညာများမှာ အောက်ပါအတိုင်းဖြစ်ပါသည်။

1. HTTP
2. CSS

ပုံမှာဖော်ပြထားတာကတော့ Code Pen ကိုသုံးပြီးရေးထားတဲ့ကုဒ်နဲ့ ယဉ်တွဲဖော်ပြထားတဲ့ ရလဒ်ဖြစ်ပါတယ်။ ရလဒ်အနေနဲ့ ရေးထားတဲ့ HTML ပေါ်မူတည်ပြီး သင့်တော်တဲ့အသွင်အပြင်နဲ့ဖော်ပြနေတဲ့ Content ကို တွေ့မြင်ရခြင်းဖြစ်ပါတယ်။ ကုဒ်တွေရေးတဲ့အခါ နမူနာမှာ ရေးပြသလို Indent လေးတွေ မှန်အောင် ရေးသင့်ပါတယ်။ Indent ဆိုတာ တစ်ခုခုရဲ့ အတွင်းထဲမှာရှိတဲ့ အကြောင်းအရာကို အတွင်းထဲမှာ ရှိမှန်းသိသာ မြင်သာအောင် Tab လေး တွေနဲ့ပြီး ရေးထားတာကို ပြောတာပါ။ ဒါလိုပါ -

### HTML

```
<body>
  <ul>
    <li>Web Browser</li>
    <li>HTTP Server</li>
  </ul>
</body>
```

ဒီလိုရေးထားတဲ့အတွက် <ul> က <body> အတွင်းမှာရှိပြီး <li> Element တွေဟာ <ul> ရဲ့ အတွင်းထဲမှာရှိတဲ့ Element တွေဖြစ်ကြောင်း ထင်ရှားမြင်သာသွားစေပါတယ်။ ဖတ်ရတာ ပိုအဆင်ပြေသွားသလို အဖွင့်အပိတ်တွေ မစုံလို မှားတဲ့အခါမှာမျိုးမှာ အမှားကို ပိုပြီးတော့ မြင်သာစေမှာ ဖြစ်ပါတယ်။ Indent တွေ မမှန်လည်း အလုပ်လုပ်ပေမယ့် Indent မှန်မှသာ ကိုယ့်ကုဒ်ကို ကိုယ်ဘာသာ ပြန်ဖတ်လို ရမှာပါ။ မဟုတ်ရင် ဖတ်ရခက်ပြီး၊ အမှားရှာရ၊ ပြင်ရခက်နေပါလိမ့်မယ်။

ဟိုးအပေါ်က နမူနာမှာ ရေးသားပါဝင်တဲ့ အမှတ်အသား တစ်ခုချင်းစီအကြောင်းကို ဆက်ပြီးတော့ ရှင်းပြပါမယ်။ ပထမဆုံးအနေနဲ့ ဒီအမှတ်အသားတစ်ခုချင်းစီကို HTML Tag လိုပေါ်ပြီး အတွဲအဖက်ပြည့်စုံတဲ့ HTML Tag အစုံလိုက်ကို Element လို ခေါ်တယ်လို မှတ်ထားပါ။ နောက်ပိုင်းမှာ အမှတ်အသားလိုပြောမယ့်အစား Element လိုပဲ ဆက်သုံးသွားမှာ ဖြစ်ပါတယ်။ စကြည့်ကြပါမယ်။

<!DOCTYPE html> - ဒီ Element ကို Document Type Declaration လို ခေါ်ပါတယ်။ အရင်က HTML နဲ့ ဆက်စပ် Document အမျိုးအစားတွေ အမျိုးမျိုးရှိလိမ့်မယ်လို ရည်ရွယ်ခဲ့ကြတာပါ။ HTML 4.0, HTML 4.01, XHTML 1.0, XHTML 1.1, HTML 5 စသဖြင့် Version အမျိုးမျိုးရှိသလို Strict, Transitional, Frameset စသဖြင့်မူကွဲတွေလည်း အများကြီးပါ။ အဲဒါတွေကို အခုခေါင်းရှုပ်ခံပြီး ပြောစရာ၊ မှတ်စရာမလိုတော့ပါဘူး။ ကနေခေတ်မှာ တစ်မျိုးတည်းပဲ သုံးကြပါတော့တယ်။ အဲဒါက HTML 5 ပါ။ HTML 5 Document တစ်ခုဖြစ်ကြောင်း အမှတ်အသားအနေနဲ့ ဒီ Element က ထိပ်ဆုံးမှာမဖြစ်မနေပါသင့်ပါတယ်။

ပါမှုမှန်တာလား၊ ပါမှုအလုပ်လုပ်တာလားဆိုရင်၊ မဟုတ်ပါဘူး၊ မပါလည်း အလုပ်တော့ လုပ်ပါတယ်။ ဒီနေရာမှာ ပြောစရာရှိတာက၊ HTML က သတ်မှတ်ထားတဲ့ရေးနည်းအတိုင်း အတိအကျမရေးဘဲ မှားပြီး ရေးမိရင် ဘာဖြစ်မလဲ ဆိုတာကို ကြားဖြတ်ပြောစရာ ရှိပါတယ်။ ဥပမာ <h1> ... </h1> လို့ ရေးရမှာကို <h1> ... </h2> လို့ရေးမိတယ်ဆိုရင် ဘယ်လိုလုပ်မလဲ။ အဖွင့်နဲ့အပိတ် မှားနေပါပြီ။ Programming Language တွေမှာဆိုရင် ရေးထုံးမှားရင် Error တက်ပါတယ်၊ အလုပ်မလုပ်ပါဘူး။ HTML မှာတော့ ရေးထုံးမှားလည်း Error မတက်ပါဘူး၊ ဆက်အလုပ်လုပ်ပါတယ်။

ဒီနေရာမှာ အလုပ်လုပ်ပုံလုပ်နည်း (J) မျိုးရှိတယ်လို့ မှတ်နိုင်ပါတယ်။ ရုံးရှိုး Normal Mode နဲ့ Quirks Mode ပါ။ Document Type ကြေညာတဲ့ အမှတ်အသားပါရင် Normal Mode နဲ့ အလုပ်လုပ်ပြီး မပါရင် Quirks Mode နဲ့ အလုပ်လုပ်တယ်လို့ မှတ်နိုင်ပါတယ်။ ဒီနှစ်ခုဘာကွာလဲဆိုရင် အခုလို လိုရင်းအတိုချုပ် မှတ်ပါ။ Normal Mode မှာ HTML စံသတ်မှတ်ချက်အတိုင်း တိတိကျကျ မှားရင် မှားတဲ့အတိုင်း ဆက်ပြေးပါတယ်။ Quirks Mode မှာ ရေးထားတာ မှားနေရင်လည်း Browser က သူကောင်းမယ်ထင်သလို ပြင်ပြီး ပြောပြီးပြတာ မကောင်းဘူးလားလို့ မေးရင်၊ မကောင်းပါဘူး။ မှားနေတာကို မှားမှန်းမသိရဘဲ မှန်တယ်ထင်မိတဲ့အခါ ကြာလေ ပြဿနာကြီးလေ ဖြစ်သွားပါလိမ့်မယ်။ မှားနေတာကို မှားနေတဲ့အတိုင်း သိရတာက ပိုကောင်းလို့ Document Type ကြေညာတဲ့ အမှတ်အသားကို မဖြစ်မနေ ထည့်သွင့်တယ်လို့ ပြောတာပါ။ မပါရင်လည်း အလုပ်တော့ လုပ်တယ်ဆိုတာကိုလည်း သတိပြုရမှာပါ။

<html><head><body> - ဒီသုံးခုကိုတော့ အတဲ့လိုက် ပြောဖို့လိုပါတယ်။ Document တစ်ခုကို နှစ်ပိုင်းခဲ့ပြီး ကြည့်သင့်ပါတယ်။ တစ်ကယ့်အချက်အလက်တွေ ပါဝင်တဲ့ Body နဲ့ ရှင်းလင်းချက်တွေ ပါဝင်တဲ့ Header ပါ။ <body> Element ကို တစ်ကယ့် အချက်အလက်တွေ စုစုပေါင်းဖို့သုံးပြီး <head> Element ကိုတော့ ရှင်းလင်းချက်နဲ့အညွှန်းတွေ စုစုပေါင်းထည့်သွင်းဖို့ သုံးပါတယ်။ ရှင်းလင်းချက်အညွှန်းဆိုတာ ဥပမာ - ဘယ်ဖွန့်ကိုသုံးထားတယ်၊ ခေါင်းစဉ်ကဘာဖြစ်တယ်၊ ရေးသားသူက ဘယ်သူဘယ်ဝါဖြစ်တယ်၊ စသဖြင့် အချက်အလက်တွေပါ။ ဒါတွေက တစ်ကယ့် Content မဟုတ်ပါဘူး၊ Content အကြောင်း ရှင်းပြထားတဲ့ ရှင်းလင်းချက်တွေပါ။ ဒီလို ရှင်းလင်းချက်တွေကို Content နဲ့ရောမထားသင့်လို့ အခုလို <head> နဲ့ <body> နှစ်ပိုင်းခဲ့ထားတာပါ။ ဒီလိုနှစ်ပိုင်းခဲ့ထားတဲ့ Element နှစ်ခုကို တဲ့ဖက်စုစုပေါင်းပေးလိုက်တဲ့ သဘောနဲ့ <html> ဆိုတဲ့ Element ထဲမှာ ရေးပေးလိုက်တာ ဖြစ်ပါတယ်။ ဒါတွေ တစ်ခုမှ မပါလည်း ရပါတယ်။ ဒီ Element တွေမပါလို့ မမှားပါဘူး။ ဒါပေမယ့် အလေ့အကျင့်ကောင်းအနေနဲ့ ထည့်ရေးသင့်ပါတယ်။ မထည့်ထားတဲ့ Document တွေ တွေရင်လည်း မမှားဘူးဆိုတာကို သတိပြုဖို့ပါပဲ။

**<meta>** - ဒီ Element ကို ပေးထားတဲ့ နမူနာမှာ Character Set သတ်မှတ်ဖို့ သုံးပြထားပါတယ်။ ဒီ အကြောင်းကလည်း ကျယ်ပြန့်ပါတယ်။ အကျဉ်းချုပ်အနေနဲ့ UTF-8 လိုခေါ်တဲ့ Encoding နည်းပညာကို Unicode Character တွေ သိမ်းဆည်း/ဖော်ပြစ် သုံးရပါတယ်။ Unicode Character ဆိုတဲ့ ထဲမှာ အက်လိပ်စာ၊ မြန်မာစာ၊ ရောဂါန်ရေးတဲ့ စာ၊ ယူနိကုဒ်နဲ့ရေးတဲ့ စာ အကုန်ပါပါတယ်။ တစ်ခြား Character Set တွေ ရှိ ကြပါသေးတယ်။ ASCII လိုခေါ်တဲ့ Character Set ကတော့ အခြေခံအကျခုံးဖြစ်ပြီးတော့ အက်လိပ်စာတွေ သိမ်းဆည်း/ဖော်ပြနိုင်ပါတယ်။ မူအားဖြင့် အက်လိပ်စာမဟုတ်ရင် မပြနိုင်ဘူးလို ပြောလိုရပါတယ်။ ဒါပေါ်မယ့် Win Myanmar ဖွန့်လို ဖွန့်တွေက ASCII ကိုသုံးထားပါတယ်။ ဒါတွေစုံအောင် လျှောက်ပြောရင်တော့ တော်တော် ပေရှည်သွားပါလိမ့်မယ်။ Latin1 ဆိုတဲ့ Character Set လည်း အသုံးများပါသေးတယ်။ အက်လိပ်စာအပြင် အတွန်အတက်၊ အစက်အဆံတွေပါတဲ့ လက်တင်စာတွေ အတွက်ပါ အဆင်ပြေပါတယ်။ ကနေ့ခေတ်မှာတော့ UTF-8 ကိုသာ စွယ်စုံသုံးအဖြစ် သုံးကြပါတော့တယ်။ Browser အဟောင်းတစ်ချို့မှာ Default က Latin1 ဖြစ်နေနိုင်ပါတယ်။ ဒါကြောင့် ဒီသတ်မှတ်ချက်မပါဘဲ မြန်မာစာလို စာမျိုးတွေရေးသား ထည့်သွင်းရင် အဆင်ပြေမှာ မဟုတ်ပါဘူး။ နောက်ပိုင်း Browser တွေကတော့ UTF-8 ကို Default ထားလို ကိုယ်မသတ်မှတ်ပေးလဲ အဆင်ပြေကြပါတယ်။ ဒါပေါ်မယ့် သေချာအောင် ထည့်ပေးသင့်ပါတယ်။

**<meta>** ကို တစ်ခြား ရှင်းလင်းချက်တွေ ထည့်သွင်းဖို့လည်း သုံးကြပါသေးတယ်။ Author တို့ Description တို့ Keywords တို့လို အကြောင်းအရာတွေ ထည့်လို ရတာပါ။ အဲဒါတွေကို အခုအသေးစိတ် မကြည့်ပါနဲ့မြို့။ နောက်လိုအပ်တော့မှ ဆက်ကြည့်သွားလိုရပါတယ်။

**<title>** - ဒီ Element ကတော့ Document ကို ခေါင်းစဉ်တပ်ပေးဖို့ သုံးပါတယ်။ Browser မှာ Document ကို ဖွင့်ကြည့်လိုက်တဲ့ အခါ အခါ Title Bar တို့မှာ ဒီနာမည်ကို ခေါင်းစဉ်အနေနဲ့ လာပြပေးမှာပါ။ မပါမဖြစ်ပါဘူးတဲ့ အချက်ဖြစ်ပါတယ်။

**<h1><h2><h3><h4><h5><h6>** - Content ထဲမှာ ခေါင်းစီးတွေ ထည့်သွင်းဖို့အတွက် Element (၆) မျိုးရှုပါတယ်။ <h1> ကနေ <h6> အထိပါ။ ခေါင်းစီးအစီအစဉ်အလိုက် သင့်တော်ရာကို သုံးပေါ်နိုင်ပါတယ်။ <h1> ကအမြင့်ဆုံး၊ အကြီးဆုံးနဲ့ အဓိကအကျခုံး ခေါင်းစီးပါ။ ကျန်တဲ့ ခေါင်းစီးတွေကို ကိုယ့် Content ပေါ်မှာ မူတည်ပြီး သူနေရာနဲ့သူ လိုအပ်တဲ့ အဆင့်ကို ရွှေ့ချယ်အသုံးပြုနိုင်ပါတယ်။

**<p>** - စာပိုဒ်တွေထည့်သွင်းဖို့အတွက် Paragraph ရဲ့ အတိုကောက်ဖြစ်တဲ့ <p> Element ကို သုံးရပါတယ်။ စာပိုဒ်တိုင်းကို <p> Element အမှတ်အသားနဲ့ ရေးပေးဖို့ပါပဲ။

**<ul> <ol> <li>** - List တွေဖော်ပြဖို့အတွက် Element နှစ်မျိုးရှိပါတယ်။ နမူနာမှာ <ul> ကိုသုံးထားပြီး <ol> လည်း ရှိပါသေးတယ်။ <ul> ဆိုတာ Unordered List ဆိုတဲ့သဘောဖြစ်ပြီး <ol> ကတော့ Ordered List ဆိုတဲ့သဘောပါ။ ဒါကြောင့် သေချာအစီအစဉ် စီထားပြီးသားစာရင်းတွေ ထည့်ချင်ရင် <ol> နဲ့ထည့်ပြီး ကြိုတင်စီထားခြင်းမရှိတဲ့ စာရင်းတွေ ထည့်ချင်ရင် <ul> နဲ့ထည့်နိုင်ပါတယ်။ သူတို့ရဲ့အထဲမှာ List Item အနေနဲ့ <li> Element ကိုသုံးပြီး Item တွေ တန်းစီ ထည့်ပေးရခြင်းဖြစ်ပါတယ်။ Browser တွေက ဖော်ပြတဲ့အခါ <ul> နဲ့ထည့်ထားတဲ့ List တွေကို Bullet နဲ့ပြပြီး <ol> နဲ့ ထည့်ထားတဲ့ List တွေကိုတော့ Number နဲ့ပြပေးလေ့ရှိပါတယ်။

List ဆိုတာမျိုးက နှစ်ဆင့်သုံးဆင့်လည်း ရှိတတ်ပါတယ်။ ဥပမာအားဖြင့် ဒီလိုပါ-

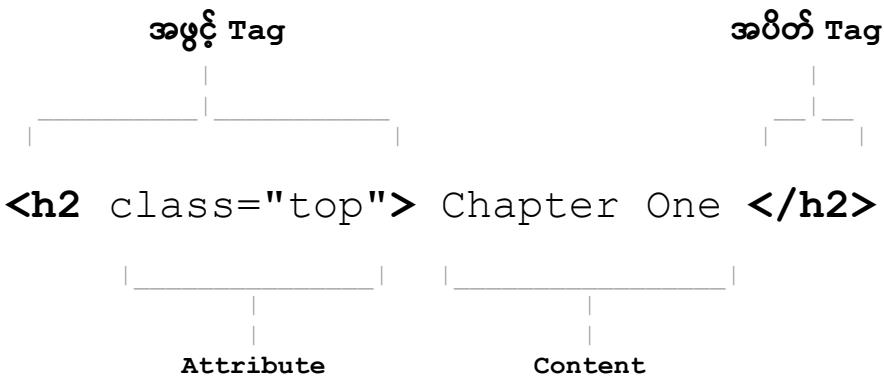
### HTML

```
<ul>
    <li>Web Browser</li>
    <li>
        HTTP Server
        <ol>
            <li>CGI</li>
            <li>MIME</li>
        </ol>
    </li>
</ul>
```

ဒီကုဒ်အရ <ul> အတွင်းမှာ <li> နှစ်ခုရှိပြီး ဒုတိယ <li> အတွင်းမှာ Content နဲ့အတူ နောက်ထပ် List တစ်ခုက <ol> နဲ့ ထပ်ဆင့်ရှိနေတာပါ။ ဒီလောက်ဆိုရင် အသုံးများမဟု အခြေခံ Element တွေ တော်တော် ပါသွားပါပြီ။

## Element Structure

နောက်ထပ် ကြည့်သင့်တဲ့ Element တွေအကြောင်း ထပ်မဖြောဆင် Element တွေရဲ့ ဖွဲ့စည်းပုံအကြောင်း အရင်ပြောပါ၌ီးမယ်။ HTML Element တစ်ခုမှာ အများအားဖြင့် အခုလို အပိုင်း (၄) ပိုင်း ပါဝင်လေ့ရှိပါ တယ်။



အဖွင့် Tag နဲ့ အပိုင် Tag ကြားထဲမှာ Content တည်ရှိပြီး၊ အဖွင့် Tag ထဲမှာ Attribute လိုက်တဲ့ သတ်မှတ်ချက်တွေ ပါဝင်နိုင်ပါတယ်။ Content နဲ့ အပိုင် Tag မပါတဲ့ Element တွေလည်း ရှိပါသေး တယ်။ Void Tag, Empty Tag စသေဖြင့် နှစ်မျိုးခေါ်ကြပါတယ်။ ဟိုအပေါ်မှာ ပေးခဲ့တဲ့ နမူနာထဲက `<meta>` Element ဟာ Empty Element အမျိုးအစား ဖြစ်ပါတယ်။ သူမှာ အဖွင့်နဲ့ Attribute သာပါပြီး Content နဲ့ အပိုင် မပါဝါဘူး။ HTML မှာ Element ပေါင်း (၁၀၀) လောက်ရှိသလို၊ Attribute တွေလည်း အများကြီး ရှိနေပါတယ်။

Attribute တွေကို ပုံစံနှစ်မျိုးနဲ့ တွေ့နိုင်ပါတယ်။ အပြည့်အစုံရေးသားခြင်းနဲ့ အတိုကောက် ရေးသားခြင်း တို့ ဖြစ်ပါတယ်။ အပြည့်အစုံရေးတဲ့အခါ ရှေ့က Attribute Property လာပြီး နောက်က Attribute Value လာရပါတယ်။ ဒီလိုပါ -

### HTML

```
<p class="alert" id="note"> ... </p>
```

နမူနာအရ <p> Element မှာ class နဲ့ id ဆိုတဲ့ Attribute နှစ်ခုရှိပြီးတော့ Value တွေလည်း ကိုယ်စိုက်  
ကြပါတယ်။ Attribute Value တွေကို Quote အဖွင့်အပိတ်နဲ့ ရေးပေးရပါတယ်။ အများအားဖြင့် Quote  
အဖွင့်အပိတ် မပါရင်လည်း အလုပ်လုပ်ပေမယ့်၊ Value မှာ Space ပါနေရင် Quote မပါလို့မရတော့ပါဘူး။  
ရှေ့ Attribute ရဲ့ Value နဲ့ နောက် Attribute ရဲ့ Property လဲ ရောသွားတတ်ပါသေးတယ်။ ဒါကြောင့်  
အလေ့အကျင့်ကောင်းအနေနဲ့ Attribute Value တိုင်းကို Quote ထဲမှာ ထည့်ပြီး ရေးပေးရပါတယ်။ HTML  
Element တွေနဲ့ Attribute Property တွေဟာ Case Insensitive ဖြစ်ပါတယ်။ အကြီးအသေး ကြိုက်  
သလိုရေးလို့ရပါတယ်။ Attribute Value တွေမှာတော့ အကြီးအသေး လွှဲလို့ မရတာတွေ ရှိပါတယ်။ ဒါ  
ကြောင့် တစ်ညီတည်း ဖြစ်သွားအောင် မှတ်ထားပေးပါ။ [HTML Element နဲ့ Attribute တွေအားလုံးကို](#)  
[စာလုံး အသေးတွေနဲ့ချဉ်းပဲ အမြဲတမ်း ရေးသင့်ပါတယ်။](#) Empty Element တွေရဲ့ ဖွဲ့စည်းပုံက ဒီလိုပါ။

**HTML**

```

<input type="text" value="22" name="age">
<br>
<hr>
```

Empty Element တွေမှာ Attribute ရှိနိုင်သလို၊ မရှိရင်လည်း ရပါတယ်။ အရင်တုံးက XHTML လိုခေါ်တဲ့  
HTML မူကဲ့တစ်မျိုးရှိခဲ့ဖူးပါတယ်။ အဲဒီမူကဲ့မှာဆိုရင် ဒီလို Empty Element တွေကို Self Close လုပ်ပေးရ<sup>1</sup>  
မယ်လို့ သတ်မှတ်ထားပါတယ်။ Self Close ဆိုတာ ဒီလိုပါ။

**HTML**

```

<input type="text" value="22" name="age" />
<br />
<hr />
```

အပိတ်မရှိဘူးဆိုတာ ပေါ်လွင်အောင် တစ်ခါတည်း ပိတ်ပေးလိုက်တဲ့ သဘောပါ။ အခုတော့ အဲဒီလို  
တစ်ခါတည်း ပိတ်ပြီးရေးပေးစရာ မလိုအပ်တော့ပါဘူး။ အကယ်၍များ အဲဒီလို ပိတ်ပြီးရေးထားတာမျိုး  
တွေရင်လည်း သူအကြောင်းနဲ့သူ ရှိတယ်ဆိုတာကို သိစေဖို့အတွက် ထည့်ပြာပြာတာပါ။ React လို  
JavaScript Framework မျိုးမှာဆိုရင် Empty Element တွေကို တစ်ခါတည်း ပိတ်ပေးရမယ်ဆိုတဲ့  
သတ်မှတ်ချက်မျိုး ရှိနေပါတယ်။

## Important Elements

ပြီးခဲ့တဲ့အပိုင်းမှာ Empty Element တစ်ခုဖြစ်တဲ့ <img> ကို နမူနာပေးခဲ့ပါတယ်။ ပုံတွေ ထည့်သွင်းဖို့ အတွက် အသုံးပြုရတဲ့ Element တစ်ခုဖြစ်ပါတယ်။ HTML Element တွေမှာ အများအားဖြင့် Attribute ဆိတာ လိုဂ်င်သုံး၊ မလိုဂ်မသုံးဘဲ နေလိုဂ်တယ်ဆိုတဲ့သော့ ရှိပေမယ့် တစ်ချို့ Element တွေမှာတော့ မပါမဖြစ်ပါရမယ့် Attribute တွေရှိပါတယ်။ <img> Element မှာ src Attribute မပါအဖြစ်ပါဝင်ရမှာ ဖြစ်ပါတယ်။ src Attribute ကိုသုံးပြီး ဖော်ပြခေါ်လိုတဲ့ ပုံရဲ့တည်နေရာကို ပေးရမှာမို့လိုပါ။ URL လိပ်စာ အပြည့်အစုံအနေနဲ့ ပေးနိုင်သလို၊ ဖိုင် Path လမ်းကြောင်းအနေနဲ့လည်း ပေးနိုင်ပါတယ်။ ထူးခြားချက် အနေနဲ့ alt Attribute လည်း ပါဝင်သင့်တယ်လို့ မှတ်သားထားရပါမယ်။ alt ဟာ Alternative Text ဆိုတဲ့ အဓိပ္ပာယ်ဖြစ်ပြီး ပုံရဲ့ကိုယ်စား အစားထိုးသုံးလိုဂ်ရနိုင်မယ့် စာကိုပေးရမှာပါ။ HTML ဟာ ကွန်ပျူးတာစနစ် တွေက နားလည်နိုင်တဲ့ Content တွေဖွဲ့စည်းဖို့လို့ ပြောခဲ့ပါတယ်။ အကယ်၍များ ကွန်ပျူးတာစနစ်က Text Only ပဲ နားလည်တဲ့စနစ် ဖြစ်နေလို့ ပုံတွေကို ဖော်ပြနိုင်ခြင်းမရှိဘူးဆိုရင် src အစား alt ကို အစားထိုး အသုံးပြုနိုင်ဖို့ဆိုတဲ့ ရည်ရွယ်ချက်မျိုးနဲ့ပါ။ alt မဖြစ်မနေပါရမှာ မဟုတ်ပေမယ့် အလေ့အကျင့်ကောင်း တစ်ခုအနေနဲ့ ထည့်ပေးသင့်ပါတယ်။ စမ်းကြည့်ချင်ရင် ဒီနမူနာတွေကို ကူးပြီးစမ်းကြည့်လို့ရပါတယ်။

### HTML

```



```

ပေးထားတဲ့ နမူနာရဲ့ နောက်ဆုံးတစ်ခုမှာ src က မှားနေပါလိမ့်မယ်။ api.jpg ဆိုတဲ့ဖို့င် မရှိပါဘူး။ api.png ပဲရှိပါတယ်။ အဲဒီ အများအတိုင်း ကူးယူပြီး Browser မှာစမ်းသပ်ကြည့်ရင် ပုံက မှားနေလို့ မပြနိုင်တဲ့အတွက် alt မှာ ပေးထားတဲ့စာကို အစားထိုး ပြပေးတယ်ဆိုတာကို တွေ့ရနိုင်ပါတယ်။

```
• HTML
1 
2
3 
4
5 
6
```

• CSS  
• JS



API Book Cover

နောက်ထပ် အရေးပါတဲ့ Element ကတော့ `<a>` Element ဖြစ်ပါတယ်။ Anchor ရဲ့အတိုကောက်ဖြစ်ပြီး Link တွေထည့်စိုး သုံးပါတယ်။ HTML ကိုတိတောင်တဲ့ အဓိကရည်ရွယ်ချက်က ဒီ Link တွေလို့ ပြောလို့ရပါတယ်။ Document တွေ အပြန်အလှန် ချိတ်လို့ရတဲ့၊ ညွှန်းလို့ရတဲ့ ဒီလုပ်ဆောင်ချက်ဟာ ပေါ့သေးသေး မဟုတ်ပါဘူး။ WorldWideWeb ခေါ် အပြန်အလှန်ချိတ်ဆက်နေတဲ့ ကွန်ယက်စနစ်ကြီးက ဒီ Link တွေကို အခြေခံပြီး ဖြစ်ပေါ်လာတာပါ။ ရေးသားပုံရေးသားနည်းက ဒီလိုပါ။

### HTML

```
<a href="https://www.google.com/" title="Search">Google</a>
```

သူမှာ Attribute နှစ်ခုပါပါတယ်။ `href` Attribute က မဖြစ်မနေပါရမှာပါ။ ချိတ်ချင်တဲ့လိပ်စာကို `href` မှာ ပေးရမှာပါ။ `title` Attribute ကတော့ `<img>` ရဲ့ `alt` လိုပါပဲ။ မပါလည်း ရပေမယ့် ပါရင်ပိုကောင်းပါတယ်။ Link က ညွှန်းထားပေမယ့် မသွားနိုင်တဲ့အခါ၊ မသွားချင်တဲ့အခါ၊ အဲဒီအညွှန်းရဲ့ အဓိပ္ပာယ်ကို `title` မှာ ကြည့်လိုက်နိုင်မှာ ဖြစ်ပါတယ်။ ဒီ `title` Attribute ကို မည်သည့် HTML Element မှာမဆိုသုံးလို့ရပါတယ်။ Browser တွေက Element ကို Mouse Pointer ထောက်လိုက်ရင် `title` မှာပေးထားတဲ့ တန်ဖိုးကို Tooltip လေးနဲ့လည်း လာပြေားကြပါတယ်။



ဒီ <a> Element ကိုသုံးပြီး URL လိပ်စာ (သိမဟတ်) Path လမ်းကြောင်းသိတဲ့ ဘယ်လို Content အမျိုးအစားကိုမဆို ချိတ်လို ညွှန်းလိုရပါတယ်။ ဝတ်ဆိုက်တွေ၊ HTML Document တွေမှ မဟုတ်ပါဘူး။ ပုံတွေ၊ ဖိုင်တွေကို ညွှန်းချင်ရင်လည်း ညွှန်းလိုရတာပါပဲ။ URL/Path ပေးဖို့ပဲလိုပါတယ်။

Link တွေကိုဖော်ပြတဲ့အခါမှာ စာလုံးအပြောရောင်/စာလုံးခရမ်းရောင် အရောင်နှစ်မျိုးနဲ့ Browser တွေကပြပေးပါတယ်။ မသွားဘူးသေးတဲ့ Link အသစ်ဆိုရင် စာလုံးအပြောရောင်နဲ့ ပြပြီး သွားဖူးတဲ့ Link ဆိုရင် ခရမ်းရောင်နဲ့ ပြပေးတာပါ။ Underline လည်း တားပြီးတော့ ပြပေးပါသေးတယ်။ ဒါကြောင့် အသုံးပြုသူ တွေက စာလုံးအပြောရောင်/ခရမ်းရောင်ကို Underline တားထားရင် နှိပ်လိုရတဲ့ Link ပဲဆိုတာ အလိုလို သိနေကြပါပြီ။ ဒီလို အလိုလိုသိနေတဲ့အတွက် သတိထားရမှာက Link မဟုတ်တဲ့ စာတွေကို စာလုံးအပြော/ခရမ်းရောင် မသုံးမိစေဖို့နဲ့၊ Underline မတားမိစေဖို့ပဲဖြစ်ပါတယ်။ အသုံးပြုသူက နှိပ်လိုရတဲ့ Link မှတ်ပြီး တစ်ကယ်တမ်း နှိပ်မရတဲ့အခါ သူရဲ့သဘာဝအသိနဲ့ ဆန့်ကျင်နေလို စိတ်ညွစ်သွားတတ်ပါတယ်။ သတိပြုသင့်တဲ့ အကြောင်းအရာတစ်ခုအနေနဲ့ ထည့်သွင်းမှတ်သားဖို့ဖြစ်ပါတယ်။

ထူးခြားချက်အနေနဲ့ ဒီ Link တွေကို သိပ်ရှည်တဲ့ Document တွေရဲ့ တစ်နေရာကနေ နောက်တစ်နေရာကို လှမ်းညွှန်းဖို့လည်း သုံးနိုင်ပါသေးတယ်။ တစ်ခြား Document ကိုလှမ်းညွှန်းတာ မဟုတ်တော့ဘဲ၊ ဒီ Document ထဲကပဲ တစ်ခြားနေရာကို ညွှန်းတဲ့သဘောပါ။ ဒီအတွက် id Attribute နဲ့ တွဲသုံးနိုင်ပါတယ်။ ဥပမာ - အခုလို Element တစ်ခုရှိတယ်ဆိုပါစို့။

**HTML**

```
<h2 id="ch2">Chapter Two</h2>
```

ဒီ Element ကို လုမ်းညွှန်းတဲ့ Link ကို အခုလိုရေးသားနိုင်ပါတယ်။

**HTML**

```
<a href="#ch2" title="Go to Chapter Two">Chapter Two</a>
```

href Attribute အတွက် တန်ဖိုးပေးတဲ့အခါ # သက်တန်အတူ id ကိုတဲ့ပေးခြင်းအားဖြင့် ညွှန်းနိုင်တာ ဖြစ်ပါတယ်။ Document တစ်ခုထဲမှာညွှန်းချင်ရင် နမူနာမှာပေးထားသလို #id ကိုပေးလိုက်ရင် ရပါပြီ။ တစ်ခြား Document ထဲက Element ကို လုမ်းညွှန်းချင်ရင်လည်း အခုလို ညွှန်းနိုင်ပါတယ်။

**HTML**

```
<a href="https://en.wikipedia.org/wiki/HTML">HTML</a>
<a href="https://en.wikipedia.org/wiki/HTML#Elements">Elements</a>
```

ပေးထားတဲ့နမူနာနှစ်ခုမှာ URL တွေကို ကရပြုကြည့်ပါ။ ပထမ URL က Wikipedia ရဲ့ HTML Article ကို ညွှန်းတဲ့ URL ဖြစ်ပြီး ဒုတိယ URL က အဲဒီ HTML Article ထဲက Elements ဆိုတဲ့အပိုင်းကို ညွှန်းထားတယ်ဆိုတာကို တွေ့နိုင်ပါတယ်။ ဒီနည်းနဲ့ <a> Element တွေကိုသုံးပြီး Document တွေ အပြန်အလှန် ချိတ်လို့ ညွှန်းလို့ ရသလို့ Document ထဲက Element တွေကိုထိ အတိအကျည်းလို့ရခြင်း ဖြစ်ပါတယ်။

**Generic Elements**

ဆက်လက်ပြီးတော့ အသုံးများပြီး အသုံးဝင်တဲ့ Element တွေကို ဖော်ပြပေးပါမယ်။

**<div>** - ဒီ Element ကို Generic Element လိုပေါ်ပါတယ်။ အသုံးအများဆုံး Element တစ်ခုပါ။ Generic Element ဆိုတာ ဘယ်လိုနေရာမျိုးမှာ သုံးဖို့အတွက်ရယ်လို့ တိတိကျကျ သတ်မှတ်ထားခြင်း မရှိဘဲ၊ လိုအပ်တဲ့ နေရာတိုင်းမှာ သုံးလို့ရတဲ့ Element ကို ဆိုတာပါ။ HTML ဆိုတာ ကွန်ပျူးတာစနစ်တွေက နားလည်တဲ့ Content Structure တည်ဆောက်ဖို့အတွက် Language တစ်မျိုးလို့ အထက်မှာ ပြောခဲ့ပါတယ်။ လက်တွေ့မှာတော့ HTML ကို Content Structure တည်ဆောက်ဖို့အတွက် သာမက App UI

တည်ဆောက်ဖို့ သုံးနေကြပါတယ်။ ဒီတော့ ထွင်ထားတဲ့ရည်ရွယ်ချက်ကတစ်မျိုး၊ လက်တွေ့သုံးနေကြတာ ကတစ်မျိုးဖြစ်နေတဲ့ သဘောပါပဲ။ ဒါကြောင့်လည်း <div> Element ကို အရမ်းအသုံးများတာပါ။ App UI အတွက် သင့်တော်တဲ့ Element တွေ HTML မှာ သိပ်မှမပြည့်စုံတာ။ ထွင်ထားတာ ဒီအတွက် ထွင်ထားတာ မဟုတ်ဘူးလေ။ Menubar တစ်ခုထည့်ချင်လား၊ <div> ကိုသုံး။ Menubar အတွက်သတ်မှတ်ထားတဲ့ သီးခြား Element မရှိလိုပါ။ Toolbar လေးတစ်ခုထည့်ချင်လား၊ <div> ကိုသုံး။ Toolbar ထည့်ဖို့အတွက် သီးခြား Element မှမရှိတာ။ စသဖြင့် App UI တည်ဆောက်ဖို့အတွက် ဆိုရင် နေရာတိုင်းမှာ <div> ကိုပဲ သုံးကြရပါတယ်။ အသုံးလွန်ပြီး မသုံးသင့်တဲ့ နေရာတွေမှာပါ သုံးကြတဲ့အထိပါပဲ။ ဥပမာ - ခေါင်းစီးထည့်ချင်ရင် <h1> <h2> စသည်ဖြင့် သုံးသင့်ပေမယ့် <div> ကိုပဲသုံးလိုက်တာတို့၊ စာပိုဒ်ထည့်ချင်ရင် <p> ရှိရှိသားနဲ့ <div> လည်း သုံးလိုက်တာတို့၊ Button တစ်ခုထည့်ချင်ရင် <button> ရှိရှိသားနဲ့ <div> ကိုသုံးလိုက်တာတို့ ရှိနေကြပါတယ်။ <div> ဟာ စွဲယုံစုံသုံးလိုရတဲ့ အသုံးဝင်တဲ့ Element တစ်ခုဖြစ်ပေမယ့် Abuse မလုပ်မဖို့တော့ သတိထားရပါလိမ့်မယ်။

**<span>** - <span> ဟာလည်း <div> လိုပဲ Generic Element ပါပဲ။ <div> ကို တင်ချင်ရာတင်လိုရတဲ့ စားပွဲတစ်လုံးလို့ သဘောထားမယ်ဆိုရင် <span> ကိုတော့ ထည့်ချင်ရာထည့်လိုရတဲ့ ပန်းကန်တစ်လုံးလို့ သဘောထားနိုင်ပါတယ်။ ပန်းကန်ထဲမှာ အသီးအနှံဆို အသီးအနှံပဲထည့်မယ်၊ သကြားလုံးဆို သကြားလုံးပဲ ထည့်မယ် မဟုတ်လား။ စားပွဲပေါ်မှာတော့ အသီးအနှံထည့်ထားတဲ့ ပန်းကန်တွေရော၊ အိုးတွေရော၊ ခွက်တွေရော အကုန်တင်မယ် မဟုတ်လား။ အဲဒီလိုကာပါတယ်။ တစ်ကယ့်ကွာခြားပုံကို ခုနေနည်းပညာသဘောက ပြောရင် ရှုပ်နေမှာစိုးလိုပါ။ နောက်တစ်ခန်းကျတော့မှ ထပ်ပြောပါမယ်။

## Layout Elements

HTML Document တွေတည်ဆောက်တဲ့အခါ Content ချဉ်းမဟုတ်ဘဲ Layout လေးတွေလည်း ထည့်သွင်း တည်ဆောက်နိုင်ပါတယ်။ မူလ HTML မှာ Layout အတွက်ရည်ရွယ်ထားတဲ့ Element ရယ်လို့ သီးခြားမပါဝင်တဲ့အတွက် <div> တွေကို Layout အတွက် သုံးခဲ့ကြရပါတယ်။ HTML5 ဆိုပြီး ထွက်ပေါ်လာတဲ့ နောက်ပိုင်းမှာတော့ Layout Element တွေကို ထည့်သွင်းပေးလာပါတယ်။ အသုံးများတဲ့ Layout Element တွေရဲ့ပုံစံက ဒီလိုပါ။



HTML Document တွေကိုအသုံးပြုပြီး ဝေါ်ဆိုက်တွေ တည်ဆောက်တဲ့အခါ Logo တွေ ခေါင်းစီးတွေ၊ Hotline နံပါတ်လို့ ဆက်သွယ်ရမယ့် အချက်အလက်တွေ၊ ဆောင်ပုဒ်လို့ အရာတွေကို ဟိုးအပေါ်မှာ Document Header အနေနဲ့ ထည့်ကြတာ ထဲးစံပါပဲ။ အဲဒီလိုသဘောမျိုးနဲ့ အသုံးပြုဖို့လိုအပ်ရင် <header> Element ကိုသုံးနိုင်ပါတယ်။ သူ့နဲ့ ပြောင်းပြန်က <footer> Element ဖြစ်ပါတယ်။ ဟိုးအောက်ဆုံးမှာ ထားကြလေ့ရှိပြီး Copyright တို့ Privacy Policy တို့လို့ အကြောင်းအရာတွေ ထည့်သွင်းဖော်ပြုကြလေ့ရှိပါတယ်။

နမူနာပုံရဲ့ ဘေးတစ်ဘက်တစ်ချက်မှာ <nav> နဲ့ <aside> တို့ကို ပေးထားပါတယ်။ နမူနာပါ။ လက်တွေ့မှာ <nav> တို့ <aside> တို့ဆိုတာ အဲဒီလို ဘေးမှာထားရတယ်ဆုံးတဲ့သဘော ပုံသေမဟုတ်ပါ

ဘူး။ ကိုယ်ကြိုက်တဲ့ နေရာမှာထားပါ။ သူအမိပြုယ်ကို သိဖိုပဲလိုပါတယ်။ <nav> ကိုတော့ ဝတ်ဆိုက် မှာ ပါဝင်တဲ့ စာမျက်နှာတွေ အပြန်အလုန်သွားလိုက်တဲ့ Navigation Link တွေ စုစည်းထည့်သွင်းဖို့ သုံးရပါ တယ်။ <aside> ကိုတော့ ပင်မ Content နဲ့ ဆက်စပ်နေတဲ့ Relevant Content တွေ စုစည်းထည့်သွင်းဖို့ သုံးရပါတယ်။

<section> ဆိုတာ Document အတွင်းမှာရှိတဲ့ သီးခြားခဲ့ထုတ်လိုတဲ့ အစိတ်အပိုင်းလို ဆိုနိုင်ပါတယ်။ သူမှာ သူကိုယ်ပိုင် <header> တွေ <footer> တွေ ရှိလိုရပါတယ်။ <article> ကတော့ စာတွေ ထည့်သွင်းဖိုပါ။ <p> Element ကို စာပိုဒ်တွေထည့်ဖိုလို ပြောထားပါတယ်။ <article> ကိုတော့ အဲဒီ <p> Element တွေနဲ့ ထည့်ထားတဲ့ စာပိုဒ်တွေ ခေါင်းစီးတွေ၊ ခေါင်းစီးအခဲ့တွေ စုစည်းဖို့ Element လို သဘောထားနိုင်ပါတယ်။

နမူနာ Layout ပုံမှာ မပါပေမယ့် နောက်ထပ်အသုံးများတဲ့ Element နှစ်ခု ရှိပါသေးတယ်။ <hgroup> နဲ့ <main> ပါ။ <hgroup> ကို ခေါင်းစီးတွေ တစ်ခုထက်ပိုတဲ့အခါ စုစည်းဖိုသုံးနိုင်ပါတယ်။ ဥပမာ -

## HTML

```
<hgroup>
  <h1>Maing Heading</h1>
  <h3>Minor Heading</h3>
  <h2>Secondary Heading</h2>
</hgroup>
```

<main> Element ကိုတော့ အဓိကကျတဲ့ Content အားလုံးကို စုစည်းဖို့ အသုံးပြုနိုင်ပါတယ်။ နမူနာပုံမှာ ပါတဲ့ <section> သဘောမျိုးပါပဲ။ ကွာသွားတာက <section> က သီးခြား ရပ်တည်နိုင်တဲ့ အစိတ်အပိုင်းတစ်ခုသဘောမျိုးဖြစ်ပြီး <main> ကတော့ လက်ရှိ Document ရဲ့ အဓိကအစိတ်အပိုင်း ဆုံး တဲ့ သဘောမျိုးပါ။

ဒါ Element တွေကို သုံးလိုက်ယုံနဲ့ နမူနာပုံမှာပြထားသလို Layout အသွင်အပြင် ရသွားမှာ မဟုတ်ဘူးဆို တာကို တစ်ခါတည်း မှတ်ဖို့ လိုပါလိမ့်မယ်။ HTML ဆိုတာ Content ကို စုစည်းဖိုသာ ဖြစ်ပါတယ်။ အခြေခံ အားဖြင့် အသွင်အပြင်ဆိုတာ သူနဲ့တိုက်ရှိက်မဆိုင်ပါဘူး။ ဘယ် Content က ဘာဆိုတာကို သတ်မှတ်ပေး ယုံသာ သတ်မှတ်ပေးတဲ့ သဘောပါ။ လိုချင်တဲ့ Layout အသွင်အပြင်အတွက်ကတော့ ကိုယ့်ဘာသာ CSS

လို Style Language တွေနဲ့ သတ်မှတ်ပေးရမှာပါ။ ဒီစာအုပ်မှာ အဓိကလေ့လာချင်တဲ့ Bootstrap လို နည်းပညာမျိုးနဲ့လည်း သတ်မှတ်ပေးနိုင်ပါတယ်။ နောက်ပိုင်းမှာ ဆက်လက်ဖော်ပြပါမယ်။

## Table Elements

Table ဟာလည်းပဲ Content တွေဖော်ပြဖို့အတွက် အရေးပါတဲ့ Element တစ်ခုပါပဲ။ တစ်ချို့ Table ပေါ်ပြရမယ့် အချက်အလက်တွေဆိုတာ ရှိလာမှာပါပဲ။ Table တစ်ခုသတ်မှတ်ဖို့အတွက် လိုအပ်တဲ့ Element (၄) မျိုး ရှိပါတယ်။ <table>, <tr>, <th> နဲ့ <td> တို့ဖြစ်ပါတယ်။ <tr> ဆိုတာ Table Row ဆိုတဲ့ အဓိပ္ပာယ်ဖြစ်ပြီး ကိုယ်သတ်မှတ်လိုတဲ့ Table မှာ Row (၃) ခုရှိရင် <tr> အဖွင့်အပိတ် (၃) စုံရှိရမှာပါ။ <th> နဲ့ <td> က သဘောသဘာဝဆင်ပါတယ်။ Table Row တစ်ခုအတွင်းထဲမှာ Data Column ဘယ်နှစ်ခုရှိသလဲဆိုတာကို <th> တို့ <td> တို့နဲ့ သတ်မှတ်ပေးရတာပါ။ <th> က Table Heading ဖြစ်ပြီး <td> က Table Data ဖြစ်ပါတယ်။ ရေးသားပုံက ဒီလိုပါ -

### HTML

```
<table>
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Alice</td>
    <td>22</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Bob</td>
    <td>23</td>
  </tr>
</table>
```

ဒီကုဒ်ကိုမစမ်းခင် ဖတ်ကြည့်လိုက်ရင်ပဲ သဘောသဘာဝ ပေါ်လွင်ပါတယ်။ Table တစ်ခုရှိပြီး Row သုံးခု ရှိပါလိမ့်မယ်။ Row တစ်ခုချင်းစီမှာ Data Column (၃) ခုစုံရှိပြီး၊ အပေါ်ဆုံး Row ထဲက Column တွေက Heading Column တွေ ဖြစ်ပါတယ်။ စမ်းကြည့်လိုက်ရင် ရလဒ်က ဒီလိုဖြစ်မှာပါ။



```

1▼ <table border="1" width="50%">
2▼   <tr>
3▼     <th>ID</th>
4▼     <th>Name</th>
5▼     <th>Age</th>
6   </tr>
7▼   <tr>
8▼     <td>1</td>
9▼     <td>Alice</td>
10▼    <td>22</td>
11   </tr>
12▼   <tr>
13▼     <td>2</td>
14▼     <td>Bob</td>
15▼    <td>23</td>
16   </tr>
17 </table>
18

```

The CSS and JS tabs are visible at the bottom of the editor.

ID	Name	Age
1	Alice	22
2	Bob	23

ပုံထဲကနူမူနာမှာ border နဲ့ width Attribute တွေ ထည့်ထားတာကို သတိပြုပါ။ မဖြစ်မနေလိုအပ်လို မဟုတ်ပါဘူး။ အဲဒါလေးတွေပါမှ ရလဒ်က ကြည့်ရတာ အဆင်ပြေမှာမိုလိုသာ ထည့်ထားတာပါ။ ကိုယ့် ဘာသာ တန်ဖိုးတွေပြောင်းပြီး စမ်းကြည့်လို ရပါတယ်။

ပိုပြီးတော့ ပြည့်စုံချင်ရင် <thead>, <tbody> နဲ့ <tfoot> လို Element တွေကို သုံးနိုင်ပါတယ်။ တစ်ချို့ ပေါင်းပြီးပြဖို့ လိုတဲ့ Column တွေအတွက် colspan Attribute ကိုသုံးနိုင်ပါတယ်။ တစ်ချို့ ပေါင်းပြီးပြဖို့ လိုတဲ့ Row တွေအတွက် rowspan Attribute ကို သုံးနိုင်ပါတယ်။ align Attribute ကိုသုံးပြီးတော့ Column တစ်ခုချင်းစီမှာပါတဲ့ Content တွေကို ဘယ်ညာ့၊ အလယ် စီထားလိုလည်းရပါ တယ်။ ဒါတွေအစုံပါတဲ့ နမူနာလေးတစ်ခုပေးပါမယ်။

**HTML**

```

<table>

  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>Age</th>
    </tr>
  </thead>

```

```

<tbody>
    <tr>
        <td>1</td>
        <td>Alice</td>
        <td>22</td>
    </tr>
    <tr>
        <td>2</td>
        <td>Bob</td>
        <td>23</td>
    </tr>
</tbody>

<tfoot>
    <tr>
        <th colspan="2" align="right">Total</th>
        <td>2</td>
    </tr>
</tfoot>

</table>

```

<thead>, <tbody>, <tfoot> တွေကိုသုံးကို Row တွေကို စုစုည်းပေးလိုက်တာပါ။ ဒီလိုစုစုည်းပေးလိုက်လို Table ရဲ့အသွင်အပြင်ဖော်ပြပုတော့ မပြောင်းပါဘူး။ အချက်အလက်တွေ စုစုစုည်းစည်း ဖြစ်သွားခြင်းသာ ဖြစ်ပါတယ်။ အသွင်အပြင်ပြောင်းမှာက အောက်ဆုံး Row မှာပါတဲ့ <th> ပါ။ colspan=2 လို့ ပြောထားတဲ့အတွက် သူက Column နှစ်ခုစာ နေရာယူမှာပါ။ ဒါကြောင့်လည်း အောက်ဆုံး Row မှာ Column (၃) ခုမရှိဘဲ (၂) ခုပဲ ရှိနေတာပါ။ Column တစ်ခုက သူတစ်ခုထဲ နှစ်ခုစာနေရာယူမှာ မိုလိုပါ။ ရလဒ်က ဒီလိုဖြစ်ပါလိမ့်မယ်။



```

1<table border="1" width="50%">
2  <thead>
3    <tr>
4      <th>ID</th>
5      <th>Name</th>
6      <th>Age</th>
7    </tr>
8  </thead>
9  <tbody>
10 <tr>
11   <td>1</td>
12   <td>Alice</td>
13   <td>22</td>
14 </tr>
15 <tr>
16   <td>2</td>
17   <td>Bob</td>
18   <td>23</td>
19 </tr>
20 </tbody>

```

The code editor shows an HTML file with Bootstrap imports. The HTML content defines a table with three columns: ID, Name, and Age. It contains two rows of data: Alice (ID 1, Age 22) and Bob (ID 2, Age 23). A summary row at the bottom shows a total of 2.

ID	Name	Age
1	Alice	22
2	Bob	23
<b>Total</b>		2

Row တွေပေါင်းတဲ့ rowspan တော့ နမူနာပေးတဲ့အထဲ မပါပါဘူး။ လိုအပ်ချက်နည်းပါတယ်။ မဖြစ်မနေ လိုအပ်လာတော့မှာသာ ဆက်လေ့လာကြည့်လိုက်ပါ။

## Form Elements

ရိုးရိုးစာရွက်တွေမှာ လက်ရေးနဲ့ ရေးဖြည့်လိုရတဲ့ ဖောင်တွေရှိသလိုပဲ HTML မှာလည်း User က ရေးဖြည့် လိုရတဲ့၊ ရွေးလိုရတဲ့ ဖောင်တွေ ရှိပါတယ်။ ဒီဖောင်တွေကို အားကိုးပြီး Web Application တွေထိ ဖုန်းကြရတာဆိုတော့ ကျယ်ပြန့်ပါတယ်။ အသုံးများတဲ့ ဖောင် Element တွေကို ရွေးထုတ်လေ့လာကြပါမယ်။ ဒီ Element (၅) မျိုးကို လေ့လာရမှာပါ။

- <label>
- <input>
- <textarea>
- <select>
- <button>

<input> Element ဟာ Empty Element ဖြစ်ပြီးတော့ type Attribute မဖြစ်မနေ ပါရပါတယ်။ text, password, radio, checkbox, email, url, date, submit, reset စာဖြင့် Value တွေအများကြီးရှိပါတယ်။ အဲဒီထဲက လက်ရှိ အဆင့်မှာ ရွေးချယ်မှတ်သားသင့်တာက text, password နဲ့ submit ဖြစ်ပါတယ်။

<input> တို့ <textarea> တို့ <select> တို့ကို <label> နဲ့ တွဲသုံးလေ့ ရှိပါတယ်။ ဒီလိုပါ -

### HTML

```
<label for="name">Your Name</label>
<input type="text" id="name"> <br>

<label for="pwd">Password</label>
<input type="password" id="pwd"> <br>

<input type="submit" value="Login">
```

<label> တွေမှာ for Attribute ပါပြီး <input> တွေရဲ့ id နဲ့တူအောင်ပေးရတာကို သတိပြုကြည့်ပါ။ ဒီနည်းနဲ့ <label> နဲ့ <input> ကို တွဲရခြင်းဖြစ်ပါတယ်။ နမူနာ <input> သုံးခုမှာ type တွေမတူကြပါဘူး။ text ကတော့ စာတွေရှိက်ထည့်လိုရတဲ့ Input ဖြစ်ပြီး password ကလည်း စာတွေရှိက်ထည့်လိုရတဲ့ Input ပါပဲ။ ကွာသွားတာကတော့ password Input မှာ ရှိက်ထည့်လိုက်တဲ့စာတွေကိုဖျောက်ထားပေးမှာပါ။ submit ကတော့ နှိပ်လိုရတဲ့ ခလုပ်တစ်ခုဖြစ်ပါတယ်။ ကြားထဲမှာပါတဲ့ <br> Element ကတော့ Line Break ဆိုတဲ့အဓိပ္ပာယ်ပါ။ နောက်တစ်လိုင်းကို ဆင်းပေးပါတယ်။

ကြားဖြတ်ပြီးမှတ်ပေးပါ။ HTML Code ထဲမှာ Enter တွေ ခေါက်ပြီးလိုင်းတွေ ခဲ့ချင်သလောက်ခဲ့၊ Tab တွေနှိပ်ပြီး စပေါ့တွေ ထည့်ချင်သလောက်ထည့်၊ အဲဒါတွေကို ထည့်ပြီး အလုပ်မလုပ်ပါဘူး။ ကိုယ့်ဘက်က အခုလို တစ်လိုင်းဆင်းစေချင်ရင် ဆင်းစေချင်တဲ့အကြောင်း <br> Element နဲ့ပြောပေးမှပဲ ဆင်းပါတယ်။

လက်တွေ့မှာ ဒီ Input တွေအားလုံးကို <form> Element တစ်ခုနဲ့ စုစုံထားရတဲ့ ထုံးစံရှိပါတယ်။ ရလဒ်က ဒီလိုဖြစ်မှာပါ။

```

1▼ <form>
2▼  <label for="name">Your Name</label>
3  <input type="text" id="name"> <br>
4
5▼  <label for="pwd">Password</label>
6  <input type="password" id="pwd"> <br>
7
8  <input type="submit" value="Login">
9 </form>
10 |

```

Your Name   
 Password

CSS  
 JS

submit Input အတွက် value Attribute သုံးထားတာကိုလည်း သတိပြုပါ။ value မှာပေးထားတဲ့ တန်ဖိုးကို ခလုပ်ပေါ်ကစာအနေနဲ့ လာပြတာကို တွေ့ရှိပါတယ်။ လက်တွေ့မှာ value နဲ့အတူ တစ် ခြားအရေးကြီးတဲ့ placeholder, readonly, checked, required စတဲ့ Attribute တွေ ရှိ ပါသေးတယ်။ အဲဒါတွေကိုတော့ လက်တွေ့အသုံးချ ၁၁ Web Application တွေလေ့လာတဲ့ အဆင့်ရောက်ဖြစ် တော့မှာပဲ ဆက်လေ့လာပါ။ အခုကတော့ ဖောင်တစ်ခုအနေနဲ့ ရေးဖြည့်လိုက်တဲ့ အဆင့်ထိပဲ လေ့လာရမှာပါ။ ရေးဖြည့်လိုက်တဲ့တန်ဖိုးတွေကို သုံးပြီးတစ်ကယ်အလုပ်လုပ်ဖိုကတော့ PHP တိုဘာတို့လို Server-side နည်းပညာတွေနဲ့ ပူးတွဲလေ့လာကြရှိမှာပါ။ ဒါကြောင့် အခုထည့်သွင်းတဲ့ဖောင်တွေဟာ ဖော်ပြုယုံ သက်သက် ဖြစ်တယ်လို့ နားလည်ပါ။ လက်တွေ့အလုပ်လုပ်တဲ့ ဖောင်တွေတော့ မဟုတ်သေးပါဘူး။

<textarea> ဟာ အဖွင့်အပိတ်အပြည့်အစုံပါတဲ့ Element ဖြစ်ပါတယ်။ text Input တွေဟာ စာတစ် ကြောင်းပဲ ရေးဖြည့်ဖို့ သင့်တော်ပြီး၊ စာများများ ရေးဖြည့်ဖို့ လိုအပ်ရင် <textarea> ကို သုံးရမှာ ဖြစ်ပါ တယ်။ <select> ကိုတော့ ရွေးလိုက်တဲ့ List တစ်ခုထည့်သွင်းလိုတဲ့အခါ သုံးပါတယ်။ ရွေးရမယ့် Option တွေကို <option> Element သုံးပြီး သတ်မှတ်ရပါတယ်။ <ul> <li> ရေးပုံနဲ့ တူပါတယ်။ <ul> ထဲ မှာ <li> တွေ ရှိသလိုပဲ <select> ထဲမှာ <option> တွေ ရှိရမှာပါ။

<button> ကတော့ submit Input နဲ့ အတူတူပါပဲ။ ကွာသွားတာက အဖွင့်အပိတ် အပြည့်အစုံနဲ့ ရေး ပေးခြင်း ဖြစ်ပါတယ်။ သူမှာလည်း type Attribute ပါရပါတယ်။ ဒါတွေအားလုံး အပြည့်အစုံပါတဲ့ နမူနာ တစ်ခု ရေးပေးပါမယ်။

**HTML**

```
<form>
    <label for="name">Your Name</label> <br>
    <input type="text" id="name"> <br>

    <label for="gender">Your Gender</label> <br>
    <select id="gender">
        <option>Male</option>
        <option>Female</option>
    </select> <br>

    <label for="address">Your Address</label> <br>
    <textarea id="address"></textarea> <br>

    <button type="submit">Send</button>
</form>
```

<button> မှာ type က မထည့်လည်း ရတော့ရပါတယ်။ အလားတူပါပဲ <label> တွေမှာ for စာတမ်း ထည့်လည်း ဘာမှတော့ မဖြစ်ပါဘူး။ Input တွေမှာလည်း id မပါမဖြစ် မဟုတ်ပါဘူး။ နောက်ဆုံးဆင့် <label> Element ကို လုံးဝမသုံးလဲ စာတွေဒီအတိုင်း ချရေးလည်း ရတာပါပဲ။ ဒါပေမယ့် သတ်မှတ်ထားတဲ့အတိုင်း ဖုန်းအောင်ရေးတော့ ပိုစနစ်ကျသွားတာပေါ့။ ရလဒ်က ဒီလိုဖြစ်မှာပါ -

The screenshot shows a code editor interface. On the left, there is a vertical list of line numbers from 1 to 16. Next to each number is a portion of the HTML code. To the right of the code is a preview pane displaying the form. The preview includes:

- A text input field labeled "Your Name".
- A dropdown menu labeled "Your Gender" with "Male" selected.
- A text area labeled "Your Address".
- A "Send" button at the bottom right.

At the bottom of the editor, there are tabs for "CSS" and "JS".

ဒီလောက်ဆိုရင် အခြေခံ Form Element တွေ ရသွားပါပြီ။ ဒီထက်နည်းနည်းပို အဆင့်မြင့်တဲ့ တစ်ချို့ Element တွေကျန်သေးတယ်ဆိုတာကိုတော့ သတိပြုပေးပါ။

## Formatting Elements

စာလုံးတွေရဲ့ ဖော်ပြပုံအသွင်အပြင်ဟာ တစ်ကယ်တမ်းတော့ HTML ရဲ့အလုပ် မဟုတ်ပါဘူး။ HTML ရဲ့ တာဝန်က Content Structure တည်ဆောက်ဖိုပါပဲ။ ဒါပေမယ့် HTML မှာ လိုအပ်ရင် အသုံးပြုနိုင်ဖို့ အတွက် စာလုံးအသွင်အပြင်တွေ Format ပြောင်းပေးနိုင်တဲ့ Element တွေရှုပါတယ်။ မှတ်သားသင့်တဲ့ Element စာရင်းကို ထည့်ပြောချင်ပါတယ်။

**<b>**, **<strong>** - စာလုံးတွေကို Bold လုပ်ပြီးဖော်ပြစေချင်ရင် **<b>** သို့မဟုတ် **<strong>** Element ကို အသုံးပြုနိုင်ပါတယ်။

**<i>**, **<em>** - စာလုံးတွေကို Italic ပုံစံ စာလုံးစောင်းနဲ့ ဖော်ပြစေချင်ရင် **<i>** သို့မဟုတ် **<em>** Element ကို အသုံးပြုနိုင်ပါတယ်။

**<s>**, **<del>** - စာလုံးတွေကို ကန့်လတ်ဖြတ်လိုင်းနဲ့ ဖျက်ပြီးပြချင်တယ်ဆိုရင် (ဥပမာ - ဖူက်ဆေး ဆောင့်စေ) **<s>** သို့မဟုတ် **<del>** ကိုသုံးနိုင်ပါတယ်။ Underline တာဖို့အတွက် **<u>** Element ရှုပေမယ့် မသုံးသင့်တဲ့ Element အနေနဲ့ ပယ်ထားကြပါတယ်။

**<code>**, **<pre>** - ကုဒ်နမူနာတွေကို HTML ထဲမှာ ထည့်ရေးပြချင်တဲ့အခါ တစ်ကြောင်းထဲဆိုရင် **<code>** Element ကို သုံးနိုင်ပါတယ်။ ကုဒ်တွေတစ်ကြောင်းထက်ပိုပြီး များတယ်ဆိုရင် **<pre>** Element ကိုသုံးပြီး ပြနိုင်ပါတယ်။ ကုဒ်တွေပြတဲ့အခါမှာ သုံးရတဲ့ Monospace ဖွန့်တွေသုံးပြီး ပြပေးပါတယ်။ **<pre>** မှာ နောက်ထပ်ထူးခြားချက် ရှုပါသေးတယ်။ HTML ရဲ့တစ်ခြားနေရာမှာ Space တွေ Tab တွေ Enter တွေ ထည့်ချင်သလောက်ထည့် အလုပ်မလုပ်ပါဘူး။ **<pre>** ထဲမှာရေးထားတဲ့ Content မှာ တော့ Space တွေ Tab တွေ Enter တွေပါရင်၊ ပါတဲ့အတိုင်း အကုန်အလုပ်လုပ်ပေးပါတယ်။

**<sup>**, **<sub>** - Superscript နဲ့ Subscript တို့အတွက်ပါ။ 4<sup>th</sup> ရဲ့ အပေါ်နည်းနည်းတင်ပြတဲ့ **<th>** ကို Superscript လိုပေါ်ပြီး HTML မှာ **<sup>** Element ကိုသုံးနိုင်ပါတယ်။ H<sub>2</sub>O ရဲ့ အောက်နည်းနည်းချုပြတဲ့ 2 ကို Subscript လိုပေါ်ပါတယ်။ HTML မှာ **<sub>** Element ကို သုံးနိုင်ပါတယ်။

**<blockquote>** - ဆောင်ပုဒ်တွေ၊ ဆိုရိုးစကားတွေ၊ အကိုးအကားတွေကို Quote လုပ်ပြီးပြချင်တယ်ဆိုရင် <blockquote> Element ကိုသုံးနိုင်ပါတယ်။ ဘယ်လိုပုံစံဖော်ပြသလဲဆိုတာ စာနဲ့ပြောရင် ရှုပ်ပါတယ်။ Codepen မှာသာ လက်တွေ့ထည့်ရေးပြီး စမ်းကြည့်လိုက်ပါ။

**<address>, <time>** - လိပ်စာတွေကို <address> Element နဲ့ဖော်ပြနိုင်ပြီး ရက်စွဲနဲ့ အချိန်တွေကိုတော့ <time> Element နဲ့ပြနိုင်ပါတယ်။ ဒါတွေက ဒီနေရာမှာ ထည့်ပြောပေမယ့် အသွင်အပြင် ပြောင်းပေးတဲ့ Formatting Element တွေတော့ မဟုတ်ပါဘူး။ အသုံးဝင်တဲ့ Element တွေအနေနဲ့ ကျိုနေလို့ တစ်ခါတည်း ထည့်ပြောလိုက်တာပါ။

**<!-- Comment -->** - ဒါကတော့ Comment Element ဖြစ်ပါတယ်။ HTML ထဲမှာ ကိုယ့်ဘာသာရေးမှတ်ချင်တာတွေရှိရင် ဒီ Element နဲ့ ရေးမှတ်နိုင်ပါတယ်။ အလုပ်လုပ်တဲ့အခါ ဒီ Comment တွေကို ထည့်သွင်း အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။

## HTML Symbols

HTML မှာ Copyright တို့ Trademark တို့လို့ သကော်တတွေအပါအဝင် အခြားသကော်တတွေကိုလည်း ထည့်သွင်းအသုံးပြုလို့ ရပါတယ်။ ဥပမာ - ဒီလိုပါ။

### HTML

```
<p>Copyright © Fairway Technology ™</p>
```

နမူနာမှာ Symbol နှစ်ခုပါပါတယ်။ © &copy; နဲ့ ™ &trade; တို့ပါ။ Symbol တွေကိုရေးတဲ့အခါ Ampersand (&) နဲ့ စံပြီး Semi-colon (;) နဲ့ အဆုံးသတ် ပေးရလေ့ရှိပါတယ်။ ရလဒ်က ဒီလိုဖြစ်မှာပါ။

```
Copyright © Fairway Technology ™
```

တစ်ခြား အလားတူ အသုံးဝင်နိုင်တဲ့ Symbol တွေကို ထည့်သွင်းဖော်ပြပေးလိုက်ပါတယ်။

- &copy; - ©
- &trade; - ™
- &reg; - ®
- &euro; - €
- &larr; - ←
- &uarr; - ↑
- &rarr; - →
- &darr; - ↓
- &crarr; - ↲
- &lArr; - ⇲
- &rArr; - ⇒
- &laquo; - «
- &raquo; - »
- &nnbsp; - [space]

&nnbsp; က နေရာလွတ်တစ်လုံးစာ Space ကိုထည့်သွင်းပေးတာပါ။ အရင်ကတော့ သက်တော့ ထည့်ချင်ရင် ဒီနည်းကိုပဲ သုံးရပါတယ်။ အခုတော့ ယူနိကုဒ်ရဲ့ အကူအညီနဲ့ Emoji တွေကို ပုံစံစုံနဲ့ ထည့်လိုရလာပါပြီ။ ဒီနည်းအတိုင်း နေရာတိုင်းမှာ ရေးထည့်စရာ မလိုတော့ပါဘူး။ ကိုယ်ထည့်ချင်တဲ့ သက်တံ့ကို ကိုဘုတ် ဆောဖို့တွေနဲ့အတူတွဲပါတဲ့ Emoji Browser ကနေ ရွေးပြီးထည့်လိုက်ယုံပါပဲ။ ဒီလိုထည့်လိုရနေပေမယ့် HTML Symbol တွေကိုတော့ ကင်းလိုတော့ မရနိုင်သေးပါဘူး။ သူ့နေရာနဲ့သူ အသုံးဝင်နေဆဲပါပဲ။

အခုဆိုရင် HTML အကြောင်း တော်တော်လေးစုံသလောက်ဖြစ်သွားပါပြီ။ HTML ရဲ့ အဓိကတာဝန်ကို မမေ့စေချင်ပါဘူး။ ကွန်ပျူးတာစနစ်တွေက နားလည်တဲ့ Content Structure ကိုတည်ဆောက်ခြင်း ဖြစ်ပါတယ်။ အဲဒါ Content တွေရဲ့ ဖော်ပြပုံအသွင်အပြင်သတ်မှတ်ခြင်းဟာ HTML ရဲ့တာဝန်မဟုတ်ပါဘူး။ နောက်တစ်ခန်းမှာဆက်လက်ဖော်ပြမယ့် CSS ရဲ့ တာဝန်ပဲဖြစ်ပါတယ်။

## အခန်း (J) - CSS

HTML ဟာ Markup Language ဖြစ်ပြီး CSS ကတေသာ Style Language ဖြစ်ပါတယ်။ HTML ကိုအသုံးပြု စုစဉ်းထားတဲ့ Content တွေရဲ့ အချက်အစား၊ အရောင်၊ အကွာအဝေး စသဖြင့် ဖော်ပြပုံအသွင်အပြင်ကို CSS နဲ့ သတ်မှတ်ပေးရမှာ ဖြစ်ပါတယ်။ CSS ရဲ့ အဓိပ္ပာယ်အရှည်က Cascading StyleSheet ဖြစ်ပါတယ်။ Cascading ဆိုတာ တစ်ခုထက်ပိုတဲ့ Style သတ်မှတ်ချက်တွေကို ရောစပ် အသုံးပြုနိုင်တဲ့သဘောလို့ အလွယ်မှတ်နိုင်ပါတယ်။ ဒီလိုရောစပ်သုံးတဲ့အခါ အလုပ်လုပ်ပုံကို ခဏနေတော့ ဆက်လေ့လာကြပါမယ်။

အရင်တုံးကတော့ Web Document တွေတည်ဆောက်ဖို့ Markup Language တွေ အမျိုးမျိုးရှိလာမယ်လို့ တိတွင်သူတွေက မျှော်မှန်းခဲ့ကြပုံ ရပါတယ်။ XHTML လိုနည်းပညာတွေ ရှိခဲ့ဖူးပေမယ့်လည်း အခုတော့ HTML တစ်မျိုးတည်းကိုသာ သုံးကြပါတော့တယ်။ အလားတူပဲ Style Language တွေလည်း အမျိုးမျိုးရှိလိမ့်မယ်လို့ မျှော်မှန်းထားခဲ့ကြမယ် ထင်ပါတယ်။ XSLT လိုနည်းပညာတွေ ရှိခဲ့ဖူးပေမယ့် အခုတော့ CSS တစ်မျိုးတည်းကိုသာ သုံးကြပါတော့တယ်။ Script Language တွေလည်း အမျိုးမျိုးရှိလိမ့်မယ်လို့ မျှော်မှန်းထားကြပါလိမ့်မယ်။ VBScript လိုနည်းပညာတွေ ရှိခဲ့ဖူးပေမယ့် အခုတော့ JavaScript တစ်မျိုးတည်းကိုသာ သုံးကြပါတော့တယ်။ ဒါကြောင့် HTML, CSS, JavaScript တို့ဟာ မူကဲ့အမျိုးမျိုးရှိပြီး အမျိုးမျိုးနဲ့ အပြန်အလှန်တွဲဖက်သုံးလို့ရစေမယ့် ပုံစံမျိုးတွေနဲ့ ရည်ရွယ်ဖန်တီးခဲ့ကြပေမယ့် လက်တွေ့မှာ မူကဲ့တွေမရှိကြတော့ ပဲ Web Document တည်ဆောက်ဖို့ဆိုရင် HTML, CSS နဲ့ JavaScript ကိုသာ အသုံးပြုရတယ် ဆိုတဲ့ အခြေအနေကို ရောက်ရှိနေခြင်းပဲ ဖြစ်ပါတယ်။

မူကဲ့တွေနှစ်မျိုးသုံးမျိုး ရှိမယ့်အစား၊ Pre-processor တို့ Superset တို့လို့ အသုံးအနှံးတွေနဲ့ တစ်ဆင့်ခံနည်းပညာတွေ ထွက်ပေါ်လာခဲ့ပါတယ်။ CSS မှာဆိုရင် LESS တို့ SASS တို့လို့ Pre-processor နည်းပညာတွေ ရှိပါတယ်။ မူလ CSS မှာ မပါတဲ့ ရေးနည်းရေးဟန်တွေ၊ လုပ်ဆောင်ချက်တွေကို ပေါင်းထည့်ပေးထားတာပါ။ ဒါပေမယ့် ရေးပြီးရင် အဲဒီ LESS တို့ SASS တို့လို့ ကုဒ်တွေကို CSS ကုဒ် ဖြစ်အောင် ပြန်ပြောင်းပြီးမှ

ပဲ သုံးလို့ရပါတယ်။ JavaScript မှာဆိုရင်လည်း CoffeeScript တို့ TypeScript တို့လို့ တစ်ဆင့်ခံ နည်းပညာတွေ ရှိနေပါတယ်။ ဒီနည်းပညာတွေလည်းပဲ ရေးပြီးရင် JavaScript ကုဒ်ဖြစ်အောင် ပြန်ပြောင်းပြီးမှ ပဲ သုံးကြော်ရပါတယ်။ ဒါက ဗဟိုသုတေသနနဲ့ ထည့်မှတ်ဖို့ပါ။

## CSS Syntax

CSS ကုဒ်တွေရဲ့ ဖွဲ့စည်းပုံကို အရင်ဆုံးစကြည့်ကြပါမယ်။ ဒီလိုပါ။

Selector



**div**

**background:** yellow;

**color:** brown;

}



Property

Value



Rules

နမူနာအရ <div> Element တွေအားလုံးကို နောက်ခံ အဝါရောင်နဲ့ စာလုံးနှိပ်ရောင် သုံးပြီးဖော်ပြရမယ် လို့ သတ်မှတ်လိုက်တာပါ။ Selector ဖြစ်တဲ့ div ရဲ့ နောက်မှာ တွန်းကွင်းအဖွင့်အဝိတ်နဲ့ သတ်မှတ်လိုတဲ့ Rule တွေကို တန်းစီပြီး ရေးပေးရတာပါ။ Rule တစ်ခုနဲ့ တစ်ခုကို Semi-colon (;) နဲ့ ပိုင်းခြားပေးရပါ တယ်။ တစ်ကြောင်းထဲပဲရောရေးရေး၊ နမူနာမှာလို့ Rule တစ်ခုကို တစ်ကြောင်းနှံးနဲ့ပဲ ခဲ့ရေးရေး ကြိုက် သလိုရေးလို့ရပါတယ်။ အလုပ်လုပ်ပါတယ်။ ဒါပေမယ့် ခဲ့ရေးမှ ဖတ်ရှုပြင်ဆင်ရ လွယ်ကူအဆင်ပြောပါ။ Rule တစ်ခုချင်းစီမှာ Property နဲ့ Value တွေပါဝင်ပြီး Full-colon (:) နဲ့ ပိုင်းခြားပေးရပါတယ်။ ရေးနည်းက ဒီတစ်နည်းထဲပါပဲ။ နှစ်နည်းမရှိပါဘူး။ ဒီရေးနည်းမှာပါဝင်တဲ့ Selector တွေ Property တွေ Value တွေကိုသာ ဆက်လက်လေ့လာကြရမှာပါ။

## HTML & CSS

HTML Document တစ်ခုအတွက် CSS Style တွေကို ပုံစံ (R) မျိုးနဲ့ ရေးသား ထည့်သွင်းလို့ရပါတယ်။ ပထမတစ်နည်းကတော့ CSS ကုဒ်တွေကို သီးခြားဖိုင်တစ်ခုနဲ့ ရေးသားပြီး HTML Document ထဲကနေ လှမ်းချိတ်တဲ့နည်းပါ။ External CSS လို့ခေါ်ကြပါတယ်။ CSS က သပ်သပ်၊ HTML က သပ်သပ်ရေးပြီး တော့ ချိတ်ပေးလိုက်တာပါ။

### CSS - style.css

```
body {
    background: cyan;
    color: brown;
}
```

### HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    ...
</body>
</html>
```

နမူနာကိုလေ့လာကြည့်လိုက်ရင် <link> Element ကိုသုံးပြီး CSS ကုဒ်ဖိုင်ကို ချိတ်ဆက်ထားခြင်းဖြစ်ပါတယ်။ ဒီလိုချိတ်ဆက်ပေးလိုက်ရင် လက်ရှိ Document ထဲက Element တွေကို ဖော်ပြတဲ့အခါ ချိတ်ဆက်ပေးထားတဲ့ CSS ဖိုင်မှာ သတ်မှတ်ထားတဲ့ အသွင်အပြင်သတ်မှတ်ချက်များအတိုင်း ဖော်ပြပေးသွားမှာပါ။

<link> အတွက် rel Attribute နဲ့ href Attribute တို့ ပါဝင်တာကို သတိပြုပါ။ နှစ်ခုလုံး မဖြစ်မနေပါဝင်ရပါတယ်။ rel ရဲတန်ဖိုးက ပုံသဏ္ဌာန်၊ stylesheet ဖြစ်ရပါတယ်။ <link> ကိုသုံးပြီး တစ်ခြား Resource အမျိုးအစားတွေကို ချိတ်လို့ ရပါသေးတယ်။ Favicon ခေါ် Icon တွေဘာတွေလည်း ချိတ်လို့ တာဖြစ်လို့ CSS Style တွေကို ချိတ်မှန်း ကဲပြားအောင် rel မှာ stylesheet လို့ သတ်မှတ်ပေးရတာပါ။ href ကတော့ CSS ဖိုင်တည်ရှိရာ URL/Path ဖြစ်ပါတယ်။ လေ့လာစမှာ အများအားဖြင့် CSS တွေ

ရေးထားပေမယ့် အလုပ်လုပ်ဘူးဆိုရင် ဒီနေရာမှာပေးတဲ့ URL/Path လွှဲနေ၊ မှားနေကြတာ များပါတယ်။ သေချာမှန်အောင် ပေးရပါတယ်။ <link> Element ကို ကြိုက်တဲ့နေရာမှာ ရေးလိုရပေမယ့် <head> အတွင်းမှာ ရေးကြလေ့ရှိပါတယ်။ အဲဒါ Document အတွက် အသွင်အပြင်သတ်မှတ်ချက် တွေကို ကြိုကြညာလိုက်တဲ့ သဘောပါ။

နောက်ထပ်ရေးနည်းကိုတော့ Internal CSS လို ခေါ်ပါတယ်။ <head> Section အတွင်းမှာပဲ <style> Element ကိုသုံးပြီး CSS ကုဒ်တွေ ရေးနိုင်ပါတယ်။ ဒါလိုပါ -

#### HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        body {
            background: cyan;
            color: brown;
        }
    </style>
</head>
<body>
    ...
</body>
</html>
```

ဒီတစ်ခါ <style> ကိုသုံးပြီး CSS ကုဒ်တွေ တစ်ခါတည်း ရေးပေးလိုက်တာပါ။ အရင်ကတော့ <style> အတွက် type=text/css ဆိုတဲ့ Attribute တစ်ခုသတ်မှတ် ပေးရပါတယ်။ အပေါ်မှာပြောခဲ့သလို Style Language နှစ်မျိုးသုံးမျိုး ရှိခဲ့ရင် သုံးလိုတဲ့ Language အမျိုးအစားကို ပြောပေးရတဲ့ သဘောပါ။ အခုတော့ CSS တစ်ခုပဲရှိလို ထည့်ပေးစရာမလိုတော့ပါဘူး။

နောက်ဆုံးတစ်နည်းကိုတော့ Inline Style လို ခေါ်ပါတယ်။ HTML Element နဲ့အတူ CSS Style တွေကို တွဲရေးတဲ့နည်းပါ။ ဒါလိုပါ -

**HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body style="background: cyan; color: brown;">
    ...
</body>
</html>
```

style Attribute ကိုသုံးရပါတယ်။ ရေးနည်းနည်းထူးခြားသွားပါတယ်။ Selector တွေ တွန်းကွင်းတွေ မပါတော့ပဲ၊ Rule တွေချည်းပဲ တန်းစီပြီး ချရေးပေးရတာပါ။

အရင်ကဆိုရင် External CSS ကိုသာလျှင် အသုံးပြုသင့်ကြောင်း။ Markup နဲ့ Style ကို ခွဲခြားထားတဲ့ အတွက် ပြုပြင်ထိမ်းသိမ်းရ ပိုမိုလွယ်ကူးကြောင်း၊ ခွဲရေးထားတဲ့ External CSS ဖိုင်ကို လိုတဲ့ Document ကနေ ချိတ်သုံးလို့ရလို ထပ်ခါထပ်ခါ ပြန်ရေးစရာမလိုတော့ကြောင်း စသဖြင့် External CSS ကိုသာ သုံးကြဖို့နဲ့ တစ်ခြားနည်းတွေ မသုံးဖို့ ပြောရပါတယ်။ အခုတော့ ဆောင်တွေပြောင်းပြီး အကုန်လုံး သူမော်ရာနဲ့သူ သုံးလာကြလို့ ရေးနည်းအားလုံးကို မှတ်သားထားဖို့ လိုအပ်ပါတယ်။

ပြီးတော့ တစ်ကြိမ်မှာ ရေးနည်း (၁) မျိုးပဲသုံးရမယ်ဆိုတဲ့ ကန်သတ်ချက်မျိုး မရှိပါဘူး။ <link> နဲ့ လိုတဲ့ ဖိုင် နှစ်ခုသုံးခုကို ချိတ်သုံးလို့ရသလို <style> တွေ Inline တွေနဲ့လည်း ရောသုံးလို့ရပါတယ်။ အဲဒီလို ရေးနည်းအမျိုးမျိုး ရောသုံးလိုပဲဖြစ်ဖြစ်၊ တစ်မျိုးတည်း သုံးပေမယ့် ရေးထားတဲ့ကူးဖြစ်တွေ တူပြီးပြန်ထပ်လာရင်ပဲ ဖြစ်ဖြစ် CSS ရဲ့ အလုပ်လုပ်သွားတဲ့ ပုံစံကိုလည်း ထည့်သွင်းမှတ်သားရပါမယ်။ အဲဒါကို Cascading Order လိုပေါ်ပါတယ်။

## Cascading Order

Cascading Order မှာ အကြမ်းဖျင်းအားဖြင့် ချိတ်ဆက်ထည့်သွင်းတဲ့နည်း တူရင် နောက်မှရေးတဲ့ကုဒ်ကို အတည်ယူတယ်လို့ မှတ်ထားပေးပါ။ ဒီလို့ နှစ်ခုချိတ်ထားတယ် ဆိုပါစို့ -

### HTML

```
<link rel="stylesheet" href="a.css">
<link rel="stylesheet" href="b.css">
```

a.css နဲ့ b.css ထဲမှာ ရေးထားတဲ့ကုဒ်တူတာပါလာရင် b.css ကို အတည်ယူသွားမှာပါ။ သူက နောက်မှ ချိတ်ပြီး ထည့်ထားတာမို့လို့ပါ။ အမြဲတမ်းတော့မဟုတ်ပါဘူး၊ အများအားဖြင့် ဒီသဘောနဲ့ အလုပ်လုပ်သွားတဲ့ အစီအစဉ်က ဒီလိုပါ -

External, Internal စသဖြင့် နည်းတွေ ရောသုံးတဲ့အခါ အလုပ်လုပ်သွားတဲ့ အစီအစဉ်က ဒီလိုပါ -

1. Browser Default
2. External Style
3. Internal Style
4. Inline Style

Browser Default ဆိုတာ ကိုယ်ရေးပေးထားတဲ့ CSS မဟုတ်ဘဲ Web Browser တွေက အလိုအလျောက် သတ်မှတ်ပေးတဲ့ Style တွေကိုပြောတာပါ။ ဥပမာ <h1> ဆိုရင် စာလုံးကြီးကြီးပြတယ်။ <p> တွေတစ်ခု နဲ့တစ်ခု နည်းနည်းခွာပြီးပြတယ်ဆိုတာ Browser Default Style တွေကြောင့်ပါ။ အဲဒါတွေကိုအရင် အလုပ်လုပ်ပါတယ်။ ပြီးတော့မှ External Style တွေကိုအလုပ်လုပ်ပါတယ်။ ပြီးတော့မှ Internal Style တွေကို အလုပ်လုပ်ပြီး နောက်ဆုံးမှ Inline Style ကိုအလုပ်လုပ်ပါတယ်။ ကုဒ်တွေ ထပ်နေ၊ တူနေရင် နောက်မှ အလုပ်လုပ်တဲ့ကုဒ်က အတည်ဖြစ်မှာ ဖြစ်ပါတယ်။

တူတယ်ဆိုတဲ့နေရာမှာ ဒီလိုပုံစံမျိုးတွေ လာနိုင်ပါတယ်။

**CSS - a.css**

```
body {
    background: cyan;
    color: brown;
}
```

**CSS - b.css**

```
body {
    background: yellow;
}
```

a.css ထဲမှာ ရေးထားတဲ့ကုဒ်နဲ့ b.css ထဲမှာ ရေးထားတဲ့ကုဒ် ဆင်တူနေပါပြီ။ ဒါကြောင့် နောက်မှရေးတဲ့ b.css ထဲကကုဒ်ကို အတည်ယူသွားမှာပါ။ ဒါပေမယ့် Rule တွေကိုကြည့်လိုက်ရင် background တစ်ခဲပဲ တူတာဖြစ်ပြီး color က မတူပါဘူး။ ဒါကြောင့် နောက်ဆုံးရလဒ်က ဒီလိုဖြစ်မှာပါ။

**CSS**

```
body {
    background: yellow;
    color: brown;
}
```

background က b.css ထဲမှာ ရေးထားတဲ့အတိုင်း yellow ဖြစ်သွားပေမယ့် color ကတော့ a.css မှာ ရေးခဲ့တဲ့အတိုင်း brown ဖြစ်နေမှာဖြစ်ပါတယ်။ CSS လေ့လာတဲ့အခါ ဒါတွေနားလည်ဖို့ အရေးကြီးပါတယ်။ မဟုတ်ရင် ရေးတဲ့အတိုင်းလည်း အလုပ်မလုပ်ဘူး၊ ဘာတွေမှန်းမသိဘူး ဆိုပြီးတော့ စိတ်တွေ ညစ်သွားတတ်ပါတယ်။ CSS ဟာ လွယ်မယောင်ယောင်နဲ့ နည်းနည်းခက်ပါတယ်။

နောက်မှရေးတဲ့ကုဒ်ကို အတည်ယူတယ်ဆိုတာနဲ့ ပက်သက်ရင် ခြင်းချက်တော့ ရှိပါတယ်။ Selector ချင်းတူနေရင် ပိုတိကျတာကို အရင်ရေးရေး နောက်မှရေးရေး အတည်ယူပါတယ်။ ဒီလိုပါ -

**CSS**

```
body div {
    background: cyan;
}

div {
    background: yellow;
}
```

div ဆိုတာလည်း <div> Element တွေကိုပြောတာပါပဲ။ body div ဆိုတာက <body> ထဲမှာရှိတဲ့ <div> လို့ပြောတာဖြစ်လို့ <div> တွေကို ပြောတာပါပဲ။ တစ်ကယ်တစ်းက အတူတူပါပဲ။ ဒါပေမယ့် body div ဆိုတဲ့ Selector က div ဆိုတဲ့ Selector ထက် ပိုတိကျလို့ သူကိုအရင်ရေးထားပေမယ့် လည်း သူကိုအတည်ယူပြီး အလုပ်လုပ်ပေးသွားမှာပါ။ နမူနာကုဒ်အရ <div> ရဲ့ background ဟာ cyan ပဲဖြစ်မှာ ဖြစ်ပါတယ်။

**CSS Selectors**

CSS မှာ Selector တွေ အမျိုးမျိုးရှိပါတယ်။ အခြေခံ Selector (၄) မျိုးကနေ စောင့်ကြပါမယ်။

- Element/Type Selector
- ID Selector
- Class Selector
- Attribute Selector

Element Selector (သို့မဟုတ်) Type Selector ဆိုတာ လိုရင်းကတော့ HTML Element ရဲ့ အမည် အတိုင်း Select လုပ်တာပါ။ body ဆိုရင် <body> Element တွေကို Select လုပ်တာပါ။ div ဆိုရင် <div> Element တွေကို Select လုပ်တာပါ။ p ဆိုရင် <p> Element တွေကို Select လုပ်တာပါ။ ဒါ ကြောင့် အရှုံးရှင်းဆုံးနဲ့ အခြေခံအကျခုံး Selector လို့ ဆိုနိုင်ပါတယ်။

ID Selector ကိုတော့ ရှုံးကနေ # သက်တလေးခံပြီး ရေးပေးရပါတယ်။ p#note ဆိုရင် <p> Element တွေထဲက id Attribute မှာ note လို့ပေးထားတဲ့ Element ကိုရွေးယူလိုက်တာပါ။ #note ဆိုရင်တော့ <p> တွေ <div> တွေမပြောတော့ဘဲ id Attribute မှာ note လို့ပေးထားတဲ့ Element ကို Select လုပ်

ယူတာပါ။ နှစ်မျိုးရေးလိုဂုဏ်သော ဖြစ်ပါတယ်။ ရွှေ့ကနေ Element ထည့်ရေးလိုရသလို၊ မထည့်ဘဲလည်း ရေးလိုရပါတယ်။

Class Selector ကိုတော့ ရွှေ့ကနေ Dot (.) သက်တလေးခံပြီး ရေးပေးရပါတယ်။ p.note ဆိုရင် <p> Element တွေထဲက class Attribute မှာ note လိုပေးထားတဲ့ Element တွေကို ရွေးယူတာပါ။ .note ဆိုရင်ရင်တော့ <p> တွေ <div> တွေ မပြောတော့ဘဲ class Attribute မှာ note လိုပေးထားသမျှ Element အားလုံးကို Select လုပ်ယူလိုက်တာပါ။

ဖြည့်စွက်သတိပြုပေးပါ။ id ဆိုတာ ပြန်မထပ်ရပါဘူး။ id တူနေတဲ့ Element တွေမရှိသင့်ပါဘူး။ တစ်ခုထဲ သီးသန်ဖြစ်သင့်ပါတယ်။ ပြန်ထပ်လို့ id အတူတူ နှစ်ခုသုံးခုပေးထားလို့ Error တက်မှာ မဟုတ်ပေမယ့်မပေးသင့်ပါဘူး။ သူ့ရည်ရွယ်ချက်ကိုက Element အတွက် Unique ID ပေးဖို့အတွက်မို့လိုပါ။ class ကတော့ အမျိုးအစားတူတဲ့ Element တိုင်းကို class အတူတူပေးလို့ရပါတယ်။ သူ့ရည်ရွယ်ချက်ကိုက အမျိုးတူရာ Element တွေကို အတူတူစုပေးဖို့ပါ။ နောက်ထပ်ဖြည့်စွက် မှတ်သားရမှာကတော့ class ရဲ့ Value ဟာ တစ်ခုထက်ပိုပေးလို့ရခြင်း ဖြစ်ပါတယ်။ ဒီလိုပါ -

#### HTML

```
<p class="alert note active"> ... </p>
```

နမူနာအရ <p> Element မှာ alert, note နဲ့ active ဆိုတဲ့ class တန်ဖိုး (၃) ခုတိရှိနေတာပါ။ အဲဒါဒီလိုပေးလို့ရပါတယ်။

Attribute Selector ကတော့ Element ရဲ့ Attribute ကိုကြည့်ပြီး Select လုပ်တာပါ။ img[alt] ဆိုရင် <img> Element တွေထဲကမှ alt Attribute ရှိတဲ့ Element တွေကိုချည်းပဲ ရွေးလိုက်တာပါ။ alt Attribute မရှိရင် ထည့်မရွေးပါဘူး။ input[type=submit] ဆိုရင်တော့ <input> Element တွေထဲကမှ type=submit တွေကိုချည်းပဲ ရွေးလိုက်တာပါ။ ဒါကြောင့် Attribute Selector နဲ့ Select လုပ်တဲ့အခါ Attribute ချည်းပဲပေးပြီး Select လုပ်လိုရသလို Attribute=Value လိုအပြည့်အစုံပေးပြီးတော့ လည်း Select လုပ်လိုရမှာ ဖြစ်ပါတယ်။ ဒီ (၄) မျိုးလုံးကို အခုလို လက်တွေ စမ်းကြည့်လိုက်ပါ။

**HTML**

```
<p>Browser List</p>
<p class="browser" id="popular">Google Chrome</p>
<p class="browser">Mozilla Firefox</p>
<input type="text">
<input type="submit">
```

နမူနာအရ <p> Element (၂) ခုပါဝင်ပါတယ်။ တစ်ခုနဲ့တစ်ခု class တွေ id တွေတော့ မတူကြပါဘူး။ <input> (၂) ခုပါဝင်ပါတယ်။ type တွေ မတူကြပါဘူး။ ဒီ Element တွေအတွက် Style သတ်မှတ်ချက် တွေကို အခုလိုရေးပြီး စမ်းကြည့်ပါ။

**CSS**

```
p {
    padding: 10px;
}

.browser {
    background: cyan;
}

#popular {
    font-weight: bold;
}

input[type=text] {
    width: 400px;
}
```

နမူနာအရ p Selector ကိုသုံးပြီးရေးထားတဲ့ padding ဟာ <p> Element များ အားလုံးပေါ်မှာ သက်ရောက်နေမှာပါ။ ဘာဖြစ်သွားတာလဲ မြင်သိစေဖို့အတွက် တန်ဖိုး 10px ကို နစ်သက်ရာတန်ဖိုးနဲ့ အစားထိုးပြီး စမ်းကြည့်နိုင်ပါတယ်။ စမ်းကြည့်သင့်ပါတယ်။ padding ရဲ့ သဘောသဘာဝကို ခဏာနေတော့မှ ထပ်ပြောပါမယ်။

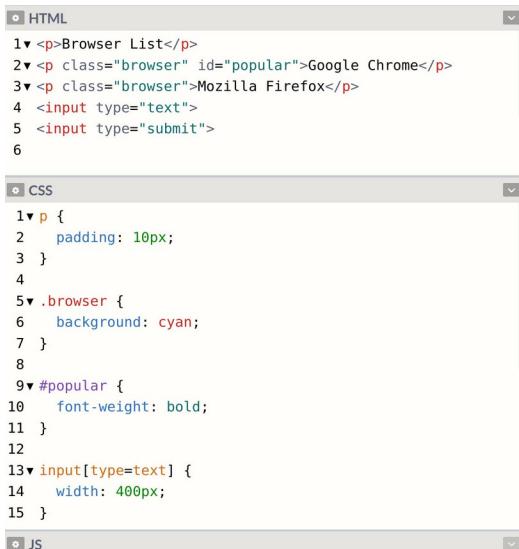
.browser Selector ကိုသုံးပြီးတော့ background ထည့်ထားပါတယ်။ ဒါကြောင့် class မှာ browser လို့ သတ်မှတ်ထားတဲ့ Element တွေမှာသာ ဒီ background အသက်ဝင်တယ်ဆိုတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ ကျေနဲ့ Element တွေနဲ့ မဆိုင်ပါဘူး။ အလားတူပဲ #popular Selector ကိုသုံး

ပြီးတော့ စာလုံးတွေကို Bold လုပ်နိုင်းထားလို့ id=popular Element ထဲမှာရှိတဲ့စာကိုပဲ ရွှေးပြီးတော့ Bold လုပ်ပေးသွားမှာဖြစ်ပါတယ်။

နောက်ဆုံးတစ်ခုအနေနဲ့ input [type=text] Selector ကိုသုံးပြီး width သတ်မှတ်ထားပါတယ်။ <input> နှစ်ခုပါဝင်ပေမယ့် ဒီ width တန်ဖိုးဟာ type=text Element ပေါ်မှာပဲ သက်ရောက်တယ် ဆိုတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

စမ်းတဲ့အခါ အပေါ်မှာပြောထားတဲ့ နည်း (၃) နည်းထဲက External CSS အနေနဲ့စမ်းလိုရသလို Internal CSS အနေနဲ့လည်း စမ်းလိုရပါတယ်။ ဒါပေမယ့် လောလောဆယ် လွယ်သွားအောင် Codepen မှာပဲ စမ်းကြည့်လိုက်ပါ။ ဒီလိုပါ -

- <https://codepen.io/pen>



```

• HTML
1▼ <p>Browser List</p>
2▼ <p class="browser" id="popular">Google Chrome</p>
3▼ <p class="browser">Mozilla Firefox</p>
4 <input type="text">
5 <input type="submit">
6

• CSS
1▼ p {
2   padding: 10px;
3 }
4
5▼ .browser {
6   background: cyan;
7 }
8
9▼ #popular {
10   font-weight: bold;
11 }
12
13▼ input[type=text] {
14   width: 400px;
15 }

• JS

```

Selector တွေအကြောင်း ပြောနေတာဆိုပေမယ့် လက်စနဲ့ နမူနာပေးထားတဲ့ Property တွေကိုလည်း တစ်ခါတည်း ထည့်မှတ်၊ ထည့်စမ်းပေးပါ။ နားလည်ရ လွယ်ပါတယ်။ Property ကိုကြည့်လိုက်ယုံနဲ့ ဘာကို ဆိုလိုတာလဲဆိုတာ အမိပိုယ် ပေါ်လွင်ပါတယ်။ တန်ဖိုးတွေကို ကိုယ့်စိတ်ကူးလေးနဲ့ကိုယ် မှန်းပြီးပြင်စမ်းကြည့်နိုင်ပါတယ်။ လက်တွေ့စမ်းကြည့်တာထက် ပိုကောင်းတဲ့လေ့လာနည်း မရှိပါဘူး။

Selector တွကို Comma ခံပြီး Group လုပ်လိုလည်း ရပါတယ်။ ဒီလိုပါ -

### CSS

```
h1, h2, h3 {
    color: brown;
}
```

ဒါဟာ Selector သုံးခုကို တစ်ခါတည်း Comma ခံပြီး တွဲရေးလိုက်တာပါ။ ဒါကြောင့် သတ်မှတ်လိုက်တဲ့ Property ဟာ <h1> <h2> <h3> အားလုံးအပေါ်မှာ သက်ရောက်တော့မှာပါ။

ဆက်လက်လေ့လာရမှာကတော့ Selector တွကို ဖွဲ့စည်းပုံပေါ် မူတည်ပြီး ရွေးယူလိုရတဲ့ နည်းတွေဖြစ်ပါတယ်။ CSS မှာ ဒီလို Selector (င) မျိုးရှိပေမယ့်၊ (ဂ) မျိုးကိုရွေးပြီးတော့ ဖော်ပြချင်ပါတယ်။

- Descendant Selector
- Child Selector

Descendant Selector ဆိုတာ Element တစ်ခုအတွင်းထဲမှာ ရှိတဲ့ Element တွကို Select လုပ်တဲ့ နည်းပါ။ ဥပမာ - ul li ဆိုရင် <ul> အတွင်းထဲက <li> တွကို Select လုပ်ခြင်းဖြစ်ပါတယ်။ တစ်ခြား <li> တွေမပါပါဘူး။ div p span ဆိုရင် <div> အတွင်းထဲက <p> အတွင်းထဲက <span> တွကို Select လုပ်တာပါ။ တစ်ခြား <span> တွေ မပါပါဘူး။ .alert b ဆိုရင် class မှာ alert လိုပေးထားတဲ့ Element တွေအတွင်းထဲက <b> ကို Select လုပ်တာပါ။ စသဖြင့် လိုအပ်သလိုတဲ့သုံးလို့ရပါတယ်။ အသုံးများပါတယ်။ Element ရဲ့ ဖွဲ့စည်းပုံကို သိရင် Select လုပ်လိုရနေပြီမိုလိုပါ။

Child Selector ဆိုတာလည်း အတွင်းထဲမှာရှိတဲ့ Element တွကို Select လုပ်တာပါပဲ။ ဒါပေမယ့် Direct Child ကိုပဲ Select လုပ်တာပါ။ ဥပမာ ul > li ဆိုရင် <ul> ရဲ့ Direct Child ဖြစ်တဲ့ <li> တွကိုပဲ ပြောတာပါ။ ထပ်ဆင့်အဆင့်ဆင့်ရှိနေတဲ့ <li> တွေမပါပါဘူး။ ဒီသဘောကို ပေါ်လွင်မယ့် ဥပမာလေး တစ်ခု ပေးပါမယ်။

**HTML**

```
<ul>
    <li>Servers</li>
    <li>
        Browsers
        <ol>
            <li>Chrome</li>
            <li>Firefox</li>
        </ol>
    </li>
</ul>
```

နမူနာမှာ List က နှစ်ထပ်ပါ။ <ul> List အတွင်းထဲမှာ နောက်ထပ် <ol> List တစ်ခု ရှိနေပါတယ်။

**CSS**

```
ul li {
    padding: 20px;
    border: 1px solid red;
}
```

နမူနာအရ Selector ကို ul li လိုပြောတဲ့အတွက် <ul> ရဲအတွင်းထဲက <li> အားလုံးပေါ်မှာ ဒီ Property တွေ သက်ရောက်မှာဖြစ်ပါတယ်။ <ol> ထဲက <li> တွေလည်း ပါပါတယ်။ သေချာစဉ်းစား ကြည့်ရင် သူတို့လည်းပဲ ပင်မ <ul> ရဲ အတွင်းထဲမှာ ရှိနေတာမို့လိုပါ။ ဒီလိုပါ -

```
HTML
1▼ <ul>
2▼   <li>Servers</li>
3▼   <li>
4     Browsers
5▼     <ol>
6▼       <li>Chrome</li>
7▼       <li>Firefox</li>
8     </ol>
9   </li>
10  </ul>

CSS
1▼ ul li {
2  padding: 20px;
3  border: 1px solid red;
4 }
```



ဘောင်ခတ်ပြီး ဖော်ပြပါလို `border` Property ကိုသုံးပြီး ပြောထားတာဖြစ်လို ရှိရှိသမျှ <li> အားလုံးကို ဘောင်ခတ်ပြီး ပြနေတာပါ။ နမူနာက Selector ကို ကိုယ့်ဘာသာ `ol li` လို ပြောင်းပြီး စမ်းကြည့်ပါ။ ဒါ ဆိုရင်တော့ <ol> အတွင်းထဲက <li> တွေပေါ်မှာသာ သက်ရောက်တယ် ဆိုတာကို တွေ့ရပါလိမ့်မယ်။

ဆက်လက်ပြီး Child Selector ဖြစ်တဲ့ `ul > li` နဲ့ စမ်းကြည့်ရင်တော့ ဒီလိုရလဒ်ကို ရမှာပါ။

```

HTML
1<ul>
2  <li>Servers</li>
3  <li>
4    Browsers
5    <ol>
6      <li>Chrome</li>
7      <li>Firefox</li>
8    </ol>
9  </li>
10 </ul>

CSS
1ul > li {
2  padding:20px;
3  border: 1px solid red;
4 }

JS

```



ဒီတစ်ခါတော့ <ol> ထဲက <li> တွေ မပါတော့ပါဘူး။ `ul > li` လိုပြောထားတဲ့အတွက် <ul> ရဲ့ Direct Child ဖြစ်တဲ့ <li> တွေအပေါ်မှာသာ သက်ရောက်ခြင်းဖြစ်ပါတယ်။ ထပ်ဆင့်ရှိနေတဲ့ <li> တွေ မပါတော့ပါဘူး။

## Selector Priority

Selector တွေမှာ ဦးစားပေးအဆင့်တွေ ရှိကြပါတယ်။ လိုရင်းကတော့ ပိုတိကျရင် ပိုဦးစားပေးပါတယ်။ ဥပမာ အခုလို HTML ဖွဲ့စည်းပုံရှိတယ် ဆိုကြပါစို့။

```

<div>
  <ol>
    <li>First</li>
    <li>Second</li>
  </ol>
</div>

```

<li> Element တွေကို အခုလို ပုံစံအမျိုးမျိုးနဲ့ Select လုပ်လိုရနိုင်ပါတယ်။

```
ol li {
    color: green;
}

li {
    color: red;
}

div li {
    color: blue;
}

div ol li {
    color: brown;
}
```

Selector ပုံစံ (c) မျိုးမှာ အေးလုံးက နမူနာမှာပေးထားတဲ့ ဖွဲ့စည်းပုံပါ <li> Element တွေပေါ်မှာ သက်ရောက်မှာတွေ ချည်းပါပဲ။ ဒီတော့ ဘာကို အတည်ယူပြီး ဘယ်လို အလုပ်လုပ်မှာလဲ စဉ်းစားစရာ ရှိလာပါတယ်။ Selector သာ တူမယ်ဆိုရင် နောက်မှရေးတာကို အတည်ယူသွားမှာပါ။ ဥပမာ -

```
ol li {
    color: red;
}

ol li {
    color: brown;
}
```

နမူနာမှာ နှစ်ခါရေးထားပါတယ်။ ol li ဆိုတဲ့ Selector တူကြပါတယ်။ ဒါမျိုးဆိုရင်နောက်မှရေးတဲ့ ကုဒ်ကို အတည်ယူမှာမို့လို <li> ရဲ့ color ဟာ brown ဖြစ်သွားမှာဖြစ်ပါတယ်။ သူအပေါ်က color: red ကို အလုပ်မလုပ်တော့ပါဘူး။

စောစောကနမူနာမှာတော့ သက်ရောက်မှုတူပေမယ့် Selector တွေ မတူကြပါဘူး။ li, ol li, div li, div ol li စသည်ဖြင့် အမျိုးမျိုးကဲ့ပြားနေပါတယ်။ ဒီလိုကဲ့ပြားတဲ့အခါမှာတော့ အပေါ်မှာပဲ ရေးရေး၊ အောက်မှာပဲ ရေးရေး ပိုတိကျတဲ့ Selector ကို အတည်ယူသွားမှာပါ။ နမူနာမှာ div ol li ဆိုတဲ့ Selector ဟာ အတိကျဆုံး ဖြစ်တဲ့အတွက် ကျန် Selector တွေနဲ့ ရေးထားတဲ့ color တွေ တစ်ခုမှ အလုပ်မလုပ်တော့ဘဲ div ol li နဲ့ ရေးထားတဲ့ color: brown ကိုသာ အတည်ယူပြီး အလုပ်လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။ ဒါကို Selector Priority လို့ခေါ်ပါတယ်။ CSS မှာ ဒါဟာ ခေါင်းစားချင်စရာ သဘောသဘာဝတစ်ခုဖြစ်ပါတယ်။

Selector တူရင် နောက်မှုရေးတဲ့ Selector ကို အတည်ယူပြီး၊ Selector မတူရင် ပိုတိကျတဲ့ Selector ကို အတည်ယူတယ်လို့ လိုတိရှင်း မှတ်နိုင်ပါတယ်။

Element Selector, Class Selector နဲ့ ID Selector တို့မှာ ID Selector က Priority အမြင်ဆုံး၊ ဦးစားပေးအလုပ်လုပ်မယ့် Selector ဖြစ်ပါတယ်။ Element Selector ကတော့ Priority အနိမ့်ဆုံးပါ။ Class Selector ကတော့ Element Selector ထက် Priority မြင့်ပြီး ID Selector ထက် နိမ့်ပါတယ်။ ဒါကိုလည်း ထည့်သွင်းမှတ်သားသင့်ပါတယ်။

Selector Priority ကို Override လုပ်ချင်ရင်လည်း လုပ်လိုတော့ ရပါတယ်။ !important လို့ခေါ်တဲ့ ရေးထုံးကို အသုံးပြုရပါတယ်။ ဥပမာ ဒီလိုပါ။

```
div li {
    color: blue !important;
}

div ol li {
    color: brown;
}
```

နမူနာအရ div ol li က ပိုတိကျလို့ Priority မြင့်ပါတယ်။ ဒါပေမယ့် div li နဲ့ ရေးထားတဲ့ color: blue မှာ !important အမှတ်အသား ပါနေတဲ့အတွက် Priority နဲ့မဆိုင်တော့ဘဲ သူကိုဦးစားပေးအတည်ယူ အလုပ်လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် <li> ရဲ့ color က blue ဖြစ်သွားမှာပဲ ဖြစ်ပါတယ်။

လက်စနဲ့ထည့်သွင်းမှတ်သားသင့်တဲ့ သဘောသဘာဝ ရှိပါသေးတယ်။ ဒီလိုပါ။

```
div li {
    background: yellow;
    color: blue;
}

div ol li {
    color: brown;
}
```

နမူနာအရ Priority နိမ့်တဲ့ div li မှာ color နဲ့ background ဆိုပြီး သတ်မှတ်ချက် နှစ်ခုပါဝင်ပါတယ်။ Priority မြင့်တဲ့ div ol li မှာတော့ color သတ်မှတ်ချက်တစ်ခုပဲ ပါဝင်ပါတယ်။ Priority မြင့်တဲ့ div ol li ကို ဦးစားပေး အလုပ်လုပ်တဲ့အတွက် <li> ရဲ့ color က brown ဖြစ်သွားမှာပါ။ ဒါပေမယ့် div li မှာ သတ်မှတ်ထားတဲ့ background: yellow လည်း အသက်ဝင်အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။ Priority နိမ့်လို့ ပယ်တယ်ဆိုပေမယ့် သတ်မှတ်ချက်ခြင်း တူတာကိုသာ ပယ်မှာပါ။ မတူတဲ့ သတ်မှတ်ချက်တွေကိုတော့ ပေါင်းပြီး အလုပ်လုပ်ပေးသွားမှာပဲ ဖြစ်ပါတယ်။

ဒါတွေက စိတ်ရှုပ်စရာ ကောင်းချင်ကောင်းနေပါလိမ့်မယ်။ ဒီလောက်ကြီး ခက်လှတဲ့သဘောသဘာဝတွေတော့ မဟုတ်ပါဘူး။ စာနဲ့ရှင်းပြရတာထက် လက်တွေပြရတာက ပိုကောင်းနိုင်တဲ့ အကြောင်းအရာ မျိုးတွေ ဖြစ်နေလိုပါ။ ဒါကြောင့် လက်တွေချရေးပြီး စမ်းကြည့်လိုက်ပါ။ စာအနေနဲ့ ဖတ်ကြည့်ရတာ စိတ်ရှုပ်စရာ ဖြစ်နေမယ်ဆိုရင်တောင် ချရေးကြည့်ပြီး မြင်သွားတဲ့အခါ ချက်ခြင်းသဘောပေါက်သွားတာကို တွေ့ရပါလိမ့်မယ်။

ဒီ Selector တွေနဲ့ တဲ့ဖက်လေ့လာရတဲ့ Pseudo-Class ဆိုတာ ရှိပါသေးတယ်။ ဆက်ကြည့်ကြပါမယ်။

## Pseudo-Classes

Pseudo-Classes တွေကို ရေတွက်ကြည့်တဲ့အခါ (၅၀) ကျော်ရှိနေတာကို တွေ့ရပါတယ်။ အကုန်လုံးတော့ တစ်ခါတည်း ကြည့်စရာမလိုပါဘူး။ အသုံးများမယ့် ဟာတွေကို ရွေးမှတ်ထားပြီး ကျွန်တာကို လိုအပ်လာတော့မှ ကြည့်လိုက်ရင်ရပါတယ်။ ပထမဆုံး :hover, :active, :visited ဆိုတဲ့ Pseudo-Class (၃) ခုကနေ စကြည့်ပါမယ်။

Pseudo-class တွက် Selector တွေနဲ့ တွဲသုံးလို့ ရပါတယ်။ Full-colon (:) နဲ့စပါတယ်။ Full-colon နှစ်ခု (::) နဲ့ရေးရတဲ့ Pseudo-Element ဆိုတာတွေ ရှိပါသေးတယ်။ အဲဒါတွေကိုတော့ ထည့်မကြည့်ချင် သေးပါဘူး။ နောက်မှ ဆက်လေ့လာရမှာပါ။ အခုကြည့်ချင်တဲ့ Pseudo-Class (၃) ခု၊ အလုပ်လုပ်ပုံကို စမ်းသပ်နိုင်ဖို့အတွက် ဒီလို့ရေးစမ်းလို့ရပါတယ်။

### HTML

```
<a href="#one">Link One</a>
<a href="#two">Link Two</a>
```

HTML Link နှစ်ခုထည့်ထားပါတယ်။ ဒီ Link တွေအတွက် CSS ကုဒ်ကို ဆက်လက်လေ့လာကြည့်ပါ။

### CSS

```
a {
    color: blue;
}

a:hover {
    color: red;
}

a:active {
    color: green;
}

a:visited {
    color: brown;
}
```

ဒီကုဒ်အရ Link တွေအားလုံးဟာ စာလုံးအပြာရောင် ဖြစ်ရပါမယ်။ တစ်ကယ်တော့ နိုက်တည်းက ပြားမြို့ သားပါ။ တမင်သဘောသဘာဝ ပိုပေါ်လွှင်အောင် ထပ်ပေးလိုက်တာပါ။ a:hover မှာ အနီရောင်လို့ ပြော လိုက်တဲ့အတွက် Link တွေပေါ်မှာ Pointer ဖြတ်သွားရင် (သို့မဟုတ်) Link တွေကို Pointer နဲ့ ထောက် လိုက်ရင် အနီရောင် ပြောင်းသွားမှာပါ။ a:active မှာ အစိမ်းရောင်လို့ သတ်မှတ်ထားတဲ့အတွက် Link တွေကို နိုပ်လိုက်ရင် နိုပ်လိုက်တဲ့အခါန်လေးမှာ အစိမ်းရောင်ဖြစ်သွားမှာပါ။ a:visited မှာ brown လို့ ပြောထားတဲ့အတွက် တစ်ခါနိုပ်ဖူးတဲ့ လင့်တွေဟာ အပြာရောင် မဟုတ်တော့ဘဲ နိုည်းရောင် ဖြစ်သွားမှာပါ။ Codepen ထဲမှာ ရေးပြီး စမ်းကြည့်လို့ ရပါတယ်။

ဒါဟာ အသုံးများမယ့် Pseudo-Class (၃) မျိုးရဲ့ သဘောသဘာဝပါပဲ။ :hover တို့ :active တို့ကို ကြိုက်တဲ့ Element တွေအတွက် သတ်မှတ်ပေးလို့ရပါတယ်။ Link အပါအဝင် နှစ်လို့ရတဲ့ ခလုပ်တွေမှာ အများအားဖြင့် သတ်မှတ်ကြလေ့ ရှုပါတယ်။ :visited ကတော့ Link တွေအတွက်ပဲ သတ်မှတ်လို့ရတဲ့ Pseudo-Class ပါ။ ဆက်လက်လေ့လာသင့်တဲ့ Pseudo-Class (၃) ခုအကြောင်းကို ထပ်ပြောပါမယ်။

:first-child, :last-child နဲ့ :nth-child ဆိုတဲ့ Pseudo-class တွေ ဟာလည်း တော်တော်လေး အသုံးဝင်တဲ့ လုပ်ဆောင်ချက်တွေ ဖြစ်ပါတယ်။ Select လုပ်ထားတဲ့ Element တွေထက် ထိပ်ဆုံးတစ်ခုပဲ လိုချင်တယ်၊ နောက်ဆုံးတစ်ခုပဲ လိုချင်တယ်၊ တစ်ခုကျဉ်းလိုချင်တယ်၊ နှစ်ခုကျဉ်းလိုချင်တယ်၊ စသဖြင့် ရွှေးချယ်ဖို့အတွက် အသုံးပြုရပါတယ်။ ဥပမာ -

### HTML

```
<ul>
  <li>Item One</li>
  <li>Item Two</li>
  <li>Item Three</li>
  <li>Item Four</li>
  <li>Item Five</li>
</ul>
```

ဘာမှအဆန်းအပြေားမဟုတ်ပါဘူး၊ Item တစ်ချို့ပါဝင်တဲ့ List တစ်ခုဖြစ်ပါတယ်။ သူ့အတွက် CSS ကို လေ့လာကြည့်ပါ။

### CSS

```
ul li:first-child {
  font-weight: bold;
}
ul li:last-child {
  font-style: italic;
}
ul li:nth-child(3) {
  color: red;
}
ul li:nth-child(2n) {
  background: cyan;
}
```

:first-child ကို သုံးပြီး ပထမဆုံး <li> ရဲစာလုံးကို Bold လုပ်ထားပါတယ်။ :last-child ကို သုံးပြီး နောက်ဆုံး <li> ကို စာလုံးစောင်း Italic လုပ်ထားပါတယ်။ :nth-child(3) လိုပြောထားတဲ့ အတွက် သုံးချမှောက် <li> ရဲစာလုံးအရောင် အနီရောင်ဖြစ်နေမှာပါ။ တစ်ကြိမ်ပဲ အသက်ဝင်မှာပါ။ :nth-child(2n) လိုပြောထားတဲ့ အတွက် နှစ်ချမှောက် <li> တိုင်းမှာ နောက်ခံအရောင် ပါဝင်သွားမှာ ဖြစ်ပါတယ်။ (J) ခုကျော် ရှိသမျှအကုန်လုံးမှာ အသက်ဝင်မှာပါ။ ဒါမျိုးတွေကြောင့် တော်တော် အသုံးဝင်တဲ့ Pseudo-Class တွေလို့ ပြောတာပါ။ ရလဒ်ကဒီလိုဖြစ်မှာပါ -

```

HTML
1▼ <ul>
2▼   <li>Item One</li>
3▼   <li>Item Two</li>
4▼   <li>Item Three</li>
5▼   <li>Item Four</li>
6▼   <li>Item Five</li>
7 </ul>

CSS
1▼ ul li:first-child {
2   font-weight: bold;
3 }
4
5▼ ul li:last-child {
6   font-style: italic;
7 }
8
9▼ ul li:nth-child(3) {
10  color: red;
11 }
12
13▼ ul li:nth-child(2n) {
14  background: cyan;
15 }

JS

```

ဒီလောက်ဆုံးရင် အခြေခံဆင့်အနေနဲ့ လေ့လာသင့်တဲ့ Selector တွေ Pseudo-Class တွေ ဖုံးပါပြီ။ ဒီ အခြေခံတွေနဲ့ အသားကျပိုင်နိုင်ပြီဆိုမှ ကျန်တဲ့ Pseudo-Class တွေ Pseudo-Element တွေကို လေ့လာပါ။ အထူးသဖြင့် အသုံးဝင်နိုင်တာတွေက :empty, :not နဲ့ :target ဆိုတဲ့ Pseudo-Class တွေ နဲ့ အတူ ::before, ::after နဲ့ ::placeholder ဆိုတဲ့ Pseudo-Element တွေဖြစ်ပါတယ်။ :focus, :checked, :readonly, :enabled, :disabled, :valid, :invalid စသဖြင့် Input တွေနဲ့ တွဲသုံးရတဲ့ Pseudo-class တွေလည်း ရှိပါသေးတယ်။ အမည်တွေလောက်ပဲ မှတ်ထားပြီး နောက်လိုအပ်လာတော့မှ ဆက်လက်လေ့လာသွားလိုက်ပါ။

## CSS Display

Element တွေမှာ မတူကွဲပြားတဲ့ Display Type အမျိုးမျိုးရှိပြီး၊ အဲဒီ Display Type ကို CSS နဲ့ စီမံလိုက်ပါတယ်။ အခြေခံအကျဆုံးနဲ့ အရေးကြီးဆုံး Display Type (J) မျိုးရှိပါတယ်။ block နဲ့ inline တို့ ဖြစ်ကြပါတယ်။ တစ်ချို့ Element တွေဟာ block Element တွေဖြစ်ပြီး တစ်ချို့ Element တွေကတော့ inline Element တွေဖြစ်ကြပါတယ်။ ဒီသဘောသဘာဝကို မြင်ဖို့ လက်တွေရေးစမ်းကြည့်မှ ပိုမြင်ပါလိမ့်မယ်။ ဒါကြောင့် နမူနာတွေ ပြပါမယ်။ တစ်ခါတည်း လိုက်ရေးကြည့်ဖို့ တိုက်တွန်းလိုပါတယ်။

```

HTML
1▼ <div>A block</div>
2▼ <div>Another block</div>
3▼ <span>Some content</span>
4▼ <span>More content</span>
5

CSS
1▼ div, span {
2    background: cyan;
3    margin-bottom: 20px;
4    padding: 10px;
5 }
6

JS

```

နမူနာကိုလေ့လာကြည့်ပါ။ <div> Element နှစ်ခုနဲ့ <span> Element နှစ်ခုရှိပါတယ်။ <div> ဟာ block Element ဖြစ်ပြီး <span> ကတော့ inline Element ပါ။ CSS မှာ div, span နှစ်ခုလုံး အတွက် တူညီတဲ့ Property တွေကို သတ်မှတ်ထားပေမယ့် တစ်ဖက်က ဖော်ပြပုံမှာကြည့်လိုက်ပါ၊ ဖော်ပြပုံမတူကြပါဘူး။ Display Type မတူကြလိုပါ။

<div> Element တွေက နေရာရှိသလောက် အပြည့်နေရာယူဖော်ပြထားတာကို တွေ့ရမှာဖြစ်ပြီး <span> Element တွေကတော့ သူ့ Content ရှိသလောက်လေးပဲ နေရာယူထားပါတယ်။ နောက်ထပ် သတိပြုရမှာက <div> တွေဟာ အပေါ်အောက် အစီအစဉ်နဲ့ဖော်ပြပြီး <span> တွေကတော့ ဘေးချင်းကပ် အစီအစဉ်နဲ့ ဖော်ပြပါတယ်။ <div> မို့လို့ ဒီလိုဖော်ပြတာ <span> မို့လို့ ဒီလိုဖော်ပြတာ မဟုတ်ပါဘူး။ Display Type ကြောင့် block ကို တစ်မျိုးဖော်ပြပြီး inline ကို တစ်မျိုးဖော်ပြနေတာပါ။

<div> ကဲ့သို့သော တစ်ခြား Block Element တွေရှိပါတယ်။ ဥပမာ - <h1> ... <h6>, <p>, <ul>, <ol> စတဲ့ Element တွေဟာ block အမျိုးအစား Element တွေပါ။ <label>, <a>, <b>, <em> စတဲ့ Element တွေကတော့ inline အမျိုးအစား Element တွေပါ။ Block အားလုံးဟာ နမူနာမှာ ပြထားတဲ့ <div> နဲ့ တူညီတဲ့လက္ခဏာရှိပြီး၊ Inline အားလုံးကတော့ နမူနာမှာပြထားတဲ့ <span> နဲ့ တူညီတဲ့ လက္ခဏာရှိပါတယ်။ နမူနာကို နည်းနည်း ထင်ပြင်ကြည့်ပါမယ်။

```


HTML



```

1▼ <div>A block</div>
2▼ <div>Another block</div>
3▼ <span>Some content</span>
4▼ <span>More content</span>
5

```



CSS



```

1▼ div, span {
2    background: cyan;
3    margin-bottom: 20px;
4    padding: 10px;
5    width: 200px;
6    height: 100px;
7 }
8

```



JS



A block



Another block



Some content More content


```

ဒီတစ်ခါ width နဲ့ height ဆိုတဲ့ Property နှစ်ခုထပ်တိုးလိုက်တာပါ။ ဒီလို အရွယ်အစားသတ်မှတ်တဲ့ Property တွေဟာ Block ဖြစ်တဲ့ <div> မှာသက်ဝင်အလုပ်လုပ်ပေမယ့် Inline ဖြစ်တဲ့ <span> မှာ သက်ဝင်ခြင်းမရှိဘူးဆိုတာကို သတိပြုရမှာပါ။ Inline Element တွေကို အရွယ်အစားသတ်မှတ်လို့ မရပါဘူး။ အဲဒေါကို မသိရင် ရေးထားတဲ့ Property တွေက အလုပ်လည်း မလုပ်ဘူးဆိုပြီးတော့ စိတ်ညွစ်သွားနိုင်ပါတယ်။ စိတ်ညွစ်စရာမလိုပါဘူး၊ သဘာဝအရ Inline တွေကို အရွယ်အစား သတ်မှတ်လို့ မရတာပါ။ နောက်ထပ်ထူးခြားချက်အနေနဲ့ Block တွေ အပေါ်အောက် အစီအစဉ်အတိုင်း ပြတယ်ဆိုတာကို ထပ်ပြောချင်ပါတယ်။ အရွယ်အစားသတ်မှတ်လိုက်လို့ နေရာလွှတ်တွေ ဘေးမှာပို့ထွက်လာပေမယ့် ဘေးမှာကပ်ပြီး မပြုဘဲ အောက်ဘက်မှာပဲ ပြတာကို သတိပြုပါ။

Display Type ကို ပြောင်းလိုရပါတယ်။ display လိုခေါ်တဲ့ CSS Property ကိုသုံးရပါတယ်။ ဒီလိုပါ။

```

HTML
1▼ <div>A block</div>
2▼ <div>Another block</div>
3▼ <span>Some content</span>
4▼ <span>More content</span>
5

CSS
1▼ div, span {
2    background: cyan;
3    margin-bottom: 20px;
4    padding: 10px;
5    width: 200px;
6    height: 100px;
7 }
8
9▼ span {
10    display: block;
11 }
12

JS

```

နမူနာမှာ span တွေကို display: block လိုပြောလိုက်ပါပြီ။ ဒါကြောင့် မူလက Inline ဖြစ်နေတဲ့ <span> Element တွေဟာ အခုတော့ Block ဖြစ်သွားပြီမို့လို့ အားသာချက်အနေနဲ့ အရွယ်အစား သတ်မှတ်လို့ရသွားသလို အားနည်းချက်အနေနဲ့ ဘေးချင်းကပ်ပြလို့မရတော့ဘဲ အပေါ်အောက်ဆင့်ပြီး ပြသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ ဒီသဘောသဘာဝဟာ CSS အခြေခံတွေထဲမှာ အရေးအကြီးဆုံးတစ်ခု အပါအဝင်ဖြစ်ပါတယ်။ display Property အတွက် အသုံးများမယ့် Value (ငါ) ခုရှိပါတယ်။

- inline
- block
- none
- flex

inline နဲ့ block ကတော့ ရှင်းပါတယ်။ Element ရဲ့ Display Type ကို လိုချင်သလို ပြောင်းလိုက်တာပါ။ ဟုတ်ပါတယ်၊ Block တွေကိုလည်း လိုအပ်ရင် Inline ပြောင်းလိုရပါတယ်။ လိုတော့လိုခဲပါတယ်။ none ကတော့ ဖျောက်ထားလိုက်တာပါ။ Element ကို ဖျောက်ထားချင်ရင် display: none လို့ သတ်မှတ်ပေးလိုက်တာ ထုံးစံပါပဲ။ flex ကတော့ နောက်မှပေါ်တဲ့ Display Type ပါ။ Layout တွေလုပ်ဖို့ အသုံးဝင်ပါတယ်။ ဘေးချင်းကပ်မပြတဲ့ Block တွေဟာ flex အတွင်းမှာဆိုရင် ပြုကြပါတယ်။ ဒီလိုပါ -

```

HTML
1 <div class="wrap">
2   <div>A</div>
3   <div>B</div>
4   <div>C</div>
5 </div>

CSS
1 .wrap {
2   display: flex;
3 }
4
5 .wrap div {
6   flex-grow: 1;
7   height: 200px;
8   background: cyan;
9   margin: 10px;
10  text-align: center;
11  line-height: 200px;
12 }
13
14 .wrap div:first-child {
15   flex-grow: 2;
16 }

JS

```

ဝင်မ <div> မှာ class ကို wrap လိုပေးထားပြီး display: flex Property ကိုသုံးထားတာ တွေ ရနိုင်ပါတယ်။ ဒါကြောင့် သူ့အတွင်းထဲက <div> တွေဟာ Block တွေဆိုပေမယ့် ဘေးချင်းကပ်စီပြီး ပြ သွားပေးတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။ အဲဒီမှာ flex-grow ဆိုတဲ့ Property ကိုလည်း သတိပြုပါ။ အတွင်း Element တွေမှာ width နဲ့ အကျယ်သတ်မှတ်မထားပါဘူး။ သတ်မှတ်ချင်ရင်သတ်မှတ်လို့ ရပါ တယ်။ နမူနာမှာတော့ width အစား flex-grow: 1 လို့ ပြောထားပါတယ်။ (၁) နေရာစာယူမယ်လို့ ပြောလိုက်တာပါ။ ပြီးတော့မှာ :first-child နဲ့ ပထမဆုံးတစ်ခုကို flex-grow: 2 လို့ ပြောထား တဲ့အတွက် သူက (၂) နေရာစာ ယူပြီးတော့ ဖော်ပြနေတာကိုလည်း သတိပြုကြည့်ပါ။ တန်ဖိုးတွေကို ကိုယ့် စိတ်တိုင်းကျ ပြောင်းပြီးတော့ စမ်းကြည့်နိုင်ပါတယ်။

Flexbox လိုခေါ်တဲ့ ဒီသဘောသဘာဝဟာ ကျယ်ပြန့်သလို အသုံးလည်း ဝင်ပါတယ်။ Element တွေကို ဘေးချင်းတိုက် ပြစေချင်တာလား၊ အပေါ်အောက်ပြစေချင်တာလား၊ မဆန့်တော့ရင် ဘာလုပ်ရမှာလည်း၊ ဆန့်အောင် ချုံပြပေးရမှာလား၊ နောက်တစ်လိုင်း ဆင်းပြပေးရမှာလား၊ အနီမ့်အမြင့် မညီရင် အပေါ်ဘက် ကို အညီယူပေးရမှာလား၊ အလယ်ကိုအညီယူပေးရမှာလား၊ စသဖြင့် သတ်မှတ်လို့ရတဲ့ Flexbox Property တွေ အများကြီးရှိနေပါတယ်။ ဒီနေရာမှာ အဲဒီလောက်ထိ ကျယ်ကျယ်ပြန့်ပြန့်မသွားသေးဘဲ၊ အခြေခံ သဘောဖြစ်တဲ့ display: flex သတ်မှတ်ပေးလိုက်ရင် သူ့အတဲ့က Block Element တွေကို ဘေးချင်း ကပ် အစီအစဉ်နဲ့ပြပေးတယ်ဆိုတဲ့ အချက်လောက်ကိုပဲ မှတ်ထားပေးပါ။

display Property မှာတစ်ခြား Value တွေ ရှိပါသေးတယ်။ inline-block, list-item, table, table-cell, inline-flex, grid, inline-grid စသဖြင့်ရှိတာတော့ အများ ကြီးပါပဲ။ လိုအပ်လာတဲ့အချိန်မှာ ဆက်လက်လေ့လာသွားရမှာဖြစ်ပါတယ်။ နည်းပညာလေ့လာတာ ကတော့ ဒီလိုပဲ တစ်ဆင့်ချင်းသွားမှပဲ ရပါမယ်။ တစ်ခါတည်း အကုန်သိချင်လိုတော့ မလွယ်ပါဘူး။ အရေးကြီးပြီး အသုံးများတာတွေ အရင်ကြည့်ပြီး အသုံးနည်းတာတွေကို လိုအပ်လာမှ ပြန်ကြည့်ရတဲ့ သဘောပါပဲ။

## CSS Box Model

ပြီးခဲ့တဲ့ နမူနာတွေမှာ margin တို့ padding တို့လို Property တွေကိုထည့်သုံးပြုခဲ့ပေမယ့် ရှင်းမပြုခဲ့ပါဘူး။ သင်သပ်ရှင်းရမှာ မို့လိုပါ။ ဒါတွေကလည်း အရေးကြီးပါတယ်။ CSS မှာ Box Model လိုပေါ်တဲ့ သဘောသဘာဝ ရှိပါတယ်။ Element တွေရဲ့ အရွယ်အစားပေါ်မှာ သက်ရောက်စေတဲ့ Property တွေပါ။ (၅) ခုရှိပါတယ်။

- width
- height
- margin
- padding
- border

ဒီ (၅) ခုမှာ width နဲ့ height ကတော့ ရှင်းပါတယ်။ Element ရဲ့လိုချင်တဲ့အရွယ်အစားရှိဖို့အတွက် ဒီ Property တွေနဲ့ သတ်မှတ်ရတာပါ။ margin Property ကတော့ တစ်ခြား Element တွေနဲ့ ဘယ်လောက်ခွာပြရမလဲဆိုတဲ့ အကွာအဝေး သတ်မှတ်ဖို့အတွက် သုံးရပါတယ်။ padding Property ကတော့ Element ရဲ့ဘောင်နဲ့ Element ထဲမှာရှိတဲ့အရာတွေ ဘယ်လောက် ခွာပြရမလဲဆိုတဲ့ အကွာအဝေးကို သတ်မှတ်ဖို့အတွက် သုံးရတာပါ။ border ကတော့ Element တွေမှာ ဘောင်ခတ်ဖို့အတွက် အသုံးပြုရပါတယ်။ ဒီနမူနာကို လေ့လာကြည့်ပါ။

```

HTML
1 <div class="wrap">
2   <div class="box1">Box One</div>
3   <div class="box2">Box Two</div>
4 </div>

CSS
1 .wrap {
2   width: 400px;
3   height: 400px;
4   border: 5px solid red;
5 }
6
7 .box1, .box2{
8   width: 400px;
9   height: 200px;
10}
11
12 .box1 {
13   background: yellow;
14 }
15
16 .box2{
17   background: cyan;
18 }

JS

```

width:400px နဲ့ height:400px သတ်မှတ်ထားတဲ့ Element အတွင်းထဲမှာ width:400px နဲ့ height:200px သတ်မှတ်ထားတဲ့ Element (J) ခရီးနေတာပါ။ သူဟာနဲ့သူ အရွယ်အစားကိုက်ညီလို ကွက်တိပါပဲ။ အဲဒီလို ကွက်တိဖြစ်နေတဲ့ Element တွေမှာ တစ်ခုနဲ့တစ်ခုလည်း ခွာပြီးတော့ ပြစေချင် တယ်။ အထဲမှုရှိတဲ့ စာနဲ့ Element ဘောင်နဲ့လည်း ကပ်နေလို ကွာသွားစေချင်တယ်။ ဒါကြောင့် margin နဲ့ padding Property တွေ သတ်မှတ်လိုက်တဲ့အခါ ဒီလိုဖြစ်သွားမှာပါ။

```

HTML
1 <div class="wrap">
2   <div class="box1">Box One</div>
3   <div class="box2">Box Two</div>
4 </div>

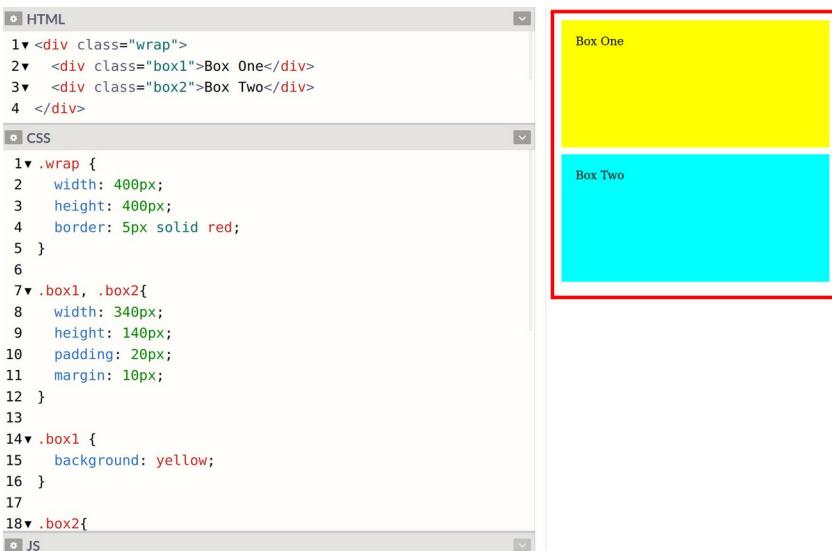
CSS
1 .wrap {
2   width: 400px;
3   height: 400px;
4   border: 5px solid red;
5 }
6
7 .box1, .box2{
8   width: 400px;
9   height: 200px;
10  padding: 20px;
11  margin: 10px;
12 }
13
14 .box1 {
15   background: yellow;
16 }
17
18 .box2{
19 }

JS

```

သတ်မှတ်ချက်အရ Element တစ်ခုနဲ့တစ်ခု ကွာသွားပါဖြီ။ အထဲကစာနဲ့ ဘောင်နဲ့လည်း ကွာသွားပါဖြီ။ ဒါ ပေမယ့် ဖော်ပြပုံအဆင်မပြတော့ပါဘူး။ အဆင်မပြရတဲ့ အကြောင်းရင်းကတော့ Element ရဲ့ အရွယ်အစား မူလထက် ပိုကြိုးသွားလိုပါ။ width တွေ height တွေ မပြောင်းဘဲနဲ့ မူလထက်ပိုကြိုးသွားရခြင်း အကြောင်းရင်းကတော့ သတ်မှတ်လိုက်တဲ့ margin တွေ padding တွေကို Element ရဲ့ အရွယ်အစားမှာ ပေါင်းထည့်သွားလိုပါ။ width က 400 ဆိုပေမယ့် padding: 20px ကြောင့် ဘယ်ညာ တစ်ဘက်ကို 20 စီထပ်တိုးသွားမှာပါ။ margin: 10px ကြောင့် ဘယ်ညာ တစ်ဘက်ကို 10 စီထပ်တိုးသွားမှာပါ။ ဒါကြောင့် width မှာ 400 လိုပြောထားပေမယ့် တစ်ကယ့် Element ရဲ့ အရွယ်အစားက 460 ဖြစ်သွားပါတော့တယ်။ height လည်းအတူတူပါပဲ။ 200 လိုသတ်မှတ်ထားပေ မယ့် တစ်ကယ့်အရွယ်အစားက 260 ဖြစ်သွားပါဖြီ။

ဒါကြောင့် ပင်မ Element ထဲမှာ မဆန့်တော့တဲ့ အတွက် အခုလို ကျော်ထွက်၊ လျှံတွက်ပြီး အဆင်မပြဖြစ်သွားရတာပါ။ ဒီသဘောသဘာဝကိုနားလည်ဖို့ အရေးကြီးပါတယ်။ နားမလည်ရင် margin လေး သတ်မှတ်လိုက်တာနဲ့ အကုန်လုံးပျက်ကုန်တယ်ဆိုပြီး စိတ်ညစ်ရပါလိမ့်မယ်။ စိတ်ညစ်စရာမလိုပါဘူး။ အရွယ်အစား ပြောင်းသွားလို ပြန်ညိုပေးလိုက်ရင် ရပါတယ်။ ဒီလိုပါ။



```

HTML
1▼ <div class="wrap">
2▼   <div class="box1">Box One</div>
3▼   <div class="box2">Box Two</div>
4 </div>

CSS
1▼ .wrap {
2   width: 400px;
3   height: 400px;
4   border: 5px solid red;
5 }
6
7▼ .box1, .box2{
8   width: 340px;
9   height: 140px;
10  padding: 20px;
11  margin: 10px;
12 }
13
14▼ .box1 {
15  background: yellow;
16 }
17
18▼ .box2{

```

အခုတော့ ကွက်တိဖြစ်သွားပါဖြီ။ width ကို 340 လိုလျှော့ပြီး သတ်မှတ်လိုက်တဲ့ အတွက် ထပ်တိုးလာတဲ့ margin, padding တွေနဲ့ ပေါင်းလိုက်တဲ့ အခါ 400 ဖြစ်သွားလို အခုလိုကွက်တိအဆင်ပြသွားခြင်းပဲ ဖြစ်ပါတယ်။

Border လည်းအတူတူပါပဲ။ Border သတ်မှတ်လိုက်တာနဲ့ မူလအရွယ်အစားမှာ ထပ်တိုးသွားမှာပါ။ နှမူနာ မှာ ပင်မ Element မှာ Border ထည့်ပြထားပါတယ်။

### CSS

```
border: 5px solid red;
```

ရှေ့ဆုံးက 5px က Border ရဲ့ အရွယ်အစားဖြစ်ပြီး၊ solid ကတေသ့ Border Style ဖြစ်ပါတယ်။ သတ်မှတ်လို့ရတဲ့ Style အမျိုးမျိုး ရှိပေမယ့် solid, dotted နဲ့ dashed (၃) မျိုးကို အသုံးများပါတယ်။ ကိုယ့်ဘာသာ ပြောင်းပြီး စမ်းကြည့်နိုင်ပါတယ်။ solid ကတေသ့ လိုင်းအပြည့် ဘောင်ခတ်တာပါ။ dotted ကတေသ့ အစက်လေးတွေနဲ့ ဘောင်ခတ်တာပါ။ dashed ကတေသ့ လိုင်းပြတ်လေးတွေနဲ့ ဘောင်ခတ်တာပါ။ နောက်ဆုံးက red ကတေသ့ Border ရဲ့ အရောင်ဖြစ်ပါတယ်။ နှစ်သက်ရာအရောင် ပေးလို့ရပါတယ်။ စမ်းရလွယ်တဲ့ red, green, blue, purple, brown, black စတဲ့အရောင် တွေနဲ့ပဲ စမ်းကြည့်လိုက်ပါ။ ခကာနောမှ အရောင်တွေ အကြောင်း ထပ်ပြောပေးပါမယ်။

အတွင်းထဲက Element တွေမှာလည်း Border ပေးကြည့်ပါ။ margin, padding တို့လိုပဲ မူလအရွယ်အစားမှာ ထပ်တိုးသွားတာကို တွေ့ရပါလိမ့်မယ်။ width, height ကို ပြန်လိုပေးလိုက်ရင် အဆင်ပြေသွားပါလိမ့်မယ်။

margin တွေ padding တွေ border တွေကို အခုလို တစ်ဘက်ချင်း ပေးလို့ရပါတယ်။

### CSS

```
margin-top: 10px;  
margin-right: 10px;  
padding-left: 10px;  
border-bottom: 5px solid red;
```

px အစား သုံးလို့ရနိုင်တဲ့ တစ်ခြား Unit အကြောင်းကို ခကာနေတော့ ပြောပြပါမယ်။ လောလောဆယ် စမ်းချင်ရင် px နဲ့ပဲရှေ့ကတန်ဖိုးကို ပြောင်းပြီး စမ်းကြည့်နိုင်ပါတယ်။ margin, padding တွေ ရေးနည်းနောက်ထပ် (၂) နည်း မှတ်သင့်ပါသေးတယ်။

**CSS**

```
padding: 10px 20px;  
margin: 10px 20px 5px 30px;
```

နမူနာမှာ padding အတွက် Value နှစ်ခုကိုပေးထားပါတယ်။ ရှေ့က 10px က top နဲ့ bottom အတွက်ပါ။ နောက်က 20px ကတော့ left နဲ့ right အတွက်ဖြစ်ပါတယ်။ ဒီနည်းနဲ့ တစ်ကြောင်းထဲရေးပြီး အပေါ်အောက်၊ ရှေ့နောက် မတူအောင် ပေးလို့ရတာပါ။ margin အတွက်တော့ Value လေးခုပေးထားပါတယ်။ top, right, bottom, left ဆိုတဲ့ သဘောပါ။ ဒီနည်းနဲ့ တစ်ကြောင်းထဲရေးပြီး မျက်နှာစာလေးဘက်လုံး Value တစ်ခုစီ ပေးလို့ရခြင်းဖြစ်ပါတယ်။ margin-left နဲ့ margin-right အတွက် auto ဆိုတဲ့ Value လည်းရှိပါသေးတယ်။ ဘယ်ဘက်ကနေရာလွတ်နဲ့ ညာဘက်ကနေရာလွတ်ကို နှစ်ဘက်ညီအောင် ယူပေးသွားမှာပါ။ ဒါကြောင့် Element ကို အလယ်မှာ ဖော်ပြစ်လိုရင် auto လို့ margin ကို သတ်မှတ်ပေးကြလေ့ ရှိပါတယ်။ ဒါပေမယ့် သတိပြုပါ။ auto ဆိုတဲ့ Value ဟာ top နဲ့ bottom အတွက် အလုပ်မလုပ်ပါဘူး။ ဒီလိုကုဒ်မျိုးကို မကြာခဏတွေ့ရနိုင်ပါတယ်။

**CSS**

```
margin: 20px auto;
```

margin-top နဲ့ margin-bottom ကို 20px လိုသတ်မှတ်ပြီး margin-left နဲ့ margin-right ကို auto လို့ သတ်မှတ်ပေးလိုက်တာပါ။

နောက်ဆုံးတစ်ခုထပ်မှတ်ပါ။ box-sizing လိုခေါ်တဲ့ Property ပါ။ နောက်မှ ထပ်တိုးလာတဲ့ တော်တော်အသုံးဝင်တဲ့ Property ဖြစ်ပါတယ်။ Element မှာ box-sizing ကို bordered-box လို့ပေးလိုက်ရင် အခုတက်နေတဲ့ ပြဿနာတွေ တော်တော်များများ ပြောလည်သွားပါတယ်။ ဘာဖြစ်လိုလဲဆိုတော့ padding တွေ border တွေ ပေးလိုက်လို့ လိုအပ်လာတဲ့နေရာကို ထပ်တိုးမယူဘဲ၊ မူလ width, height ထဲကနေ ဖွဲ့ယူသွားမှာ ဖြစ်လို့ padding ထည့်လိုက်တာ size ပြောင်းသွားတယ်ဆိုတာမျိုး ဖြစ်စရာမလိုတော့ပါဘူး။

```

HTML
1▼ <div class="wrap">
2  <div class="box1">Box One</div>
3  <div class="box2">Box Two</div>
4 </div>

CSS
1▼ .wrap {
2   width: 400px;
3   height: 400px;
4   border: 5px solid red;
5 }

7▼ .box1, .box2{
8   width: 400px;
9   height: 200px;
10  padding: 20px;
11  box-sizing: border-box;
12  border: 5px solid green;
13 }

15▼ .box1 {
16  background: yellow;
17 }

JS

```

နမူနာကိုလေ့လာကြည့်ပါ။ box-sizing ပေးထားတဲ့အတွက် Element ရဲ width, height ကိုယ်ပေးချင်တဲ့အတိုင်း 400, 200 လိုပေးထားပါတယ်။ padding စွဲ border တွေပါပေမယ့်လည်း ပြန်ညိုစရာ မလိုတော့ပါဘူး။ margin ပေးချင်ရင်တော့ မရပါဘူး။ margin အတွက် ပြန်ညိုပေးရမှာပါ။

## CSS Position

CSS ကိုအသုံးပြုပြီး Element တွေဖော်ပြရမယ့် တည်နေရာကို ကိုယ့်စိတ်တိုင်းကျ အတိအကျလည်းသတ်မှတ်ပေးလိုရပါတယ်။ ဒီနည်းက ဝဘ်ဆိုက် Layout တွေဘာတွေ လုပ်တဲ့နေရာမှာ သိပ်အသုံးမဝင်ပေမယ့် UI Component တွေအတွက်တော့ တော်တော်အသုံးဝင်ပါတယ်။ Layout အတွက် သိပ်အသုံးမဝင်ဘူးဆိုတာက၊ စဉ်းစားကြည့်ပါ။ Layout မှာပါတဲ့ Element တွေကိုသာ တစ်ခုချင်း စိတ်တိုင်းကျ အတိအကျ နေရာသတ်မှတ်ထားမယ်ဆိုရင်၊ ပြင်ချင်တဲ့အခါ ဘယ်လိုလုပ်မလဲ။ အကုန်လုံးကို လိုက်ပြင်ရတော့မှာပါ။ လက်တွေ့မကျပါဘူး။ ဒါကိုတမင်ကြိုပြောတာပါ။ CSS Position Property အကြောင်း သိသွားရင် အရမ်းသဘောကျသွားပြီး နေရာတိုင်းမှာ သုံးချင်စိတ် ပေါ်လာတတ်ပါတယ်။ နေရာတိုင်း သုံးလိုတော့ အဆင်ပြေမှာမဟုတ်ဘူး၊ သူသင့်တော်ရာနေရာမှာသာ အသုံးပြုရမှာဖြစ်တယ် ဆိုတဲ့သဘောပါ။ နမူနာကိုကြည့်ပါ။



```

1 <div class="box1"></div>
2 <div class="box2"></div>
3

4 .box1 {
5   width: 300px;
6   height: 200px;
7   background: cyan;
8   position: absolute;
9   top: 100px;
10  left: 100px;
11 }
12
13 .box2 {
14   width: 300px;
15   height: 200px;
16   background: yellow;
17   position: absolute;
18   top: 150px;
19   left: 150px;
20 }
21
22
23

```

<div> Element နှစ်ခုရှိပြီး နှစ်ခုလုံးအတွက် ဆင်တူတဲ့ Property တွေသတ်မှတ်ထားပါတယ်။ ထူးခြားချက် အနေနဲ့နှစ်ခုလုံးမှာ position: absolute သတ်မှတ်ချက်ပါဝင်ခြင်း ဖြစ်ပါတယ်။ ဒါကြောင့် ဒီ Element တွေရဲ့ တည်နေရာကို အတိအကျသတ်မှတ်လိုရသွားပါပြီ။ နမူနာမှာ top နဲ့ left Property တွေကိုသုံးပြီး နေရာမတိမ်းမယိမ်း သတ်မှတ်ထားလို့ တစ်ခုပေါ်တစ်ခု ထပ်ပြီး ဖော်ပြနေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ အဲဒီတန်ဖိုးတွေကို ပြောင်းပြီးစမ်းကြည့်လို့ရပါတယ်။

position မှာ fixed လိုပေါ်တဲ့ Value လည်း ရှိပါသေးတယ်။ absolute နဲ့ အခြေခံအားဖြင့် တူပြီး Scroll Behavior မှာ ကွာသွားပါတယ်။ absolute က Scroll ဆွဲလိုက်ရင် Scroll နဲ့အတူပါသွားပြီး fixed ကတော့ Scroll နဲ့မပါဘဲ သတ်မှတ်ထားတဲ့ နေရာမှာ အမြဲတည်ရှိနေမှာပါ။ ဘာကိုပြောတာလဲ မျက်စိတဲ့မှာ မဖြင်ရင် CSS ကုဒ်ထဲမှာ ဒီလိုဖြည့်ပြီး စမ်းကြည့်ပါ။

## CSS

```

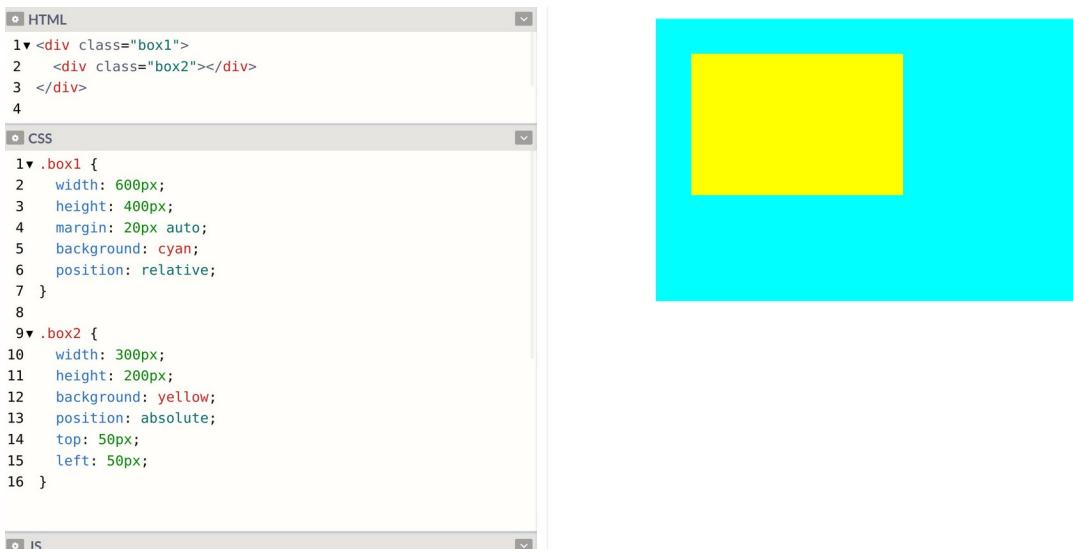
body {
  height: 2000px;
}

.box1 {
  position: fixed;
  ...
}

```

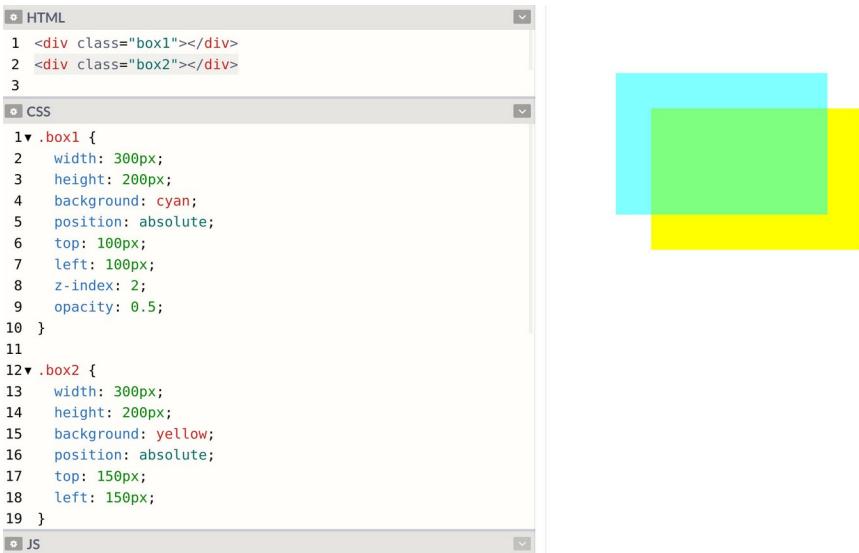
Body Height ကို 2000 ပေးလိုက်လို Scroll Bar ပေါ်လာပါလိမ့်မယ်။ .box1 ရဲ့ position ကိုတော့ absolute မဟုတ်တော့ဘဲ fixed လိုပြောင်းပေးထားပါတယ်။ ဒါကြောင့် Box နှစ်ခု Scroll Behavior မတူတော့ပါဘူး။ စမ်းရေးပြီး Scroll ဆွဲကြည့်ပါ ဘာကွာလဲဆိုတာကို ကိုယ်တိုင် တွေ့မြင်ရပါလိမ့်မယ်။

position မှာ relative ဆိုတဲ့ Value တစ်ခုလည်းရှုပါသေးတယ်။ သူရဲ့သဘောသဘာဝက နည်း နည်း ထူးခြားပါတယ်။ နမူနာကိုကြည့်ပါ။



နမူနာမှာ .box2 ကို .box1 ထဲမှာ ထည့်ထားပါတယ်။ .box1 အတွက် Style တွေကိုလေ့လာကြည့်ရင် margin ကိုသုံးပြီးအလယ်ပို့ထားတာ တွေ့ရနိုင်ပါတယ်။ ပြီးတော့ position: relative လိုလည်း ပေးထားပါတယ်။ ဒီလိုပေးထားတဲ့အတွက် သူအထဲမှာရှိတဲ့ .box2 ရဲ့ Position ကိုတွက်တဲ့ထဲမှာသူနဲ့ Relative ယူပြီးတွက်ပေးသွားမှာပဲဖြစ်ပါတယ်။ ဒီ position: relative သာမပါခဲ့ရင် .box2 ရဲ့ top နဲ့ left ကို 50px လိုသတ်မှတ်ထားလို Document Border ကနေ 50px စီစွာပြသွားမှာပါ။ အခုတော့ Document Border ကနေ 50px စီစွာပြတာ မဟုတ်တော့ဘဲ .box1 ရဲ့ Border ကနေ 50px စီစွာပြနေတာကို တွေ့မြင်ရခြင်း ဖြစ်ပါတယ်။ ပိုပြီးတော့ မြင်သာစေဖို့ ကိုယ်တိုင် position: relative Property ကို ထည့်ပြီးတစ်ခါ ဖြုတ်ပြီးတစ်ခါ စမ်းကြည့်လိုက်ပါ။ မြင်သွားပါလိမ့်မယ်။

Position အကြောင်းလေ့လာတဲ့အခါ တွဲဖက်အသုံးပြုလေ့ရှိတဲ့ Property (J) ခုရှုပါသေးတယ်။ z-index နဲ့ opacity တို့ဖြစ်ပါတယ်။ ဒီနမူနာလေးကို ထပ်ကြည့်ပေးပါ။



The screenshot shows a browser developer tools interface with three tabs: HTML, CSS, and JS. The HTML tab contains the following code:

```

1 <div class="box1"></div>
2 <div class="box2"></div>
3

```

The CSS tab contains the following styles:

```

1▼ .box1 {
2   width: 300px;
3   height: 200px;
4   background: cyan;
5   position: absolute;
6   top: 100px;
7   left: 100px;
8   z-index: 2;
9   opacity: 0.5;
10 }
11
12▼ .box2 {
13   width: 300px;
14   height: 200px;
15   background: yellow;
16   position: absolute;
17   top: 150px;
18   left: 150px;
19 }

```

The JS tab is empty. To the right of the code editor, there is a preview window showing two overlapping rectangular boxes. The top box is cyan (z-index: 2) and has a semi-transparent effect (opacity: 0.5). The bottom box is yellow (z-index: 1).

နမူနာမှာ .box1 ရော .box2 ပါ position: absolute ဖြစ်ပါတယ်။ တည်နေရာမတိမ်းမယိမ်းမို့ လို တစ်ခုနဲ့တစ်ခု ထပ်နေပါတယ်။ .box1 အတွက် z-index: 2 လိုပေးထားတဲ့အတွက် .box1 ကို အပေါ်ကထပ်ပြီး မြင်ရမှာဖြစ်ပါတယ်။ ကိုယ်တိုင်လက်တွေ စမ်းကြည့်ပါ။ z-index ပါရင်တစ်မျိုး၊ မပါ ရင်တစ်မျိုး စမ်းကြည့်လိုက်ပါ။ z-index ရဲ့ အလုပ်လုပ်ပုံက ရိုးရိုးလေးပါ။ Element တွေထပ်လာတဲ့ အခါ z-index တန်ဖိုး မြင့်တဲ့သူကို အပေါ်ကထပ်ပြီး ပြပေးမှာ ဖြစ်ပါတယ်။ opacity တော့ နမူနာမှာ မြင်တွေ့နေရတဲ့အတိုင်းပါပဲ။ Element ရဲ့ Transparency Level ကို ညိုဖို့ သုံးနိုင်ပါတယ်။ Value 0 ဆိုရင် မိန့်လွန်းလို့ လုံးဝပျောက်သွားပါလိမ့်မယ်။ Value 1 ဆိုရင် ထင်ရှားလွန်းလို့ မူရင်းအတိုင်း မြင်ရပါလိမ့်မယ်။ 0 နဲ့ 1 ကြားထဲမှာ ကိုယ်လိုသလောက်တန်ဖိုးကို ပေးထားခြင်းအားဖြင့် ထွင်းဖောက်မြင်ရတဲ့ Element တွေကို ရရှိမှာ ဖြစ်ပါတယ်။

ဒီလောက်ဆိုရင် CSS နဲ့ပက်သက်ပြီး သိသင့်တဲ့ သဘောသဘာဝတွေ အတော်စုံနေပါပြီ။ Selector, Display Type, Box Model နဲ့ Position တို့ကို လေ့လာခဲ့ခြင်းဖြစ်ပါတယ်။ Property တွေ Value တွေက လေ့လာရတာ မခက်ပါဘူး။ အကုန်အလွတ် မှတ်ထားစရာလည်း မလိုပါဘူး။ လိုတော့မှ ပြန်ကြည့်ပြီးရေးသွားလို့ ရပါတယ်။ အခုဖော်ပြခဲ့တဲ့ သဘောသဘာဝတွေကို ကောင်းကောင်းနားလည်ဖို့က ပိုအရေးကြီးပါတယ်။ ဒါတွေနားလည်မှသာ CSS ကို ကျွမ်းကျင်ပိုင်နိုင်စွာ အသုံးချုပ်မှာပါ။

## CSS Unit

CSS နဲ့ Element တွေရဲ့ အရွယ်အစားတို့ ဖွန်းအရွယ်အစားတို့လို အရွယ်အစားတွေ သတ်မှတ်တဲ့အခါမှာ သုံးရတဲ့ Unit တွေရှိပါတယ်။ ဒဲ့ဒီထဲက ရွေးချယ်သတိပြုသင့်တာတွေကတော့ -

- px
- %
- em
- rem
- fr

- တိုဖြစ်ပါတယ်။ px ကို Fixed Unit ခေါ်ပါတယ်။ အရွယ်အစားကို အတိအကျပံ့သေ သတ်မှတ်လိုက်တာပါ။ တစ်လက်မလို ပြောလိုက်ရင် ဘယ်လောက်အရွယ်အစားလဲ မျက်စိတဲ့ တန်းမြင်သလိုပဲ 12px လိုပြောလိုက်ရင် ဘယ်လောက်အရွယ်အစားလဲဆိုတာ မျက်စိတဲ့မှာ မြင်ကြလေ့ရှိပါတယ်။ အားလုံးနဲ့ ရင်းနှီးပြီး သား Unit ပါ။

10px 12px 16px 21px 26px 32px 40px

အရွယ်အစားတွေ သတ်မှတ်ဖို့ Unit တွေရေးတဲ့အခါ တန်ဖိုးနဲ့ ကပ်ရေးရပါတယ်။ ခွာရေးလိုမရပါဘူး။ % နဲ့ em ကိုတော့ Relative Unit လိုခေါ်ပါတယ်။ 200% ဆိုရင် နှစ်ဆဆိုတဲ့ အဓိပ္ပာယ်ပါ။ ဒါကြောင့်မှုလ အရွယ်အစားရဲ့နှစ်ဆ အရွယ်အစားကို ရရှိမှာပါ။ Browser တွေရဲ့ Default Font Size က အများအားဖြင့် 16px ဖြစ်တယ်လို မှတ်နိုင်ပါတယ်။ ဒါကြောင့် Element တစ်ခုအတွက် font-size: 200% လိုပြောရင် နှစ်ဆဖြစ်တဲ့အတွက် 32px အရွယ်အစားကို ရရှိမှာဖြစ်ပါတယ်။ 1em ဆိုရင် စာလုံးတစ်လုံးစာဆိုတဲ့ အဓိပ္ပာယ်ပါ။ ဒါကြောင့် font-size: 2em လိုပြောလိုက်ရင် နှစ်လုံးစာလို ပြောလိုက်တဲ့ သဘောဖြစ်လို font-size: 200% နဲ့ အတူတူပဲလို ပြောမယ်ဆိုရင် ပြောလိုရပါတယ်။

ဒါပေမယ့် width, height လိုဟာမျိုးမှာတော့ ကဲလဲမှုရှိပါတယ်။ width: 100% ဆိုရင် အကျယ်နေရာ ရှိသလောက် အပြည့်ယူမယ်လို ပြောလိုက်တာပါ။ width: 100em ဆိုရင် စာလုံး အလုံး(၁၀၀)စာ နေရာအကျယ်ယူမယ်လို ပြောတာဖြစ်သွားလို သဘောသာဝမတူတော့ပါဘူး။ ဒီ px, % နဲ့ em တို့

ဟာ အသုံးအများဆုံး Unit တွေပါ။ rem နဲ့ fr တို့ကတော့ နောက်မှထပ်တိုးလာတဲ့ Unit တွေဖြစ်ပြီး တစ်ဖြည့်ဖြည့် အသုံးတွင်ကျယ်လာနေပေမယ့် ဒီအဆင့်မှာ လိုတာထက် ပို၍၍သွားမှာစိုးလို့ ချိန်ထားခဲ့ပါ မယ်။ ထုံးစံအတိုင်း ဆက်လက်လေ့လာသင့်တဲ့ အရာတွေမှန်းသိအောင် ထည့်ပြောခဲ့တဲ့ သဘောပါ။

## CSS Color

CSS မှာ အရောင်တန်ဖိုး အမျိုးမျိုးရှိကြပါတယ်။ အသုံးများကြတာတွေကတော့ Color Name, RGB, Hex နဲ့ RGBA တို့ဖြစ်ပါတယ်။ Color Name တွေကိုတော့ ကုဒ်တွေစမ်းရေးတဲ့အခါ အမြန်ရေးထည့်လို့ရတဲ့ အတွက် အသုံးဝင်ပေမယ့်၊ လက်တွေ့အသုံး နည်းပါတယ်။ သုံးလို့ရတဲ့ Color Name စာရင်း အပြည့်အစုံ ကို ဒီမှာကြည့်လို့ရပါတယ်။

- [https://developer.mozilla.org/en-US/docs/Web/CSS/color\\_value](https://developer.mozilla.org/en-US/docs/Web/CSS/color_value)

မှတ်ရလွယ်တာလေးတစ်ချို့ ရွေးမှတ်ချင်ရင် ဒီ Color Name တွေကို မှတ်ထားလို့ရပါတယ်။

black, white, gray, silver, red, green, lime, blue, navy, cyan, yellow, gold, purple, orange, brown, pink, violet

အဲဒါကိုမှ lightblue, darkblue စသဖြင့် ရှေ့က light တို့နဲ့ dark တို့နဲ့ တွဲစမ်းကြည့်နိုင်ပါ တယ်။ အရောင် အားလုံးအတွက် light, dark မူကွဲတွေ မရှိပေမယ့် အများအားဖြင့် ရှိကြပါတယ်။ ဒီ လောက်ဆိုရင်ကိုပဲ Color Name တွေ တော်တော်သုံးလို့ရနေပါပြီ။

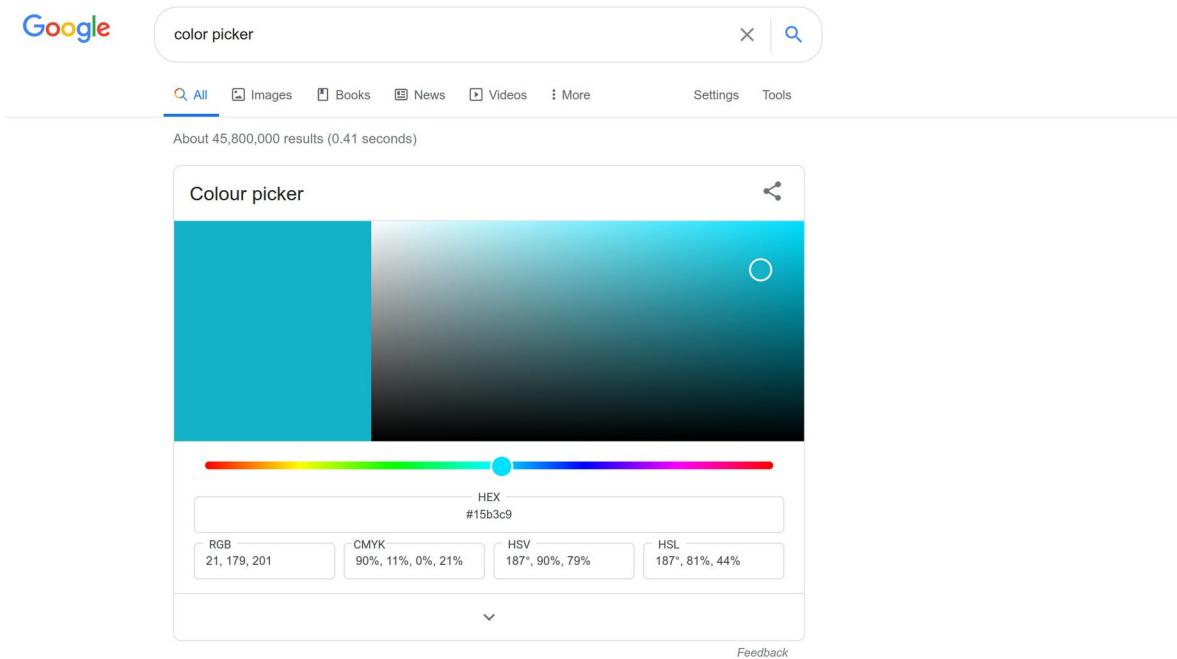
Color Code တွေထက် RGB ရဲ့ ရေးထုံးနဲ့ Hex ရဲ့ ရေးထုံးက ဒီလိုပါ။

### CSS

```
p {
    color: rgb(21, 179, 201);
    background: #C0EEF5;
    border: 5px solid #55E;
}
```

color Property အတွက် rgb() ကိုသုံးထားပါတယ်။ ရေးထုံးအရ rgb ရဲနောက်က ကွင်းစကွင်းပိတ်ထဲမှာ အရောင်ကုဒ် (၃) ခုကို Comma ခံပြီးပေးရတာပါ။ ရွှေဆုံးက အနီရောင်အတွက်၊ အလယ်က အစိမ်းရောင်အတွက်၊ နောက်ဆုံးက အပြောရောင်အတွက် ဖြစ်ပါတယ်။ တန်ဖိုးတွေမှာ အနီမ့်ဆုံးက ၀ ဖြစ်ပြီး အမြင့်ဆုံးက 255 ဖြစ်ပါတယ်။ ဒါကြောင့် rgb(255, 0, 0) ဆိုရင် အနီရောင်ကို ရပါတယ်။ အနီတန်ဖိုး အမြင့်ဆုံး ဖြစ်နေလိုပါ။ နမူနာမှာပေးထားတဲ့ rgb(21, 179, 201) ကတော့ အပြောရောင်ဘက်ကိုပါတဲ့ စိမ်းပြောရောင်ပါ။ အပြောရောင်တန်ဖိုး အမြင့်ဆုံးဖြစ်ပြီး အစိမ်းရောင်တန်ဖိုးလည်း မြင့်လိုပါ။

ခန့်မှန်းလိုရအောင် ပြောပြတာပါ။ လက်တွေ့မှာ ဘယ်ကုဒ်ဆိုရင် ဘာအရောင်လဲဆိုတာ ကိုယ်ဘာသာတွက်နေစရာ မလိုပါဘူး။ Color Code ယူလိုရတဲ့ နည်းပညာတွေမှ အများကြီးပါ။ Google မှာ Color Picker လိုရိုက်ရှာလိုက်ရင်တောင် အရောင်ရွေးလိုရတဲ့လုပ်ဆောင်ချက်ကို တန်းရပါတယ်။ ဒီလိုပါ -



ကိုယ်လိုချင်တဲ့အရောင်ကို ထောက်လိုက်တာနဲ့ Hex နဲ့ RGB သာမက တစ်ခြား Color Code အမျိုးအစားတွေကိုပါ ရရှိမှာဖြစ်ပါတယ်။ Hex Code တွေကိုတော့ ရွှေဆုံးက # သက်တလေးနဲ့စပ်း ရေးပေးရပါတယ်။ သူမှာလည်း (၃) ပိုင်းပါတာပါပဲ။ အနီမ့်ဆုံးက ၀၀ ဖြစ်ပြီး အမြင့်ဆုံးက FF ဖြစ်ပါတယ်။ Hexadecimal Number System ကိုသုံးလို အမြင့်ဆုံးက FF ဖြစ်နေတာပါ။ ဒီနေရာမှာ ကြားဖြတ်ပြီးတော့

Hexadecimal အကြောင်း မပြောတော့ပါဘူး။ အနိမ့်ဆုံးက ၀၀ ဖြစ်ပြီး အမြင့်ဆုံး FF ဖြစ်တယ်လို့သာ မှတ်ထားပါ။ ဒါကြောင့် #FF0000 ဆိုရင် အနိရောင်ရပါတယ်။ ရှုံးဆုံးက အနိတန်ဖိုးအမြင့်ဆုံး ဖြစ်နေလို ပါ။ အတိုကောက်ရေးမယ်ဆိုရင် တန်ဖိုး စုစုပေါင်း (၆) လုံးမဟုတ်ဘဲ (၃) လုံးပဲပေးလို့ရပါတယ်။ #F00 ဆို ရင် #FF0000 နဲ့အတူတူပါပဲ။

RGBA Color ကတော့ RGB ကိုနောက်ဆုံးကနေ Alpha Transparency ပါသွားတာပါ။ RGB ရေးသလိုပဲ ရေးရပါတယ်။ ဒီလိုပါ။

### CSS

```
p {
    background: rgba(255, 0, 0, 0.5);
}
```

နောက်ဆုံးက 0.5 ကတော့ Color ရဲ့ Transparency Level ဖြစ်ပြီးတော့ 0.5 လိုပြောထားတဲ့အတွက် တစ်ဝက်တစ်ပျက် ထွင်းဖောက်မြင်ရတဲ့ အရောင်ကိုရရှိမှာပဲဖြစ်ပါတယ်။

### CSS Properties

ရှုံးပိုင်းမှာ နမူနာ Property တွေကို လိုအပ်သလို ထည့်သွင်းဖော်ပြခဲ့ပေမယ့် ကျွန်ုန်းတော်လည်း ရှိပါ သေးတယ်။ အဲဒီတဲ့က အခြေခံကျိုး အသုံးများတဲ့ Property တွေကို စုစုပေါင်းဖြိုးတော့ ဆက်လက်ဖော်ပြ ပေးပါမယ်။

**background** - background က ရှုံးနမူနာတွေမှာ ပါခဲ့ပြီးသားပါ။ ဒါပေမယ့် Color သတ်မှတ်ပုံပဲ ပါ ခဲ့တာပါ။ Image တွေကိုလည်း Background အနေနဲ့ သုံးချင်ရင်သုံးလို့ရပါတယ်။ ဒီလိုပါ -

### CSS

```
background: url(image/path);
```

url() နဲ့ သုံးချင်တဲ့ Image ရဲ့တည်နေရာကို ပေးရခြင်းဖြစ်ပါတယ်။ url() အစား linear-gradient() ကိုလည်းသုံးနိုင်ပါတယ်။ နှစ်ရောင်စပ်ထားတဲ့ ရောင်ပြီးကိုရပါလိမ့်မယ်။ ဒီလိုပါ -

**CSS**

```
background: linear-gradient(45deg, blue, green);
```

45deg နေရာမှာ 90deg, -45deg စသဖြင့် ကိုယ့်စိတ်တိုင်းက ဒီဂရီပြောင်းပေးလိုရပါတယ်။ သူ့နောက်မှာ တွဲစပ်ချင်တဲ့ အရောင်နှစ်ရောင် လိုက်ရတာပါ။ ဒီတိုင်းပြောနေရတာ သိပ်မမြင်ရင် Codepen ထဲမှာ ဒါလေး လက်တွေ့ရေးပြီး စမ်းကြည့်လိုက်ပါ။

**HTML**

```
<div class="box"></div>
```

**CSS**

```
div {
    width: 600px;
    height: 400px;
    background: linear-gradient(45deg, cyan, green);
    border-radius: 20px;
}
```

ကိုယ့်စိတ်ကူး ကောင်းရင် ကောင်းသလို တွဲစပ်အရောင်ဖော်လိုရတဲ့အတွက် တော်တော်အသုံးဝင်တယ် ဆိုတာကိုတွေ့ရပါလိမ့်မယ်။

**border-radius** - နမူနာတွေမှာ ကြည့်လိုက်ရင် Block တွေအကုန်လုံးက လေးထောင့်စပ်စပ်တွေဆိုတာကို တွေ့ရနိုင်ပါတယ်။ အဲဒီလိုလေးထောင့်စပ်စပ်မဟုတ်ဘဲ ထောင့်ချိုးလေးတွေကို ကျွေးပြီးတော့ပြနေချင်ရင် border-radius ကိုသုံးနိုင်ပါတယ်။ အပေါ်ကနူနာမှာ ထည့်သုံးပြထားခဲ့ပါတယ်။ စမ်းကြည့်ပါ။ Element ရဲ့ width, height နဲ့ border-radius ကို တူအောင်ပေးလိုက်ရင် လေးထောင့် မဟုတ်တော့ဘဲ စက်ပိုင်းပုံစံဖော်ပြတဲ့ Element ကိုရပါလိမ့်မယ်။ ဒါလည်းပဲ လက်တွေ့စမ်းကြည့်သင့်ပါတယ်။

**cursor** - Mouse Pointer ရဲ့အသွင်အပြင်ကို cursor Property နဲ့ ပြောင်းနိုင်ပါတယ်။ Value အနေနဲ့ pointer, wait, crosshair, text, move စသဖြင့်အမျိုးမျိုးပေးလိုရပါတယ်။ pointer ဆိုရင် လက်ညှိုးလေးထောက်ထားတဲ့ပဲ၊ wait ဆိုရင် Hourglass ပဲ၊ move ဆိုရင် လေးဘက်လေးတန်များပြထားတဲ့ပဲ စသဖြင့် ရနိုင်ပါတယ်။

**font-family** - စာတွေဖော်ပြတဲ့အခါ အသုံးပြုဖော်ပြစေလိုတဲ့ဖွန့်ကို သတ်မှတ်ပေးဖို့အတွက် သုံးရတဲ့ Property ဖြစ်ပါတယ်။

### CSS

```
font-family: roboto, helvetica, arial, sans-serif;
```

နှမူနာအရ roboto ဖွန့်ကိုသုံးပြီး စာတွေကိုပြစေချင်တဲ့သဘောပါ။ အကယ်၍ User ရဲ့ Device မှာ roboto ဖွန့်မရှိရင် helvetica ကို သုံးပေးပါ။ မရှိရင် arial ကိုသုံးပေးပါ။ မရှိရင် sans-serif ကို သုံးပေးပါလို့ အဆင့်ဆင့် သတ်မှတ်ပေးထားလိုက်တာပါ။

**@font-face** - User ရဲ့ Device မှာ ဖွန့်မရှိလို့ ဖော်ပြစေလိုတဲ့ပုံစံ မပေါ်ဘူးဆိုတာမျိုး မဖြစ်စေဖို့ အတွက် ဒီသတ်မှတ်ချက်နဲ့ ဖွန့်ဖိုင်ကို ချိတ်ထားပေးလို့ရပါတယ်။ Property တွေကြားထဲမှာ ထည့်ပြောထားပေမယ့် ဒါက Property မဟုတ်ပါဘူး။ ဒီလိုရေးရပါတယ်။

### CSS

```
@font-face {
    font-family: roboto;
    src: url(path/to/roboto.font);
}
```

နှမူနာအရ roboto ဖွန့်ဖိုင်ရဲ့တည်နေရာကို url() နဲ့ပေးထားပါတယ်။ ဒါကြောင့် Download လုပ်ပြီး သုံးပေးသွားမှာဖြစ်လို့ User ရဲ့ Device ထဲမှာ အဲဒီဖွန့် ရှိရှိ မရှိရှိ အဆင်ပြေသွားမှာ ဖြစ်ပါတယ်။ ဒီကုဒ်က ရေးထုံးကို နှမူနာပြတာပါ။ ဒီအတိုင်းရေးလို့ မပြည့်စုံသေးပါဘူး။ လက်တွေ့မှာ ဖွန့်ဖိုင်တစ်ခု မှန်မှန်ကန်ကန် အလုပ်လုပ်ဖို့ဆိုရင် Format မှန်ဖို့လည်း လိုပါသေးတယ်။ .ttf, .otf, .woff, .eot စဲ သဖြင့် ဖွန့် Format အမျိုးမျိုးရှိပါတယ်။ ဒါကြောင့် ရေးထုံးကိုပဲ မှတ်ထားပါ။ တစ်ကယ် အလုပ်လုပ်ဖို့ အတွက်တော့ ကိုယ်တိုင်ရေးစရာမလိုပါဘူး။ Google Fonts လိုနေရာမျိုးကနေ သွားယဉ်လိုက်ရင်ရပါတယ်။ နည်းနည်းဆက်လေ့လာကြည့်လိုက်ပါ။ သိပ်မခက်ပါဘူး။

- <https://fonts.google.com/>

**@import** - ဒါလည်းပဲ လိုအပ်လို ထည့်ပြေတာပါ။ Property တော့ မဟုတ်ပါဘူး။ HTML Document ကနေ CSS ဖိုင်ကို <link> Element သုံးပြီးချိတ်ရပါတယ်။ CSS ဖိုင်ကနေ အခြား CSS ဖိုင်ကို ချိတ်ချင်ရင်တော့ @import ကို သုံးပြီးချိတ်လိုရပါတယ်။ ဒီလိုပါ -

**CSS**

```
@import url(path/to/css.file)
```

ချိတ်ချင်တဲ့ CSS ဖိုင်ရဲ့ တည်နေရာကို url() သုံးပြီးချိတ်ရတာပါ။

**list-style** - <ul> တွေ <ol> တွေမှာဖော်ပြတဲ့ Bullet တွေ Number တွေကို list-style နဲ့ ပြောင်းလိုရပါတယ်။ ဒီလိုပါ -

**CSS**

```
ul {
    list-style: square;
}
```

Bullet Value အနေနဲ့ square, circle, disc တို့ကို အသုံးများပါတယ်။ Number Value အနေနဲ့ lower-alpha, upper-alpha, lower-roman, upper-roman တို့ကိုအသုံးများပါတယ်။

**text-align** - စာတွေကို left, right, center, justify စသဖြင့် Value တွေနဲ့ လိုသလို စီမြေးပြလို ရပါတယ်။

**CSS**

```
p {
    text-align: center;
}
```

**text-decoration** - စာတွေကို Underline တားဖို့ (သို့မဟုတ်) ဖျက်ထားသကဲ့သို့ ကန်လတ်ဖြတ်လိုင်းထည့်ဖို့ သုံးပါတယ်။ Underline တွေ၊ လိုင်းဖြတ်တွေ ပြန်ဖြတ်ချင်ရင်လည်း သုံးလိုရပါတယ်။

**CSS**

```
a {
    text-decoration: none;
}
```

နမူနာအရ <a> Element ခဲ့ Underline ကိုဖြတ်လိုက်တာပါ။ Underline ထည့်ချင်ရင် underline Value ကိုသုံးနိုင်ပြီး လိုင်းဖြတ်ထည့်ချင်ရင် line-through Value ကိုသုံးနိုင်ပါတယ်။

**line-height, letter-spacing, word-spacing** - စာကြောင်းတစ်ခုကြောင်းနဲ့ တစ်ကြောင်းကြား အကွာအဝေး၊ စာလုံးတစ်လုံးနဲ့တစ်လုံးကြားက အကွာအဝေးသတ်မှတ်ဖို့နဲ့ Word တစ်ခုနဲ့ တစ်ခုကြား အကွာအဝေးသတ်မှတ်ဖို့အတွက် သုံးနိုင်ပါတယ်။

**CSS**

```
p {
    line-height: 2em;
    letter-spacing: 2px;
    word-spacing: 5px;
}
```

နမူနာအရ line-height ကို 2em လိုပြောထားတဲ့အတွက် စာကြောင်းတွေကို နှစ်ဆွဲပြီးပြပေးမှာပါ။ လိုင်းတွေ အရမ်းကျွေားပါလိမ့်မယ်၊ စာတွေဖတ်လို သိပ်ကောင်းမှာမဟုတ်ပါဘူး။ နမူနာအနေနဲ့ ပေးထားတာပါ။ အဲဒီလို line-height မထည့်ဘဲ သူ Default အတိုင်းကလည်း သိပ်အဆင်မပြေပါဘူး။ စာကြောင်းတွေ တစ်ခုကြောင်းနဲ့တစ်ခုကြောင်း ကပ်လွန်းပါတယ်။ အင်းလိပ်စာတွေအတွက် အသင့်တော်ဆုံးလို ပြောလိုရတဲ့ line-height ပမာဏကတော့ 1.5em ဖြစ်ပါတယ်။ မြန်မာစာတွေအတွက်ကတော့ သုံးထားတဲ့ ဖွန့်ပေါ်မှုတည်လို ဖတ်လိုကောင်းလောက်မယ့် အကွာအဝေးပမာဏကို အမျိုးမျိုးပြောင်းစမ်းပြီး သင့်တော်တဲ့ပမာဏကို သတ်မှတ်ပေးဖို့ လိုပါလိမ့်မယ်။

**CSS Comments**

နောက်ဆုံးတစ်ချက်အနေနဲ့ CSS တွေမှာ ကိုယ်ဘာသာရေးမှတ်ချင်တာတွေရှိရင် /\* နဲ့ \*/ ကြားထဲမှာ Comment တွေကို ရေးမှတ်နိုင်တယ်ဆိုတာလေး ထည့်မှတ်ပါ။ အလုပ်လုပ်တဲ့အခါ အဲဒီ Comment တွေကို ထည့်အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။

**CSS**

```
/* Some CSS Comments */

.menu {

    /* Another comments */

    color: white;
    background: purple;    /* Some more comments */
}
```

တစ်ကယ်တော့ CSS ဟာ အရမ်းကြီးမခက်ပေမယ့် သူ့ဟာနဲ့သူ တော်တော်လေး ကျယ်ပြန့်တဲ့ဘာသာရပ် တစ်ခုပါ။ Text Effect တွေ Image Effect တွေလည်း အများကြီးရှိပါသေးတယ်။ စကေးချုပ်ချွဲနဲ့ 3D အသွင်အပြင်လို ကိစ္စမျိုးတွေထိ CSS နဲ့စီမံလိုပါတယ်။ Animation တွေလည်း ဖန်တီးအသုံးချဝို့ ရပါ တယ်။ Grid Layout တွေဖန်တီးလို ရပါတယ်။ အခုဖော်ပြခဲ့တဲ့ အခြေခံတွေကိုသာ သေချာရအောင်လုပ် ထားပါ။ အချိန်တန်လို ဆက်လေ့လာတဲ့အခါ အဆင်ပြေသွားပါလိမ့်မယ်။

တစ်ကယ်တော့ Style တွေ အကုန်လုံးကို ကိုယ်တိုင်လုပ်စရာတော့ မလိုပါဘူး။ ဒီစာအုပ်ရဲ့ ရည်ရွယ်ချက် ကိုက Bootstrap လိုနည်းပညာမျိုးကပေးတဲ့ အသင့်သုံးလိုရတဲ့ လုပ်ဆောင်ချက်တွေကို သုံးတတ်စေဖို့ ဖြစ်ပါတယ်။ ဒါပေမယ့် အသင့်သုံးလိုရတာပဲ သုံးတတ်ပြီး ကိုယ့်ဘာသာ မလုပ်တတ်တော့ဘူးဆိုရင် ရေရှည်မှာ အဆင်ပြမှာမဟုတ်လို အခုလိုသိသင့်တဲ့ အခြေခံတွေကို ကြော်အောင် အရင်ပြောနေခြင်း ဖြစ်ပါတယ်။ ဆက်လက်ပြီးတော့ Bootstrap အကြောင်းကို ဖော်ပြပါတော့မယ်။

## အခန်း (၃) – Bootstrap Intro

Bootstrap CSS Framework ဟာ Web Design နဲ့ Web Development လောကကို ကိုင်လှပ်ပြီး တစ်ခေတ်ဆန်းသွားစေခဲ့တဲ့ နည်းပညာတစ်ခုပါ။ Bootstrap မတိုင်ခင်က Web Designer တွေ Web Developer တွေဟာ HTML, CSS, JavaScript တွေကို ကိုယ်တိုင် ချရေးပြီးတော့ ကိုယ့်စိတ်ကူးဥပဒေရှုရင် ရှိသလို ဝဘ်ဆိုက်တွေကို ဖန်တီးခဲ့ကြပါတယ်။ Bootstrap ထွက်ပေါ်လာပြီး နောက်မှာတော့ ကိုယ်တိုင် အကုန်လုပ်စရာ မလိုတော့ဘဲ Bootstrap က ပေးထားတဲ့ အသင့်သုံး Components နဲ့ Layouts လုပ်ဆောင်ချက်တွေကို အသုံးချပြီးတော့ ဖန်တီးနိုင်လာကြပါတယ်။ ကိုယ့်စိတ်ကူးဥပဒေနဲ့ တိတွင်ချင်ရင် လည်း လုံးဝကွဲပြား ဆန်းပြားတဲ့ ဖန်တီးမှုတွေထက် Bootstrap ကို အခြေခြားထားတဲ့ ဖန်တီးမှုတွေကို ပိုပြီး တော့ လုပ်လာကြပါတယ်။

Web Developer တစ်ဦးအနေနဲ့ Bootstrap ပေါ်ခါစက မကြိုက်ပါဘူး။ ဘာဖြစ်လိုလဲဆိုတော့ ဒီလိုပါ။

တစ်ခြား Software အမျိုးအစားတွေနဲ့ယူဉ် Web ရဲ့ အားသာချက်ကတော့ HTML, CSS, JavaScript ရဲ့ အကူအညီနဲ့ အကန့်အသတ်ဘောင် တော်တော်နည်းပြီး စိတ်ကူးဥပဒေရှုရင်ရှိသလောက် ထူးခြားဆန်းပြား တဲ့ User Interface တွေဖန်တီးနိုင်ခြင်းပဲ ဖြစ်ပါတယ်။ ၂၀၀၅-၂၀၁၀ ကြားကာလဟာ Web ရဲ့ ရွှေခေတ်ပါပဲ။ Web Designer တွေ Web Developer တွေ အပြိုင်အဆိုင် လက်စွမ်းတွေပြုပြီး တိတွင်လိုက်ကြတာမှ အပြိုင်အဆိုင်ပါပဲ။ ဒါပေမယ့် အဲဒီအားသာချက်ကပဲ Web ရဲ့ အားနည်းချက်လည်း ဖြစ်နေပြန်ပါတယ်။ စိတ်ကူးရှိသလို ဖန်တီးလိုရတာ မှန်ပေမယ့်၊ အကုန်ကိုယ့်ဘာသာ ဖန်တီးနေရပါတယ်။ ထပ်ခါထပ်ခါ ပြန်ပြီးတော့ ဖန်တီးရပါတယ်။ တစ်ယောက်ကို တစ်မျိုးစီ ထွင်ကြတော့၊ သုံးတဲ့ User က၊ ဟိုနေရာမှာတစ်မျိုး ဒီနေရာမှာတစ်မျိုး၊ မျက်စီတွေ လည်ကြပါတယ်။

Bootstrap ပေါ်လာတဲ့အခါ သူမှာ User Interface တွေတည်ဆောက်ဖို့ အသင့်သုံးနိုင်တဲ့ Layouts တွေ

Components တွေ ပါဝင်လာပါတယ်။ တော်တော်လေး အဆင်ပြေတဲ့အတွက် အချိန်တိုအတွင်း လူကြိုက် များပြီး လူသုံးများသွားပါတယ်။ ဒီတော့ ဟိုဝဘ်ဆိုက် ကြည့်လိုက်လဲ ဒီပုံစံ၊ ပုံစံတူတွေ များလာတော့တာပါပဲ။ အဲဒါကို မကြိုက်ခဲ့တာပါ။ မူလ Web နည်းပညာရဲ့ လွှဲလွှဲလပ်လပ် လပ် ဖန်တီးတဲ့အလေ့အကျင့်တွေ တစ်ဖြည်းဖြည်း နည်းပါးပျောက်ကွယ်သွားပြီလို့ မြင်ခဲ့တာပါ။

အချိန်ကာလတစ်ခု ရောက်လာတော့မှ အဲဒါကသာလျှင် ပိုကောင်းတဲ့နည်းဆိုတာကို သိလာခဲ့ရတာပါ။ အသုံးပြုသူ User က ရှုချင်စွေယ်၊ သုံးချင်စွေယ်ဖြစ်တာကို လိုချင်ပေမယ့် အလွန်အမင်း ဆန်းပြားတာကို တော့မလိုချင်ပါဘူး။ Consistence ဖြစ်တာကို လိုချင်တာပါ။ သုံးရလွယ်ကူတာကို လိုချင်တာပါ။ Bootstrap ထွက်ပေါ်လာပြီးနောက်မှာတော့ Menu ရဲဖွဲ့စည်းပုံ၊ ပါဝင်တဲ့ Component တွေရဲ့ ဖွဲ့စည်းပုံ၊ Layout ရဲဖွဲ့စည်းပုံ၊ ဒါတွေဟာ ခပ်ဆင်ဆင်တွေ ဖြစ်လာတော့ User အတွက်က တော်တော် အဆင်ပြေပါတယ်။ ဘယ်သွားသွား အရောင်အသွေးနဲ့ ဖွဲ့စည်းပုံသာ ကွဲပြားသွားမယ်၊ အသုံးပြုနည်းက အသစ်အဆန်း မဟုတ်တော့ဘဲ သိရှိကျမ်းဝင်ပြီး ဖြစ်တဲ့ ပုံစံကိုသာ ရရှိမှာဖြစ်ပါတယ်။

ဒီသဘောသဘာဝကြောင့်ပဲ Bootstrap ဟာ လက်ရှိမှာလူသုံးအများဆုံး နည်းပညာတစ်ခုဖြစ်နေတာပါ။ Bootstrap နဲ့ အပြိုင် Foundation လို အလားတူ နည်းပညာတွေ ရှိသေးပေမယ့် Bootstrap ကသာလျှင် အဓိကနည်းပညာ ဖြစ်လာခဲ့ပါတယ်။ အခုနောက်ပိုင်းမှာ Bootstrap နဲ့ ရည်ရွယ်ချက်တူပေမယ့် သဘောသဘာချင်း မတူတော့တဲ့ Tailwind လိုပေါ်တဲ့ နည်းပညာတစ်ခု ခေတ်စားစ ပြုနေပါတယ်။ Bootstrap ကို ကျဉ်းဖြတ်ပြီး အဓိကနည်းပညာနေရာကို ယူသွားမလားဆိုတာတော့ ပြောဖို့စောပါသေးတယ်။ စောင့်ကြည့်ကြရနိုင်းမှာပါ။

Bootstrap ဟာ လေ့လာရလွယ်ကူတဲ့နည်းပညာတစ်ခုပါ။ HTML/CSS အခြေခံရှိသူ မည်သူမဆို ကိုယ့်ဘာသာ လေ့လာအသုံးချလို့ ရနိုင်ပါတယ်။ Bootstrap Documentation ဟာ ရှင်းလင်းပြီး အစီအစဉ်ကျတဲ့အတွက် ဒီ Documentation နဲ့တင် လေ့လာသူတွေအတွက် တော်တော်အဆင်ပြေနေပါပြီ။

- <https://getbootstrap.com/docs>

ဒါပေမယ့် Documentation ဆိုတာ ရှိသမျှအကုန်ပါအောင် ဖော်ပြုရပါတယ်။ သင်ယူလေ့လာ ဖို့ထက်၊ လက်တွေအသုံးချချိန် လိုအပ်လာတဲ့အခါ ကိုးကားဖို့အတွက် ပိုပြီးတော့သင့်တော်ပါတယ်။ သင်ယူလေ့လာ

နေတဲ့အချိန်မှာတော့ လိုလိုမလိုလို ရှိသမျှအကုန်ကြည့်တယ် ဆိုတဲ့နည်းဟာ ထိရောက်တဲ့လေ့လာမှု မဟုတ်ဘူးလို့ ဆိုချင်ပါတယ်။ အစပိုင်းမှာ အခြေခံကျတဲ့ သဘောသဘာဝတွေကို နားလည်အောင်လုပ်၊ ပြီး တဲ့အခါ အသုံးများတဲ့ လုပ်ဆောင်ချက်တွေကို ဦးစားပေးပြီး ရွေးချယ်လေ့လာရတာပါ။ အခြေခံသဘော သဘာဝကို ကောင်းကောင်းနားလည်ရင် အသုံးနည်းတဲ့ ကိစ္စတွေက ချက်ခြင်းလုပ်စရာ မလိုပါဘူး။ လိုအပ် လာတော့မှ ကြည့်လိုက်လို့ ရနိုင်ပါတယ်။ အဆက်မပြတ် လေ့လာရင်းအသုံးချသွားတဲ့ Continuous Learning စနစ် ဆိုပါတော့။ ဒီနည်းနဲ့သာ ကနေ့ခေတ်လို လေ့လာစရာတွေ မဆုံးနိုင်အောင် များလှတဲ့ အခြေအနေမှာ ထိထိရောက်ရောက် လေ့လာအသုံးချနိုင်မှာပါ။ ဒါကြောင့် ဒီစာအပ်မှာ HTML/CSS လို အခြေခံသဘောသဘာဝတွေကို အရင်ဗြီးစားပေးဖော်ပြခဲ့ပြီး၊ နောက်တစ်ဆင့်အနေနဲ့ Bootstrap ရဲ့ အရေးကြီးပြီး အသုံးများမယ့် အကြောင်းအရာတွေကို ရွေးထုတ်ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။

အပေါ်မှာပေးထားတဲ့ Link ကနေ Bootstrap Documentation ကိုသွားကြည့်လိုက်မယ်ဆိုရင် တွေ့ရမယ့် အခေါ်အဝေါ် အသုံးအနှံးလေးတွေ ရှိပါတယ်။ ဒီအသုံးအနှံးလေးတွေက သင့်တင့်တဲ့အတွေ့အကြုံ ရှိထားပြီး Web Developer တစ်ယောက်အတွက် အထူးအဆန်း မဟုတ်ပေမယ့်၊ အခုံမှစလေ့လာမယ့် သူအတွက် တော့ အထူးအဆန်း ဖြစ်နေနိုင်ပါတယ်။ အခြေခံသဘောသဘာဝကို အရင်နားလည်အောင် လုပ်ရမယ်ဆို တဲ့ လေ့လာမှုလမ်းစဉ်နဲ့အညီ Bootstrap ကိုလက်တွေ့မလေ့လာခင် ဒီအခြေခံအသုံးအနှံးလေးတွေကို အ ရင်ကြိုးပြီး ရှင်းပြထားချင်ပါတယ်။

## CSS Reset

ပြီးခဲ့တဲ့အခန်းမှာ CSS အကြောင်းပြောတုံးက Browser Default Style ဆိုတဲ့ အသုံးအနှံးတစ်ခု ပါသွားခဲ့ပါ တယ်။ <h1> ဆိုရင် စာလုံးကြီးကြီးနဲ့ ပြပေးတယ်။ <p> ဆိုရင် တစ်ခုနဲ့တစ်ခု နည်းနည်းစွာပြီး ပြပေး တယ်။ <ul><li> ဆိုရင် Bullet စာရင်းနဲ့ပြပေးတယ်။ <a> ဆိုရင် စာလုံးအပြောရောင်နဲ့ Underline တားပြီးပြပေးတယ်။ စသဖြင့် လုပ်ဆောင်ချက်တွေကို Browser Default Style လိုက်တော့ဘူး။ ကိုယ်ရေး ပေးထားတာ မဟုတ်ဘဲ Browser တွေက မူလကာတည်းက သတ်မှတ်ထားတဲ့ Default Style တွေ ဖြစ်ပါ တယ်။

ပြဿနာက၊ အဲဒီ Default Style တွေဟာ တစ်ကယ်တမ်း အသုံးမဝင်ခြင်း ဖြစ်ပါတယ်။ သူကသာ Style လုပ်ပေးထားတာ၊ ပေးထားတဲ့အတိုင်း သုံးလို့လည်း အဆင်မပြပါဘူး။ အဆင်ပြေတဲ့ပုံစံဖြစ်အောင် ပြန် ပြင်ပြီး အမြတမ်းရေးရတာပဲ။ ပြီးတော့ Browser တစ်ခုနဲ့တစ်ခု အဲဒီ Default Style တွေက မတူကြပ်နဲ့

ဘူး။ နည်းနည်းကဲ့ကြပြန်တယ်။ ဒီတော့ တစ်ချို့ Element တွေရဲ့ ဖော်ပြပုံရလဒ်က Browser တစ်ခုမှာပုံစံတစ်မျိုး၊ နောက် Browser ကျတော့ နောက်ပုံစံတစ်မျိုး ဖြစ်နေတတ်ပါတယ်။ အများကြီးကွာသွားတာတွေရှိသလို၊ နည်းနည်းလေး မသိမသာ ကွာတာတွေလည်း ရှိနေပါတယ်။

CSS Reset ဆိုတာ အဲဒီလို အသုံးမဝင်တဲ့အပြင် Browser တစ်ခုနဲ့တစ်ခု မတူဘဲကဲ့ပြားနေတဲ့ Default Style တွေကို ဖယ်ထုတ်ပစ်လိုက်တာပါ။ ကိုယ့်ဘာသာ ရေးလိုရသလို၊ အသင့်ရေးထားပြီးသား Reset ကုဒ်တွေကို ယူသုံးလိုလည်း ရနိုင်ပါတယ်။ ကိုယ်ဘာသာ ရေးမယ်ဆိုရင် ဥပမာက ဒီလိုပါ -

```

HTML
1▼ <h1>Main Heading</h1>
2▼ <h2>Sub Heading</h2>
3
4▼ <ul>
5▼   <li>Item One</li>
6▼   <li>Item Two</li>
7   </ul>
8

CSS
1▼ h1, h2 {
2    font-size: inherit;
3    font-weight: normal;
4    margin: 0;
5    padding: 0;
6 }
7
8▼ ul {
9    list-style: none;
10   margin: 0;
11   padding: 0;
12 }

JS

```

HTML ထဲမှာ <h1><h2><ul><li> Element တွေထည့်ထားပေးမယ့် ရလဒ်ကိုကြည့်လိုက်ရင် ဘာ Style မှမပါတော့ဘဲ ရှိုးရှိုးစာတွေလို တန်းစီပြီး ပြနေတာကို တွေ့ရမှာပါ။ CSS နဲ့ပဲ <h1><h2> ရဲ့ font-weight တန်ဖိုးကို normal လိုသတ်မှတ်ပြီး နို့ပါနေတဲ့ Bold ကို ဖြုတ်လိုက်ပါတယ်။ <ul> အတွက် list-style တန်ဖိုး none လိုပြောပြီး နို့ပါနေတဲ့ Bullet တွေကို ဖြုတ်လိုက်ပါတယ်။ နို့ပါနေတဲ့ margin, padding တွေ အကျန် ဖြုတ်လိုက်ပါတယ်။ ဒါကြောင့် Default Style တွေ အကျန် ပြုတ်သွားလို အားလုံးကို ရှိုးရှိုးစာတွေလို တန်းစီပြီးပြနေတာပါ။ ဒီသဘောကို CSS Reset လိုပေါ်ခြင်း ဖြစ်ပါတယ်။

ဒါ Reset ကုဒ်မျိုးကို ကိုယ့်ဘာသာ ရေးစရာမလိုပါဘူး။ အသင့်ရေးပြီးသားတွေ ရှိပါတယ်။ Element အားလုံးအတွက် တစ်ခုမကျန် ကိုယ့်ဘာသာရေးရင် စုံမှာမဟုတ်ပါဘူး။ ဒါကြောင့်လိုအပ်ရင် အထပ်ထပ် စမ်းထားပြီးသား ပြည့်စုံတဲ့ အသင့်သုံး Reset ကုဒ်တွေကို အသုံးပြုသင့်ပါတယ်။ Bootstrap ကတော့ အရင် Version အဟောင်းတွေမှာ normalize.css လိုပေါ်တဲ့ နည်းပညာကိုသုံးပါတယ်။ သူလည်း Reset တစ်မျိုးပါပဲ။ ဒါပေမယ့် သူကတော့ ရှိသမျှ Default Style တွေကို အကုန်ဖြေစုံမပြစ်ဘဲ၊ <h1> ဆိုရင် ခေါင်းစီးနဲ့တူအောင် ခိုက်ခိုက်ပြမယ်။ <ul> ဆို Bullet နဲ့ပဲ ဆက်လက်ဖော်ပြပေးသွားမှာ ဖြစ်ပါတယ်။ သူလုပ်ပေးမှာက Browser မတူလို ကဲပြားနေတတ်တဲ့ အသင်အပြင်တွေကို ညီသွားအောင် ညိုပေးလိုက်မှာပါ။ ဒါကြောင့် အကျဉ်းချုပ်အနေနဲ့ Reset နဲ့ Normalize ဆိုပြီး နှစ်မျိုးရှိပါတယ်။ Reset က Default Style တွေ အကုန်ရှင်းပြစ်ပြီး Normalize ကတော့ Default Style တွေကို ညီအောင် ညိုပေးလိုက်တယ်လို မှတ်နိုင်ပါတယ်။ Bootstrap က နောက်ပိုင်း Version တွေမှာ Reboot လိုအမည်ပေးထားတဲ့ normalize.css နဲ့ သဘောသဘာဝ ဆင်တူတဲ့ နည်းစနစ်ကို သုံးပါတယ်။

## Vendor Prefix

CSS နဲ့လုပ်လိုရတာတွေအများကြီးပါ။ ပြီးခဲ့တဲ့အခန်းမှာလည်း ပြောခဲ့ပါတယ်။ ဒီလုပ်ဆောင်ချက်တွေမှာ အဆင့်အမျိုးမျိုးရှိပါတယ်။ ဆွေးနွေးဆဲအဆင့်၊ စမ်းသပ်ဆဲအဆင့်၊ လက်တွေ့အသုံးချုအဆင့် စသာဖြင့် အဆင့်ဆင့် ရှိကြပါတယ်။ ဒီစာအုပ်မှာ ဖော်ပြထားတာတွေ အားလုံးက လက်တွေ့အသုံးချုအဆင့် လုပ်ဆောင်ချက်တွေ ဖြစ်ပါတယ်။ တစ်ချို့ စမ်းသပ်ဆဲအဆင့် CSS ရဲ့ လုပ်ဆောင်ချက်တွေကို ထည့်သွင်းအသုံးပြုလိုရင် ဒီအတိုင်းသုံးလို မရပါဘူး။ Vendor Prefix လိုပေါ်တဲ့ ရေးထုံးတစ်မျိုးကို အသုံးပြုရလေ့ ရှိပါတယ်။ ဥပမာ - ဒီလိုကုဒ်မျိုးကို ရုပ်နံပါတယ်။

### CSS

```
p {
    background: yellow;
    background: linear-gradient(90deg, yellow, green);
    background: -moz-linear-gradient(90deg, yellow, green);
    background: -webkit-gradient(linear, top, yellow, green);
    background: -o-linear-gradient(90deg, yellow, green);
    background: -ms-linear-gradient(90deg, yellow, green);
}
```

ဒါကတော့ Linear Gradient လုပ်ဆောင်ချက်ကို စမ်းသပ်ဆဲအဆင့်မှာ ရေးခဲ့ကြရတဲ့ကုဒ်ပါ (အခုတော့

လက်တွေအသုံးချလို ရန်ပြီ့မြဲလို ဒါမျိုးတွေ မလိုတော့ပါဘူး)။ background တစ်ခုထကိုပဲ (၆) ခါရေးထားရပါတယ်။ ပထမဆုံးတစ်ခုမှာ Background ကို ရိုးရိုး Color အနေနဲ့ သတ်မှတ်ထားပါတယ်။ ဒါကြောင့် ဆက်လက်ရေးသားတဲ့ စမ်းသပ်အဆင့်ကုဒ်တွေကို Browser က နားမလည်ရင်လည်း ကိစ္စမရှိပါဘူး၊ ဒါ Color ကိုပဲအသုံးပြု အလုပ်လုပ်သွားမှာပါ။ ဒီနည်းကို Fallback Style လို ခေါ်ပါတယ်။ သုံးချင်တာက ဆက်လက်ရေးသားထားတဲ့ စမ်းသပ်အဆင့် ကုဒ်တွေဖြစ်ပြီး၊ အကယ်၍ အဆင်မပြောင် ရိုးရိုး Color ကိုပဲသုံးမယ်ဆိုတဲ့ သဘောမျိုး ဖြစ်သွားပါတယ်။

Linear Gradient အတွက် တစ်ကယ်ရေးနည်းအမှန်က linear-gradient() ဖြစ်ပါတယ်။ ဒါကို တစ်ချို့ Browser တွေက လိုက်နာတယ်။ တစ်ချို့ Browser တွေကပိုကောင်းတယ်ထင်တဲ့ နည်းကိုသုံးတယ်။ ဒါကြောင့် Browser တစ်ခုနဲ့တစ်ခု ဒီအဆင့်မှာ အလုပ်လုပ်ပုံမတူကြပါဘူး။ ဒါကြောင့်သုံးချင်ရင် သက်ဆိုင်ရာ Browser က သတ်မှတ်ထားတဲ့ Prefix တွေ ရွှေ့ကခံပြီး ရေးပေးရပါတယ်။ တစ်ခြား Browser တွေမှာ ဒီကုဒ်က အလုပ်လုပ်မှာ မဟုတ်ဘူးဆိုတာကို ပေါ်လွင်သွားအောင်လိုပါ။

-moz- Prefix နဲ့စတဲ့လုပ်ဆောင်ချက်တွေက Mozilla Firefox Browser အတွက်ပါ။ -webkit- Prefix နဲ့စတဲ့ လုပ်ဆောင်ချက်တွေကတော့ Google Chrome နဲ့ Apple Safari Browser တွေ အတွက်ပါ။ ဒီနှစ်ခုက Browser အနေနဲ့ မတူပေမယ့် သုံးထားတဲ့ Rendering Engine ခေါ် HTML/CSS ကုဒ်တွေပေါ်မှာ အခြေခံပြီး သင့်တော်တဲ့ရလဒ်ကို ဖော်ပြပေးတဲ့ နည်းပညာက အတူတူပဲမြို့လိုပါ။ ဒီအကြောင်းကို ဒီနေရာမှာ အကျယ်မချွဲတော့ပါဘူး။ တစ်ချို့ Browser တွေက Browser သာမတူတာ၊ Rendering Engine တူကြတယ်လို အကျဉ်းချုပ် မှတ်နိုင်ပါတယ်။ -o- Prefix က Opera Browser အတွက်ဖြစ်ပြီးတော့ -ms- Prefix ကတော့ Microsoft Internet Explorer အတွက်ပါ။

အခုနောက်ပိုင်းမှာ Google Chrome, Apple Safari, Opera, Brave, Microsoft Edge စတဲ့ Browser တွေ အားလုံးက သုံးထားတဲ့ Rendering Engine တူကြပါတယ်။ မတူတာဆိုလို Major Browser ထဲမှာ Firefox တစ်ခုပဲ ကျွန်တော့တယ်လိုတောင် ဆိုနိုင်ပါတယ်။ Vendor Prefix တွေကိုလည်း အသုံးတော့ နည်းလာကြပါပြီ။ ဒါပေမယ့် အချို့နေရာတွေမှာ ဆက်သုံးနေကြရဆဲပါပဲ။ Vendor Prefix ဆိုတဲ့အသုံးအနှစ်းက ဘာကို ဆိုလိုတာလဲဆိုတာကို သိစေဖို့နဲ့ ရုံဖွံ့ဖြိုးပဲ။ Vendor Prefix တွေသုံးပြီး ရေးထားတဲ့ကုန်တွေကို တွေ့တဲ့အခါသူတို့ရဲ့အမိပါယ်ကို သိရှိစေဖို့အတွက် ထည့်သွင်းဖော်ပြခြင်း ဖြစ်ပါတယ်။

## Preprocessor

ပြီးခဲ့တဲ့အခန်းမှာလည်း ပြောခဲ့ပါတယ်။ Web Document တွေ တည်ဆောက်ဖို့အတွက် Style Language မူကဲတွေ နှစ်မျိုးသုံးမျိုး မရှိဘဲ CSS တစ်မျိုးတည်းသာ ရှိပါတယ်။ Language အနေနဲ့ မူကဲမရှိပေမယ့် LESS လိုခေါ်တဲ့ နည်းပညာနဲ့ SASS လိုခေါ်တဲ့ Preprocessor နည်းပညာတွေတော့ ရှိပါတယ်။ Bootstrap က အရင် Version အဟောင်းတွေမှာ LESS ကိုသုံးပြီး နောက်ပိုင်း Version တွေမှာ SASS ကိုသုံးပါတယ်။

ဒီနည်းပညာတွေက CSS မှာ မူလကမပါတဲ့ ရေးထုံးတွေကို ဖြည့်စွက်ပေးထားကြပါတယ်။ ဥပမာ - LESS ကိုအသုံးပြုရေးသားထားတဲ့ ဒီကုဒ်ကိုလေ့လာကြည့်ပါ။

### LESS

```
@primary: blue;

button {
    background: @primary;
}

a {
    color: @primary;
}
```

@primary ဆိုတဲ့ Variable တစ်ခုနဲ့ blue ဆိုတဲ့ Color Value ကို သတ်မှတ်ပေး ထားလိုက်တာပါ။ ဒါ ကြောင့် နောက်ပိုင်း လိုအပ်တဲ့နေရာမှာ ပြန်သုံးလို့ရသွားပါတယ်။ နူးနာအရ button ရဲ့ background နဲ့ a ရဲ့ color တို့ဟာ blue ဖြစ်သွားမှာပါ။ blue လိုတိုက်ရှိက်မပေးတော့ဘဲ blue တန်ဖိုးရှိနေတဲ့ @primary ကိုပေးလိုက်တာပါ။ ဒီနည်းကပေးတဲ့ အားသာချက်ကတော့၊ အကြောင်းအမျိုးမျိုးကြောင့် စိတ်ကူးပြောင်းပြီး အရောင် blue ကို မသုံးချင်တော့ဘူး purple ပြောင်းသုံးချင်တယ်ဆိုရင် တစ်ခုချင်း လိုက်ပြင်ဖို့ မလိုတော့ပဲ @primary ရဲ့ တန်ဖိုးကို purple လို့ ပြောင်းပေးလိုက်ယုံပါပဲ။ @primary ကို သုံးထားသမျှ နေရာအားလုံး purple ဖြစ်သွားမှာပါ။ ဒီလိုပါ -

**LESS**

```
@primary: purple;

button {
  background: @primary;
}

a {
  color: @primary;
}
```

SASS နဲ့ဆိုရင် ဒီလိုရေးရပါတယ်။

**SASS**

```
$primary: blue;

button {
  background: $primary;
}

a {
  color: $primary;
}
```

အတူတူပါပဲ။ Variable အဖြစ်သတ်မှတ်ဖို့ ၉ ကိုမသုံးတော့ဘဲ ၄ ကိုသုံးသွားတာပဲ ကွာသွားပါတယ်။ တစ်ချို့ Style Rule တွေကို LESS မှာ ဒီလိုလည်း ပြန်ခေါ်သုံးလို့ ရပါသေးတယ်။

**LESS**

```
.button {
  background: blue;
  color: white;
  padding: 6px 12px;
}

button {
  .button();
}

a {
  .button();
}
```

.button မှာရေးထားတဲ့ သတ်မှတ်ချက်တွေကို button အတွက်ယူသုံးလိုက်သလို၊ a အတွက်လည်း ယူသုံးလိုက်တာပါ။ ဒါကြောင့် တူညီတဲ့ Rule တွေကို တစ်ခါရေးထားယုံနဲ့ လိုတဲ့နေရာက ယူသုံးလိုရသွား သလို၊ ပြင်ဖို့လိုရင်လည်း တစ်နေရာမှာ ပြင်လိုက်ယုံနဲ့ ယူသုံးထားတဲ့နေရာအားလုံးမှာ သက်ရောက်သွား စေမှာပဲ ဖြစ်ပါတယ်။

အလားတူကုဒ်ကို SASS မှာ ဒီလိုရေးရပါတယ်။

### SASS

```
@mixin button {
    background: blue;
    color: white;
    padding: 6px 12px;
}

button {
    @include button;
}

a {
    @include button;
}
```

သူကတော့ @mixin Keyword ကိုသုံးဖိုး Rule တွေကို ကြိုးပေးရပြီး ယူသုံးချင်တဲ့နေရာမှာ @include နဲ့ ပြန်ယူသုံးလိုက်တာပါ။

SASS မှာ SASS နဲ့ SCSS ဆိုပြီးရေးထုံးမှုကဲ (၂) မျိုးရှုပါသေးတယ်။ တစ်ကယ်တော့ အခုန်မူနာပေးခဲ့တဲ့ ကုဒ်တွေကို SCSS လိုခေါ်မှ ပိုမျန်ပါမယ်။ SASS ရေးထုံးအမှန်နဲ့ဆိုရင် ဒီလိုဖြစ်မှာပါ။

### SASS

```
@mixin button
background: blue
color: white
padding: 6px 12px

button
@include button

a
@include button
```

သိပ်မကွာပါဘူး။ Bracket တွေ ပါခြင်း/မပါခြင်း နဲ့ Semi-colon တွေ ပါခြင်း/မပါခြင်း ကွာသွားတာပါ။ နှစ်မျိုးလုံး ဘယ်လိုရေးရေး ကြိုက်တဲ့ နည်းကို သုံးပြီး ရေးနိုင်ပါတယ်။ ဒါကြောင့် SASS နဲ့ SCSS ဆိုတဲ့ အသုံးအနှစ်း (၂) မျိုးတွေ ရင် မျက်စိမလည်ပါနဲ့။ အတူတူပါပဲ၊ ရေးထုံးနည်းနည်းလေး ကွာသွားတာပါ။

ဒီနည်းပညာတွေကိုသုံးပြီး ရေးထားတဲ့ ကုဒ်တွေကို တိုက်ရိုက်သုံးလို့ မရပါဘူး။ Browser တွေက CSS ကိုပဲ နားလည်ကြတာပါ။ LESS တွေ SASS တွေကို နားမလည်ကြပါဘူး။ ဒါကြောင့် ဒီကုဒ်တွေကို Browser နားလည်တဲ့ CSS ဖြစ်အောင် အရင်ပြောင်းပေးရပါတယ်။ ဒီလိုမျိုး CSS ဖြစ်အောင် အရင်ကြိုပြောင်းပြီး တော့ မှသာ သုံးလို့ရတဲ့ အတွက် Preprocessor နည်းပညာတွေလို့ ခေါ်ကြတာပါ။

Preprocessor နည်းပညာတွေရဲ့ ရေးနည်းအသေးစိတ်ကို အခုက်ခါတည်းလေ့လာဖို့ မဟုတ်သေးပါဘူး။ လိုအပ်လာတော့မှ ဆက်လက်လေ့လာ ကြရမှာပါ။ ဒီစာအပ်ရဲ့ နောက်ဆုံးခန်းမှာတော့ Bootstrap ကို Customize လုပ်လိုရတဲ့ SASS ကုဒ်တစ်ချို့ကို နှမူနာထည့်ပေးထားပါတယ်။ လောလောဆယ်မှာ Preprocessor ဆိုတဲ့ အသုံးအနှစ်းကိုတွေ့ရင် ဘာကိုပြောနေတာလဲဆိုတာ သိဖို့က အမိကပါ။

## CDN

Bootstrap အပါအဝင် CSS နည်းပညာတွေ၊ JavaScript နည်းပညာတွေ၊ Font နဲ့ Icon နည်းပညာတွေကို ပုံစံ (၃) မျိုးနဲ့ ရယူအသုံးပြုနိုင်လေ့ ရှိပါတယ်။ Download, CDN နဲ့ NPM တို့ဖြစ်ပါတယ်။

Download ကတော့ ရှင်းပါတယ်။ ပေးထားတဲ့ ဖိုင်တွေကို Download လုပ်ပြီး ကိုယ့်ပရောဂျက်ထဲမှာ ထည့်သွင်းအသုံးပြုခြင်း ဖြစ်ပါတယ်။ ဥပမာ ဒီလိုပါ -

```
<link rel="stylesheet" href="css/bootstrap.min.css">
```

နှမူနာအရ bootstrap.min.css ဆိုတဲ့ ဖိုင်က CSS ဖို့အတဲ့မှာ ရှိတယ်ဆိုတဲ့ သဘောနဲ့ Path လမ်းကြောင်းပေးပြီး ချိတ်ဆက်ထားတာပါ။ ဒါကြောင့် bootstrap.min.css ဖိုင်ကို ကြိုတင် Download ယူပြီး CSS ဖို့အတဲ့မှာ ထည့်ထားပေးဖို့ လိုအပ်မှာ ဖြစ်ပါတယ်။

CDN ကတေသ့ ဖိုင်တွေကို Download လုပ်စရာမလိုဘဲ ဆာဗာကနေ တိုက်ရှိက်ချိတ်သုံးတဲ့နည်း ဖြစ်တယ် လို အတိုချုပ် ဆိုနိုင်ပါတယ်။ ဥပမာ ဒီလိုပါ -

```
<link rel="stylesheet" href="https://cdnjs.com/css/bootstrap.min.css">
```

နမူနာမှာ bootstrap.min.css ဖိုင်ရဲ CDN ဆာဗာလိပ်စာအပြည့်စုံကိုပေးပြီး ချိတ်ဆက်လိုက်တာပါ။ နမူနာဖြစ်ပါတယ်၊ တစ်ကယ်ချိတ်ဖို့ဆိုရင် URL က ဒီထက်ပိုရည်ပါလိမ့်မယ်။ Version နံပါတ်တွေ ဘာတွေ ပါသေးလိုပါ။

လိုအပ်တဲ့ဖိုင်တွေကို CDN ဆာဗာကနေတိုက်ရှိက် ချိတ်သုံးလိုက်ရင် လက်တွေမှာ အကျိုးရှုပါတယ်။ CDN ဆိုတာ Content Distribution Network (သို့) Content Delivery Network ရဲ အတိုကောက် ဖြစ်ပါတယ်။ Google CDN, Microsoft CDN, Cloudflare CND စသေဖြင့် CDN Network တွေ ရှိကြပါတယ်။ Google တို့ Microsoft တို့က အသုံးများတဲ့ ဖိုင်တွေကို အများအဆင်ပြေစေဖို့အတွက် သူတို့ရဲ CDN ဆာဗာတွေပေါ်မှာ တင်ထားပေးကြပါတယ်။ CDN ဆာဗာတွေဆိုတာ ကမ္မာအနှစ်မှာ ဖြန့်ပြီးတော့ ထားကြတာပါ။ ဒါကြောင့် အသုံးပြုသူ User နဲ့ အနီးဆုံးဆာဗာကနေ လိုတဲ့ဖိုင်တွေကို ချပေးနိုင်မှာ ဖြစ်ပါတယ်။ စင်ကာပူနေ အသုံးပြုသူအတွက် လိုတဲ့ဖိုင်ကို စင်ကာပူ ဆာဗာကနေ ချပေးပြီး၊ ရန်ကုန်ကနေ အသုံးပြုသူအတွက် လိုတဲ့ဖိုင်ကို ရန်ကုန်ဆာဗာကနေ ချပေးတဲ့ အလုပ်မျိုးကို CDN ကလုပ်ပေးနိုင်ပါတယ်။

CDN ရဲ တစ်ခြားအားသာချက်တွေလည်း ရှုပါသေးတယ်။ ဒီနေရာမှာတော့ အကျယ်မချဲနိုင်ပါဘူး။ လိုရင်းအနေနဲ့ CDN ဆာဗာတွေက အသုံးပြုသူ User နဲ့ အနီးဆုံး ဆာဗာကနေ ဖိုင်တွေကိုချပေးနိုင်တယ်လိုသာ အတိုချုပ် မှတ်ထားပါ။

## NPM

NPM ကတေသ့ Node Package Manager ရဲ အတိုကောက် ဖြစ်ပါတယ်။ အရင်က JavaScript နည်းပညာတွေအတွက်ပဲ သုံးကြပေမယ့် အခုတေသ့ နည်းပညာအစုံအတွက် သုံးကြပါတယ်။ CSS နဲ့ JavaScript နည်းပညာတွေမှာ Dependency လိုခေါ်တဲ့ ဆက်စပ်လိုအပ်ချက်တွေ ရှိကြပါတယ်။ Bootstrap ရဲအရင် Version တွေမှာ jQuery လိုခေါ်တဲ့ JavaScript နည်းပညာ လိုအပ်ပါတယ်။ jQuery မပါရင် Bootstrap က အပြည့်အဝ အလုပ်မလုပ်ပါဘူး။ ဒါကြောင့် jQuery ဟာ Bootstrap အတွက် Dependency လို ဆိုနိုင်

ပါတယ်။ ဖိုင်တွေကို ကိုယ့်ဘာသာ Download လုပ်မယ်ဆိုရင် Dependency တွေကိုလည်း ကိုယ့်ဘာသာ Download ထပ်လုပ်ရပါတယ်။ Bootstrap ကို Download လုပ်၊ ပြီးရင် jQuery ကိုလည်း ထပ်ပြီး Download လုပ်ရမှာပါ။ NPM ကတော့ Dependency တွေကို အလိုအလျောက် Download လုပ်ပေးနိုင် တဲ့ နည်းပညာဖြစ်ပါတယ်။ Command Line နည်းပညာဖြစ်ပါတယ်။ ဥပမာ -

```
npm install bootstrap
```

ဒါ Command က Bootstrap ကို Download လုပ်ပေးပါလို့ ပြောလိုက်တာပါ။ ထူးခြားချက်အနေနဲ့ NPM က jQuery အပါအဝင် Bootstrap ရဲ့ Dependency တွေကို အလိုအလျောက် တစ်ခါတည်း Download လုပ်ပေးသွားမှာ ဖြစ်ပါတယ်။

NPM လိုနည်းပညာမျိုးဟာလည်း ကျယ်ပြန့်ပါတယ်။ ထုံးစံအတိုင်း ဒီအဆင့်မှာ ဘယ်လိုနည်းပညာမျိုးလဲ သိစေဖို့သာ ရည်ရွယ်ပါတယ်။ NPM နဲ့ ဖိုင်တွေကို ဒေါင်းလိုက်ရင် လိုအပ်တဲ့ Dependency တွေကို တစ်ခါတည်း အလိုအလျောက် တဲ့ဒေါင်းပေးတယ်လို့ အတိုချုပ် မှတ်ထားလိုက်ပါ။

## Minify

CSS နည်းပညာတွေ JavaScript နည်းပညာတွေကို တိတိုင်ကြသူတွေက ပုံစံနှစ်မျိုးနဲ့ ပေးကြလေ့ရှိပါတယ်။ မူရင်းကုံး နဲ့ အဲဒီကုဒ်ကို ကျိုးသွားအောင် ချုံထားတဲ့ ကုဒ်ဖြစ်ပါတယ်။ ကုဒ်တွေကိုကျိုးသွားအောင် ချုံလိုက်တဲ့လုပ်ငန်းကို Minify လုပ်တယ်လို့ ခေါ်ပါတယ်။ ချုံတယ်ဆိုတာ အများအားဖြင့် ကုဒ်ထဲမှာပါတဲ့ Space တွေ Indent တွေ Comment တွေကို ဖယ်ထဲတိုက်တာပါ။ ဒီ Space တွေ Comment တွေပါလို့သာ ကုဒ်က ဖတ်ကြည့်လို့ရှိနိုင်တာပါ။ ဒါကြောင့် Minify လုပ်ထားတဲ့ ကုဒ်ကတော့ ဘာတွေရေးထားလဲဖတ်ကြည့်လို့ အဆင်ပြေတော့မှာမဟုတ်ပါဘူး။

အကျဉ်းချုပ်အားဖြင့် ရေးထားတဲ့ကုဒ်ကိုလေ့လာချင်ရင် မူရင်းကုဒ်ကိုသုံးရမှာဖြစ်ပြီး၊ လက်တွေ့အသုံးပြုဖို့ အတွက်တော့ Minify ကုဒ်ကိုယူရမှာပါ။ ချုံထားတဲ့အတွက် ဖိုင်အရွယ်အစား သေးသွားလို့ User အတွက် Download လုပ်ရတာ ပိုမြန်သွားမှာဖြစ်ပါတယ်။

မူရင်းကုဒ်နဲ့ Minify လုပ်ထားတဲ့ကုဒ် ကဲပြားအာင် ဖိုင်အမည်ပေးတဲ့အခါ နောက်ကနေ .min ဆိုတာလေး ထည့်ပေးလေ့ရှိပါတယ်။ ဥပမာ bootstrap.css ဆိုရင် မူရင်းကုဒ်ဖြစ်ပြီး bootstrap.min.css ဆိုရင် Minify လုပ်ထားတဲ့ ကုဒ်ဖိုင် ဖြစ်နိုင်ပါတယ်။

ပေးထားတဲ့နမူနာမှာ မူရင်းကုဒ်နဲ့ Minify ကုဒ်တို့၏ ကွာခြားပုံကို ယူဉ်တဲ့လောကြည့်ပါ။

#### Normal CSS

```
html {
    font-family: sans-serif;
    line-height: 1.15;
}

article, aside {
    display: block;
}

body {
    margin: 0;
    font-family: Roboto, Arial;
    font-size: 1rem;
    font-weight: 400;
    line-height: 1.5;
    color: #212529;
    text-align: left;
    background-color: #fff;
}

hr {
    box-sizing: content-box;
    height: 0;
    overflow: visible;
}
```

#### Minify CSS

```
html{font-family:sans-serif;line-height:1.15}article,aside{display:block}body{margin:0;font-family:Roboto,Arial;font-size:1rem;font-weight:400;line-height:1.5;color:#212529;text-align:left;background-color:#fff}hr{box-sizing:content-box;height:0;overflow:visible}
```

ဒီအခန်းရဲ့ရည်ရွယ်ချက်ကတော့ အခုလိုအသုံးအနှစ်းတွေကို ကြိုတင်ရှင်းလင်းထားခြင်းအားဖြင့် နောက်ပိုင်း မှာ ဆက်လက်လေ့လာရတာ ပိုမိုဖြန်ဆန်သွားစေဖို့ ဖြစ်ပါတယ်။ ကြိုရှင်းမထားရင် ဘာကိုဆိုလိုမှန်း မသိတဲ့ အတိုကောက် အသုံးအနှစ်းတွေကြောင့် နောက်ပိုင်းမှာ မျက်စိလည်နောက်ပိုင်းမှာ စိုးလိုပါ။

## အခန်း (၄) – CSS Components

Bootstrap ကို CSS Components, JavaScript Components, Layouts စသဖြင့်အပိုင်းလိုက်ခဲ့ပြီး လေ့လာ သွားကြပါမယ်။ ဒီအခန်းမှာ လေ့လာမယ့် Components တွေကတော့ CSS Component တွေပါ။ ရိုးရိုးလေးပြောရရင် ကြိုတင်ရေးသားပေးထားတဲ့ CSS ကုဒ်တွေပါပဲ။ ဥပမာ - <a> Element တွေကို CSS နဲ့ Button ပုံစံလေးတွေ ဖြစ်အောင် ကိုယ့်ဘာသာ ရေးမယ်ဆိုရင် ရပါတယ်။ ဒီလိုရေးရမှာပါ။

### HTML

```
<a href="#" class="first">Link Button</a>
<a href="#" class="second">Link Button</a>
```

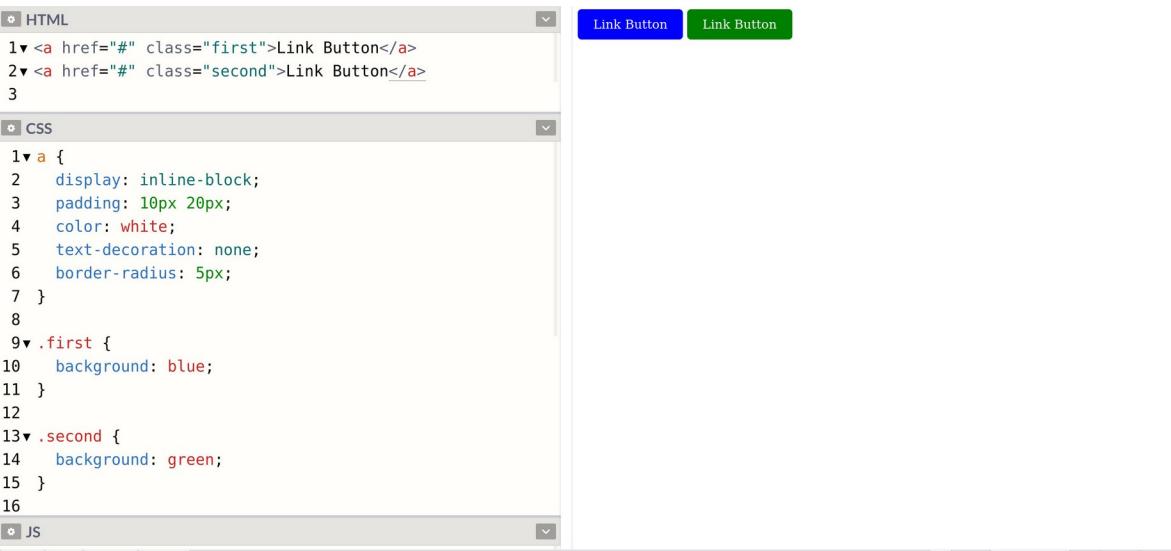
### CSS

```
a {
    display: inline-block;
    padding: 10px 20px;
    color: white;
    text-decoration: none;
    border-radius: 5px;
}

.first {
    background: blue;;
}

.second {
    background: green;
}
```

နမူနာအရ <a> Element နှစ်ခုလုံးအတွက် padding တွေ border-radius တွေ သတ်မှတ်ပေးလိုက်တဲ့အတွက် Button လေးတွေနဲ့ တူသွားပါတယ်။ ပြီးတော့မှ အရောင်မတူချင်လို့ သက်ဆိုင်ရာ class အလိုက် background တွေ သတ်မှတ်ပေးထားတာပါ။ စမ်းကြည့်လိုက်ရင် ရလဒ်က ဒီလိုပုံစံဖြစ်ပါလိမ့်မယ်။



```

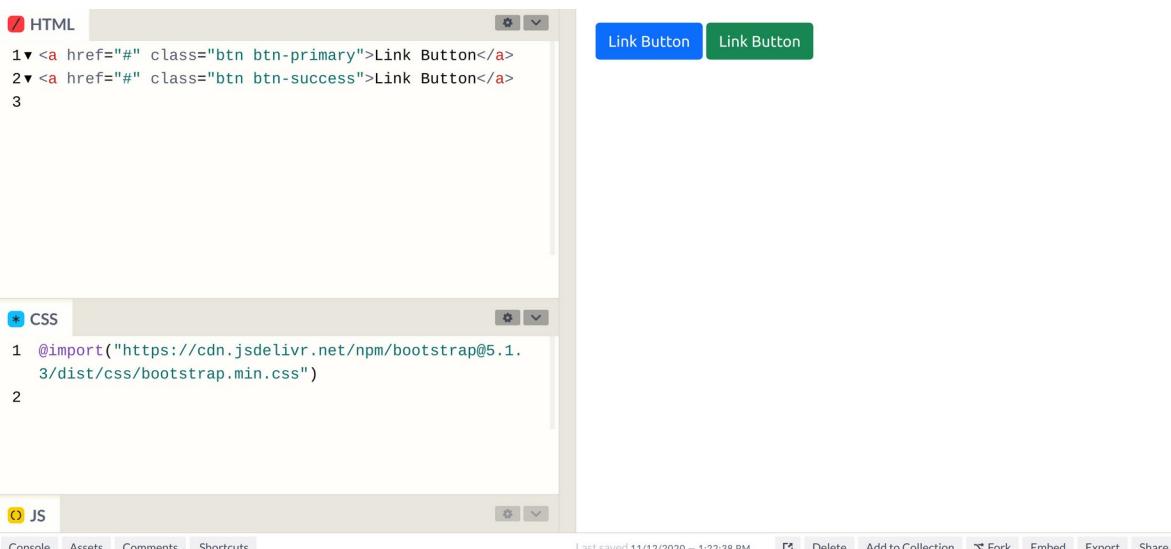
* HTML
1▼ <a href="#" class="first">Link Button</a>
2▼ <a href="#" class="second">Link Button</a>
3

* CSS
1▼ a {
2  display: inline-block;
3  padding: 10px 20px;
4  color: white;
5  text-decoration: none;
6  border-radius: 5px;
7 }
8
9▼ .first {
10   background: blue;
11 }
12
13▼ .second {
14   background: green;
15 }
16

* JS

```

အလားတူ ရလဒ်မျိုးရဖို့အတွက် Bootstrap ကိုအသုံးပြုပြီး အခုလို ရေးနိုင်ပါတယ်။



```

* HTML
1▼ <a href="#" class="btn btn-primary">Link Button</a>
2▼ <a href="#" class="btn btn-success">Link Button</a>
3

* CSS
1 @import("https://cdn.jsdelivr.net/npm/bootstrap@5.1.
3/dist/css/bootstrap.min.css")
2

* JS

```

Console Assets Comments Shortcuts Last saved 11/12/2020 – 1:22:38 PM Delete Add to Collection Fork Embed Export Share

CSS ကုဒ်တွေ ကိုယ့်ဘာသာ မရေးတော့ပါဘူး။ `@import` နဲ့ Bootstrap CSS ဖိုင်ကို ချိတ်ပေးလိုက်ပြီး Element တွေမှာ Bootstrap က သတ်မှတ်ထားတဲ့အတိုင်း `class` ကို မှန်အောင်ပေးလိုက်ယုံနဲ့ လိုချင်တဲ့ ရလဒ်ကို ရရှိခြင်းဖြစ်ပါတယ်။ Bootstrap က CSS ကုဒ်တွေကို [ကြိုးပေးထားတဲ့အတွက်ကိုယ်က အခုလို အသင့်ထည့် သုံးနိုင်ခြင်း ဖြစ်ပါတယ်။ Bootstrap သဘောသဘာဝ အနှစ်ချုပ်က ဒါပါပဲ။ သူက CSS ကုဒ်တွေ ရေးထားပေးတယ်။ ကိုယ်က ယူသုံးနိုင်ပါတယ်။](#)

Bootstrap ကို စတင်အသုံးပြုနိုင်ဖို့အတွက် နည်းလမ်းအမျိုးမျိုး ရှုပါတယ်။ သူ့ဝဘ်ဆိုက်ကိုသွားပြီးတော့ Download ရယူနိုင်သလို NPM နဲ့လည်း Download ရယူနိုင်ပါတယ်။ CDN ကနေ တိုက်ရှိက်ချိတ်ပြီးတော့ လည်း အသုံးပြုလိုရပါတယ်။ ဒီစာအုပ်မှာတော့ Code Pen ကိုအသုံးပြုပြီး နမူနာတွေ ဖော်ပြန်သလို စာဖတ်သူကိုလည်း Code Pen မှာပဲ တစ်ခါတည်း လိုက်ရေးစမ်းစေလိုတဲ့အတွက် CDN ကနေ တိုက်ရှိက်ချိတ်ဆက် အသုံးပြုတဲ့နည်းကို သုံးပါမယ်။

CDN ကနေ ချိတ်ချင်ရင် နည်းလမ်းနှစ်မျိုးနဲ့ ချိတ်နိုင်ပါတယ်။ တစ်နည်းကတော့ HTML ထဲမှာ အခုလို `<link>` Element ကိုသုံးပြီး ချိတ်နိုင်ပါတယ်။

### HTML

```
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css">
```

နောက်တစ်နည်းအနေနဲ့ အထက်ကန်မူနာမှာသုံးခဲ့သလို CSS ထဲမှာ `@import` နဲ့ ချိတ်ပြီးသုံးနိုင်ပါတယ်။

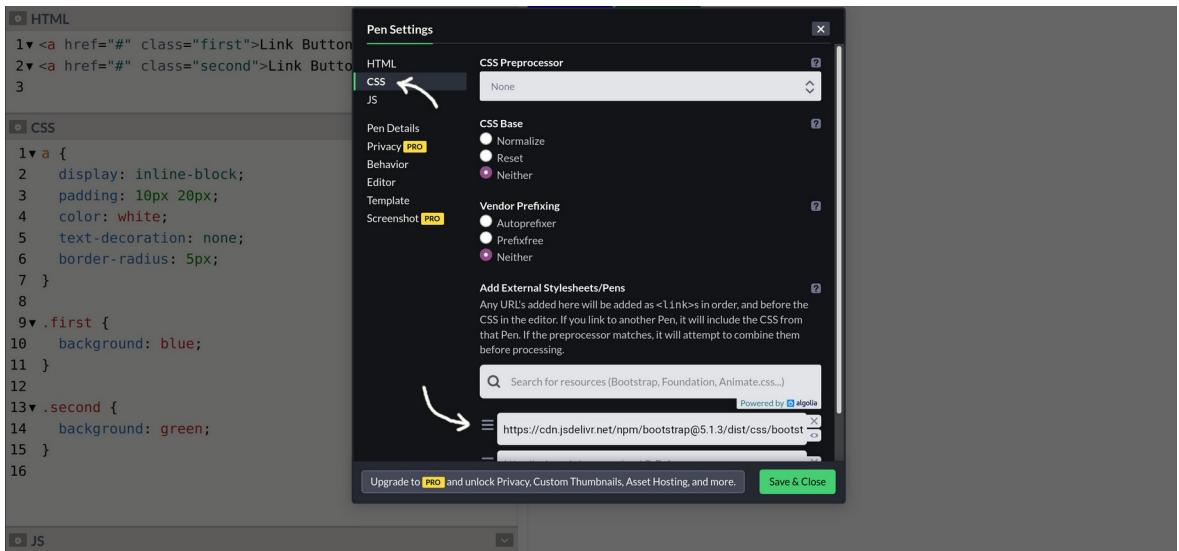
### CSS

```
@import ("https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css");
```

ဒီနေရာမှာ Version နံပါတ်ကိုသတိထားပါ။ နမူနာပေးထားတဲ့ Bootstrap ဖိုင်လိပ်စာမှာ 5.1.3 လိုပါနေပါတယ်။ ဒီစာရေးနေစဉ်မှာ ထွက်ထားတဲ့ နောက်ဆုံး Version ပါ။ စာဖတ်သူက လက်တွေ့စမ်းသပ်စဉ်မှာ နောက်ဆုံးရောက်ရှိနေတဲ့ Version ကိုလေ့လာပြီး အဲဒီ Version ကို အသုံးပြုသင့်ပါတယ်။ Bootstrap ရဲ့ Documentation မှာပဲ [ကြည့်လိုက်လို့ ရပါတယ်။](#)

- <https://getbootstrap.com/>

ဆက်လက်ပြီး Bootstrap ရဲ့ Component နမူနာတွေ စတင်လေ့လာပါတော့မယ်။ Code Pen မှာ CDN ကဖိုင်ကို <link> ထွေ @import ထွေနဲ့တောင်ချိတ်စရာမလိုပါဘူး။ Pen Setting ရဲ့ CSS Section မှာ ကိုယ်သုံးချင်တဲ့ ဖိုင်ကို ကြိုထည့်ပေးထားလို့ရပါတယ်။ ဒီလိုပါ –



Setting ရဲ့ CSS Section ကိုရွေးလိုက်ရင် Preprocessor, Reset, Vendor Prefix စသဖြင့် Option ထွေ ကို ထွေရပါလိမ့်မယ်။ ဒါတွေအကြောင်းကို ကြိုပြောပြထားပြီးသားပါ။ ဒါပေမယ့် ကိုယ့်ဘာသာ အဲဒီနည်းပညာတွေ တစ်ခုချင်းရွေးပေးစရာ မလိုပါဘူး။ Bootstrap ကို သုံးမှာမူ့လို့ Bootstrap ထဲမှာ ဒါတွေအကုန်ပါပြီးသားပါ။ External Stylesheets ဆိုတဲ့ နေရာမှာသာ Bootstrap CSS ဖိုင်တည်နေရာကို ထည့်ပေးလိုက်ရင်ရပါပြီ။

ထည့်ရမယ့်လိပ်စာကို ဖော်ပြပေးလိုက်ပါတယ်။ ကူးရေးမယ့်အစား Bootstrap Documentation မှာ သွားရှာပြီး ထည့်ပေးလိုက်သင့်ပါတယ်။ ကူးရေးမယ်ဆိုရင်၊ ရည်တဲ့အတွက် စာလုံးတွေကျွန်ုပြီး မှားနိုင်လို့ နည်းနည်းကရှိက်ပေးပါ။

<https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css>

ဒီလိုထည့်သွင်းပြီးပြီဆိုရင်တော့ Bootstrap Component နမူနာတွေ စတင်ရေးသား စမ်းသပ်လိုပါပြီ။ တစ်ခါတည်း လက်တွေ့လိုက်လုပ် ကြည့်စေချင်ပါတယ်။ မခက်သလို ပျော်စို့လည်းကောင်းပါတယ်။ လွယ်လွယ်လေးနဲ့ ရလဒ်တွေ့မြင်ရမှာ မို့လိုပါ။

## Alerts

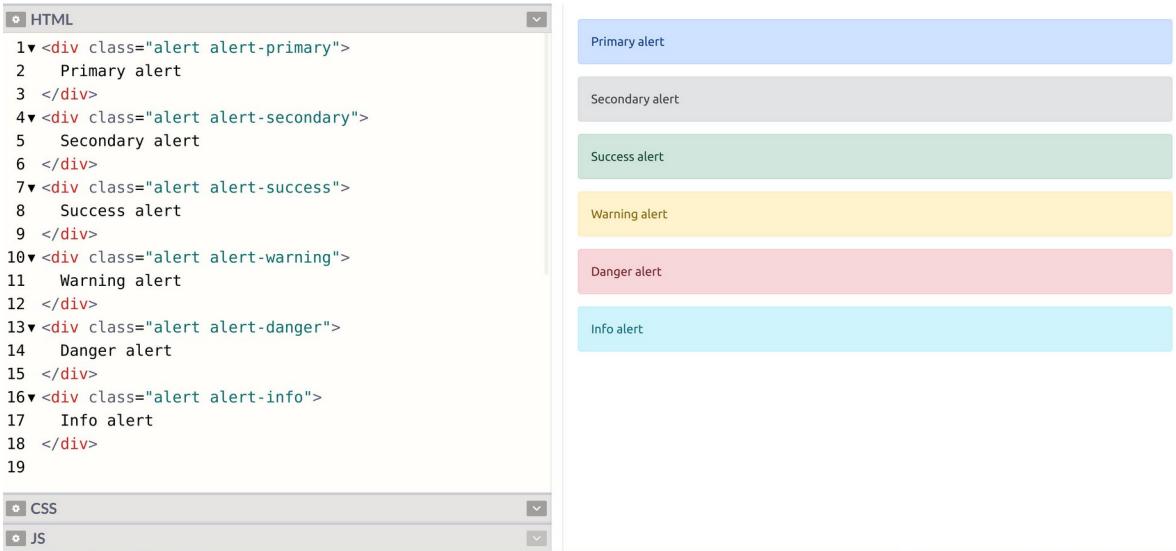
Alert Component ဟာ အသုံးဝင်တဲ့ Component တစ်ခုပါ။ ဝဘ်ဆိုက်နဲ့ User Interface တွေ တည်ဆောက်တဲ့အခါ User ကို အသိပေးစာ၊ သတိပေးစာလေးတွေ ပြချင်တဲ့အခါ သုံးကြပါတယ်။ အလုပ် တစ်ခု ပြီးသွားကြောင်း အသိပေးမယ်။ Error တက်သွားကြောင်း သတိပေးမယ် စသဖြင့်ပါ။ alert Class ကို သုံးရပါတယ်။ ဒီလိုပါ။

```
<div class="alert alert-success">
    Successfully completed something
</div>
```

ဒါဆိုရင် သပ်သပ်ရပ် အရောင်ချယ်ပြီး ပြပေးတဲ့ Alert Box ပုံစံလေးတစ်ခုကို ရရှိမှာဖြစ်ပါတယ်။ Bootstrap မှာ Color Class တွေရှိပါတယ်။ နမူနာမှာပါတဲ့ alert-success ဟာ Color Class တစ်ခု ဖြစ်ပါတယ်။ တစ်ခြား Color Class တွေရှိပါသေးတယ်။

- primary
- secondary
- success
- danger
- warning
- info
- light
- dark

alert-success ရဲ့ success နေရာမှာ ပေးထားတဲ့ Color Class တွေထဲက ကြိုက်တာနဲ့ အစားထိုး ပြီး သုံးလိုပါတယ်။ အခုလို နမူနာလေးတစ်ချို့ စမ်းကြည့်လိုက်ပါ။



alert Class ကိုသုံးထားလို့ Alert Box လေးတွေနဲ့ ပြတာချင်းတူပေါ်ယို Color Class မတူလို့ အရောင် လေးတွေ ကွဲပြားသွားတာကို တွေ့ရမှာပါ။ Bootstrap နဲ့ ဒါမျိုးလေးတွေကို အလွယ်တကူရရှိနိုင်တာပါ။ Color Class လေးတွေကို မှတ်ထားပေးပါ။ နောက်ပိုင်းမှာ ခဏခဏ အသုံးပြုရမှာပါ။

## List Groups

ဆက်လက်လေ့လာချင်တာကတော့ List Group ဖြစ်ပါတယ်။ သူလည်း အသုံးများပါတယ်။ <ul> တို့ <ol> တို့လို့ Bullet/Number List တွေကို App တွေမှာတွေ့ရလေ့ရှိတဲ့ Block List လေးဖြစ်အောင် ဖန်တီးပေးပါတယ်။ ဒီလို့ရေးရပါတယ်။

### HTML

```

<ul class="list-group">
  <li class="list-group-item">Item One</li>
  <li class="list-group-item">Item Two</li>
  <li class="list-group-item">Item Three</li>
  <li class="list-group-item">Item Four</li>
  <li class="list-group-item">Item Five</li>
</ul>

```

တစ်ကယ်တော့ <ul> တွေ <ol> တွေမှ မဟုတ်ပါဘူး၊ မည်သည့် Element ကိုမဆို သုံးလို့ရပါတယ်။ ပင်မ Element မှာ list-group Class ပေးပြီး အတွင်းထဲက Item တွေမှာ list-group-item ကို ပေးဖိုသာလိုပါတယ်။ ရလဒ်ကိုကြည့်လိုက်မယ်ဆိုရင် အခုပို သပ်သပ်ရပ်ရပ် ဘောင်ခတ်ပေးထားတဲ့

Block List လေးနဲ့ ပြပေးတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

```

1 <ul class="list-group">
2   <li class="list-group-item">Item One</li>
3   <li class="list-group-item">Item Two</li>
4   <li class="list-group-item">Item Three</li>
5   <li class="list-group-item">Item Four</li>
6   <li class="list-group-item">Item Five</li>
7 </ul>
8

```

Item One
Item Two
Item Three
Item Four
Item Five

Item တွေထဲက တစ်ခုကို ရွေးထားတဲ့ပုံစံ (သို့မဟုတ်) ဦးစားပေးဖော်ပြတဲ့ ပုံစံ ဖြစ်စေခဲင်ရင် active Class ကိုသုံးနိုင်ပါတယ်။ active Class ပါဘားတဲ့ Item ကို Highlight လုပ်ပြီးပြပေးတာပါ။

### HTML

```

<ul class="list-group">
  <li class="list-group-item">Item One</li>
  <li class="list-group-item active">Item Two</li>
  <li class="list-group-item">Item Three</li>
  <li class="list-group-item">Item Four</li>
  <li class="list-group-item">Item Five</li>
</ul>

```

ရလဒ်ကိုကြည့်လိုက်မယ်ဆိုရင် အခုလိုပုံစံဖြစ်ပါလိမ့်မယ်။

The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab displays the following code:

```

1▼ <ul class="list-group">
2▼   <li class="list-group-item">Item One</li>
3▼   <li class="list-group-item active">Item Two</li>
4▼   <li class="list-group-item">Item Three</li>
5▼   <li class="list-group-item">Item Four</li>
6▼   <li class="list-group-item">Item Five</li>
7 </ul>
8

```

The CSS and JS tabs are empty. To the right of the editor, the rendered output is shown in a list:

- Item One
- Item Two**
- Item Three
- Item Four
- Item Five

active Class ပါသွားတဲ့ Item ကို Highlight လုပ်ပြီးပြပေးတာပါ။ active ကိုမသုံးဘဲ စောစောက ပြောခဲ့တဲ့ Color Class တွေကို သုံးမယ်ဆိုရင်လည်းရပါတယ်။ ဒီလိုပါ -

The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab displays the following code:

```

1▼ <ul class="list-group">
2▼   <li class="list-group-item">Item One</li>
3▼   <li class="list-group-item list-group-item-success">Item Two</li>
4      list-group-item-success
5▼   <li class="list-group-item">Item Three</li>
6▼   <li class="list-group-item">Item Four</li>
7▼   <li class="list-group-item">Item Five</li>
8 </ul>
9

```

The CSS and JS tabs are empty. To the right of the editor, the rendered output is shown in a list:

- Item One
- Item Two**
- Item Three
- Item Four
- Item Five

နမူနာမှာ list-group-item-success ကို သုံးပြထားပါတယ်။ success အစား နှစ်သက်ရာ Color Class နဲ့အစားထိုးပြီးစမ်းကြည့်နိုင်ပါတယ်။

## Tables

Bootstrap Documentation ကိုသွားကြည့်လိုက်ရင် Table ကို Component စာရင်းထဲမှာ တွေ့ရမှာ မဟုတ်ပါဘူး။ Content စာရင်းထဲမှာ ဝါယာတွေကို Table ဟာလည်းပဲ Component တစ်ခုအနေနဲ့ အသုံးများပါတယ်။ သူ့အတွက် Class ကလည်း အထူးမှတ်စရာ မလိုပါဘူး။ table ဆိတဲ့ Class ကိုပဲ သုံးရတာပါ။ ဒီလိုပါ –

### HTML

```
<table class="table">
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>1</td>
    <td>Alice</td>
    <td>22</td>
  </tr>
  <tr>
    <td>2</td>
    <td>Bob</td>
    <td>23</td>
  </tr>
</table>
```

table နဲ့အတူ ပူးတဲ့ပြီးသုံးကြလေ့ရှိတဲ့ Class တွေတော့ ရှိပါတယ်။ table-striped နဲ့ table-bordered တို့ကို အသုံးများကြပါတယ်။ table-striped က Row တွေကို ပြတဲ့အခါ တစ်ကြောင်း ကျော်စီ အရောင်ခွဲပြစေဖို့ ဖြစ်ပါတယ်။ အသုံးဝင်ပါတယ်။ ပါဝင်တဲ့အချက်အလက်များတဲ့ Table တွေမှာ အဲဒီလိုခွဲပြမှာသာ အချက်အလက်တွေကို ဖတ်ရှား အဆင်ပြစေမှာ ဖြစ်ပါတယ်။ table-bordered ကတော့ Table ကို ဘောင် အပြည့်ခတ်ပြီး ပြစေချင်တဲ့အခါ သုံးဖို့ပါ။ မဖြစ်မနေ ထည့်ပေးရမှာ မဟုတ်ဘဲ လိုအပ်ရင်သုံးဖို့ ဖြစ်ပါတယ်။ စမ်းကြည့်လိုက်မယ်ဆိုရင် ရလဒ်ကို အချလိုတွေ့မြင်ရမှာပါ။

```

1▼ <table class="table table-striped table-bordered">
2▼   <tr>
3▼     <th>ID</th><th>Name</th><th>Age</th>
4   </tr>
5▼   <tr>
6▼     <td>1</td><td>Alice</td><td>22</td>
7   </tr>
8▼   <tr>
9▼     <td>2</td><td>Bob</td><td>23</td>
10  </tr>
11▼  <tr>
12▼    <td>3</td><td>Carl</td><td>24</td>
13  </tr>
14▼  <tr>
15▼    <td>4</td><td>Dean</td><td>25</td>
16  </tr>
17 </table>
18

```

ID	Name	Age
1	Alice	22
2	Bob	23
3	Carl	24
4	Dean	25

နှမူနာမှာ table-striped နဲ့ table-bordered တိုကိုပါထည့်ပြထားပါတယ်။ မပါဘဲလည်း စမ်းကြည့်သင့်ပါတယ်။ ဒီတော့မှ ဘာကွာလဲဆိတာကို လက်တွေ့မြင်သွားမှာပါ။ Table နဲ့ပက်သက်ဖြီး လုပ်လို ရတာတွေ အများမြှိုးရှိပါတယ်။ ဒါပေမယ့် ဒီနေရာမှာ အသုံးများမယ့် လုပ်ဆောင်ချက်လေးတွေကိုသာ ရွေးမှတ်ပါ။ Table မှာလည်း Color Class တွေသုံးလို့ရပါတယ်။ ဒီလိုပါ -

```

1▼ <table class="table table-dark table-striped">
2▼   <tr>
3▼     <th>ID</th><th>Name</th><th>Age</th>
4   </tr>
5▼   <tr>
6▼     <td>1</td><td>Alice</td><td>22</td>
7   </tr>
8▼   <tr>
9▼     <td>2</td><td>Bob</td><td>23</td>
10  </tr>
11▼  <tr>
12▼    <td>3</td><td>Carl</td><td>24</td>
13  </tr>
14▼  <tr>
15▼    <td>4</td><td>Dean</td><td>25</td>
16  </tr>
17 </table>
18

```

ID	Name	Age
1	Alice	22
2	Bob	23
3	Carl	24
4	Dean	25

နှမူနာမှာပေးထားတဲ့ table-dark ရဲ့ dark နေရာမှာ တစ်ခြား Color Class နဲ့အစားထိုးပြီး စမ်းကြည့်လို့ရပါတယ်။

## Forms

Form ဟာလည်းပဲ Documentation မှာသွားကြည့်ရင် Component စာရင်းထဲမှာ မပါပါဘူး။ သီးခြား ခေါင်းစဉ်တစ်ခုနဲ့ ခွဲပေးထားပါတယ်။ ဒီနေရာမှာတော့ တစ်ခါတည်းပဲ တဲ့ပြီးဖော်ပြပါမယ်။ သူမှာ ရေးစရာ နဲ့ မှတ်စရာနည်းနည်းတော့ များပါတယ်။ ဒီလိုပါ -

### HTML

```
<form>
  <div class="mb-3">
    <label for="name">Name</label>
    <input type="text" id="name" class="form-control">
  </div>

  <div class="mb-3">
    <label for="address">Address</label>
    <textarea id="address" class="form-control"></textarea>
    <div class="form-text">Enter your full address</div>
  </div>

  <div class="mb-3">
    <label for="gender">Gender</label>
    <select id="gender" class="form-select">
      <option>Male</option>
      <option>Female</option>
    </select>
  </div>

  <button class="btn btn-primary">Submit Form</button>
</form>
```

ပထမဆုံး <div> မှာ ပေးထားတဲ့ mb-3 Class ကို သတိပြုပါ။ Margin Bottom သတ်မှတ်လိုက်တာပါ။ တစ်ခုနဲ့တစ်ခု နည်းနည်းကွာသွားစေဖို့အတွက်ပါ။ အသုံးဝင်တဲ့ Utility Class ဖြစ်ပါတယ်။ အရင် Version တွေမှာ form-group လိုပေါ်တဲ့ Class တစ်ခုပါပေမယ့် Bootstrap 5 မှာ မပါတော့တာကို တွေ့ရပါ တယ်။ ဒါကြောင့် Input Group တွေ တစ်ခုနဲ့တစ်ခု ကွာသွားစေဖို့ mb Class ကိုပဲ သုံးရတော့မှာပါ။ 3 နေရာမှာ 1-5 ကြိုက်တဲ့တန်ဖိုး ပြောင်းပေးကြည့်ပါ။ အမိက Input သုံးမျိုးဖြစ်တဲ့ Text Input, Textarea နဲ့ Select တို့ကို နှမူနာပေးထားပါတယ်။ Text Input နဲ့ Textarea တို့အတွက် form-control ဆိုတဲ့ Class ကိုသုံးပါတယ်။ Select အတွက်လည်း form-control ကိုပဲ သုံးလိုရပါတယ်။ ဒါပေမယ့် Select Box မှန်းပေါ်လွင်စေတဲ့ Down Arrow လေးနောက်ဆုံးမှာ ပါစေခဲ့လို့ form-select ဆိုတဲ့ Class ကို သုံးထားတာကို သတိပြုရမှာ ဖြစ်ပါတယ်။

နမူနာမှာ form-text Class ကိုသုံးထားတဲ့ Element တစ်ခုပါသေးတာကိုလည်း တွေ့ရနိုင်ပါတယ်။ ဘာကိုရေးဖြည့်ရမှာလဲရင်းပြတဲ့ ရှင်းလင်းချက်လေးတွေ တွဲထည့်ဖို့အတွက် သင့်တော်ပါတယ်။ နောက်ဆုံး တစ်ခုဖြစ်တဲ့ Button ကတော့ ဟိုးအပေါ်မှာလည်း တစ်ခါတွေ့ခဲ့ဖူးပြီးသားပါ။ btn Class ကိုသုံးပြီး primary နေရာမှာ တစ်ခြား Color Class တွေကို လိုအပ်ရင် သုံးနိုင်ပါတယ်။ ရလဒ်ကို စစ်ဆေးတဲ့ မယ်ဆိုရင် အခုလိုတွေ့ရမှာပါ။

The screenshot shows a web-based form builder interface. On the left, there's a code editor with numbered lines of HTML code. Lines 1-18 show the main form structure with input fields for Name, Address, and Gender, and a submit button. Lines 19-20 show the closing form tag. Below the code editor are three tabs: 'CSS' and 'JS' are visible, while 'HTML' is currently selected. On the right, there's a preview area showing a simple form with four fields: 'Name' (text input), 'Address' (text area), 'Enter your full address' (placeholder text), and 'Gender' (dropdown menu set to 'Male'). A blue 'Submit Form' button is at the bottom.

```

1  <form>
2   <div class="mb-3">
3    <label for="name">Name</label>
4    <input type="text" id="name" class="form-control">
5   </div>
6   <div class="mb-3">
7    <label for="address">Address</label>
8    <textarea id="address" class="form-control"></textarea>
9    <div class="form-text">Enter your full address</div>
10  </div>
11  <div class="mb-3">
12   <label for="gender">Gender</label>
13   <select id="gender" class="form-select">
14    <option>Male</option>
15    <option>Female</option>
16   </select>
17  </div>
18  <button class="btn btn-primary">Submit Form</button>
19 </form>
20

```

နောက်ထပ်ဖြည့်စွက်လေ့လာသင့်တာကတော့ Input Group လိုခေါ်တဲ့ လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။ Input တွေကို Button တွေ၊ စာတွေနဲ့ ပူးတဲ့ပြီး ကြည့်ကောင်းအောင် ပြတဲ့လုပ်ဆောင်ချက်မျိုးပါ။

### HTML

```

<form>
  <div class="input-group mb-3">
    <input type="text" class="form-control">
    <button class="btn btn-secondary">Search</button>
  </div>

  <label for="email">Enter Your Gmail Address</label>
  <div class="input-group mb-3">
    <input type="text" id="email" class="form-control">
    <span class="input-group-text">@gmail.com</span>
  </div>
</form>

```

နမူနာမှာ <div> ရဲ့ Class ကို input-group လို့ သတ်မှတ်ပေးထားတာကို သတိပြုပါ။ နမူနာနှစ်မျိုး ပေးထားပါတယ်။ ပထမတစ်ခုက Text Input နဲ့ Button ကို Input Group ထဲမှာ ထည့်လိုက်တဲ့အခါ ပူးတဲ့ ဖြီးကြည့်ကောင်းအောင် ပြပေးမှာပါ။ Button မဟုတ်ဘဲ ရိုးရိုးစာကို Input နဲ့တွဲပြချင်တယ်ဆိုရင်တော့ input-group-text Class သတ်မှတ်ထားတဲ့ Element ကိုသုံးတယ်ဆိုတာကို တွေ့ရနိုင်ပါတယ်။ စမ်းကြည့်လိုက်ရင် ရလဒ်ကဒီလိုဖြစ်မှာပါ။

```

1▼ <form>
2▼  <div class="input-group mb-3">
3  <input type="text" class="form-control">
4▼  <button class="btn btn-secondary">Search</button>
5  </div>
6
7▼ <label for="email">Enter Your Gmail Address</label>
8▼ <div class="input-group mb-3">
9  <input type="text" id="email" class="form-control">
10▼ <span class="input-group-text">@gmail.com</span>
11 </div>
12
13▼ <label for="price">Enter Price</label>
14▼ <div class="input-group mb-3">
15▼   <span class="input-group-text">$</span>
16   <input type="text" id="price" class="form-control">
17 </div>
18
19 </form>
20

```

Search

Enter Your Gmail Address @gmail.com

Enter Price \$

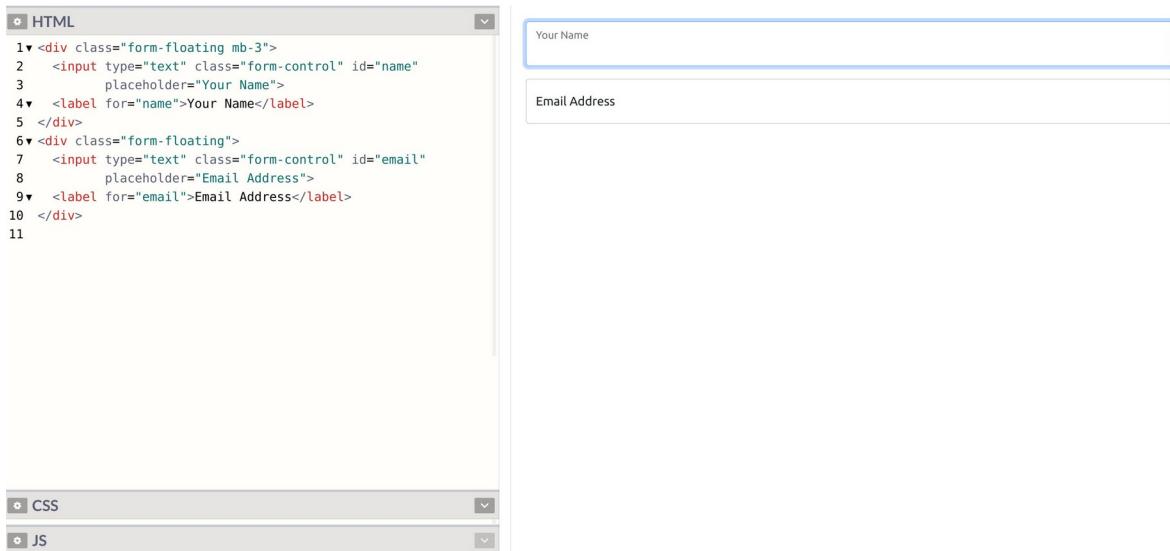
input-group-text Class သတ်မှတ်ထားတဲ့ Element ကို Input ရဲရှေ့မှာထားလို့ရသလို နောက်မှာ ထားလိုလည်း ရပါတယ်။ ရှေ့နောက်နှစ်ခုထည့်ချင်လည်း ရပါတယ်။ ရှေ့မှာချည်းပဲနှစ်ခု၊ နောက်မှာချည်းပဲနှစ်ခုထည့်ချင်ရင်လည်း ရတာပါပဲ။ အမျိုးမျိုးစမ်းကြည့်နိုင်ပါတယ်။

Bootstrap 5 Alpha 3 ကျတော့မှ စပါလာတဲ့ Floating Label လုပ်ဆောင်ချက်ကိုလည်း ဖြည့်စွက် ဖော်ပြချင်ပါတယ်။ Bootstrap မှာ ဒီလုပ်ဆောင်ချက်က အခုမှပါပေမယ့် လက်တွေ့ပရောဂျက်တွေမှာ လူသုံးများ နေပြီးသားပါ။ Google ရဲ့ Material Design လိုခေါ်တဲ့ နောက်ထပ် ဒီဇိုင်းနည်းပညာ တစ်ခုကနေလာတဲ့ လုပ်ဆောင်ချက်ပါ။ Android Mobile App တွေမှာ ဒီလုပ်ဆောင်ချက်ကို မကြာခဏတွေ့ရနိုင်ပါတယ်။ ပထမ Input အတွင်းထဲမှာ Label ကရှိနေပြီး Input မှာ Focus ဖြစ်သွားတော့မှာသာ Label လေးက နေရာ ဖယ်ပေးတဲ့ သဘောနဲ့ အပေါ်ရွှေ့ပြီး ပြပေးတဲ့ လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။ အခုလိုရေးရပါတယ်။

**HTML**

```
<div class="form-floating mb-3">
  <input type="text" class="form-control" id="name"
    placeholder="Your Name">
  <label for="name">Your Name</label>
</div>
<div class="form-floating">
  <input type="text" class="form-control" id="email"
    placeholder="Email Address">
  <label for="email">Email Address</label>
</div>
```

ပင်မ Element မှာ form-floating Class ပါဝင်ပြီး Input တွေမှာ placeholder Attribute ပါတာ ကို သတိပြုပါ။ Placeholder ကို အရင်ပြုပြီး Focus ဖြစ်တော့မှ <label> ကို ပြောင်းပြေားမှာ ဖြစ်ပါတယ်။ Placeholder မပါဘဲလည်း စမ်းကြည့်နိုင်ပါတယ်။

**Button Groups & Pagination**

Button Group ကို Toolbar ပုံစံ Button တွေစွဲပြီး သပ်သပ်ရပ်ရပ်ပြစ်လိုတဲ့အခါ သုံးနိုင်ပါတယ်။ သူ ကတော့ မှတ်ရလွယ်ပါတယ်။ ထူးထူးဆန်းဆန်း မဟုတ်ပါဘူး။ btn-group ထဲမှာ btn တွေကို စုစုပေါ်လိုက်တာပါပဲ။ ဒီလိုပါ -

**HTML**

```
<div class="btn-group">
  <a href="#" class="btn btn-primary">Left</a>
  <a href="#" class="btn btn-primary">Center</a>
  <a href="#" class="btn btn-primary">Right</a>
</div>
```

<button> Element ကိုမသုံးဘဲ <a> Element တွေကို သုံးထားတာကို သတိပြုပါ။ ကိုယ့်လိုအပ်ချက်ပေါ်မှုတည်ပြီး ကြိုက်တဲ့ Element ကိုသုံးပါ။ သုံးလိုရပါတယ်။ btn တွေမှာ Color Class တွေသုံးတဲ့အခါ Background Color နဲ့ ပြေားတာကို တွေ့ခဲ့ကြပြီး ဖြစ်ပါလိမ့်မယ်။ Background Color နဲ့မဟုတ်ဘဲ Border Color နဲ့ပြစ်ချင်ရင်လည်းရပါတယ်။ ဒီလိုရေးရပါတယ် –

**HTML**

```
<div class="btn-group">
  <a href="#" class="btn btn-outline-primary">Left</a>
  <a href="#" class="btn btn-outline-primary">Center</a>
  <a href="#" class="btn btn-outline-primary">Right</a>
</div>
```

Pagination ဆိုတာကတော့ Content တွေများလို ခွဲပြီးပြတဲ့အခါ 1, 2, 3, 4 စသဖြင့် လိုချင်တဲ့စာမျက်နှာကို သွားလိုရတဲ့ ခလုပ်လေးတွေပါ။ တွေ့ဖူးကြပါလိမ့်မယ်။ သူက Button Group နဲ့ ရေးသားပဲ မတူပေမယ့် ဖော်ပြပုံဆင်တူပါတယ်။ ဒါကြောင့် တစ်ခါတည်းအတွဲလိုက် ထည့်ကြည့်ချင်ပါတယ်။ <ul><li> <a> တို့ကို အသုံးပြုပြီး အခုလိုရေးရပါတယ်။

**HTML**

```
<ul class="pagination">
  <li class="page-item">
    <a href="#" class="page-link">1</a>
  </li>
  <li class="page-item">
    <a href="#" class="page-link">2</a>
  </li>
  <li class="page-item">
    <a href="#" class="page-link">3</a>
  </li>
</ul>
```

<ul> အတွက် pagination Class ကိုသတ်မှတ်ပေးရပါတယ်။ <li> အတွက် page-item ကိုသတ်မှတ်ပေးရပြီး <a> အတွက် page-link ကိုသတ်မှတ်ပေးရခြင်းဖြစ်ပါတယ်။ မှတ်စရာ (၃) ခုဖြစ်သွားပေမယ့် မှတ်ရတော့ မခက်လှပါဘူး။ ဒါတွေအားလုံးကို ပေါင်းပြီး အခုလို နှမူနာစမ်းကြည့်နိုင်ပါတယ်။

```

1▼ <div class="btn-group">
2  <a href="#" class="btn btn-primary">Left</a>
3▼ <a href="#" class="btn btn-primary">Center</a>
4▼ <a href="#" class="btn btn-primary">Right</a>
5 </div>
6
7▼ <div class="btn-group">
8  <a href="#" class="btn btn-outline-primary">Left</a>
9▼ <a href="#" class="btn btn-outline-primary">Center</a>
10<a href="#" class="btn btn-outline-primary">Right</a>
11 </div>
12
13▼ <ul class="pagination mt-4">
14  <li class="page-item"><a href="#" class="page-link">Previous</a></li>
15  <li class="page-item"><a href="#" class="page-link">1</a></li>
16  <li class="page-item"><a href="#" class="page-link">2</a></li>
17  <li class="page-item"><a href="#" class="page-link">3</a></li>
18  <li class="page-item"><a href="#" class="page-link">Next</a></li>
19 </ul>
2A

```

## Cards

Card Component ကတော့ တစ်ချို့ စုစုပြီး အတွဲလိုက်ပြရမယ့်အချက်အလက်တွေ ပြန့်အတွက် သုံးရတဲ့ Component ပါ။ ခေါင်းစဉ်၊ စာကိုယ်၊ ခလုပ်၊ လင့်၊ ပုံ စသဖြင့် သင့်တော်သလို တွဲဖက်ဖော်ပြန့် လိုအပ်ရင် သုံးရတာပါ။ အသုံးဝင်ပါတယ်။ ခေါင်းစဉ်၊ စာကိုယ်နဲ့ ခလုပ်တစ်ခုပါတဲ့ Card တစ်ခုကို အခုလိုဖန်တီးယူနိုင်ပါတယ်။

```

HTML
<div class="card">
  <div class="card-body">
    <h5 class="card-title">Card Title</h5>
    <p>Some card content</p>
    <a href="#" class="btn btn-primary">More...</a>
  </div>
</div>

```

ပင်မ Element မှာ card Class ကို သတ်မှတ်ပေးရပြီး၊ ကျန်တဲ့ ခေါင်းစဉ်တွေ စာကိုယ်တွေ အကုန်လုံးကို card-body ထဲမှာ အကုန်စုထည့် ပေးလိုက်တာပါ။ ခေါင်းစဉ်အတွက် card-title Class ကိုသုံးပါတယ်။ card-subtitle လည်း လိုအပ်ရင်သုံးလို့ရပါသေးတယ်။ ခလုပ်ကိုတော့ btn Class ပဲသုံးထားပြီး Link တွေထည့်ချင်ရင် card-link Class ကို btn အစားသုံးနိုင်ပါတယ်။ စမ်းကြည့်လိုက်ရင် ရလဒ်ကို အခုလိုတွေ့မြင်ရမှာပါ -

The screenshot shows a code editor interface with two tabs: 'HTML' and 'CSS'. The 'HTML' tab contains the following code:

```

1▼ <div class="card">
2▼   <div class="card-body">
3▼     <h5 class="card-title">Some Title</h5>
4▼     <p>
5       Some card content. Lorem ipsum dolor sit amet
       consectetur adipisicing elit. Omnis molestiae
       totam at quos iure earum reprehenderit velit
       aspernatur tempore dolorem qui, incident
       officiis illo, dicta vero, obcaecati culpa
       illum consequuntur.
6     </p>
7▼     <a href="#" class="btn btn-primary">See More</a>
8   </div>
9 </div>
10

```

The 'CSS' and 'JS' tabs are also visible below the 'HTML' tab.

The right side of the interface shows the rendered preview of the card component. It features a title 'Some Title', some placeholder text, and a blue 'See More' button.

Card အတွင်းထဲမှာ Header, Body နဲ့ Footer ဆိုပြီး အပိုင်းလိုက်ခွဲထည့်ချင်ရင်လည်း ထည့်လို့ရပါတယ်။

The screenshot shows a code editor interface with a single tab labeled 'HTML'.

```

<div class="card">
  <div class="card-header">
    <strong>Card Header</strong>
  </div>
  <div class="card-body">
    Some card content
  </div>
  <div class="card-footer">
    <small>Card Footer</small>
  </div>
</div>

```

card-header, card-body, card-footer Class တွက် သူများနဲ့သူ သုံးပေးလိုက်တာပါ။ စမ်းကြည့်လိုက်မယ်ဆိုရင် အခုလို အပိုင်းလိုက်ခဲ့ပြီးဖော်ပြပေးတဲ့ ရလဒ်ကိုရရှိမှာဖြစ်ပါတယ်။

**HTML**

```

1▼ <div class="card">
2▼   <div class="card-header">
3▼     <strong>Card Title</strong>
4   </div>
5▼   <div class="card-body">
6     Lorem ipsum dolor sit amet consectetur
7     adipisicing elit. Omnis molestiae totam at quos
8     iure earum reprehenderit velit aspernatur
9     tempore dolorem qui, incidenti officiis illo,
10    dicta vero, obcaecati culpa illum consequuntur.
11   </div>
12▼   <div class="card-footer">
13▼     <small>Card footer</small>
14   </div>
15 </div>
16

```

**CSS**

**JS**

**Card Title**

Lore ipsum dolor sit amet consectetur adipisicing elit. Omnis molestiae totam at quos iure earum reprehenderit velit aspernatur tempore dolorem qui, incidenti officiis illo, dicta vero, obcaecati culpa illum consequuntur.

**Card footer**

List တွေ Table တွေ Image တွက်လည်း Card နဲ့တွဲပြီး သုံးချင်ရင် သုံးလိုရပါသေးတယ်။ ဒီလိုပါ -

**HTML**

```

1▼ <div class="card">
2▼   <div class="card-header">
3▼     <strong>Card Title</strong>
4   </div>
5▼   <div class="card-body">
6     Lorem ipsum dolor sit amet consectetur
7     adipisicing elit. Omnis molestiae totam at quos
8     iure earum reprehenderit velit aspernatur
9     tempore dolorem qui, incidenti officiis illo,
10    dicta vero, obcaecati culpa illum consequuntur.
11   </div>
12▼   <ul class="list-group list-group-flush">
13▼     <li class="list-group-item">Item One</li>
14▼     <li class="list-group-item">Item Two</li>
15▼     <li class="list-group-item">Item Three</li>
16   </ul>
17 </div>
18

```

**CSS**

**JS**

**Card Title**

Lore ipsum dolor sit amet consectetur adipisicing elit. Omnis molestiae totam at quos iure earum reprehenderit velit aspernatur tempore dolorem qui, incidenti officiis illo, dicta vero, obcaecati culpa illum consequuntur.

**Item One**

**Item Two**

**Item Three**

နမူနာမှာ list-group Component ကို Card ထဲမှာ ထည့်သုံးထားပါတယ်။ list-group-flush Class ကို တွဲပေးထားတာသတိပြုပါ။ list-group မှာ ဘေးဘောင်တွေကို မပါစေချင်ရင် သုံးရတဲ့ Class ဖြစ်ပါတယ်။ List မှာ ဘေးဘောင်တွေပါနေရင် Card ရဲ့ဘောင်နဲ့ရော်းနှစ်ထပ်ဖြစ်သွားရင် ကြည့်မကောင်းလို ဒါ Class ကို တွဲထည့်ပေးထားတာပါ။ Card တွေကို အရောင်တွေခွဲဗြို့ဗြို့ သုံးချင်ရင်တော့ bg, text, border စတဲ့ Utility Class တွေကို သုံးနိုင်ပါတယ်။ ဒီလိုပါ -

The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab contains the following code:

```

1 <div class="card bg-primary text-light">
2   <div class="card-header">
3     <strong>Card Title</strong>
4   </div>
5   <div class="card-body">
6     Lorem ipsum dolor sit amet consectetur
7     adipisicing elit. Omnis molestiae totam at quos
8     iure earum reprehenderit velit aspernatur
9     tempore dolorem qui, incidunt officiis illo,
10    dicta vero, obcaecati culpa illum consequuntur.
11  </div>
12 </div>
13

```

The CSS and JS tabs are empty. To the right of the code editor is a preview window showing a blue card with a white header containing the text "Card Title" and a white body containing placeholder text from the Lorem ipsum.

နမူနာမှာ bg-primary ကိုသုံးပြီး အရောင်ပြောင်းထားပါတယ်။ တစ်ခြား Color Class တွေထဲက နှစ်သက်ရာကို သုံးနိုင်ပါတယ်။ နောက်ခံ အရောင်ထည့်ထားတော့ စာတွေမဲ့နောရင် ဖတ်ရတာအဆင်မပြေ လို text-light ကိုသုံးထားပါတယ်။ သူလည်းပဲ လိုအပ်ရင်တစ်ခြား Color Class တွေ သုံးနိုင်ပါတယ်။

နောက်ခံအရောင် အပြည့်မထည့်လိုပဲ Border လောက်ကိုပဲအရောင်ပြောင်းရင်လည်း ကြည့်လိုကောင်းပါတယ်။ border-success လို Class မျိုးထည့်ပြီး စမ်းကြည့်နိုင်ပါတယ်။ ထုံးစံအတိုင်း success အစား နှစ်သက်ရာ Color Class နဲ့သုံးလိုရှုနိုင်ပါတယ်။

## Navs & Tabs

ဆက်ကြည့်မှာကတော့ Tab UI အကြောင်းပါ။ အသုံးဝင်ပြီး နေရာတိုင်းမှာ တွေ့မြင်ရတဲ့လုပ်ဆောင်ချက် တစ်ခု ဖြစ်ပါတယ်။ Bootstrap ကတော့ Navs လိုခေါ်ပါတယ်။ သူလည်းပဲ <ul><li><a> ကိုသုံးရပါတယ်။ ဒါလိုပါ -

### HTML

```
<ul class="nav nav-tabs">
    <li class="nav-item">
        <a href="#" class="nav-link active">All User</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link">All User</a>
    </li>
    <li class="nav-item">
        <a href="#" class="nav-link">All User</a>
    </li>
</ul>
```

မှတ်စရာများပေမယ့် မှတ်ရလွယ်ပါတယ်။ ပင်မ <ul> အတွက် nav nav-tabs ဆိုတဲ့ Class တွေကို ပေးရပြီး <li> တွေအတွက် nav-item ကိုပေးရပါတယ်။ <a> တွေအတွက်တော့ nav-link Class ကို သတ်မှတ်ပေးရပါတယ်။ active Class ကတော့ လက်ရှိ ရွေးထားသကဲ့သို့ ဖော်ပြုစေလိုတဲ့ တစ်ခုမှာ သတ်မှတ်ပေးရတာပါ။ စမ်းကြည့်ရင် ရလဒ်ကို အခုလိုတွေ့ရမှာ ဖြစ်ပါတယ်။

The screenshot shows a browser's developer tools with three tabs: HTML, CSS, and JS. The HTML tab is active, displaying the following code:

```

1<ul class="nav nav-tabs">
2    <li class="nav-item">
3        <a href="#" class="nav-link active">All User</a>
4    </li>
5    <li class="nav-item">
6        <a href="#" class="nav-link">New User</a>
7    </li>
8    <li class="nav-item">
9        <a href="#" class="nav-link">Other User</a>
10   </li>
11   <li class="nav-item">
12       <a href="#" class="nav-link">More User</a>
13   </li>
14 </ul>

```

To the right of the code, there is a preview of the navigation bar. It has four tabs: "All User" (which is highlighted in blue), "New User", "Other User", and "More User".

နမူနာမှာ Tab တွေက ဘယ်ဘက်တစ်ခြမ်းမှာ စုဖြိုး နေရာယဉ်ထားတာပါ။ Screen အပြည့် နေရာယဉ်စေ ချင်ရင်တော့ nav-fill Class ကို သုံးပေးနိုင်ပါတယ်။

### HTML

```
<ul class="nav nav-tabs nav-fill">
  ...
</ul>
```

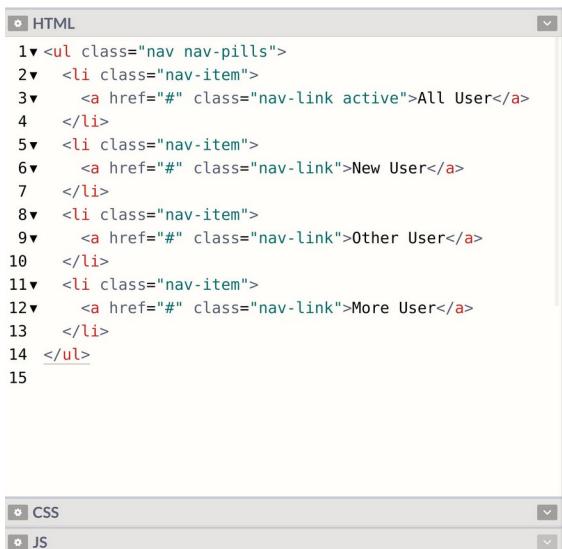
```
1▼ <ul class="nav nav-tabs nav-fill">
2▼   <li class="nav-item">
3▼     <a href="#" class="nav-link active">All User</a>
4   </li>
5▼   <li class="nav-item">
6▼     <a href="#" class="nav-link">New User</a>
7   </li>
8▼   <li class="nav-item">
9▼     <a href="#" class="nav-link">Other User</a>
10 </li>
11▼   <li class="nav-item">
12▼     <a href="#" class="nav-link">More User</a>
13 </li>
14 </ul>
15
```

nav-tabs အစား nav-pills ကိုလည်း သုံးနိုင်ပါတယ်။

### HTML

```
<ul class="nav nav-pills">
  ...
</ul>
```

တူညီတဲ့ပုံစံနဲ့ပဲအလုပ်လုပ်ပေမယ့် Tab UI ပုံစံတော့ မဟုတ်တော့ပါဘူး။ Item လေးတွေက ထောင့်ကျွေး Pill Box လေးတွေပုံစံ ဖြစ်သွားတာပါ။



The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab is active, displaying the following code:

```

1▼ <ul class="nav nav-pills">
2▼   <li class="nav-item">
3▼     <a href="#" class="nav-link active">All User</a>
4   </li>
5▼   <li class="nav-item">
6▼     <a href="#" class="nav-link">New User</a>
7   </li>
8▼   <li class="nav-item">
9▼     <a href="#" class="nav-link">Other User</a>
10  </li>
11▼ <li class="nav-item">
12▼   <a href="#" class="nav-link">More User</a>
13  </li>
14 </ul>
15

```

The CSS and JS tabs are visible but empty.

ဒီနေရာမှာ သတိပြုရမှာကတော့၊ လက်ရှိလေ့လာနေတာဟာ Tab UI တွေ Pill UI တွေရဲ့ ဖော်ပြပုံ အသွင်အပြင်ကိုသာ လေ့လာနေခြင်းဖြစ်ပါတယ်။ လက်တွေအလုပ်လုပ်ဖို့ကတော့ JavaScript နဲ့ ဆက်စပ် နည်းပညာတွေ လိုအပ်ပါသေးတယ်။ CSS ချဉ်းသက်သက်နဲ့ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။

## Badge

တစ်လက်စထဲ Notification တွေမှာ တွေ့ရလေ့ရှိပြီး Count အရေအတွက် ဖော်ပြရာမှာသုံးလေ့ရှိတဲ့ Component လေးတစ်ခုကို ဆက်ကြည့်ကြပါမယ်။ Bootstrap က Badge လိုခေါ်ပါတယ်။ စမ်းလက်စ Tab နဲ့ အခုလိုတဲ့ပြီး စမ်းကြည့်နိုင်ပါတယ်။



The screenshot shows a code editor interface with an active HTML tab, displaying the following code:

```

<ul class="nav nav-tabs">
  <li class="nav-item">
    <a href="#" class="nav-link active">
      All User
      <span class="badge bg-primary rounded-pill">20</span>
    </a>
  </li>
  ...
</ul>

```

<span> Element မှာ badge Class သုံးပေးလိုက်တာပါ။ အရောင်အတွက် bg နဲ့အတူ နှစ်သက်ရာ Color Class ကို တွဲသုံးနိုင်ပါတယ်။ နမူနာမှာပေးထားတဲ့ rounded-pill ကတော့ ပိုစိုင်းသွားအောင်

ထည့်ပေးထားတာပါ။ မထည့်လည်းရပါတယ်။ rounded-pill မပါရင်တော့ ဖော်ပြုပုံက နည်းနည်း  
လေးထောင့် ပိုဆန်နေမှာပါ။ မိမိနှစ်သက်ရာကို အသုံးပြုနိုင်ပါတယ်။ သူ့ဖော်ပြုပုံက ဒီလိုဖြစ်မှာပါ -

```

HTML
1▼ <ul class="nav nav-tabs">
2▼   <li class="nav-item">
3▼     <a href="#" class="nav-link active">
4       All User
5▼       <span class="badge bg-primary rounded-pill">20</span>
6     </a>
7   </li>
8▼   <li class="nav-item">
9▼     <a href="#" class="nav-link">New User</a>
10    </li>
11▼   <li class="nav-item">
12▼     <a href="#" class="nav-link">Other User</a>
13    </li>
14▼   <li class="nav-item">
15▼     <a href="#" class="nav-link">More User</a>
16    </li>
17 </ul>
18

```

**CSS**

**JS**

ဒါလေးကလည်း အသေးအဖွဲ့လေးပေမယ့် အသုံးဝင်တဲ့ လုပ်ဆောင်ချက်တစ်ခု ဖြစ်ပါတယ်။

## Navbar or Menubar

ဒီအခန်းမှာ လေ့လာမယ့် Component တွေထဲမှာ နောက်ဆုံးတစ်ခုအနေနဲ့ Menubar အသုံးပြုပုံကို  
လေ့လာကြပါမယ်။ Menubar ဆိုတာ ပရောဂျက်တိုင်းမှာ လိုအပ်တဲ့လုပ်ဆောင်ချက်တစ်ခု ဖြစ်ပါတယ်။  
Bootstrap မှာ Navbar လိုပေါ်ပါတယ်။ တစ်ကယ်တော့ Navbar ရဲ့ အလုပ်လုပ်ပုံ ပြီးပြည့်စုံဖို့အတွက်  
JavaScript လိုပါတယ်။ JavaScript Component တွေအကြောင်းကို နောက်တစ်ခန်းကျတော့မှ သပ်သပ်  
ပြောမှာပါ။ ဒီမှာထည့် မပြောသေးပါဘူး။ ဒါကြောင့် JavaScript မလိုတဲ့ဖော်ပြုပုံကိုပဲ မှတ်ထားပေးပါ။

Navbar တစ်ခုရရှိဖို့အတွက် ပင်မ Element မှာ Class (၄) ခု ပေးဖို့ လိုပါတယ်။ navbar, navbar-expand-{size}, navbar-{textcolor}, bg-{color} တို့ဖြစ်ပါတယ်။ နမူနာမှာ ပင်မ  
Element အနေနဲ့ <nav> ကိုသုံးပါမယ်။ Navigation Menu ဖြစ်လို <nav> နဲ့ပိုသင့်တွဲအတွက်  
<nav> ကို သုံးထားပေမယ့် <div> သုံးရင်လည်း ရပါတယ်။

**HTML**

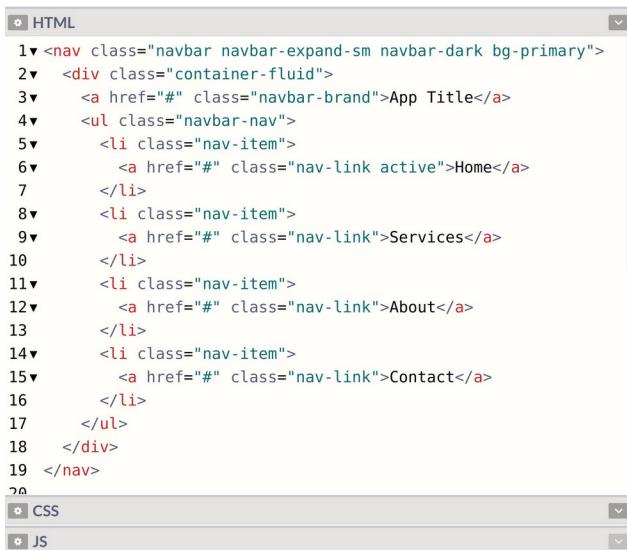
```
<nav class="navbar navbar-expand-sm navbar-dark bg-primary">

    <div class="container-fluid">

        <a href="#" class="navbar-brand">App Title</a>

        <ul class="navbar-nav">
            <li class="nav-item">
                <a href="#" class="nav-link active">Home</a>
            </li>
            <li class="nav-item">
                <a href="#" class="nav-link">About</a>
            </li>
        </ul>
    </div>
</nav>
```

navbar-expand-{size} အတွက် navbar-expand-sm လိုပေးထားပါတယ်။ sm ဆိုတာ Screen ရဲ့ Size ကိုပြောတာပါ။ တစ်ခြား lg, md စသဖြင့် Size တွေရှိပါသေးတယ်။ Layouts အခန်းရောက်တော့မှ ဒီအကြောင်းတွေ ပြောပြပါမယ်။ လောလောဆယ်တော့ ပေးထားတဲ့အတိုင်းပဲ စမ်းကြည့်ပေးပါ။ စာတွေကို အဖြူရောင်ဖော်ပြစေချင်လို့ navbar-light ကိုသုံးထားပြီး နောက်ခံအရောင်အတွက်ကတော့ bg နဲ့အတူ ကြိုက်တဲ့ Color Class ကို စွဲသုံးလို့ရပါတယ်။ စမ်းကြည့်လိုက်ရင် ရလဒ်က ဒီလိုဖြစ်မှာပါ -



The screenshot shows the browser's developer tools with the 'HTML' tab selected. The code editor on the left displays the following HTML:

```
1▼ <nav class="navbar navbar-expand-sm navbar-dark bg-primary">
2▼   <div class="container-fluid">
3▼     <a href="#" class="navbar-brand">App Title</a>
4▼   <ul class="navbar-nav">
5▼     <li class="nav-item">
6▼       <a href="#" class="nav-link active">Home</a>
7      </li>
8▼     <li class="nav-item">
9▼       <a href="#" class="nav-link">Services</a>
10     </li>
11▼     <li class="nav-item">
12▼       <a href="#" class="nav-link">About</a>
13     </li>
14▼     <li class="nav-item">
15▼       <a href="#" class="nav-link">Contact</a>
16     </li>
17   </ul>
18 </div>
19 </nav>
20
```

The browser window on the right shows the rendered navigation bar with the following menu items: Home (active), Services, About, and Contact.

အထဲမှာ container-fluid လိုပေးထားတဲ့ <div> တစ်ထပ် ပါသေးတာကို သတိပြုပါ။ အဲဒီ အကြောင်းကိုလည်း Layouts အကြောင်း ပြောတော့မှ ရှင်းပြုပါမယ်။ အခုတော့ ပေးထားတဲ့အတိုင်းပဲ စမ်းကြည့်ပေးပါ။ အထဲမှာ navbar-brand Class ကိုသုံးထားတဲ့ <a> Element တစ်ခု ပါပါတယ်။ ကိုယ့် App ရဲ့အမည်ကိုသတ်မှတ်ပေးဖို့အတွက် သုံးရတဲ့ Class ဖြစ်ပါတယ်။ ဆက်လက်ထည့်သွင်းထားတဲ့ Menu ရဲ့ဖွံ့စည်းပုံကတော့ ပြီးခဲ့တဲ့ Tab မှာတုံးက ရေးသားပုံနဲ့အတူတူပါပဲ။ nav nav-tabs အစား navbar-nav ကိုသုံးပေးရတာတစ်ခုပဲ ကွာမှာဖြစ်ပါတယ်။

လက်ရှိဖော်ပြခဲ့သမျှတွေထဲမှာ အရှုပ်ဆုံး Component ဖြစ်ပါတယ်။ ဒါတောင် အတတ်နိုင်ဆုံး မလိုတာ တွေချုန်ပြီး မဖြစ်မနေ လိုတာတွေချည်းပဲ ရွေးပေးထားတာပါ။ Navbar နဲ့ပက်သက်ပြီး နောက်ထပ် အသုံးဝင်နိုင်တဲ့ လုပ်ဆောင်ချက်ကတော့ sticky-top လုပ်ဆောင်ချက်ဖြစ်ပါတယ်။

## HTML

```
<nav class="navbar navbar-expand-sm sticky-top navbar-dark bg-primary">
...
</nav>
```

တစ်ချို့ App တွေမှာ တွေ့ဖူးပါလိမ့်မယ်။ Scroll ဆဲလိုက်တဲ့အခါ ဟိုးအပေါ်က Bar က ပျောက်မသွားဘဲ အပေါ်ဆုံးမှာ အမြဲတမ်းဖော်ပြနေတဲ့ လုပ်ဆောင်ချက်မျိုးပါ။ အဲဒီကို Sticky Top လို ခေါ်တာပါ။ စမ်းကြည့်နိုင်ဖို့အတွက် CSS နည်းနည်း ရေးထည့်ပေးရပါမယ်။

The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS.

- HTML:**

```
1▼ <nav class="navbar"
2      navbar-expand-sm
3      sticky-top
4      navbar-dark
5      bg-primary">
6▼  <div class="container-fluid">
7▼   <a href="#" class="navbar-brand">App Title</a>
8▼   <ul class="navbar-nav">
9▼     <li class="nav-item">
10▼       <a href="#" class="nav-link active">Home</a>
11     </li>
12▼     <li class="nav-item">
13▼       <a href="#" class="nav-link">Services</a>
14     </li>
15▼     <li class="nav-item">
16▼       <a href="#" class="nav-link">About</a>
17     </li>
```
- CSS:**

```
1 body { height: 2000px }
```
- JS:** (This tab is empty)

နမူနာ body ရဲ့ height ကို 2000px လိုပေးလိုက်တဲ့အတွက် Screen မှာ မဆန့်တော့လို့ Scrollbar ပေါ်လာပါလိမ့်မယ်။ Scroll ဆဲကြည့်လိုက်ရင် Navbar က ပျောက်မသွားဘဲ နေရာမှာအမြတ်မီ ရှိနေတာ ကို တွေ့ရမှာ ဖြစ်ပါတယ်။

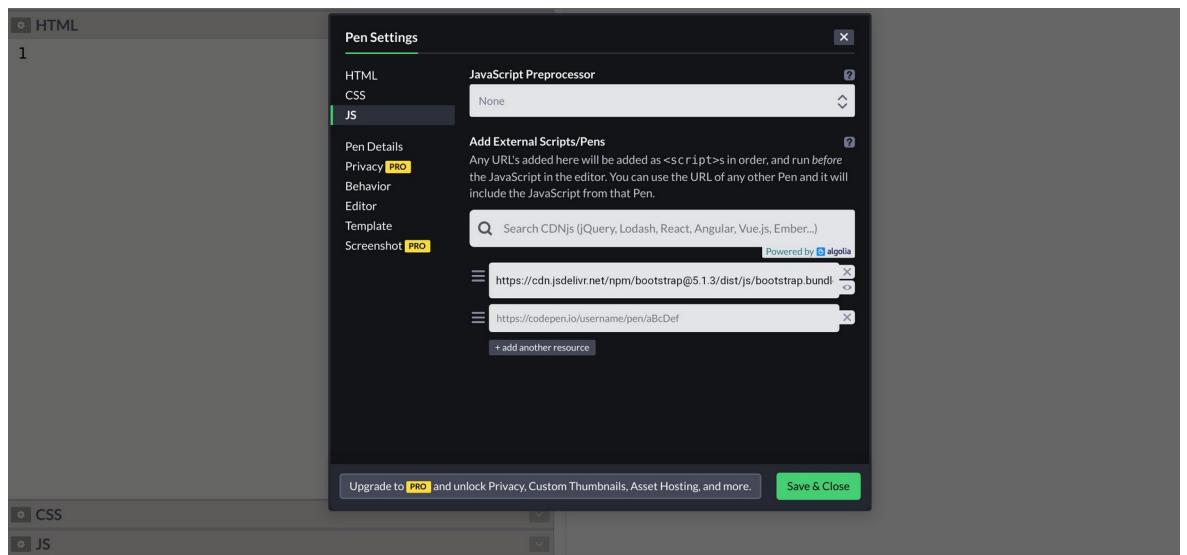
အခုခံရင် ဒီအခန်းမှာဖော်ပြချင်တဲ့ Component တွေစုံသွားပါပြီ။ ဒီလောက်လေ့လာမိမြို့ခံရင် Bootstrap ရဲ့ အကူအညီနဲ့ လက်တွေ့အသုံးဝင်တဲ့ App UI တွေကို မြန်မြန်ဆန်ဆန်နဲ့ အလွယ်တစ်ကူ ရရှိနိုင်တယ်ဆို တာကို သတိပြုမိလောက်ပါပြီ။ တစ်ချို့ အသုံးနည်းတဲ့ Component တွေတော့ ချုန်ထားခဲ့ပါတယ်။ မလိုအပ်ဘဲ မှတ်စရာတွေ များပြီး ရောကုန်မှာစိုးလိုပါ။ ဒီလောက် အစ ရသွားပြီဆိုရင် ကျုန်နေတာတွေက ကိုယ့်ဘာသာ ဆက်ကြည့်သွားလို့ ရနေပါပြီ။

JavaScript နဲ့တွဲသုံးဖို့လိုတဲ့ Component တွေရှိပါသေးတယ်။ နောက်တစ်ခန်း ခွဲပြီးတော့ ဆက်လက်ဖော်ပြပေးမှာပါ။ Layouts နဲ့ပက်သက်တဲ့အကြောင်းတွေ၊ အသုံးဝင်တဲ့ Utility Classes အကြောင်းတွေနဲ့ Icons တွေအကြောင်းလည်း ပြောဖို့ကျုန်ပါသေးတယ်။ နောက်အခန်းတွေမှာ သူ့နေရာနဲ့သူ့ ဆက်ပြီးတော့ ဖော်ပြပေးသွားပါမယ်။

## အခန်း (၅) – JavaScript Components

Bootstrap မှာ JavaScript ကို အသုံးပြုထားတဲ့ Components တွေ ပါဝါတယ်။ JavaScript အကြောင်းမ သိရင်လည်း ကိစ္စမရှိပါဘူး။ Bootstrap က JavaScript ကုဒ်တွေ ရေးစရာမလိုဘဲ သူရဲ့ JavaScript Components တွေကို အသုံးပြုလို ရအောင် စီစဉ်ပေးထားပါတယ်။ ပထမဆုံးအနေနဲ့ JavaScript Component တွေကို စမ်းသပ်အသုံးပြုနိုင်ဖို့ Bootstrap JavaScript ဖိုင်ကို CDN ကနေ ချိတ်ပေးဖို့ လိုပါ လိမ့်မယ်။ CSS တုံးကလိုပဲ Codepen ခဲ့ Setting ထဲက JS Section မှာ ထည့်ထားပေးလိုက်ရင် ရပါတယ်။ External Scripts မှာ ဒီလိပ်စာကို ထည့်ပေးရမှာပါ။

<https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js>



Setting ထဲမှာ မထည့်ဘဲ HTML ကုဒ်ထဲမှာ ထည့်သုံးချင်ရင် အခုလိုတည့်လိုရပါတယ်။

### HTML

```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"></script>
```

<script> Element ရဲ့ src Attribute တန်ဖိုးမှာ CDN ဖိုင်လိပ်စာကို ပေးလိုက်ရတာပါ။

ဖိုင်အမည်က bootstrap.bundle.min.js ပါ။ Bundle ဆိုတဲ့အသုံးအနဲ့ အကြောင်းလေး ထည့်ပြောချင်ပါတယ်။ Bootstrap အလုပ်လုပ်ဖို့အတွက် အရင် Bootstrap Version တွေမှာ JavaScript Library နှစ်ခုလိုပါတယ်။ jQuery နဲ့ Popper လိုပေါ်ကြတဲ့ နည်းပညာတွေပါ။ Bootstrap 5 မှာတော့ jQuery မလိုအပ်တော့ပါဘူး။ ဒါပေမယ့် Popper တော့ လိုပါသေးတယ်။ တစ်ကယ်တမ်း ထည့်မယ်ဆိုရင် Popper နဲ့ Bootstrap ဆိုပြီး ဖိုင်နှစ်ခု ထည့်ရမှာပါ။ ဒီတော့မှ ပြည့်စုံပြီး အလုပ်လုပ်မှာပါ။ အဲဒါကို နှစ်ခု ထည့်စရာမလိုဘဲ တစ်ခုထဲနဲ့ ပြီးသွားအောင် Bootstrap က Bundle ဆိုပြီး ပေါင်းပေးထားပါတယ်။ ဒါကြောင့် Bundle ဖိုင်ကိုသုံးလိုက်ရင် Bootstrap အပြင် Popper ပဲ တစ်ခါတည်း ပါဝင်သွားတယ် လိုနားလည်ရမှာ ဖြစ်ပါတယ်။

- <https://getbootstrap.com/>

### Dropdowns

JavaScript Component တွေထဲမှာ ပထမဆုံးလေ့လာချင်တာကတော့ Dropdown ဖြစ်ပါတယ်။ နှိပ်လိုက်တော့မှ ပေါ်လာတဲ့ Menu လေးတွေပါ။ နှိပ်ရတဲ့ခလုပ်အနေနဲ့ <button> <a> စသာဖြင့် ကြိုက်တဲ့ Element နဲ့တဲ့သုံးလိုရပါတယ်။ ဒါကြောင့် သူ့ကို Menubar လို နေရာမျိုးမှာသာမက နှိပ်လိုက်မှပေါ်လာတဲ့ Menu လိုအပ်တဲ့ မည်သည့်နေရာမှာမဆို သုံးလိုရပါတယ်။ ရေးနည်းက ဒီလိုပါ -

### HTML

```
<div class="dropdown">
    <button class="btn btn-primary dropdown-toggle"
           data-bs-toggle="dropdown">Send Now</button>

    <ul class="dropdown-menu">
```

```

<li><a href="#" class="dropdown-item">Send Now</a></li>
<li><a href="#" class="dropdown-item">Save Draft</a></li>
<li><a href="#" class="dropdown-item">Preview</a></li>
</ul>
</div>

```

သုံးရတဲ့ Class နည်းနည်းများပါတယ်။ ပထမဆုံးအနေဖြင့် ပင်မ Element မှာ dropdown Class ကို သတ်မှတ်ပေးရပါတယ်။ အထဲမှာ ခလုပ်တစ်ခုနဲ့ List တစ်ခုပါပါတယ်။ ခလုပ်အတွက် dropdown-toggle Class ကို သတ်မှတ်ပေးထားပြီး List အတွက် dropdown-menu ကို သတ်မှတ်ပေးထားပါတယ်။ ပြီးတော့မှ List ထဲက <a> Element တွေမှာ dropdown-item Class ကို ပေးလိုက်ရင် ပြည့်စုံသွားပါပြီ။ ဒါက အသွင်အပြင်ပဲ ရှိပါသေးတယ်။ တစ်ကယ် အလုပ်မလုပ်သေးပါဘူး။ နှိပ်လိုက်မှ ပေါ်လာတဲ့ အလုပ်ကို လုပ်ပေးဖို့အတွက် data-bs-toggle=dropdown ဆိုတဲ့ Attribute ကိုထည့်ပေးရပါတယ်။ Bootstrap က JavaScript ကုဒ်တွေ ရေးစရာမလိုဘဲ JavaScript Component တွေကို သုံးလို ရအောင် လုပ်ပေးတယ်ဆိုတာ အဲဒီလို Attribute တွေနဲ့ လုပ်ပေးထားတာပါ။ ဒီ Attribute ပါရင် ကိုယ်ဘက်က ကုဒ်တွေထပ်ရေးပေးရာ မလိုတော့ဘဲ၊ နှိပ်လိုက်ရင် Dropdown Menu ကို ပြရမယ်ဆိုတာ Bootstrap က သိသွားပါပြီ။



ရလဒ်နမူနာမှာ <hr> Element တစ်ခုကိုသုံးပြီး dropdown-divider Class သတ်မှတ်ပေးထားတာ ကိုလည်း သတိပြုပါ။ ဒီလို သတ်မှတ်ပေးထားတဲ့အတွက် Menu အတွင်းမှာ Item တွေကို ပိုင်းခြားပြီး ပြပေးတာကို တွေ့ရပါမယ်။ နောက်တစ်ခုအနေဖြင့်ထပ်စမ်းကြည့်ချင်ရင် Dark Menu ကိုစမ်းကြည့်ပါ။

The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab is active, displaying the following code:

```

1▼ <div class="dropdown">
2▼   <button class="btn btn-dark dropdown-toggle"
3     data-bs-toggle="dropdown">Send Now</button>
4
5▼   <ul class="dropdown-menu dropdown-menu-dark">
6▼     <li><a href="#" class="dropdown-item">Send Now</a></li>
7▼     <li><a href="#" class="dropdown-item">Save Draft</a></li>
8     <li class="dropdown-divider"></li>
9▼     <li><a href="#" class="dropdown-item">Preview</a></li>
10    </ul>
11  </div>
12

```

To the right of the code editor, a live preview window shows a dropdown menu with three items: "Send Now", "Save Draft", and "Preview". The "Send Now" item is highlighted.

ကျန်တဲ့ကုဒ်တွေအတူတူပါပဲ ခလုပ်က btn-dark ဖြစ်သွားပြီး Dropdown Menu မှာ dropdown-menu-dark ဆိုတဲ့ Class တစ်ခုထပ်ပါသွားတာပါ။ ဒီနည်းနဲ့ Bootstrap ရဲ့ Dropdown လုပ်ဆောင်ချက်ကို Menubar တွေ၊ Toolbar တွေ၊ Form တွေနဲ့ တစ်ခြားလိုအပ်တဲ့ နေရာတွေမှာ ထည့်သုံးလို့ ရပါတယ်။

## Collapses

Collapse ကလည်း Dropdown နဲ့ ဆင်ပါတယ်။ သူလည်းပဲ နိုပ်လိုက်မှ ပေါ်လာမယ့် Component တစ်ခုပါပဲ။ Menu မဟုတ်တော့ဘဲ ကြိုက်တဲ့ Component နဲ့ တဲ့သုံးရတာ ဖြစ်သွားပါတယ်။ ဒီလိုပါ –

The screenshot shows a code editor interface with one tab: HTML. The HTML code is as follows:

```

<p>
    <a class="btn btn-primary" data-bs-toggle="collapse" href="#item">
        Link Button
    </a>
</p>

<div class="collapse" id="item">
    <div class="card card-body">
        <h5 class="card-title"></h5>
        <p>Some sample content</p>
    </div>

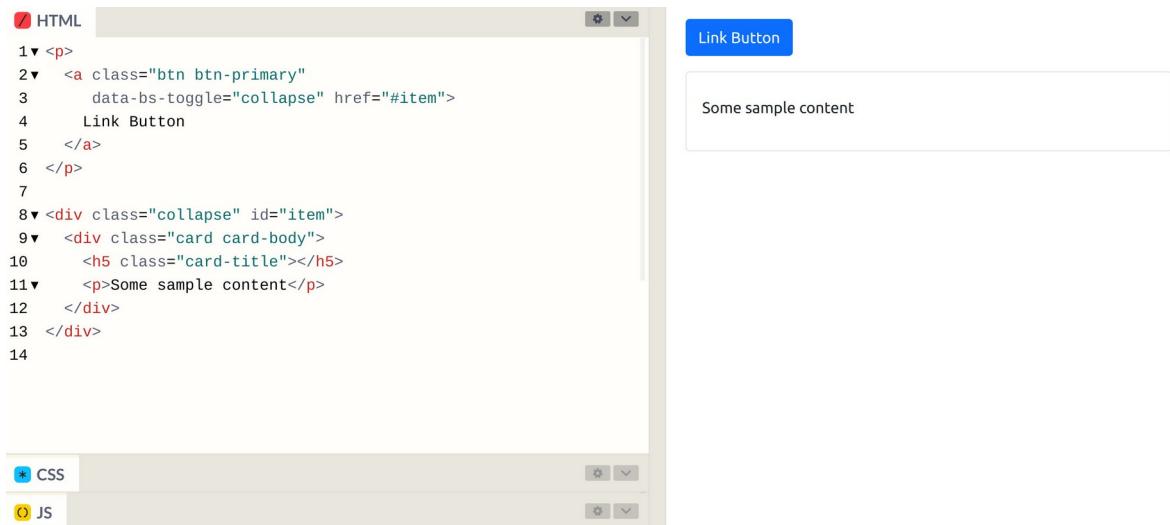
```

```
</div>
```

နမူနာမှာ နိုပ်တဲ့ ခလုပ်အနေနဲ့ <a> Element တစ်ခုကို သုံးထားပါတယ်။ href မှာ နိုပ်လိုက်ရင် ပြရမယ့် Element ရဲ့ ID ကို ပေးထားတာ သတိပြုပါ။ လိုအပ်ပါတယ်။ အကယ်၍ <a> အစား <button> ကိုသုံးချင်တယ်ဆိုရင်လည်း ရပါတယ်။ Button မှာ href Attribute မရှိပေမယ့် data-bs-target Attribute ကို အစားထိုးပြီး သုံးနိုင်ပါတယ်။ နိုပ်လိုက်မှ ပေါ်လာစေချင်တဲ့ Element မှာ ID တစ်ခုရှိဖို့လိုပြီး ခလုပ်ကနေ ညွှန်းထားတဲ့ ID နဲ့ တူဖို့လိုပါတယ်။ ပြီးတဲ့ အခါ collapse Class ကို သတ်မှတ်ပေးထားရ မှာ ဖြစ်ပါတယ်။ အထဲမှာ ကြိုက်တာထည့်လို့ ရသွားပါပြီ။ နမူနာမှာ တော့ card တစ်ခုကိုထည့်ပြထားပါတယ်။

သူမှာလည်း JavaScript လုပ်ဆောင်ချက်ကို ရရှိဖို့ အတွက် data-bs-toggle Attribute ကိုသုံးထားတာ သတိပြုပါ။ Dropdown အတွက် data-bs-toggle ကို dropdown လိုသတ်မှတ်ပေးခဲ့ရသလိုပဲ Collapse အတွက်တော့ data-bs-toggle ကို collapse လိုသတ်မှတ်ပေးရခြင်း ဖြစ်ပါတယ်။

စမ်းကြည့်လိုက်ရင် ရလဒ်က အခုလိုရရှိမှာ ဖြစ်ပါတယ်။



```

 HTML
1▼ <p>
2▼   <a class="btn btn-primary"
3     data-bs-toggle="collapse" href="#item">
4     Link Button
5   </a>
6 </p>
7
8▼ <div class="collapse" id="item">
9▼   <div class="card card-body">
10    <h5 class="card-title"></h5>
11▼   <p>Some sample content</p>
12   </div>
13 </div>
14

```

Link Button

Some sample content

CSS

JS

သူလည်းပဲ တော်တော်အသုံးဝင်ပါတယ်။ အတိုကောက် Summary လေးပဲ ပြထားပြီး ခလုပ်နှိုပ်လိုက်တော့ မှ Detail အပြည့်အစုံ ပေါ်လာတယ် ဆိုတဲ့ လုပ်ဆောင်ချက်မျိုးက မကြာမကြာ လိုအပ်တတ်ပါတယ်။ အဲဒီ

လို လိုအပ်လာတဲ့အခါ Collapse Components တွက် အသုံးပြုနိုင်မှာပါ။

## Modals

ဆက်လက်လေ့လာမှာကတော့ Modal Component ဖြစ်ပါတယ်။ သူလည်းပဲ နှိပ်မှပေါ်မယ့် အရာတစ်ခုပါပဲ။ သူကတော့ Dialog Box တစ်ခုအနေနဲ့ Page တစ်ခုလုံးပေါ်မှာ ဖုံးလွှမ်းပြီး ဖော်ပြုမယ့် လုပ်ဆောင်ချက်ပါ။ ရေးရမယ့်ကုဒ်တော့ နည်းနည်းများပါတယ်။ များလွန်းလို့ မျက်စီမလည်ရအောင် အတတ်နိုင်ဆုံး ပြောပြေးပါမယ်။ ဂရိုစိုက်ကြည့်ပေးပါ။

### HTML

```
<button class="btn btn-primary"
        data-bs-toggle="modal"
        data-bs-target="#feedback">Show Modal</button>

<div class="modal" id="feedback">

    <div class="modal-dialog">
        <div class="modal-content">

            <div class="modal-header">
                <h5 class="modal-title">Feedback</h5>
                <button class="btn-close"
                        data-bs-dismiss="modal"></button>
            </div>

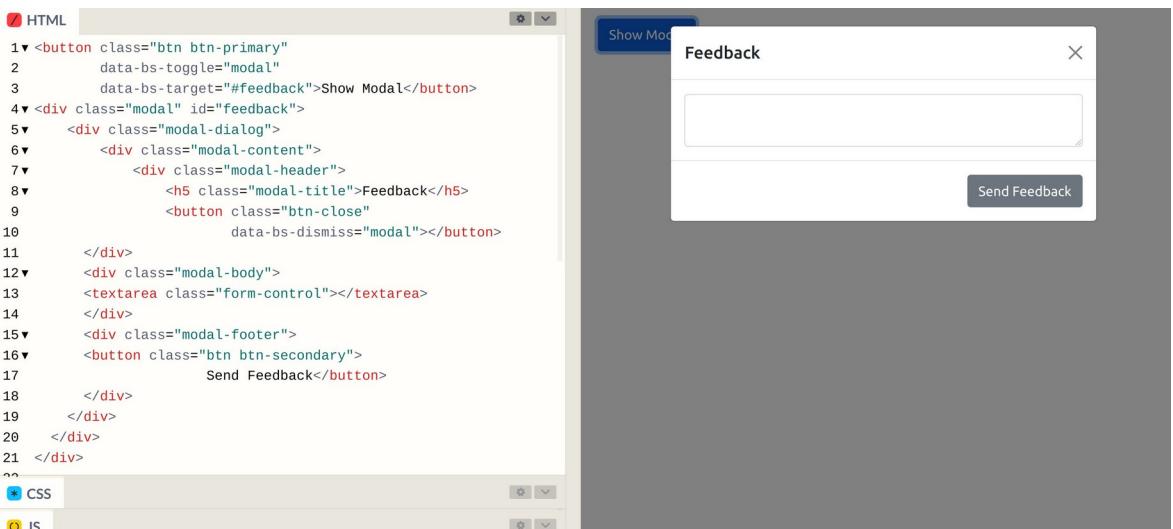
            <div class="modal-body">
                <textarea class="form-control"></textarea>
            </div>

            <div class="modal-footer">
                <button class="btn btn-secondary">
                    Send Feedback</button>
            </div>
        </div>
    </div>
</div>
```

data-bs-toggle မှာ modal လိုသတ်မှတ်ထားတဲ့ခလုပ်တစ်ခုပါပါတယ်။ ဒါကြောင့်သူကိုနိုင်ရင် Modal Dialog ကို ပြေားမှာပါ။ data-bs-target နဲ့ ပြရမယ့် Modal ရဲ့ ID ကိုညွှန်းပေးထားတာ သတိပြုပါ။

ပြီးတဲ့အခါ Modal Dialog Component ကို ဆက်လက်ရေးသားပါတယ်။ Class ကို modal လို့သတ်မှတ်ပြီး အပေါ်ကခလုပ်မှာ ညွှန်းထားတဲ့ ID နဲ့ တူညီတဲ့ id ကိုပေးထားပါတယ်။ အထဲမှာတော့ (၃) ထပ်ဖြစ်နေတာကို တွေ့ရပါလိမ့်မယ်။ modal-dialog → modal-content → modal-body တို့ဖြစ်ပါတယ်။ modal-header နဲ့ modal-footer တို့ကိုထည့်သုံးလို့ ရတဲ့အတွက် သုံးပြထားပါတယ်။

modal-header အတွင်းထဲမှာ ခေါင်းစဉ်အဖြစ်ဖော်ပြစ်လိုတဲ့ Element ကို modal-title Class ပေးထားတာလည်း သတိပြုပါ။ ပြီးတဲ့အခါ Close Button တစ်ခုလည်း ပါဝါသေးတယ်။ btn-close Class ကိုသုံးထားပြီး နှိပ်လိုက်ရင် Modal ကို ပိတ်ပေးစေဖို့အတွက် data-bs-dismiss=modal လို့လည်း သတ်မှတ်ထားပါသေးတယ်။ ဒါကြောင့် နှိပ်လိုက်ရင် Modal ကို ပြန်ပိတ်ပေးသွားမှာပဲဖြစ်ပါတယ်။



Modal Body ထဲမှာပြတဲ့ Content ကတော့ ကိုယ်ကြိုက်တာ ပြလို့ရပါတယ်။ ဘာဖြစ်ရမယ်ဆိုတဲ့ ကနဲ့သတ်ချက်လျှိုးမရှိလို့ ကြိုက်တဲ့ Component ကို ထည့်သုံးနိုင်ပါတယ်။ နမူနာမှာတော့ <textarea> တစ်ခုကို ထည့်ပြထားပါတယ်။ နှိပ်လိုက်လို့ပေါ်လာတဲ့အခါ ဒီအတိုင်းပေါ်မလာဘဲ Animation Effect လေးနဲ့ ပေါ်လာစေချင်ရင် fade Class ကိုသုံးနိုင်ပါတယ်။ ဒီလိုပါ -

**HTML**

```
<button class="btn btn-primary"
        data-bs-toggle="modal"
        data-bs-target="#feedback">Show Modal</button>

<div class="modal fade" id="feedback">
    ...
</div>
```

ဒါဆိုရင် Modal Dialog Box ကိုပြတဲ့အခါ Fade Effect ကိုသုံးပေးတဲ့အပြင် Box ကအပေါ်ကနေ ကျလာတဲ့ပုံစံလေးနဲ့ ပြပေးမှာဖြစ်ပါတယ်။ ကိုယ်တိုင်သာ ထည့်ပြီးစမ်းကြည့်လိုက်ပါ။

**Carousels**

Carousel Components ကိုတော့ Slideshow သဘောမျိုး တစ်ခုပြီးတစ်ခု ပြောင်းပြတဲ့ လုပ်ဆောင်ချက်မျိုး လိုအပ်တဲ့အခါ သုံးနိုင်ပါတယ်။ ရေးပုံရေးနည်းက ဒီလိုပါ -

**HTML**

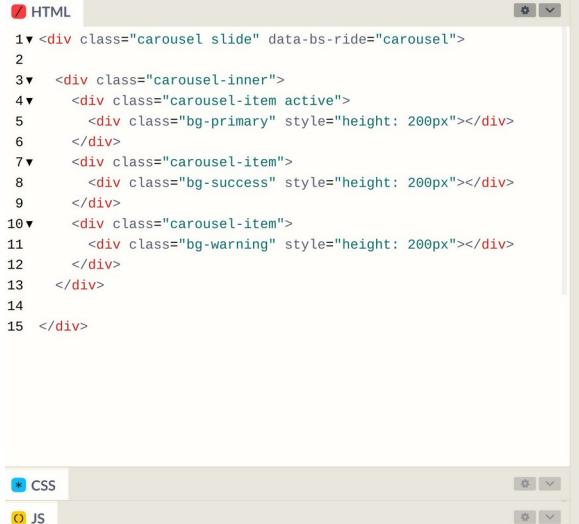
```
<div class="carousel slide" data-bs-ride="carousel">

    <div class="carousel-inner">
        <div class="carousel-item active">
            <div class="bg-primary" style="height: 200px"></div>
        </div>
        <div class="carousel-item">
            <div class="bg-success" style="height: 200px"></div>
        </div>
        <div class="carousel-item">
            <div class="bg-warning" style="height: 200px"></div>
        </div>
    </div>

</div>
```

carousel → carousel-inner → carousel-item ဆိုပြီးတော့ (၃) ဆင့်ရှုပါတယ်။ ပင်မ Element မှာပါတဲ့ slide ကတော့ Slide Effect အတွက်ပါ။ မထည့်လည်း ရပါတယ်။ မထည့်ရင် Effect ပါမှာ မဟုတ်တော့ပါဘူး။ carousel-item တဲ့မှာတော့ နမူနာအနေနဲ့ Inline Style ကိုသုံးပြီးတော့ height တွေသတ်မှတ်ထားတဲ့ <div> အလွတ်တွေ ပေးထားပါတယ်။ active Class ကိုသုံးပြီး ပထားစပေါ်စေချင်တဲ့ Slide Item ကိုသတ်မှတ်ထားတာကိုလည်း တွေ့ရမှာဖြစ်ပါတယ်။

data-bs-ride=carousel Attribute ကို သုံးထားတဲ့အတွက် Slide Item တွေကို (၅) စဉ်နဲ့ကြား  
တိုင်း အလိုအလျောက် တစ်ခုပြောင်းပြတဲ့လုပ်ဆောင်ချက်ကို ရရှိသွားမှာ ဖြစ်ပါတယ်။



```

1▼ <div class="carousel slide" data-bs-ride="carousel">
2
3▼   <div class="carousel-inner">
4▼     <div class="carousel-item active">
5       <div class="bg-primary" style="height: 200px"></div>
6     </div>
7▼     <div class="carousel-item">
8       <div class="bg-success" style="height: 200px"></div>
9     </div>
10▼    <div class="carousel-item">
11      <div class="bg-warning" style="height: 200px"></div>
12    </div>
13  </div>
14
15 </div>

```



Slide တွေ ရှေ့နောက်ပြောင်းစေချင်ရင် ခလုပ်တွေထည့်လိုရပါတယ်။ ဒီလိုထည့်ရပါတယ်။

### HTML

```

<div class="carousel slide" data-bs-ride="carousel" id="slide">
  ...
  <a href="#slide" class="carousel-control-prev" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </a>

  <a href="#slide" class="carousel-control-next" data-bs-slide="next">
    <span class="carousel-control-next-icon"></span>
  </a>
</div>

```

ပင်မ Element မှာ ID ပါသွားတာကို အရင်သတိပြုပါ။ ပြီးတဲ့အခါ <a> Element တွေနဲ့ အဲဒီ ID ကိုချိတ်  
ပြီး Previous, Next ခလုပ်တွေ ထည့်ထားပါတယ်။ ခလုပ်ထဲမှာ များပုံလေးတွေ ပေါ်စေချင်တဲ့အတွက်  
carousel-control Icon တွေကို ထည့်ပေးထားပါတယ်။ ခလုပ်တွေကို နှိပ်လိုက်ရင် Slide ပြောင်း

စော့အတွက် data-bs-slide Attribute ကိုသုံးပေးထားပါတယ်။

ဒီလို့ Slide Carousel Component တွေမှာ Indicator ဆိုတဲ့လုပ်ဆောင်ချက်လည်း ပါလေ့ရှိပါတယ်။  
လက်ရှိ ဘယ် Slide ကို ရောက်နေပြီလဲဆိုတာကို ပြပေးတဲ့ လုပ်ဆောင်ချက်ပါ။ ဒီလို့ထည့်ပေးရပါတယ်။

### HTML

```
<div class="carousel slide" data-bs-ride="carousel" id="slide">

    <ol class="carousel-indicators">
        <li data-bs-target="#slide" data-bs-slide-to="0"
            class="active"></li>
        <li data-bs-target="#slide" data-bs-slide-to="1"></li>
        <li data-bs-target="#slide" data-bs-slide-to="2"></li>
    </ol>

    ...
</div>
```

နံပါတ်စဉ် 1, 2, 3 အစီအစဉ်အတိုင်းပြမယ့်သဘောမြို့လို့ <ol> Element ကိုသုံးထားပါတယ်။ <ul> သုံးလည်း ရတော့ရပါတယ်။ carousel-indicators Class သတ်မှတ်ပေးပြီး အထဲက <li> Element တွေမှာ data-bs-target နဲ့ ID ကို ချိတ်ပေးရတာပါ။ သူတို့ကို နှိပ်ရင်လည်း နှိပ်လို့စော့အတွက် data-bs-slide-to နဲ့ နှိပ်လိုက်ရင် ပြရမယ့် Slide နံပါတ်ကို သတ်မှတ်ပေးထားနိုင်ပါတယ်။



```
✓ HTML
1 <div class="carousel slide" data-bs-ride="carousel" id="slide">
2
3     <ol class="carousel-indicators">
4         <li data-bs-target="#slide" data-bs-slide-to="0"
5             class="active"></li>
6         <li data-bs-target="#slide" data-bs-slide-to="1"></li>
7         <li data-bs-target="#slide" data-bs-slide-to="2"></li>
8     </ol>
9
10    <div class="carousel-inner">
11        <div class="carousel-item active">
12            <div class="bg-primary" style="height: 200px"></div>
13        </div>
14        <div class="carousel-item">
15            <div class="bg-success" style="height: 200px"></div>
16        </div>
17        <div class="carousel-item">
18            <div class="bg-warning" style="height: 200px"></div>
19        </div>
20    </div>
21
22    <a href="#slide" class="carousel-control-prev" data-bs-slide="prev">
23        <span class="carousel-control-prev-icon"></span>
24    </a>
25    <a href="#slide" class="carousel-control-next" data-bs-slide="next">
26        <span class="carousel-control-next-icon"></span>
27    </a>
28 </div>
```



နမူနာရလဒ်မှာ ရှေ့နောက် Previous, Next သွားလိုရတဲ့ များလေးတွေနဲ့ အောက်နားမှာ Indicator လေးတွေကို တွေ့မြင်ရမှာဖြစ်ပါတယ်။ အားလုံးက နိပ်ရင်အလုပ်လုပ်တဲ့ လုပ်ဆောင်ချက်လေးတွေ ဖြစ်ပါတယ်။ ကိုယ်တိုင်စမ်းသပ်ရတာ အဆင်ပြေဖော် ကုဒ်အပြည့်အစုံကို ထပ်ပြီးတော့ ဖော်ပြပေးလိုက်ပါတယ်။

### HTML

```
<div class="carousel slide" data-bs-ride="carousel" id="slide">

    <ol class="carousel-indicators">
        <li data-bs-target="#slide" data-bs-slide-to="0" class="active"></li>
        <li data-bs-target="#slide" data-bs-slide-to="1"></li>
        <li data-bs-target="#slide" data-bs-slide-to="2"></li>
    </ol>

    <div class="carousel-inner">
        <div class="carousel-item active">
            <div class="bg-primary" style="height: 200px"></div>
        </div>
        <div class="carousel-item">
            <div class="bg-success" style="height: 200px"></div>
        </div>
        <div class="carousel-item">
            <div class="bg-warning" style="height: 200px"></div>
        </div>
    </div>

    <a href="#slide" class="carousel-control-prev" data-bs-slide="prev">
        <span class="carousel-control-prev-icon"></span>
    </a>
    <a href="#slide" class="carousel-control-next" data-bs-slide="next">
        <span class="carousel-control-next-icon"></span>
    </a>
</div>
```

ရေးရတာများပေမယ့် အားလုံးကသူအမိပိုယ်လေးတွေနဲ့ သူမြို့လို မှတ်ရတော့ မခက်လုပါဘူး။ ချက်ခြင်း အကုန်မှတ်မိမိ မလွယ်ပေမယ့် နမူနာ နှစ်ခုသုံးခုလောက် ရေးစမ်းလိုက်ရင်တော့ မှတ်မိသွားမှာပါ။ အခုလို ပြည့်စုံတဲ့ လုပ်ဆောင်ချက်တစ်ခုကို ကိုယ့်ဘာသာလုပ်စရာမလိုဘဲ၊ အလွယ်တစ်ကူ အသုံးချခွင့်ရတာဟာ တော်တော် အသုံးဝင်တာပါ။

## Spinners

ဒီအခန်းမှာ ဖော်ပြချင်တဲ့ JavaScript Component တွေထဲမှာ နောက်ဆုံးတစ်ခုအနေဖြင့် Loading Spinners အကြောင်းကို ကြည့်ကြပါမယ်။ တစ်ခုခု Loading လုပ်နေစဉ်မှာ အစိုင်းလေး လည်နေတာမျိုး ကို တွေ့ဖူးကြပြီးသားပါ။ Bootstrap မှာ အဲဒီလို Spinner တွေကို အလုယ်တစ်ကူ ထည့်လိုပါတယ်။

### HTML

```
<span class="spinner-border text-primary"></span>
<span class="spinner-border text-success"></span>
<span class="spinner-border text-warning"></span>
```

spinner-border Class ကိုသုံးပေးလိုက်ရင် လိုချင်တဲ့ Spinner ရနေပါပြီ။ အရောင်ပြောင်းချင်ရင်သာ text-{color} Class တွေနဲ့ တွဲသုံးဖို့ လိုတာပါ။ တစ်ကယ်တော့သူက တော်တော်ရှင်းပါတယ်။ CSS Components တွေထဲမှာ ထည့်ပြာခဲ့ရင်တောင် ရပါတယ်။ ဒါပေမယ့် JavaScript နဲ့တွဲအသုံးများလို့သာ အခုမှ ထည့်ပြာလိုက်တာပါ။

```
HTML
1 <span class="spinner-border text-primary"></span>
2 <span class="spinner-border text-success"></span>
3 <span class="spinner-border text-warning"></span>
4 <span class="spinner-border text-danger"></span>
5
6▼ <div class="mt-5 mb-5">
7▼   <button class="btn btn-primary" disabled>
8     <span class="spinner-border
9       spinner-border-sm"></span>
10▼    <span class="ml-2">Loading ...</span>
11   </button>
12 </div>
13
14 <span class="spinner-grow text-primary"></span>
15 <span class="spinner-grow text-success"></span>
16 <span class="spinner-grow text-warning"></span>
17 <span class="spinner-grow text-danger"></span>
18
```

CSS

JS

နမူနာရလဒ်မှာ <button> တစ်ခုနဲ့လည်း တွဲသုံးပြထားပါတယ်။ Button ကို နိုင်လိုက်တဲ့အခါ အလုပ် လုပ်နေစဉ် Button ကို Disable ခေါ်လုပ်ပြီး Loading ပြကြတာ ထုံးစံမို့လိုပါ။ ဒါကြောင့် <button> Element မှာ disabled Attribute ပါတာကို သတိပြုပါ။ disabled Attribute က Bootstrap နဲ့ မဆိုင်ပါဘူး။ HTML Attribute တစ်ခုဖြစ်ပါတယ်။

ပြီးတော့ spinner-grow ဆိုတဲ့ အလားတူလုပ်ဆောင်ချက်လည်း ရှိပါသေးတယ်။ သူကတော့ အစိုင်းလေး လည်နေတာ မဟုတ်တော့ဘဲ အစိုင်းလေးက ကြီးလိုက်သေးလိုက်နဲ့ Effect ကိုဖော်ပြပေးမှာ ဖြစ်ပါတယ်။

ဒီလောက်ဆိုရင် အသုံးများမယ့် Components တွေ စုံသလောက် ဖြစ်သွားပါပြီ။ JavaScript Component တွေထဲမှာ အသုံးဝင်ပေမယ့် JavaScript ကုဒ်တစ်ချို့ မဖြစ်မနေ ထည့်ရေးပေးဖို့လိုတဲ့ လုပ်ဆောင်ချက်တစ်ချို့တော့ ကျန်ပါသေးတယ်။ ဒီစာအုပ်မှာ JavaScript အကြောင်းကို ထည့်သွင်း မဖော်ပြနိုင်လို့ အဲဒီလုပ်ဆောင်ချက်တွေတော့ ချုပ်ထားခဲ့လိုက်ပါတယ်။

နောက်တစ်ခန်းမှာ Layouts တွေအကြောင်း ဆက်လက်ဖော်ပြပါမယ်။

## အခန်း (၆) – Bootstrap Layouts

Bootstrap Layouts အကြောင်းမပြောခင် Responsive Web Design လိုခေါ်တဲ့ သဘောသဘာဝတစ်ခု အကြောင်းကို အရင်ပြောချင်ပါတယ်။ Responsive Web Design ဆိတာ လိုရင်းအနှစ်ချုပ်ကတော့ Device အရွယ်အစား ပြောင်းသွားရင် Layout က အလိုအလျောက် ပြောင်းပြီး ပြပေးနိုင်အောင် ဖန်တီးတဲ့ နည်းစနစ် ဖြစ်ပါတယ်။ တစ်ခုနဲ့တစ်ခု အရွယ်အစားမတူကြတဲ့ Device တွေမှာ ကိုယ့်ဝဘ်ဆိုက်နဲ့ App တွေကို ဖွင့်လိုက်တဲ့အခါ ကွန်ပျူးတာအတွက် လုပ်ထားလို ဖုန်းနဲ့ကြည့်လိုမရဘူး၊ ဖုန်းအတွက် လုပ်ထားလို Tablet နဲ့ကြည့်လိုမရဘူးဆိုတာမျိုး မဖြစ်စေဖို့အတွက်ပါ။ Layout လေးတစ်ခု အခုလိုရှိတယ် ဆိုကြပါစို့။

### HTML

```
<section>
  <nav></nav>
  <main></main>
  <aside></aside>
</section>
```

### CSS

```
section {
  display: flex;
}

nav, main, aside {
  height: 400px;
  background: cyan;
  margin: 10px;
  flex-grow: 1;
}

main {
  flex-grow: 3;
}
```

<section> ရဲအတွင်းထဲမှာ <nav><main><aside> ဆိုပြီး Layout Element တွေရှိနေပါတယ်။ section ရဲ display ကို flex လိုပြောထားတဲ့အတွက် nav, main နဲ့ aside တို့ကို Column Layout နဲ့ပြေားမှာပါ။ စမ်းကြည့်လိုက်ရင် ရလဒ်က ဒီလိုပါ –

```

HTML
1▼ <section>
2   <nav></nav>
3   <main></main>
4   <aside></aside>
5 </section>

CSS
1▼ section {
2   display: flex;
3 }
4▼ nav, main, aside {
5   height: 400px;
6   background: cyan;
7   margin: 10px;
8   flex-grow: 1;
9 }
10▼ main {
11   flex-grow: 3;
12 }

JS

```

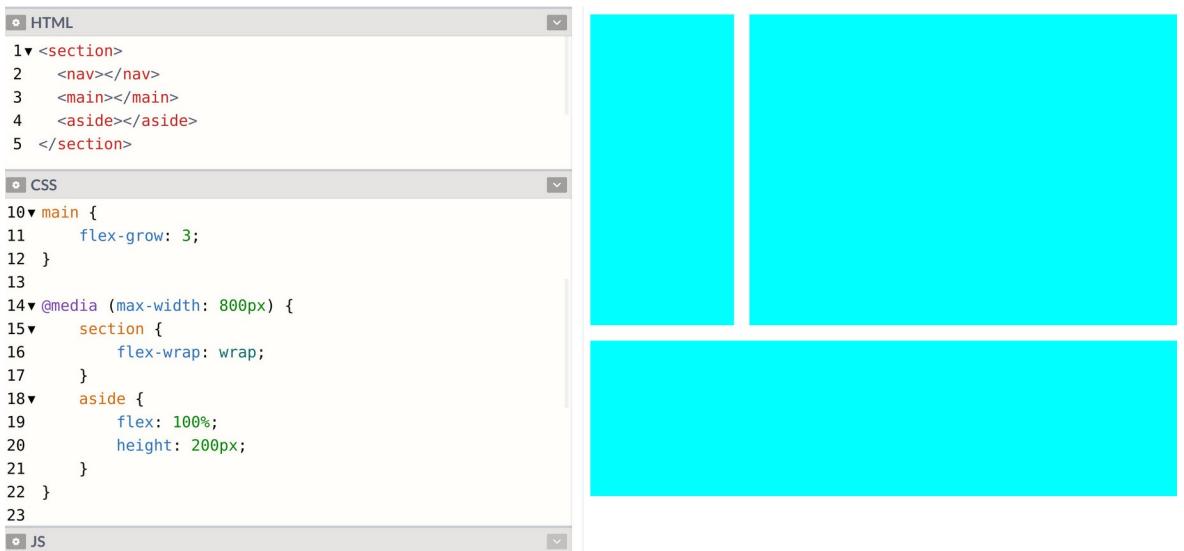
အားလုံးအတွက် height, background, margin တွေ ကိုယ်စိသတ်မှတ်ပြီး flex-grow မှာ 1 လို ပြောထားတဲ့အတွက် ပထမ width တွေက ရွယ်တူပါ။ နောက်မှ main အတွက် flex-grow တန်ဖိုး 3 လို ပြောင်းပေးလိုက်တဲ့အတွက် သူက သူများတွေရဲ့ (၃) ဆဖြစ်နေတာပါ။ ဒါဟာ မကြာမကြာ တွေရတဲ့ 3 Columns Layout တစ်ခုပုံးမျိုးပါပဲ။

ပြဿနာက၊ ဒီ Layout ဟာ Screen အရွယ်အစားကြီးတဲ့ ကွန်ပျူးတာတွေမှာ အဆင်ပြေပေမယ့် Screen အရွယ်အစားသေးတဲ့ Tablet တွေ၊ ဖုန်းတွေမှာတော့ အဆင်ပြေမှာ မဟုတ်ပါဘူး။ ဒါကြောင့် အသုံးပြုတဲ့ Screen ရဲ့ အရွယ်အစားပေါ်မှုတည်ပြီး သင့်တော်သလို ပြောင်းပြန့်လိုပါတယ်။ ဒီလိုလေး ထပ်ထည့် လိုက်ပါမယ်။

**CSS**

```
@media (max-width: 800px) {
    section {
        flex-wrap: wrap;
    }
    aside {
        flex: 100%;
        height: 200px;
    }
}
```

Media Query လိုခေါ်တဲ့ CSS ရေးထံးကို သုံးလိုက်တာပါ။ @media ကိုသုံးပြီးတော့ ရေးရပါတယ်။ max-width မှာ 800px လိုပြောထားတဲ့အတွက် Screen Width အရွယ်အစား 800px အောက်ရောက်တော့ မူ ဒါ CSS တွေ အလုပ်လုပ်မှာပါ။ ဒါကြောင့် စမ်းကြည့်လိုက်ရင် အခုလုပ်လိမ့်မယ်။



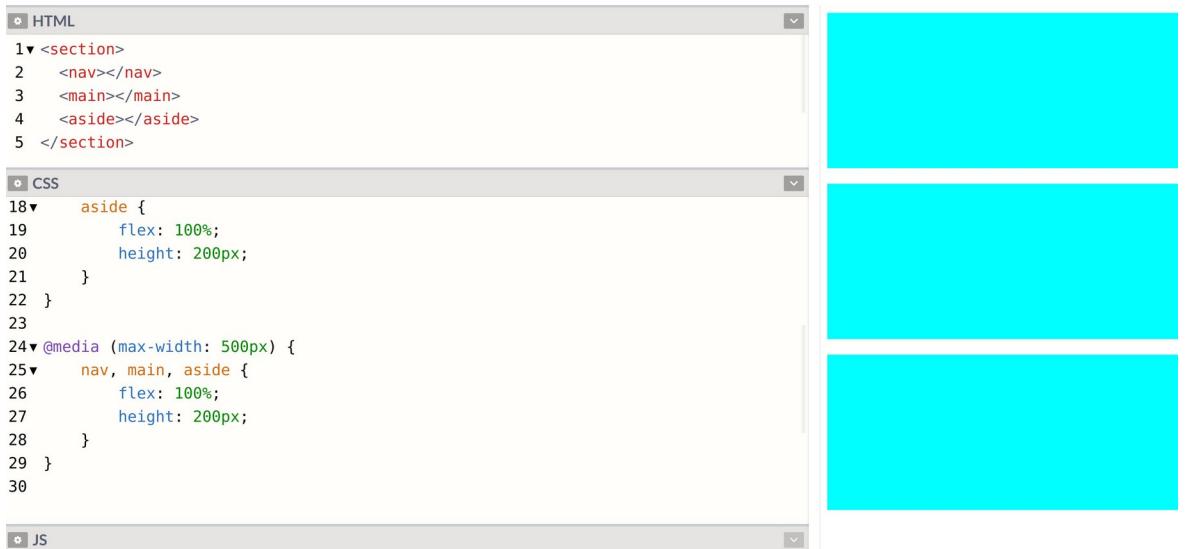
aside ရဲ့ flex တန်ဖိုး 100% ဆိုတော့ သူ့တစ်ခုထဲ အပြည့်ပြသွားတာပါ။ ဒီလိုပြတဲ့အခါ နောက်တစ်လိုင်း ဆင်းပြီးပြစေဖို့အတွက် section ရဲ့ flex-wrap ကို wrap လိုပြောထားတာ ဖြစ်ပါတယ်။ ဒါကြောင့် Table အရွယ်အစားလောက်ဆိုရင် အဆင်ပြသွားပါပြီ။ ဂွန်ပျူးတာမှာ Column (၃) ခုနဲ့ပြမာဖြစ်ပြီး Tablet မှာဆုံးရင်တော့ Column (၂) ခုနဲ့ပြပေးသွားမှာ ဖြစ်ပါတယ်။ Codepen ထဲမှာပဲ ရလဒ်ဖော်ပြတဲ့ ရော်ယာကို အကျဉ်းအကျယ် ပြောင်းပြီး စမ်းကြည့်နိုင်ပါတယ်။

Tablet အတွက် အဆင်ပြေသွားပေမယ့် ဖုန်းလို Screen အရမ်းသေးတဲ့အခါမှာတော့ အဆင်ပြေခြီးမှာ မဟုတ်ပါဘူး။ ဒါကြောင့် အခုလိုလေး ထပ်ထည့်ပေးလိုက်ပါမယ်။

## CSS

```
@media (max-width: 500px) {
    nav, main, aside {
        flex: 100%;
        height: 200px;
    }
}
```

ဒီတစ်ခါတော့ Screen Width က 500px အောက်ဆိုရင် လုပ်ရမယ့် CSS တွေကို ပေးထားတာပါ။



nav, main, aside အားလုံးကို flex: 100% လိုပြောလိုက်တာပါ။ ဒါကြောင့် အပြည့်နေရာ ယူသွားတဲ့အတွက် Column တွေ မရှိတော့ပါဘူး။ အားလုံးကို အပေါ်အောက် တန်းစီးပွဲသွားလို ဖုန်းလို Screen သေးတဲ့အခါမျိုးမှာလည်း အဆင်ပြေသွားမှာပဲ ဖြစ်ပါတယ်။

ဒီနည်းစနစ်ကို Responsive Web Design လိုပေါ်ကြတာပါ။ Screen Size ပြောင်းရင် Layout က အလိုအလျောက် Respond လုပ်ပြီး ဖော်ပြပုံ ပြောင်းပေးနိုင်တဲ့အတွက် ဖြစ်ပါတယ်။

## Layout Size

ဒီနေရာမှာ ပြောစရာရှိလာတာက အမျိုးမျိုးအဖို့ဖို့ ကဲ့ပြားနေကြတဲ့ Device တွေရဲ့ Size ပါ။ Laptop ကွန်ပူးတာတွေမှာ ၁၂ လက်မ၊ ၁၃ လက်မ၊ ၁၄ လက်မ၊ ၁၅ လက်မ၊ ၁၆ လက်မ၊ ၁၇ လက်မ စသဖြင့် အရွယ်အစား အမျိုးမျိုး ရှိကြသလို Desktop တွေပါ ပေါင်းလိုက်ရင် ဒီထက်ပိုများပါ၌းမယ်။ Tablet ဆိုရင် လည်း iPad, iPad Mini, iPad Pro စသဖြင့် အမျိုးမျိုးရှိသလို့ Android Tablet တွေပါ ပေါင်းလိုက်ရင် အများကြီးရှိပြီးမှာပါ။ ဖုန်းတွေမှာလည်း အတူတူပါပဲပဲ။ ၄ လက်မ၊ ၅ လက်မ၊ ၆ လက်မ အမျိုးမျိုးရှိကြတာမှ 5.4, 6.2 စသဖြင့် ဒသဗောဓိနဲ့ပြောရတဲ့ Size တွေမှာအများကြီးပါ။ ဒါက Screen Size ပဲ ရှိပါသေးတယ်။ Resolution ကိုလည်း ထည့်တွက်ရပါ၌းမယ်။ တစ်ချို့က Screen သေးပေးမယ့် Resolution မြင့်ကြပါ တယ်။ တစ်ချို့က Screen သာကြီးတာ Resolution နိမ့်ကြပြန်ပါတယ်။ ထောင်ထားတာလား၊ လုံထားတာ လား စသဖြင့် Portrait, Landscape Orientation ကလည်း ကဲ့ပြားဦးမှာပါ။

အဲဒီလောက်ထိ အရွယ်အစား စုံလင်လှတဲ့ Device တွေမှာ ဖုန်းဆိုရင် ဘယ် Size ဖြစ်တယ်၊ Tablet ဆိုရင် ဘယ် Size ဖြစ်တယ် ဆိုပြီး တိတိကျကျ ပြောလို့မရနိုင်ပါဘူး။ ဒီကိစ္စကို Touch Screen Device တွေ ပေါ် ခါစက Web Designer တွေ တော်တော်လေး ခေါင်းစားခဲ့ကြသလို့ အဖြောက်လည်းတွေ့ပြီးကြပြီ ဖြစ်ပါတယ်။ ဒါကြောင့် ပြဿနာကို သတိပြုမိအောင်သာ ပြောပြတာပါ၊ ကိုယ်တိုင်ခေါင်းစား ဖြေရှင်းနေဖို့တော့ မဟုတ်ပါဘူး။ အများလက်ခံ အသုံးပြုတဲ့နည်းတွေ ရှိနေပြီးဖြစ်သလို့ Bootstrap ကလည်း အဲဒီနည်းတွေအတိုင်း ပဲ သွားထားပါတယ်။ Bootstrap မှ Screen Size ကို ဖုန်း၊ Tablet စသဖြင့် Device အမျိုးအစားနဲ့ မပြောပါဘူး။ Small, Medium, Large ဆိုတဲ့အသုံးအနှံးတွေနဲ့ပဲ ပြောပါတယ်။ ဒီပေါ်ကွက်လေးကို လေ့လာကြည့်ပါ။

Breakpoint	Class infix	Dimensions
X-Small	<i>None</i>	< 576px
Small	<i>sm</i>	≥ 576px
Medium	<i>md</i>	≥ 768px
Large	<i>lg</i>	≥ 992px
Extra large	<i>xl</i>	≥ 1200px
Extra extra large	<i>xxl</i>	≥ 1400px

Size အချယ်အစား သတ်မှတ်ချက် (၆) ခုရှိပါတယ်။ 576px ရဲအောက် အချယ်အစားကို X-Small လိုခေါ်ပါတယ်။ ဖုန်းပဲဖြစ်ဖြစ် Tablet ပဲဖြစ်ဖြစ်၊ တစ်ခြား Device တွေပဲဖြစ်ဖြစ်၊ 576px အောက် သေးတဲ့ Screen အားလုံးကို X-Small လို သတ်မှတ်ပြီး အလုပ်လုပ်သွားမှာပါ။ 576px နဲ့ 768px ကြားကိုတော့ Small လိုပဲသတ်မှတ်ပြီး 768px နဲ့ 992px ကြားထဲက Size ကိုတော့ Medium လိုသတ်မှတ်ပါတယ်။ ဒါနည်းအတိုင်း ဆက်ကြည့်သွားရမှာပါ။

အချယ်အစားတစ်ခုချင်းစီအတွက် sm, md, lg စသဖြင့် Size Class တွေလည်း ပေးထားပါတယ်။ Pixel Size တွေ မှတ်ရခေါ်လို Device အမျိုးအစားနဲ့ မှတ်ချင်ရင်လည်း ဒီလိုမျိုး အကြမ်းပျဉ်းမှတ်နိုင်ပါတယ်။ Extra Small အုပ်စုထဲမှာ ဖုန်းတွေ ပါပါတယ်။ Small (sm) အုပ်စုထဲမှာ Landscape Mode နဲ့ သုံးတဲ့ဖုံးတွေ၊ Tablet အသေးတွေ ပါနိုင်ပါတယ်။ Medium (md) အုပ်စုထဲမှာ Tablet တွေနဲ့ သေးတဲ့ Laptop တွေ ပါနိုင်ပါတယ်။ Large (lg) အုပ်စုထဲမှာ Landscape Mode နဲ့သုံးတဲ့ Tablet တွေ၊ iPad Pro လို Screen ကြီးတဲ့ Tablet တွေနဲ့ Laptop အများစု ပါဝင်နိုင်ပါတယ်။ Extra Large (xl) အုပ်စုထဲမှာ Resolution မြင့်တဲ့ Laptop တွေ၊ Desktop တွေ ပါနိုင်ပါတယ်။ Extra Extra Large (xxl) အုပ်စုထဲမှာ တော့ 8k, 4k, HD Screen အကြီးကြီးတွေနဲ့ ကွန်ပျူးတာတွေ၊ Smart TV တွေဘာတွေ ပါနိုင်ပါတယ်။

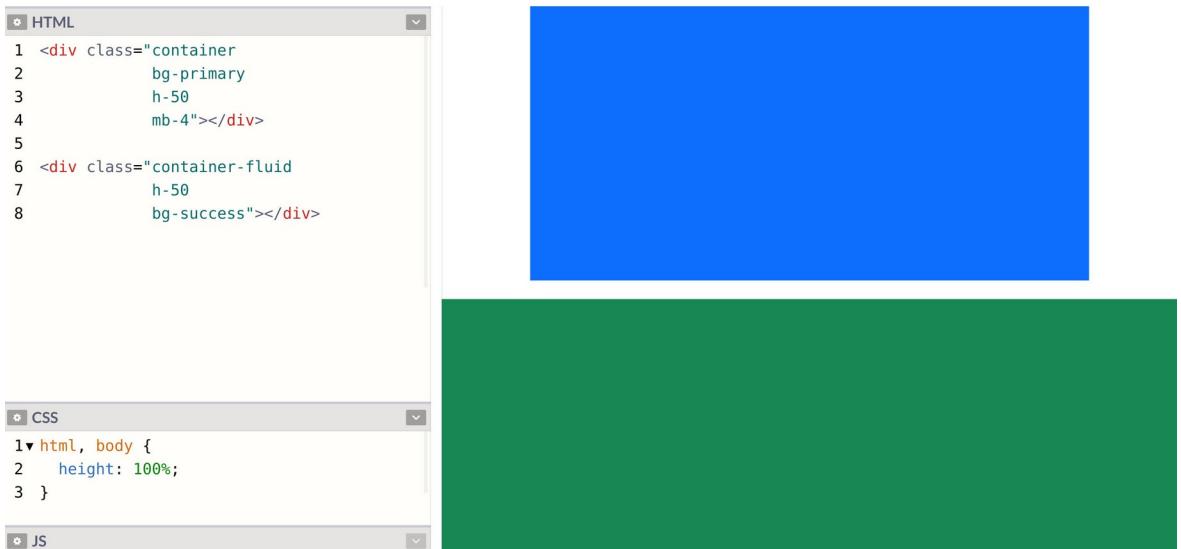
sm, md, lg, xl, xxl စတဲ့ Size Class လေးတွေကို သေချာမှတ်ထားပေးပါ။ Layout မှာသာမက နေရာအတော်များများမှာ Size သတ်မှတ်ဖို့လိုတိုင်း ဒီ Class တွေကို သုံးပါတယ်။

## Layout Container

အခြေခံအားဖြင့် ဝဘ်ဆိုက်တစ်ခုရဲ Content အားလုံးဟာ Container ထဲမှာ ရှိသင့်ပါတယ်။ Container Class တွေ Size ပေါ်မှုတည်ပြီး အမျိုးမျိုးရှိပေမယ့် နှစ်ခုရွေးပြီး မှတ်ထားရင် ရပါပြီ။ container နဲ့ container-fluid ဖြစ်ပါတယ်။

Facebook တို့ Twitter တို့လို ဝဘ်ဆိုက်မျိုးတွေကို မျက်စိထဲမှာ မြှင်ကြည့်ပါ။ Content တွေကို Layout တစ်ခုနဲ့ အလယ်မှာစုပြီး ဖော်ပြထားကြပါတယ်။ ဒီသဘောကို Fixed Width Layout လိုခေါ်ပါတယ်။ Gmail တို့ YouTube တို့လို App တွေကို မျက်စိထဲမှာ မြှင်ကြည့်ပါ။ Screen အကျယ်ရှိသလောက် အပြည့် ယူပြီး Content တွေကို ဖော်ပြထားကြပါတယ်။ ဒီသဘောကို Fluid Layout လိုခေါ်ကြပါတယ်။

Fixed Width Layout တွေ ဖန်တီးလိုရင် container Class ကိုသုံးနိုင်ပါတယ်။ သူက Layout ကို Width သတ်မှတ်ပြီး Screen ရဲ့အလယ်မှာ ပြပေးပါတယ်။ Fluid Layout တွေ ဖန်တီးလိုရင်တော့ container-fluid ကို သုံးနိုင်ပါတယ်။ သူကတော့ Screen အပြည့် နေရာယူပေးပါတယ်။



နမူနာမှာပြထားသလိုရေးပြီး စမ်းကြည့်လိုရပါတယ်။ container Class ပေးထားတဲ့ Element က အလယ်မှာ နေရာယူဖော်ပြပြီးတော့ container-fluid Class ပေးထားတဲ့ Element ကတော့ အပြည့်နေရာယူ ဖော်ပြတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ CSS ထဲမှာ html နဲ့ body အတွက် height: 100% ပေးထားတာကိုလဲ သတိပြုပါ။ အဲဒီလိုပေးထားမှ Container တွေမှာသတ်မှတ်ထားတဲ့ h-50 (height: 50%) က အလုပ်လုပ်မှာ မို့လိုပါ။

## Grid System

Bootstrap က Layout တွေဖန်တီးဖိုအတွက် 12 Columns Grid ခေါ် အပိုင်း (၁၂) ပိုင်းကို အခြေခံတဲ့ စနစ်ကို သုံးပါတယ်။ ဒီလိုပုံစံပါ။

1	1	1	1	1	1	1	1	1	1	1	1	1
12												
2	4				6							

Column တစ်ခုက Row တစ်ခုလုံးအပြည့် နေရာယဉ်ချင်ရင် (၁၂) ပိုင်းလုံးကို ယူလိုက်လိုပါတယ်။ တစ်ဝက်ပဲလိုချင်ရင် (၆) ပိုင်းယဉ်လိုပါတယ်။ ဒီသဘောနဲ့ (၁၂) ပိုင်းရှိတဲ့ထဲက ကိုယ်လိုသလောက် ပိုင်းယဉ်လိုရတဲ့စနစ်မျိုးပါ။

ဘာကြောင့် (၁၂) ပိုင်းကိုသုံးသလဲဆိုတော့၊ စဉ်းစားကြည့်ပါ။ (၁၂) ကို ၂, ၃, ၄, ၆ အားလုံးနဲ့ စားလိုပြေတဲ့ အတွက် (၂) ပိုင်း (၃) ပိုင်း (၄) ပိုင်း စသည်ဖြင့် အသီခွဲယဉ်လိုနိုင်ပါတယ်။ ဒီလိုပါ -

12					
6		6			
4		4		4	
3		3		3	

ဒါကြောင့် Layout တွေကို စီစီညိညိနဲ့ သပ်သပ်ရပ်ရပ် ရရှိမှာဖြစ်ပါတယ်။ ဒါမျိုးတွေက အကြောင်းမဲ့ ဖြစ်ပေါ်လာတာမျိုး မဟုတ်ဘဲ လက်တွေ့ အတွေ့အကြုံတွေပေါ်မှာ အခြေခံဖြစ်ပေါ်လာတဲ့ ကိစ္စမျိုးတွေပါ။

Bootstrap ရဲ့ Grid System ကို အသုံးပြုဖို့အတွက် row နဲ့ col ဆိုတဲ့ Class တွေကို သုံးနိုင်ပါတယ်။ အထက်မှာ ဖော်ပြထားတဲ့ Grid ပုံစံရှိဖို့ရင် Bootstrap နဲ့ ဒီလိုရေးပေးရမှာပါ။

### HTML

```
<div class="row">
    <div class="col-12"></div>
</div>
<div class="row">
    <div class="col-6"></div>
    <div class="col-6"></div>
</div>
<div class="row">
    <div class="col-4"></div>
    <div class="col-4"></div>
    <div class="col-4"></div>
</div>
```

သိသင့်တဲ့ သဘောသဘာဝတွေကိုသာ သိထားမယ်ဆိုရင် ရေးနည်းက မခက်ပါဘူး။ row ထဲမှာ col တွေ

ရှိပြီး ပိုင်းပြီးတော့လိုချင်တဲ့ အရေအတွက်ကို ၂၀၁ ရဲနောက်မှာ တွဲထည့်ပေးရတာပါ။ အရေအတွက် ထည့်မပေးရင်လည်း ရပါတယ်၊ တစ်ခုရှိရင်တစ်ခု၊ နှစ်ခုရှိရင်နှစ်ခု၊ ရှိသလောက် ရွယ်တူ အညီယူပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် အပေါ်ကကုဒ်ကို ဒီလိုရေးရင်လည်း ရလဒ်အတူတူပါဘူး။

### HTML

```
<div class="row">
    <div class="col"></div>
</div>
<div class="row">
    <div class="col"></div>
    <div class="col"></div>
</div>
<div class="row">
    <div class="col"></div>
    <div class="col"></div>
    <div class="col"></div>
</div>
```

အပေါ်ဆုံး row မှာ col တစ်ခုထဲရှိလိုက်တစ်ခုထဲ အပြည့်ယူလိုက်မှာပါ။ ဒုတိယ row မှာ နှစ်ခုရှိလိုက်နှစ်ခု အညီတစ်ဝက်စီယူပေးလိုက်မှာပါ။ လက်တွေ့ရေးသား စမ်းသပ်လိုရင်လည်း ဒီလိုလေးစမ်းကြည့်ပါ။

```
• HTML
1▼ <div class="container">
2▼   <div class="row content">
3    <div class="col"></div>
4    <div class="col"></div>
5    <div class="col"></div>
6  </div>
7 </div>
8
```

```
• CSS
1▼ .content div {
2  height: 100px;
3  border: 2px solid brown;
4 }
```

```
• JS
```



CSS ထဲမှာ height တွေ border တွေကို မြင်သာအောင် ထည့်ပေးထားတာကို သတိပြုပါ။ Layout ရဲ့ နေရာယဉ်ပုံကတော့ ကိုယ်ရေးပေးစရာ မလိုတော့ပါဘူး။ row တွေ col တွေ ပေးလိုက်ယုံနဲ့ လိုချင်တဲ့ ရလဒ်ကို ရရှိမှာ ဖြစ်ပါတယ်။ Column တွေကို အညီမယူဘဲ ကိုယ်လိုသလောက် ယူပြီးစမ်းကြည့်ချင်ရင် ဒီလိုလေး စမ်းလိုက်ပါ။

```

HTML
1▼ <div class="container">
2▼   <div class="row content">
3    <div class="col-3"></div>
4    <div class="col-6"></div>
5    <div class="col-3"></div>
6  </div>
7 </div>
8

CSS
1▼ .content div {
2  height: 100px;
3  border: 2px solid brown;
4 }

JS

```

ဒီတစ်ခါတော့ col-3, col-6 စသဖြင့် ကိုယ်လိုသလောက် ပိုင်းယူလိုက်လို အလယ်က Column ကို ပို့ပြီးနေရာယူပြီး ဘေးတစ်ဘက်တစ်ချက်က Column တွေကို ပို့သေးသေး နေရာယူပေးတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

## Responsive Layouts

Responsive Layout တွေ ရရှိဖို့အတွက်တော့ အပေါ်နားမှာ ပြောခဲ့တဲ့ Size Class တွေကို တွဲသုံးပေးရပါတယ်။ ဥပမာ - ဒီကုဒ်လေးကို လေ့လာကြည့်ပါ။

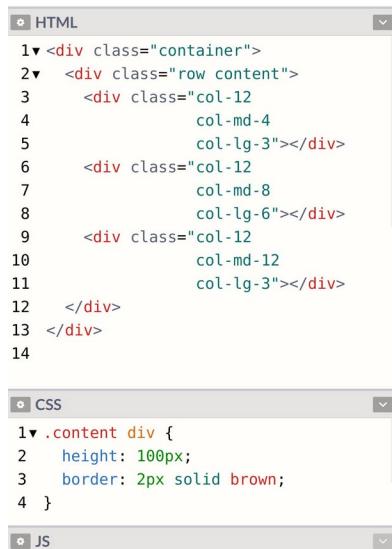
**HTML**

```

<div class="row">
  <div class="col-12 col-md-4 col-lg-3"></div>
  <div class="col-12 col-md-8 col-lg-6"></div>
  <div class="col-12 col-md-12 col-lg-3"></div>
</div>

```

ဒီကုဒ်ရဲအမိပါယ်က၊ Column (၃) ခုလုံးအတွက် col-1-12 လို့ ပေးထားတဲ့အတွက် အပြည့်နေရာယူမှာဖြစ်ပါတယ်။ ဒါကြောင့် Default အနေနဲ့ အပေါ်အောက်ဆင့်ပြီးတော့ပဲ ဖော်ပြမှာပါ။ အကယ်၍ Medium Size (Tablet) ဖြစ်ခဲ့မယ်ဆိုရင်တော့ col-md-{x} ကိုသုံးပြီး 4-8-12 လို့ Column Layout ကို သတ်မှတ်ပေးထားပါတယ်။ ဒါကြောင့် Column နှစ်ခုပါတဲ့ 2 Columns Layout အဖြစ်ကို ပြောင်းသွားမှာ ဖြစ်ပါတယ်။ Large Size (ကွန်ပျူးတာ) ဖြစ်ခဲ့မယ်ဆိုရင်တော့ col-lg-{x} ကိုသုံးပြီး 3-6-3 လို့ သတ်မှတ်ထားတဲ့အတွက် Column သုံးခုပါတဲ့ 3 Columns Layout ဖြစ်သွားမှာပဲ ဖြစ်ပါတယ်။ လက်တွေ့ ရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။



```

HTML
1▼ <div class="container">
2▼   <div class="row content">
3    <div class="col-12
4      col-md-4
5      col-lg-3"></div>
6    <div class="col-12
7      col-md-8
8      col-lg-6"></div>
9    <div class="col-12
10     col-md-12
11     col-lg-3"></div>
12  </div>
13 </div>
14

CSS
1▼ .content div {
2  height: 100px;
3  border: 2px solid brown;
4 }

JS

```



အခုန်မူနာရလဒ်မှာ ဖော်ပြစ်ရာ နေရာကျယ်တဲ့အတွက် ၁၅ Size သက်ဝင်နေလို့ 3 Columns Layout တစ်ခုကို ရရှိနေခြင်း ဖြစ်ပါတယ်။ 3-6-3 ပုံစံနေရာယူထားပါတယ်။ ဒီကုဒ်ကိုပဲ နည်းနည်းနေရာချုပြီး စမ်းကြည့်လိုက်ရင်တော့ အခုလိုပုံစံဖြစ်သွားမှာပါ။

HTML

```

1▼ <div class="container">
2▼   <div class="row content">
3    <div class="col-12
4      col-md-4
5      col-lg-3"></div>
6    <div class="col-12
7      col-md-8
8      col-lg-6"></div>
9    <div class="col-12
10     col-md-12
11     col-lg-3"></div>
12  </div>
13 </div>
14

```

CSS

```

1▼ .content div {
2  height: 100px;
3  border: 2px solid brown;
4 }

```

JS



ဒီရလဒ်မှာတော့ နေရာ ကျဉ်းသွားပြီဖြစ်လို့ md Size သက်ဝင်ပြီး 2 Columns Layouts တစ်ခုအနေနဲ့ အလုပ်လုပ်နေတာကို တွေ့မြင်ရခြင်းပဲဖြစ်ပါတယ်။ ဒီထက်ထပ်ချုံလိုက်ရင်တော့ အခုလိုတွေ့ရမှာပါ။

HTML

```

1▼ <div class="container">
2▼   <div class="row content">
3    <div class="col-12
4      col-md-4
5      col-lg-3"></div>
6    <div class="col-12
7      col-md-8
8      col-lg-6"></div>
9    <div class="col-12
10     col-md-12
11     col-lg-3"></div>
12  </div>
13 </div>
14

```

CSS

```

1▼ .content div {
2  height: 100px;
3  border: 2px solid brown;
4 }

```

JS



ဒီရလဒ်မှာ နေရာတော်တော်လေး ကျဉ်းသွားပြီမို့လို့ md တွေ 1g တွေ အလုပ်မလုပ်တော့ဘဲ Default အတိုင်း အပြည့်တွေ နေရာယူထားတဲ့အတွက် Single Column Layout တစ်ခုကို ရှုံးခြင်းပဲဖြစ်ပါတယ်။

ဒီနည်းနဲ့ Bootstrap ကိုအသုံးပြုပြီး Responsive Layouts တွေဖန်တီးနိုင်မှာ ဖြစ်ပါတယ်။ အတွေ့ရများ တဲ့ ဝဘ်ဆိုက် Layout လေးတစ်ခုကို နမူနာအနေနဲ့ ထပ်ပေးချင်ပါတယ်။ ရေးရမယ့်ကုဒ်က ဒီလိုပါ -

### HTML

```
<main class="bg-secondary py-5">

    <div class="container bg-light">
        <div class="bg-light" style="height: 400px"></div>
    </div>

</main>

<section class="container py-5">
    <div class="row g-5">

        <div class="col-12 col-md-6 col-lg-3">
            <div class="bg-secondary" style="height: 200px"></div>
        </div>

        <div class="col-12 col-md-6 col-lg-3">
            <div class="bg-secondary" style="height: 200px"></div>
        </div>

        <div class="col-12 col-md-6 col-lg-3">
            <div class="bg-secondary" style="height: 200px"></div>
        </div>

        <div class="col-12 col-md-6 col-lg-3">
            <div class="bg-secondary" style="height: 200px"></div>
        </div>

    </div>
</section>

<footer class="container">

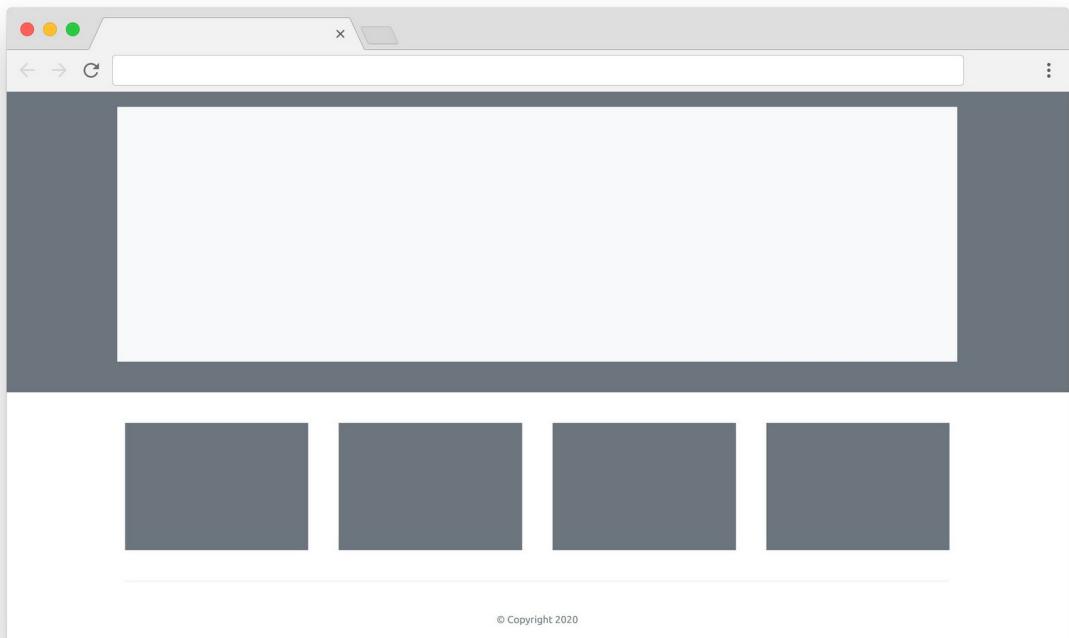
    <div class="border-top border-top-2 py-5 text-center text-muted">
        © Copyright 2020
    </div>

</footer>
```

တစ်ချို့သတိပြုသင့်တဲ့ လုပ်ဆောင်ချက်တွေအကြောင်း ပြောပြပါမယ်။ py-5 ဆိုတဲ့ Class ၂ Padding Top/Bottom အတွက်ပါ။ Padding Left/Right အတွက်လိုချင်ရင်တော့ px- ကိုသုံးနိုင်ပါတယ်။ g-5 Class မှာပါတဲ့ ၅ ရဲ့ အဓိပ္ပာယ်က Gutter ဖြစ်ပါတယ်။ Column တစ်ခုနဲ့တစ်ခုကြား အကွာအဝေးပါ။ ၅ က အမြင့်ဆုံးဖြစ်ပြီး 1, 2, 3, 4 တန်ဖိုးတွေ ပြောင်းစမ်းကြည့်နိုင်ပါတယ်။ border Class တွေကတော့

အထူးပြောစရာ မလိုပါဘူး။ Class အမည်မှာ အဓိပ္ပာယ်ပေါ်နေပါဖြီ။ text-muted ကတေသာ စာတွေကို နည်းနည်းမိန့်ပြီး ပြစ်ဖြစ်ပါတယ်။ ဒီလိုအသုံးဝင်တဲ့ Utility Class တွေအကြောင်းကို နောက်တစ်ခန်းမှာ သီးခြားထပ်လေ့လာကြပါ့။

ကျွန်ုတဲ့ လုပ်ဆောင်ချက်တွေအကြောင်းကိုတော့ ရေးထားတဲ့ကုဒ်ကို သေချာဖတ်ပြီးတော့ပဲ လေ့လာကြည့်လိုက်ပါ။ စမ်းကြည့်လိုက်ရင် ရမယ့်ရလဒ်ကတော့ အခုလိုဖြစ်မှာပါ။



ဒါဟာ အတွေ့ရများတဲ့ ဝဘ်ဆိုက် Layout ပုံစံတစ်ခုပါပဲ။ Responsive Layout Class တွေလည်း တစ်ခါတည်း ထည့်ရေးပြီးသားမို့လို့ Screen ကို ချုံချုပ်ပြီး အမျိုးမျိုးစမ်းကြည့်နိုင်ပါတယ်။ Layout က သင့်တော်အောင် အလိုအလျောက် ပြောင်းပြီး ပြပေးတယ်ဆိုတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

ဒီ Layout ထဲမှာ Carousel တွေ Navbar တွေ Card တွေ သူနေရာနဲ့သူ အစားထိုး ထည့်ပေးလိုက်ရင် လက်တွေ့ အသုံးချုံ ဝဘ်ဆိုက် Template တစ်ခု ဖြစ်သွားနိုင်ပါတယ်။ စမ်းသပ် ထည့်သွင်း ကြည့်ဖို့ တိုက်တွန်းပါတယ်။

## အခန်း (၇) – Utility Classes

Bootstrap ကပေးထားတဲ့ Components တွေ Layouts လုပ်ဆောင်ချက်တွေဟာ အတော်လုံးပြည့်စုံပေမယ့် လက်တွေ့မှာ ကိုယ့်လိုအပ်ချက်နဲ့ ကိုက်ညီအောင် ဖြည့်စွက်ရတာတွေ၊ ပြင်ဆင်ရတာတွေ ရှိပါတယ်။ အဲဒီလို ဖြည့်စွက်ပြင်ဆင်ရတာတွေ လုပ်တဲ့အခါ CSS တွေအမြတ်စီး ရေးစရာမလိုပါဘူး။ လိုအပ်လေ့ရှိတဲ့ လုပ်ဆောင်ချက်တွေအတွက် Bootstrap က ကြိုးရေးထားတဲ့ Utility Classes တွေ ရှိကြပါတယ်။ အဲဒီ Class တွေအကြောင်းကို စုစုပေါင်းမြို့တော့ ဖော်ပြချင်ပါတယ်။

### Borders

Border Utility Classes တစ်ချို့ကို ရှေ့ပိုင်းနမူနာတွေမှာလည်း သုံးဖြစ်ခဲ့ပါတယ်။ Element တွေ Component တွေမှာ Border ထည့်သွင်းဖို့လိုရင် Bootstrap ရဲ့ border Class ကိုသုံးနိုင်ပါတယ်။ border-top, border-left စသဖြင့် တစ်ဘက်ချင်းစီလည်း ထည့်သွင်းလိုရပါတယ်။ Card တို့ List Group တို့လို Border ရှိပြီးသား Component တွေအတွက် Border ရဲ့ အရောင်ကိုပြောင်းချင်ရင် တော့ border-{color} Classes တွေကို သုံးနိုင်ပါတယ်။

#### HTML

```
<p class="border border-primary">
    Some Content
</p>
```

ပေးထားတဲ့နမူနာက <p> Element မှာ Border ထည့်လိုက်ပြီး အဲဒီ Border ရဲ့အရောင်ကို primary လို သတ်မှတ်ပေးလိုက်တာပါ။ ဒီနည်းနဲ့ မည်သည့် Element မှာမဆို Border တွေ ထည့်သွင်းနိုင်ပါတယ်။ Border ရဲ့ Size တွေ Radius တွေလည်း သတ်မှတ်လိုရပါသေးတယ်။ ဒီလိုပါ -

**HTML**

```
<p class="border
        border-2
        border-primary
        rounded
        p-2">Some Content</p>
```

border-2 နဲ့ Size ကို သတ်မှတ်ပေးထားပါတယ်။ 2 အစား 5 ထိပေးလို့ရပါတယ်။ rounded နဲ့ Border Radius ထည့်ထားပါတယ်။ rounded-circle နဲ့ rounded-pill လည်းရှိပါသေးတယ်။ ဘာကွာလဲ သိရဖို့အတွက် ကိုယ်တိုင်သာရေးထည့်ပြီး စမ်းကြည့်လိုက်ပါ။ Border ပါနေပြီး မလိုချင်လို့ ပြန်ဖြတ်ချင်ရင်လည်း border-0 နဲ့ ပြန်ဖြတ်နိုင်ပါတယ်။ border-top-0, border-left-0 စသည်ဖြင့်လည်း ရှိပါသေးတယ်။ တစ်ခုချင်းလိုက်မှတ်နေရင် မှတ်စရာတွေ များပါတယ်။ လက်တွေ့စမ်းကြည့်လိုက်လို့ သဘောသဘာဝ သိသွားရင် ပိုမှတ်လို့ကောင်းပါတယ်။

**Color**

Color Classes တွေကိုလည်း ရှုံးပိုင်းနမူနာတွေမှာ သုံးခဲ့ကြပြီးဖြစ်ပါတယ်။ အထူးသဖြင့် background နဲ့ text အတွက် အရောင်တွေသတ်မှတ်လိုရင် သုံးရတာပါ။ text-primary, text-success, bg-info, bg-warning စသဖြင့် လိုအပ်တဲ့ နေရာတိုင်းမှာ သတ်မှတ်နိုင်ပါတယ်။ ကျွန်ုတ္တာ အသုံးဝင်တာလေးတစ်ချို့ ထည့်ပြောချင်ပါတယ်။

- text-muted
- text-black-50
- text-white-50
- bg-white
- bg-transparent

text-muted ကိုတော့ ပြီးခဲ့တဲ့ နမူနာတစ်ခုမှာ ထည့်သုံးပေးခဲ့ပါတယ်။ စာကို နည်းနည်း မိန့်ပြီးပြမာပါ။ text-black-50 နဲ့ text-white-50 ကလည်း အလားတူပဲ၊ နည်းနည်းစီ အရောင်မိန့်ထားပေးတဲ့ စာတွေကို လိုချင်တဲ့အခါ သုံးနိုင်ပါတယ်။ bg-transparent ကိုတော့ မူလက Background အရောင်ပါနေတဲ့ Component တစ်ခုမှာ Background အရောင် ပြန်ဖြတ်ချင်တဲ့အခါ သုံးနိုင်ပါတယ်။

ထူးခြားချက်အနေး ဗိုလ်-gradient ဆိုတာလည်း ရှိပါသေးတယ်။ ဒီလိုလေးစမ်းကြည့်လိုက်ပါ –

#### HTML

```
<p class="bg-danger
        text-white
        bg-gradient
        p-2">Some Content</p>
```

Background အရောင်ကို ပုံသေတစ်ရောင်ထဲ မဟုတ်ဘဲ Gradient ပုံစံ ရောင်ပြီးလေးနဲ့ ပြပေးမှာပါ။

#### Display

Element တွေရဲ့ Display Type ကို d- နဲ့စတဲ့ Class နဲ့ပြောင်းနိုင်ပါတယ်။ d-block, d-inline, d-none စသည်ဖြင့်ပါ။ ဒီနေရာမှာ ပိုအရေးကြီးတာက Responsive Size တွေဖြစ်ပါတယ်။ အလယ်မှာ Responsive Size Class တွေထည့်ပြီး Screen Size အလိုက် Display Type ကိုပြောင်းနိုင်ပါတယ်။ ဒီလိုပါ

#### HTML

```
<p class="d-none d-md-block">Some Content</p>
```

d-none လိုပြောထားတဲ့အတွက် Default မှာ ပျောက်နေပါလိမ့်မယ်။ ဒါပေမယ့် d-md-block လိုပြောထားတဲ့အတွက် Medium Device တွေမှာ ပေါ်လာမှာဖြစ်ပါတယ်။ ဒါကြောင့် Screen သေးရင် ပျောက်သွားပြီး Screen ကြီးမှ ပေါ်လာတဲ့ လုပ်ဆောင်ချက်ကို ရရှိမှာ ဖြစ်ပါတယ်။ md အစား sm, lg, xl, xxl စသဖြင့် တစ်ခြား Size Class တွေကို လိုအပ်သလို အသုံးပြုနိုင်ပါတယ်။

တော်တော် အသုံးဝင်တဲ့ လုပ်ဆောင်ချက်ပါ။ Responsive Web Design တွေလုပ်တဲ့အခါ Screen Size ပေါ်မှုတည်ပြီး Element တွေကို ပြသင့်အချိန်မှပြတယ်၊ မပြသင့်ရင် ဖျောက်ထားတယ်ဆိုတာဟာ လိုအပ်တဲ့ လုပ်ဆောင်ချက်တစ်ခု ဖြစ်ပါတယ်။

## Flexbox

CSS Flexbox အကြောင်းကို လိုရင်းလေးတွေ ရွေးထုတ်ပြောနေလိုသာပါ၊ တစ်ကယ်တော့ အတော်လေးကျယ်ပြန်တဲ့ အကြောင်းအရာ တစ်ခု ဖြစ်ပါတယ်။ အခုလည်း Bootstrap က ပေးထားတဲ့ အရေးကြီးတဲ့ လုပ်ဆောင်ချက်လေးတစ်ချို့ကို ရွေးထုတ်ပေးချင်ပါတယ်။

Element တစ်ခုကို Flexbox ဖြစ်စေချင်ရင် `d-flex` Class ကိုသုံးနိုင်ပါတယ်။ အဲဒီ Flexbox ထဲက Element တွေကို စီမံဖို့အတွက် Flex Utility Class ပေါင်း (၆၀) ကျော်ထိ ရှိနေပါတယ်။ အကုန်သာမှတ်ရင် မလွယ်ပါဘူး။ (၃) ခုပဲ ရွေးမှတ်စေချင်ပါတယ်။

- `flex-row`
- `flex-column`
- `flex-fill`

`flex-row` က Element တွေကို ဘေးတိုက်စီပြီး ညီအောင်ပြပေးတဲ့ လုပ်ဆောင်ချက်ပါ။ Default Value ဖြစ်ပါတယ်။ ပုံမှန်အားဖြင့် ကိုယ့်ဘာသာ ပေးစရာမလိုပါဘူး။ `flex-column` ကတော့ အပေါ်အောက် စီပြပေးမှာပါ။ အဲဒီနှစ်ခုကို Responsive Size Class တွေနဲ့ တွဲသုံးရင် အသုံးဝင်ပါတယ်။ ဒီလိုပါ –

### HTML

```
<div class="d-flex flex-column flex-md-row">
  <div class="bg-primary p-5"></div>
  <div class="bg-danger p-5 flex-fill"></div>
  <div class="bg-success p-5"></div>
</div>
```

နမူနာအရ `flex-column` လိုပေးထားတဲ့အတွက် Default အနေနဲ့ အပေါ်အောက်စီပြီး ပြမှာပါ။ ပြီးတော့မှ `flex-md-row` လိုပြောထားတဲ့အတွက် Medium Device Size ဖြစ်လာတဲ့အခါ ဘေးတိုက်စီပြီးတော့ ပြမှာပါ။ ဒါကြောင့် Screen Size ပြောင်းရင် Layout လိုက်ပြောင်းတဲ့ လုပ်ဆောင်ချက်ကို ရသွားပါတယ်။ ထုံးစံအတိုင်း `md` အစား တစ်ခြား Size Class တွေကို လိုအပ်သလို အစားထိုးပြီး သုံးနိုင်ပါတယ်။ `flex-fill` ကတော့ "ဘေးတိုက်" နေရာလွှတ်ကျန်သလောက် အကုန်အပြည့် နေရာယူစေချင်တဲ့ Element တွေမှာ သတ်မှတ်ပေးနိုင်ပါတယ်။ ဒါကြောင့် ကျန်တဲ့ Element တွေကို ရှိသလောက်ပဲပြုပြီး `flex-fill` ပါတဲ့ Element ကို အပြည့်နေရာယူ ပြပေးမှာပါ။

## Float

Float CSS Property ဟာ တော်တော်လေးအရေးကြီးတဲ့ Property ဖြစ်ခဲ့ပါတယ်။ အရင်က Flexbox လုပ်ဆောင်ချက် CSS မှာမရှိလို့ Float Property တွေကိုသုံးပြီး Layout တွေကို ဖန်တီးခဲ့ကြရပါတယ်။ အခုတော့ Layout အတွက် Float ကို အားကိုးဖို့ မလိုအပ်တော့ပါဘူး။ ဒါပေမယ့် Float လုပ်ဆောင်ချက် အသုံးဝင်တဲ့နေရာတွေ ရှိပါသေးတယ်။ တစ်ချို့ Element တွေ ဘယ်ဘက်ကပ်ပြီး ပြစ်ချင်ရင် float-start ကိုသုံးနိုင်ပါတယ်။ ညာဘက်ကပ်ပြီး ပြစ်ချင်ရင် float-end ကို သုံးနိုင်ပါတယ်။ သူ့ကိုလည်း Size Class နဲ့ တွဲသုံးနိုင်တဲ့အတွက် အသုံးဝင်တာပါ။ ဒီလိုပါ –

### HTML

```
<div class="bg-warning p-4 clearfix">
  <h1 class="float-md-start">Some Title</h1>
  <h2 class="float-md-end">Some sub-title</h2>
</div>
```

နမူနာအရ ပုံမှန်ဆိုရင် <h1> နဲ့ <h2> ကို အပေါ်အောက်ဆင့်ပြီး ပြမာဖြစ်ပေမယ့် Medium Size ကို ရောက်လာတဲ့အခါ <h1> ကို ဘယ်ဘက်ကပ်ပြမာဖြစ်ပါတယ်။ <h2> ကိုတော့ ညာဘက်ကပ်ပြမာပါ။ ဒီလို Float ကိုသုံးပြီး ဘယ်ညာ ကပ်တဲ့အခါ ပင်မ Element မှာ clearfix လိုပေါ်တဲ့ လုပ်ဆောင်ချက် ပါဖို့လိုတာကို သတိပြုပါ။ ဒါကိုပြည့်စုံအောင် ရှင်းရရင် တော်တော်ရှည်ပါလိမ့်မယ်။ ဒါကြောင့် တို့တို့ လိုရင်းလေးပဲ ပြောချင်ပါတယ်။ float Class တွေနဲ့ Element တွေကို ဘယ်ညာကပ်လို့ရတယ်။ ဒီလိုကပ်လိုက်လို့ ပင်မ Element ရဲ့ ဖော်ပြပုံမှန်တော့ရင် clearfix ထည့်ပေးရတယ်လို့သာ မှတ်ထားပါ။

## Width & Height

Width တွေ Height တွေနဲ့ပက်သက်တဲ့ Class တွေကတော့ Percentage ကိုပဲအခြေခံပြီး အလုပ်လုပ်လို သိပ်မပြည့်စုံဘူး။ ကိုယ့်ဘာသာ နည်းနည်းတော့ ထပ်ရေးပေးရတယ်။ ဒီနမူနာကို ကြည့်ပါ –

### HTML

```
<div style="height: 300px" class="bg-dark p-2">
  <div class="h-50 w-50 bg-light"></div>
</div>
```

h-50 ဆိုတာ height: 50% ကိုပြောတာပါ။ အလုပ်လုပ်ပါတယ်။ w-50 ဆိုတာ width: 50% ကိုပြောတာပါ။ အလုပ်လုပ်ပါတယ်။ ဒါပေမယ့် အဲဒီလို အလုပ်လုပ်ဖို့အတွက် ပင်မ Element မှာ height: 300px ကို ကိုယ့်ဘာသာပေးထားရပါတယ်။ အဲဒီ height ကို အခြေခံပြီး အထဲက Element တွေက အလုပ်လုပ်တာမို့လို့ မပါရင် အဆင်မပြေပါဘူး။

ပြီးတဲ့အခါ 25%, 50%, 75%, 100% ဆိုပြီး လေးမျိုးပဲ ရှိပါတယ်။ ဒါကြောင့် h-25, h-50, h-75 နဲ့ h-100 တိုကိုပဲ သုံးလိုရမှာပါ။ တစ်ခြားတန်ဖိုးတွေ မရှိပါဘူး။ w- လည်းအတူတူပါပဲ။ h-auto နဲ့ w-auto တော့ ရှိပါတယ်။ အဲဒါက ပေးထားတဲ့ Width တွေHeight တွေကို လိုအပ်လို့ ပြန်ဖြုတ်ချင်တဲ့ အခါမျိုးမှာ အသုံးဝင်နိုင်ပါတယ်။

## Margin & Padding

Margin တွေ Padding တွေနဲ့ပက်သက်တဲ့ Utility Classes တွေကိုတော့ ရွှေ့နှမူနာတွေမှာလည်း တွေ့ခဲ့ကြပြီးသားပါ။ 1, 2, 3, 4, 5 ဆိုပြီး Size က (၅) မျိုးရှိပါတယ်။ Margin အတွက် m- နဲ့စပြီး Padding အတွက် p- နဲ့စပါတယ်။ Top, Right, Bottom, Left တစ်ဘက်စီလည်းပေးလို့ရပါတယ်။

- mt-{size} (Margin Top)
- me-{size} (Margin End or Right)
- mb-{size} (Margin Bottom)
- ms-{size} (Margin Start or Left)
- my-{size} (Margin Top/Bottom)
- mx-{size} (Margin Left/Right)

Padding အတွက်လည်း အတူတူပါပဲ။ m- အစား p- နဲ့စတာပဲ ကွာသွားမှာပါ။ Size အတွက် auto လည်းရှိပါသေးတယ်။ margin: auto လုပ်ဆောင်ချက်မျိုးကို လိုချင်ရင် သုံးနိုင်ပါတယ်။ 0 လည်းရှိပါသေးတယ် m-0 p-0 ဆိုရင် Margin တွေ Padding တွေ အကုန်ဖြုတ်ပေးလိုက်မှာပါ။

## Text

Text နဲ့ ပက်သက်တဲ့ Class တွေကတော့ Alignment တို့ Formatting တို့အတွက် အစုရိပါတယ်။ Left, Right, Center, Justify စတဲ့ Alignment လုပ်ငန်းတွေအတွက် text-start, text-end, text-center, text-justify စသဖြင့် ကုန်သုံးလိုပါတယ်။ ဥပမာ -

### HTML

```
<h1 class="text-center">Centered Title</h1>
```

နမူနာအစာ ခေါင်းစီးအတွက်စာကိုအလယ်မှာ Align လုပ်ပြီး ပြပေးမှာပါ။ Responsive Class တွေနဲ့ တဲ့ပြီး သုံးနိုင်တဲ့အတွက် ပိုပြီးတော့ အသုံးဝင်နိုင်ပါသေးတယ်။ ဒီလိုပါ -

### HTML

```
<h1 class="text-center text-md-start">Centered Title</h1>
```

နမူနာအရ Default အနေနဲ့ Center Align ထားပြီးပြပေမယ့် Medium Screen Size ဖြစ်သွားပြီးဆိုရင် Left Align နဲ့ပြောင်းပြီး ပြပေးသွားမှာဖြစ်ပါတယ်။ Bold, Italic, Underline, Strike-through စတဲ့ Formatting လုပ်ငန်းတွေအတွက် ဒီလို Class တွေရှိပါတယ်။

- fw-bold
- fw-bolder
- fw-normal
- fw-light
- fw-lighter
- fst-italic
- fst-normal
- text-decoration-underline
- text-decoration-line-through
- text-decoration-none

ဒါတွေကိုတော့ တစ်ခုချင်းရှင်းပြဖို့ မလိုအပ်ဘူးလိုထင်ပါတယ်။ Class အမည်မှာ သူအဓိပ္ပာယ်နဲ့သူ ပေါ်လွင်ပြီးဖြစ်နေလိုပါ။

စာကြောင်းတွေရဲအပေါ်အောက် အစိတ်အကြ Line Height နဲ့ပက်သက်ပြီး (၃) မျိုးမှတ်သင့်ပါတယ်။

- lh-sm
- lh-base
- lh-lg

lh-base က မူလပမာဏအတိုင်းဖြစ်ပြီး lh-sm ဆိုရင် Line Height ကျဉ်းသွားလို စာကြောင်းတွေတစ်ခုနဲ့တစ်ခု ပိုပြီး ကဲသွားမှာပဲဖြစ်ပါတယ်။ လိုရမယ်ရထည့်ပေးထားတာပါ။ Bootstrap ကပေးထားတဲ့ Line Height က အများအားဖြင့် အဆင်ပြေပါတယ်။ မြန်မာစာလို စာမျိုးတွေတော့ သုံးထားတဲ့ ဖွန့်ပေါ်မှတည်ပြီး ရုပ်နှုန်းကို Line Height လေးချွေပေးထားမှ ဖတ်ရတာအဆင်ပြေတာမျိုး ဖြစ်တတ်ပါတယ်။ အဲဒီလို လိုအပ်လာရင် lh-lg Class ကို သုံးနိုင်ပါတယ်။

## Position

Absolute, Relative, Fixed စတဲ့ Position နဲ့ပက်သက်တဲ့ Class တွေလည်းရှိပါတယ်။ position-absolute, position-fixed, position-relative ဆိုတဲ့ (၃) မျိုးကို မှတ်ထားသင့်ပါတယ်။ Position တွေရဲ သဘောသဘာဝကို CSS အခန်းမှာ ပြောခဲ့ပြီးသားပါ။ Position ပေးထားပြီးနောက် Element ရဲ ဖော်ပြုပုံတည်နေရာ သတ်မှတ်ဖို့အတွက် left, right, bottom, top စတဲ့ Property တွေနဲ့ CSS မှာ တွဲသုံးရသလိုပဲ Bootstrap မှာ တွဲသုံးပေးရမှာပါ။ Bootstrap မှာတော့ start, end, bottom, top ဖြစ်သွားပါတယ်။ ဥပမာ -

### HTML

```
<div class="position-relative bg-dark" style="height: 200px">
    <div class="position-absolute
        bg-light top-50 start-50
        w-25 h-25"></div>
</div>
```

နှမူနာအရ ပင်မ Element မှာ position-relative လိုသတ်မှတ်ပေးထားပြီး အတွင်းထဲက Element မှာ position-absolute လိုသတ်မှတ်ထားပါတယ်။ top-50 ဆိုတာ top: 50% ဆိုတဲ့ သဘောမျိုးပါ။ start-50 ကတော့ left: 50% ဆိုတဲ့သဘောမျိုးပါ။ ဒါကြောင့် အတွင်းထဲက

Element က အလယ်မှာ ရောက်နေရမှာပါ။ အလယ်တည့်တည့်တော့ ရောက်မှာ မဟုတ်ပါဘူး။ အလယ်တည့်တည့်ရောက်ချင်ရင် start က 50% ဖြစ်လိုမရပါဘူး။ start က (50% - Element Width / 2) ဖြစ်ရမှာပါ။ top လည်းအတူတူပါပဲ။ ဒီပြဿနာက Position မှာ တွေ့ရနေကြ ပြဿနာဖြစ်ပါတယ်။ ဒါကို Bootstrap က translate-middle ဆိုတဲ့ Class နဲ့ဖြေရှင်းပေးထားပါတယ်။ ဒီလိုပေးပါ။

### HTML

```
<div class="position-relative bg-dark" style="height: 200px">
    <div class="position-absolute
        bg-light top-50 start-50
        w-25 h-25
        translate-middle"></div>
</div>
```

ဒီတော့မှ တစ်ကယ့်အလယ်တည့်တည့်ကို ရောက်မှာဖြစ်ပါတယ်။ ဘာကိုပြောတာလဲ သိပ်မရှင်းရင် လက်တွေ့ချရေးပြီး နှစ်ခုနှင့်ယူဉ် စမ်းသပ်ကြည့်သင့်ပါတယ်။ ဒီလိုပါ -

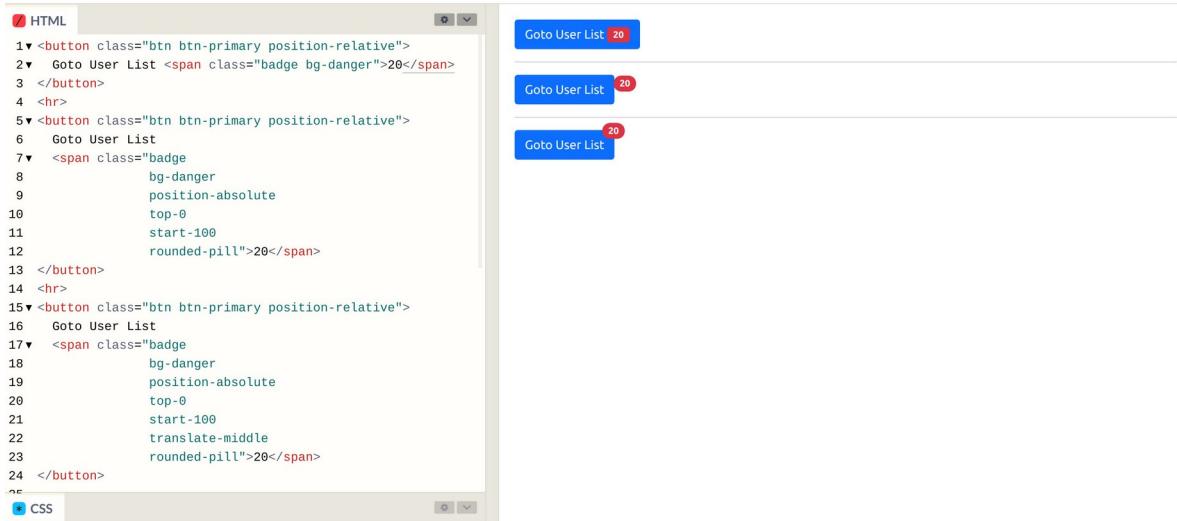


top-50, start-50 ပေးထားတာချင်းအတူတူ ပထမတစ်ခုက အလယ်တည့်တည့်မရောက်ဘဲ နောက်တစ်ခုက အလယ်တည့်တည့် ရောက်တယ်ဆိုတာကို တွေ့ရမှာဖြစ်ပါတယ်။ ဒါတော်တော် အသုံးဝင် တဲ့ လုပ်ဆောင်ချက်တစ်ခုပါ။ ပိုပြီးစိတ်ဝင်စားဖို့ကောင်းတာလေး စမ်းကြည့်ချင်ရင် ဒီကုဒ်ကို စမ်းကြည့်နိုင်ပါတယ်။

**HTML**

```
<button class="btn btn-primary position-relative">
    Goto User List
    <span class="badge
        bg-danger
        position-absolute
        top-0
        start-100
        translate-middle
        rounded-pill">20</span>
</button>
```

bth ထဲမှာ badge တစ်ခုရှိပြီး position-absolute နဲ့ translate-middle ကိုတွဲသုံးထားလို့ Notification စနစ်တွေမှာ တွေ့နေကြပုံစံအတိုင်း နှိမ်အရေအတွက်က Button ရဲ့ ထိပ်ဖျားမှာ ချိတ်နေတဲ့ ပုံစံကို ရရှိမှာဖြစ်ပါတယ်။



ပေးထားတဲ့ နမူနာမှာ (၃) မျိုးရေးပြထားပါတယ်။ ပထမတစ်ခုက ရှိုးရှိုးပါပဲ။ btn ထဲမှာ badge တစ်ခု ရှိနေပါတယ်။ ဒုတိယတစ်ခုမှာတော့ badge မှာ position-absolute ဖြစ်သွားပြီး start-100 လိုပြောလိုက်တဲ့ အတွက် ကျော်ထွက် သွားတာကို တွေ့ရနိုင် ပါတယ်။ တတိယတစ်ခု ကျေတော့မှ translate-middle ပါသွားလို့ ထိပ်ဖျားလေးမှာ ချိတ်နေတဲ့ပုံစံလေး ရသွားတာကို တွေ့ရမှာပါ။

## Shadow

Element တွေအတွက် အရိပ်ကျနေစဲပဲဖံ့ဌံ Drop Shadow တွေထည့်ချင်ရင် ထည့်လိုရအောင်လည်း ပေးထားပါသေးတယ်။ သူလည်းအသုံးဝင်ပါတယ်။ သုံးရတာလည်း လွယ်ပါတယ်။ shadow Class ကို သုံးပေးလိုက်ယုံပါပဲ။ shadow-sm နဲ့ shadow-lg ဆိုပြီး မူကွဲ (၂) မျိုးရှိပါတယ်။

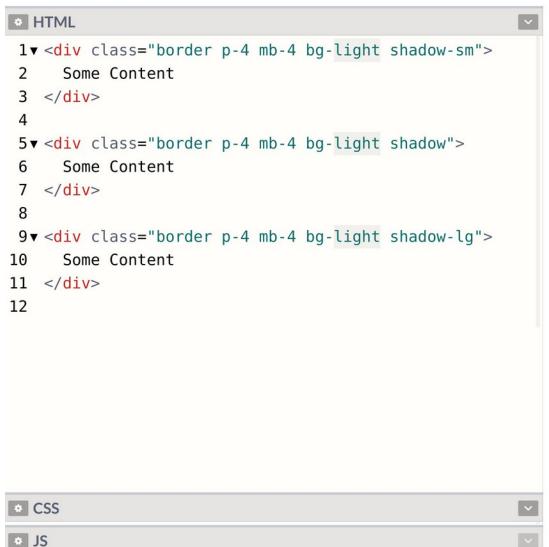
### HTML

```
<div class="border p-4 mb-4 bg-light shadow-sm">
    Some Content
</div>

<div class="border p-4 mb-4 bg-light shadow">
    Some Content
</div>

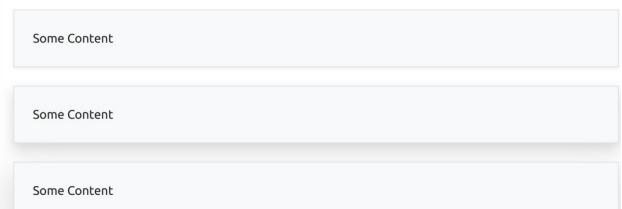
<div class="border p-4 mb-4 bg-light shadow-lg">
    Some Content
</div>
```

shadow-sm က အရိပ်ကို ပါးပါးလေး မသိမသာထည့်ပေးလိုက်မှာ။ shadow-lg ကတော့ အရိပ်ကို တော်တော်ကြီးကြီး ထည့်ပေးမှာပါ။ ဒီလိုပါ -



The screenshot shows a code editor interface with three sections: HTML, CSS, and JS. The HTML section contains the code provided above. The CSS and JS sections are currently empty.

```
1▼ <div class="border p-4 mb-4 bg-light shadow-sm">
2    Some Content
3 </div>
4
5▼ <div class="border p-4 mb-4 bg-light shadow">
6    Some Content
7 </div>
8
9▼ <div class="border p-4 mb-4 bg-light shadow-lg">
10   Some Content
11 </div>
12
```



ກືລາກົດຂຶ້ນ ໂຕວຕອນລະຄົມວູວະປີປີ|| ແອນົມຕັບອອນີ້:ຕົກລົງອົນີ້: ພຸມັນກາເສັ້ນັ້ນ ແລະ ເຕັມໆ. Admin Dashboard UI ່ລະຫຼິບຕົກລົງອົນີ້: ບຸນັບປີ|| ແລະ ອົບມູນ ສີ Utility Class ເຕັມໆ ລົກ້າເຕັມໆ ອົບມູນ ພຸມັນກາເສັ້ນັ້ນ ແລະ ດັບປິລາປີໄລດີໜີ້ມຍີ||

Admin Dashboard ພຸມັນກາລົບມຕົກລົງອົນີ້: ໄດ້ແລ້ວ ດັບປິລາປີໄລດີໜີ້ມຍີ|| ແລະ ອົບມູນ ສີ Icon ເຕັມໆ ດັບປິລາປີໄລດີໜີ້ມຍີ||

## အခန်း (၈) - Icons

Icons တွေဟာလည်း UI တွေဖန်တီးတည်ဆောက်ဖို့အတွက် အရေးကြီးပါတယ်။ နှုတ်တစ်ရာ စာတစ်လုံး လို့ ပြောကြသလို၊ A picture is worth a thousand words လို့လည်းပြောကြပါတယ်။ ပုံလေးတစ်ပုံ၊ Icon လေးတစ်ခုနဲ့ ရှုပ်နေတာတွေကို ရှင်းသွားစေနိုင်ပါတယ်။ ရောနေတာတွေကို ကဲပြားသွားစေနိုင်ပါတယ်။

Bootstrap 3 တုံးက Framework နဲ့အတူ Icons တွေ တစ်ခါတည်း ပါခဲ့ဖူးပါတယ်။ Bootstrap 4 မှာတော့ Icons တွေ ထည့်မပေးတော့လို့ Third-party Icons တွေနဲ့ တွဲသုံးကြပါတယ်။ လူသုံးအများဆုံးလို့ ပြောလို့ ရတဲ့ Icons နည်းပညာ တစ်ခုကတော့ Font Awesome ပဲ ဖြစ်ပါတယ်။ အခုံ Bootstrap 5 မထွက်ခေါ်လေး မှာ Bootstrap Icons ဆိုပြီးတော့ ဖြည့်စွက်နည်းပညာအသစ်တစ်ခုကို Bootstrap Framework တို့ထွင် သူများကပဲ ဖန်တီးပေးလာတာကိုလည်း တွေ့ရပါတယ်။

ဟိုးအရင်တုံးက Icon ဆိုရင် ICO တို့ GIF တို့ PNG တို့လို့ ပုံး Format တွေအနေနဲ့ အသုံးများခဲ့ကြပါတယ်။ ဒီပုံး Format တွေက အရွယ်အစားအားဖြင့် သေးငယ်ပြီး Background Transparency လိုလုပ်ဆောင်ချက် မျိုးတွေ ပါဝင်တဲ့အတွက် Icon နဲ့ သင့်တော်ပါတယ်။ ဒါပေမယ့် ဒီလို့ ပုံး Format ကို Icon အတွက်သုံးတဲ့ အတွက် ကြံးတွေ့ရတဲ့ ပြဿနာတွေ ရှိပါတယ်။ အဲဒီတဲ့က တစ်ခုကတော့ Icon တွေဆိုတာ အများကြီး သုံးရတာပါ။ ဝတ်ဆိုက်တစ်ခုမှာ Icon ပေါင်း ဆယ်ခု၊ အခုနှစ်ဆယ်ကနေ အခုငါးဆယ်၊ အခုတစ်ရာထိ လည်း ပါနိုင်ပါတယ်။ ဒီလောက် ဖိုင်အရေအတွက်များတဲ့အခါ ဝတ်ဆိုက်ကို နေးစေပေါ်တယ်။ ဝတ်ဆိုက်တစ်ခုက Icon ဖိုင်အခု (၅၀) သုံးထားရင် Web Browser က အကြိမ် (၅၀) ဆာဗာကိုဆက်သွယ်မှု ပြုလုပ်ရတဲ့ အတွက်ကြောင့်ပါ။ ဒါကြောင့် ဖိုင်တွေရဲ့အရွယ်အစားလေးတွေက သေးသေးလေးတွေပေမယ့် ဆက်သွယ် ရတဲ့ အကြိမ်ရေများလို့ နေးသွားစေတဲ့ သဘောမျိုး ဖြစ်ပါတယ်။ ဒီပြဿနာကို CSS Sprite လို့ နည်းစနစ် မျိုးတွေနဲ့ ဖြေရှင်းခဲ့ကြပါတယ်။ အခုသိပ်မသုံးကတော့လို့ ဒီအကြောင်းကို အကျယ်ချုပြီး ထည့်မရင်း တော့ပါဘူး။ စိတ်ဝင်စားရင် နောက်မှာ ရှာဖွေလေ့လာကြည့်ပါ။

နောက်တော့ CSS မှာ `@font-face` လုပ်ဆောင်ချက် ပါဝင်လာခြင်းနဲ့အတူ Icon Fonts တွေကို အသုံး များလာကြပါတယ်။ Icon Fonts ဆိုတာ A, B, C D တို့က, e, g, ဟု တို့လို စာလုံးတွေကိုရေးဆွဲထည့်သွင်းရတဲ့ ဖွန့်ဖိုင်ထဲမှာ စာလုံးတွေအစား ရုပ်ပုံတွေကို ရေးဆွဲထည့်သွင်းထားခြင်း ဖြစ်ပါတယ် လို့ အလွယ်ပြောနိုင်ပါတယ်။ ဒီတော့ Icon Fonts ကိုသုံးပြီးစာရေးရင် စာလုံးပုံတွေမပေါ်ဘဲ ရုပ်ပုံတွေ ပေါ်တယ်ဆိုတဲ့ သဘော မျိုးပါ။ ဒီ Icon Fonts နည်းစနစ်မှာ အားသာချက်တွေရှုပါတယ်။ ပထမဆုံးအားသာချက်ကတော့ ဖွန့်ဖိုင်တစ်ခုမှာစာလုံးတွေ အများကြီး ပါလိုရသလိုပဲ ပုံတွေလည်းအများကြီး ရေးဆွဲထည့်သွင်းထားလို့ရတဲ့ အတွက် ဖိုင်တစ်ခုထဲနဲ့ လိုချင်တဲ့ Icon တွေကို စုစုပေါင်းရရှိပြီး ဖြစ်ပေါ်ပါတယ်။ ဒါကြောင့် ရှိုးရှိုး ပုံးပုံး ပုံးပုံး Icon တွေ မှာ အရေအတွက်များလို့ နေးသွားတယ်ဆိုတဲ့ ပြဿနာကို Icon Fonts မှာ တွေ့ရမှာ မဟုတ်ပါဘူး။

Icon Fonts ရဲ့ ဒုတိယအားသာချက်ကတော့ ပုံအရည်အသွေးကို ပြောင်းလဲစေခြင်းမရှိဘဲ အရွယ်အစားကို လိုသလို ချုံလိုချုံလိုရခြင်း ဖြစ်ပါတယ်။ ရှိုးရှိုး GIF, PNG ပုံတွေက ချုံလိုက်ရင် ဝါးသွားတာတို့ ချုံလိုက်ရင် ကြည့်မကောင်းတော့တာတို့ ဖြစ်စေနိုင်ပါတယ်။ Bitmap Graphic တွေမို့လိုပါ။ A, B, C, D စာလုံးတွေမှာ ချုံလိုက်လို့ ဝါးသွားတာမျိုး မရှိသလိုပါပဲ၊ Icon Fonts ထဲက Icon ပုံတွေကိုလည်း လိုသလို အချို့အခြားလုပ်လိုနိုင်ပါတယ်။ Vector Graphic တွေမို့လိုပါ။ ပြီးတော့ ပုံးပုံးတွေကို ဆွဲထားရင် ဆွဲထားတဲ့အတိုင်းပဲ ရမှာပါ။ အနီရောင် ဆွဲထားရင် အနီရောင်ပဲ ရမှာပါ။ အပြာရောင် ပြောင်းချင်ရင် နောက်တစ်ပုံ ထပ်ဆွဲပြီး ထည့်မှုပဲရ ပါမယ်။ A, B, C, D စာလုံးတွေကို အရောင်လိုသလို ပြောင်းပြီး ပြုလိုရသလိုပဲ Icon Font ထဲက Icon ပုံတွေ ကိုလည်း လိုသလို အရောင်အမျိုးမျိုး ပြောင်းပြီး ပြုလိုရနိုင်ပါတယ်။ ဒီလိုအားသာချက်တွေကြောင့်ပဲ နောက်ပိုင်းမှာ Icons အတွက် ရှိုးရှိုးပုံကို မသုံးကြတော့သလောက်ပါပဲ။

ဒီလိုအားသာချက်တွေနဲ့အတူ အားနည်းချက်တစ်ခုလဲ တွေ့ပြီးတော့ပါလာပါတယ်။ Icon Font တစ်ခုမှာ Icon ပေါင်းများစွာထည့်သွင်းလို့ရတဲ့အတွက် တစ်ချို့ Icon Fonts တွေမှာ ပုံပေါင်း (၅) ထောင် (၆) ထောင် လောက်ထိ ပါနိုင်ပါတယ်။ ဒါကြောင့် Icon လေးတစ်ချို့ကို သုံးချင်လို့ Icon Font တစ်ခုကို ချိုတ်ဆက်လိုက်တာနဲ့ မလိုအပ်ဘဲ ရှိသမှု Icon တွေ အကုန်ထည့်သွင်းလိုက်ရသလို ဖြစ်ပေါ်ပါတယ်။

နောက်ထပ် ထပ်ပေါ်လာတာကတော့ SVG Icons ဖြစ်ပါတယ်။ SVG ဆိုတာ Scalable Vector Graphic ရဲ့ အတိုကောက်ဖြစ်ပြီးတော့ HTML နည်းပညာရဲ့ အစိတ်အပိုင်းတစ်ခု ဖြစ်ပါတယ်။ XML ရေးထုံးကိုသုံးပြီးတော့ HTML Document အတွင်းထဲမှာ Vector Graphic တွေကို ရေးဆွဲထည့်သွင်းစေနိုင်တဲ့ နည်းပညာ ဖြစ်ပါတယ်။ အဲဒီ SVG နည်းပညာကိုသုံးပြီး Icon တွေ တိတွင်လာကြတဲ့အခါ စောစောက Icon Font မှာ

လို ရှိသမျှအကုန်ထည့်ရတဲ့ ပြဿနာမျိုး မရှိတော့ဘဲ ကိုယ်လိုချင်တဲ့ Icon ကို HTML ထဲမှာ လိုသလောက် ပဲ ရွေးထည့်လို ရလာပါတယ်။ ပြီးတော့ ရှိုးရှိုး GIF, PNG ပုံတွေလို အခု (၅၀) သုံးထားလို အကြိမ် (၅၀) ဆက်သွယ်ရတယ်ဆိုတာမျိုးလည်း မဖြစ်ပါဘူး။ သူက ပုံကို HTML နဲ့ ချိတ်ထားတာ မဟုတ်ဘဲ၊ ပုံကို HTML အထဲမှာ တစ်ခါတည်း ရောရေးထားတဲ့သဘောမျိုး ဖြစ်သွားလိုပါ။ ဒါကြောင့် SVG Icon တွေကို တစ်ဖြည်းဖြည်း ပိုသုံးလာကြပါတယ်။

Font Awesome က Icon Font နည်းပညာဖြစ်ပါတယ်။ SVG Icons အနေနဲ့လည်း သုံးလိုရပါတယ်။ နှစ် မျိုးပေးထားတဲ့သဘောပါ။ Free နဲ့ Pro ဆိုပြီး Version နှစ်ခုလာရာမှာ အခုလက်ရှိထွက်ရှိထားတဲ့ Font Awesome 5 Pro Version မှာ Icon ပေါင်း (၇၀၀၀) ကျော်ပါဝင်ပါတယ်။ Pro Version က လိုင်စင်ဝယ်ပြီး သုံးရပါတယ်။ Free Version ကတော့ အခမဲ့ရပြီး Icon ပေါင်း (၁၀၀၀) ကျော်ပါဝင်ပါတယ်။ Icon ပေါင်း (၁၀၀၀) ကျော်ဆိုတာတင် တော်တော် စုံနေပြီမဲ့လို Free Version နဲ့တင် ပရောဂျက် တော်တော်များများ အတွက် အဆင်ပြေစေနိုင်လောက်ပါတယ်။ ဒီစာရေးနေချိန်မှာ Font Awesome 6 ထွက်တော့မယ်လို လည်း ကြေညာထားပါတယ်။ အသစ်ထွက်တာမကြာသေးတဲ့ Bootstrap Icons တွေကတော့ SVG Icon တွေဖြစ်ကြပါတယ်။ အခမဲ့ရပြီး သူမှာလည်း Icon ပေါင်း (၁၀၀၀) ကျော်ပါဝင်ပါတယ်။

## Font Awesome

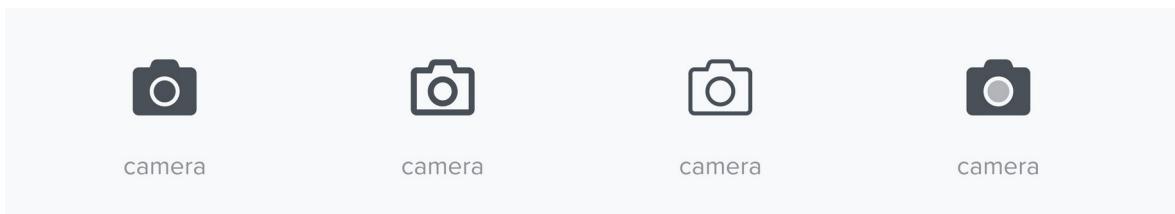
Font Awesome ကို စတင်အသုံးပြုနိုင်ဖို့အတွက် လိုအပ်တဲ့ဖိုင်တွေကို Download ရယူလို ရသလို CDN ကနေ တစ်ဆင့်လည်း အသုံးပြုနိုင်ပါတယ်။ ဒီနေရာမှာတော့ CDN ကနေချိတ်ပြီးတော့ပဲ နမူနာ ပေးသွားပါ မယ်။ Code Pen Setting ရဲ့ CSS Section မှာ ဒီ Font Awesome CDN လိပ်စာကို ထည့်ပေးလိုက်ပါ။

```
https://use.fontawesome.com/releases/v5.15.1/css/all.css
```

လက်ရှိဒီစာရေးနေချိန် ထွက်ရှိထားတာက v5.15.1 ဆိုတာကို သတိပြုပါ။ Font Awesome က နောက်ပိုင်း Version တွေမှာ အဲဒီလို CDN လိပ်စာ တိုက်ရှိက်မပေးတော့ဘူးလို သိရပါတယ်။ ဒါကြောင့် Version သစ် အတွက် CDN လိပ်စာကိုလိုချင်ရင် Font Awesome ဝေါ်ဆိုက်မှာ Register လုပ်ထားဖို့ လိုအပ်ပါလိမ့်မယ်။ အလွယ်တစ်ကူ Register လုပ်လိုရနိုင်ပါတယ်။

- <https://fontawesome.com>

Register လုပ်ပြီးရင်တော့ Version သစ်အတွက် CDN လိပ်စာတွေကို ရရှိပါလိမ့်မယ်။ နောက်ပိုင်းမှာ CDN လိပ်စာ ပြောင်းစရာမလိုဘဲ Version အသစ်ထွက်တိုင်း အလိုအလျောက် Update ဖြစ်စေဖို့ အဲဒီနည်းကို ပြောင်းသုံးတယ်လို့ ပြောထားပါတယ်။ လောလောဆယ် စမ်းကြည့်ဖို့သက်သက် အတွက်တော့ မလိုသေးပါဘူး။ အထက်မှာပြောထားတဲ့ CDN လိပ်စာကိုပဲ ဆက်လက်အသုံးပြုနိုင်ပါတယ်။ CDN လိပ်စာထည့်ပြီး သွားရင် အသုံးပြုနည်းက မခက်တော့ပါဘူး။



Font Awesome မှာ မူကွဲအပ်စု (၅) စုရှိပါတယ်။ အပေါ်က ကင်မရာပုံလေးတွေကို နှိုင်းယှဉ် ကြည့်ပါ။ ရှေ့ဆုံးပုံက Solid ဆိုတဲ့ အုပ်စုထဲမှာ ပါပါတယ်။ အရောင်အပြည့် ဖြည့်ပြီး ပုံကို ဆွဲထားပါတယ်။ ဒုတိယပုံက Regular အုပ်စုပါ။ လိုင်းလေးတွေနဲ့ ဆွဲထားပါတယ်။ တတိယပုံကတော့ Light အုပ်စုဖြစ်ပါတယ်။ လိုင်းပါးပါးလေးနဲ့ ဆွဲထားတာပါ။ နောက်ဆုံးပုံကိုတော့ Duo-Tone လိုခေါ်ပါတယ်။ နှစ်ရောင်စပ်ပြီး ဆွဲထားတာပါ။ Free Version မှာ Solid Icons တွေနဲ့ Regular Icons တစ်ချို့ကို အသုံးပြုခွင့် ပေးထားပါတယ်။ Light နဲ့ Duo-Tone Icons တွေကတော့ Pro Version ကျတော့မှ သုံးလို့ရမှာပါ။ နမူနာပုံထဲမှာ မပါတဲ့ အုပ်စုကိုတော့ Brand လိုခေါ်ပါတယ်။ Google တို့ Facebook တို့ YouTube တို့တို့ Twitter တို့တို့ Brand တွေရဲ့ Icons တွေပါ။ Brand Icons တွေကိုလည်း Free Version မှာ ထည့်ပေးထားပါတယ်။ ဒါကြောင့် Free Version မှာ Icons အုပ်စု (၃) ခုပါတယ်လို့ မှတ်နိုင်ပါတယ်။ Solid, Regular နဲ့ Brand တို့ဖြစ်ပါတယ်။ အသုံးပြုဖို့အတွက် အခုလိုရေးသားအသုံးပြုနိုင်ပါတယ်။

### HTML

```
<i class="fas fa-camera"></i>
<i class="far fa-user"></i>
<i class="fab fa-github"></i>
```

ဒီနမူနာကို Code Pen ထဲမှာ လက်တွေ့ရေးစမ်းကြည့်လိုပါတယ်။ CDN တော့ ကြိုပြီးမှန်အောင် ထည့်ထားပေးဖို့ မမေ့ပါနဲ့။ သင့်တော်တဲ့ Icon ပုံလေးတွေ ပေါ်လာတာကို တွေ့ရပါလိမ့်မယ်။

Solid Icons တွက်အသုံးပြုလိုရင် `fas` Class ကိုသုံးရပြီး Regular Icons တွက် သုံးချင်ရင်တော့ `far` Class ကိုသုံးရပါတယ်။ Brand Icons တွက် သုံးချင်ရင်တော့ `fab` Class ကိုသုံးရပါတယ်။ ပြီးတဲ့အခါနာက်ကနေ အသုံးပြုလိုတဲ့ပုံရဖို့ `fa-{icon-name}` Class လိုက်ရပါတယ်။ Icon Name တွကတော့ (၁၀၀၀) ကျော် အကုန်လုံးမှတ်ထားဖို့ မဖြစ်နိုင်ပါဘူး။ လိုအပ်လာတော့မှာ ဒီလိပ်စာမှာ ကိုယ်လိုချင်တဲ့ Keyword နဲ့ရိုက်ထည့်ပြီး ရှာသုံးသွားရမှာ ဖြစ်ပါတယ်။

<https://fontawesome.com/icons>

တစ်ကယ်တော့ Font Awesome အသုံးပြုနည်းက ဒီမှာတင်ပြီးသွားပါပြီ။ ဒါပေမယ့် လက်တွေရေးစမ်းဖြစ်သွားအောင် Bootstrap Component တွေနဲ့တဲ့ပြီး နမူနာတစ်ချို့ ပေးချင်ပါတယ်။ ရေးစမ်းကြည့်ပါ။

### HTML

```
<ul class="list-group">
  <li class="list-group-item">
    <i class="far fa-square me-3"></i>
    Something todo
    <i class="fas fa-trash float-end text-danger"></i>
  </li>
  <li class="list-group-item">
    <i class="fas fa-check me-3"></i>
    Another thing to do
    <i class="fas fa-trash float-end text-danger"></i>
  </li>
  <li class="list-group-item">
    <i class="fas fa-check me-3"></i>
    More thing to do
    <i class="fas fa-trash float-end text-danger"></i>
  </li>
</ul>
```

ဒါဟာ List Group Component ထဲမှာ Checkbox Icons လေးတွေ၊ Trash Icons လေးတွေ ပေါင်းစပ်ပြီး Todo List App UI လေးတစ်ခု ဖန်တီးလိုက်တာပါ။ ရလဒ်က အခုလိုဖြစ်ပါလိမ့်မယ်။

နောက်နမူနာတစ်ခုအနေနဲ့ Tab UI မှာ Icons လေးတွေ ထည့်ကြည့်ပါမယ်။



The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab is active, displaying the following code:

```

1▼ <ul class="list-group">
2  <li class="list-group-item">
3    <i class="far fa-square me-3"></i>
4    Something todo
5    <i class="fas fa-trash float-end text-danger"></i>
6  </li>
7  <li class="list-group-item">
8    <i class="fas fa-check me-3"></i>
9    Another thing to do
10   <i class="fas fa-trash float-end text-danger"></i>
11 </li>
12  <li class="list-group-item">
13    <i class="fas fa-check me-3"></i>
14    More thing to do
15   <i class="fas fa-trash float-end text-danger"></i>
16 </li>
17 </ul>
18

```

The CSS tab contains the following styles:

```

* CSS

```

The JS tab is empty.

ပုံမှန်သုံးနေကြ Tab UI လေးကိုပဲ Icons လေးတွေထည့်လိုက်လို ပိုအသက်ဝင်သွားတာကို အခုလို တွေမြင်ရမှာ ဖြစ်ပါတယ်။

The screenshot shows a code editor with three tabs: HTML, CSS, and JS. The HTML tab contains the following code:

```

1<ul class="nav nav-tabs">
2  <li class="nav-item">
3    <a href="#" class="nav-link active">
4      <i class="fas fa-list"></i> All Users
5    </a>
6  </li>
7  <li class="nav-item">
8    <a href="#" class="nav-link">
9      <i class="fas fa-users"></i> New Users
10   </a>
11 </li>
12 <li class="nav-item">
13   <a href="#" class="nav-link">
14     <i class="fas fa-user"></i> My Profile
15   </a>
16 </li>
17 </ul>
18

```

The CSS and JS tabs are empty. To the right of the code editor is a browser window showing a navigation bar with three items: 'All Users' (active), 'New Users', and 'My Profile'. Each item has a corresponding icon.

နောက်တစ်ခုအနေနဲ့ Form မှာ Icons လေးတွေ ထည့်ကြည့်ကပါမယ်။

**HTML**

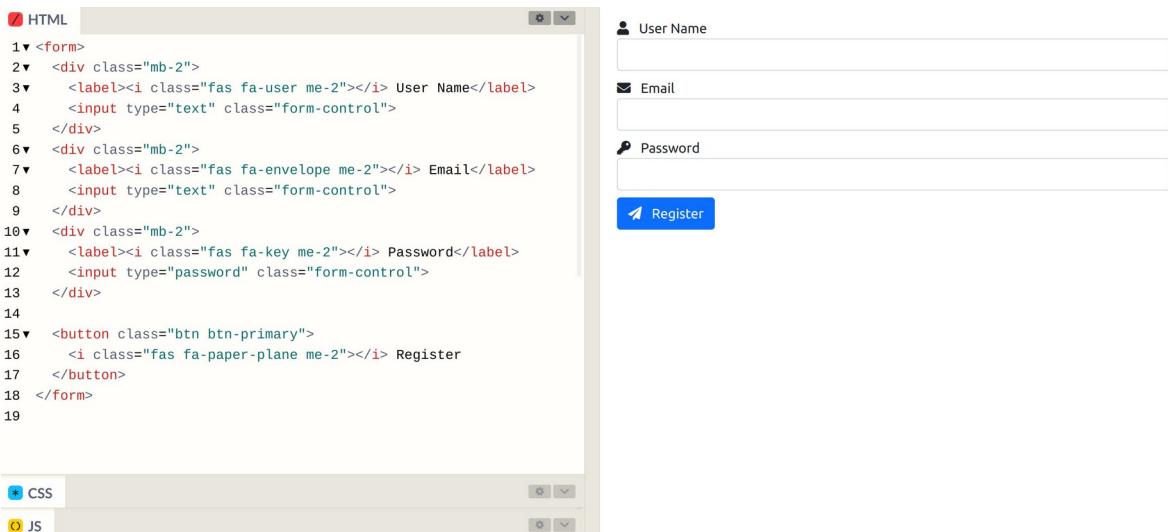
```

<form>
  <div class="mb-2">
    <label><i class="fas fa-user me-2"></i> User Name</label>
    <input type="text" class="form-control">
  </div>
  <div class="mb-2">
    <label><i class="fas fa-envelope me-2"></i> Email</label>
    <input type="text" class="form-control">
  </div>
  <div class="mb-2">
    <label><i class="fas fa-key me-2"></i> Password</label>
    <input type="password" class="form-control">
  </div>

  <button class="btn btn-primary">
    <i class="fas fa-paper-plane me-2"></i> Register
  </button>
</form>

```

ဒီတစ်ခါမှာလဲ ပုံစံတူဖို့လို့ ရောတတ်တဲ့ Form Input တွေဟာ Icons လေးတွေကြောင့် ကွဲပြားသွားတာကို အချလို တွေ့ရမှာ ဖြစ်ပါတယ်။



နောက်တစ်ခုအနေနဲ့ Table မှာ Icons လေးတွေ ထည့်ကြည့်ကြပါမယ်။

<i class="fas fa-hashtag"></i>	ID
<i class="fas fa-bug"></i>	Issue
<i class="fas fa-calendar"></i>	Date
<i class="fas fa-sort-amount-down"></i>	Level
<i class="fas fa-user"></i>	Assign

1	A problem with something
2020-08-11	
<span class="badge bg-danger">critical</span>	
Alice	

2	Another problem with ...
2020-08-12	
<span class="badge bg-warning">important</span>	
Bob	

Feature တွေ ပိုပြီးတော့ ဝေဆာင်ရန် ဖြစ်သွားသလို Table ပါ အချက်အလက်တွေကိုလည်း ပိုပြီးတော့

ရှင်းလင်းမြင်သာသွားတာကို အခုလို တွေ့ရမှာ ဖြစ်ပါတယ်။



The screenshot shows a code editor interface with three tabs: HTML, CSS, and JS. The HTML tab displays the following code:

```

1<table class="table table-bordered table-striped">
2  <tr>
3    <th><i class="fas fa-hashtag"></i> ID</th>
4    <th><i class="fas fa-bug"></i> Issue</th>
5    <th><i class="fas fa-calendar"></i> Date</th>
6    <th><i class="fas fa-sort-amount-down"></i> Level</th>
7    <th><i class="fas fa-user"></i> Assign</th>
8  </tr>
9  <tr>
10    <td>1</td>
11    <td>A problem with something</td>
12    <td>2020-08-11</td>
13    <td><span class="badge bg-danger">critical</span></td>
14    <td>Alice</td>
15  </tr>
16  <tr>
17    <td>2</td>
18    <td>Another problem with ...</td>
19    <td>2020-08-12</td>
20    <td><span class="badge bg-warning">important</span></td>

```

The CSS and JS tabs are currently empty.

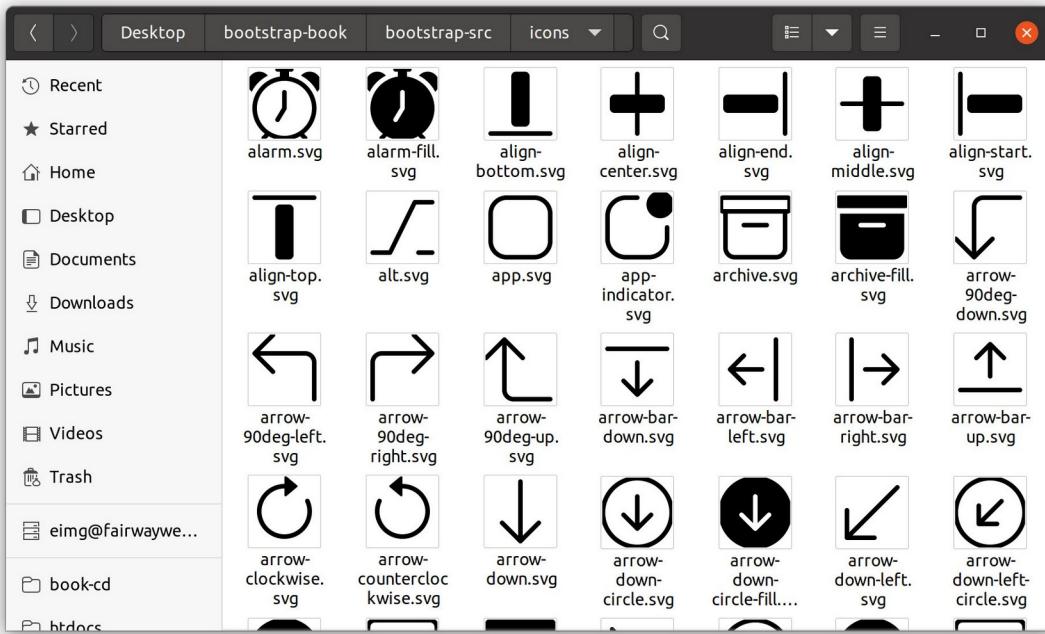
ဒီလောက်ဆိုရင် Icons တွေရဲ့ အသုံးဝင်ပုံနဲ့ Font Awesome Icons တွေ အသုံးပြုပုံကို ကောင်းကောင်း သဘောပေါက်သွားလောက်ပါ၍။ ဆက်လက်ပြီးတော့ Bootstrap Icons အသုံးပြုပုံလေးတွေ ဆက်ပြောပါ ဦးမယ်။

## Bootstrap Icons

Bootstrap Icons တွေ အသုံးပြုပုံကလည်း လွယ်ပါတယ်။ Icon Font CND အနေနဲ့လည်း ရနိုင်ပေမယ့် မူလစတင်တိထွင်စဉ်က SVG Icon တွေ အနေနဲ့ တိထွင်ခဲ့တာမို့လို့ SVG Icon အနေနဲ့ အသုံးပြုပုံကို ဖော်ပြ သွားပါမယ်။ သူ့ ဝတ်ဆိုက်ကနေ Download ရယူနိုင်ပါတယ်။

- <https://icons.getbootstrap.com>

Download လုပ်ရမယ်ဆိုပေမယ့် Icons ပေါင်း (၁၀၀၀) ကျော်ကိုမှ ဖိုင် Size က 650KB လောက်ပဲရှိတာ ပါ။ ရလာတဲ့ Zip ဖိုင်ကို ဖြည့်ချလိုက်ရင် အခုလို SVG Format နဲ့ Icons ဖိုင်တွေကို တွေ့မြင်ရမှာ ဖြစ်ပါ တယ်။



အဲဒီဖိုင်တွေကိုပဲ တစ်ခုချင်း <img> Element နဲ့ ထည့်သွင်းအသုံးပြုလို ရပါတယ်။ ဥပမာ -

### HTML

```

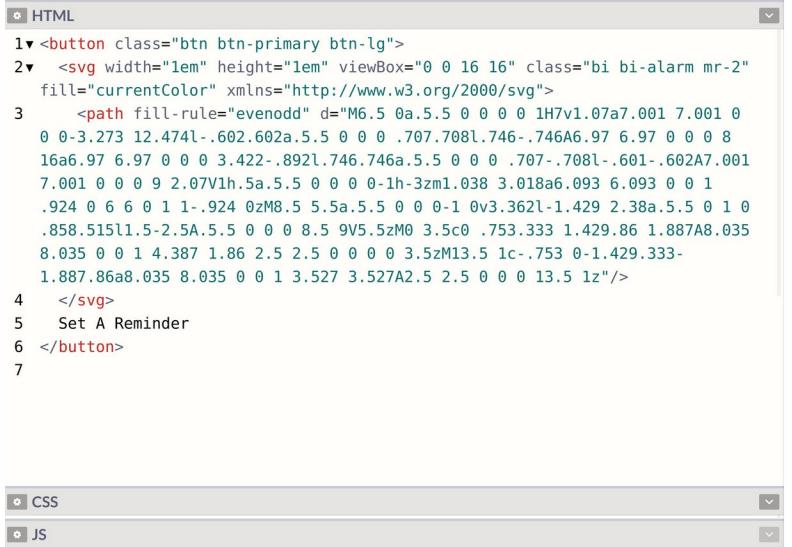
```

ဒါမှမဟုတ် နှစ်သက်ရာပုံကို Code Editor နဲ့ဖွင့်လိုက်ပါ။ ပုံဆိုပေမယ့် XML Format နဲ့ ရေးထားတဲ့ Text တွေပဲမို့လို Code Editor နဲ့ဖွင့်လို ရပါတယ်။ ရလာတဲ့ ကုဒ်က ဒီလိုပုံစံဖြစ်နိုင်ပါတယ်။

### SVG

```
<svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-alarm"
fill="currentColor" xmlns="http://www.w3.org/2000/svg">
  <path fill-rule="evenodd" d="M6.5 0a.5.5 0 0 0 1H7v1.07a7.001 7.001 0 0 0
3.273 12.4741-.602.602a.5.5 0 0 0 .707.7081.746-.746A6.97 6.97 0 0 0 8 16a6.97
6.97 0 0 0 3.422-.8921.746.746a.5.5 0 0 0 .707-.7081-.601-.602A7.001 7.001 0 0
0 9 2.07V1h.5a.5.5 0 0 0 0-1h-3zm1.038 3.018a6.093 6.093 0 0 1 .924 0 6 6 0 1
1-.924 0zM8.5 5.5a.5.5 0 0 0 0-1 0v3.3621-1.429 2.38a.5.5 0 1 0 .858.51511.5-
2.5A.5.5 0 0 0 8.5 9V5.5zM0 3.5c0 .753.333 1.429.86 1.887A8.035 8.035 0 0 1
4.387 1.86 2.5 2.5 0 0 0 0 3.5zM13.5 1c-.753 0-1.429.333-1.887.86a8.035 8.035
0 0 1 3.5z"/>
</svg>
```

ဒါ alarm.svg မှာပါတဲ့ ကုဒ္ပဗေဒဖြစ်ပါတယ်။ ဒီကုဒ်အတိုင်း ကိုယ်သုံးလိုတဲ့ Component ထဲမှာ ထည့်ပြီးတော့ သုံးနိုင်ပါတယ်။ ဒီလိုပါ -



```

HTML
1▼ <button class="btn btn-primary btn-lg">
2▼   <svg width="1em" height="1em" viewBox="0 0 16 16" class="bi bi-alarm mr-2"
3     fill="currentColor" xmlns="http://www.w3.org/2000/svg">
4     <path fill-rule="evenodd" d="M6.5 0a.5.5 0 0 0 1H7v1.07a7.001 7.001 0
0 0-3.273 12.474l-.602.602a.5.5 0 0 0 .707.708l.746-.746A6.97 6.97 0 0 0
16a6.97 6.97 0 0 0 3.422-.892l.746.746a.5.5 0 0 0 .707-.708l-.601-.602A7.001
7.001 0 0 0 9 2.07V1h.5a.5.5 0 0 0 0-1h-3zm1.038 3.018a6.093 6.093 0 0 1
.924 0 6 6 0 1 -.924 0zM8.5 5.5a.5.5 0 0 0 0-1 0v3.362l-1.429 2.38a.5.5 0 1 0
.858.515l1.5-2.5A.5.5 0 0 0 8.5 9V5.5zM0 3.5c0 .753.333 1.429.86 1.887A8.035
8.035 0 0 1 4.387 1.86 2.5 2.5 0 0 0 3.5zM13.5 1c-.753 0-1.429.333-
1.887.86a8.035 8.035 0 0 1 3.527 3.527A2.5 2.5 0 0 0 13.5 1z"/>
5   Set A Reminder
6 </button>
7

```

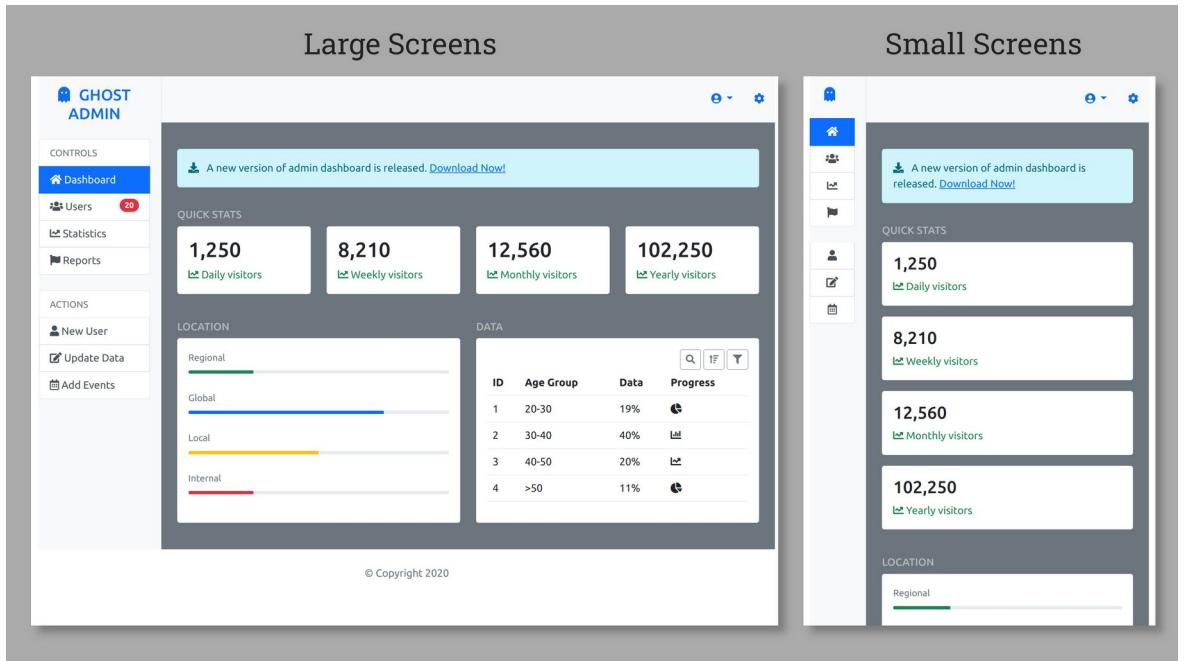
ဒါ Button Component ထဲမှာ SVG Icon ကို ထည့်သုံးလိုက်တာပါ။ ဒီ Icon ပေါ်မှာ အရွယ်အစား၊ အရောင်၊ အပါအဝင် လိုအပ်တဲ့ Style တွေကို သတ်မှတ်လိုက်ပါတယ်။

အားသာချက်အားနည်းချက်တွေကတော့ သူ့နေရာနဲ့သူပါပဲ။ Icon Font ကတော့ Class လေးတွေ ပေးလိုက်ယုံနဲ့ လိုချင်တဲ့ Icon ကိုရလို အသုံးပြုရတာ ပိုပြီးတော့ အဆင်ပြုလွယ်ကူတဲ့သဘောမှာ ရှိပါတယ်။ ဒါပေမယ့် SVG Icons တွေရဲ့ အားသာချက်အနေနဲ့ JavaScript နဲ့တွဲသုံးပြီး Animation အပါအဝင် ပိုပြီးဆန်းပြားတဲ့ လုပ်ဆောင်ချက်တွေ ရရှိနိုင်ပါသေးတယ်။ ဒီအကြောင်းတွေကိုတော့ အခုထည့်ကြည့်လိုမရသေးပါဘူး။ ဆက်လက်လေ့လာရန်လိုပဲ မှတ်ထားရှုံးမှာပါ။

နောက်တစ်ခန်းမှာ လက်ရှိလေ့လာခဲ့ပြီးဖြစ်တဲ့ Component တွေ Layouts တွေနဲ့ Icons တွေကို ပေါင်းစပ် ဖန်တီးထားတဲ့ နှမူနာ Admin Dashboard UI လေးတစ်ခုကို ဖန်တီးကြည့်ကြပါမယ်။

## အခန်း (၉) – Admin Dashboard

ရျေပိုင်းမှာလေ့လာခဲ့တဲ့ Component တွေနဲ့ Icons တွေကို လက်တွေ့စမ်းသပ် လေ့ကျင့် နှင့် ဖော်အတွက် Admin Dashboard UI တစ်ခုကို နမူနာအနေတဲ့ ဖန်တီးကြည့်ချင်ပါတယ်။ ပြီးသွားတဲ့အခါ ရလာမယ့်ရလဒ်က ဒီလိပါ –



Responsive လုပ်ဆောင်ချက်ကို တစ်ခါတည်း ထည့်လုပ်မှာဖြစ်ပြီး Large Screens တွေမှာ မြင်တွေ့ရမယ့် ရလဒ်နဲ့ Small Screens တွေမှာ မြင်တွေ့ရမယ့် ရလဒ်ကို ယူညွှန်ပြထားပါတယ်။ သေချာဂရရှိက် ကြည့်လိုက်ရင် အများကြီးကွာသွားတာမျိုး မဟုတ်ပါဘူး။ Large Screens တွေမှာ ဘယ်ဘက်ခြမ်း Sidebar Menu ဖော်ပြတဲ့အခါ Icon နဲ့ စာ တွဲပြပြီး၊ Small Screens တွေမှာ Icon တွေချည်းပဲ ပြလိုက်တာပါ။ ပြီးတော့ Large Screens အတွက် Main Content ဧရိယာမှာ Block လေးတွေကို ဘေးချင်းကပ် ဖော်ပြရာကနေ Small Screens အတွက် အပေါ်အောက်စီပြီး ပြလိုက်တာပါပဲ။

## Step-1 – HTML Structure

တစ်ဆင့်ချင်း Step by step ပြောပြချင်သလို၊ တစ်ခါတည်း အဆင့်လိုက် လိုက်လုပ်ကြည့်စေချင်ပါတယ်။ ရေးရမယ့်ကုပ်တွေများအတွက် Code Pen ကို မသုံးတော့ဘဲ၊ ကိုယ့်ဘာသာ HTML Document တစ်ခု တည်ဆောက်ပြီးတော့ ရေးကြည့်သင့်ပါတယ်။ ပထမအဆင့်အနေနဲ့ လိုအပ်တဲ့ အခြေခံ HTML Structure ကိုရေးပေးရပါမယ်။

### HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Dashboard</title>
    <link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
    >
    <link rel="stylesheet"
    href="https://use.fontawesome.com/releases/v5.15.1/css/all.css">
</head>
<body>

    <script
    src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/
bootstrap.bundle.min.js"></script>
</body>
</html>
```

အခြေခံ HTML Structure တစ်ခုဖြစ်ပြီး viewport လိုခေါ်တဲ့ <meta> Element ကိုသတိပြုပါ။ Responsive Layout အတွက် မဖြစ်မနေ ပါဝင်ဖို့လိုအပ်တဲ့ Element ဖြစ်ပါတယ်။ ဒါ Element မပါရင် Browser တွေက Screen Size သေးတဲ့အခါ ဝဘ်ဆိုက်ကိုဆန့်သွားအောင် ဆွဲချုံပြီး ပြတတ်ကြပါတယ်။ ဒါတော့အကုန်လုံး သေးသေးလေးတွေ ဖြစ်ကုန်ပြီး သုံးရတာအဆင်မပြေတော့ပါဘူး။ Viewport Meta Element က အဲဒီလို မချုံဘဲ သူအရှိအတိုင်းပြပေးဖို့ ပြောထားတာပါ။ Screen သေးလိုမဆန့်ရင် အဆင်ပြအောင်ပြတဲ့ Responsive လုပ်ဆောင်ချက်ကို ကိုယ့်ဘာသာ လုပ်ပေးမှာမို့လို့ Browser ဘက်က လုပ်ဖို့မလိုတဲ့သော့ ဖြစ်ပါတယ်။

ပြီးတဲ့အခါ လိုအပ်တဲ့ CSS နဲ့ JavaScript တွေကို CDN ကနေချိတ်ပြီး ထည့်ထားလိုက်ပါတယ်။ တစ်လုံး ချင်းကူးရေးမယ့်အစား သက်ဆိုင်ရာ Documentation ကနေ CDN လိပ်စာကို ကူးယူသင့်ပါတယ်။

- <https://getbootstrap.com>
- <https://fontawesome.com>

HTML/CSS မှာ မှားစရာ သိပ်မရှိပါဘူး။ အများဆုံး မှားကြရင် ချိတ်ဆက်ထားတဲ့ လိပ်စာတွေလဲနေကြတာ များပါတယ်။ ကူးရေးမယ်ဆိုရင်လည်း သေသေချာချာလေး ဂရုစိုက်ပြီးရေးပေးပါ။

## Step-2 – Sidebar Navigation

ဘယ်လက် Sidebar နဲ့ Main နောက် နှစ်ခြမ်းခွဲပြီး ပြချင်လို့ Layout လုပ်ဆောင်ချက်တစ်ခု စတည့်ပါ မယ်။ <body> အတွင်းမှာ ဒီလိုထည့်ရေးပေးပါ။

### HTML

```
<div class="container-fluid">
  <div class="row g-0">
    <nav class="col-2 bg-light pe-3">
      </nav>
    <main class="col-10 bg-secondary">
      </main>
    </div>
  </div>
```

Fluid Container တစ်ခုအတွင်းမှာ <nav> အတွက် Column (၂) ခုစာနေရာယူပြီး <main> အတွက် Column (၁၀) ခုစာနေရာယူထားတာပါ။ တစ်ခုနဲ့တစ်ခုကြေားထဲမှာ Gutter မရှိစေချင်လို့ g-0 Class ကို ထည့်ပေးထားတာ သတိပြုပါ။

Logo နဲ့ Title ဆက်ထည့်ကြပါမယ်။ <nav> အတွင်းတဲ့မှာ ဒီလိုရေးပေးပါ။

**HTML**

```
<div class="container-fluid">
    <div class="row g-0">
        <nav class="col-2 bg-light pe-3">
            <h1 class="h4 py-3 text-center text-primary">
                <i class="fas fa-ghost me-2"></i>
                <span class="d-none d-lg-inline">
                    GHOST ADMIN
                </span>
            </h1>
        </nav>
        <main class="col-10 bg-secondary">
            </main>
    </div>
</div>
```

<h1> Element တစ်ခုအတွင်းမှာ Icon လေးတစ်ခုနဲ့အတူ ခေါင်းစဉ်တပ်ပေးလိုက်တာပါ။ <h1> က အရမ်းကြီးနေမှာ စိုးလို့ h4 Class ထည့်ပေးထားပါတယ်။ ဒါကြောင့် <h1> ဆိုပေမယ့် အရွယ်အစားကို h4 အရွယ်အစားလောက်နဲ့ ပြပေးမှာပါ။ ကျွန်တဲ့လုပ်ဆောင်ချက်တွေက ဆန်းပြားတာ မပါဘဲ ရှုံးမှာလေ့လာ ခဲ့ဖြီးသား လုပ်ဆောင်ချက်တွေပဲမို့လို့ ကိုယ့်ဘာသာ ကုဒ်ကိုဖတ်ပြီး လေ့လာကြည့်လိုက်ပါ။

ခေါင်းစဉ်မှာ d-none လို့ ပြောထားတဲ့အတွက် Small Screen တွေမှာ ပျောက်နေမှာပါ။ d-lg-inline လို့ထပ်ပြောထားတဲ့အတွက် Large Screen တွေမှာတော့ ပေါ်နေမှာဖြစ်ပါတယ်။ Responsive လုပ်ဆောင်ချက်တစ်ခုအနေနဲ့ ထည့်ပေးထားတာပါ။

ဆက်လက်ပြီး Menu Item တွေကို List Group သုံးပြီး အခုလို ထည့်ပေးပါ။ <h1> ရဲအောက်မှာ ကပ်ပြီး ထည့်ရမှာပါ။

**HTML**

```
<div class="list-group text-center text-lg-start">
    <span class="list-group-item disabled d-none d-lg-block">
        <small>CONTROLS</small>
    </span>
    <a href="#" class="list-group-item list-group-item-action active">
        <i class="fas fa-home"></i>
        <span class="d-none d-lg-inline">Dashboard</span>
    </a>
```

```

<a href="#" class="list-group-item list-group-item-action">
    <i class="fas fa-users"></i>
    <span class="d-none d-lg-inline">Users</span>
    <span class="d-none d-lg-inline badge bg-danger rounded-pill float-end">20</span>
</a>
<a href="#" class="list-group-item list-group-item-action">
    <i class="fas fa-chart-line"></i>
    <span class="d-none d-lg-inline">Statistics</span>
</a>
<a href="#" class="list-group-item list-group-item-action">
    <i class="fas fa-flag"></i>
    <span class="d-none d-lg-inline">Reports</span>
</a>
</div>

<div class="list-group mt-4 text-center text-lg-start">
    <span class="list-group-item disabled d-none d-lg-block">
        <small>ACTIONS</small>
    </span>
    <a href="#" class="list-group-item list-group-item-action">
        <i class="fas fa-user"></i>
        <span class="d-none d-lg-inline">New User</span>
    </a>

    <a href="#" class="list-group-item list-group-item-action">
        <i class="fas fa-edit"></i>
        <span class="d-none d-lg-inline">Update Data</span>
    </a>
    <a href="#" class="list-group-item list-group-item-action">
        <i class="far fa-calendar-alt"></i>
        <span class="d-none d-lg-inline">Add Events</span>
    </a>
</div>

```

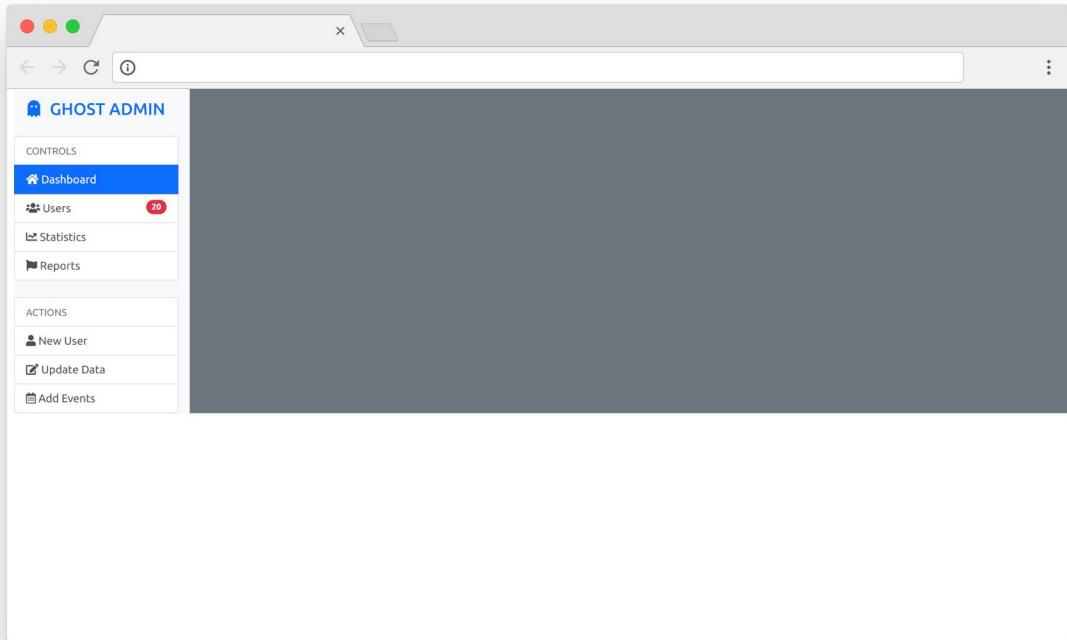
ရေးရမယ့်ကုဒ်တွေများပေါ်ပေါ် ကြိုးစားပြီးရေးပေးပါ။ များများရေးမှာ အလေ့အကျင့်ရပြီး မြန်မြန် ကျမ်းကျင့်မှာပါ။ တစ်ကယ်တော့ ကုဒ်တွေသာများနေတာပါ အဆန်းအပြားတော့ မပါပါဘူး။ အများစုက ကြိုးတင်လေ့လာခဲ့ကြပြီးသား လုပ်ဆောင်ချက်တွေပါပဲ။ ထူးခြားချက်အနေနဲ့ List Group တွေမှာ text-center လို့ ပြောထားတဲ့အတွက် စာတွေကို Center Alignment နဲ့ပြပါလိမ့်မယ်။ ဒါပေမယ့် text-lg-left လို့ပြောထားတဲ့အတွက် Large Screen တွေမှာတော့ Left Alignment နဲ့ပြမှာဖြစ်ပါတယ်။ Screen သေးသွားလို့ စာတွေကိုဖျောက်ပြီး Icon တွေချည်းပြတဲ့အခါ အလယ်မှာပြမှ ကြည့်ကောင်းမှာပါ။

ပြီးတဲ့အခါ List Group နဲ့ List Item တွေအတွက် <ul><li> ကိုမသုံးပါဘူး။ <div><a> ကိုသုံးထားပါတယ်။ နိုင်လို့ရတဲ့ Link တွေဖြစ်စေချင်လိုပါ။ နိုင်လို့ရတဲ့ Link မှန်းပေါ်လွင်အောင် List Item တွေမှာ

list-group-item-action လိုခေါ်တဲ့ Class တွဲထည့်ပေးထားတာကို သတိပြုပါ။ ကိုယ်တိုင် ပါရင် တစ်မျိုး၊ မပါရင်တစ်မျိုး စမ်းကြည့်နိုင်ပါတယ်။

Screen သေးတဲ့အခါ ဖျောက်ထားရမယ့် Element အားလုံးမှာ d-none Class ထည့်ပေးထားပါတယ်။ Screen ကြီးတဲ့အခါ ဖော်ပြုဖော်အတွက် d-lg-block နဲ့ d-lg-inline တို့ကို သူနေရာနဲ့သူ သုံးပေးထားပါတယ်။ လေ့လာကြည့်လိုက်ပါ။

ဒီအဆင့်ထိရေးပြီးပြီဆိုရင် စစမ်းလိုရပါပြီ။ ရေးထားတဲ့ HTML Document ကို Save ပြီး Browser တစ်ခုနဲ့ဖွင့်ကြည့်လိုက်ရင် အခုလိုရလာတ်ကို တွေ့မြင်ရမှာပဲဖြစ်ပါတယ်။



လိုချင်တဲ့အတိုင်း Sidebar Navigation နဲ့ Content ဧကယာအလွတ်တစ်ခုကို ရရှိနေခြင်းဖြစ်ပါတယ်။ ခေါင်းစီးတွေ Icon တွေအပြည့်အစုံပါသလို Responsive လုပ်ဆောင်ချက်လည်းပါလို့ Screen Size အကြီးအသေး အမျိုးမျိုးလည်း စမ်းလိုရနေပါပြီ။ ဘာမှာခက်ခက်ခဲ့ခဲ့ သိပ်မလုပ်လိုက်ရဘဲ ရရှိနေခြင်း ဖြစ်ပါတယ်။

### Step-3 – Horizontal Navbar

Main ရေးယာအပေါ်ပိုင်းမှာ Navbar တစ်ခုဆက်ထည့်ကြပါမယ်။ <main> အဖွင့်အပိတ်အတွင်းမှာ ဒီလို ရေးထည့်ပေးပါ။

#### HTML

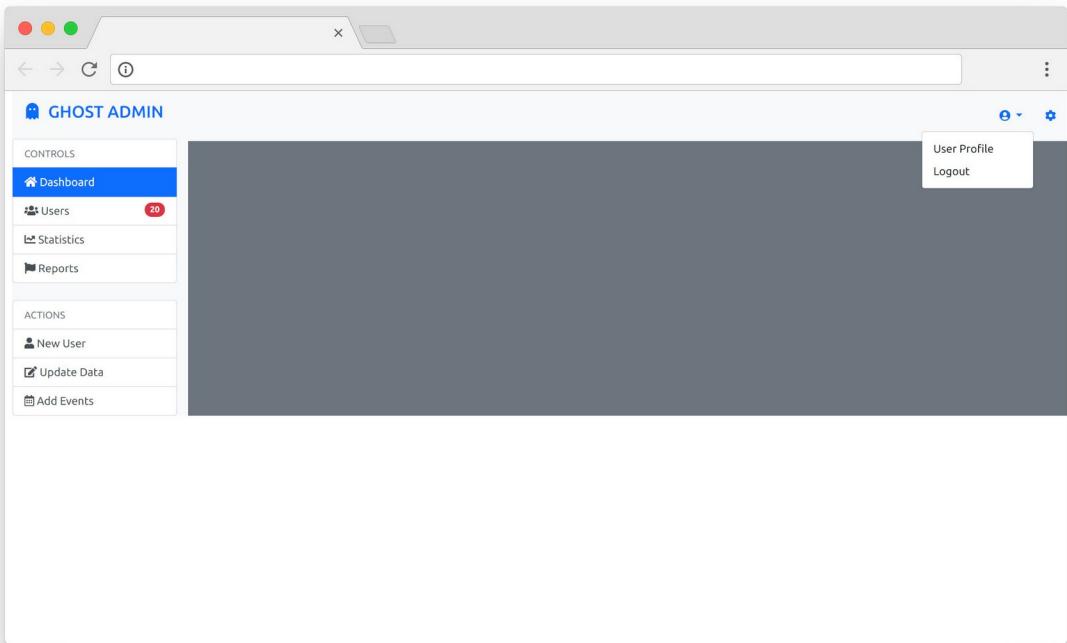
```
<nav class="navbar navbar-expand-lg navbar-light bg-light">

    <div class="flex-fill"></div>

    <div class="navbar nav">
        <li class="nav-item dropdown">
            <a href="#" class="nav-link dropdown-toggle"
               data-bs-toggle="dropdown">
                <i class="fas fa-user-circle"></i>
            </a>

            <ul class="dropdown-menu">
                <li>
                    <a href="#" class="dropdown-item">User Profile</a>
                </li>
                <li>
                    <a href="#" class="dropdown-item">Logout</a>
                </li>
            </ul>
        </li>
        <li class="nav-item">
            <a href="#" class="nav-link"><i class="fas fa-cog"></i></a>
        </li>
    </div>
</nav>
```

ထူးခြားချက်အနေနဲ့ Menu ကို ညာဘက်ခြမ်းမှာ ကပ်ပေါ်စေချင်လို့ flex-fill Element အလွတ်တစ်ခု ထည့်ထားပေးတာကို သတိပြုပါ။ တစ်ခြားနည်းတွေ ရှိပေမယ့်၊ ဒီနည်းရှိကြောင်း သိစေချင်လို့ ထည့်ပေးထားတာပါ။ ဒါကြောင့် Dropdown အပါအဝင် Menu က အခုလို ညာဘက်ကပ် ဖော်ပြန်မှာပါ။



တစ်ဖြည်းဖြည်းနဲ့ ရပ်လုံးပေါ်လာပါပြီ။ Main ဒရိယာထဲမှာ လိုချင်တဲ့ Block လေးတွေကို Card Component တွေဆက်ပြီး ထည့်ကြပါမယ်။

#### Step-4 – Alert & Stat Blocks

လိုချင်တဲ့ လုပ်ဆောင်ချက်တွေ ထပ်ထည့်ဖို့အတွက် Main ဒရိယာထဲမှာ နောက်ထပ် Layout တစ်ခုထပ်လို ပါတယ်။ ဒါကြောင့် Container တစ်ခုကို Navbar အောက်မှာ အခုလိုကပ်ထည့်ပေးပါ။

#### HTML

```
<div class="container-fluid mt-3 p-4">
</div>
```

Navbar နဲ့ နည်းနည်းစွာပြစေချင်လို Margin Top ထည့်ထားပြီး အထဲက Element တွေကို ဘောင်ကနေ စွာပြစေချင်လို Padding လည်းထည့်ထားပါတယ်။ ဒဲဒဲ Container ထဲမှာ Alert Component တစ်ခုကို အခုလို ဆက်ထည့်လိုက်ပါ။

**HTML**

```
<div class="row mb-3">
    <div class="col">
        <div class="alert alert-info">
            <i class="fas fa-download me-2"></i> A new version of admin
            dashboard is released. <a href="#">Download Now!</a>
        </div>
    </div>
</div>
```

row တစ်ခုအတွင်းထဲမှာ အပြည့်နေရာယူထားတဲ့ col တစ်ခုနဲ့ ထည့်ပေးထားတာပါ။ ပြီးတဲ့အခါ သူ အောက်မှာ Stat Blocks လေးတွေကို နောက် row တစ်ခုနဲ့ အခုလို ဆက်ထည့်ပေးလိုက်ပါ။

**HTML**

```
<div class="row flex-column flex-lg-row">
    <h2 class="h6 text-white-50">QUICK STATS</h2>
    <div class="col">
        <div class="card mb-3">
            <div class="card-body">
                <h3 class="card-title h2">1,250</h3>
                <span class="text-success">
                    <i class="fas fa-chart-line"></i>
                    Daily visitors
                </span>
            </div>
        </div>
    </div>
    <div class="col">
        <div class="card mb-3">
            <div class="card-body">
                <h3 class="card-title h2">8,210</h3>
                <span class="text-success">
                    <i class="fas fa-chart-line"></i>
                    Weekly visitors
                </span>
            </div>
        </div>
    </div>
    <div class="col">
        <div class="card mb-3">
            <div class="card-body">
                <h3 class="card-title h2">12,560</h3>
                <span class="text-success">
                    <i class="fas fa-chart-line"></i>
                    Monthly visitors
                </span>
            </div>
        </div>
    </div>
    <div class="col">
        <div class="card mb-3">
            <div class="card-body">
                <h3 class="card-title h2">102,250</h3>
```

```

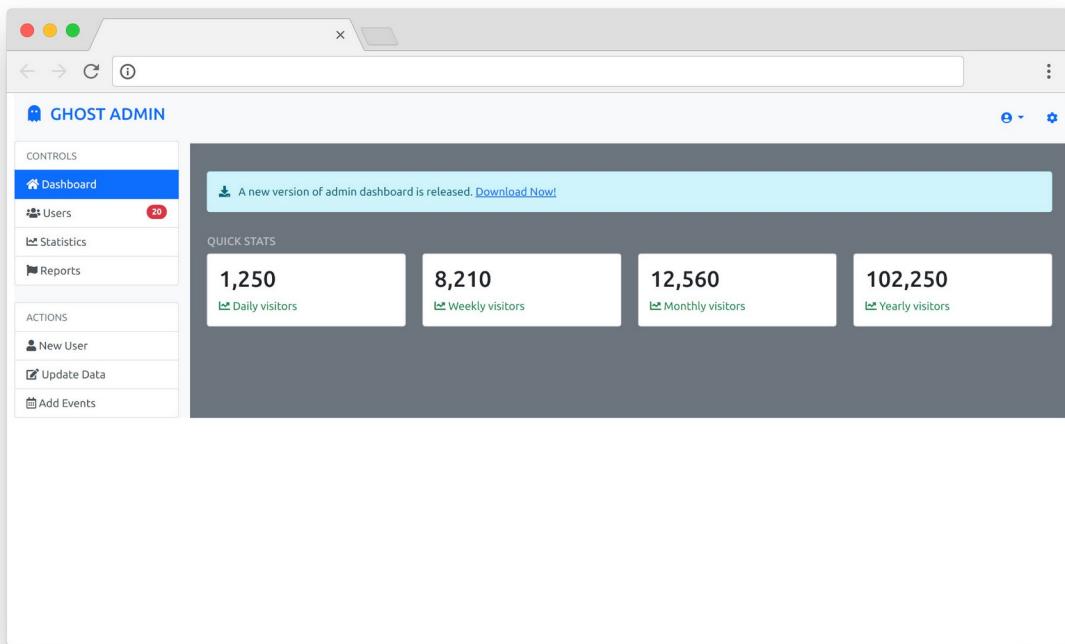


Yearly visitors


```

သူလည်းပဲ ပြချင်တာများလို ရေးရတာများပေမယ့် အထူးအဆန်းမပါပါဘူး။ row အတွင်းမှာ col (၄) ချို့  
ပြီး col တစ်ခုစီအတွင်းမှာ Card Component တွေထည့်ထားတာပါ။ ထူးခြားချက်ဆိုလို Screen သေး  
ရင် flex-column နဲ့ အပေါ်အောက်ဖြစ်ပြခိုင်းပြီး flex-lg-row နဲ့ Screen ကြီးတဲ့အခါ ဘေးတိုက်ဖြိုး  
ပြုံးပြခိုင်းထားပါတယ်။

ဒီအဆင့်မှာ ရလဒ်ကိုကြည့်လိုက်ရင် အခုလိုပုံစံဖြစ်နေပါပြီ။



တော်တော်လေး ပြည့်စုံနေပါပြီ။ ရေးနည်းအရ မခက်ပါဘူး။ ဘာလေးနဲ့ဘယ်လို ပေါင်းစပ် ဖန်တီးရင်  
ကောင်းမလဲဆိုတာကို မြင်တတ်ဖို့သာလိုတာပါ။ မြင်တတ်ဖို့ ဆိုတာကတော့ အခုလို နမူနာတွေအပါအဝင်  
လက်တွေ့တွေ များများလုပ်ပေးရင် တစ်ဖြည့်းဖြည့်း ရလာပါလိမ့်မယ်။

## Step-5 – Location & Data Blocks

နောက်ထပ် col (j) ခုပါတဲ့ row တစ်ခုထပ်ထည့်ပြီး လက်စသတ်လိုက်ကြပါမယ်။ ဒီလိုရေးထည့်ပေးပါ။

### HTML

```
<div class="row mt-4 flex-column flex-lg-row">

    <div class="col">
        <h2 class="h6 text-white-50">LOCATION</h2>

        <div class="card mb-3" style="height: 280px">
            <div class="card-body">
                <small class="text-muted">Regional</small>
                <div class="progress mb-4 mt-2" style="height: 5px">
                    <div class="progress-bar"
                        bg-success w-25"></div>
                </div>
                <small class="text-muted">Global</small>
                <div class="progress mb-4 mt-2" style="height: 5px">
                    <div class="progress-bar"
                        bg-primary w-75"></div>
                </div>
                <small class="text-muted">Local</small>
                <div class="progress mb-4 mt-2" style="height: 5px">
                    <div class="progress-bar"
                        bg-warning w-50"></div>
                </div>

                <small class="text-muted">Internal</small>
                <div class="progress mb-4 mt-2" style="height: 5px">
                    <div class="progress-bar bg-danger w-25"></div>
                </div>
            </div>
        </div>

        <div class="col">
            <h2 class="h6 text-white-50">DATA</h2>

            <div class="card mb-3" style="height: 280px">
                <div class="card-body">
                    <div class="text-end">
                        <button class="btn btn-sm
                            btn-outline-secondary">
                            <i class="fas fa-search"></i>
                        </button>
                        <button class="btn btn-sm
                            btn-outline-secondary">
                            <i class="fas fa-sort-amount-up"></i>
                        </button>
                        <button class="btn btn-sm
                            btn-outline-secondary">
                            <i class="fas fa-filter"></i>
                        </button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

```


| ID | Age Group | Data | Progress                          |
|----|-----------|------|-----------------------------------|
| 1  | 20-30     | 19%  | <i class="fas fa-chart-pie"></i>  |
| 2  | 30-40     | 40%  | <i class="fas fa-chart-bar"></i>  |
| 3  | 40-50     | 20%  | <i class="fas fa-chart-line"></i> |
| 4  | 50+       | 11%  | <i class="fas fa-chart-pie"></i>  |


```

သူမှာလည်း flex-column နဲ့ flex-lg-row ကိုတွဲပြီး Responsive လုပ်ဆောင်ချက် ထည့်ထားပါတယ်။ Component အနေနဲ့ ရွှေ့နမူနာမှာ မပြောခဲ့တာလေးတစ်ခု ပါနေပါတယ်။ Progress Component ပါ။ Progress ဆိတာ အလုပ်တစ်ခုခုလုပ်နေစဉ် ပြီးစီးမှုအခြေအနေကို ပြတဲ့ Component ဆိုတော့ JavaScript နဲ့တဲ့မသုံးရင် သိပ်အမိပို့ယ်မရှိလို ထည့်မပြောခဲ့တာပါ။ ဒီနေရာမှာတော့ အသွင်အပြင်ဖော်ပြုပုံ အရ သင့်တော်နေလို ထည့်သုံးထားပါတယ်။ ရေးနည်းကမခက်ပါဘူး။ ဒီလိုပါ -

**HTML**

```
<div class="progress">
  <div class="progress-bar bg-danger"></div>
</div>
```

ပင်မ Element ကို progress လိုပေးပြီး အထဲမှာ progress-bar ထည့်ပေးလိုက်ယုံပါဘဲ။ Bar ရဲ့ အရောင်ကိုသာ ကိုယ်လိုချင်တဲ့ bg-{color} နဲ့ တွဲသုံးရတာပါ။ Bar ရဲ့ အမြင့်နဲ့အရည်အတွက် width, height Property တွေကိုသုံးပြီး ကိုယ်လိုသလောက် ပေးထားလို့ရပါတယ်။

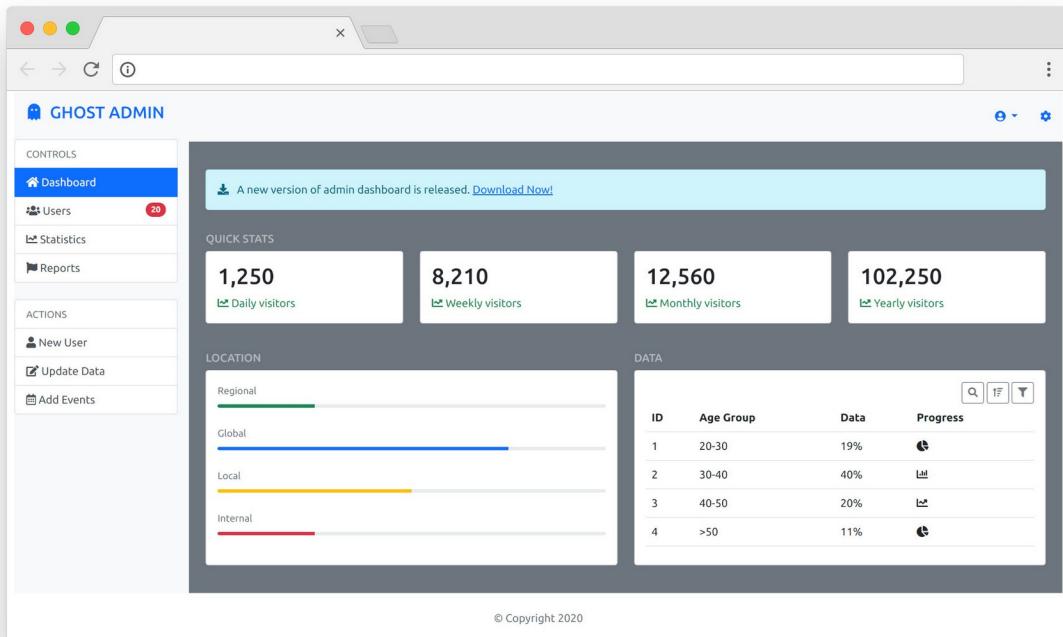
ဒီ Row ထဲမှာ Card နှစ်ခုပါပြီး တစ်ခုနဲ့တစ်ခု ညီစေဖို့အတွက် Inline Style ကိုသုံးပြီး height တွေ သတ်မှတ်ပေးတာကို သတိပြုပါ။ ပြီးတော့ Progress Bar တွေကို ပါးပါးလေးပဲ ပေါ်စေချင်လို့ height တွေသတ်မှတ်ပေးထားပါတယ်။ Width အတွက်တော့ w- Utility Class တွေကိုသုံးထားပါတယ်။

အားလုံးစုံသွားပါပြီ။ အောက်ဆုံးမှာ Footer လေးတစ်ခုအခုလိုထပ်ထည့်ပေးလိုက်ပါ။

**HTML**

```
<footer class="text-center py-4 text-muted">
  © Copyright 2020
</footer>
```

အခုလိုရင်ကျွန်တော်တို့ နမူနာလုပ်ကြည့်ချင်တဲ့ Admin Dashboard UI လေး ပြည့်စုံသွားပါပြီ။ အခုနေ့ စမ်းကြည့်ရင်ရမယ့် နောက်ဆုံးရလဒ်ကတော့ ဒီလိုဖြစ်မှာပါ။



ဟိုးအပေါ်မှာလည်း ရမယ့်ရလဒ်ကို ကြိုပြခဲ့ပြီးသားပါ။ Responsive လုပ်ဆောင်ချက် တစ်ခါတည်း ထည့်ထားလို့ Screen Size ကို ပြောင်းပြီးတော့လည်း စမ်းကြည့်နိုင်ပါတယ်။

ဒီနမူနာလေးကို လုပ်ကြည့်လိုက်တဲ့အတွက် Bootstrap ရဲ့လုပ်ဆောင်ချက်တွေကို နားလည်ယုံသာမက လက်တွေ့အသုံးချခဲ့ အလေ့အကျင့်လေး တစ်ခုကိုပါ ရရှိသွားလိမ့်မယ်လို့ မျှော်လင့်ပါတယ်။

ရေးသားထားပြီးကုဒ်နမူနာကို Download ရယူလိုက်တော့ ဒီလိပ်စာမှာ ရယူနိုင်ပါတယ်။

- <https://github.com/eimg/bootstrap-book>

## အခန်း (၁၀) – Customizing Bootstrap

လက်ရှိမှာ Component တွေ Layout တွေကို အသုံးပြုတဲ့အခါ Bootstrap ကပေးထားတဲ့အတိုင်း အသုံးပြုနေခြင်း ဖြစ်ပါတယ်။ အကယ်၍များ ဆန္ဒရှိတယ်ဆိုရင် Bootstrap ကို ကိုယ့်စိတ်တိုင်းကျလည်း Customize လုပ်ထားလို့ ရနိုင်ပါတယ်။ SASS နည်းပညာကို အသုံးပြုပြီး Customize လုပ်ရတာပါ။

ဒီနေရာမှာ SASS အကြောင်း အပါအဝင် Customize လုပ်ပုံ အသေးစိတ်ကို ထည့်သွင်းမဖော်ပြနိုင်ပေမယ့်၊ Bootstrap ကို Customize လုပ်ချင်ရင် ဘယ်လိုလုပ်ရသလဲဆိုတာ စာဖတ်သူတွေအိုင်ဒီယာရရေးအတွက် အကျဉ်းလောက်တော့ ထည့်သွင်းဖော်ပြချင်ပါတယ်။ ကြိုးစားပြီး လိုက်လုပ်ကြည့်ပါ။

### Step-1 – Install Node

ပထမဆုံးအနေနဲ့ Node ကို Install လုပ်ထားဖို့လိုပါလိမ့်မယ်။ ပရောဂျက်တည်ဆောက်တာတွေ SASS ကုဒ်တွေကို CSS ကုဒ်ဖြစ်အောင် Compile လုပ်တာတွေကို Node ကနေတစ်ဆင့် လုပ်ရမှာမို့လိုပါ။ ဒီလိပ်စာမှာ Download လုပ်လို့ရနိုင်ပါတယ်။

- <https://nodejs.org>

Download လုပ်ပြီးရင် Install လုပ်လိုက်ပါ။ Install လုပ်ပြီးသွားတဲ့အခါ Node နဲ့အတူ NPM လိုခေါ်တဲ့ နည်းပညာ တစ်ခါတည်း ပါဝင်သွားပါလိမ့်မယ်။ NPM အကြောင်း အကျဉ်းချုပ်ကို ရွှေ့ပိုင်းမှာပြောထားပြီး သွားပါ။

## Step-2 – Create Project Folder

ပရောဂျက်ဖိုဒ်အတစ်ခုကို မိမိနှစ်သက်ရာအမည်နဲ့ဆောက်လိုက်ပါ။ ဥပမာ - theme ဆိုကြပါစို့။ ပြီးတဲ့အခါ အထဲမှာ scss နဲ့ css ဆိုတဲ့ ဖိုဒ်နှစ်ခုတည်ဆောက်လိုက်ပါ။ index.html ဖိုင်ကို အပြင်မှာ တည်ဆောက်ပြီး custom.scss အမည်နဲ့ဖိုင်တစ်ခုကို scss ဖိုဒ်ထဲမှာ တည်ဆောက်လိုက်ပါ။ ဖွဲ့စည်းပုံက ဒီလိုပါ။

```
theme/
| - css/
| - scss/
|   | - custom.scss
| - index.html
```

## Step-3 – Install Bootstrap & SASS

တည်ဆောက်ထားတဲ့ ပရောဂျက်ဖိုဒ်ထဲမှာ Command Prompt (သို့) Terminal ကိုဖွင့်လိုက်ပါ။ ပြီးရင် ဒီ Command ကို Run ပေးပါ။

```
npm init -y
```

ဒါဟာ လက်ရှိပရောဂျက်ဖိုဒ်ကို NPM ပရောဂျက်ဖြစ်အောင် ပြောင်းလိုက်တာပါ။ ပြီးတဲ့အခါ ဒီ Command နဲ့ Bootstrap ရော SASS ကိုပါ Install လုပ်လိုက်ပါ။

```
npm install bootstrap sass
```

အခုချိန်မှာ ပရောဂျက်ဖိုဒ်အဲ ဖွဲ့စည်းပုံက ဒီလိုဖြစ်နေပါလိမ့်မယ်။

```
theme/
| - node_modules/
|   | - bootstrap/
|   | - sass/
| - css/
| - scss/
|   | - custom.scss
| - index.html
| - package.json
```

package.json ဆိုတဲ့ ဖိုင်ဟာ npm init ကို Run လိုက်လို ပါဝင်သွားတဲ့ဖိုင်ဖြစ်ပြီး npm install နဲ့ Install လုပ်လိုက်တဲ့ Bootstrap နဲ့ SASS တို့ကတော့ node\_modules ဆိုတဲ့ဖိုဒါထဲကို ရောက်ရှိသွားခြင်းဖြစ်ပါတယ်။ ဒါဆိုရင် Bootstrap ကို စတင် Customize လုပ်ဖို့အတွက် အသင့်ဖြစ်ပါပြီ။

#### Step-4 – Writing Customization Code

scss/custom.scss ဖိုင်ကိုဖွင့်ပြီး အခုလိုရေးပေးလိုက်ပါ။

##### SCSS

```
$enable-rounded: false;
$primary: #6610f2;
$secondary: #d63384;
$success: #20c997;
$warning: #fd7e14;

@import "../node_modules/bootstrap/scss/bootstrap";

.note {
    margin: $spacer 0;
    padding: $spacer;
    border: 1px solid $warning;
    border-left: 5px solid $warning;
}
```

Bootstrap က အလွယ်တစ်ကူပြင်လိုရအောင် ကြိုရေးပေးထားတဲ့ Variable တွေရှိပါတယ်။ အဲဒီ Variable တွေထဲက \$enable-rounded ဆိုတဲ့ Variable တန်ဖိုးကို false လိုပေးလိုက်တဲ့အတွက်၊ Button တွေ List Group တွေ Card တွေမှာ ထောင့်ကွွေးလေးတွေ မပါတော့ဘဲ ရှိုးရိုးလေးထောင့်နဲ့ပဲ ဖော်ပြပေးတော့မှာပါ။

ပြီးတဲ့အခါ Color Variable တွေကိုလည်း ကိုယ်ကြိုက်တဲ့အရောင်နဲ့ ပြောင်းပေးထားပါတယ်။ \$primary ဟာ မူလက အပြာရောင်ပါ။ အခုတော့ ခရမ်းရောင်နဲ့ ပြောင်းပေးထားပါတယ်။ \$secondary ဟာ မူလက မိုန်တဲ့အရောင်ပါ။ အခုတော့ ပန်းရောင်နဲ့ ပြောင်းပေးထားပါတယ်။ \$success ကိုလည်း မတူကွဲပြားတဲ့ အစိမ်းရောင်တစ်မျိုးနဲ့ ပြောင်းထားပြီး \$warning ကို အဝါအစား လိမ့်မှောင်နဲ့ အစားထိုးထားပါတယ်။ Color ကုဒ်တွေနေရာမှာ ကိုယ်ကြိုက်တဲ့အရောင်ကို ကိုယ်ကြိုက်တဲ့ Format နဲ့ပေးနိုင်ပါတယ်။

ပြီးတဲ့အခါ @import နဲ့ ပင်မ Bootstrap SCSS Source Code ကို ချိတ်ယူထားပါတယ်။ ဒီတော့မှ ဒီဖိုင် ကို CSS ပြောင်းလိုက်ရင် မူလ Bootstrap SCSS ကုဒ်တွေကို ကိုယ်ဖြည့်ရေးပေးထားတဲ့ ကုဒ်တွေနဲ့ ပေါင်းစပ်ပေးသွားမှာပါ။ Bootstrap Variable တန်ဖိုးတွေကို ပြောင်းချင်ရင် Import မလုပ်ခင် ရေးပေးရပါ တယ်။ Import လုပ်ပြီးမှ ဆက်ရေးထားတဲ့ ကုဒ်တွေကတော့ Bootstrap ကို ပြင်တာ မဟုတ်တော့ဘဲ ကို ယုံဘာသာ ထပ်ဖြည့်ထားတဲ့ လုပ်ဆောင်ချက်တွေပါ။

နမူနာမှာ note Class အတွက် Style တစ်ချို့ရေးထားပေးပါတယ်။ \$spacer ဆိုတာ Bootstrap Variable ပါပဲ။ Default Value 1rem လို သတ်မှတ်ထားပါတယ်။ ဆန္ဒရှိရင် ပြင်လိုရပါတယ်။ အခုကာ ပြင် ထားတာ မဟုတ်ပါဘူး။ ယူသုံးထားတာပါ။ note Component အတွက် margin နဲ့ padding ကို \$spacer ရဲ့တန်ဖိုးအတိုင်းပဲ ယူလိုက်တာပါ။ ဒါကြောင့်နောက်ပိုင်း \$spacer တန်ဖိုးပြောင်းလိုက်ရင် ဒါ တန်ဖိုးတွေလည်း လိုက်ပြောင်းသွားမှာပါ။

တော်ပါပြီ။ နမူနာအနေနဲ့ ဒီလောက်ပဲ စမ်းမှာပါ။ တစ်ကယ်တမ်း လုပ်မယ်ဆိုရင်တော့ လုပ်လိုရတာတွေ အများကြီးပါ။ နောက် အသင့်ဖြစ်ပြီဆိုတော့မှ Bootstrap Documentation မှာ ဆက်လေ့လာလိုက်ပါ။

## Step-5 – Compiling and Using

ရေးထားတဲ့ Custom SCSS ကုဒ်တွေကို CSS ပြောင်းကြပါမယ်။ ဒါကြောင့် ပရောဂျက်ဖိုဒ်မှာ ဒီ Command ကို Run ပေးပါ။

```
npx sass scss/custom.scss css/custom.css
```

sass ကိုအသုံးပြုပြီး scss ဖိုဒ်မှာရှိတဲ့ custom.scss ဖိုင်ကို css ဖိုဒ်မှာ custom.css ဆို တဲ့အမည်နဲ့ Compile လုပ်ပေးသွားမှာဖြစ်ပါတယ်။ ပြီးသွားရင် custom.css ဖိုင်ကိုဖွင့်ကြည့်နိုင်ပါ တယ်။ အထဲမှာ Bootstrap CSS ကုဒ်တွေပါဝင်တာကို တွေ့ရပါလိမ့်မယ်။ ထူးခြားသွားတာက ကျွန်ုတ် တို့ ပြုပြင်ဖြည့်စွက်ပြီး ရေးပေးလိုက်တဲ့ ကုဒ်တွေပါ ရောပါသွားတာမို့လို့ အခုနေ အသုံးပြုရင် ပြုပြင် ဖြည့်စွက် ထားတဲ့အတိုင်း ရရှိမှာဖြစ်ပါတယ်။

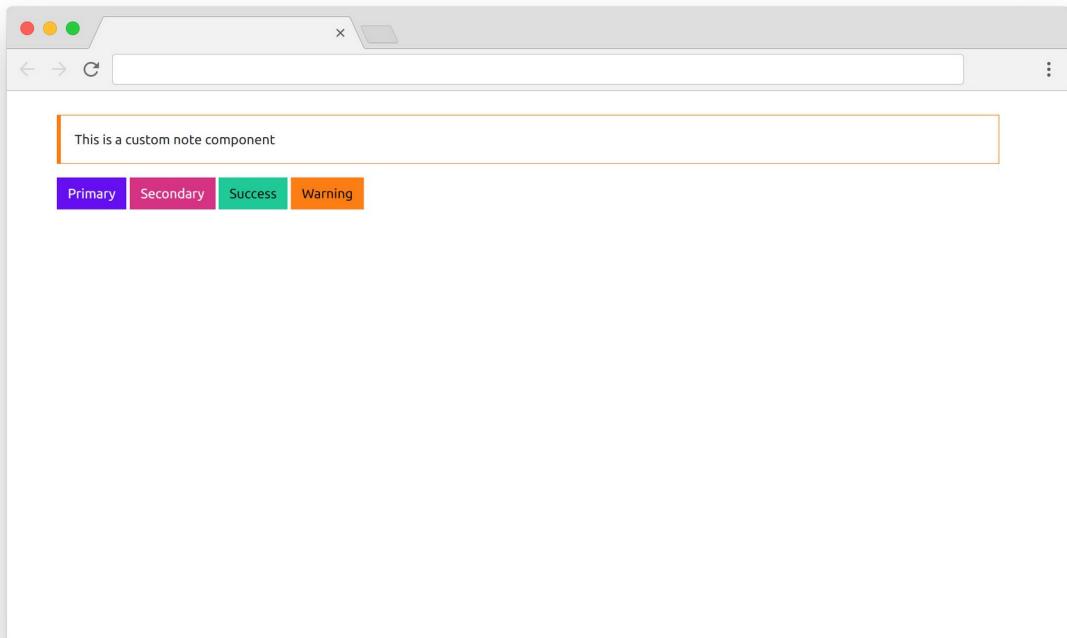
စမ်းသပ်နိုင်ဖိုအတွက် index.html ထဲမှာ အခုလိုရေးပြီး စမ်းကြည့်နိုင်ပါတယ်။

**HTML**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, initial-scale=1.0">
    <title>Theme</title>
    <link rel="stylesheet" href="css/custom.css">
</head>
<body>
    <div class="container">
        <div class="note">This is a custom note component</div>

        <button class="btn btn-primary">Primary</button>
        <button class="btn btn-secondary">Secondary</button>
        <button class="btn btn-success">Success</button>
        <button class="btn btn-warning">Warning</button>
    </div>
</body>
</html>
```

<link> နဲ့ချိတ်ထားတာ Bootstrap မဟုတ်တော့ပါဘူး။ css ထဲက custom.css ဖြစ်သွားပါပြီ။ စင်းကြည့်လိုက်ရင် ရလဒ်က အခုလိုဖြစ်မှာပါ။



ဖြည့်စွက်ထည့်သွင်းထားတဲ့ note Component အလုပ်လုပ်နေတာကိုတွေ့ရမှာ ဖြစ်သလို primary, secondary, success, warning စတဲ့ Color တွေကလည်း ပြုပြင်ပေးလိုက်တဲ့အတိုင်း ဖော်ပြနေတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ Button တွေမှာ Rounded Corner လေးတွေမပါတော့ဘဲ လုံးဝ လေးထောင့် ဖြစ်နေတာကိုလည်း သတိပြုပါ။

ဒီနည်းနဲ့ Bootstrap ကို Customize လုပ်ထားခြင်းအားဖြင့် အများနဲ့မတူ ကဲပြားတဲ့ အသွင်အပြင်တွေကို ရရှိနိုင်ပါတယ်။ ဒီနည်းကိုသုံးပြီး ကြိုရေးပေးထားတဲ့ Bootstrap Themes တွေလည်းအများအပြား ရှိနေပါတယ်။ တစ်ချို့က အခပေး Themes တွေဖြစ်ပြီး တစ်ချို့ကတော့ အခမဲ့ရတဲ့ Themes တွေပါ။ ရှာဖွေလေးလာပြီး အသုံးပြုကြည့်သင့်ပါတယ်။

ဒီစာအုပ်မှာဖော်ပြချင်တဲ့ အကြောင်းအရာတွေကတော့ စုံသွားပြီဖြစ်လို့ ဒီမှာတင် နိဂုံးချုပ်လိုက်ပါတယ်။ လေးလာသူများအတွက် အသုံးဝင်ပြီး အထောက်အကူ ဖြစ်စေလိမ့်မယ်လို့ မျှော်လင့်ပါတယ်။

## နိဂုံးချုပ်

ဒီစာအုပ်ဟာ ကမ္ဘာတစ်ရွမ်း COVID-19 ကပ်ရောဂါတွေ ဖြစ်ပွားနေချိန်၊ အိမ်တွင်းအောင်းပြီးရေးဖြစ်ခဲ့တဲ့ လိုတိရှင်း စာအုပ်တွေထဲမှာ တစ်အုပ်အပါအဝင် ဖြစ်ပါတယ်။ ဒီစာအုပ်ရဲ့ ထူးခြားချက်ကတော့ အခြေခံ အဆင့်ကနေ စတင်ဖော်ပြထားခြင်းဖြစ်လို့ Web Development ကို အခုမှစတင်လေ့လာမယ့်သူများဟာ ဒီ **Bootstrap လို - တိ - ရှင်း** စာအုပ်ကနေ စတင်လေ့လာဖို့ အထူးသင့်တော်ပါတယ်။ ရေးသားထားပြီးဖြစ် တဲ့ **React လို - တိ - ရှင်း** | **Laravel လို - တိ - ရှင်း** နဲ့ **API လို - တိ - ရှင်း** အမည်ရ စာအုပ်တွေလည်း ရှိပါ သေးတယ်။ အခြေခံအသင့်အတင့် ရှိပြီးသူတွေ လေ့လာနိုင်ပါတယ်။ အခုမှစလေ့လာသူ့အတွက်တော့ ကြား ထဲမှာ ခါးဆက်နည်းနည်း ပြတ်နေနိုင်ပါတယ်။ ဒီလိုအပ်ချက်ကို ဖြည့်စည်းဖို့အတွက် **JavaScript လို - တိ - ရှင်း** စာအုပ်နဲ့ **PHP လို - တိ - ရှင်း** စာအုပ်များကိုလည်း ဆက်လက်ရေးသားဖို့ အစီအစဉ်ရှိပါတယ်။ လက်ရှိ ရေးသားထားပြီးဖြစ်တဲ့ စာအုပ်တွေအကြောင်းကို သိချင်ရင် ကျွန်ုတ်တော်စာရေးသူရဲ့ ဝဘ်ဆိုက်မှာ ဝင်ရောက်လေ့လာနိုင်ပါတယ်။

- <https://eimaung.com>

ဒီစာအုပ်ကို အစက Bootstrap Components တွေနဲ့ Layouts သဘောသဘာဝ အကြောင်းသက်သက်ပဲ ရေးသားပြီး အခမဲ့ ဖြန့်ဝေပေးဖို့ စီစဉ်ခဲ့တာပါ။ ဒါပေမယ့် လက်တွေမှာ HTML/CSS အခြေခံကနေစပြီး တော့ ပြည့်ပြည့်စုစုဖြစ်သွားတဲ့အတွက် မဖြန့်ဝေနိုင်ခဲ့ပါဘူး။ အခမဲ့ ဖြန့်ဝေပေးမယ်လို့ ကြို့ပြော ထားပြီးမှ မပေးနိုင်ခဲ့ပေမယ့်လည်း စာဖတ်သူများကတိုင်းဝန်းအားပေးကြတဲ့အတွက် ကျေးဇူးအထူးတင်ပါတယ်။ စာဖတ်သူများအားလုံး ကပ်ဘေးတွေကို ကျော်လွှားနိုင်ပြီး ကိုယ်စိတ်နှစ်ဖြာ ကျွန်ုတ်မာချမ်းသာ ကြပါစေလို့ ဆုတောင်းလိုက်ပါတယ်။

## အိမောင် (Fairway)

၂၀၂၀ ပြည့်နှစ်၊ အောက်တိုဘာ (၂၈) ရက်နေ့တွင် ရေးသားပြီးစီးသည်။

၂၀၂၁ ခုနှစ်၊ ဒီဇင်ဘာလ (၂) ရက်နေ့တွင် Bootstrap 5 သို့ အဆင့်မြှင့်ပြင်ဆင် ပြီးစီးသည်။